October 1985

Report No. STAN-CS-85-1072

A Programming and Problem-Solving Seminar

by

Ernst W. Mayr

Richard J. Anderson

Peter H. Hochschild

Department of Computer Science

Stanford University
Stanford, CA 94305

A Programming and Problem-Solving Seminar

by

Ernst W. Mayr
Richard J. Anderson
Peter H. Hochschild

This report contains edited transcipts of the discussions held in Stanford's course CS204, Problem Seminar, during winter quarter 1984. The course topics consisted of five problems coming from different areas of computer science. The problems were discussed in class and solved and programmed by the students working in teams.

The production of this report was partially supported by NSF grants MCS-82-03405 and DCR-8351757 and by an IBM Faculty Development Award.

TABLE OF CONTENTS

1. Introduction

The course CS204 is a problem solving class offered to first year Phd. students at Stanford University. The class consists of five problems drawn from various areas of computer science. The problems are discussed in class and solved and programmed by the students working in teams. The purpose of the class is to expose the students to the major paradigms of computer science research.

In this report, the following abbreviations instead of full names will be used to identify the participants.

EM	Ernst Mayr, the Instructor
RA	Richard Anderson, Teaching Assistant
PPH	Peter Hochschild, Teaching Assistant
AAS	Alejandro Schaffer
ANS	Arun Swamy
AS	Ashok Subramanian
CWC	Clyde Carpenter
GP	George Papagcorgiou
HD	Helen Davis
JP	Joseph Pallas
MA	Mar tin Abadi
MGB	Miriam Blatt
RC	Ross Casley
RWH	Ramsey Haddad
SR	Shaibal Roy
SS	Sriram Sankar
ST	Steve Tjiang

2. Integer Bricks

2.1 **Problem Statement**

We want to investigate whether there is an *integer brick, i.e.*, a *cuboid* whose three sides, three face diagonals, and body diagonal are all positive integers.

Let the three sides of the brick be a, 6, and c. To make the problem computationally more feasible, we shall restrict ourselves to $b, c < 10^6$ and a in some reasonable interval below **10".** The exact meaning of "reasonable" will depend on the details of your algorithm. One goal is to make this interval as large as possible.

Hint: Let x, y, z be the integer sides of a right-angled triangle, i.e.,

$$x^2 + y^2 = z^2.$$

Then there arc two relatively prime integers r and s, r > s, such that

$$x : y : z = 2rs : r^2 - s^2 : r^2 + s^2.$$

While wc are searching for integer bricks, we also want a list of all "almost" solutions *(a, 6,* c) in the above range. Hcrc, an "almost" solution is a brick with all three sides and all three face diagonals integer, but with the body diagonal irrational.

2.2 Notes for Thursday, January 12

Today's discussion focussed primarily on the subproblem of generating Pythagorean triples, that is, right-angled triangles with integer sides.

Let, for the moment, *a,* 6, and *c* be the integer sides of a right-angled triangle, with *c* the hypotenuse, and *a* and 6 the two legs. In the interest of efficiently generating such triples, one may very well assume that the *greatest common divisor* of all three of the numbers is one, because otherwise one could factor it out to obtain a triple with smaller elements. (Of course, if (II, 6, c) is a Pythagorean triple, then so is (mu, mb, mc), for any positive integer m.)

AS remarked that it is desirable to gcncratc the triples without mistakes, i.e., to have a method to generate all and only Pythagorean triples. SR said that in addition to gcd $(a, b, c) = 1$ one could also require gcd(a, 6) = gcd(a, c) = gcd(b, c) = 1. This follows easily from the observation that, if say gcd(a, 6) = d > 1 then d^2 tlividcs the lefthand side

of $a^2 + b^2 = c^2$, and therefore also the righthand side. But from the unique factorization theorem, one then obtains that d must also divide c. That contradicts the assumption that $\gcd(a, b, c) = 1$. The argument for the other two pairs runs in an analogous way. One may therefore look only for *reduccd* Pythagorcan triples, that is Pythagorcan triples in which every pair of the sides are relatively prime. One may also, without loss of generality, take into account the symmetry of a and 6 by considering only reduced triples with $6 < a < c$.

EM asked whcther more restrictions, for example parity restrictions, can bc placed on a, 6, c. Obviously, not all threc clements can be even bccause the triples are rcduced, and for the same reason no two of thcm can bc even. On the other hand EM suggested, assume that all threc numbers are odd. Then, for any integer m, *we* have $m^2 \equiv 0 \pmod 4$ or $m^2 \equiv \mathbf{1} \pmod 4$. If m is odd, it is exactly the second possibility that applies. IIence, if one considers $a^2 + b^2 = c^2$ with all numbers taken modulo 4, one obtains 2 for the lcfthand side, and 1 for the righthand side, a contradiction. Similarly, one can excludc the case that a and 6 are both odd and c is even. Summarizing, in a reduced Pythagorcan triple $(a$, 6, c), exactly one of a and 6 must be **evcn** while thc other and c must be odd.

The class made the following agreement.

Definition 2.1: Let (a, 6, c) be a Pythagorean triple, not necessarily reduced, and let (a', b', c') be the corresponding reduccd triple. Let a (respectively, 6) be called the ***evcn lcg*** and 6 (respectively, a) the ***odd*** leg of the triple if a'(respectively, 6') is **evcn.**

AS remarked that thc hint givcn in the problem statement givcs a mcthod to gcncrate reduced Pythagorean triples. The class dccidcd to derive the formulas givcn in that hint.

Assume that $a^2 + b^2 = c^2$ with (a, 6, c) reduced and a even. Then $b^2 = (c - a)(c + a)$, and it is easy to scc that thc two factors on thc right must bc relatively prime. But since thc lcfthand side is a pcrfcct squarc, we dcducc from thc uniquc factorization theorem, that both (c − a) and $(c + a)$ must bc pcrfcct squares, say $c - a = x^2$ and $c + a = y^2$. Bccausc of thc parity of c and a, x and y must both bc odd. IIence, $r = \frac{y+x}{2}$ is an integer, and so is $s = \frac{y-x}{2}$. It is straightforward now to verify that with thc quantitics so dcfined,

$$a = 2rs;$$
$$b = r^2 - s^2;$$
$$c = r^2 + s^2.$$

Wc also know that r and s must bc relatively primc bccausc othcrwisc $(a$, 6, $c)$ would not bc rcduced. And what is morc, since

$$c - a = (r - s)^2$$

is odd, exactly onc of r, s must be **cven.**

AS thcn rcmarkcd that so far thc class had only donc thc casc whcrc a is cvcn. EM respondcd that the argument for odd valucs of a basically worked thc samc way, and left it to the class to fill in the details. He gave the final results for this computation:

$$a = rs;$$
$$b = \frac{1}{2}(r^2 - s^2);$$
$$c = \frac{1}{2}(r^2 + s^2).$$

Again, r and s are relatively prime, but in this case they must both be odd.

EM also observed that interchanging a and b in a Pythagorean triple can easily be seen to correspond to replacing r and s by $r + s$ and $r - s$ (assuming without loss of generality that $r > s$).

SR asked whether a (reduced) triple (a, b, c) uniquely determines (r, s) (assuming again that $r > s$). EM said that it does, and that this can be seen from the fact that the equation for r,

$$r^2 = b + \frac{a^2}{4r^2},$$

which we obtain from the equations above by eliminating s, only has one positive solution for r^2.

The discussion then turned to getting a crude estimate for the number of (r, s) pairs involved. The following restrictions were exhibited:

- Since $a, b < 10^6$ we get $c < \sqrt{2} \times 10^6$. Therefore, for the case that $a = 2rs$ is the even leg, $1 \le s < r < 1190$, and exactly one of r and s is even. For the case that $a = rs$ is the odd leg, $1 \le s < r < 1682$, and r and s are both odd.
- Since $a = 2rs$ if a is the even leg we get $s < \frac{500{,}000}{r}$. Otherwise $a = rs$ and thus $s < \frac{10^6}{r}$.
- Since $a > b$ we obtain (in both cases) $s > (\sqrt{2} - 1)r$.

A crude estimate yields about 300,000 (r, s)-pairs satisfying the above constraints.

2.3 Notes for Tuesday, January 17

EM began with a brief review of the facts established at the previous meeting. These facts concern the parametric generation of integer sided triangles. Discussion then turned to the subject of implicit constraints on the side lengths of integer bricks.

AS presented the following observation. Let A, B, and C be the dimensions of a brick. Then at least two of A, B and C must be congruent to zero modulo four, and one must be congruent to zero modulo sixteen. This is established by first considering a single face with legs a and b, and hypotenuse c. Assume that (a, b, c) is reduced, and that four divides neither a nor b. Then, recalling that exactly one of a and b must be even, $a^2 + b^2$ is congruent to 5 or 13 modulo 16. Unfortunately (maybe fortunately), none of these values is a quadratic residue modulo 16 and thus this contradicts that $a^2 + b^2 = c^2$. Therefore one leg of each face is divisible by four. This in turn implies that at least two legs of any brick are divisible by four. To see that one leg is divisible by sixteen, consider the face formed by the two legs that are divisible by four. Shrink both legs by a factor of four and apply the above fact about faces to reveal that one leg must still be divisible by four.

EM noted that this fact considerably reduces the size of the space that needs to be searched. He also noted that one may assume that the sides of the brick are relatively prime. JP pointed out that that implies that exactly two sides of the brick are divisible by four.

AS announced another similar constraint, namely, that exactly two sides must be divisible by 3 and one by 9.

EM responded by claiming a similar result for 5 (though possibly less useful for our purposes since it could he the hypotenuse whose length is divisible by 5) and asked whether this could be generalized. No generalization came forth and discussion shifted to the question of whether the modulo 3, 4 and 5 constraints could be nicely combined. EM suggested that it might bc reasonable to employ merely the modulo 4 constraint; attempting more might well result in an unpleasant tangle of rules.

ANS wondered whether the divisible-by-four rule would be used to cut down the number of admissible *r, s* pairs. AS replied that it would provided that the program generated the divisible-by-four sides first. At this point an impenetrable controversy arose.

EM restored order by pointing out that there arc at least two ways of structuring the brick hunt. The first method is to use an outer loop to generate all admissible *B*, C pairs. Then for each such pair, an inner loop could hunt for suitable values of *A*.

Another approach would be to employ an outer loop for enumerating candidate values of A and to use an inner loop to find compatible *B*, C values.

EM added that it was not clear in advance which general outline would be preferable.

JP suggested that a table of potential *r*, s (or *B*,*C*) pairs might be useful. Naturally EM wanted to know how big the table would be. ANS replied that the *r*, *s* table contains about 100,000 entries. The "two sides divisible by four" business came up again; AS argued that the constraint doesn't reduce the size of the table. EM stated that it, restricts the m's but not *r* or s. CW responded that it does force *r* and s to remain smaller.

EM noted that there was an asymmetry in the statement of the problem concerning the range of A versus the range of *B* and C. If one decided to allow A the same range as *B* and C one might force, for example, A to be divisible by 4 and *B* to be divisible by 16. However, if A is confined to a smaller range (in order to keep the computation Within reason) such an approach is faulty. Thus he feels it would bc preferable to use an unsymmetric algorithm (the alternative being to have A, *B* and *C* share a common but smaller range).

ANS suggested that the second general framework for the program would be most appropriate (with some kind of case statement in the inner loop conditioned on *A*). EM preached the value of keeping an open mind, while AAS pointed out that it would be more efficient to have several sets of inner loops corresponding to the different cases of *A*.

AS mentioned that no use had yet been made of the fact that we are interested in bricks with integer length body diagonals. He added that the requirement of reporting all almost bricks would preclude taking advantage of the body diagonal constraint.

EM provided a brief historical perspective on the problem. He noted that the subject of Diophantine equations is classic (dating as it does, back to Diophant). Hilbert's Tenth problem was the question of decidability of Diophantine equations; this question was relatively recently answered in the negative. Thus there is no algorithm capable of solving general Diophantine equations. However, linear Diophantine equations are solvable, albeit slowly. (The problem is NP-complete.) Euler is responsible for the parametrization of integer length triangles. Some not very thorough tables of almost bricks have been produced; no examples of bricks have yet been discovered. EM added that he had heard that a search over the range $0 < s < r < 30$ has been conducted (without success).

MGB suggested starting our search with $r > s > 30$ but EM pointed out that it wouldn't do to force both r and s to exceed 30.

AAS wondered whether it would help to make use of the body diagonal constraint. EM doubted that it would help much, but said it would be worthwhile if it significantly increased the range of A that could be searched.

The next topic of discussion was generation of r, s pairs. AAS said that testing the greatest common divisor of every pair of integers in the suitable range was too inefficient. It is better to use an outer loop to run through values of s, factor these values and then, in an inner loop, test candidates for r to see whether they are divisible by any factor of s. EM suggested a refinement of this idea based on sieves. The idea is to use a bit vector corresponding to possible values of r (something like $s + 1 \leq r \leq 1189$) and to sift this vector with all multiples of the prime factors of s. AAS pointed out that some bits of the vector would thereby be sifted out several times; RWH replied that the number of such repetitions would be limited to the number of prime factors of s, which is always fairly small. MA noted that one might save work by using information gained from previous values of s. For example the r vector corresponding to $s = 10$ can be derived from the r vector corresponding to $s = 5$ merely by sifting out all multiples of two.

EM asked whether it would be possible to store all r, s pairs. AAS volunteered that there were about **336,000** pairs in the range (his program took **10** seconds to count them). EM claimed that storing 336,000 pairs of integers would take up too much space. ANS suggested storing at any given time only the r values for a single value of s. EM suggested a bit vector; a rough estimate of the required size came out to $35k$-words. AAS argued that even this modest sized table might be plagued by page faults. EM postponed discussion of page faults until it is known how the table will be used. He suggested that everyone try to generate the r, s table. He also urged thought on the question of how to bring the third leg into play. ANS wanted to know if there were likely to be more constraints. EM didn't think so unless anyone could come up with a proof that no bricks exist!

2.4 Notes for Thursday, January 19

EM briefly described the constraints that were presented or alluded to in the last meeting. These include the facts that:

(I) Two legs of every brick must be divisible by four and one of them must bc divisible by sixteen.

(II) Two legs of every brick must be divisible by three and one of them must be divisible by nine.

EM noted that some information can bc obtained by analyzing the equation $a^2 + b^2 = c^2$ modulo 5 (the quadratic residues modulo 5 are **1** and 4). The resulting constraint is not too helpful however. One possibility is that only the hypotenuse is divisible by five. In this case one leg must be congruent to ± 1 modulo 5 while the other must be congruent to ± 2 modulo 5. This kind of constraint promises to be much more unwieldy than the modulo 3 and 4 constraints.

RC had investigated the situation modulo higher primes; he reported that the results seemed quite unattractive. EM agreed and remarked that the additional programming complexity would likely outweigh the potential benefits.

RWH connected the leg constraints with constraints on r, s values by claiming that if 16 divides A then one member of the corresponding r, s pair (the even one) must be divisible by 8. SS suggested that rather than imposing the condition r > s, one might insist that r bc even. EM noted that this would not reduce the space of r, s values.

SR discussed the effect of directly applying the modulo 3 and 4 constraints to the brick legs. He noted that there are six cases of which the following is a typical example:

(i) Modulo 4 constraint:

$$A \not\equiv 0 \bmod 2 \qquad B \equiv 0 \bmod 4 \qquad C \equiv 0 \bmod 16$$

(ii) Modulo 3 constraint:

$$A \not\equiv 0 \bmod 3 \qquad B \equiv 0 \bmod 3 \qquad C \equiv 0 \bmod 9$$

The implications of these simultaneous constraints reduce the space of candidates for A, B and C. For example, in this case C must be divisible by 144, B by 12, and A by neither 2 nor 3.

Some controversy erupted at this point. Eventually a consensus was reached that these constraints alone do not bring the size of the search space within acceptable limits.

CWC asked a question about modulo 9 constraints. EM didn't see any generally applicable rule coming from 9; though he did mention that there might exist some kind of modulo 7 rule. This topic was dropped in favor of the matter of generating r, s pairs.

SS triggered a discussion of merits of generating only the pairs with odd members. EM suggested that this would trade storage for computation; HD noted that it had already been established that it would bc possible to store (as a giant bit vector) the r, s table in the computer; SR topped them both by drawing attention to the possibility that forcing both r and s to have odd parity would increase the range of r, s values that had to bc considered. It was eventually decided that the r, s range would have to bc expanded by a factor of 2 or $\sqrt{2}$ or something like that.

SS was then permitted to continue explaining his plan for generating r, s pairs. He apparently had in mind to use the r, s pairs in the sequence defined by increasing value of the product of r and s. In order to conserve storage, he proposed a kind of pipeline. The r, s pairs are computed in lexicographic order and fed into the pipe. The pipe transmits these pairs to a data structure wherein they are temporarily stored. This data structure is composed of a set of lists. Each list contains those r, s pairs with a given product. As soon as each list is completed, it is shoved into an output pipe (in order of increasing product value). As these lists emerge from the output pipe they are consumed by some process he did not describe, and are then discarded.

Because this scheme demands some form of dynamic storage allocation, a debate about the capabilities of various languages ensued. Some argued that most PASCAL implementations fail to support storage reclamation. Others, including EM, maintained that the PASCAL on local machines does correctly implement dynamic storage allocation. EM also noted that one could provide one's own dynamic storage facility (using one's own free storage list).

EM then turned to the issue of the general structure of suitable algorithms. At the last meeting hc noted two possible outlines. He proposed spending the rest of the period looking at his second outline. In this outlint:, an outer loop generates candidate values for A, while an inner loop looks for compatible B, C values. EM pointed out that there would be various cases of the inner loop, depending, for example, on whether A was divisible by four. He suggested postponing discussion of the details of the different cases, and the order in which the A's would be examined. Thus he asked what the program should do to investigate a given value of A.

SR asked whether there was any reason to believe that bricks with widely different side lengths exist. EM replied that, at any rate, there are almost bricks whose three sides differ in length by several orders of magnitude.

After some desultory conversation that will not be reported, talk turned back to the question of what to do when presented with a particular value of A.

People generally agreed that the first step would be to find the prime factorization of A. EM suggested that it would be helpful to find all factorizations of A *of* the form $A = mrs$ with $r > s$ and $\gcd(r, s) = 1$. This suggestion raised the question of how many such factorizations exist. It was quickly noticed that the number of such factorizations is related to the number of three way partitions of the prime factors of A (if repeated factors are properly taken into account). An attempt was made to find a rough upper bound on the number of factorizations; however it was dropped before it yielded specific numbers. Nevertheless people felt optimistic that, at least for a lot of A's, the number of factorizations wouldn't be enormous. This optimism was derived from the observation that either A would have few prime factors or that it would have many repeated prime factors.

Having agreed that it would probably be feasible (and possibly be useful) to enumerate all mrs factorizations of A, the class was faced with the question of what exactly to do with them. RC observed that one could, at least in principle, consider every pair of mrs factorizations. One factorization would define B, the other would define C; and it would only remain to check the length of the BC-face diagonal in order to detect almost brickness. He noted that unfortunately there would probably be far too many pairs of factorizations.

SS attempted to rescue matters by considering the impact of the modulo 4 and 16 constraints. EM intervened by begging the class to consider what if anything might be done given A and a factorization $A = mrs$.

HD suggested that if it could be done quickly, it might pay to factor B (where $B = m(r^2 - s^2)$). Somehow conversation then drifted back towards the idea of examining all pairs of mrs factorizations. That seemed to lead nowhere in particular and RC pointed out that the discussion was going in circles.

EM and HD again brought up the question of factoring B. EM recalled a remark he made at the last meeting, namely that (modulo some handwaving) reversing the roles of A and B correspond to replacing r and s by $r + s$ and $r - s$ respectively.

At this point, the question of why one would even want to factor B arose. It emerged that candidates for C could be generated from the factors of B in the same way that B was generated from the factors of A. (i.e. by considering all mrs factorizations of B.)

AS clarified the procedure by pointing out that since $B = m(r^2 - s^2) = m(r + s)(r - s)$ it is possible to quickly factor B by factoring $r + s$ and $r - s$, both of which are small.

2.5 Notes For Tuesday, January 24

This class session was devoted to a discussion of the various solutions to Problem **1.** Four groups of students consbructed programs for finding bricks. The groups were: {RWH, MGB, ST}, {HD, RC, SR}, {CWC, JP, SS}, {AS, AAS, ANS}.

HD presented her group's solution first. An outline of their program follows:

```
Construct an $A$ where $A$ is not, prime, not even, and $A \leq 10^6$.
    Choose a multiplier value m.
        Choose values for $r$ and s (with $r \leq 1681$, $s < r$ and $\gcd(r, s) = 1$).
            Compute value of "$B$" corresponding to $r,s$.
            Put $B$ on the appropriate list (there is a separate list for
            each of the various cases of $B$ relative to the modulo 3 and 4
            constraints).
    Search for suitable $B,C$ values from all pairs of compatible lists.
```

HD noted that her group's program can be run in a mode where it only examines values of A that have at most six distinct prime factors. She gave some statistics for running the program (written in PASCAL) in the unrestricted case: in examining all $A < 421520$, the program looked at a total of 3 million multipliers, 1 million values of B, and 1.5 million B, C pairs. This feat consumed 6000 seconds of VAX-11/780 CPU time and unearthed 6207 almost bricks (not necessarily with relatively prime edges).

