SPACE BOUNDS FOR A GAME ON GRAPHS

by

Wolfgang J. Paul
Robert Endre Tarjan
James R. Celoni

STAN-CS-76-545
MARCH 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Space Bounds for a Game on Graphs

Wolfgang J. Paul */
Computer Science Department
Cornell University
Ithaca, New York 14850

Robert Endre Tarjan **/
Computer Science Department
Stanford University
Stanford, California 94305

James R. Celoni **/
Computer Science Department ,
Stanford University
Stanford, California 94305

Abstract.

We study a one-person game played by placing pebbles, according to certain rules, on the vertices of a directed graph. In [3] it was shown that for each graph with n vertices and maximum in-degree d , there is a pebbling strategy which requires at most c(d) n/log n pebbles. Here we show that this bound is tight to within a constant factor. We also analyze a variety of pebbling algorithms, including one which achieves the O(n/log n) bound.

Keywords: pebble game, register allocation, space bounds, Turing machines.

*/ Research partially supported by DAAD (German Academic Exchange Service) Grant No. 530/402/563/5.

**/ Research partially supported by National Science Foundation grant DCR72-03663-A03.

1. Introduction.

Let $G = (V,E)$ be an acyclic directed graph with vertex set V and edge set E. If (i,j) is an edge of G, we say i is a *predecessor* of j and j is a *successor* of i. We denote the set of predecessors of a vertex j by $B(j)$. The number of predecessors of a vertex is its *in-degree*; the number of successors of a vertex is its *out-degree*. A vertex of in-degree zero is a *source*; a vertex of out-degree zero is a *sink*. We denote by $\mathcal{G}(n,d)$ the class of all acyclic directed graphs G with n vertices, each having in-degree no more than d. We use $c, c_1, c_2, \ldots$ to denote positive constants. We use $c(d), c_1(d), c_2(d), \ldots$ to denote positive constants depending on d but not on n. Finally, we let $[i,j]$ 'denote the set of integers $\{k : i \leq k \leq j\}$.

In this paper we study a one-person game played on graphs. The game involves placing pebbles on the vertices of a graph $G \in \mathcal{G}(n,d)$ according to certain rules. A given step of the game consists of either placing a pebble on an empty vertex v of G (this is called *pebbling* v) or removing a pebble from a previously pebbled vertex. A vertex may be pebbled *only* if all its predecessors have pebbles. (As a special case of this rule, any source may be pebbled at any time.)

The object of the game is to pebble a given vertex of G subject to the constraint that at most a given number of pebbles are ever on the graph simultaneously. We may in addition require that this pebbling be accomplished in the minimum number of steps. We can pebble any vertex of G in n steps, using n pebbles, by pebbling the vertices in topological order*/ and never deleting pebbles. We are interested in pebbling methods which use fewer than n pebbles but possibly many more than n steps.

---

*/ I.e., if $(i,j) \in E$, then i is pebbled before j [4].

The number of pebbles used in the pebble game models the storage requirements of a computation, in the following intuitive sense. Each vertex represents a value. This value is computed by applying a particular operation to the values represented by the predecessors of the vertex. Sources represent input values. Each pebble represents a storage location. Pebbling a vertex corresponds to computing the value represented by the vertex and storing the value in the location represented by the pebble. Deleting a pebble from a vertex corresponds to freeing the storage location represented by the pebble, thus making unavailable the value represented by the vertex. Should this value be needed at a later time, it must be recomputed. The pebble game has been used as a model for register allocation problems [7] and as a tool for studying the relationship between time and space bounds for Turing machines [1,3].

Known results concerning the pebble game include the following.

Theorem A. If $G \in \mathcal{G}(n,d)$ and G has maximum out-degree one (i.e., G is a tree), then any vertex of G can be pebbled in time n using $c_2(d) \log n$ pebbles [1]. For infinitely many n , there is a graph $G \in \mathcal{G}(n,2)$ of maximum out-degree one which requires $c_1 \log n$ pebbles to pebble some vertex [5].

Theorem B [1]. For infinitely many n , there is a graph $G \in \mathcal{G}(n,2)$ which requires $c\sqrt{n}$ pebbles to pebble some vertex.

Theorem C [3]. If $G \in \mathcal{G}(n,d)$, then any vertex of G can be pebbled using $c_3(d)\, n/\log n$ pebbles.

In Section 2 we construct, for infinitely many n , a graph $G(n) \in \mathcal{G}(n,2)$ such that G(n) requires $c_1\, n/\log n$ pebbles to pebble some vertex. This shows that the bound in Theorem C is tight to within a constant factor. In Section 3 we give upper bounds for various -pebbling methods, including a method which achieves the $O(n/\log n)$ bound of Theorem C. In Section 4 we present some further remarks.

