CS340 Winter 1995
Ming Li

CS340 Final Examination: Model Solutions

- (15 marks)

1. To sort three elements, one needs to make exactly three comparisons in the worst case.

   True. The first comparison gives $a < b$. Next we can do no better than comparing c with a. After finding $a < c$, one more comparison is needed between b and c.

2. $lim_{n\to\infty} \frac{(\log n)^{100}}{n} \to \infty$.

   False; $polylog(n) = o(n)$.

3. Give the name of a data structure which implements a dictionary (with insert, delete, and search operations) and guarantees $O(\log n)$ time for each of the three operations. Here n is the number of keys in the dictionary.

   An AVL-tree or a B-tree.

4. $-n^2$ is $O(n)$.

   False. Big-Oh notation concerns growth in *absolute* value. Since this question is outside the scope of the book, no marks have been subtracted for wrong answers.

5. Suppose you are asked to implement a sorting program that will always sort at most 10 keys, which algorithm will you use?

   Insertion sort is simplest to code and runs faster in practice on only 10 elements.

6. In the Lempel-Ziv compression algorithm, the full dictionary is always kept even after the compression process, since otherwise we would not be able to do the decoding or uncompression.

   False; the dictionary is reconstructed during decoding.

7. Suppose you are asked to implement a general sorting program that must *always* run very fast, which algorithm will you use?

   Only Heapsort has a worst-case $O(n \log n)$ bound.

8. Define a *funny-tree* to be a binary tree such that for each of its nodes x, the number of nodes in each subtree of x is at most $2/3$ the number of nodes in the tree rooted at x. Draw the tallest funny-tree of 5 nodes.

   Root with path of 3 nodes as left subtree and single node as right subtree.

9. What is the height of a funny-tree, defined in the previous question, on n nodes in Θ notation?

   $\Theta(\log n)$

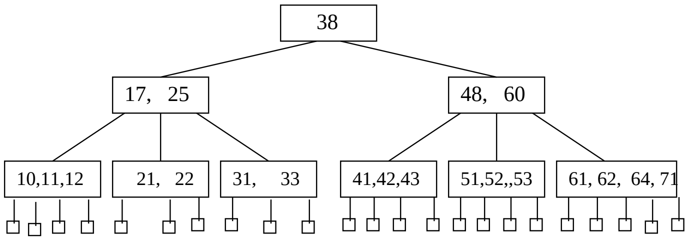

Figure 1: A B-tree of order 5

- (15 marks) Consider the following B-tree of order 5 in Figure 1. (Order of a B-tree is defined as max branching factor.) Draw the resulting B-trees after the following operations.

  (a) Insert 65 to the B-tree in Figure 1.

  Split $(61, 62, 64, 65, 71)$ into $(61, 62)$ and $(65, 71)$. Move 64 up into $(48, 60, 64)$.

  (b) Delete 21 from the (original) B-tree in Figure 1.

  Borrow from left to make $(10, 11)$ and $(17, 22)$ as first two children of $(12, 25)$.

  (c) Delete 33 from the (original) B-tree in Figure 1. (Assume you can only "borrow" from the immediate neighbors.)

  Merge into $(21, 22, 25, 31)$ and top three nodes into $(17, 38, 48, 60)$.

- (10 marks) Hashing.

  (a) Describe the concept of secondary clustering in hashing.

  Secondary clustering refers to the forming of large fully occupied regions in the case of open addressing with linear probing. The larger such a region, the more likely it grows.

  (b) Describe the scheme of double hashing.

  Using two hashing functins h_1 and h_2 to probe positions $h_1(x)+k*h_2(x)$ with $k = 0, 1, 2...$

  (c) In double hashing, we require the table size to be a prime number. Why?

  If the table size m is a prime, then $gcd(m, h(x)) = 1$, so that we are guaranteed to probe all positions of the table.

  (d) Define the term "load factor".

  Load factor is the ratio of number of keys to hash table size.

  (e) Give a reasonable hash function.

  Key modulo hash table size.

- (10 marks)

  1. How many AVL trees of 7 nodes with keys 1,2,3,4,5,6,7 are there?

     17. One with 4 as root, and 2^4 with 3 or 5 as root. E.g. root of 3 leaves 2 choices for 1,2 tree, choice of 5 or 6 as root of right subtree, and 2 choices for remaining subtree of 2 nodes.