EM remarked that it would be interesting to have an analysis of the length of the lists of candidate B and C values and of the total running time of the algorithm. HD pointed out that, empirically, it seems that there are only about twice as many B, C pairs to be examined as there are B values farmed out to the lists.

HD elaborated on the procedure for constructing A values by explaining that they are produced by a set of nested loops. Each such loop corresponds to a possible prime factor p of A; the loop index value determines the number of times that p divides A.

SR suggested a possible variation on HD's algorithm; namely to generate the possible r, s values in the outer loop rather than in an inner loop. EM noted that this table could be stored and consulted in the search for C values. SR raised the difficulty of searching through the r, s table in a useful order. EM proposed that storing several copies (organized in different, ways, corresponding to several small prime factors of rs) might alleviate the troubles.

AS described his group's program. Their program is quite similar to that of HD's group. The main difference is that AS's program chooses the r, s values *before* choosing m. The program uses the factorization of A to generate the relatively prime r, s pairs. The other difference between the programs is that the B and C candidates are not split into separate lists. AS pointed out that this choice reflected his group's estimate of the cost of separating the candidates into categories, versus the benefits of having fewer eligible pairs. AS noted that the largest encountered list had **1215** entries, but that it usually has only 30 or 40 en tries.

This program, written in C, was run on the VAX. (After it was discovered that the C compiler for the **DE-20** produces amazingly inefficient code.) It took 20 minutes of CPU time to search the space $A < 250000$, $B < 10^6$, C $< 10^6$ wherein it found 385 almost bricks (counting only those with relatively prime sides). With 71 minutes of CPU time it managed to examine the range A, B, C $< 10^6$ and find 242 almost bricks.

EM mentioned that it might be a good idea to have several nlgori thms; one to deal with values of A with lots of prime factors, and one to deal with values of A *composed* of few factors. AS noted that his program does this, at least implicitly. It uses different code to enumerate r, s pairs according to the "shape" (*i.e.* the number of factors and their multiplicities) of the factor table of A.

EM wondered whether anyone had managed to save computation by using B values from one choice of A in constructing B values for a different (but related) A. For example, it appears possible to efficiently transform the list of B, C candidates corresponding to one value of A into the list for any other value of A having more prime factors (or greater multiplicities of factors). No one confessed to having attempted such a scheme. However, under severe authoritarian pressure, SS was persuaded to give it a try.

AS admitted that his group's program did not examine even values of A. He noted that since an even value of A might have a lot of factors of 2, it might give rise to unhealthy amounts of computation. (Of course, this restriction matters only when the search space is asymetric in A, B and C.)

RWH then described his group's approach. They at tempted to find, for each value of A, two factorizations $m_1r_1s_1 = m_2r_2s_2 = A$. They discovered a number of constraints concerning common factors among the numbers m_1, m_2, r_1, r_2, s_1, s_2 (based o 1 the fact that A $= m_1m_2k$ where $k = \gcd(r_1 s_1, r_2s_2)$). By generating the m's, r's and s's from a table of factors, these constraints can be automatically satisfied. No run-time statistics were yet available for this approach.

SS's group wrote a program roughly along the lines of HD's group. However, SS wrote it in assembly language. (The program uses a special trick to rapidly eliminate pairs of incompatible multipliers (i.e. multipliers with common factors). The idea is to attach a bit vector to each B, C candidate which indicates the presence or absence of each possible prime factor in the candidate's multiplier. Whenever the logical *AND* of the vectors corresponding to a pair of candidates is non-zero, the pair is incompatible.) With an efficient program to generate the primes less than 333,333, his program took only about eight minutes of CPU time (DEC-20) to search the range A, $B, C < 10^6$.

Listed below are a few references to papers concerned with the topic of bricks.

[1] J. Leech, "The Rational Cuboid Revisited," American Math. Monthly, **84** (1977) pp.518-533.

[2] J. Leech, "Corrections to The Rational Cuboid Revisited," American Math. Monthly, **85** (1977) p.472

[3] M. Lal and W.J. Blundon, "Solutions of the Diophantine Equations $x^2 + y^2 = l^2$, $y^2 + z^2 = m^2$, $z^2 + x^2 = n^2$," Math. Comp., **20** (1966) pp.144-147

Reference [3] contains a list of 130 almost bricks.

2.6 Notes for Thursday, January 26

EM began the class with a brief summary of the results of Problem 1. The highlight was that SS had coded the problem in assembly language, achieving a running time of 8 minutes. To generate the primes up to about **400,000** required only 25 seconds. Discussion then shifted to Problem 2.

2.7 Notes for Tuesday, March 6

SS began the class by presenting his results from Problem 1. He had run his program up to $n = 10,000,000$ over the weekend. His complete results are summarized in the table below.

	$n = 1 \times 10^6$	$n = 4 \times 10^6$	$n = 10 \times 10^6$	Complexity
Time:				
Prime number generation	0:25 min.	2:55 min.	25:00 min.	$O(n^{1.5})$
Finding constants	2:00 min.	8:00 min.	29:00 min.	$O(n^{1+\epsilon})$
Rest of program	4:14 min.	20:34 min.	56:36 min.	$O(n^{1+\epsilon})$
space:				
Bit vector for primes	4.52 K	18.1 K	45.2 K	$O(n)$
Block size for $\langle m, r, s\rangle$ records	3.32 K	5.76 K	7.99 K	$O(n^{0.4})$
Results:				
Almost bricks	242	472	704	
Largest smallest prime factor	19	29	31	
Integer bricks	0	0	0	

EM had derived bounds on the running time for the various components of the program. For the finding of the constants and the rest of the program, the running time is $O(n \log^3 n)$. The space requirement for storing the bit vector of primes could be reduced to $O(\sqrt{n})$. The program would have to be modified so that the bricks were generated by the size of their largest side as opposed to the size of their smallest side. If this was done then the sieve could be constructed in blocks of size $\sqrt{n}$ as opposed to all at once. Only two blocks would be needed at a time. The program did not find any integer bricks in the range up to 10 million. EM conjectured that there are no integer bricks. A proof of this is left as an exercise to the reader.

A table of almost bricks is presented in the appendix. This list was compiled from the computer outputs provided by SS. The list is printed in two different sequences, once in increasing odd side length, and once in increasing shortest side length.

3. Scheduling

3.1 Problem Statement

A *scheduling problem* is given by a number m of identical *parallel* processors, and a system of n tasks together with *precedence constraints* (or a *partial order)* among these tasks. If $t \prec t'$ for two tasks t and t' the execution of task t has to be finished before task t' can be started. We assume that $\prec$ is *transitively reduced*, i.e., if $t \prec t'$ then there is no t'' different from t and t' such that $t \prec t'' \prec t'$.

A *scheduling method* is a predicate which tells us, for every task system and number m' of *idle* processors, which tasks to start executing on the idle processors. A (nonpreemptive) schedule can, therefore, be obtained in the following way.

(i) at time 0, all m processors are idle; we use the scheduling method to determine, which tasks of the task system should start execution. We assign these tasks to the appropriate number of processors and remove them from the task system.

(ii) at any time when a processor becomes idle, and there is at least one task left to be executed, we determine the number m' of idle processors at that time. Again, we use the scheduling method to determine which tasks of the task system should start execution, we remove them from the task system and assign them to the appropriate number of processors.

Note that in general, if at some time, $m' > 0$ processors are idle, not necessarily all of them are assigned a task at this moment. It could be that there are not enough tasks available to be executed, or the scheduling method could intentionally leave processors idle! We insist, however, that new tasks are started only (except at time 0) whenever a task is finished.

We now consider the case that the execution time t_i of the i-th task is not a **fixed** integer but randomly distributed. In particular, we assume that all execution times t_i are independent, identically distributed random variables with a negative exponential distribution and mean 1. Thus,

$$\text{Prob}(t_i \le a) = 1 - e^{-a} \quad \text{for all } a \ge 0$$

where e = 2.71828.. . .

We also assume that the number of processors is fixed, m = 3, and that the precedence constraints are restricted in the sense that they form, in graphical terms, *in-trees* with **a** small number of branches, say 3 or 4. We only look at task systems with at most 30 **tasks.**

Every scheduling method determines, for every task system with precedence constraints, an average execution time for this system. Our goal is to devise several scheduling methods and compare them with respect to minimizing this average execution time.

3.2 Notes for Thursday, January 26

EM asked if there were any initial observations about the second problem.

SR pointed out that with the greedy algorithm, the number of active processors will never increase, i.e. once there are k idle processors, there will always be at least k idle processors.

RWI-I brought up the memoryless property of exponential distribution, after a job has run t units, its time until finishing still has the same exponential distribution.

EM then began providing some background for scheduling problems in general. The basic problem is given a collection of tasks and several processors, what gets scheduled when and where. There are many different scheduling problems, depending on what features are chosen. Some of the facets of scheduling problems are:

- a single processor or several processors.
- execution times could be all the same, different, or random.
- the tasks could be subject to precedence constraints, either as a tree or as a directed acyclic graph.
- there could be release times, deadlines, or due times for the tasks.
- there are many possible optimality criteria, such as the total time taken or minimizing the total lateness of the jobs.

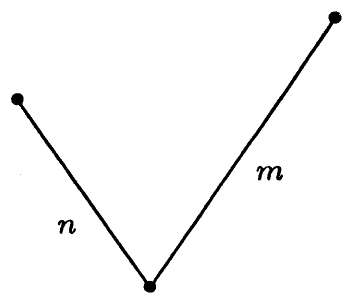
Taking various combinations of these, one count gives about 9000 scheduling problems, 80% of which are NP-complete, 10% are known to be polynomial, and 10% are open. J. K. Lenstra et. al. have developed a computer data base to keep track of results on scheduling problems (CACM November, 1982).

The precedence constraints for the tasks, are in general a partial order. Possibilities include no constraints, a linear order, or as in the case of this problem, the constraints form a tree. The tasks can be viewed as being in levels, either with the tasks pushed down as far as possible to give a latest possible time (LPT) they could be run, or pushed up to give the earliest possible time (EPT) they could be run. With the LPT levels, if the tasks are taken off from the highest level first (the HLF algorithm) the tasks are done first which are the longest distance from the root. A classic result of scheduling theory is that the HLF algorithm is optimal for the case of in-trees (T.C. Hu, 1961). The problem of two processors for arbitrary precedence constraints was solved by Coffman and Graham in 1971. Most generalizations of these problems are NP complete, including m-processors with arbitrary precedence constraints and 2 processors with execution times of 1 and 2 (Ullman).

The motivation for looking at stochastic scheduling is that in real applications it is not generally known how long a task will take when it is scheduled. To model this it is necessary to assume some probability distribution for the execution times of the tasks. The probability distribution that will be studied is exponential distribution. For the case of 2 processors with arbitrary precedence constraints it is known that the HLF algorithm is the best in the sense that it gives the lowest average execution time. For the case of 3 processors, the problem is open, although examples are known where HLF is not optimal.

In our case the distribution for the task execution times is the exponential distribution with mean 1. The distribution function for exponential distribution with mean μ is $f(x) = 1 - e^{-x/\mu}$ for $x \geq 0$ and the probability density function is p(z) = $\frac{1}{\mu}e^{-x/\mu}$ for $x \geq 0$. One important property of this distribution is that if X_1, X_2 arc independent random variables with exponential distribution and mean μ then $\min(X_1, X_2)$ has exponential distribution with mean $\frac{1}{2}\mu$. Similarly the minimum of three indcpcnclcnt exponential random variables with mean μ is exponential with mean $\frac{1}{3}\mu$. This is about the extent of the probability theory that will be required for the problem. More information can be found in Knuth section 3.4.1. The basic idea is to commit three processors to three leaves of the tree and run them until one finishes. The task that is finished can be removed and a new leaf is assigned to the processor. The exponential distribution allows viewing these as all three processors starting anew. The memoryless property of the distribution allows execution times of the old tasks to bc regenerated and a time to be generated for the new task.

To begin the actual problem, EM suggested to look at the problem of computing the expected time for scheduling trees on two processors. The first case is to consider scheduling a tree with just two branches.



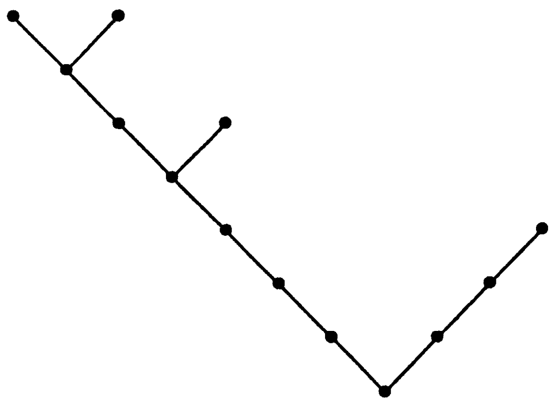
As long as neither branch has been retnovcd, both processors will work, and when there is only one branch left a single processor can work. The time will be the average number of steps to remove a branch plus the average number of tasks left when one branch is removed. This problem is essentially the Toilet Paper Problem from an AA qual. The setting for the problem is a stall with two rolls of t.p., each of which initially has n sheets. People randomly select a roll and use a sheet from it, the question tltat is asked is what is the expected number of sheets left on a roll when the other one is used up. It is natural to view this problem using a lattice. The point (n, m) contains the expected number of steps to remove all the tasks from a bush initially with n and m tasks on each branch. EM suggested a dynamic programming approach to constructing a table of the expected values. He recommended that the entries in the table be normalized so that the entries were intcgcrs. Since the trees that the problem deals with are small ($\leq$ 30 nodes), intcgcrs will not get too big. A closed form for the expectation would bc interesting, but the best EM said ho had come up with was a fairly ugly summation. As problems for the next class, EM suggested thinking about how to handle the various cases of trees with 3 leaves and how to enumerate the topological classes of trees with 4 and 5 leaves.

3.3 Notes for Tuesday, January 31 .

The discussion began with a clarification of what the actual problem is. The first part of the problem is to compute the expected time for scheduling 2 and 3 leaf trees. A precedence tree will be given as input. With this information stored in tables, it will then be possible to compute the expected times for scheduling using various rules for assigning tasks to processors. The goal will be to come up with several methods and to compare their performance.

The question of where the scheduler resided arose. EM said that it should be regarded as an oracle that is consulted every time a decision must be made, the oracle requires no time to answer.

GP presented an example of a tree where the HLF method does not give the best average performance. The qualitative argument is that you want to keep as many processors busy for as long as possible. With the HLF algorithm, the number of leaves would be reduced to 2 very quickly. A better approach would bc to assign a processor to the lowest leaf so as to keep 3 processors busy for as long as possible.



EM suggested that the best thing to do is to keep as many processors busy as possible since the mean time for 3 processors to complete a task is $\frac{1}{3}$, the mean for 2 is $\frac{1}{2}$ and the mean for **1** is 1. The problem will be to minimize a weighted sum. There will be some trade offs as to how long a number of processors will be busy.

SR formulated a heuristic of deferring removing leaves for as long as possible.

EM said one thing to pay attention to was the points where it went from 3 processors working to 2 processors to 1 processor.

The discussion then returned to computing the average time for a two branch bush. The branches will have m and n nodes each, with $m \geq n$. What is needed is to compute the expected number of nodes left when one branch is used up. To compute this a lattice is used, with the point (m, n) representing a bush with branches of length m and n. The two leaves have the same probability of being removed, so there is a $\frac{1}{2}$ chance of going down and a $\frac{1}{2}$ chance of going left. The probability of having j nodes left on the second branch when the other one is removed is half the probability of going from (m, n) to (1, j). The number of paths form (m, n) to (1, j) is $\binom{m+n-j-1}{n-j}$ so the probability is $\binom{m+n-j-1}{n-j} 2^{-(n+m-j-1)}$. The expected number of nodes left when one branch is finished is

$$A_{m,n} = \sum_{0 \leq i < n} \frac{1}{2} i \binom{m+n-i-1}{n-i} 2^{-(m+n-i-1)} + \sum_{0 \leq i < m} \frac{1}{2} i \binom{m+n-i-1}{m-i} 2^{-(m+n-i-1)}.$$

SR had computed the exact value of this in the case that the branches have the same length. He defined $T_{m,n}$ to bc the expected time to remove all of the tasks. It is easy to sec that $T_{m,n} = \frac{1}{2} A_{n,n} + n$. His result is that $T_{m,n} = n + \frac{n}{2^{2n}} \binom{2n}{n}$.

EM said that these values could be computed by dynamic programming. The expected number of tasks left when one branch is finished satisfies the recurrence:

$$A_{m,n} = \frac{1}{2}(A_{m,n-1} + A_{m-1,n}), \qquad A_{m,0} = A_{0,m} = m.$$

The values are symmetric with respect to the diagonal, so only the part of the matrix below the diagonal really needs to be computed. Another consequence of this is that

$$A_{i,i-1} = A_{i,i}.$$

Sittcc the values on the diagonal are the same as the values on the subdiagonal, it is not necessary to compute the values on the diagonal. The values on the diagonal are

$$A_{i,i} = \binom{2i}{i} \frac{i}{2^{2i-1}}.$$

EM asked the class to compute the values for a 30 x 30 table being careful of round off, perhaps using double or triple precision integers. The generalization of this to the 3 leaf case is fairly straight forward. In that case, the values should be computed for the slice of the *30* x *30* x 30 cube that has $i + j + k \leq 30$. One interesting thing to look at are the contours of approximately equal expected times in these tables.

There arc a number of cases of trees to consider when the tree has 3 or more leaves. One way to get canonical members of the topological classes is to rotate the branches so that the highest branch is on the left. There are two classes for 3 leaf trees. The class was asked to identify the classes for 4 and 5 leaf trees.

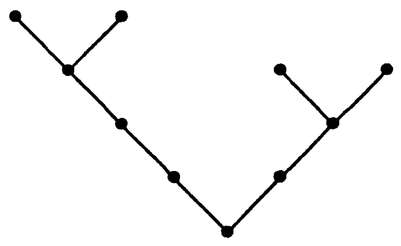
Here is a small table for the $A_{i,j}$:

					$\frac{2772}{1024}$
				$\frac{630}{256}$	$\frac{1386}{512}$
			$\frac{140}{64}$	$\frac{315}{128}$	$\frac{756}{256}$
		$\frac{30}{16}$	$\frac{70}{32}$	$\frac{175}{64}$	$\frac{441}{128}$
	$\frac{6}{4}$	$\frac{15}{8}$	$\frac{40}{16}$	$\frac{105}{32}$	$\frac{266}{64}$
$\frac{1}{1}$	$\frac{3}{2}$	$\frac{9}{4}$	$\frac{25}{8}$	$\frac{65}{16}$	$\frac{161}{32}$

3.4 Notes for Thursday, February 2

EM began the class by asking what progress had been made on the scheduling problem.

RC had looked into the problem of enumerating all trees that would be in the range of interest. For trees of 2 and 3 branches there is only one type of each, while there are two distinct types of 4 branch trees. The trees can be characterized by the lengths of the various segments of the trees. For 4 branch trees, 6 parameters are needed. RWH pointed out that by allowing some of the parameters to be zero, nodes of degree greater than two can be handled. RC said that the obvious way to store the trees would be to use a 6 dimensional array, but that the array would be so big that it wouldn't be feasible and only a small portion of the array would be relevant. A different approach would be to use some kind of encoding of trees. One possibility would be to use a LISP like encoding where the first element would be the distance from the root to the first branch and the remaining elements would be lists for the subtrees. For example the tree



would be (1(311)(211)). With an encoding of trees, they could be ordered lexicographically, and then they could be accessed in log n time using binary search.

RC then pointed out that it is also necessary to keep track of which leaves had processors assigned to them. There was some discussion of this before it was agreed that it probably is necessary to record which tasks have been assigned. The problem is that this information further increases the number of states to keep track of. EM said that it might be possible to use the scheduling rules to reduce the number of combinations of leaves that had been assigned. RC suggested that restricting the classes of scheduling rules might also help.

GP had developed a similar characterization of trees as RC's. GP said that allowing preemption would get rid of the problem of recording active tasks and might also allow better schedules.

EM agreed that things would be simpler with preemption, but recommended that the class continue working on the given problem. EM also said that for the two processor case, preemption did not help.

GP brought up the problem of large numbers for computing the three dimensional case, saying that the denominators would be larger than the two dimensional case and would contain powers of both 2 and 3.

SR wondered if it might be better to store probabilities instead of expectations in the tables. Overflow might not be quite as bad with probabilities, it wouldn't occur until tables were 18 x 18. EM thought it was still better to store expectations since look up could be done with one access, while working with probabilities there would bc quite a few accesses needed to compute the expected value. This probably is an important efficiency consideration. --

AS said that the number of trees of interest at a given time will be small, so that it will only be necessary to look at neighboring trees. AS hoped that it would be possible to avoid generating all subtrees. EM wanted the method to work so that given any tree (within the size of interest), the best way to schedule the tree could be found. EM said it would bc best to try to generate all trees with some mechanism so that table look up would be very fast. One problem that would be nice to solve is to End the smallest counter example to HLF.