2. A Lower Bound.

We prove the claimed lower bound by recursively constructing an appropriate family of graphs. We use the following result of Valiant [8]. For any value of i, there is a graph with $c_1 2^i$ edges, 2^i sources, and 2^i sinks, which has the following property:

> For any $j \in [i,2^i]$, if S is any subset of j sources and T is any subset of j sinks, then there are j vertex-disjoint paths in C(i) from S to T .

The vertices in this graph may have arbitrary in-degree. Replacing each vertex with in-degree $d > 2$ by a binary tree with d leaves at most doubles the number of edges. In the new graph each vertex has in-degree two, and the graph still has the same property. Thus we have the following lemma.

Lemma 1. For any value of i there is a graph $C(i) \in \mathcal{G}(c2^i,2)$, with 2^i sources and 2^i sinks, such that: For any $j \in [1,2^i]$, if S is any subset of j sources and T is any subset of j sinks, then there are j vertex-disjoint paths in C(i) from S to T .

Corollary 1. For any $j \in [0,2^i-1]$, if j pebbles are placed on any j vertices of C(i), and T is any subset of at least $j+1$ sinks, then at least 2^i-j sources are connected to T via pebble-free paths.

Proof. For any $j \in [0,2^i-1]$, let j pebbles be placed on C(i) and let T be any subset of at least $j+1$ sinks. Any subset S of $j+1$ sources is connected to T via $j+1$ vertex-disjoint paths, at least one of which must be pebble-free. Thus j is the maximum size of the set of sources not connected to T by a pebble free path. □

Using copies. of C(i), we recursively define a set of graphs $\{G(i):\ i = 8, 9, 10, \ldots\}$. $G(8) = C(8)$. We form $G(i+1) = (V(i+1),E(i+1))$

from two copies of $G(i)$ and two copies of $C(i)$ as follows.. Let $G(i) = (V(i),E(i))$ have sources $S(i) = \{s(i,j): j \in [1,2^i]\}$ and sinks $T(i) = \{t(i,j): j \in [1,2^i]\}$. Let $C(i)$ have sources $SC(i) = \{sc(i,j): j \in [1,2^i]\}$ and sinks $TC(i) = \{tc(i,j): j \in [1,2^i]\}$. Let $G_1(i)$, $G_2(i)$ be two copies of $G(i)$ and let $C_1(i)$, $C_2(i)$ be two copies of $C(i)$. Let $S(i+1) = \{s(i+1,j): j \in [1,2^{i+1}]\}$ and $T(i+1) = \{t(i+1,j): j \in [1,2^{i+1}]\}$ be two new sets of vertices. Let $G(i+1) = (V(i+1),E(i+1))$, where

$$V(i+1) = S(i+1) \cup T(i+1) \cup V_1(i) \cup V_2(i) \cup VC_1(i) \cup VC_2(i) , \quad \text{and}$$

$$\begin{aligned} E(i+1) = {} & E_1(i) \cup E_2(i) \cup EC_1(i) \cup EC_2(i) \\ & \cup \{(s(i+1,j), t(i+1,j)): j \in [1,2^{i+1}]\} \\ & \cup \{(s(i+1,j), sc_1(i,j)): j \in [1,2^i]\} \\ & \cup \{(s(i+1,j+2^i), sc_1(i,j)): j \in [1,2^i]\} \\ & \cup \{(tc_1(i,j), s_1(i,j)): j \in [1,2^i]\} \\ & \cup \{(t_1(i,j), s_2(i,j)): j \in [1,2^i]\} \\ & \cup \{(t_2(i,j), sc_2(i,j)): j \in [1,2^i]\} \\ & \cup \{(tc_2(i,j), t(i+1,j)): j \in [1,2^i]\} \\ & \cup \{(tc_2(i,j), t(i+1,j+2^i)): j \in [1,2^i]\} . \end{aligned}$$

Figure 1 illustrates $G(i+1)$.

Let $m(i) = |S(i)| = |T(i)| = 2^i$. Let $n(i) = |V(i)|$. Then $n(8) \le c2^8$ and $n(i+1) \le 2n(i) + (2c+9)2^i$, where c is the constant given in Lemma 1. It is easy to prove by induction that $n(i) \le c_0 i 2^i$ for some constant c_0 , and that $G(i) \in \mathcal{G}(n(i),2)$.

