Stanford University Computer Science Department
Solved CS347 Spring 2001 Mid-term.

Question 1: (4 points)

Shown below is a portion of the positional index in the format

term: doc1: position1,position2 …; doc2: position1, position2 …; etc.

angels:	*2*:36,174,252,651;	*4*:12,22,102,432;	*7*:17;
fools:	*2*:1,17,74,222;	*4*:8,78,108,458;	7:3,13,23,193;
fear:	*2*:87,704,722,901;	*4*:13,43,113,433;	*7*:18,328,528;
in:	*2*:3,37,76,444,851;	*4*:10,20,110,470,500;	*7*:5,15,25,195;
rush:	*2*:2,66,194,321,702;	*4*:9,69,149,429,569;	*7*:4,14,404;
to:	*2*:47,86,234,999;	*4*:14,24,774,944;	*7*:199,319,599,709;
tread:	*2*:57,94,333;	*4*:15,35,155;	*7*:20,320;
where:	*2*:67,124,393,1001;	*4*:11,41,101,421,431;	*7*:16,36,736;

Which document(s) if any meet each of the following queries, where each expression within quotes is a phrase query?

(i) ***"fools rush in"***

Answer:
All three documents (2, 4, and 7) satisfy the query. (2 points for a right answer, 1 point for a partially correct answer such as 2 and 4)

(ii) ***"fools rush in"*** *AND* ***"angels fear to tread"***

Answer:
Only document 4.

Question 2: (1+1+3 points)

(i) How many postings entries are generated in the bigram index for the following text (underlined portion only): *Far from the madding crowds ignoble strife* ?

Answer:
There are 43 bigrams in the sentence. 39 of them are distinct, so the bigram index would consist of 39 entries. We accepted either 39 or 43 as the answer. (Those who explicitly said 'F' was different from 'f' also got credit for 40)

(ii) For this bigram index, how would the wild-card query ***madd*ing*** be expressed as an *AND* query?

Answer:
$m AND ma AND ad AND dd AND in AND ng AND g$

No other answers were accepted.

(iii) Consider a corpus with 10 million documents, with an average of 1000 words each. The average word length across the corpus is 4 characters, not counting spaces or punctuation. Estimate the number of trigram occurrences for this corpus.

Answer:
Consider one document: there are on average 1,000 words of an average of 4 characters each, with around 1,000 word boundaries (spaces between words and beginning/end of the document). So each document contains roughly 1,000 x 4 + 1000 = 5,000 trigrams (not necessarily distinct). Thus the corpus contains 5,000 x 10M = 50 billion trigrams (not necessarily distinct). We also accepted 40 billion trigrams, because some students assumed 'hello world' would not generate the 'o$w' trigram. We did **not** accept 60 billion, because double counting that occurrence 'o$w' is incorrect (-1 pt for 60 billion)

Question 3: (5 points)
Recall the estimate of the total size of the postings entries (45Mbytes using γ codes) from Lecture 1 using Zipf's law. Using the same parameters (1 million documents, 500,000 terms), re-compute this estimate if we were to omit from indexing the 1% of the most frequently occurring terms.

Answer:
This problem can be solved using the equations given in the Homework 1 solution:

$$2n\sum_{i=13}^{19} i + n\sum_{13}^{19} 1 = 217 \text{ x n bits} = 217 \text{ Mb} = 27.125 \text{ MB}$$

This is the upper bound. An approximate lower bound is

$$2n\sum_{i=14}^{19} i + n\sum_{14}^{19} 1 = 192 \text{ x n bits} = 192 \text{ Mb} = 24 \text{ MB}$$

Yes, it's i=13 and i=14, because (referring to the HW solution), k ranges from $[2^{i-1},2^{i})$. We gave credit for those who used i=12 and i=13 though. (-1 pt for minor error, -2 pt for substantial error, -3 pt for describing how to solve without properly solving)

Question 4: (5 points)
Mark each of the following assertions as True or False.

	Assertion	T/F
(i)	The optimal order for query processing in an *AND* query is always realized by starting with the term occurring in the fewest documents.	False
(ii)	The γ code for 17 is 111000001.	False
(iii)	The base of the logarithm used in the tf×idf formula makes no difference to the cosine distance between two documents (provided they both use the same base).	True
(iv)	If we were to take a document and double its length by repeating	

	every occurrence of every word, then the normalized tf×idf values for all terms in this document remain unchanged.	True
(v)	The tf x idf value for a term t in a document D is non-zero if and only if there is at least on occurrence of t in D.	False

Explanation:

(i) Counter example: for query "x AND y AND z", suppose ((x AND y) AND z) is the processing order based on frequency counts. It is possible that (x AND y) is non-empty whereas (y AND z) is in fact empty. Then, using the order ((y AND z) AND x), we can altogether avoid reading the postings list for x.

(ii) The correct code is 111100001.

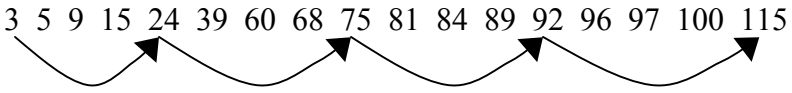
(iii) A different base for the logarithm only changes the length of the document vectors, not the angle between them. Therefore the cosine distance, which is computed after normalizing the vectors, remains unchanged.