ST had also looked into the problem of enumerating trees. ST had a recursive procedure for doing this, one aspect of it was a way to avoid generating duplicate trees.

RWH had generated the 30 x 30 table discussed during the previous class. He had used his own multi-precision routines. The value that was calculated for $A_{30,30}$ was about 6.5.

MGB had thought about experimental simulations as a way to test various scheduling strategies. EM said a problem with this approach would be that the differences in expected times would be very small. In order to get statistically significant results the number of cxpcrimcnts might have to bc very large.

AAS had an idea about generating trees by considering the junction types. To get the sizes of trees, partitions could be used. The partitions could bc precomputed and stored in tables. Duplicate trees would only be generated when the trees were symmetric.

There was a question about how the tables for 3 branch trees would be used since the tables were just based on the branch sizes. EM said that the tables would aid in computing the transitions from 3 leaf trees to 2 leaf trees.

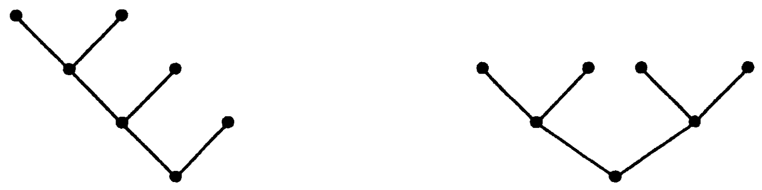
CWC had counted the topological classes of 4 and 5 branch trees by a brute force approach. There are 27 classes of 4 branch trees and 236 classes of 5 branch trees.

AS brought up the question of what the expected time for an algorithm would bc if the input tree was random. AS pointed out that this would require a probability space of trees. EM said that this was a different problem since we want to be able to compute the expected time for a given tree as input.

The discussion then turned to various scheduling methods, to come up with approaches that would be different from HLF. The idea is to come up with a predicate that can be used as a scheduling rule. SR pointed out that by dynamic programming optimal tables could be constructed. This would give a rather complicated predicate that would be optimal. As a simple predicate, EM suggested that one might want to keep three leaves for as long as possible. SS suggested working on the longest chains first. RWH said that it might be a good idea to choose the task at greatest distance from the two running tasks. Some rule would also be needed to start things up.

3.5 Notes for Tuesday, February 7

RWH began the discussion by presenting a method to represent trees for the dynamic programming approach to the problem. For 4 branch trees there are two distinct types of trees to be considered. These must bc handled separately. The trees are:



Each of these trees can be represented with 6 parameters indicating the lengths of -each segment (segments of length 0 arc allowed so as to handle trees with nodes of degree greater than two). Since any segment can be of length up to 30, (since wc arc restricting ourselves to up to 30 node trees), to store these in a 6-dimensional array would require $30^6 \approx 7 \times 10^8$ words. The basic problem with the array representation is there are many states that correspond to trees with more that 30 nodes. What is needed is an efficient way to order the sixtuples which sum up to at most 30. RWH had computed the numbers of ordered k-tuples that sum up to n. His results were:

All terms positive, sum to exactly n	$\binom{n-1}{k-1}$
All terms positive, sum to at most n	$\binom{n}{k}$
All terms non-negative, sum to exactly n	$\binom{n+k-1}{k-1}$

All terms non-negative, sum to at most n	$\binom{n+k}{k}$

The method used to compute the first of these is to consider n points on a line and to count the number of ways dividers can be put down to divide them into k groups. Each division corresponds to a sum of k terms. The same method works for non-negative terms, except that k extra elements need to be added and each group of j elements gives a term of $j - 1$. The number of trees of each type that have less than thirty nodes is about $\binom{36}{6} \approx 2,000,000$.

Knowing the number of trees with a given number of nodes suggests a way of storing the values in a one dimensional array. We first divide the array into n blocks to store all trees of exactly j nodes. Then for each block we divide it into subblocks corresponding to the first branch having exactly i nodes on it. We continue subdividing the blocks until all branches are done. The location is computed as follows. Assume the 6-tuple is stored in the array $e[1..6]$. We define $c[i] = \sum_{1\leq j\leq i} e[j]$. The location is $\sum_{1\leq i\leq 6} \binom{c[i]+i-1}{i}$.

One important facet of this approach is that it should be good for paging. With the dynamic programming approach, the lower branches of the tree will change very infrequently, it will be just the upper branches that will be accessed frequently. If the indexing is done properly, the portion of the table that is being used will be small enough that few page faults will be generated.

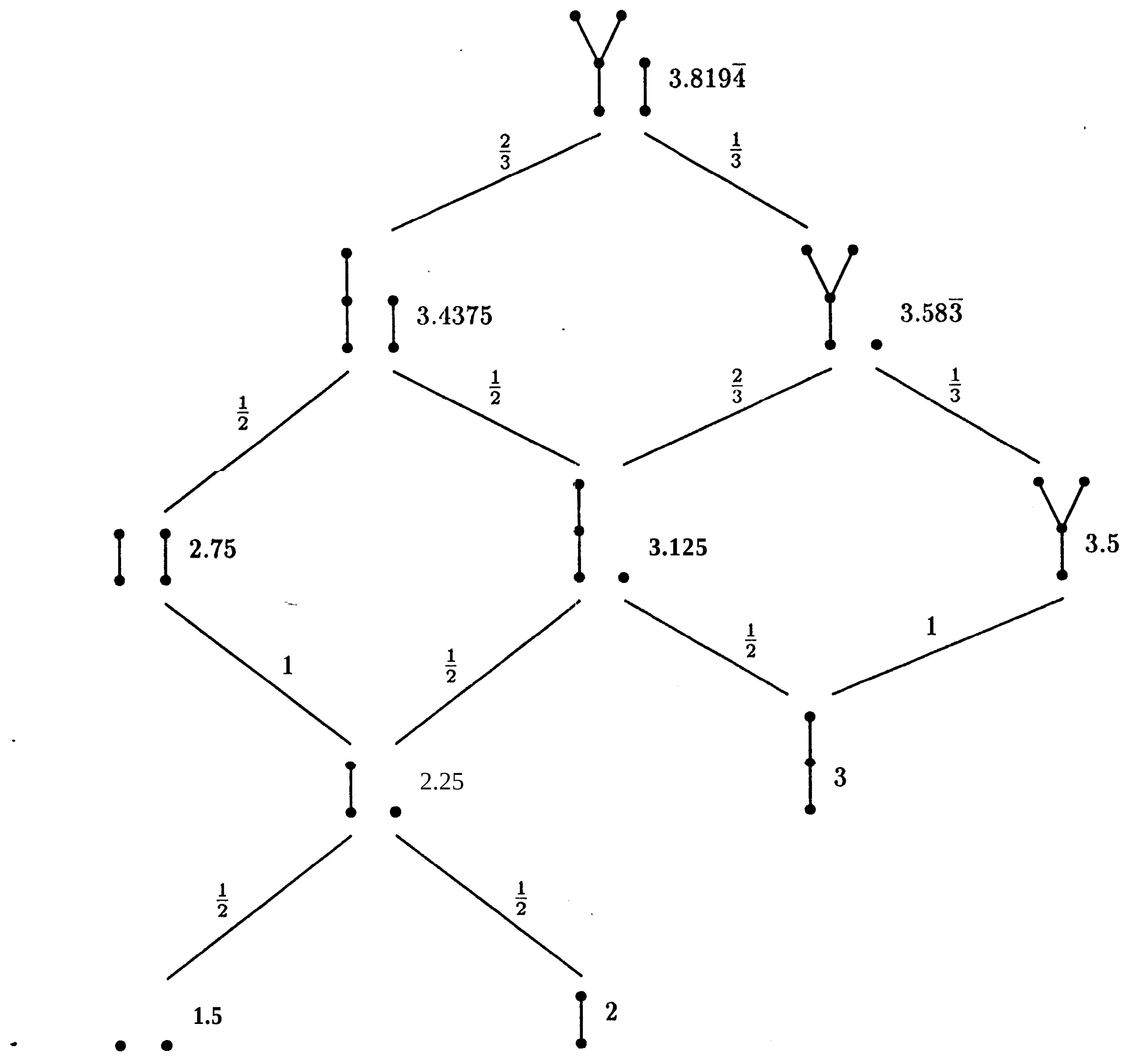
Using the symmetries of the trees, the number of table entries may be reduced. For the first case of tree, it may be assumed that the left branch is longer than the right, so the storage requirement may be reduced by roughly half. For the second type of tree there are several symmetries that may be used. This complicates the indexing somewhat. SR outlined a method to handle this complication.

The discussion then digressed to the merits of C over Pascal. The consensus was that C is much better for bit twiddling and manipulating large arrays.

HD suggested that hashing would be an alternate approach to RW II's approach of direct indexing. The problem with hashing is that the values would be scattered throughout the table which would cause a large number of page faults to occur if the table was large. AS suggested a hybrid approach, where the lower branches of the tree would be used as indices and the upper branches would be hashed using reasonable sixed tables. SR said that computing a hash function would probably be faster than computing an index.

EM presented an example of computing the expected finishing time for a tree to illustrate the details of the dynamic programming computation. The numbers by the subtrees are the expected finishing times and the values on the arcs are the probability of going from one tree to another. The computation of the expected times for the lowest trees are straightforward. The computation goes from the bottom up to the top. For example,the value at the top is $\frac{2}{3} \times 3.4375 + \frac{1}{3} \times 3.583 + \frac{1}{3}$. This is the weighted sum of the expected times for the two trees reachable from it plus the expected time for the task to be con&ted.

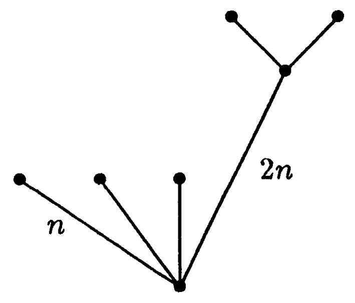
EM said that the programs only needed to be run until a counter example to each scheduling proposal was found. EM expected that counter examples most likely exist with trees Of less than 20 nodes.



A couple more proposals for scheduling methods were made. One idea was to assign weights to branches and take the highest weights first. EM and IID suggested different weighting schemes. Another idea would be to look at various cuts of a tree which reduce it to a 3 leaf tree. There might be some way to use this to form a useful heuristic for scheduling tasks.

3.6 Notes for Thursday, February 9

MGB began the class by presenting an example of a tree which would serve as a counter example to the "longest twig" method for scheduling. The tasks are assigned which have the longest distance to a branching point. In this example the three lowest branches would be done before tackling the long branch. It is fairly obvious that a better approach would be to start with at least one processor on the long branch. EM said that he would still like to know what the smallest counter example to this method is.



The groups summarized what approaches they were taking. None of the groups had their programs running.

The group of RWH, MGB, and ST had decided to use 8 tables for dynamic programming. Each table would represent a different type of tree. The trees with nodes of degree 3 or 4 were not considered special cases of 2 node trees. Working with the extra trees means that the segments would all be positive. This both reduces the size of the tables and cuts down on some duplication. RWH commented that the really messy part of this was figuring out the transitions from one tree to another when a branch is exhausted. They were planning to store all levels for 2 and 3 branch trees and only to save the two previous levels for 4 branch trees. The estimate for storage required was 170K for trees up to 20 nodes.

SR said that the table could be reduced to 65,000 elements by using the various conditions to remove duplicates. This did not count keeping track of which tasks were already assigned. To keep track of which tasks were #assigned, SR said it is better to view it as 3 tasks busy and 1 unassigned as opposed to 2 tasks busy and 2 tasks unassigned. This reduces the number of cases from 6 to 4. However this does make the transitions from state to state more complicated.

The next group to present their ideas was $A(\epsilon + A + N)S$. They were planning to use only 2 tables, so that the transition from one table to another would not be too complicated. They raised the question of how the scheduling predicate should be represented. They wanted to make it as general as possible, allowing predicates to be entered interactively and to be able to experiment with preemptive schedules. EM said that it was desirable to have the scheduling predicate as a separate module. EM pointed out that a more general method would require extra computation.

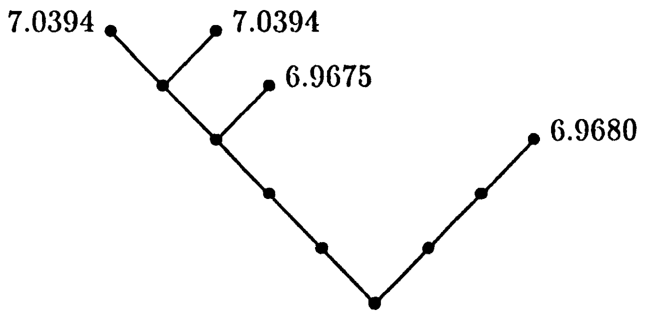
The group SR, HD, GP had decided to use a table look up method which differed from both hashing and direct indexing. The first step is to enumerate the trees. A tree can be represented by 4 numbers, since the interior branches will not be reduced. The table for n node trees is generated from the table for $n - 1$ node trees. Once the list of trees has been generated it is converted to an array. They were planning to save intermediate results on disk so that if the program crashed from running out of space, all would not be lost.

CWC pointed out that there was a subtlety in the problem that was being overlooked. In looking for a counter example, the groups were looking for a state where the scheduling rule would make a non-optimal choice. The state however might not be one which the scheduling method would ever reach, since the state contains both assigned processors and a tree. It might be the case that whenever that tree is reached by the scheduling method, a different set of tasks is assigned. AS pointed out that a scheduling rule tells what to do initially as well as when several tasks are running, in fact completely different rules can be used for these cases.

There are two cases to consider when looking for counter examples. The first case is the initial step when 3 tasks are assigned. If this initial assignment is non-optimal, then a counter example has been found. The other case is when two tasks are assigned and a new one is assigned. If this differs from the optimal choice the method is locally non-optimal. To prove it is a true counter example, a different method must be employed to show that this state is reachable from some initial configuration.

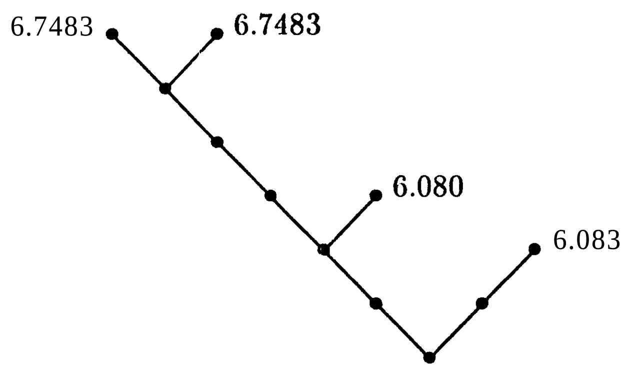
3.7 Notes for Tuesday, February 14

The class began with reports on the status of programs for problem 2. Most groups were still debugging their programs. RC and SR had gotten their program at least partially operational; it has examined all trees with four leaves and at most 15 nodes. The program has discovered a couple of counterexamples to the optimality of HLF scheduling. The smallest counterexample is presented below. The number attached to each leaf is the best expected total running time when that leaf is *not* among the three tasks first initiated.



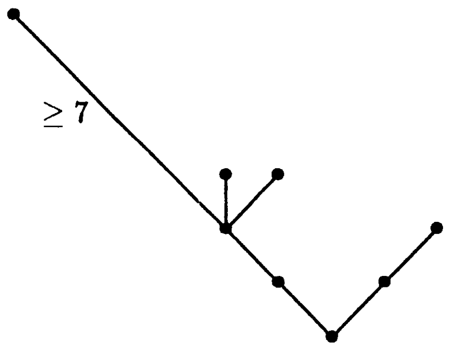
RC pointed out that this counterexample serves to demolish the optimality of all the scheduling heuristics proposed at the last meeting.

SR drew another counterexample shown below. RWH's program had also discovered it.

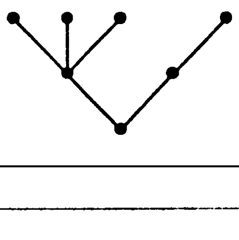


ANS mentioned that when one branch is long, the total running time is dominated by that branch, and the variations in quality of the heuristics are small. EM added that in practice, the errors committed by heuristics are rarely expensive. Nevertheless, he would be interested in further analysis of their behavior.

SR noted another class of counterexamples:



EM contributed an example where HLF scheduling *might* fail to be optimal; results depend on how tics are broken:



EM suggested thinking about other heuristics; SR said he had tried a weighting function based on the sum of leaf depth and twig length but that it performed worse than HLF. SR noted that his group's program was quite modular and that it would be feasible to experiment with alternative heuristics.

4. Presburger Arithmetic

4.1 Problem Statement

Let $A = (a_{i,j})_{1 \le i,j \le n}$ and $B = (b_{i,j})_{1 \le i,j \le n}$ be two matrices, and set $C = AB$. To compute $C = (c_{i,j})_{1 \le i,j \le n}$, we might define

$$c_{i,j,t} = c_{i,j,t-1} + a_{i,t}b_{t,j} \quad \text{for } t = 1,\ldots,n;$$
$$c_{i,j,0} = 0,$$

and we would then have $c_{i,j} = c_{i,j,n}$. However, if we try to implement this algorithm in a straightforward manner on a parallel (say, systolic) architecture, we find that at time t the *single* datum $a_{i,t}$ is simultaneously used to compute the n quantities $c_{i,j,t}$.

If, on the other hand, we set

$$c'_{i,j,t} = c'_{i,j,t-1} + a_{i,t+2-i-j}b_{t+2-i-j,j},$$

we avoid this problem. Of course, t now runs from 1 through $3n-1$, and the array elements not within the original index range have to be preset to zero. Since index transformations can be quite tricky, and we are almost bound to make some error in the computations, we'd like to have a program check them. As a matter of fact, such verification is a (small) part of an interactive program transformation system for parallel programs and architectures under development.

More specifically, we are interested in the class $RP = RP(n_1, n_2, \ldots)$ of sets of integers or integer vectors given in the following way (here, $n_1, n_2, \ldots$ are *global* variables ranging over integers):

(i) *RP* contains all *simple* sets of integers, i.e., the intervals $\{i;\ lb \le i \le ub\}$ with *lower* and *upper bound expressions* lb and ub, respectively. A *bound expression* is any (quantifier free) arithmetic expression containing rational constants and global variables. The latter are allowed to occur only linearly. Besides addition, subtraction, and multiplication (by constants), we also include taking the modulus with respect to constants as arithmetic operation.

(ii) If A is in *RP*, and a and b are some integer constants, then $\{i;\ i \in A \text{ and } i \equiv b \pmod{a}\}$ is in *RP*.

(iii) *RP* is closed under finite union, intersection, and cartesian product.

(iv) *RP* is the smallest such class.

We also consider the subclass *RF* of the class of functions between members of *RP*. For any function $f(i_1,\ldots,i_r) = (f_1(i_1,\ldots,i_r),\ldots,f_m(i_1,\ldots,i_r)) \in RF$, each $f_k(i_1,\ldots,i_r)$ is of the form:

```
if $C_1$ then $E_1$
    elsf $C_2$ then $E_2$
        elsf ...
            elsf $C_k$ then $E_k$ else $E_{k+1}$ fi.
```

Each C_i is an arithmetic condition (comparison of two arithmetic expressions as above wrt. $<$) or a boolean combination of such conditions. Each E_i is an arithmetic expression as above.

Example: From the matrix multiplication example, we would have sets

$$A', B', C' = \{(i,j,t);\ 1 \le i,j \le n, 1 \le t \le 3n-1\},$$

which are members of *RP* (though we have used a syntax here not provided in the definition).

In *RF*, WC would have the *initialization function* ι from A' to *A*, given by

$\iota(i,j,t)$ = **if** $1 \le t+2-j-i \le n$ **then** $(i, t+2-j-i)$
else "some escape value"

and an analogous function for B'.

If WC now assign $a'_{i,j,t}$, $b'_{i,j,t}$, and $c'_{i,j,t}$ to some processor $p_{i,j}$ for all t, we obtain one version of systolic matrix multiplication.

Your task is to design routines to

— check set inclusion for pairs of sets in *RP*;

— compute the cardinality of sets in *RP* (in terms of the global variables);

— check whether functions in *RF* are 1-1;

— check when given *A*, $B \in RP$ and $f \in RF$, whether f is defined on all of A and whether f is into (respectively, onto) *B*.