Let $c_1 = 14/256$, $c_2 = 3/256$, $c_3 = 34/256$, and $c_4 = 1/256$. The following inequalities are immediate.

$$c_3 \; m(i)/2 \geq c_2 m(i+1)+1$$

$$(1-2c_2) > c_3$$

$$c_2 m(i) \geq c_4 m(i+1)+1$$

$$c_1 \; m(i)/2 \geq c_2 m(i+1)+1$$

$$(1-c_2) \geq c_1$$

$$(c_3/2 - c_2) \geq c_1 \; .$$

Lemma 2. To pebble at least $c_1 m(i)$ sinks of G(i) in any order,*/ starting from an initial configuration of no more than $c_2 m(i)$ pebbled vertices, requires a time interval $[t_1,t_2]$ during which at least $c_3 m(i)$ sources are pebbled and at least $c_4 m(i)$ pebbles are always on the graph.

Proof. By induction on i . Let i = 8 . Consider an initial configuration on G(8) of no more than three pebbled vertices and suppose 14 sinks are pebbled during time interval [0,t] . Any four of these sinks are connected, via initially pebble-free paths, to at least 253 sources, by Corollary 1. Thus at least one of these sinks, say v , is connected, via initially pebble-free paths, to at least 64 of the sources. When v is pebbled, none of the 64 sources is connected to v via a pebble-free path. Furthermore the set of sources connected to v via a pebble-free path can decrease by at most one at each time step. Let t_1-1 be the last time at which 64 sources are connected to v via

*/ i.e., each sink is pebbled at some time, but not all sinks need be pebbled simultaneously.

pebble-free paths. During time interval $[t_1,t]$, $63 \geq 34$ sources of $G(8)$ must be pebbled, while at least one pebble is always on the graph. This proves the lemma for $i = 8$.

Suppose the lemma is true for i. To prove the lemma for $i+1$, consider an initial configuration on $G(i+1)$ of no more than $c_2m(i+1)$ pebbled vertices and suppose at least $c_1m(i+1)$ sinks are pebbled during time interval $[0,t]$. We must consider several cases.

Case 1. There exists a time interval $[t_1,t_2] \subseteq [0,t]$ during which at least $c_3\ m(i)/2$ sources of $G_1(i)$ are pebbled and at least $c_2m(i)$ pebbles are always on the graph. The subgraph of $G(i+1)$ consisting of all vertices and edges on paths from the set of sources $\{s(i+1,j): j\epsilon[1,2^i]\}$ to the set of sinks $\{s_1(i,j): j\epsilon[1,2^i]\}$ satisfies Lemma 1 and Corollary 1. So does the subgraph of $G(i+1)$ consisting of all vertices and edges on paths from the set of sources $\{s(i+1,j+2^i): j\epsilon[1,2^i]\}$ to the set of sinks $\{s_1(i,j): j\epsilon[1,2^i]\}$. Let t_0 be the last time before t_1 at whichthereare no more than $c_2m(i+1)$ pebbles on the graph. At time t_0, since $c_3\ m(i)/2 \geq c_2m(i+1)+1$, there are at least $2(m(i) - c_2m(i+1)) = (1-2c_2)m(i+1) \geq c_3m(i+1)$ sources of $G(i+1)$ connected via pebble-free paths to the $c_3m(i)/2$ sources of $G_1(i)$ pebbled from t_1 to t_2. During the interval $[t_0,t_2]$, at least these sources of $G(i+1)$ must be pebbled, and at least $c_2m(i)-1 \geq c_4m(i+1)$ pebbles must be constantly on the graph. Thus the lemma holds in this case.

Case 2. There exists a time interval $[t_1,t_2] \subseteq [0,t]$ during which at least- $c_3\ m(i)/2$ sources of $G_2(i)$ are pebbled and at least $c_2m(i)$ pebbles are always on the graph. The lemma holds by a proof like that in Case 1.

Case 3. There exists a time interval $[t_1,t_2] \subseteq [0,t]$ during which at least $c_1\ m(i+1)/2$ sinks of $G(i+1)$ are pebbled and at least $c_2m(i)$ pebbles are always on the graph. During $[t_1,t_2]$ either $c_1\ m(i+1)/4$ sinks in $\{t(i+1,j): j \in [1,2^i]\}$ are pebbled or $c_1\ m(i+1)/4$ sinks in $\{t(i+1,j+2^i): j \in [1,2^i]\}$ are pebbled. The lemma holds by a proof like that in Case 1, using the inequalities $c_1\ m(i+1)/4 \geq c_2m(i+1)+1$, $(1-2c_2) \geq c_3$, and $c_2m(i)-1 \geq c_4m(i+1)$.