  2. How many heaps of 5 keys 1,2,3,4,5 are there?

     8. Root is 1, left child is either 2, giving 3!=6 choices, or 3, forcing 2 as right child and leaving 2 choices.

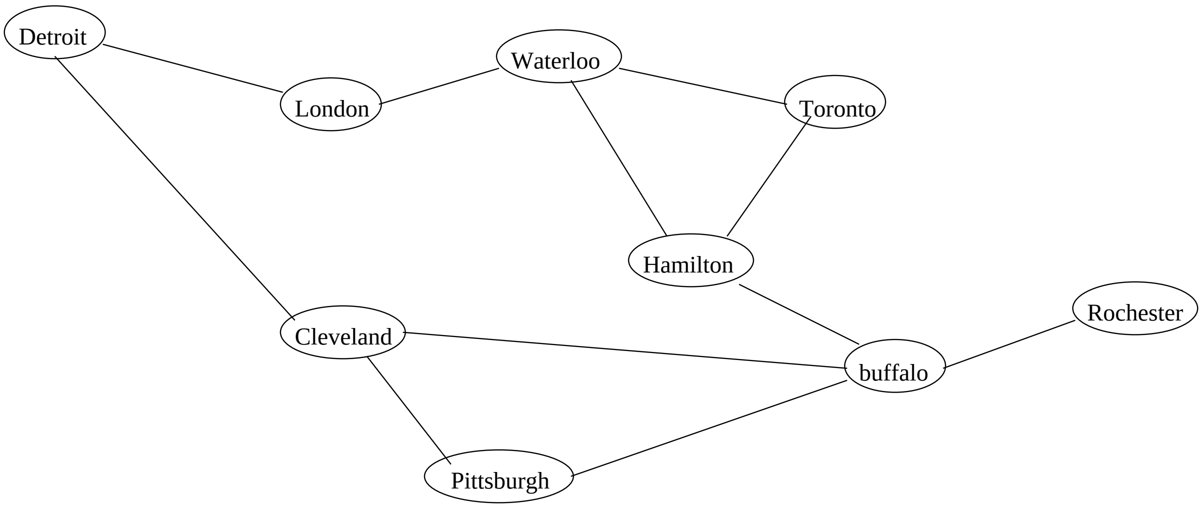

Figure 2: An undirected graph

- (10 marks) Consider the graph in Figure 2. Assume that the edges are ordered alphabetically. That is, when facing with alternatives, choose the edges in alphabetical order.

  1. List the nodes (cities) of the graph in Figure 2 by depth-first search starting from Waterloo.

     W,H,B,C,D,L,P,R,T.

  2. List the nodes (cities) of the graph in Figure 2 by breadth-first search starting from Waterloo.

     W,H,L,T,B,D,C,P,R.

  3. Suppose that in the above graph, all edges have weight 1 except that the edges (Buffalo, Cleveland) and (Pittsburgh, Cleveland) each have weight 2. Draw a minimum spanning tree for the above graph.

     All 1-edges except one of W-T,T-H, or H-W.

- (10 marks) Let T_1 and T_2 be two arbitrary binary trees, each having n (unlabeled) nodes. Show that it is sufficient to apply at most $2(n-1)$ single rotations to T_1 so that it becomes equal to T_2.

  First, in *one sentence*, clearly state the idea of your strategy.

  Either tree can be changed into a degenerate tree by up to $n-1$ rotations each of which makes the rightmost branch one longer.

- (10 marks)

  1. Simulate the Quicksort algorithm on the following input to sort the input into the ascending order. Always use the first element as the pivot

     `5, 100, 50, 1, 4, 3, 20`

     Give the intermediate result for each recursive call (and assume that the algorithm does not use the insertion sort and so on when the list is short).
     Pivot 5: swap 100,3; swap 50,4; swap 5,1.

     $$3, 4, 1, (5), 50, 100, 20$$

     Pivot 3; swap 4,1; swap 3,1. Pivot 50: swap 100,20; swap 50,20:

     $$1, (3), 4, \quad , 20, (50), 100.$$

  2. Suppose you have an array of n elements containing only three distinct values. How fast can you sort the array? How? (Explain in a few sentences.)
     Linear time by counter-sort (variation of bucket sort); initialize 3 counters cntx,cnty,cnt to 0; go through array identifying the 3 distinct values x,y,z, and counting them. Next sort x,y,z, and fill out array with right number of each value in order.

- (10 marks) Let $A[1..n]$ be an array of numbers. Design a data structure and an algorithm to perform any sequence of the following two operations

  - $Add(i, y)$: add the value y to the ith number;
  - $Partial\_sum(i)$: return the sum of the first i numbers, $\sum_{j=1}^{i} A[j]$.

  Notice that the number of elements remains fixed; the only changes are to the values. Each operation should take $O(\log n)$ steps. You can use one more array of size n as a work space.

  Please specify the data structure first. Then specify the algorithm. You should write no more than 10 lines.

  Add a tree on top of the array, with the array elements as leaves, storing in each node the sum of all values in its subtree.

  To add y to the i'th number, add it to all nodes in the path from leaf i up to the root of the tree.

  To obtain the sum of the first i numbers, take the path from leaf $i+1$ up to the root, summing for each right child we visit the value at the left-child.

- (10 marks) String matching.

1. Draw the (KMP) skip table for the pattern $ABABD$.

   The last column is used in part (b) and is not needed for part (a).

	A	B	A	B	D	\$
A	0	1	0	3	2	5
B	1	0	3	0	5	6
D	1	2	3	4	0	6
other	1	2	3	4	5	6

2. Show how to modify the KPM algorithm we have learned in class (which finds an occurrence of the pattern in text in linear time) to a linear time algorithm which, on input of a text string $a_1 \dots a_n$ and a pattern string $b_1 \dots b_m$, finds *all* occurrences of the pattern in the text. Write only a few lines of English explaining how you would do this. No details required.

   Extend pattern with a special \$ character. A mismatch at this position is reported as a real match and matching continues in accordance with the table above.