4.2 Notes for Tuesday, February 14

EM gave a short lecture on Presburger Arithmetic. Presburger Arithmetic is a first-order logic system for describing the behavior of the integers under addition. It was first presented in 1929 by a Mr. M. Presburger in a paper entitled "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen in welchem die Addition als einzige Operation hervortritt," which appeared in Comptes-Rendus du 1er Congrès des Mathématiciens des Pays Slavs. In first-order logic, quantified variables range over elements of a domain (as opposed to second-order theories wherein variables are also allowed to vary over subsets of the domain). In Presburger Arithmetic the domain is taken to be the integers. (In some formulations the domain is the non-negative integers. However this apparent distinction is eradicated by representing numbers with pairs (x, y) whose meaning is $x - y$. When the domain is restricted to the non-negative integers the relation symbol '$<$' adds nothing new to Presburger Arithmetic.)

The language of Presburger Arithmetic is given by the following set of recursive definitions.

- *Symbols*: The permitted *symbols* are $\vee, \wedge, \neg, (,), =, \forall, \exists, <, +, -, 0, 1, x, y, z, \ldots$.

- *Terms*: The constants (0 and 1) and all the variables are *terms*. Also, if t_1 and t_2 are *terms*, then so are $(t_1 + t_2)$ and $(-t_1)$.
- Atomic *formulae*: If t_1 and t_2 are *terms*, then $(t_1 < t_2)$ and $(t_1 = t_2)$ are atomic *formulae*.
- *Formulae*: Every *atomic formula* is a *formula*. Furthermore, if A_1 and A_2 are atomic *formulae*, then $\forall x A_1$, $\exists x A_1$, $(A_1 \vee A_2)$, $(A_1 \wedge A_2)$, and $\neg A_1$ arc all *formulae*.

A sentence in the language of Presburger Arithmetic is a formula in which all variables are "bound". (Variables arc "bound" when quantified; the scope rules are just what you'd expect if you know PASCAL).

The interpretation of sentences will not be presented formally; interpretations are intended to be the "natural" ones for integers under addition. Thus a sentence will be true, false, or, for all we know at this point, indeterminate. For example, the sentences $1 < 1$ and $\forall y \exists x \forall z [x + y = z]$ are false.

To simplify things, we will freely use additional symbols such as $\geq$, $\Rightarrow$, etc. These symbols add nothing new; they can all be translated into the language defined above.

EM asked whether the constants zero and one are really needed. RC noted that zero can be represented by the formula $\exists x_0 [x_0 + x_0 = x_0]$. EM elaborated by explaining that given a sentence S, it is possible to construct an equivalent sentence by replacing all zeros in S with x_0, and inserting the result into $\exists x_0 [(x_0 + x_0 = x_0) \wedge \ldots]$.

After a few false starts, the class found the following way to represent the constant one: $\exists x_1 \forall y ((x_1 > x_0) \wedge (y > x_0 \Rightarrow y \geq x_1))$.

EM ordered the class to think about how to represent any constant c using as a short a formula as possible (ideally $O(\log |c|)$ symbols). RWH ruined the assignment by hinting that it would help to start by defining powers of two.

EM noted that there arc other unnecessary symbols in the language. For instance, $\forall x A$ is equivalent to $\neg \exists x \neg A$. He suggested that it would bc useful to introduce abbreviations such as $2x$ into the language.

EM emphasized that all variables occur linearly in Presburger *sentences*. (One could allow rational coefficients -- they can always be eliminated by appropriately multiplying the offending formulae). Thus the following formula (intended to represent compositeness) is illegal: $\exists d_1 \exists d_2 [d_1 \neq 1 \wedge d_1 \leq d_2 \wedge x = d_1 d_2]$.

RC wondered whether multiplication could be defined somehow in Presburger Arithmetic. EM claimed not. RC suggested that perhaps at least compositeness could be somehow defined. Again EM claimed otherwise. EM did point out that it is possible to express divisibility by constants. He asked what other concepts are expressible. RWH suggested additive inverses, RC mentioned commutativity, and AAS tried induction.

EM argued that the axiom of induction cannot be expressed in first-order logic; to do so requires variables ranging over predicates (equivalent to sets) and thus requires second-order logic. (EM noted the concept of weak second-order logic, in which variables arc permitted to range over finite sets.)

EM then explained how Presburger Arithmetic can be used to represent sets. For example, the even integers can be expressed by $\{x \mid \exists y [x = 2y]\}$. Thus systems of linear equations over the integers can be described in Presburger Arithmetic.

The class observed that it is possible to express the fact that a given set is finite (or infinite); the necessary idea is that a set (of integers) is finite if and only if it has both a smallest and a largest member.

EM related Presburger Arithmetic to semi-linear sets.

Definition 4.1: Let 6 and $p_1, p_2, \ldots, p_r$ be vectors in $\mathcal{Z}^m$. Then the set $\{x = b + n_1p_1 + n_2p_2 + \ldots + n_rp_r \mid n_1, \ldots, n_r \in \mathcal{N}\}$ is *linear.*

The intuition here is that 6 is the apex of a "cone" formed by the members of the set. Note that the p's need not be linearly independent and that the cone may have holes in it.

Definition 4.2: A send-linear set is the finite union of linear sets.

It is a theorem that a set is describable by Presburger Arithmetic if and only if it is semi-linear. It is also true that the intersection of two semi-linear sets is semi-linear.

EM also noted that semi-linear sets are related to context free languages by Parikh's Lemma. Each word in a context free language defines a *count* vector as follows: the i^{th} entry of the vector gives the number of occurrences of symbol i in the word. Parikh's Lemma states that the set of count vectors corresponding to any context free language is semi-linear. This lemma is proved by applying the Pumping Lemma.

4.3 Notes for Thursday, February 16

This session was devoted largely to an explanation of a decision method for Presburger Arithmetic. The method is based on a fairly general idea called quantifier elimination (rather than, for example, theorem-proving heuristics from Artificial Intelligence).

EM first presented an outline of the method. Suppose that it is desired to check the sentence

$$s = Q_1x_1Q_2x_2 \ldots Q_rx_r\exists xF(x_1, \ldots, x_r, x)$$

(where each Q_i is a universal or existential quantifier) for validity. Assume that F contains no quantifiers, i.e. that S is in prenex standard form. Then the method of quantifier elimination calls for replacing the formula $\exists xF(x_1, \ldots, x_r, x)$ with an equivalent formula $F'(x_1, \ldots, x_r)$ containing no quantifiers (or new variables). The process is then repeated on the sentence

$$S' = Q_1x_1, \ldots, Q_rx_rF'(x_1, \ldots, x_r).$$

After r iterations, one is left with a sentence equivalent to S which contains **no** quantifiers. Checking such a sentence for validity is trivial.

To simplify matters, we will forbid the use of universal quantifiers. Thus a formula $\forall xF(x)$ must be rewritten as $\neg\exists x\neg F(x)$. In light of this restriction and the above outline we will consider from here on only formulas of the form $\exists xF(x)$, where F contains no quantifiers (but may contain unbound variables).

In order to explain the method, EM proposed the following additional restrictions on Presburger syntax.

- The equality relation is not allowed. $[u = v]$ is replaced by $[(u < v + 1) \wedge (v < u + 1)]$.
- The only allowed relations are: $<, |, \nmid$. (As before, we require that the left hand sides of the $|$ and $\nmid$ relations be constants.)
- Negations must be "pushed inwards" and eliminated. For example, the formula $\neg(a < x) \Longrightarrow (a \geq x) \Longrightarrow [(x < a) \vee (a = x)] \Longrightarrow [(x < a) \vee [(a < x + 1) \wedge (x < a + 1)]]$ (the symbol "$\Longrightarrow$" means "is transformed to").
- Like terms must be combined. Thus, $[x < 3 + x + x] \Longrightarrow [-3 < x]$.

After the above transformations arc made, every atomic formula in F must be of one of the following types:

(1.) $c_i x < a_i$
(2.) $b_i < c'_i x$
(3.) $d_i | (c''_i x + r_i)$
(4.) $e_i \nmid (c'''_i x + s_i)$

where the c's, c''s, d's, e's, r's and s's are constants, and the a's and b's are expressions containing no x's (though they may contain other unbound variables).

The next step toward eliminating the quantifier is to eliminate the coefficients of x. Let lcm denote the least common multiple. Let $c = \text{lcm}(\{c_i\} \cup \{c'_i\} \cup \{c''_i\} \cup \{c'''_i\})$, and multiply every relation so that x appears with coefficient c. (Those interested in the complexity of quantifier elimination will note that this operation can greatly increase the size of the formula.) Now our formula $\exists x F(x)$ can be equivalently expressed as $\exists x F'(cx)$. Let $F''(x) := [F'(x) \wedge c|x]$. Then the formula $\exists x F''(x)$ has every coefficient of x equal to one and is equivalent to $\exists x F(x)$.

At this point, every atomic formula in F'' is in one of the following forms:

(1''.) $x < a_i$
(2''.) $b_i < x$
(3''.) $d_i | (x + r_i)$
(4''.) $e_i \nmid (x + s_i)$

Let $\delta = \text{lcm}(\{d_i\} \cup \{e_i\})$. Note that the relations $3''$ and 4'' are invariant mod δ. The point of this observation is that it suggests "searching" for a value of x within a finite range of size δ.

Define a formula $F_{-\infty}(x)$ equal to $F''(x)$ except that atomic formulae of type (1'') are replaced by TRUE and those of type (2'') by FALSE. The intuition is that $F_{-\infty}(x)$ is equivalent to $F''(x)$ whenever x is very small (i.e. close to negative infinity). (We could also define a formula $F_{\infty}(x)$ by reversing the substituted TRUE's and FALSE's; this would make $F_{\infty}(x)$ equivalent to $F''(x)$ for *large* values of x. This can in fact be used to speed up the decision procedure.)

Definition 4.3: Let

$$F^{-\infty} = (\bigvee_{1 \leq j \leq \delta} F_{-\infty}(j)) \vee (\bigvee_{b_i} \bigvee_{1 \leq j \leq \delta} F''(b_i + j)).$$

Note that x has been eliminated!

Theorem 4.1: $F^{-\infty}$ is equivalent to $\exists x F''(x)$.

The proof of this theorem relies on the idea that if there is an x that satisfies F'' it is either very small or it is close to some b_i.

Proof 4.1: First suppose that $F^{-\infty}$ is true (for some particular values given to the unbound variables). We need to show that there is a value of x which satisfies $F''(x)$, with the values of the unbound variables as in $F^{-\infty}$. There are two subcases:

One possibility is that there is a b_i and a j (with $1 \le j \le \delta$) such that $F''(b_i + j)$ is true. In this case, we have a value, namely $b_i + j$, which satisfies $F''(x)$.

The other possibility is that for some value of j, the formula $F_{-\infty}(j)$ is satisfied. In this case, let x = $j - \delta(1 + \max\{|a_i|\})$. We claim that x satisfies F''. Since x is very small, relations of the form (1") are certainly satisfied. Also relations of form (3") and (4") take the same value as they do for x = j (since these relations are invariant mod δ). Now the relations of type (2") were all set false in $F_{-\infty}$; thus, if anything, they are "more true" in $F''(x)$. Therefore, since F'' is monotone (it has only the boolean operators $\wedge$ and $\vee$), it must be that $F''(x)$ is true.

To complete the proof, suppose that for some value a, the formula $F''(a)$ is true (for some fixed values assigned to the unbound variables). We must demonstrate that $F^{-\infty}$ is true. Again there are two cases.

If a = b_i + j for some b_i and $1 \le j \le \delta$, then $F^{-\infty}$ is clearly satisfied.

Suppose otherwise. Consider $F''(a - \delta)$. We claim that it must be true. Assume not. Relations of type (3") and (4") are unaffected by substituting a $- \delta$ for a. Relations of type (1") can only become "more true" by subtracting δ. Thus, if $F''(a - \delta)$ is false, it can only be that some relation of type (2") has turned false in the replacement of a by a $- \delta$. That is, a $- \delta \le b_i < a$ for some i. But that would contradict the non-existence of a satisfying value of the form $b_i + j$. Hence $F''(a - \delta)$ must be true. Thus for all m it is the case that $F''(a - m\delta)$ is true; therefore there is a j (namely a mod δ) such that $F_{-\infty}(j)$ is true. ▌

We have now seen how to eliminate one quantifier. The cost is high (there is a multiplication by a potentially large least common multiple, and a large number of disjunctions are generated). After repeating this process until all quantifiers are eliminated, there is left a sentence containing no variables; this (very long) equivalent sentence can then easily be checked for validity.

This decision procedure for Presburger Arithmetic takes time (and space) $2^{2^{2^{cn}}}$ to check a sentence of length n. The best lower bound known for this problem is $2^{2^{cn}}$ steps on a non-deterministic machine. EM pointed out that these bounds are probably close; however a proof of that fact would be, to say the least,, very interesting.

EM turned discussion to the project assignment. He pointed out that it was necessary to define a suitable subset of Presburger Arithmetic and to establish a syntax for it. In the dehumanizing terminology currently fashionable, this amounts to resolving the issue of the "human interface." To encourage discussion, EM proposed representing sentences by their Gödel numbers.

ST suggested that LISP-like expressions might be more reasonable. RC concurred in this view.

EM promised that a LISP-like syntax (and programs written in LISP) would be acceptable. He argued however that it would be a good idea to defer discussion of the merits of LISP until some higher level questions are answered. These questions include how sets and functions are to be specified. Noting that Presburger sets are all semi-linear , EM mentioned that they might be presented in some kind of normal form (perhaps as unions of linear sets).

After some inconclusive discussion took place, EM urged the class to think about the relative merits of several alternative representations of sets and functions.

RWH wondered whether the intent was to provide full Presburger Arithmetic.

EM said not; that restrictions could be placed on quantifiers. An argument ensued concerning whether or not all variables (which it was agreed are "global", whatever that may mean) should be universally quantified. Eventually the class decided that permitting only universal quantifiers would be inadequate. EM noted one restriction, namely that only finite sets are to be permitted.

EM concluded by suggesting that the class consider the kinds of questions the system ought to be able to answer.

4.4 Notes for Tuesday, February 21

AS began by discussing the "$3n+1$" problem. Let f be the minimal function satisfying

$$f(n) = \begin{cases} 1 & \text{if } n = 1; \\ f(\frac{n}{2}) & \text{if } 2|n; \\ f(3n+1) & \text{otherwise.} \end{cases}$$

(f can be thought of as the function obtained by interpreting the above equation as a recursive procedure.) It is not known whether the minimal such f is a total function. AS had thought that it would be possible to write a Presburger sentence expressing the totality of f. Thus it would be possible to solve the $3n+1$ problem by feeding this sentence to a Presburger theorem prover. However, on reflection, he realized that recursion is **not** expressible in Presburger Arithmetic.

EM pointed out that if one could quantify over sets it would be possible to represent the $3n+1$ problem in Presburger Arithmetic. (One would write a formula describing the minimal set closed under the properties derived from the definition of f.) He claimed that because Presburger Arithmetic is a first-order logic, it is too weak to resolve the $3n+1$ problem. The $3n+1$ problem is properly attributed to Herrn Collatz of Hamburg; however it travels under a variety of aliases including 'Conway's function" and the "Syracuse function." Much effort has been expended on the problem, so far without result. Observing that the $3n + 1$ problem has nothing whatever to do with the matters at hand, EM turned conversation to the question of specifying input sentences.

RC suggested representing sets by unions and cartesian products of intervals. He added that further sets could be defined as the ranges of functions.

EM wanted to hear some of the pros and cons of the representation suggested by RC.

ST politely remarked that since EM had presented sets in the form of unions and products of intervals it would prove easiest to think about them in these terms.

EM replied that the problem statement defines only what kinds of sets are to be represented (in conventional language) and should not be interpreted as an endorsement of any particular representation scheme. He attempted to clarify the issue by asking "What are numbers?" and "Do numbers exist if you don't write them down?" As the class fidgeted nervously, EM allowed as to how he might be catching the flu. Then he explained that one way to represent numbers is to use radix notation. Another way is to specify their prime factorizations. The former method makes addition relatively simple, while the later facilitates multiplication. This is an example of the kinds of trade-offs one has to consider when deciding upon representations for objects.

RC asked how one would write down the primes in the second scheme. EM replied that one wouldn't; one would simply write down their exponents (perhaps in radix notation).

ST suggested that before choosing a representation it would be helpful to have more sentence-writing experience.

EM tried to explain his intent in presenting the example of representing numbers. If he plans on adding lots of numbers, the radix representation will work well. If he plans on multiplying lots of numbers, the prime decomposition will be effective. (He mentioned that no representation is particularly good when both operations are to be performed.) It may even be worthwhile to convert from one representation to another (or to change radix) in certain circumstances.

RWH mentioned that he had considered *semi-linear* sets, but that it was not clear how to use them to represent intervals. Some members of the class also feared that allowing intersections of unions of intervals might result in a "blow up" (*i.e.* a huge number of intervals).

EM agreed but suggested that that price might have to be paid no matter what representation is chosen. There was some discussion of how to represent intervals. This prompted ANS to ask what primitive sets were under consideration (his point being that if intervals are taken to be primitive then there is no need for further discussion of how to represent them).

EM noted that the problem statement defines allowable sets by inductive construction. Such a construction is one way to represent a set; another way is as a union of intervals.

RC wondered whether the matter under discussion was representation inside the computer. EM thought that it was too early to discuss internal representation; he felt that the issue at hand was analogous to the question of representing numbers. In particular he noted that two extremes on the spectrum of representation had been presented: on one end sets are represented as unions of intervals (disjunctive normal form), on the other, as general expressions involving arbitrary unions and intersections of intervals. EM urged discussing the relative merits of such alternatives.

RC observed that in disjunctive normal form many tests (such as set inclusion and cardinality) are easy to perform.

AS argued that it isn't so easy to compute, for instance, cardinality, when the intervals fail to be disjoint. RC replied that he had in mind only disjoint intervals. To this, AS responded that it might not be easy to ensure disjointness, particularly in the presence of lots of global variables.

EM clarified the issue with the following example. Suppose that $A = \{1, \ldots, n\}$ and that $B = \{1, \ldots, m\}$. Then the question of how to represent $A \cup B$ (or $A \cap B$) arises. The disjunctive normal representation depends on the relative magnitudes of m and n.

GP observed that since sets can be represented in Presburger Arithmetic, and since at least most questions about them can be phrased in Presburger Arithmetic, it might suffice to simply write a decision procedure for Presburger Arithmetic. He admitted however that he didn't know how to express cardinality in Presburger Arithmetic.

EM replied that cardinality cannot be expressed. His reasoning was that the combination of cartesian products and cardinality provides multiplication, but that Presburger Arithmetic is too weak to talk about multiplication. In any case, he felt that GP's idea merely trades one problem for an equally difficult one. That for some reason reminded EM of a joke about boiling water. Unfortunately he had forgotten the funny part.

EM recapitulated: with the disjoint union representation the basic operations are easy, but constructing the sets may be difficult.

SS and AAS argued that it might be extremely difficult to construct the sets. ANS, referring to the $A \cap B$ example, pointed out that in the first place the system wouldn't know anything about m and n; and that even if it did, it wouldn't know what to do with the information. In support of his second point he observed that the system would need to "know" about such matters as transitivity. In fact, he claimed, things could get arbitrarily complex.

SS back-pedalled a bit by questioning whether things were quite so bad. AAS held firm and suggested that modular arithmetic would further complicate the situation (quite possibly beyond repair).

EM barely had time to express some optimism before AS proposed that at least as a start, one might tell the system about all relations among the global variables (even ones that are derivable from other facts).

EM asked whether given non-empty intervals $[A, B]$ and $[B, C]$ one could prove in Presburger Arithmetic that $[C, A]$ is empty. AAS answered "yes," that one could show that if $A \neq C$, then exactly one of the intervals $[A, C]$ or $[C, A]$ is empty. Since $B \in [A, C]$ it follows that $[C, A]$ is empty. EM agreed that the fact that exactly one of the intervals is empty could be expressed in Presburger Arithmetic; but the question of where this information would come from still remained.

ANS suggested that the user of the system should provide all needed relations among the global variables. EM proposed two alternatives: the system could ask questions when it needed more information (*e.g.* it could ask "Is $m < n$?") or it could build a tree of cases where the children of each node correspond to the possible outcomes of comparing global variables.

AAS objected that it might be impossible to know which comparisons need be made in the absence of a general "deduction" system. EM disagreed, ANS agreed. Perhaps alluding to his aborted joke, EM then claimed that everything boiled down to the $<$ relation. He said that it might be necessary to accept cumbersome answers from the system, such as

$$|A \cap B| = \begin{cases} n - m & \text{if } n > m; \\ m - n & \text{if } m \geq n. \end{cases}$$

AAS wanted to know what happens when modular arithmetic enters. He wondered whether the system would need much knowledge about primes, *etc.* RWH suggested that a few facts about greatest common divisors and related subjects would suffice. EM said that a trick could be employed to eliminate the modular arithmetic. To put it briefly, it is possible to "change variables" (in the calculus sense) and rewrite formulas by introducing a clause to enforce the eliminated congruence constraints.

Since no alternative schemes were proposed, EM announced that he would represent sets as unions of disjoint intervals. He noted that there are several potential syntaxes for sets defined with congruence relations. One possibility is "$x \in [lb \ldots ub]$ such that $x \equiv b \pmod a$", another is "$\{an + b \mid n \in [lb \ldots ub]\}$".