Case 4. None of the previous cases hold. Since Case 3 does not hold, there must be a time $t_1 \in [0,t]$ such that fewer than $c_1\ m(i+1)/2$ sinks of $G(i+1)$ are pebbled during $[0,t_1]$ and the number of pebbles on $G(i+1)$ at time t_1 is no more than $c_2m(i)$. During $[t_1,t]$, at least $c_1m(i)$ sinks of $G(i+1)$ are pebbled. Since $c_1\ m(i)/2 \geq c_2m(i+1)+1 \geq c_2m(i)+1$, the number of sinks of $G_2(i)$ connected to these sinks of $G(i+1)$ via pebble-free paths is at least $(1-c_2)m(i)$. Thus at least $(1-c_2)m(i) \geq c_1m(i)$ sinks of $G_2(i)$ are pebbled during $[t_1,t]$, starting from an initial configuration of no more than $c_2m(i)$ pebbled vertices. By the induction hypothesis there is a time interval $[t_2,t_3] \subseteq [t_1,t]$ during which $c_3m(i)$ sources of $G_2(i)$ are pebbled and $c_4m(i)$ pebbles are always on $G_2(i)$.

Since Case 2 does not hold, there must be a time $t_4 \in [t_2,t_3]$ such that fewer than $c_3\ m(i)/2$ sources of $G_2(i)$ are pebbled during $[t_2,t_4]$ and the number of pebbles on $G(i+1)$ at time t_4 is no more than $c_2m(i)$. During $[t_4,t_3]$ at least $c_3\ m(i)/2$ sources of $G_2(i)$ are pebbled. At time t_4 , since $c_3\ m(i)/2 - c_2m(i) \geq c_1m(i)$, at least $c_1m(i)$ sinks of $G_1(i)$ are connected via pebble-free paths to these sources of $G_2(i)$. During $[t_4,t_3]$ these sinks of $G_1(i)$ must be pebbled, starting with no more than $c_2 m(i)$ pebbled vertices. By the induction hypothesis there is a time interval $[t_5,t_6] \subseteq [t_4,t_3]$ during which $c_3 m(i)$ sources of $G_1(i)$ are pebbled and $c_4 m(i)$ pebbles are always on $G_1(i)$.

Since Case 1 does not hold, there must be a time $t_7 \in [t_5,t_6]$ such that fewer than $c_3\, m(i)/2$ sources of $G_1(i)$ are pebbled during $[t_5,t_7]$ and the number of pebbles on $G(i+1)$ at time t_7 is no more than $c_2 m(i)$. During $[t_7,t_6]$ at least $c_3\, m(i)/2$ sources of $G_1(i)$ are pebbled. At time t_7 , since $c_3\, m(i)/2 \geq c_2 m(i+1)+1 \geq c_2 m(i)+1$, at least $(1-2c_2)m(i+1) \geq c_3 m(i+1)$ sources of $G(i+1)$ are connected via pebble-free paths to these sources of $G_1(i)$. Thus, during $[t_7,t_6] \subset [t_5,t_6] \subseteq [t_2,t_3]$ at least $c_3 m(i+1)$ sources of $G(i+1)$ are pebbled and at least $c_4 m(i) + c_4 m(i) = c_4 m(i+1)$ pebbles are always on the graph. This completes the proof. □

<u>Theorem 1</u>. For infinitely many n , there is a graph $G \in \mathcal{G}(n,2)$ such that pebbling some vertex in G requires $c_5\, n/\log n$ pebbles.

<u>Proof</u>. For $n = n(i)$, $i = 8, 9, 10, \ldots$ let $G = G(i)$ .Since pebbling all sinks of $G(i)$ from an initial configuration of no pebbled vertices requires $c_4 m(i)$ pebbles, there must be <u>some</u> sink whose pebbling requires $c_4 m(i)$ pebbles. (Otherwise, the procedure of pebbling the sinks one after another using a minimum number of pebbles for each sink, and removing all pebbles after each sink is pebbled, would pebble all sinks using fewer than $c_4 m(i)$ pebbles.) Since $n(i) \leq 2c_0\, i\, 2^i$ and $m(i) = 2^i$, the number of pebbles required is $c_5\, n(i)/\log n(i)$ for some constant c_5 . □

3. Upper Bounds.

In this section we derive upper bounds on the number of pebbles required by various pebbling methods. Let remove (set S) be a procedure which removes all pebbles from vertices in . . Most of our results depend upon the following algorithm, which pebbles vertices in a "depth-first" manner.

```
procedure depth-first pebble (graph G, vertex v, set S);
    begin
        for u ∈ B(v) do if u not pebbled then
            depth-first pebble (G, u, B(v) ∪ S);
        pebble v;
        remove (V-(W(v)));
    end;
```

The following lemma is implicit in [3].

Lemma 3. If $G = (V,E) \in \mathcal{G}(n,d)$ has the property that any path to v has no more than ℓ vertices, then depth-first pebble $(G,v,\emptyset)$ pebbles v using no more than $(d-1)(\ell-1)+2$ pebbles.

