FACULTY OF SCIENCE

FINAL EXAMINATION

COMPUTER SCIENCE 308-251B

DATA STRUCTURES AND ALGORITHMS

Examiner: Luc Devroye — Friday, May 2, 1997

Associate Examiners: Nick Roy, Hao Biao and M.S. Zhang — 14:00–17:00

Instructions: This is a closed-book examination. However, each student is allowed to bring two colored regular-sized (8 by 11) sheets with any amount of personal handwriting on front and back, as well as a magnifying glass. Answer in the examination booklet.

Identification.

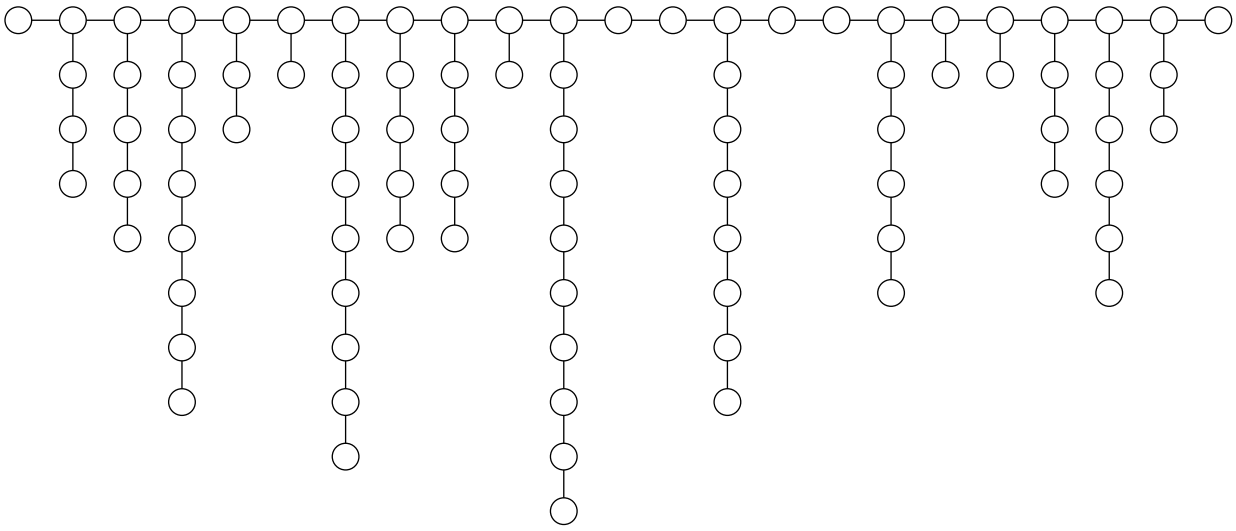
YOUR NAME	STUDENT ID NUMBER

Marks.

QUESTION	WEIGHT	EST. TIME	YOUR SCORE
1	15%	25 min.	
2	20%	20 min.	
3	7%	10 min.	
4	10%	15 min.	
5	7%	10 min.	
6	6%	10 min.	
7	6%	10 min.	
8	6%	10 min.	
9	6%	5 min.	
10	5%	5 min.	
11	12%	15 min.	
TOTAL	100%	175 min.	

Questions about algorithms

Question 1. Creative question on graphs: recognizing caterpillars. A caterpillar is a graph on n nodes that consists of a chain of $k \leq n$ nodes (the caterpillar's body) and k legs with $n - k$ nodes in all, each a chain of zero or nonzero length attached to a node of the body, as in the figure below. The number k is unknown. A graph with n nodes, numbered 1 through n is given in adjacency list format.



A. Give a C or general algorithmic type definition for storing a graph in adjacency list format.
B. Give an algorithm that takes an arbitrary graph and verifies in $O(n)$ worst-case time whether the graph is a caterpillar.

Question 2. Diameter of a binary tree. In a binary tree, the diameter is the maximal distance between any two nodes. The purpose of this question is to write an algorithm that finds the diameter in $O(n)$ worst-case time where n is the number of nodes. We will do this in various steps. Assume that each node `x` of the tree has left and right pointers `l[x], r[x]` and a key `k[x]`. The root is denoted by `root`.