AS asked whether EM would allow multiple congruences. EM refused to say; he noted that there are many possibilities including intersections, Chinese remaindering, etc.

EM requested that everyone examine a couple more representations and think about how they affect the difficulty of performing the basic operations.

AAS stated that intersecting a bunch of sets defined with congruence relations could result in an arbitrarily complicated representation. EM took exception to the word "arbitrarily"; AAS agreed that the complexity of the result would depend on the number of global variables involved. EM suggested writing a program that would always work and not to worry too much about its time and space requirements. He noted though that it would be vital to decide how the system would deal with inter-relationships among the global variables. There are a few very basic operations, perhaps just the intersection of sets. In order to find the representation of an intersection of sets, the system might have to ask a question, try to prove a (universally quantified) sentence, or construct a set of cases. One has to decide which, if any, of these methods to implement.

SR observed that the system might return unnecessarily complex expressions (for example, humongous formulas that are always true). He noted that this problem also arises with the resolution method.

EM agreed that the problem is probably unavoidable.

4.5 Notes for Thursday, February 23

The student groups for Problem 3 are as follows: [AS, ANS, AAS], [MGB, RWH, ST], [GP, RC, SR], [CWC, SS].

EM asked each group to report on its progress.

RC announced that his group had been thinking about using unions of intervals to represent sets. He felt that it would be difficult and unnecessary to require that the intervals be disjoint.

MGB disputed the difficulty of ensuring disjointness; she pointed to the possibility that the system could ask the user for information relating the global variables to each other. RC replied that it wasn't just a matter of comparing one variable to another; also needed are congruence relations. He illustrated this point with the following problem: compute the cardinality of the set $\{x \in [l \ldots u] \mid x \equiv 0 \pmod 2\}$. Here it is necessary to know not only whether $l < u$, but also the values l mod 2 and u mod 2. AAS argued that $u - l$ mod 2 would suffice.

AS noted that much difficulty stems from lack of knowledge about the global variables. He wondered what options were available for getting information about them.

EM wrote down two sets defined in terms of global variables m and n, and a putatively bijective function between them. He asked how the system should check that the function is indeed bijective. It was noted that this could be done by attempting to validate a Presburger sentence universally quantified over m and n. EM then asked whether a system which only allowed universal quantification would be interesting. (He noted that this is not a mathematical question.)

RC returned to the topic of set representation. He suggested that intersections of basic sets could serve in place of unions; but that that appeared even more difficult to implement.

SR proposed having the system build a tree of cases to deal with the various possible relations among the global variables. He noted that the system may ask what turn out to be irrelevant questions; thus the answers may be pointlessly complex. He suggested that the system might do some simplification. In general however, the number of leaves will be exponential in the number of global variables.

EM observed that building a tree of cases is equivalent to using the system interactively. He asked whether it would be possible to automatically collapse cases that can be combined. RC replied that it would be possible, at least in theory, to write Presburger sentences expressing equivalence of cases and then to have these sentences checked for validity. EM agreed, but noted that it was a matter of guessing which cases might be combinable. RC asked whether it was necessary to implement simplification. EM replied that it wasn't; he merely wished to point out that the simplification so far described did not leave the realm of Presburger Arithmetic. He added that the notion of "simplest" fails to have a satisfactory definition.

RWH pointed out that many cases are generated when two big unions of intervals are combined. The representation of each resulting cross term may depend upon a new relationship between global variables. EM thought that usually most cases would drop out. However, in the worst case, disjunctive normal representation causes exponential blow-up.

ST observed that even the cases that go away need to be looked at. RC suggested asking the user which cases are interesting in order to cut down the amount of work. EM felt that, by analogy to experience with expert systems, exponential blow-up would not be a problem. Usually most cases will collapse; thus he did not think it desirable to ask bothersome questions about the user's interests.

CWC and EM noted that it would be necessary to obtain information such as $3m < 2n + 5r$ as well as simpler relations such as $m < n$. CWC resurrected the idea of allowing only universal quantifiers. EM agreed that this might suffice; he also mentioned that most languages do not support sets because of the inherent difficulties.

CWC asked for a clarification of the permitted uses of cartesian products. EM observed that it is not possible to define a bijection between an n-by-n square and an interval of length n^2. In response to a question from CWC he stated that elements of the cartesian product $[I_1 \times I_2] \times [I_3 \times I_4]$ are of the form (i, j, k, l). EM claimed that cartesian products only add quantitatively to the language, not qualitatively. This would not be the case if such definitions as $\{(i, j) \mid i \leq j \wedge 1 \leq j < n\}$ were allowed.

AAS wondered if it wouldn't be possible to define the above triangular set as the range of a function defined on the n-by-n cartesian product. EM replied that it was not allowed to define sets in terms of functions. RC observed that the constructible multidimensional sets are hypercubes, possibly with holes resulting from congruence constraints. AAS and EM observed that if S is a set and F a function in our language, then it is not necessarily possible to define the set $F(S)$. Indeed, to check whether F is one-to-one, the range of F must be embedded in a possibly larger cube. This may preclude checking whether a function F is onto.

EM explained that he had added cartesian products so that cardinalities would be polynomials in the global variables. Without cartesian products, all expressions would be linear in the variables and this would be boring. Thus cartesian products introduce non-linear expressions without departing from Presburger Arithmetic.

AAS regarded as potential trouble the possibility that identical sets could have different disjunctive normal representations. EM noted that a Presburger sentence could be written to test whether two sets are the same. RC objected that there would be no easy way to check the sentence. EM noted that it might be expensive to check the sentence but that it would nevertheless be possible.

EM concluded by observing that many problems have been successfully programmed despite the fact that in the worst case, the running times may be indistinguishable from infinity. For example, the simplex method can take exponentially long. However that never happens in practice, and the simplex algorithm is very widely used. (A recent result by Smale shows that the simplex method takes "on the average" only a linear number of iterations, and hence quadratic time.)

4.6 Projects

RC and SR implemented a decision procedure for Presburger Arithmetic. Their LISP program is based upon the method of quantifier elimination. Because they intend their procedure to be driven by a user interface program rather than by the user directly, they have chosen a particularly simple syntax for Presburger sentences. The relation (DIV c e) is TRUE if and only if the expression e is divisible by the constant c. The relation (NDIV c e) is true if and only if e is not divisible by c. The final basic relation (POS e) is true if and only if e is greater than zero. Expressions are linear combinations of variables; (EXP (5) (3.X) (-5-Y))represents the expression $5 + 3x - 5y$. Sentences are formed (in prefix notation) from the basic relations, expressions, boolean functions such as $\wedge$ and $\vee$, and existential and universal quantifiers.

ST also implemented a quantifier-elimination decision procedure for Presburger Arithmetic. Both ST and the group of RC and SR considered the problems inherent in representing and manipulating sets. They agreed that there is no ideal representation for sets. Either the union operation or the intersection operation, or both, will be complicated to program and expensive to execute, as will be the operation of determining set cardinality. Furthermore, the expressions resulting from the set operations may become complicated due to their dependence upon ordering relations among the global variables. Unfortunately neither ST nor RC and SR had time to implement their set-representation ideas. Nevertheless each group felt that with an appropriate user interface their Presburger Arithmetic decision program could serve as the core of a verification system.

5. Graphics

5.1 Problem Statement

The purpose of this problem is to develop data structures and algorithms that allow the representation of (wire models of) three-dimensional objects on a SUN terminal. Such objects could be spheres (given by wires along their latitudinals and longitudinals), or (ordinary) bottles, or three-dimensional cross-sections through 4-dimensional bottles (or more general bodies). The implementation should be chosen such that it is possible for the user to move (within bounds) his/her standpoint and distance to the object.

5.2 Notes for Tuesday, February 28

The class started discussing problem 4. The basic problem is to draw nice pictures on a SUN terminal that can be manipulated interactively. The first questions that were discussed were what types of objects should be drawn and what operations should be applied to them. EM suggested that the class concentrate on wire frame models. In this model objects are represented as collections of polygons and the boundaries of the polygons are displayed. There are quite a few features that could be considered such as shadows, illumination, shading, and coloring the polygons with patterns. Another option would be to remove hidden lines from the objects. The wires could be straight lines or else they could be curves such as conic sections or splines. A possible restriction on the objects is that they could be required to be convex. RWH pointed out that hidden line removal is much easier for convex objects. The operations on the objects are to allow them to be viewed from various angles and distances. The idea is that the object remains fixed while the viewer moves. The basic operations would be such things as rotations and zooming. One goal would be to make the movement as close to being continuous as the hardware and software constraints allow. To make the transformations fast the operations should be incremental in some manner.

A basic concept in computer graphics is the distinction between world coordinates and view coordinates. World coordinates are the coordinates with which the object is modelled. If the object is stationary then the world coordinates will remain fixed. View coordinates are the coordinates as seen from a particular location. The view coordinates will change when the object is viewed from a different location. Screen coordinates are the coordinates that an object is displayed with. The screen coordinates will be a projection

of the view coordinates. In this problem both the world and view coordinates will be three dimensional, while the screen coordinates are two dimensional.

SS pointed out that the problems of transformation of graphical objects had been well studied and were fairly easy using 4 x 4 transformation matrices. These matrices allow the view to be changed and perspective to be represented. Three dimensional objects are represented using homogeneous coordinates. Each point is represented by a 4-tuple (x, y, z, w). The first three coordinates correspond to the normal coordinates and the last one is a scaling factor. If $w \neq 0$, then the coordinates are normalized by dividing each entry by w so that the last entry is 1. Tuples denote the same point if they have the same value when normalized. The operations of scaling, translation, and rotation can all be expressed by matrix multiplication. The matrix for translation by Dx, Dy, Dz is

$$T(Dx, Dy, Dz) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{pmatrix}.$$

For example translating the point $(3, 4, 5)$ by $(1, -4, 3)$ is accomplished by:

$$(3 \quad 4 \quad 5 \quad 1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & -4 & 3 & 1 \end{pmatrix} = (4 \quad 0 \quad 8 \quad 1).$$

The scaling matrix for scaling by factors of Sx, Sy, Sz is

$$S(Sx, Sy, Sz) = \begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A rotation matrix has a 3 x 3 orthogonal submatrix. The matrix for rotation around the x-axis by θ is

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The rotation matrices for the other axes are similar.

Projection onto a viewing plane can also be expressed as an 4 x 4 matrix. If the object is being viewed from the origin and the screen is the plane parallel to the x-y plane at $z = d$, then the perspective projection matrix is

$$M^{per} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

A discussion of the use of homogeneous coordinates in computer graphics may be found in Fundamentals of Interactive Graphics by Foley and Van Dam.

EM said that an important aspect of this problem is the resolution problem. If objects are viewed from a long way away, small objects will be too small to display. If an object is so small that it is only one pixel and it is rotated for example, there is no point to compute how its faces are changed. Similarly if an object is viewed from very close, only part of it will be visible. The resolution problem is how to restrict the computation so that unnecessary computation is not done.

EM outlined one possible method for dealing with this problem that uses several levels of resolution. A naive approach would be to compute how the object appeared from distances at logarithmic intervals. For example a representation would be computed for how the objected appeared when viewed from 1 meter, 10 meters and 100 meters. If the object is viewed from 50 meters, the 10 meter representation is used. There is some overhead in this method and there are some problems such as what levels to compute.

The objects are to be displayed on the SUN terminal. The graphics routines are part of the VGTS system. To understand the graphics package there are two basic concepts. The first is the *structured display file* or *sdf.* All of the graphical objects are put into the sdf with their world coordinates. The coordinates for objects in the sdf are two dimensional coordinates. This is an intermediate level between view coordinates and the graphical display. The basic objects are lines, points, text and maybe splines. Composite objects are created by making a symbol. A symbol is essentially a list of other symbols and basic objects. Into an sdf there is a view which is what is displayed. A view only displays a portion of the sdf. Both the sdf and the view must be manipulated. The commands are given in the V system manuals.

The VGTS software allows programs to be run on either the SUN or a VAX. The advantage to running on a VAX is that it has hardware floating point arithmetic while the SUN does floating point with software. This is possibly a major consideration. On the other hand, communication from a VAX to the SUN is over the ethernet which might slow things down. This will probably be a significant problem if the SUN and the VAX are on different ethernets and the communication must go through a gateway.

EM brought up some specific data structures that might be useful. One feature the data structure should have is that it helps in identifying all objects that are close together. A good data structure would be traversed until the objects got too small for the resolution of the display. A type of structure that meets these criteria is a geometric tree. There are a number of variants of geometric trees which are attempts to generalize binary trees. A quad tree is a tree with each node of degree 4. Each node has a coordinate and each child is a quad tree which corresponds to the area northeast, northwest, southwest, or southeast of the coordinate. Another approach is a *k-d* tree. Each level splits the points in a rectangle in half in either the x or y direction with different directions being used for alternate levels. AAS pointed out that one problem with these two approaches is that the trees can become unbalanced. AAS suggested cell techniques instead. The points would be put into rectangular cells. The cells may be of different sizes. They are sorted by y-coordinate and x-coordinate. To add a new point, two binary searches must be done.

This representation would be useful since points that are close together will either be in the same or neighboring cells.

5.3 Notes for Thursday, March 1

EM began the class by giving two references for computer graphics. The first reference was W. M. Newman and R. F. Sproull; ***Principles of Interactive Computer Graphics,*** **2nd** edition, McGraw-Hill, 1979. This is the standard text book for computer graphics and contains lots of useful information. The other reference was L. E. Sutherland, R. F. Sproull, and R. A. Schumacker: "A Characterization of Ten Hidden-Surface Algorithms," ***Comput. Surv.,*** **6**(1):1, March 1974. This paper summarizes a number of different methods for removing lines and surfaces that are obstructed by other objects.

ANS recommended that it would be best to start with a straightforward implementation and postpone the complex parts until later. The first step would be to construct some objects and transformation matrices and display them on the screen. The object would be constructed with fixed world coordinates. A simple outer loop could call the various transformations. The problem of resolution would not be addressed until after this first part was running. A sophisticated user interface could also be added later. EM suggested that the outer loop could contain a trajectory for the viewer. This would be a compiled procedure that computed a parameterized curve. One possibility would be to have the viewer approach the object along a parabola. EM asked the class how difficult they thought it would be to convert a trajectory to a collection of transformation matrices. The consensus was that this probably wouldn't be too difficult.

MGB had checked up on the spline package for the VGTS. MGB had found out that the spline package did exist, but unfortunately the author of the package had graduated and left no documentation on it. This provoked a general discussion on the politics of software that need not be reported here. AS had tried to figure out how the splines worked by looking at the code and the calling sequences. He found that the code was incredibly dirty and confusing but was able to decipher some of it. The conjecture was that the order of the spline and the interpolation points would be supplied by the user. Two functions $x = x(s)$ and $y = y(s)$ would be computed. The functions would probably be low degree polynomials, probably allowing splines of order 2 through 5. EM gave a brief summary of spline curves. Splines originated by considering the problem of how a curve would appear if it had a few fixed pivots that it had to pass through but otherwise it would take the shape that minimized its potential energy. The curves formed this way are exponential splines. The solutions turn out to be elliptic integrals which are very difficult to compute numerically and the methods used are not particularly stable. These curves are quite pleasing to the eye and are of considerable practical significance in certain applications. They are not used for interactive graphics since they are so difficult to compute. The main problem with splines for this problem is that when the view changes the curvature of the spline would also change. An alternative to using splines would be to just approximate curves with li ne segments.

EM brought up the subject of what data structures should be used for the project. SR presented a cell data structure that would avoid having to display objects that were too close together. The basic objects would be described in terms of points, lines, and polygons. The viewer would be assumed to be fixed at the origin and the objects would be moved. The screen would be considered to be at a fixed location. The basic idea would be to pass objects through a filter. Only the objects that made it through the filter would have to be displayed, so this could cut down the work in displaying things substantially. The screen would be subdivided into a number of cells. The working figure was to have the screen subdivided into 80 x 100 cells. All points would be projected onto the screen. The points that were not on the screen would not be put into the data structure. Each point would be projected into a cell. The cell would have all the points contained in the pyramid with apex the origin and sides passing through the boundaries of the cell. Each cell would have a count of the number of objects in it. When the number of points in a bucket exceeded a certain threshold, some of the points would not be displayed. The cells would also provide information to reduce the number of lines that had to be drawn. When an object is moved, only the cells it is in need to be altered. The data structure is applicable if there are a number of objects which can be moved independently. If the view is changed, then everything is moved, so the entire data structure must be recomputed.

EM brought up the relation of the sdf coordinates to the other coordinates that would be worked with. At the top level, the object would be represented with three dimensional world coordinates. These would be translated into two dimensional view coordinates. Putting objects into the sdf would convert their view coordinates into sdf world coordinates. The two dimensional sdf world coordinates are finally transformed into screen coordinates. The transformation would allow translation and primitive scaling. The latter transformations would be handled by the VGTS. RC pointed out that the sdf must be recreated when display is changed. The question was raised as to how much was involved in recreating the sdf. The new sdf would have the same structure as the old one, only the coordinates would have been changed. The VGTS does have a change command for the sdf which is equivalent to deleting and then inserting. Whether the sdf is recreated or just modified depends upon the implementation of the VGTS.

EM said that there were two different approaches to the resolution problem. In the approach taken by SR, objects would be collapsed that would be displayed too close together on the screen. These objects could be very far apart and still collapse. One drawback to this approach is that objects collapsing depends on the angle that they are viewed from and not just on their distance. The other approach is only to attempt to merge points that are close together in the real world. The world can be viewed in terms of wire figures, which are collections of lines and vertices. When objects are collapsed, they would form super vertices. This would create a hierarchical graph, where nodes can expand to form subgraphs when viewed from not too far away. Varying levels of resolution could be achieved by expanding a different number of nodes. The decision to expand nodes could be just based on the distance from the viewer and not depend on the angle of the view.

5.4 Notes for Tuesday, March 6

RWH had his program for problem 4 running. His program displayed a pyramid and allowed it to be rotated around the x, y, or z axis. The surfaces of the object that were not visible were not displayed. The main loop of the program accepted commands which specified the axis of rotation and the number of degrees to rotate. The object is displayed at 1 degree increments. RWH had attempted to have perspective as well, but it broke when the hidden surface removal was added. To handle rotations around other lines, RWH had attempted to have the one degree rotations input from a file. For example, if the inputs were $x, y, x, y, \ldots$ the rotation should appear to be around the line $x = y$. Unfortunately for some mysterious reason this approach did not work, the SUN encountered synchronization problems.

RWH described the data structures that he was using in considerable detail. At each step of the display, he computed a 3 x 3 rotation matrix and applied it to each point. One of the important aspects of the data structure was that the transformation was applied once to each point and not once to every line, so that points that were adjacent to several edges were only transformed once. The hidden surface removal was handled by displaying only faces of the object that were visible. The object was assumed to be convex, so that faces would not partially obscure other faces or lines. It was also assumed that only one object was present. The data structure that was used made it easy to identify which points, lines, and edges were visible. The data structure was essentially a collection of arrays with pointers associating various objects. The world coordinates of all the points are stored in an array. These are not changed and are created when the program is started. There is also an array of the transformed points, which give the view coordinates after a transformation is applied. Lines are stored by storing pointers to the two endpoints, these are just indices into the point array. A polygon is a list of edges. These edges are stored in an array with an endmarker indicating the last edge. The polygon stores an index into an array that contains all of the edges, the index gives the first edge of the polygon. The polygon also stores its normal vector. When the normal vector is transformed by a rotation, it is easy to tell if the polygon is visible. If the normal is pointing towards the viewer then it is visible, and it is not otherwise. This just requires checking one coordinate of the normal. If perspective was added, the computation would have to compare the angle of the normal with the angle between the viewer and the face. Once the visible faces have been found, the data structure is traversed to find the visible edges and points. When a face is visible, all of its associated lines and points are visible. Boolean arrays are used to keep track of which objects are visible. When all of the visible objects have been found, they can be drawn on the screen.

RWH had optimized his program in a number of ways. He was running it entirely on the SUN so as to avoid problems of communication over a network. Since the SUN doesn't have hardware floating point, he was doing all of his computations using scaled integer arithmetic. He chose a scaling factor of 1024 so that scaling could be done by left shifting by 10 bits. Using scaled arithmetic meant that whenever he did a multiplication he had to rescale, so to compute $a = b * c$ it is necessary to do $a = (b * c) >> 10$ where $>> 10$ is

C syntax for right shift by 10 bits. The sines and cosines are precomputed at one degree increments and stored in arrays sin[0 . . 359] and cos[0 . . 359].

AS had looked into the spline package and had figured out how they were used. He had looked at a draw program and had discovered that certain default parameters had to be reset to get the splines to work. The splines are from orders two to five and can be either open or closed. The closed splines may also be filled with a pattern. The second order splines give a way of constructing polygons. The higher order splines have the feature that the tangent at the endpoints is in the direction of the next point. This means that the curves often come out looking rather strange. AS said that he was able to generate nice pictures, but he was not able to generate the pictures that he was trying to get.