Proof-. By induction on the length ℓ of the longest path to v . If v is a source, the procedure uses $1 \le (d-1)\cdot 0+2$ pebbles. Suppose the lemma is true for ℓ and let the longest path to v have length $\ell+1$. Then the procedure uses $\max\{(d-1)+(d-1)(\ell-1)+2,\ d+1\} = (d-1)\ell+2$ pebbles. □

The following more general method uses "permanent" pebbles, which once placed on the vertices of a set P , are never removed.

```
procedure permanent pebble (graph G, vertex v, set P);
    begin
        for u∈P in topological order do depth-first pebble (G,u,P);
        depth-first pebble (G,v,P);
    end;
```

Lemma 4. If $|P| = k$ and if $G = (V,E) \in \mathcal{G}(n,d)$ has the property that any path to v which avoids vertices in P contains no more than ℓ vertices, then permanent pebble (G,v,P) pebbles v using no more than $k+(d-1)(\ell-1)+2$ pebbles.

Proof. When depth-first pebble (G,u,P) is called by permanent pebble, any pebble-free path to u contains no more than ℓ vertices, since every vertex in P on a path to u has been pebbled previously. The bound follows from Lemma 3. □

Erdös, Graham, and Szemerédi [2] have proved that in any acyclic directed graph of n edges there is a subset P of $c_1 n \log \log n/\log n$ vertices such that every path which avoids P has length at most $c_2 n \log \log n/\log n$. (Furthermore they provide an easy way to find such a subset P.) Their result combines with Lemma 4 to give the following theorem.

Theorem 2. If $G = (V,E) \in \mathcal{G}(n,d)$ and $P \subseteq V$ is properly chosen, then permanent pebble (G,v,P) uses at most $cdn \log \log n/\log n$ pebbles.

To come closer to the Theorem C bound, we must use an algorithm somewhat more complicated than permanent pebble. We defer discussion of this algorithm to the end of the section.

Theorem 1 and Lemma 4 also yield:

Corollary 2 [2]. For infinitely many n there is a graph $G \in \mathcal{T}(n,2)$ such that every subset P of the vertices of G with the property "Every path which avoids P has length at most $|P|$" has at least $c_1 n/\log n$ vertices.

We now give good methods for pebbling two special classes of graphs. We call $G = (V,E) \in \mathcal{G}(n, d)$ a level graph if V can be partitioned into levels $L(1),L(2),\ldots,L(m)$ such that if $(v,w) \in E$ and $v \in L(i)$, then $w \in L(i+1)$. Let G be a level graph and let k be any positive integer. Call level i large if $|L(i)| \geq k$ and small otherwise. Let $\{i(j): 1 \leq j \leq \ell\}$ be the set of indices of small levels, in increasing order. Let $i(0) = 0$, $i(\ell+1) = m+1$, and $L(0) = \emptyset$. Let $v \in L(j)$ be any vertex and let ℓ' be the integer such that $i(\ell') < j$ and $i(\ell'+1) \geq j$.

The following algorithm efficiently pebbles v.

```
procedure level pebble (graph G, set array L, array i, vertex v, integer ℓ');
    begin
        for j := 1 until ℓ' do
            begin
                for u ∈ L(i(j)) do
                    depth-first pebble (G, u, L(i(j-1)) ∪ L(i(j)));
                remove (L(i(j-1)));
            end;
        depth-first pebble (G,v,L(i(ℓ')));
    end;
```

Lemma 5. Procedure level pebble pebbles any vertex of a level graph $G = (V,E) \in \mathcal{G}(n,d)$ using no more than $2k + (d-1)\frac{n}{k}$ pebbles.

Proof. During the pebbling process, no more than two small levels ever contain pebbles simultaneously. Thus at most $2k-2$ pebbles are ever on small levels. The number of levels between two small levels is at most $\frac{n}{k}$, since there are at most $\frac{n}{k}$ large levels. Thus the number of pebbles used in an outermost call of depth-first pebble is at most $(d-1)\frac{n}{k} + 2$, and the total number of pebbles used is at most $2k + (d-1)\frac{n}{k}$. □

Theorem 3. If $G \in \mathcal{B}(n,d)$ is a level graph, any vertex of G can be pebbled using $\sqrt{8(d-1)n}$ pebbles.

Proof. Immediate from Lemma 5, choosing $k = \sqrt{(d-1)\ n/2}$. □

The bound in Theorem 3 is tight (to within a constant factor which depends on d), because the graphs Cook used to prove Theorem B are level graphs.

