Name: _____ Student Number: _____

*Please Print*

Signature: _____

Instructor: J. Ellis

# UNIVERSITY OF VICTORIA

## APRIL EXAMINATIONS 2001

## Computer Science 225 -- S01

| 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 11 pages + cover.

This is a closed book exam.

Duration: 3 hours

**Question 1 [50 marks]**

Suppose I insist on using a linked list to represent a priority queue (a poor choice, but there it is). Assume that we set up the list links so as not to waste time unnecessarily, and that we make the (obvious!) best choices within the given constraints.
No need to justify your contentions, just state a big oh result in terms of $n$ the number of items in the list.

a) What would be the time complexity of the basic priority queue operations if I choose to keep the list ordered by priority?

- For an insertion?
- For a delete next highest priority?
- For a return next highest priority (without deletion)?

b) What would be the time complexity of the basic priority queue operations if I choose *not* to keep the list ordered by priority?

- For an insertion?
- For a delete next highest priority?
- For a return next highest priority (without deletion)?

c) What was the only purpose of introducing the more elaborate data structure called the "left-ist heap"?

d) What was the particular property of the leftist heap that guarantees fast operations? Be precise.

**Question 2 [50 marks]**

a) You are given a set of sorted lists $L_1, L_2, \ldots\ldots, L_l$.
If the sum of the lengths of the lists is $n$, show that the lists can be merged into one sorted list in time $O(n \log l)$.
Outline your method and justify the time complexity.

b) People have answered a survey about which is their favourite season: winter, spring, summer or fall. How would you sort the responses so that the winter people precede the spring people who precede the summer people who precede the fall people? Of course we want $O(n)$ time.

c) There does exist an algorithm which computes the median of a set of numbers in time $O(n)$.
Argue carefully and clearly how you can use this fact to remove the $O(n^2)$ worst case from Quicksort's performance.

**Question 3 [50 marks]**

a) We are sorting lists that have so many duplicate keys that it is worth considering whether we can take advantage of this fact.
Suppose there are $d$ distinct keys where $d < n$, the number of items in a list.
Show that the lists can be sorted in time $O(n \log d)$ by a comparison based sort (i.e., no buckets).
Hint: maybe a modified partitioning?

b) Suppose you are given two sets $A$ and $B$, represented by lists, not necessarily ordered. Outline a method for checking to see if the sets are identical.

c) What is the time complexity of your method?

**Question 4 [50 marks]**

a) How many possible outcomes are there if we sort a list with 5 elements?

b) Consequently, what does the decision tree model tell us about the minimum number of comparisons necessary for some instances of the problem of sorting lists of 5 items?

c) I have six items to sort, and I proceed as follows:
- o I sort the first 3 items
- o I sort the second 3 items
- o I merge the two sorted lists.

What is the worst case number of comparisons used by this method?

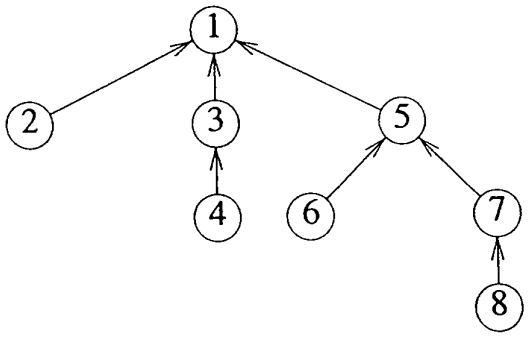
d) Why is it that this method does not use the minimum possible number of comparisons?

**Question 5 [50 marks]**

a) Suppose we represent disjoint sets by rooted trees. What is the time complexity of union-ing two such sets, assuming that we are given pointers to the roots of the trees representing the sets.

b) Why is this method inadequate if the sets are not disjoint?

c) Show the result of doing a find(7) on the following tree-set, when *path compression* is used.




d) Suppose we start out with sets each containing just one element. Show that after $u$ unions, no set can contain more than $u+1$ elements.
Hint: consider the number of unions each particular set has been involved in.

**Question 6 [50 marks]**

a) Suppose we adopt the standard "adjacency list" representation of an undirected graph, in which nodes are named by integers. Recall that the nodes in any list in the structure are not necessarily ordered by node number.
Suppose that for some application you would like them to be ordered.
Outline an algorithm that traverses an adjacency list representation of a graph and creates a new adjacency list in which each list is ordered by node number.

b) If a graph has $n$ nodes and $e$ edges, how many bits *at least* does the adjacency list representation require?
Note: *bits*

c) If a graph has $n$ nodes and $e$ edges, how many bits *at least* does the adjacency matrix representation require?
Hint: *bits*

d) For what kinds of graph is the adjacency list representation inferior to the adjacency matrix representation, in terms of memory usage?

**Question 7 [50 marks]**

a) Outline how you would use a Depth First Search to find the *shortest* cycle containing a particular vertex $u$.
Assume that the length of a cycle is the number of edges in the cycle.

b) Explain clearly why it works.

c) Let $C$ be a cycle in an edge weighted graph $G$. Argue why it is that the heaviest edge in $C$ can not be an edge in any Minimum Weight Spanning Tree for $G$.