AS described the approach that his group was going to take on the problem. His basic model was the same as RWH had, with the world consisting of points, lines, and polygons. AS planned to allow several objects, but required them all to be convex. AS defined the bounding box around the object to be the smallest rectangular box oriented parallel to the axes that contained it. For his hidden surface removal algorithm to work, he assumed that all of the bounding boxes were disjoint (in 3-space). The basic hidden surface algorithm was to sort the boxes according to whether the box was in front or behind other boxes. The boxes would be ordered so that the furthest away from the viewer came first. The contents of the boxes would then be displayed in order. Filled polygons will be used to draw the object, so that when one is drawn on top of another, the obstructed edges are erased. A paper that describes various conditions other than having disjoint bounding boxes, for this algorithm to work properly is F. F. Yao, "On the priority approach to hidden-surface algorithms", Proc. 21st Symp. on FOCS, (1980), pp. 301–307. AS planned to allow the viewer to follow an arbitrary trajectory as long as the viewer did not enter any of the bounding boxes. If the viewer entered a bounding box, then the results would not be guaranteed, most likely a core dump or some other disaster would occur. The testing of which boxes obstructed the others is an easy computation since it just involves computing the projection of points onto surfaces. EM asked about what techniques could be used to speed this up taking advantage of the incremental nature of the trajectory. It should be possible to save quite a bit of the computation by being able to anticipate when objects will start and end obstructing one another.

5.5 Notes for Thursday, March 8

Class began with a brief discussion concerning Problem 4. RWH noted that displaying curved objects is more difficult than displaying flat objects. This is due to the existence of horizons. The matter arises for example in the display of a globe.

With regard to displaying the globe, EM mentioned the possibility of subdividing along lines of latitude and longitude, perhaps every ten degrees. Each of the resulting pieces could be rendered as a four sided polygon. EM wondered how long it would take to display them all.

RWH suggested that a region such as the United States could be cut into a number of polygons by introducing new edges. By applying an appropriate rule governing whether such edges are displayed, it might be possible to easily approximate a horizon. The idea would be to suppress the display of an introduced edge unless one of its adjacent faces falls below the horizon.

EM seemed a bit dubious about the merits of this method of approximating the effect of a horizon. He noted that a current Texas Instruments computer demonstration shows a rotating globe. The globe is represented by a grid of longitude and latitude lines.

AS pointed out that such a grid has singularities at the north and south poles. He suggested using something else, for instance a hexagonal tiling, that eliminates singularities. RWH suggested using large regular polyhedra (e.g. dodecahedra) and triangulating the faces.

5.6 Projects

While working on the problem, several groups of students outlined ambitious plans for programs. However, due to time constraints, only one group ({ RWH, MB, ST }) completed a program. Their program displayed a two-dimensional projection of a three-dimensional object on the screen. The user could move around the object to look at it from different views. The movement was done in real time, and appeared continuous. The object that was used was a pyramid, although it could be changed by creating a new initialization file. Two different versions of the program were created, one provided a perspective projection, and the other provided hidden surface removal. The program put major emphasis on efficiency. Such techniques as evaluating trigonometric functions by table lookup were used so that the program would be fast enough to give reasonable real time motion.

6. Parallel Computation Bottlenecks

6.1 Problem Statement

This last problem concerns GORC, the "Game of the Research Community" (though we don't want to take too seriously the model presented here; the author doesn't seem to be able to come up with any realistic, attractive games anyway, since otherwise that would be what he is doing). GORC is played by $n \geq 2$ players, but it involves $n + 2$ parties. The n players represent young, hopeful computer scientists who set out to prove $\mathcal{P} = \mathcal{NP}$. The $n + 1^{\text{st}}$ party is the prestigious **Journal on** *Unproven Knowledge* or **J.UNK**, in which, contrary to what its title may seem to suggest, only scientifically sound and verified results of research into the problems of mathematical and computer knowledge are published. For the n young and aspiring scientists, it is basically the only place to publish the results of their work in order to establish a reputation in the hope (albeit slim) to ever get tenure.

The last participant in GORC is an *oracle*. At the beginning of the game, the oracle sets up the complete theory of the field on which our researchers intend to work. Of course, initially this theory is completely unknown to the players of the game. The theory (it is an average theory) can be envisioned, for the purpose of our game, as a randomly generated tree with about 10,000 nodes of out-degree at most two, with the leaves at a depth between 10 and 20. The nodes of the tree represent conclusive results of research. About 99% of them represent the realization of *failure*, and about one per cent represent *Theorems* (it should be noted here that these numbers seem to be completely arbitrary!). The theorems tend to occur clustered in small subtrees at the bottom fringe of the tree, or in long, skinny chains.

Let's take a closer look at the Theorems, and let's assume without loss of generality that they are called $T_1, T_2, \ldots$, until about T_{100}. These Theorems are of varying significance. In our case, concerned with the $\mathcal{P} = \mathcal{NP}$ problem, the first 90 theorems are probably results that imply that out of the 937 presently open scheduling problems, 884 are in fact NP-complete (using a number of technically very involved reductions from some so far rather neglected variants of SAT, namely m-CNF-SAT for several $m > 13$). Obviously, there must be some publications in **J.UNK** on the same result, just judging by the numbers. It should be emphasized, however, that such things only happen by accident, and that the editorial board of **J.UNK** takes great care to avoid any duplications. Theorem T_{95} could be a theorem that proves that there is an $O(n^{17})$ algorithm to factor numbers (this paper, incidentally, had been rejected by two referees who also happened to be working on cryptology), and T_{100} would be a theorem with a short statement (namely "$\mathcal{P} = \mathcal{NP}$"), together with a short proof as we should have expected (again incidentally, this proof consists of 24 easy Lemmata, almost all of them previously published in **J.UNK**).

Now to the details of how GORC is played. We already mentioned that at the beginning, the oracle sets up the tree of the complete theory of the field under investigation by the participating scientists. The scientists work on *projects* which are determined by two nodes in the tree. The first node represents the proposal for the project, the second node (which must be a descendant of the first) the current working hypothesis. The proposals are located relatively high in the tree, and they are, at the beginning of the game, determined at random by the oracle. The project itself is the set of nodes on the path between the proposal and the current node.

The oracle maintains the following *local* information about the nodes of the tree:

(i) for every node of the tree, its outdegree;

(ii) for every node of the tree, whether it represents a failure or a Theorem, and in the latter case a unique identifier for the Theorem.

On the other hand, **J.UNK** represents a *global, central* database with the following pieces of information:

(iii) for every Theorem in the theory, whether it has been submitted to the journal for publication;

(iv) for every subfield of the theory (represented by the subtrees rooted at the nodes of the tree) a bit indicating whether that subfield has been explored completely (*i.e.*, whether every node in the subtree has been visited by at least one scientist);

(v) for every node of the tree, whether it is currently contained in a project.

The game proceeds in steps, each step representing one day (with, it should be understood for aspiring young researchers, seven days per week). At every step, every scientist can do one of the following:

(i) He can start a new project. In this case, the node for the proposal is determined randomly by the oracle. The scientist obtains the local information about this node.

(ii) He can move to a direct descendant of the current node. This node then becomes the current node, and he is told the local information available for it.

(iii) He can back up to the predecessor of the current node. This move works analogous to (ii).

(iv) If he has found a Theorem in the previous step, he can submit it to **J.UNK**. It is accepted for publication, if it hasn't been submitted before, otherwise it is rejected.

(v) He can inquire at the Editorial Office of **J.UNK** about the global information on a given node in the tree.

The Editorial Office of **J.UNK** works such that incoming requests (submissions or inquiries) are completed one per day. Requests arriving during the same day are put in some random order and queued.

The purpose of this problem is to develop methods which allow the scientists to discover, in a limited amount of time, say 1,000 steps, as many Theorems as possible.

Remark: If, as a possible idea for a solution, you should decide to distribute the global database, i.e., found a number of new journals (with you, of course, on the Editorial Board), stick to the spirit of this problem of one message per step for every "real" actor. The oracle, and its interface, will be implemented by the author.

6.2 Notes for Thursday, March 8

EM provided an introduction to Problem 5. The problem is motivated by issues that arise in parallel processing. The goal of parallel processing research is to provide a computational environment in which many agents can work in efficient cooperation on a single problem. This is of course in distinction to the problem of providing computational resources to a number of independent users. The latter problem can be solved, for example, by building greater numbers of conventional machines. The former problem is apparently much more difficult; particularly if the goal is to provide an architecture which permits (at reasonable cost) a high (and scaleable) degree of parallelism. Difficulty arises whenever it is necessary to share information among processes; for sharing generally leads to bottlenecks. There are two extreme solutions to the problem, both usually unsatisfactory. One solution is to concentrate information at certain locations and to require all interested processes to query those locations. Of course, if many processes are interested, a bottleneck develops. At the other extreme, information can be broadcast by its producer to all other processes. This unfortunately results in what is known as the "junk mail" problem.

EM asserted that in this issue lurks a question of reasonable compromise. He illustrated that claim by constructing an analogy to research. A researcher might choose to visit the library every day, or might choose to visit once a month. The once-a-day schedule consumes a great deal of time, but reduces the risk inherent in the once-a-month schedule of wasting prodigious effort on already solved problems.

Returning to the computational motivation, EM described the traveling salesperson problem. The input is a graph whose edges bear weights (possibly representing distances). The problem is to find a simple tour that visits each vertex and that has minimal total weight. This problem is known to be NP-complete and thus there is unlikely to exist a polynomial time algorithm for solving it.

There are however some heuristics which find good, but sub-optimal, tours. One such class of heuristics is based on local optimization. The idea here is to find any tour (which it should be noted can be difficult for certain types of graphs) and then to make iterative, local, improvements. For example, at least for complete graphs, one can consider the effect of replacing any pair (a, b), (c, d) of edges in the so far best tour by the edges (a, c), (d, b). Notice that the resulting path is also a simple tour. If that tour has smaller total weight, it is used as the basis of the next iteration; otherwise it is forgotten. Of course one could also consider more complicated local changes. In any case, the process is repeated until no local change reduces the weight of the tour. Thus, the process is terminated when it reaches a local (and hopefully global) minimum.

ST asked whether it couldn't be proved that no heuristic could perform well. EM replied that that depends on what is meant by good. He outlined a method based on minimum spanning trees. This method, which works when the graph satisfies the triangle inequality, is guaranteed to achieve a tour with weight at most twice optimal. (A refinement due to Christofides guarantees a tour of weight at most 3/2 optimal.) The first step is to find a minimal spanning tree of the graph. By using each edge of the tree edges at most twice, a (non-simple) tour of the graph can be found. This tour has weight at most twice optimal (since any tour is a spanning tree). By "taking shortcuts" to avoid traversing edges more than once, the tour can be made simple. Because of the triangle inequality, this simplification cannot increase the total weight. These heuristics are described in "A Case Study in Applied Algorithm Design" by Jon Bentley (Computer Vol.17, 2 (Febr. 1984), pp.75–88).

Returning to the topic of local optimization heuristics, EM observed that the currently popular method of *simulated annealing* can be applied. The advantage of this technique is that it stands a chance of escaping from local minima; thus it has better odds of reaching a better, if not global, minimum.

EM discussed the problem of parallel local optimization. Suppose that a number of processors are attempting local improvements on the tour and that the best tour yet discovered is stored in global memory. The question then arises of how often each processor should examine the globally stored tour. If the processors check frequently, a lot of time is wasted in the resulting bottleneck. If they check rarely, they will often be working on tours worse than one already known and will thereby waste time. Thus there is a trade off here between communication and computation. This trade off is the subject of "A Study in Parallel Computation — the Traveling Salesman Problem" by J. Mohan (Tech. ReportCMU-CS-82-136(August1982)) and the motivation for Problem 5. Programs for Problem 5 should permit experimentation with different policies. They should also permit "changing the system" by distributing computation in various ways (*e.g.* in the language of Problem 5, by founding new journals).

SR asked what an "average theory" is. EM replied that it is a tree of ten thousand nodes, with branch lengths between ten and twenty, and a branching factor between zero and two. There is a certain probability that each vertex is a theorem; theorems are biased towards the leaves and are clustered.

ST asked whether changing the oracle is permitted. EM replied affirmatively, but suggested that the general model should be adhered to (*i.e.* don't charge nothing for something that ought to cost something). He added that when many scientists are at work, perhaps more than one query ought to be answered per day.

ST asked whether "stochastic" research practices are to be condoned. By this he meant whether scientists are allowed to probabilistically abandon projects and start new ones. EM thought that such strategies might prove acceptable.

EM pointed out that oracles will select as projects nodes on level six or seven. He added that oracles could inform scientists of the distance between a given node and the root. However it would be too informative to provide the distance to the closest leaf. This restriction is designed to encourage focusing on bottleneck effects rather than on strategies depending on the distance to the goal. He noted that because several scientists may be working in the same area of the theory, clustering of theorems complicates matters. Hopefully extreme strategies will prove sub-optimal (otherwise the theory will be adjusted).

AS asked whether it should be assumed that all scientists pursue the same strategy. EM replied not necessarily; in any case the goal is socialistic, namely to maximize the output of the group rather than that of the individual. RC and EM observed that scientists descending the tree should probably select sub-branches randomly (but Of course remembering which choice was made) so as to avoid duplicating the steps of others following the same strategy. EM noted that it was likely that several scientists would be given projects in the same area of the theory.

ST wondered whether scientists should be told how many people are working on a given project. EM thought that this might be a good idea. AAS asked how multiple journals should be administered. EM replied that there were many possibilities; one is to have journals cover separate (but well defined) areas.

EM announced that he would supply the theory in a file of ten thousand vertex records of the form (l, r, t) where l and r specify the children vertex numbers (zero if non-existent) and where t is one if the node represents a theorem and zero otherwise.

6.3 Notes for Tuesday, March 13

EM announced the functioning of the theory-generating program. Contrary to a statement in the problem definition, leaves appear as high as level seven of the tree (levels counted from the root which is here assumed to reside on level 1).

SS raised several questions regarding the cost of traversing a tree edge. First, he suggested that perhaps it should take a random amount of time to descend a level (rather than always one day). Second, he wondered whether it really should take a day for every step up the tree.

EM replied that it might be acceptable to allow jumping up any distance at unit cost (though not beyond the proposal node). Going down a level should always cost a day.

EM suggested discussing the speed of the editorial office. MGB said that she was planning on having the office process three or four queries per day. RC pointed out that the critical question concerns the speed of the office relative to the number of active scientists. In particular, if the office processes as many queries per day as there are scientists, then the extreme strategy of checking with the office every day will be appropriate. EM and MGB contemplated a population of twenty to fifty scientists. EM suggested making the number of scientists a "variable constant" and setting up the editorial office to process a quantity of queries per day equal to the ceiling of the number of scientists divided by ten.

RWH asked what factors scientists can reckon into their strategies. For example, is a scientist permitted to ask how far it is to a theorem? EM answered that divulging the distance to the nearest theorem gives away too much. The only information to be divulged about a node is whether it lies in someone else's project, whether it is a theorem, and whether its subtree has been completely explored.

AAS suggested that one should be allowed to ask whether there are any theorems in a given subtree. EM didn't go for that proposition, thinking again that to do so would be to give away too much information. AAS then suggested that it would improve scientist's strategy if they were permitted to ask whether there were any theorems within a given number of levels. MGB pointed out that scientists prove theorems every day but that unfortunately most of these theorems are too boring to publish. EM objected to releasing any information concerning the location of theorems on the grounds that it would make possible strategies guaranteed to find a theorem in bounded time (*e.g.* in time equal to three times the depth of the tree). ANS and EM agreed that one could partially remedy this objection by giving answers to such questions with some probability of lying. EM was not enthusiastic however; he felt that the point of the problem was not really to accurately model the bitter realities of research, but rather to study certain trade-offs in parallel computation.

AAS wondered why it was reasonable to charge a fixed, single unit of cost for traversing every edge. EM responded that computation costs incurred at single nodes could be easily countered by parallel processing. However, attempting to ensure that each node be visited only once would exact a very high communications toll. By allowing multiple visits, this toll is reduced.

EM observed that since theory is random, it would be advisable to try each strategy on a number of theories. In this way one might hope to find a good policy regarding the communication versus computation trade-off.

ST pointed out that the problem statement permits no direct communication between scientists. Such communication could, he felt, be useful. For example, a scientist might decide to abandon a field and should be able to tell other scientists about that decision. EM replied that there were two issues here. The first concerns models of reality, and the second concerns tradeoffs between communication and computation given some model. He did not want to expend much effort on the first issue. As to the second issue, it arises in almost any parallel environment, and thus he felt that one could afford to fix on some particular model.

AAS announced that he was still troubled by the uniform cost of traversing edges. He wondered traversing a previously-traversed edge costs just as much as traversing an unvisited edge. He pointed out that one ought to be able to read the publications produced in previous traversals to reduce one's effort. AS paraphrased AAS by saying that in the proposed model, no one ever reads the publications. MGB added that previous work expended in reaching failure nodes is not preserved for later comers. MGB and EM observed that this holds in reality; no one publishes their failures. EM noted that when you get to a node you know whether it represents a failure or a theorem. AAS replied that there should be a cost corresponding to the difficulty of progressing from conjecture to proof. EM agreed that this was a valid criticism of the model; greater realism could be had by taking such matters into account.

EM, MGB and AAS agreed (after some argument) that although there was no advantage in "reading the journal" it might pay to send inquiries to the editorial office. The information thereby obtained would indicate whether other scientists were working on (or had already completed) an area of the theory. AS asked whether completion of a subtree referred to completion by an individual or completion by the group. EM replied that it was a group concept, and that it could be computed by attaching two flags to each vertex: one to indicate whether the vertex has been visited, and one to indicate whether the subtree has been completely explored.

SS observed that the theory consists of ten thousand nodes and that there are twenty scientists at work for a thousand days. Multiplication suggests that it might be possible to exhaustively search the tree. EM responded that this might not be the case since there might be unavoidable duplication of effort. AA S suggested that it might be possible to give each scientist a separate subtree to explore (by some kind of synchronization procedure) and thereby eliminate duplication. AS, SS and EM replied that should the subtrees vary drastically in length, some scientists would run out of new material and some subtrees would be incompletely examined. EM added that if only negligible duplication occurs the theory would be readjusted.

MGB raised the topic of multiple journals. She pointed out that the idea of two journals linked by a hot line would be equivalent to one journal working at twice the speed. ST suggested alternative journals, for instance a journal of failures and a journal of graffiti.

EM mentioned a variation in which "local reading rooms" are established. Theorems submitted to the journal are propagated to a set of local data bases each of which services queries from a subset of the scientists. Because it takes time to transmit information from the editorial offices to the local reading rooms, the information is not quite up to date. However, each reading room services only a few scientists, thus query queues remain short and service times are reduced. AAS asked for motivation for this variation. EM replied that such a scheme could be implemented in a parallel computation environment. With a tree architecture there would be a logarithmic (in the number of reading rooms) delay incurred in sending information from the editorial office to the local reading rooms. Delays at the reading roms would be short. This contrasts with a linear service time when all queries are sent to the editorial office.

AAS suggested that it was unreasonable to assume that all queries require the same amount of time to process; for example a cache could be employed to rapidly answer frequently-asked questions. EM replied that there exists a more fundamental issue. Most memory systems, whether addressed by location or by content, permit only one access (or at most a small fixed number of accesses) per unit time. Thus the queries must be serialized and that in itself ensures that only one query (or at most a couple of queries) can be answered per time step even in the best of circumstances. However it might be possible to reduce further processing delays by means of such things as caches.

RC raised the issue of allowing scientists to communicate directly with each other. EM pointed out that this really amounts to providing an additional communications network (analogous to a postal system or an ARPANET) and thus might befog the computation versus communication tradeoff.

6.4 Notes for Thursday, March 15

EM opened with the remark that at the last meeting discussion centered more on questions of modeling reality than on methods for overcoming bottlenecks in parallel computation. Two suggestions had been made, namely founding additional journals and localizing communication by establishi ng "reading rooms." (Caches had also been suggested, but EM noted that such improvements still depend upon serial access and thus do not address the fundamental issue.) EM solicited further suggestions for solving the problem.

RC asked for a clarification of the problem to be solved. EM replied that the problem was one of inventing a method for sharing data among processes.

EM suggested that another idea would be to employ broadcast communications such as those provided by transmitters, busses, or ethernets. RC pointed out that in evaluating the performance of this discipline it would be necessary to account for both sending and receiving costs. EM agreed that scientists would have to listen to the traffic on the bus. One possibility would be for them to store every message in local memory; another would be for them to screen all messages and just store the relevant ones. Using a broadcast channel would eliminate the central bottleneck. However, if traffic is heavy, the burden of storing everything locally would prove excessive. Furthermore, scientists would still face the tradeoff of deciding how frequently to check their local memories. Such checks would still cost at least a unit of time; though the queue delays associated with a central data base would be eliminated.

MGB and SR wondered what would be broadcast. One candidate that they and EM agreed on was theorems. EM quashed the erupting controversy by suggesting that the general problem (of distributing data to everybody when you don't know who needs what) be considered.

