THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2007
Campus: Tamaki

COMPUTER SCIENCE

Algorithms and Data Structures

(Time allowed: TWO hours)

NOTE: Attempt *all* questions. Write answers in the boxes below the questions. You may continue your answers onto the "overflow" page provided at the back if necessary.
Marks for each question are shown below the answer box. The use of calculators is NOT permitted.

Section:	A	B	C	Total
Possible marks:	30	30	40	100
Awarded marks:				

SURNAME:

FORENAME(S):

STUDENT ID:

Student ID: ______________________________

Section A: Algorithm Analysis

1. A modified `mergesort` separates an array `a` of size $n = 3^m$ (with integer $m = \log_3 n$; $m > 0$) into three successive parts of equal size 3^{m-1} each, recursively sorts each part separately, and merges the three sorted parts into a desired sorted array:

```
public static void modifiedMergesort( int a[] ) {
     modifiedMergesort( a, 0, a.length - 1 );
}
private static void modifiedMergesort( int a[], int frst, int last ) {
  if ( frst < last ) {
      int size = (last - frst) / 3;
      int lst1 = frst + size;
      int lst2 = lst1 + size;
      modifiedMergesort( a, frst, lst1);     // Sorting the first part
      modifiedMergesort( a, lst1 + 1, lst2 );// Sorting the second part
      modifiedMegresort( a, lst2 + 1, last );// Sorting the third part
      merge( a, frst, lst1, lst2, last );    // Merging the sorted parts
   }
}
```

Assuming the merging step involves $2cn$ elementary operations where $c > 0$ is a constant scale factor, write down the basic recurrence for the processing time $T(n)$ of this algorithm and derive a closed-form formula for $T(n)$ by "telescoping". [10 marks]

The basic recurrence for the processing time $T(n)$ of this algorithm is $T(n) = 3T(n/3) + 2cn$, i.e. $\frac{T(n)}{n} = \frac{T(n/3)}{n/3} + 2c$, or $\frac{T(3^m)}{3^m} = \frac{3^{m-1}}{3^{m-1}} + 2c$. "Telescoping" of the recurrence is as follows:

$$
\begin{array}{rcrcl}
\frac{T(3^m)}{3^m} & = & \frac{T(3^{m-1})}{3^{m-1}} & + & 2c \\
\frac{T(3^{m-1})}{3^{m-1}} & = & \frac{T(3^{m-2})}{3^{m-2}} & + & 2c \\
\frac{T(3^{m-2})}{3^{m-2}} & = & \frac{T(3^{m-3})}{3^{m-3}} & + & 2c \\
\ldots & & \ldots & & \ldots \\
\frac{T(3^2)}{3^2} & = & \frac{T(3)}{3} & + & 2c \\
\frac{T(3)}{3} & = & \frac{T(1)}{1} & + & 2c
\end{array}
$$

After summing the left-hand sides and right-hand sides of these equalities and reducing the same terms in the sums, the close-form formula for $T(n)$ is $\frac{T(3^m)}{3^m} = \frac{T(1)}{1} + 2cm$, or $T(3^m) = 3^m T(1) + 3^m 2cm$, or $T(n) = nT(1) + 2cn \log_3 n$.

The equivalent solution telescopes the recurrence $T(3^m) = 3T(3^{m-1}) + 2c3^m$:

$$
\begin{array}{rcrcl}
T(3^m) & = & 3T(3^{m-1}) & + & 2c3^m \\
3T(3^{m-1}) & = & 3^2T(3^{m-2}) & + & 2c3^m \\
3^2T(3^{m-2}) & = & 3^3T(3^{m-3}) & + & 2c3^m \\
\ldots & & \ldots & & \ldots \\
3^{m-2}T(3^2) & = & 3^{m-1}T(3) & + & 2c3^m \\
3^{m-1}T(3) & = & 3^mT(1) & + & 2c3^m
\end{array}
$$

After summing the left-hand sides and right-hand sides of these equalities and reducing the same terms in the sums, the close-form formula for $T(n)$ is just the same: $T(3^m) = 3^m T(1) + 2cm3^m$, or $T(n) = nT(1) + 2cn \log_3 n$. Both the solutions are admissible.

Student ID: ____________________

2. Prove that the height h_n of a complete binary tree with n nodes is at most $\lfloor \log_2 n \rfloor$ where $\lfloor z \rfloor$ denotes the closest integer smaller than or equal to a real number z. Use this fact to prove that insertion of a new node into a heap of n elements takes logarithmic, $O(\log n)$, time. [10 marks]

A complete binary tree is a binary tree which is completely filled at all levels except, possibly, the bottom level, which is filled from left to right with no missing nodes. Depending on the number of nodes at the bottom level, a complete tree of height h_n contains between 2^h and $2^{h+1} - 1$ nodes, so that $2^{h_n} \leq n < 2^{h_n+1}$, or $h_n \leq \log_2 n < h_n + 1$, that is, $h_n \leq \lfloor \log_2 n \rfloor$.