The class of m-tape Turing machine graphs $\mathcal{T}(n,m)$ is the subset of $\mathcal{B}(n,m+1)$ containing graphs $G = (V,E)$ of the following type.

$$V = \{(i,j_1(i),\ldots,j_m(i)) \colon 1 \le i \le n\ ,\ j_k(1) = 1 \text{ for all } k\ ,$$
$$\text{and } j_k(i+1) \in (j,(i)-1,\ j,(i)\ ,\ j_k(i)+1\} \text{ for all } i < n$$
$$\text{and all } k)\ ;$$

$$E = \bigcup_{k=1}^{m} \{((i',j_1(i'),\ \bullet\ J,(W\ ,(i,j_1(i),\ \bullet\ J,(i>>)\colon$$
$$i' = \max\{\ell < i\colon j_k(\ell) = j,(i)))$$
$$\cup \{((i,j_1(i),\ldots,j_m(i))\ ,\ (i+1,j_1(i+1),\ldots,j_m(i+1)))\colon 1 \le i < n)\ .$$

The pebble game on m-tape Turing machine graphs was used in [1,3] as a tool for relating the time and space requirements of Turing machines. It has been conjectured that there are graphs in $\mathcal{T}(n,1)$ which can require $cn/\log n$ pebbles to pebble some vertex. We disprove this conjecture by adapting a proof of Paterson for space-efficient simulation of one-tape Turing machines [6].

Let $G = (V,E) \in \mathcal{T}(n,1)$. For any j , let $H(j) = \{(i,j_1(i)) \in V\colon j,(i) = j\}$. For any $S \subseteq V$, let $\underline{\text{width}}(S) = \max\{|j_1(i) - j_1(i')|\colon (i,j_1(i))\ ,\ (i',j_1(i')) \in S)$. For any $S \subset V$, there must be some j such that $\max\{|j_1(i)-j| : (i,j_1(i)) \in S\} \leq \frac{2}{3}$ width(S)+1 and $|H(j)| \leq 3n/$width(S) . Removal of the vertices in $H(j)$ splits S into two parts, $S_1 = \{(i,j_1(i)) \in S: j_1(i) < j\}$ and $S_2 = \{(i,j_1(i)) \in S: j_1(i) > j\}$. Any path in G which contains a vertex in S_1 and a vertex in S_2 must contain an intervening vertex in $H(j)$.

The following recursive algorithm efficiently pebbles a vertex v in $G = (V,E) \in \mathcal{T}(n,1)$.

```
Bone-tape pebble (graph G, set S, vertex v);
  if width(S) < k then
    begin
      for (i,j1(i)) ∈ S in topological order while i ≤ v do
        begin
          pebble (i,j1(i));
          let (i',j1(i')) be the vertex (if any) in S with
            largest i' < i and j1(i') = j1(i);
          remove pebble from (i',j1(i'));
        end;
      remove (S-{v});
    end
  else
    begin
      set S1, S2;
      find j such that max{|j1(i)-j| : (i,j1(i)) ∈ S} ≤ 2/3 width(S)+1
        and |H(j)| ≤ 3n/width(S);
      S1 := {(i,j1(i)) ∈ S: j1(i) < j};
```

```
    S2  := {(i,j1(i)) εS:  j,(i) >j);
    for (i,j1(i)) ε H(j) in topological order do
       begin
          if (i-1,j1(i-1)) ε S1 then one-tape pebble
             (G,S1, (i-1,j1(i-1)))
          else if (i-1,j1(i-1)) ε S2 then one-tape pebble
             (G,S2,(i-1,j1(i-1)));
          pebble (i,j1(i));
       end;
    if vεS1 then one-tape pebble (G, S1, v)
       else if vεS2 zone-tape pebble (G,S2,v);
    remove (S-(v)) ;
end one-tape pebble;
```

Lemma 6. Procedure one-tape pebble pebbles any vertex of a graph $G = (V,E) \in \mathcal{T}(n,1)$ using $\frac{9n}{k} + k$ pebbles.

Proof. Let $p(n,x)$ denote the number of pebbles used by one-tape pebble (G,S,v) when $G = (V,E) \in \mathcal{T}(n,1)$ and $S \subset V$ has width$(S) = x$. Then

$$p(n,x) \leq k \quad \text{if } x < k \text{, and}$$

$$p(n,x) \leq p\left(n, \left\lfloor \frac{2}{3} x \right\rfloor\right) + \frac{3n}{x} \qquad \text{if } x \geq k .$$

Let x be such that $x < k$, $\frac{3}{2} x \geq k$. Then

$$p\left(n, \left(\frac{3}{2}\right)^j x\right) \leq \sum_{i=1}^{j} \frac{3n}{\left(\frac{3}{2}\right)^i x} + k$$

$$\leq \frac{3n}{\left(\frac{3}{2}\right)x} \sum_{i=0}^{j-1} \left(\frac{2}{3}\right)^i + k$$

$$\leq \frac{9n}{k} + k \qquad \text{for any positive } j .$$

The maximum number of pebbles required to pebble any graph in $\mathcal{T}(n,1)$ is no more than $p(n,n) \leq \frac{9n}{k} + k$. □

Theorem 4. If $G \in \mathcal{T}(n,1)$, any vertex of G can be pebbled using $6\sqrt{n}$ pebbles.