AAS proposed that the journal should simply record the status of the tree. Whenever a scientist visits a node, a message so indicating should be sent to the journal. AAS asserted that since there would be little or no conflict (since few nodes would be visited by more than one scientist) these messages could be queued. He also argued that, by analogy to multiple-user operating systems, simultaneous reads would present no problem. EM disagreed on the grounds that memories (whether disk, IC, etc.) are serially accessed. He also argued that reading and writing are in general analogous and that while simultaneous-read, single-write memories are of theoretical interest, they cannot be built at reasonable cost.

Returning to the topic of broadcast channels, EM and SR noted a range of possibilities. Each piece of information could be broadcast just once, under the assumption that each process would notice and store it. On the other hand, each piece of information could be broadcast repeatedly. This would trade memory for time.

SR argued that using a broadcast channel simply replaces the journal query queue with a channel access queue. He proposed multiple channels, each dedicated to a particular area of the theory. AAS objected, pointed out that if a scientist only records messages relevant to his area, he will be completely ignorant when he switches to a new project. ANS tried to pursue the question of channel contention but was drowned out by an argument about the merits of "mailing lists."

AAS asked for a clarification of the framework of the problem. He noted that he had assumed that the available hardware (say 25 scientists) was to be regarded as fixed. EM replied that part of the point of the problem was to find out what sort of hardware, and particularly what sort of architecture, was needed in a parallel environment. AAS suggested that in the absence of a bound on the amount of available hardware, one could, in the context of Problem 5, add more scientists to the editorial office. EM responded that such approaches failed to address the communication versus computation tradeoff. In order to encourage legitimate solutions, EM urged the class to read the Mohan paper cited previously. He also noted that the broadcast channel model raises tradeoff issues. For example, messages could be sent once and stored locally. This is inefficient since each piece of information will be stored in many places. Another possibility is repeated broadcasting; a third is to store locally only short approximations to the data.

SR raised the possibility that channel contention might cause the same bottleneck as a central data base. This would certainly be the case, he felt, if write operations occur as frequently (up to a constant factor) as read operations. EM replied that he didn't necessarily accept SR's hypothesis on the relative frequencies of reads and writes. Thus he felt that the broadcast channel might have some merit.

EM urged further thought on the general (and unsolved) problem of maintaining central data bases in parallel environments. He concluded by mentioning that the S 1 parallel computer project (conducted by the Livermore National Laboratory, a place better known for other activities) had compared loosely coupled systems with tightly coupled ones, and had found results similar to those given in the previously cited paper.

6.5 Projects

RC and HD wrote a simulation program to investigate the behavior of several research strategies. Their program implements a single journal capable of answering four queries per day. Five strategies were tried:

1. No communication: Each researcher receives a project at random at the beginning of the simulation, and searches that subtree until it is exhausted. Then the researcher requests another (randomly-chosen) project. Any theorems found are reported to the journal, but the reply (indicating whether the theorem has already been published) is ignored. Whenever a researcher has to choose between several children in deciding where next to visit, a random selection is made. When a leaf node is reached, the researcher backs up to the closest ancestor in the project possessing nodes unexplored by the researcher.
2 Check upon project assignment: Upon embarking on a new project each researcher inquires whether the project is already exhausted. If so, a new project is immediately selected.
3 Random communication: Upon reaching a node each researcher, with probability p, sends an inquiry to the journal. The purpose of the inquiry is to determine whether the subtree rooted at the node has been already completely explored. If so the researcher backs up to the closest ancestor with children unexplored by the researcher. (If no such node exists, the researcher selects a new project.) The researcher does not idle while waiting for a reply to the inquiry. Replies that arrive too late to be relevant are ignored. By varying the parameter p it is possible to observe the effect of different amounts of communication. Strategy 1 is obtained by setting p to zero.
4. Coordinated project assignment without communication: In this strategy projects are assigned by the journal (rather than randomly by the oracle). The journal assigns top priority to requests for projects; nevertheless researchers may be delayed when

requesting projects. The journal answers each request for a project by assigning the requesting researcher to the starting node of the project in which the fewest number of researchers are working.

5. Coordinated project assignment with communication: Projects are assigned by the journal as in the previous strategy, and researchers communicate as in strategy 3.

Simulation results are shown below for the case of 10, 20 and 30 researchers. RC and HD found that for a given strategy the number of theorems discovered is varies greatly from run to run. Therefore they chose to report instead a more stable quantity, the number of nodes explored.

Researchers	Strategy	p	Explored Nodes
10	1	- -	4944
	2	–	4887
	3	0.1	4878
		0.2	4574
		0.3	4065
		0.4	3734
	4	–	4764
20	1	- -	6175
	2	–	6591
	3	0.05	6713
		0.1	7254
		0.15	6892
		0.2	6793
	4	–	7499
	5	0.05	7008
		0.1	7399
		0.15	6985
30	1	–	7925
	2	–	8105
	3	0.03	7692
		0.07	7796
		0.1	8426
		0.13	7844
	4	–	8084

RC and HD drew the following conclusions from their data:

- For low numbers of researchers *any* communication is a waste of time. The more communication, the worse the performance. This can be explained by noting that it will be rare that two researchers will find themselves in the same project; so rare that it takes less time to search the project than to check whether someone else is there.

- For higher numbers of researchers the optimal amount of communication is an amount just less than that which causes the journal queue to saturate.
- The strategy of check upon project assignment is poor, but the strategy of coordinated project assignment may be best for intermediate numbers of researchers.
- Adding communication to the strategy of coordinated project assignment provides only marginal gains while swelling the journal queue to enormous proportions.

A. A List of “Almost Bricks”

A.1 Almost Bricks Sorted by Odd Leg

Presented below is a list of all almost bricks with side lengths less than 10,000,000. This list was computed by SS. It is sorted by odd side length.

85	132	720	19635	21964	166320	73017	141856	5196120
117	44	240	20163	33660	332384	77805	30240	141284
187	1020	1584	22473	20864	850080	83475	14500	29568
195	748	6336	23751	7800	29920	84609	187488	556160
231	160	792	24035	30636	70752	85425	810568	839520
275	240	252	24225	22304	51480	96075	164164	556320
429	880	2340	26649	7920	15232	97825	4044096	7656472
496	4888	8160	26775	50880	176176	9854Q	16380	62832
693	140	480	27027	62700	573040	QQGO3	295460	654720
835	832	2640	27755	42372	62160	100035	16016	207900
935	17472	25704	28083	43056	105820	100125	199056	691708
1105	9360	35904	29601	90480	156032	100485	209328	1131020
1155	1008	1100	30195	100100	137904	101565	240900	1041392
1155	6300	6688	32175	169600	339552	102765	65520	394196
1575	1672	9120	32375	49896	427200	103075	28644	281808
1755	4576	6732	33201	59400	362080	103095	66528	446600
1881	1080	14560	34965	62900	358512	106227	154660	237120
2035	828	3120	35075	237GO	35604	10803 1	212160	289800
2079	44080	65472	35321	7560	13728	108405	83804	432960
2163	15840	37100	35409	54288	79040	110979	08172	141680
2295	1560	5984	3G::OQ	203080	2414412	111159	122760	176800
2475	780	2992	37835	269280	1244484	115115	330372	408090
2925	3536	11220	38475	2964	6160	115805	5004 12	5060016
4599	18308	23760	38571	21328	25740	117117	85956	1006480
4599	23760	144832	39 195	1188	16016	J17469	161040	194220
4901	4308	13860	4247 1	54280	59040	117711	255200	450360
549 1	41580	46512	42053	240240	315 180	118035	225492	748880
5643	14160	21470	46371	210672	1277080	118755	149600	455532
5643	4 3680	76076	46431	269600	3629208	119669	48 1740	1227600
6325	528	5796	47975	84840	107712	128205	28704	247940
6435	24080	24684	51129	30080	85800	131157	167440	272580
7579	8820	17472	51205	36432	5 1324	134805	118404	241072
7885	16320	85932	52185	19800	302 176	137025	737880	1591744
8415	5720	157248	52GJ 1	83952	109340	141075	129888	536900
8415	157248	643720	54087	54784	364320	141525	806652	1451120
8789	10560	17748	57275	8532	36960	144837	363440	404700
9045	3696	121940	58425	300608	729144	148005	81840	122636
9405	2964	9152	59085	166012	2585610	151625	22572	56640
9405	23600	53196	59675	163152	587 100	151803	301200	845020
10395	63364	327360	GO885	302004	820820	153725	1498380	1622304
10395	95004	220400	612 15	121204	14 1120	154671	58400	105528
10726	4928	30780	61845	9504	3 1372	156519	47200	J06392
10725	7840	9828	61975	4 12920	425508	157157	593676	942480
11753	10296	16800	62205	19604	55440	160185	219600	375232
12075	J 008	1100	62415	145404	362848	160257	100776	209440
12915	36720	290444	63063	5320	353760	167531.	168300	3144960
14715	148148	267 120	66495	53856	277100	168245	495264	633052
15225	17792	308880	66495	146160	313472	169575	49088	360360
15939	18460	48720	67575	23936	33120	171171	112860	429520
16929	GO72	18560	67925	86580	332112	173565	2577QG	5465920
17157	4900	23700	69165	566720	599748	176715	1077188	1927200
17325	100320	264404	69355	28512	138510	177177	38080	47736
18525	71060	90576	69513	41360	83520	179333	25344	109140
19175	J 12320	293832	72G 11	9180	200448	182457	2371'20	457840
19305	14112	15400	72765	320128	511980	185025	191620	1270704

186065	310896	3476928	341649	4114240	5538768	570505	164808	411840
186615	321816	983080	343827	149500	1050960	570843	234960	1128524
188859	97812	245440	350493	7531524	8448640	574425	134288	732480
190405	102828	252000	352275	485316	1608880	575575	2152512	5678640
191065	81576	1399200	354123	934800	1490020	576375	204160	1175328
191345	853944	2126592	356235	3983980	6656832	581009	4063488	4809240
191709	773300	1623888	361665	57904	116928	588945	1313128	1805760
192465	145376	2796600	362805	1901900	4880304	592999	524160	2700432
195415	4773G	3051648	366125	694260	746592	G00357	153076	570960
200 165	277200	970590	370125	699600	4421950	603075	84420	1570016
200385	169312	211200	373175	1055808	1932840	G07725	2627148	6231280
200385	169312	467016	376363	96900	205920	611325	264860	562848
200583	9856	61560	384615	3293000	5075136	613795	471276	979440
20 1285	95004	791120	387849	1143040	4451832	615195	322400	767052
201663	186416	262080	389367	426880	1190160	615615	2086200	3564704
206625	222200	421344	392535	2069120	2816208	616605	70700	134064
209385	119680	402696	400365	1245332	2406624	617419	835380	946608
209825	223104	293040	401115	475776	2941208	617715	1723984	3873012
211327	1358640	2487930	406245	299884	2098800	625053	277300	848160
213785	325008	770880	406315	34452	134064	631125	200300	644688
214305	46816	122760	412425	4267872	4637400	631533	514080	2257244
217217	279744	378000	414869	294492	327600	636115	708180	2203344
218595	431460	544544	415989	1233980	3321648	638685	86508	102256
219965	684372	761904	416075	1106772	2165700	641355	609860	880992
221805	315172	1165104	419525	1421244	1847040	645975	230112	256360
224025	215072	434304	421245	537152	2207700	647955	978576	1798940
227799	48960	181720	426105	2307360	2619904	657041	251160	465120
229229	1274580	2380560	426173	2873340	2923530	667575	390816	1343320
237575	214368	1492920	430355	134640	948948	667755	599676	1675600
240669	234780	900592	430911	5748160	6307152	673475	77268	99360
242535	560120	713952	432837	2403500	9397200	685425	631072	3674880
246675	754000	1734084	442035	142912	892620	686565	1712580	2991328
252637	1422960	3369780	448533	1489620	2845744	689481	146960	1110720
260107	774 180	1475760	449757	1439000	2280720	699567	3345056	9719640
263835	97092	1308944	450225	5368	163680	700557	1848924	2141360
263865	3874328	6763680	451269	568480	937860	712725	102340	643104
265353	1104320	3686760	460845	233772	653200	720005	2661660	5782064
267813	852720	4326484	461619	261580	1000800	720291	1029600	1175300
272745	101952	684400	462825	109600	130152	733623	801864	3053120
272987	371280	550116	463095	359040	2530528	735885	823680	1814510
273581	25908	95040	469395	75152	413364	736255	539352	1815840
274275	97152	198220	474045	2837536	8004348	749133	701760	2079844
274527	7336	480480	474145	72864	143640	749595	1187316	3801200
2749.1 J	157760	526680	474903	348920	2503290	750295	154440	308448
278355	3508388	4264416	477369	263120	691008	758043	336300	1902160
287287	216720	369984	477763	296400	3654540	772145	1041408	4848000
291885	1521036	3096848	486875	108 108	250800	777483	2536380	4 143200
295113	133400	2100384	489555	875472	996740	7793:: J	4117680	4493500
296829	857472	950300	480991	105600	2110680	784125	2115828	5159440
296989	943920	2484300	405349	180 180	215760	799389	351648	452980
299145	34632	66976	500175	80080	229824	806949	284700	437360
301587	156960	752284	503685	102080	343470	807675	252000	401212
301045	201300	204336	504075	461916	5216288	811965	181702	272844
303195	66700	1050192	604735	623480	1789344	820105	1619904	2502072
305877	135700	147000	508635	497904	627628	833745	492800	504648
306603	77004	769360	508875	1444300	2738736	839475	432432	1947500
306891	2821500	4562800	512325	1043100	1782352	839575	216720	272832
308499	588008	1750320	512533	6215220	7694544	847665	1031872	1705704
316635	120320	204516	518661	230100	271040	851499	5800GO	799920
318175	3357792	6431880	523341	2056912	4357980	853655	812448	1339404
323323	112320	300900	524349	273840	609620	855855	3303304	4511360
325975	1063392	4995144	524797	2711280	3072900	859815	68672	313896
326895	211040	645696	526311	245520	914048	863811	301248	2966660
330165	672220	3502992	536877	127136	452100	865305	83770	1244760
331177	483840	1550736	539847	42240	63296	876645	774384	2433800
331485	350020	379008	556605	457028	1025904	894801	406980	2026640
331721	475200	1914960	561105	759360	1329328	908105	943488	1029600
332469	314160	422300	564311	128520	459300	910665	776776	7915200
335049	J 87200	2734600	567153	535920	4647104	918099	118000	265980
335825	426360	753984	567801	201960	1758400	922077	1721764	2574000

933075	332384	837540	1611675	1580304	4276340	2595775	1494768	5765760
935253	761904	1967420	1620465	1186328	3182400	2612225	1426920	7390656
935715	804QG	380380	1621477	664020	789264	2621619	8549200	9181380
939339	2889348	4319840	1638555	108 1652	3491136	2636361	2983160	3321120
954771	403172	5377680	1645699	933660	1407120	2654135	5278680	9439872
968253	328320	430996	1654653	720720	1981396	2654619	1010592	4302180
970437	1487200	6195420	1666665	52096	2650128	2673585	1108536	4523552
970783	66456	542400	1696761	3946680	6073760	2673585	4523552	9235200
992405	914628	1049040	1732599	6596920	7210560	2677103	1458504	1990560
1013829	772772	5710320	1733809	540960	1089000	2678571	2553980	4263000
1013850	1437040	2832900	1743525	92 1388	2605680	2699097	1158840	3907904
1018875	744800	748836	1749825	1212000	2812888	2736405	3158804	6857328
1019711	354240	553320	1752751	554760	1374432	2750345	1504800	5844696
1027675	223440	1948710	1769229	3786640	5067972	2757755	38304	80340
1039419	812592	2534620	1791075	1282644	1282960	2760307	88176	1337220
1052623	1419840	2861136	1792317	1532060	2974356	2796915	3744676	4910400
1056965	579684	4245120	1801107	3395980	8955024	2800083	4107740	6271056
1067605	116688	1280916	1823041	3734640	4202688	2807805	810900	5335616
1073995	208692	869856	1826181	5498592	8476700	2807805	1527372	4240490
1077615	1333568	2951424	1839941	2476080	5637060	2812095	985872	1654400
1090635	774180	2072512	1840575	258400	3333528	2822391	854488	2518080
1095633	390720	506456	186 1755	964512	3579884	2841075	4854564	9883952
1107225	521664	2591480	1873989	6621120	8469340	2843145	191352	275264
1107743	1071840	1309176	1900965	5751508	6375600	2844205	2018940	2023632
1110675	180180	269072	1902375	796928	1157520	2847105	621984	1085812
1118425	1461240	2810304	1907451	1012460	3336432	2856425	1374144	8452080
1146035	2536380	3138816	1916475	2788240	3634092	2863245	823004	1992672
1146845	425040	676972	1918545	636768	1615976	2866149	1651580	4255440
1153427	910800	1283100	1930467	1055050	2698740	2873045	938676	953568
1163085	905324	1227600	1953105	2583360	3350672	2883595	361152	792540
1178931	525460	654 192	1955511	1155840	1909600	28904 59	3612240	8355900
1180971	268780	4054128	1961375	591888	964800	292634 1	1779440	4192188
1198925	282348	669120	1965183	132 1856	6383520	2982525	323380	1430352
1203895	460200	705 1968	1982695	2690400	6064344	2984247	2122120	8985504
12 10825	1120056	1893600	1987557	163676	472320	3008745	147840	481712
1214955	4 18 132	5809440	1992681	1431360	3905792	30013435	191268	885920
1233513	481440	9462200	1993005	310088	362340	3035725	1082628	5776800
1239315	759924	1566200	2015775	509888	995280	3057093	1818960	3456100
1264545	2101440	3168088	2034747	613600	902700	3063995	2265588	3782160
1264835	131196	4242672	2054565	328944	2428580	3153645	4106492	5937360
1281735	2220128	5063520	2061675	1010276	3917760	3168825	2016824	4749120
1288963	474240	3045180	2066265	3783736	6776352	3198195	2650004	3059760
1294995	6126 12	3066800	2070705	83776	11 J 7440	32204 J 9	1556240	2960100
1297525	150012	1750320	2J 19935	2977920	7558048	325348 l	1920000	3504240
1306305	2801656	0158208	2135705	29664	10 18248	3292341	84 1340	842688
1307859	619920	1589060	2139585	81 1800	2153536	33 16907	2340624	7619940
J322685	923052	31 13264	2144115	498212	3219210	3337225	2533 160	3537792
J 33 1275	2580864	2688300	2157285	195408	7364500	3338205	858 176	2777940
1353275	1605708	8633856	2179485	2469676	3341520	3357285	2497572	3172400
1354815	2574528	3626896	2236125	643500	4445584	3364725	2669348	7398864
1356075	459360	1306396	225094 1	4729140	5628480	3378485	1546116	2508480
13741555	1024452	5400064	2253537	2010560	6640920	3388185	580920	981376
1374891	422812	720720	2254825	2138640	5062464	3455641	3369960	9653280
1379125	1968000	4970532	2288265	136136	351648	3461179	706860	1997520
1392075	249964	1330560	2329509	813020	4712400	3462225	1226368	5163120
1420419	607392	3968900	234 1449	937568	1883520	352 1583	4093056	742GQGO
1427415	41184	312280	2342359	526680	1133088	3522375	3816120	4268768
1437975	2765560	3613248	2362635	1126692	1935680	3608605	30156	772992
1440285	1368900	2575664	2399265	1983696	6494272	3612141	2984688	4710860
1446723	5061964	5918880	24 13675	285360	364572	3716115	955328	7334820
1448655	1345960	42644 16	2446725	1622060	5493312	3724875	5642240	6555276
1491903	801360	1316480	2447703	1804560	6267492	37324 17	2195840	3103344
150 1383	1173744	5304040	2477875 .	49280	1970892	3795825	408720	4507048
J506375	26364 8	7050120	250483 l	205920	5 15592	3800745	5319352	6219840
1517373	757900	7998336	2516319	668800	1476600	3806859	2051280	2578900
1587355	335916	2531712	2518725	732780	1474528	382 1565	2358972	2732080
1599565	1929312	3137916	25 18725	818496	3170860	3828995	561132	7046160
1600435	46176	211008	2519153	92400	732090	3893197	1518804	1845360
1601145	662704	1383360	2564661	2278 100	2423952	3950325	142240	1027356
1606165	5796720	7250628	257 1233	17850	1660560	3974855	345576	1205568

3993535	1299408	2882880	5288547	5122780	5245200	7162085	863280	1137948
4003857	2779840	8849880	5306301	682000	2354100	7243775	3092760	4009824
4009005	2600048	6980340	5351995	8903664	9278052	7290435	5783580	7138624
4054505	2821728	9816840	5406555	141372	176800	7300293	2484720	2828540
4123405	1024800	4948524	5540535	512992	1169256	7301151	2293368	5069120
4150575	3982680	5710352	5544825	54720	614384	7336945	2548800	2738736
4158115	100320	759132	5595975	762280	1619904	7344289	8131200	8605080
4168675	1572480	5039892	5673549	4657660	9540432	7352107	7759920	9969300
4180185	1804200	2063776	5682159	1339200	1688720	7514493	617760	3001180
4210679	1313760	6151080	5799825	4113120	5725688	7545875	4046988	4984560
4216245	214524	1645600	5811165	2106780	4910048	7567263	434816	3180600
4293015	1888480	3332664	5827965	404404	1983600	7604835	2945052	6 195280
4293315	4803708	5778080	5906667	2304540	5205200	7650825	1850904	7044128
4350645	598400	2003868	G015581	3902800	6543108	7654535	3519912	4634784
4351459	539220	(387312	GO49455	4165272	8790496	7773147	970140	597 1504
4355085	2106720	9369436	6088803	3191760	9028700	7850557	8298576,	9004260
43632115	534576	3197700	6111721	4512480	6809472	7891885	2530116	6169680
4307853	3734640	6582796	(3117045	3462592	5757444	8118357	3437280	5099500
4399241	1150560	6469080	6213375	2601720	2642624	8191161	1147600	4488000
4410055	1475760	5208096	6237605	733044	1291392	8253905	1756512	3500184
4423545	2179232	2889000	6238080	706552	8303040	8421885	2910820	5233008
4460103	88704	111520	6403683	6965244	8901920	8430525	3973200	5634412
4477187	2402016	2857140	6403775	7702296	7920000	8709987	706420	5403600
4519515	5820480	7715708	6413913	4952640	5845616	8722105	1232640	2025408
4533815	3162816	3385200	6425679	67724 80	8015400	8729875	313632	8430300
4540526	4160772	4717440	6505191	2734200	3409120	8730579	1556100	7227440
4559685	180732	843920	6530111	281160	885600	8794979	956340	GO72528
4574955	1349084	3310560	6551919	209920	1336608	8830459	2894688	9866340
4575483	59280	606956	6058431	793408	4614480	8841417	3365856	6027560
4620147	1712304	3403540	6696795	610060	731952	8858795	3794400	9027012
4669203	2097396	2438480	6794645	150384	688860	9072063	2717416	5206080
4702887	1128920	7207200	6794865	152.1312	4527320	9004371	376740	1268672
4734639	959552	1407120	6936501	166500	1707200	9261569	3825360	7788480
475J285	1974000	5008068	6944861	1636800	5507460	9299225	6503112	753 1680
4799015	488370	2431908	6954519	1591300	1901240	9555741	3273940	7266512
4801797	1468896	7385140	6956235	3172148	4302480	9569749	3977820	5453760
4838625	3047352	0515200	6956565	13004763	2541968	9646011	452848	3318900
4924071	5208840	8557120	7083747	1972496	2450580	9828013	4229940	6715584
5046195	1322288	1631700	7 149205	2372832	3002940	9915675	1536596	1718640
5111703	3736304	7423680	7161165	1637916	3715712			

A.2 Almost Bricks Sorted by Shortest Leg

Presented below is a list of all almost bricks with side lengths less than 10,000,000. This list was computed by SS. It is sorted by shortest side length.

