Name: _____ Student Number: _____
Please Print

Signature: _____

Instructor: J. Ellis

UNIVERSITY OF VICTORIA
DECEMBER EXAMINATIONS 1995
Computer Science 225 -- F01

Question	Value	Mark
1	50	
2	50	
3	50	
4	50	
5	50	
6	50	
7	50	
8	50	
9	50	
10	50	
Total	**500**	

Instructions:
ALL QUESTIONS MUST BE ANSWERED ON THE EXAMINATION PAPER IN THE SPACE PROVIDED.
This exam consists of 10 pages + cover.
This is a closed book exam.

Question 1 [50 marks]

Consider the "heap" (partially ordered tree) data structure, and assume the largest item resides at the root.

a) Suppose we build a heap out of n items by starting with an empty data structure and invoking *insert* n times. Argue convincingly that there exist inputs for which the entire building process must take time proportional to $n \log n$.
Hint: Consider a worst case for the insertion of the last $n/2$ items.

b) Why is this inferior to the "standard" *build* process

c) When we delete an item from the top of the heap, the "standard" method moves the rightmost leaf into the vacant slot and lets it "bubble down". Why don't we just move up the largest of the root's children, and so on down the tree?

Question 2 [50 marks]

a) Suppose we represent a list using pointers (i.e., a linked list) and the elements in the list are sorted. Can we apply binary search to this list so as to find things quickly? Why/Why not?

b) Explain why a stack (LIFO) can be represented by one way linked list but a queue (FIFO) needs 2 way links.
Hint: A diagram might help.

c) How can both stacks and queues be represented by arrays? Outline the principles used (a diagram can help).

d) What is the main advantage and the main disadvantage of representing lists using pointers?

e) Mention 2 common list operations that are fast (O(log n)) using an array representation of a sorted list, but linear time using pointers.

Question 3 [50 marks]

Suppose we simulate sets A and B by lists. Consider the problem of computing the *difference* A - B, i.e., the set of elements that are in A but not in B.

a) Outline in (high level) pseudo-code an algorithm to compute this operation, assuming that you are restricted to the list operations *traverse* on one list and *search* the other.

b) Using big-oh notation, discuss how representation (pointers vs. arrays) and sorted vs. unsorted lists affects the run time of your algorithm.

c) Now suppose you are free to traverse both lists simultaneously. Explain why and in what circumstances you can get a much better result than in a).

Question 4 [50 marks]

Remember the Longest Common Subsequence Problem.

a) Fill in the table for the given strings. (Starting at the bottom row, left to right and moving upwards.)

c	0					
c	0					
b	0					
a	0					
b	0					
	0	0	0	0	0	0	0
		a	b	a	c	b	c

b) Do we need row i-1 to compute row i+1 (counting from bottom upwards)?

So what do you deduce about the space requirements of the algorithm, (if only the length of an LCS is to be computed)?

c) We can put the longer or the shorter string along the horizontal. Which possibility minimises space usage?

d) If it is not possible to deduce row i-1 from row i (counting from the bottom upwards), what does this say about the space needed to output an actual LCS? Why?

Question 5 [50 marks]

We wish to represent sets A and B that consist of $n/2$ integers chosen from the integers 1 through n. We will consider three possible data structures:

A: a linked list, unsorted and with no duplicates

B: a hash table with n buckets

C: a Boolean array used as a characteristic vector.

Give big-oh run times for the 2 operations below. Consider average time for the hash table and worst case for the others.

a) Insertion of an element.

A: B: C:

b) Computing the union of A and B.

A: B: C:

(c) Under the special conditions defined above (on the elements), how long does it take to convert from the list representation to the characteristic vector?

Is this true in general? (without the special conditions). Why/why not?

(d) Does this affect your answer under (b) for A?

Question 6 [50 marks]

Consider the representation of a ternary tree (each node had no more than three children) that allocates 3 slots/node for pointers. Prove by structural induction (or by induction on the height of the tree, which is almost the same thing) that a non-empty ternary tree has a number of NIL pointers equal to twice the number of nodes plus one.

a) Formalise the statement to be proved, i.e., invent names of the number of nodes in the tree and the number of nils.

b) What is a suitable basis case?

c) Prove the basis case.

d) Prove the inductive step. Note that there are 3 distinct cases, (the root has 1, 2 or 3 children). Since time is limited, restrict your attention to the case of 2 children.

Case 2: root has two children

Question 7 [50 marks]

Suppose you want to implement the operation *max* on a set of integers, which would return the value of the largest integer in the set, (without removing it). For each of the following data structures, plausible ways of representing the set, give worst- and average-case, big-oh running times for this particular operation.

a) A hash table with a number of buckets about equal to the size of the set.

Worst case: _____ Average case: _____

b) A binary search tree (necessarily using pointer representation).