(iv) Clearly, idfs do not change as they depend only on the number of documents containing each word. Also, since every occurrence of every word is repeated, tf, does not change. Hence, the normalized tf×idf values remain unchanged.

(v) It is true that if tf×idf value is non-zero, then there is at least one occurrence of t in D. However, the converse is false, since idf can be 0 if t occurs in every document in the corpus.

Question 5: (3+1+2 points)

(i) Consider the following postings list augmented with skip pointers at a uniform skip size of 4.



The entries in the list are NOT gap encoded; they directly represent document IDs. At some stage in processing an AND query, we need to merge the entries in this postings list with the entries in the candidate list (3,5,89,95,97,99,100,101). We define the following four operations:

- **Compare(A,B):** compare entry A in postings list with entry B in the candidate list
- **Output(X):** output value X as a result of the merge process
- **LookAheadTo(X):** peek ahead to take a look at the target X of a skip pointer
- **SkipTo(X):** follow a skip pointer to reach entry X

a) In performing this merge, list the instances of Output, LookAheadTo, and SkipTo operations in the order in which they would be executed. (Note: Do not list instances of Compare).

Answer: This question has a number of possible solutions, depending on the details of the merging algorithm. For example, at skip points on the postings list, one could first look ahead and then compare with the current postings entry or do the operations in the reverse order. Similarly, one could either discard or remember the last lookahead value (and the associated pointer) between successive skip points. Full points were awarded for any consistent order of operations accompanied by clearly stated assumptions, if any. Below are two possible answers:

Case 1: The last look ahead value is discarded; At skip points, lookahead before operating on the current postings entry

LookAheadTo(24)
Output(3)
Output(5)
LookAheadTo(75)
SkipTo(75)
LookAheadTo(92)
Output(89)
LookAheadTo(115)
Output(97)
Output(100)

Case 2: The last look ahead value and pointer are stored; At skip points, look ahead before operating on current postings entry

LookAheadTo(24)
Output(3)
Output(5)
SkipTo(24)
LookAheadTo(75)
SkipTo(75)
LookAheadTo(92)
Output(89)
SkipTo(92)
LookAheadTo(115)
Output(97)
Output(100)

Common error: Computing the union, instead of intersecting the postings and candidate lists

b) How many Compare operations would be executed? (Note: Do not list the actual instances).

Answer: Here the answer is obviously a function of the assumptions made for Part (a).

For instance, for Case 1, there are **19** compares:
(24 3) (3 3) (5 5) (9 89) (15 89) (24 89) (75 89) (92 89) (81 89) (84 89) (89 89) (115 95) (92 95) (96 95) (96 97) (97 97) (100 99) (100 100) (115 101)

For Case 2, there are **21** compares:
(24 3) (3 3) (24 5) (5 5) (24 89) (75 89) (92 89) (81 89) (84 89) (89 89) (92 95) (115 95) (96 95) (115 97) (96 97) (97 97) (115 99) (100 99) (115 100) (100 100) (115 101)

Common error: For Case 2, not counting the compares between the last look ahead value and the current value in the candidate list

(ii) Suppose we wish to gap-encode postings list containing skip pointers. One approach is to store all entries in the list as gaps, except for targets of the skip pointers which will be stored as absolute values. For example, in the postings list of Part (i), 24, 75, 92, and 115 will be stored as is, whereas the remaining entries will be gap-encoded, yielding (3 2 4 6 24 15 21 .... 4 1 3 115). Suppose (7 8 9 39 28 60 130 70 10 215) represents a gap-encoded postings list with skip pointers at **skip size 3**. What will be the result of merging this postings list with the candidate list (9 24 127 135 210)?

Answer: (24 127 210)

(7 8 9 39 28 60 130 70 10 215) decodes to (7 15 24 39 67 127 130 200 210 215). Intersecting this with the candidate list, we get (24 127 210).

1 point was awarded as long as the postings list was decoded correctly, even if the final answer was incorrect.

Question 6: (5 points)

(i) Represent the following simplified graph of the web as a Markov chain by providing the corresponding transition probability matrix. Assume teleportation to a random page (including the start page) occurs with **50%** probability.

Answer:

$$0.5 \text{ x } \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0.5 & 0.5 & 0 \end{bmatrix} + 0.5 \text{ x } \begin{bmatrix} 0.333 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.333 \end{bmatrix} = \begin{bmatrix} 0.167 & 0.667 & 0.167 \\ 0.167 & 0.167 & 0.667 \\ 0.417 & 0.417 & 0.167 \end{bmatrix}$$

(this was worth 3 pts. –1 pt for each error)

(ii) Using the initial probability vector [0 1 0], carry forward the Markov chain 1 time step. (I.e., give the probability vector for time t = 1)

Answer:

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \text{ x } \begin{bmatrix} 0.167 & 0.667 & 0.167 \\ 0.167 & 0.167 & 0.667 \\ 0.417 & 0.417 & 0.167 \end{bmatrix} = \begin{bmatrix} 0.167 & 0.167 & 0.667 \end{bmatrix}$$

(this was worth 2 pts. Mostly all or nothing)