Hint: If it were to be in an MST then .....

d) Dijkstra's shortest path algorithm has the property that the nodes moving from "unknown" to "known" status are chosen in order by increasing distance.
This implies that the $O(|E|\log|V|)$ bound we achieved for sparse graphs is optimal, i.e., we could not possibly do better. Why?

**Question 8 [50 marks]**

Consider the following problem. We are given a dictionary containing only 5-letter words. We need an algorithm which, given any pair of words, tells us whether or not there is a way of transforming the first word into the second word by way of a sequence of *single letter replacements*.
For example: *bleed* -- *blend* -- *blond* -- etc.

a) How would you model this problem as a graph problem? In particular:

What do the nodes represent?

What do the edges represent?

What does the algorithm need to compute on the graph to answer the word question?

b) What algorithm would you recommend to answer any such question, given the graph representation?

c) Suppose you expect a long sequence of such questions, what data structure and what preprocessing would you apply so that all questions could be answered in constant time?

**Question 9 [50 marks]**

Consider the following suggestions as a possible algorithm for the single source shortest path problem on directed graphs.

a) Write down an equation that specifies the length of the shortest path from the source $s$ to some node $u$ in terms of the lengths of the shortest path from $s$ to the predecessors $v$ of $u$, and the weight of the edges $(v,u)$.

b) Use this observation to compute the shortest paths from node 1 to every other node in the rather special graph defined on the last page of the exam. Detach that page if it is convenient for you.

Start with the nodes in "stage 1", then "stage 2', etc, etc.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Stage |
|---|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | - | 0 |
| | | | | | | | | | 1 |
| | | | | | | | | | 2 |
| | | | | | | | | | 3 |
| | | | | | | | | | 4 |
| | | | | | | | | | 5 |

Distance Array

**Question 10 [50 marks]**

a) Use the informal, i.e., guess and check, definition of the class **NP** to show that the *Travelling Salesman* problem is in **NP**.
Recall that the TSP asks whether or not a tour of length not greater than $k$ *exists*. This is not the optimisation version of the same problem.

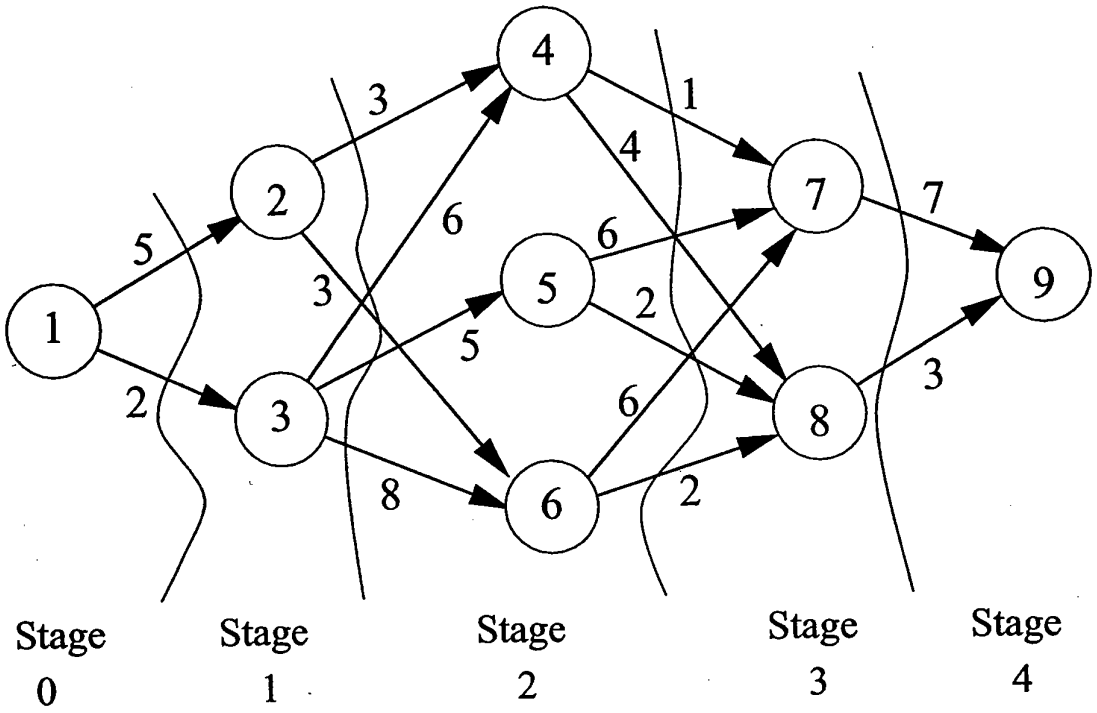
b) The *Longest Path* problem takes a graph with weighted edges and two nodes $s$ and $t$ as input. It asks whether or not there exists a simple path from $s$ to $t$ of weight *at least* some number $k$.

The *Hamiltonian Path between Two Nodes* problem asks whether a simple path exists between the given two nodes that passes through *all* other nodes.

A *simple* path is one that does not pass through any node more than once.

Show that if we had a fast solution to the *Longest Path* problem, we would also have a fast solution to the *Hamiltonian Path between Two Nodes* problem.

c) If we know that the *Hamiltonian Path between Two Nodes* problem is **NP**-complete, what do we now know about the *Longest Path* problem?




The Graph for Question 9

END OF THE EXAM