Proof. Immediate from Lemma 6, choosing $k = 3\sqrt{n}$. □

Cook's graphs can be embedded in one-tape Turing machine graphs with an increase of only a constant factor in the number of vertices. Thus the Theorem 4 bound is tight to within a constant factor. By modifying the construction of Section 2, we can show that two-tape Turing machine graphs require $cn/(\log n)^2$ pebbles in the worst case, and we believe but cannot prove that this lower bound can be improved to $cn/\log n$.

The last result of this section is an algorithm, based on the proof of Theorem C in [3], which efficiently pebbles any vertex of an arbitrary graph. The algorithm is recursive and operates on a graph G in the following manner. If G is small, the vertices of G are pebbled in topological order without removing pebbles. If G is large, G is split into two parts, G_1 and G_2 , of roughly the same number of edges, such that no edges run from G_2 to G_1 . If a vertex in G_1 is to be pebbled, the method is applied recursively to G_1 . If a vertex in G_2 is to be pebbled and the number of edges from G_1 to G_2 is small (i.e., less than $\ell/\log \ell$, where ℓ is the number of edges in G), then the vertices in G_1 with successors in G_2 are permanently pebbled by applying the method recursively to G_1 and all pebbles except the permanent ones are removed. Then the vertex in G_2 is pebbled by applying the method recursively to G_2 . If a vertex in G_2 is to be pebbled and the number of edges from G_1 to G_2 is large, the algorithm is applied recursively to G_2 . Whenever the next vertex v to be pebbled in G_2 has some predecessors $u_1,\dots,u_k$ in G_1 , the algorithm is applied recursively to G_1 to pebble $u_1,\dots,u_k$ and all pebbles in G_1 but the ones on $u_1,\dots,u_k$ are removed. After v is pebbled, all pebbles are deleted from G_1 , and the method continues on G_2.

This algorithm is given more precisely below. The parameter $\mathcal{S}$ is the partition of the vertex set V of G created by nested recursive calls of the procedure. Set T gives a set of vertices which, once pebbled, are not to be unpebbled during the current recursive call of the procedure. Integer k is some suitable positive constant. The procedure call <u>best pebble</u> (G, $\{V\},v,\emptyset)$ will pebble vertex v in graph $G = (V,E) \in \mathcal{G}(n,d)$.

```
procedure best pebble (graph G, partition 𝒮, vertex v, set T);
    begin
        find S∈𝒮 such that v∈S;
        ℓ := |{(u,w): u,w ∈ S}|;
        if ℓ < k then
            begin
                for u∈B(v) do if u not pebbled then best pebble (G,S,u,T ∪ B(v));
                pebble v;
                remove (V- (T ∪{v}));
            end
        else
            begin
                divide S into S1 , S2 such that (u,w) ∈ E and u ∈ S2
                    implies w ∈ S2 and ℓ/2 -d ≤ |{(u,w): u∈S1}| ≤ ℓ/2 + d;
                if |{(u,w): u∈S1 , w∈S2}| < ℓ/log ℓ then
                    begin
                        set C;
                        C := (u | ∃(u,w) with u∈S1, w∈S2};
                        for u∈C do if u not pebbled then
                            best pebble (G , 𝒮-{S} ∪ {S1,S2} , u , T ∪ C);
                        best pebble (G , 𝒮-{S} ∪ {S1,S2} , v , T ∪ C);
                        remove (V - (T U(v)));
                    end
                else best pebble (G , 𝒮-{S} ∪ {S1,S2} , v , T);
    end end best pebble;
```

Theorem 5. Procedure best pebble pebbles any vertex of a graph $G = (V,E) \in \mathcal{G}(n,d)$ using $c(d)\ n/\log n$ pebbles.

Proof. Let $q(m)$ be the maximum number of pebbles used by best pebble to pebble any vertex in any graph with m or fewer edges and maximum in-degree d. Then

$$q(m) \leq k \qquad \text{if } m \leq k \ ,$$

$$q(m) \leq \max\left\{q\left(\frac{m}{2} + d\right) + \frac{m}{\log m}\ ,\ 2q\left(\frac{m}{2} + d - \frac{m}{\log m}\right)\right\} \text{ if } m > k \ .$$

It is easy to show by induction that $q(m) \leq c\ m/\log m$ for a suitable positive constant c . The theorem follows. ☐

4. Remarks.

Theorem 1 gives a lower bound of $cn/\log n$ for the number of pebbles necessary to pebble every graph in $\mathcal{G}(n,2)$. This result implies that the upper bound in Theorem C is tight to within a constant factor. The result also shows that the space-efficient simulation for multi-tape Turing machines given in [3] cannot be improved without using new techniques.