Heaps are complete binary trees. Insertion of a new node into a heap adds one more node to the heap of n elements. First, a new, $(n+1)$-st leaf position is created and the new node with its associated key is placed in this leaf. If the inserted key preserves the heap order, the insertion is completed. Othewise, the new key has to swap with its parent. This process of bubbling, or percolating up the key is repeated toward the root until the heap order is restored. Therefore, there are at most h_n swaps, so that the running time is at most $O(\log n)$.

3. Mark each statement in the following table true (T) or false (F) in the third column. For each correct answer you score $+2$, for each incorrect one -1, and for each question not answered, 0. [10 marks]

	Statement	T or F?
1	The worst-case time complexity of static binary search is $O(n \log n)$	F
2	Insertion sort has quadratic average-case time complexity	T
3	Quickselect has logarithmic worst-case time complexity	F
4	The time for removing a node from a binary search tree of height h is $O(h)$	T
5	Search time in a hash table is completely independent of its load factor	F

Student ID: ______________________________

Section B: Graph algorithms

4. Mark each question true (T) or false (F). For each correct answer you score $+1$, for each incorrect one -1, and for each question not answered, 0. [20 marks]

   (a) If Kruskal's algorithm is run on a graph that is not connected, it terminates after finding a minimum weight spanning forest.

   TRUE

   (b) We can always find a topological order for a DAG by outputting the nodes in order of increasing "seen" time.

   FALSE

   (c) There is a linear-time algorithm for determining whether a graph is 2-colourable.

   TRUE

   (d) If DFS is run on a digraph and no tree arcs are created, then the graph has no arcs at all.

   FALSE

   (e) Let G be a digraph and $(v, w) \in E(G)$. If DFS is run on G and v is seen before w, then w is a descendant of v in the DFS forest.

   TRUE

   (f) If DFS is run on a graph, there can be no cross edges.

   TRUE

   (g) Floyd's algorithm is preferable to running Dijkstra's algorithm n times, if we are solving the all-pairs shortest path problem on a class of large dense digraphs.

   TRUE

   (h) There can be no arcs between different strongly connected components of a digraph.

Student ID: ______________________________

FALSE

(i) Consider the DAG G on nodes $0, 1, \ldots, n$, whose arcs are precisely those of the form (i, j) with $i < j$. Then $0, 1, \ldots, n$ is a topological order for G.

TRUE

(j) The graph with nodes $0, \ldots, 4$ and edges $\{i, j\}$ whenever $i \neq j$ has girth 4.

FALSE

(k) The graph with nodes $0, \ldots, 5$ and edges $\{i, j\}$ whenever $i + j = 5$ or $i = 0, j = 2$ will have precisely two trees in its BFS forest for any choice of roots.

TRUE

(l) To solve the all-pairs shortest path problem in an unweighted digraph, Floyd's algorithm is always preferable to using BFS from each node.

FALSE

(m) Breadth-first search creates only tree or cross edges when run on a graph.

TRUE

(n) Kruskal's algorithm solves the MST problem in linear time.

FALSE

(o) If DFS is run on a digraph and v is visited before w, then w finishes processing before v.

FALSE

(p) Priority-first search using a priority queue is an efficient way to simulate BFS and DFS, as well as being the basis for Dijkstra's algorithm and Prim's algorithm.

FALSE

Student ID: ______________________________

(q) Suppose that G is a connected graph on which we have run BFS, and $\{v, w\} \in E(G)$ is a cross edge. Then v and w are at the same distance from the root of the BFS tree.

FALSE

(r) Suppose that we run DFS on a digraph G and keep the $seen, done$ timestamps but no other information about the search forest created. Then we can still always distinguish cross arcs from forward arcs.

TRUE

(s) A good way to compute the girth of a graph is to run DFS from each node in turn and return the length of the smallest cycle found (a cycle is found when we see a back arc).

FALSE

(t) The Bellman-Ford algorithm solves the SSSP in time $O(ne)$ for any weighted digraph.