A. Write a recursive linear time algorithm that stores the nodes of a binary tree in postorder in a table of size n. Thus, at the end of this, the tree is stored in a table with generic `i`-th entry `key[i], left[i], right[i]`. (The question asks you to transform the collection `l[.], r[.], k[.]` into the collection `left[i], right[i], key[i]` where `i` ranges from 1 through n.)
B. Given the table of part A, augment each entry by the height of the node (the height of a node is the maximal distance from that node to any leaf that can be reached from the node by moving away from the root) in $O(n)$ time. Thus, at the end of this, the tree is stored in a table with generic i-th entry `key[i], left[i], right[i], height[i]`.
C. Write a simple recursive algorithm for computing the diameter given the table of part B.

Data structure design questions

Question 3. Double hashing. We consider open addressing. Suggest a table size m, a hash function $h(.)$ and a probe sequence for double hashing in the following situation: the data consist of 10,000 colors obtained by a scanner from a scanned picture, where each color is a triple of integers (r,g,b), each integer taking values between 0 and 255. For each color we also have a frequency of occurrence f, the number of pixels of that color, and this is an integer in the range 1 through 1 million. The hash table will be used in a color-reduction algorithm in which `search` and `delete` are the main operations.

Question 4. ADT. The `pool` ADT is used for jobs in a computer system. Assume that the jobs have ID numbers that are integers between 1 and N. The operations are

1. `insert(x)`: insert job `x` (an integer between 1 and N). Duplicates are not allowed, so duplicate checking is necessary.
2. `remove(y)`: remove any job `y` from the `pool`, and output its number `y`. (This is like deleting from a priority queue, in which all items are of equal priority.)

A. Assume that N is sufficiently small so that you can allocate memory of size $O(N)$. Suggest a data structure for the `pool`, and explain how you can insure that both operations may be implemented in $O(1)$ worst-case time.

B. Assume that N is too large and memory of size N cannot be allocated. Let n be the maximal number of jobs in the pool at any given moment. Assume furthermore that n is much smaller than N, but is not known beforehand, so no array of size n can be allocated either. What data structure would you use? What is the worst-case complexity of each operation as a function of n?

The following questions require proofs.

Question 5. Merging sorted lists. Given are d sorted lists. The total number of elements is n. Design an $O(n \log d)$ worst-case time algorithm for merging the sorted lists to obtain one big sorted list. We have seen in class that one should always merge the two smallest sorted lists if one uses ordinary pairwise merging. So you should do one of two things:

(i) Show that the above strategy gives the desired complexity.

(ii) Discard the strategy of pairwise merging, and design a new method that is guaranteed to work in the given time.

Question 6. Lower bounds. Derive a lower bound for finding the median of 5 elements based upon combinatorial considerations. Show your work.

Question 7. Minimal spanning tree. Consider n points in the plane, and form the complete graph in which an edge (u, v) has weight equal to the distance from u to v. Why is it impossible that the minimal spanning tree has two crossing edges?

Question 8. Suffix tries. Show that the space requirement for a suffix trie for a binary file consisting of n bits is $\Omega(n^2)$ in the worst case.

Answer the following questions without explanations.

Question 9. Recursive algorithms. The knapsack problem was formulated as follows: given n items of positive integer sizes $a_1, \ldots, a_n$, and a knapsack of size `V`, decide if there is a subset of items (without repetition) whose sum equals `V`. The recursive boolean function `knapsack(i,v)` checks if there is a subset of $a_i, \ldots, a_n$ whose sum is `v`. It is called initially as `knapsack(1,V)`.

```
boolean function knapsack(i,v)
  if v = 0 then return true
  if i > n or v < 0 then return false
  if knapsack(i+1, v-a_i) then return true
                          else return knapsack (i+1, v)
```

What is the worst-case time for this algorithm as a function of n?

Question 10. DFS and BFS trees. Give exact (no $O(.)$ answers!) answers to the following five questions:

A. Give the height of the DFS tree for the complete graph K_n.
B. Give the height of the BFS tree for the complete graph K_n.
C. Give the height of the DFS tree for the complete bipartite graph $K_{n,n}$.
D. Give the height of the BFS tree for the complete bipartite graph $K_{n,n}$.
E. How many back edges are encountered by a DFS traversal on the complete undirected graph K_n?

Question 11. Huffman code. Construct a Huffman code for the 48-character sentence

`"one two three four five six seven eight nine ten"`

where the apostrophes are not part of the input. Report the number of bits used when the sentence is compressed with this code.