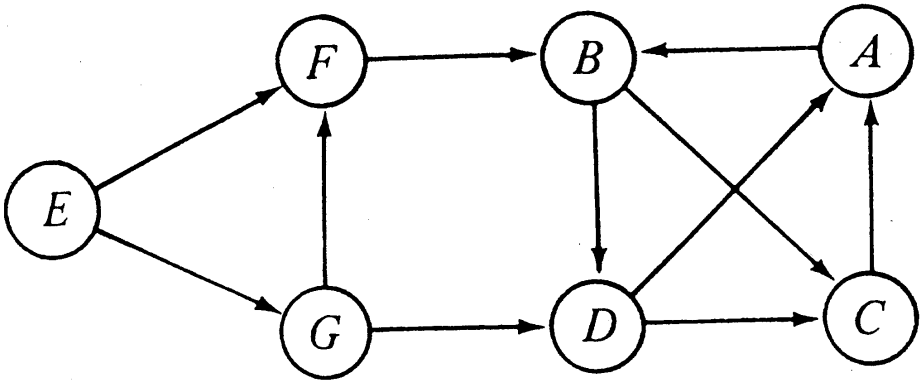
Worst case: _____ Average case: _____

c) A heap represented by an array with $A[i/2] \geq A[i]$ for all i

Worst case: _____ Average case: _____

Question 8 [50 marks]

Consider the following directed graph.



a) Suppose a depth first search is started at node A, and that neighbours are taken in alphabetical order. Mark (clearly, on the diagram) the "tree edges" that would be defined by the dfs algorithm.

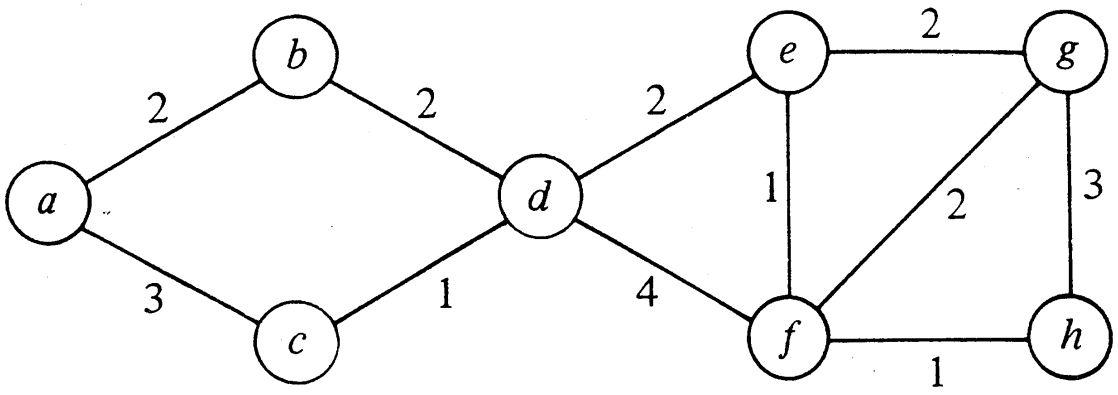
b) Mark each node (on the diagram) with its "post-order" numbering.

c) Which edge(s) are identified as "back edge(s)" by the numbering and why?

d) How many edges minimum would have to be removed from the graph before some topological ordering can exist?

Question 9 [50 marks]

Consider the following weighted graph, and remember Kruskal's MST algorithm (add the next lightest edge, so long as it does not create a cycle). Assume we use time proportional to $|E|\log(|E|)$ to sort the edges.



a) Starting with edge $\{c, d\}$, list the edges in the order that they would be chosen by the algorithm. (arbitrary order for equal weight edges)

b) Suppose we were so naive as to run depth first search (on the partial MST) to check for cycles. What would the overall run time of the algorithm be (asymtotically)? Why?

c) Instead of initially sorting the edges (before anything else is done), we could begin by setting up the edges into a heap, and then proceed by pulling off the top item whenever we wanted a new edge to play with. Would this improve the run time (asymtotically)? Why/why not?

d) Suppose there existed (and there does exist) Union/Find algorithms taking "almost" constant time (amortised). Would their use improve the run time (asymtotically)? Why/why not?

Question 10 [50 marks]

Describe a mathematical model for the following scheduling problem. Suppose there are tasks: $T_1, T_2, \ldots\ldots T_n$ which require times: $t_1, t_2, \ldots\ldots t_n$ respectively to execute and suppose there is a set of constraints of the form: $\{T_i$ must finish before T_j can start, ... $\}$.

a) How would you model the information? (i.e., what would represent the tasks, what would represent the task times and what would represent the scheduling constraints?)
Hint: we are looking for a graph model.

b) In terms of this model which are the tasks that can start immediately and which are those that no other task depends on?

c) Suppose we have the possibility of working on tasks in parallel. In terms of this model, how would you compute (i.e., what problem needs to be solved) a non-trivial lower bound on the time required to complete all tasks.

d) Name the algorithm we have studied which could be used, and say clearly how the output should be interpreted.

END OF THE EXAM!!!