TRUE

5. Consider the weighted graph G whose weighted adjacency matrix is shown below and answer the following questions. No working is required to be shown. Be careful — no partial credit will be given for wrong answers.

$$\begin{bmatrix} 0 & 5 & 9 & 3 & \infty \\ 5 & 0 & 4 & \infty & 3.5 \\ 9 & 4 & 0 & 5 & 1 \\ 3 & \infty & 5 & 0 & 3 \\ \infty & 3.5 & 1 & 3 & 0 \end{bmatrix}$$

(a) First consider the SSSP. Fill in the entries of the distance vector computed by each iteration of Dijkstra's algorithm when run on G with source node 0. The initial values have been filled in. [5 marks]

Iteration	Node 0	Node 1	Node 2	Node 3	Node 4
0	0	∞	∞	∞	∞
1	0	5	9	3	∞
2	0	5	8	3	6
3	0	5	8	3	6
4	0	5	7	3	6

Student ID: ______________________________

(b) Now consider the MST problem. List, in the order that they are added to the tree, the edges used in the minimum spanning tree of G found by Prim's algorithm. Write each edge in the form $\{a, b\}$ where a, b are the vertices at the endpoints and $a < b$. [5 marks]

The solution starting from the root 2: $\{2, 4\}, \{3, 4\}, \{0, 3\}, \{1, 4\}$.

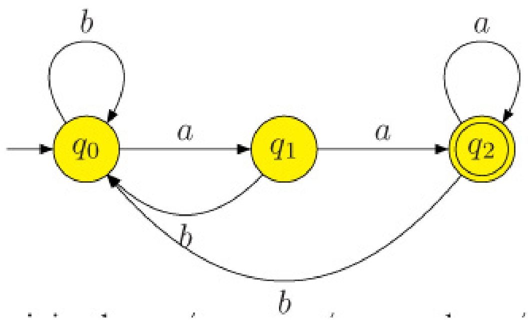
The solutiuon starting from the root 0: $\{0, 3\}, \{3, 4\}, \{2, 4\}, \{1, 4\}$.

Both solutions as well as those started from the other nodes are valid.

Section C: Automata Theory and Grammars

6. Consider the language L consisting of all strings over the alphabet $\{a, b\}$ whose last two letters are both a. a) Write explicitly the language L defined above. b) Construct (by drawing the diagram) a three state DFA M that recognises L. c) Determine whether M is minimal. [10 marks]

a) $L = \{xaa \mid x \in \{a, b\}^*\}$.
b) The DFA that recognises L is



c) M is minimal: $q_0 \not\equiv_0 q_2, q_1 \not\equiv_0 q_2$ and $q_0 \not\equiv_1 q_1$ because $\delta(q_0, a) = q_1 \neq q_2 = \delta(q_1, a)$.

Student ID: ______________________________

7. Mark each statement in the following table true (T) or false (F) in the third column. The first entry (0) is an example. For each correct answer you score 2 marks. [10 marks]

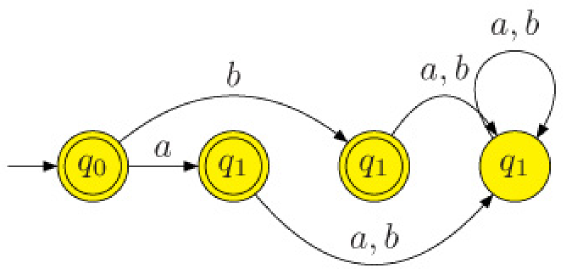
0	every language accepted by an NFA is infinite	false
1	every finite language is accepted by some DFA	true
2	every NFA is a DFA	false
3	the language $\{a^n b^m \mid n, m > 0\}$ is not accepted by any NFA	false
4	it is algorithmically decidable whether a DFA M accepts finitely many strings	true
5	it is not algorithmically decidable whether an NFA N accepts finitely many strings	false

8. Enumerate all steps for constructing a DFA accepting exactly the language denoted by the regular expression: $(ab)^* + a$. [10 marks]

Construct NFAs N_1 and N_2 accepting the languages $\{a\}$ and $\{b\}$, respectively.
Construct an NFA N_3 for the concatenation of $L(N_1)$ and $L(N_2)$ obtaining the language $\{ab\}$.
Construct an NFA N_4 for the Kleene closure of $L(N_3)$ so obtaining $\{ab\}^*$.
Construct an NFA N_5 for the union of $L(N_4)$ and $L(N_2)$ obtaining the language $\{ab\}^* \cup \{a\}$.
Transform N_5 into an equivalent DFA M.

9. Show that there is an algorithm which receives as input a DFA M over the alphabet $\{a, b\}$ and decides whether $L(M) = \{\varepsilon, a, b\}$ or $L(M) \neq \{\varepsilon, a, b\}$. Clearly state all results you use. [10 marks]

It is known that there is an algorithm deciding whether two DFAs accept the same language. The language $L = \{\varepsilon, a, b\}$ is accepted by the DFA M':



So, we can apply the above algorithm to the DFAs M and M' to decide whether $L(M) = L(M')$, that is, $L(M) = \{\varepsilon, a, b\}$.

Student ID: ______________________________

Additional work page

Student ID: __________

Additional work page