44	117	240	9856	61560	200583	41360	69513	83520
85	132	720	10296	11753	16800	42240	G3296	539847
140	480	693	10395	63364	327360	42471	54280	59040
160	231	792	10395	95004	220400	426353	240240	315180
187	1020	1584	12915	30720	290444	46176	211068	1600435
195	748	6336	14112	15400	19305	46371	210672	1277980
240	252	275	14500	29568	83475	46431	2GQGOO	3629208
429	880	2340	14715	148148	267120	46816	122760	214305
495	4888	8160	15225	17792	308880	47200	106392	156519
528	5796	6325	15939	18460	48720	47736	195415	3651648
780	2475	2992	16016	100035	207900	47975	84840	107712
828	2035	3120	16380	G2832	98549	48960	181720	227799
832	855	2640	17325	100320	264404	49088	169575	360360
935	17472	25704	17856	1660560	2571233	49280	1970892	2477835
1008	1100	1155	18525	7 1060	90576	52096	1666665	2650128
1008	1100	12075	19175	112320	293832	52611	83952	109340
1080	1881	14560	19604	55440	62205	538563	66495	277 160
1105	9360	35904	19635	21964	166320	54087	54784	364320
1155	6300	6688	19800	52185	302176	54720	614384	5544825
1188	16016	39195	20163	33660	332384	57904	116928	361665
1560	2295	5984	20864	22473	850080	58400	154671	165528
1575	1072	9120	21328	25740	38571	58425	300608	720144
1755	4576	6732	22304	24225	51480	59085	166012	2585616
2079	44080	65472	22572	56640	151525	59280	606956	2575483
2163	15840	37100	23760	35075	35604	59675	163152	587 100
2925	3536	11220	23936	33120	67575	60885	302064	820820
2964	6160	38475	24035	30636	70752	61215	121264	141120
2964	9152	9405	25344	109140	179333	61975	412920	425568
3696	0045	121940	25908	95040	273581	62415	145464	362848
4368	4901	13860	26775	50880	176176	65520	102765	394196
4599	18308	23760	27027	62700	573040	66456	542400	970783
4599	. 23760	144832	27755	42372	62160	66495	146160	313472
4000	17157	23760	28083	43056	105820	66528	103005	446600
4028	10725	30780	28512	69355	138516	66700	303 195	1050192
5320	63063	353700	28644	103075	281808	67925	86580	332112
5368	163680	450225	28704	128305	247040	68172	110979	141680
549 I	41580	46512	29601	904 80	150032	68672	313896	859815
5643	14100	2 1476	29664	1018248	2135705	69165	566720	599748
5643	43680	76076	30080	51129	85800	70700	134064	616605
5720	8415	157248	30156	772992	3608605	72765	326128	511980
GO72	16929	18560	30195	100100	137904	72864	143640	474 145
Q435	24080	24684	30240	77805	141284	73017	141856	5 196120
7336	274527	480480	32 175	169600	339552	75152	413364	469395
7560	13728	35321	32375	49896	427200	77004	306603	769360
7579	8820	17472	33201	59400	362080	77268	99360	673475
7800	23751	29920	34452	134064	406315	80080	220824	500175
7840	9828	10725	34632	66976	299145	80496	380380	Q357 l5
7885	16320	85932	34965	62900	358512	81576	191065	1399200
7020	15232	26649	35409	54288	79040	8 1840	122636	148005
8415	157248	043720	36309	203680	24 144 12	83776	865305	1244760
8532	36960	57275	36432	51205	51324	83776'	I IL7440	2070705
8789	10560	17748	37835	200280	1244484	83804	108405	432960
9180	72611	200448	38080	47736	177177	84420	GO3075	15700 16
9405	23600	53196	38304	80340	2757755	84609	187488	550100
9504	31372	6 1845	41184	312280	1427415	85425	810508	839520

85956	117117	lOQG480	157760	274911	526680	267813	852720	4326484
86508	1022563	638685	160185	2 19600	375232	268780	1180971	4054128
88176	1337220	2760307	163676	472320	1987557	272087	371280	550110
88704	111520	4460103	164808	411840	570505	273840	524349	609620
92400	732006	2519153	166500	1707200	6936501	277300	625053	848100
95004	201285	791120	167531	168300	3144960	278355	3508388	4264416
96075	164164	550320	168245	495264	533052	281160	885600	6530111
QGQOO	205920	376363	169312	200385	211200	282348	669 120	1198925
97092	263835	1308944	169312	200385	467016	284700	437360	806949
97152	198220	274275	173565	257796	5465920	285360	364572	2413675
97812	188850	245440	176715	1077188	1927200	291885	1521036	3096848
97825	4044096	7655472	180180	215760	495349	294492	327600	414869
QQGO3	295460	654720	180180	260072	1110675	296400	477763	3654540
100125	199056	691708	181702	272844	811965	296829	857472	950300
100320	750132	4158115	182457	237120	457840	296989	943920	2484300
100485	209328	1131020	185925	191620	1276704	299884	406245	2098800
100776	166257	209440	186065	310896	3476928	306891	2821500	4562800
101565	240900	1041392	186416	201663	262080	308499	588008	1750320
101052	272745	684400	186615	321816	983680	313632	8430300	8729875
102080	343476	503685	186732	843920	4550085	314160	332460	422300
102340	643104	712725	187200	335049	2734600	318175	3357702	6431880
102828	100405	252000	191268	885920	3009435	319088	362340	1993005
105000	489901	2110680	191345	853444	2126592	322400	615195	767052
106227	154660	237120	101352	275264	2843145	323380	1430352	2082525
108031	212160	289800	191709	773300	1623888	325975	1063392	4995 144
108 108	250800	486875	195408	2157285	7364500	328320	430QQG	968253
109600	130152	462825	200165	277200	970596	328944	2054565	2428580
111159	122700	170800	201300	204336	301045	330165	672220	3502992
112320	300900	323323	201960	567801	1758400	331177	483840	1550736
112860	171171	429520	204100	576375	1175328	331485	350020	379008
115115	330372	408096	205020	515592	2504831	331721	475200	1914960
115805	500412	5060016	206625	222200	421344	332384	837540	933075
110088	1067605	1280916	208692	869856	1073995	335825	426300	753084
1174GQ	161040	194220	209300	631125	644688	335916	1587355	2531712
117711	255200	450360	209825	223104	293040	336300	758043	1902160
118000	265980	918099	200920	1336608	6551919	341649	4114240	5538768
118035	225492	748880	211327	1358640	2487936	345570	1205568	3974855
118404	134805	241072	211640	326895	645696	348920	474903	2503296
118755	149GOO	455532	213785	325008	770880	350493	7531524	8448640
119669	481740	1227600	214368	237575	1402920	351648	452980	799389
119680	209385	402696	214524	1645600	4216245	352275	485316	1608880
120320	204516	316635	215072	224025	434304	354123	934800	1490020
127136	452100	536877	216720	272832	830575	354240	553320	1019711
128520	459360	564311	210720	287287	369984	356235	3083980	6656832
129888	141075	530900	217217	270744	378000	350040	463095	2530528
131157	167440	272580	218595	431460	544544	361152	703540	2883595
131196	1264835	4242672	219965	684372	761904	362805	190 1900	4880304
133400	295113	2100384	221805	315172	1165104	366125	694260	746502
134288	574425	732480	223440	1027675	1048716	370125	699600	4421956
134640	430355	948948	229229	1274580	2380560	373175	1055808	IQ32840
135700	147GOO	305877	230 100	271040	518661	370740	126 8G73	(3004371
136136	351648	2288265	230112	250300	G45975	384615	3293000	5075136
137025	737880	1591744	233772	460845	653200	387849	1143040	4451832
141372	176800	5400555	234780	240669	900592	389367	426880	1190160
141525	806652	1451120	234QGO	570843	1128524	390720	506456	1005633
142241)	1027356	3950325	242535	500130	713952	390816	GG7575	1343320
142912	442035	892620	2455L0	526311	914048	391248	863811	2966660
144837	363440	404700	246675	754000	1734084	392535	2069120	2810208
145376	192465	2796600	249964	1330560	1392075	400365	1245332	2406624
146960	689481	1110720	251100	465120	057041	401115	47577G	2941208
147840	481712	3008745	353000	401212	807675	403172	954771	5377680
149500	343827	1050960	252637	1422960	3360780	404404	1983600	5827965
150384	688860	6794645	258400	1840575	3333528	406980	894861	'2020040
150612	1297525	1750320	260107	774180	1475700	412425	4307873	4657400
151803	361200	845020	261580	461619	1000800	415989	1233980	3321048
153076	570900	G00357	263120	477369	691008	416075	1106772	2165760
153725	1498380	1622304	263648	1506375	7050120	418132	1214955	5809440
154440	308448	750295	263865	3874328	6763680	419525	1421244	1847040
156960	301587	752284	264860	562848	(111325	421245	537152	2297700
157157	593676	942480	265353	1104320	3686760	422812	720720	1374891

425040	576072	1146845
426105	2307360	2619904
426173	2873340	2023536
430911	5748160	6307152
432432	839475	1947500
432837	2403500	9397200
434816	3180600	7567263
448533	1489620	2845744
449757	1439900	2280720
451269	568480	937860
452848	3318000	9646011
457028	556605	1025904
450360	1306396	1356075
460200	1203895	7051968
461916	504075	5216288
468720	3705825	4507648
471276	613795	079440
474045	2837536	8004348
474240	1288063	3645180
481440	1233513	9462200
488376	2431968	4709015
489555	875472	996740
402800	504648	833745
497904	508635	627628
498212	2144115	3219216
504735	G23480	1789344
508875	1444300	2738736
509860	641355	880992
500888	995280	2015775
512325	104 3100	1782352
512533	(3215220	76394544
512092	1169256	5540535
514080	631533	2257244
521664	1107225	2591480
523341	2056012	4357980
524160	592999	2700432
524797	2711280	3672900
525460	654192	1178931
526680	1133088	2342359
534576	3197700	4362115
535920	567153	4647104
530220	687312	4351459
539352	736255	1815840
540960	1089000	1733809
554 760	1374432	1752751
561105	759360	1329328
561132	3828995	7046160
575575	2152512	5678640
579684	1056965	4245120
580060	799920	851499
580020	981376	3388185
58 1009	4063488	4809240
588045	1313128	1805760
591888	964800	1961375
598400	2063868	4350645
599676	667755	1675600
GO7302	1420419	3968900
GO7725	2637148	6231280
610060	731952	6696795
612612	1294995	3066800
613600	902700	2034747
6156 I5	2086200	3504704
617419	835380	946608
617715	1723984	3873012
617760	3001180	7514493
619920	1307859	1589060
621984	1085812	2847195
631072	685425	3674880
636115	708180	2203344
636768	1515976	1018545
643500	2236125	4445584
647955	978576	1798940
662704	1383360	1601145
664020	789264	1621477
668800	147GGOO	2516319
682000	2354100	5306301
686565	1712580	3991328
699567	334505G	9719640
700557	1848924	2141360
701760	740133	2079844
706420	5403600	8709987
706552	6238089	8303040
706860	1997520	3461179
720005	2661660	3782064
720291	1029600	1175300
720720	16354653	1981396
732780	1474528	2518725
733044	1291392	6237605
733623	801864	3053120
735885	823680	1814516
744800	748836	1018875
749595	1187316	3801200
757900	1517373	7998336
759924	1230315	1565200
761904	933253	1967420
762280	1619904	5595975
772145	1041408	4848000
772772	1013829	5710320
774180	1090635	2072512
774384	876645	2433860
776776	910005	7915200
777483	2536380	4145200
770331	4117680	4493500
784125	2115828	5159440
793408	4614480	6658431
796928	1157520	1902375
801360	1316480	1491903
810900	2807805	5335616
81 1800	2130585	2153536
812448	853655	1339464
812502	1039419	2534620
813030	2329509	4712400
818496	2518725	3170860
820105	1619904	2502072
823004	1992672	2863245
841340	842688	3292341
847665	1631872	1705704
854488	25.18080	2822391
855855	3362304	4511360
858176	2777940	3338205
863280	1137948	7162085
005324	1163085	1227600
908 105	943488	1029600
910800	1153427	1283100
914628	992405	1049040
021388	1743525	2605680
922077	1721764	2574000
923052	1322685	3113264
933660	1407120	1645699
937568	1863520	2341449
938676	953568	2873045
939339	2889348	4319840
955328	3716115	7334820
956340	6072528	8794979
959552	1407120	4734039
964512	1801755	3579884
970140	5971504	7773147
970437	1487200	0195420
985872	1654400	2812095
1010276	2061675	3917760
1010592	2654619	4301180
1012460	1907451	3336432
1013859	1437040	2832800
1024452	1374555	5400064
1024800	4123405	4948524
1052623	1410840	2861136
1055056	1030467	2698740
1071840	1107743	1309176
1077615	1333568	2951424
1081652	1638555	3491136
1082628	3035725	5776800
1108536	2673585	4523552
1118425	1461240	2810304
1120056	1210825	1893600
1126092	1935680	2362635
1128920	4702887	7207200
11463035	2536380	3138816
1147600	4488000	8191161
1150560	4399241	6469080
1155840	1909600	1055511
1158840	2699697	3907904
1173744	1501383	5304640
1186328	1620465	3182400
1212000	1749825	2812888
1226368	3462225	5163120
1232640	2025408	8722105
1264545	2101440	3168088
1281735	2229128	5063520
1282644	1282960	1701075
1299408	2882880	3993535
1306305	2801656	G158208
1313760	4210679	6151680
1321856	1965183	G383520
1322288	16331700	5046195
1331275	2580864	2688300
1339200	1688720	5682159
1345960	1448655	4264416
1349684	3310560	4574955
1353375	1605708	8633850
1354815	2574528	3626896
1360476	2541068	6956565
1368900	1440285	2575664
1374144	2856425	8453080
1370125	1968000	4976532
1426920	2612225	7300656
1431360	1992681	3905792
1437975	2765560	3613248
1440723	5061964	5918880
1458504	1990560	2677103
1468896	4801797	7385140
1475760	44 10055	5268096
1494768	2595775	57635760
1504800	2750345	5844696
1518804	1845360	3803197
1521312	4527320	6794865
1527372	2807805	4240496
1532960	1702317	2074356
1536596	1718640	9915675
1546116	2508480	3378485
1556100	7227440	8730579
1556240	2960100	3226419
1572480	4168675	5030802
1580304	1611675	4276340
1591200	IQ01240	6954519
1599565	1020312	3137910
1606165	5796720	7250628
1622060	2440725	5403312
1636800	5507460	6944861
1637916	3715712	7161165
1651580	2866149	4255440

1696761	3946680	6073760	2278100	2423952	2564061	3191760	GO88803	9028700
1712304	3403540	4020147	2203368	5069120	7301151	3272940	7260512	9555741
1732500	6596920	7210560	2304540	5205200	5906667	3365856	6027560	8841417
1756512	3500184	8253905	23406324	3316907	7619940	3369960	3455641	9653280
1769229	3786640	5067972	2358972	2732080	3821565	3437280	5099500	8118357
1779440	2926341	4192188	2372832	3002940	7149205	3462592	5757444	6117045
1801107	3395980	8055024	2402016	2857140	4477187	3519912	4634784	7654535
1804200	2063776	4189185	2464720	2828540	7300293	3521583	4093056	7426960
1804560	2447795	6267492	2497572	3172400	3357285	3522375	3816120	4268768
1818960	3057093	3450100	2530116	6169680	7801885	3724875	5642240	6555276
1833041	3734640	4202688	2533160	3327225	3537792	3734640	4367853	6582796
1836181	5498592	8476700	2548800	2738736	7336945	3736304	5111703	7423080
1839941	2476080	5637060	2553980	2678571	4263600	3794400	8858795	9027012
1850904	7044128	7650825	2600048	4009005	6980340	3800745	5319352	6219840
1873989	6021120	8469340	2601720	2642624	6213375	3825360	7788480	9261509
1888480	3332664	4293015	2621619	8549200	9181380	3973200	5634412	8430525
1900965	5751508	6375600	2636361	2983160	3321120	3977820	5453760	9569749
1916475	2788240	3634092	2650004	3059760	3198195	3982680	4150575	5716352
1920000	3253481	3504240	2654135	5278680	9439872	4046988	4984560	5'545875
1953105	2583360	3350672	2669348	3364725	7398864	4113120	5725688	5799825
1972496	2450580	7083747	2673585	4523552	9235200	4160772	4540525	4717440
1974000	4751285	5008068	2717416	5206080	9072063	4165272	6049455	8790496
1982695	2690400	6064344	2734200	3409120	6505191	4229940	6715584	9828013
1983GQG	2399265	6404272	2736405	3158804	G857328	4293315	4803708	5778080
2010560	2253537	6640920	2779840	4003857	8849880	4512480	6111721	6309472
2016824	3168825	4749120	2796915	3744670	4910400	4519515	5820480	7715708
2018940	2023632	2844205	2800083	4107740	6271056	4657660	5673549	9540432
2051280	2578900	3806859	2821728	4054505	9816840	4924071	5208840	8557120
2066265	3783736	6776352	2841075	4854564	0883952	4952040	5845610	0413913
2097396	2438480	4669203	2890459	3612240	8355900	5122780	5245200	3288547
2106720	4355085	9369436	2894688	8830450	9866340	5351995	8903664	8278052
2106780	4010048	5811165	2002800	GO15581	6543108	57835811	7136624	7290435
2119935	2977920	7558048	2010820	5233008	8421885	6403683	6965244	8001920
2122120	2984247	8085504	2945052	6195280	7604835	0403775	7702296	7920000
2138640	2254825	5062464	2984088	3612141	4710860	6425679	6772480	8015400
2179232	2889600	4423545	3047352	4838625	G515200	6503112	7531680	9299225
2179485	2469676	3341520	3032760	4000824	7243775	7344289	8131200	6005080
2195840	3193344	3732417	3153645	4106492	5937360	7352107	7759920	9969300
2250041	4729140	5628480	3162816	3385200	4533815	7850557	8208576	QUO4260
2265588	3063995	3782160	3172148	4302480	6956235			