Many questions about the pebble game remain unanswered and several application areas remain to be explored. For instance, how much time must one sacrifice to achieve a given savings in pebbles? How many pebbles can be saved while preserving a polynomial running time? How much time can be saved while preserving a $cn/\log n$ pebble bound?

A possible application area lies in the derivation of lower bounds on the time necessary for various computations. For instance, suppose we wish to prove a lower bound of $cn \log n$ on the size of a Boolean circuit necessary to do some computation. If we can prove that any circuit either has size $c_1 n \log n$ or requires simultaneous storage of $c_2 n$ intermediate results, the bound follows from Theorem C.

Acknowledgment. For helpful and inspiring discussions we thank Professors V. Claus and K. Mehlhorn.

References

[1] S. A. Cook, "An observation on time-storage trade off," Proceedings of the Fifth Annual ACM Symp. on Theory of Computing (1973), 29-33.

[2] P. Erdös, R. L. Graham, and E. Szemerédi, "On sparse graphs with dense long paths," STAN-CS-75-504, Computer Science Department, Stanford University (1975).

[3] J. Hopcroft, W. Paul, and L. Valiant, "On time versus space and related problems," Sixteenth Annual Symp. on Foundations of Computer Science (1975), 57-64.

[4] D. Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, Mass. (1968), 258-265.

[5] M. S. Paterson and C. E. Hewitt, "Comparative schematology," Record of Project MAC Conf. on Concurrent Systems and Parallel Computation (1970), 119-128.

[6] M. S. Paterson, "Tape bounds for time-bounded Turing machines," Journal of Comp. and Sys. Sci. 6 (1972), 116-124.

[7] R. Sethi, "Complete register allocation problems," Proceedings of the Fifth Annual ACM Symp. on Theory of Computing (1973), 182-195.

[8] L. Valiant, "On non-linear lower bounds on computational complexity," Proceedings of the Seventh Annual ACM Symp. on Theory of Computing (1975), 45-53.

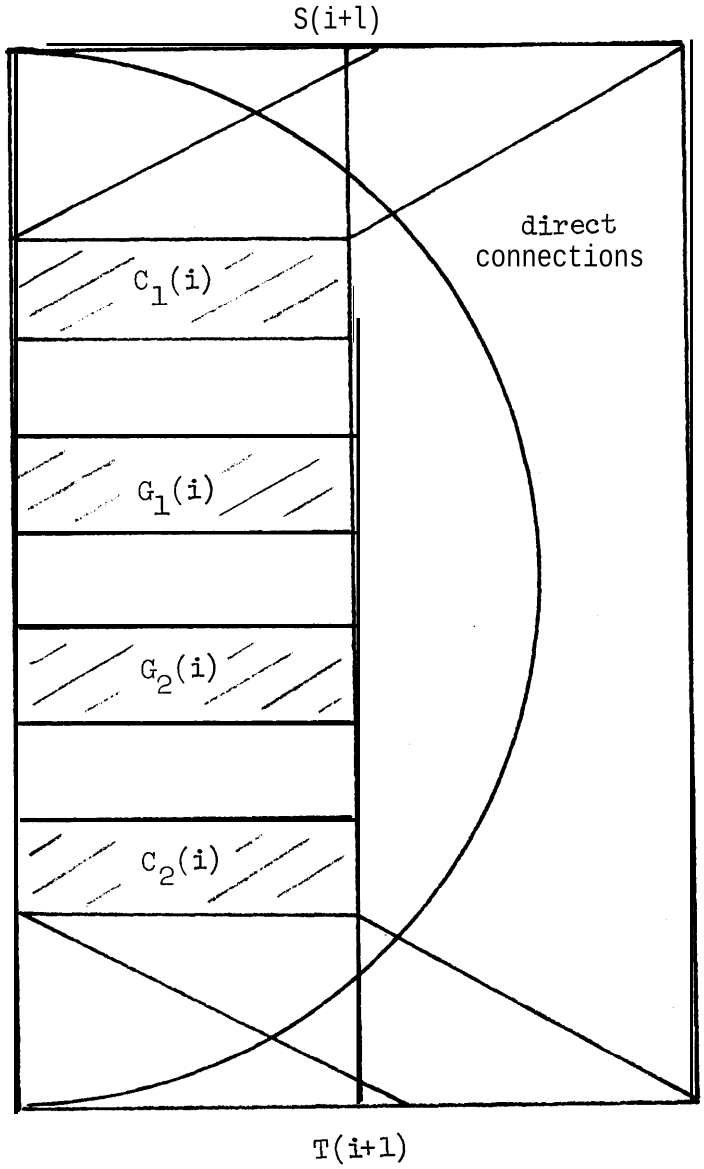

Figure 1. $G(i+1)$.