Department of Computing and Information Systems
COMP20007 Design of Algorithms
Semester 1, 2013
Mid Semester Test

Instructions

- **Do not open this paper until instructed to do so.** You may read this page now.
- You may fill out your name and student number now.
- You must have your student card on display during this test.
- The test will start at 5:30pm and finish at 6:00pm.
- The total time allowed for this test is 30 minutes.
- This is a closed book exam. You should **not** have any study notes of any kind, including electronic (no calculators, phones, etc).
- Any student seen looking at their phone (or similar) during the test will have their paper removed immediately, and will be referred to the Engineering School for a breach of academic honesty.
- Throughout you should assume a RAM model of computation where input items fit in a word of memory, and basic operations such as $+ - \times /$ and memory access are all constant time.
- Answer all questions on this paper.
- Remember, -1 mark for an incorrect answer in Question 1 true/false (advised not to guess).

Name: ______________________

Student Number: ______________________

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 1 [8 marks, minimum 0 marks]

Answer True or False for each of these statements. You will gain one mark for a correct answer, and **lose** one mark for an incorrect answer.

1.	$f(n) = 10n \log(10n)$ is in $\Theta(n \log n)$	
2.	$f(n) = n^2$ is in $\Omega(\log n)$	
3.	$f(n) = \sqrt{n}$ is in $\Theta(n)$	
4.	If the running time of an algorithm is described by a recurrence relation $T(n) = aT(n/b) + O(n^d)$, and it is known that $d = \log_b a$, then $T(n)$ is in $O(n^d \log n)$.	
5.	Any algorithm that runs in $\Omega(\log^* n)$ time can be described as efficient.	
6.	An adjacency matrix representation of a graph with n vertices and m edges requires $\Theta(V^2)$ space.	
7.	Counting Sort on an array of n integers where the maximum element number K requires $\Theta(n + K)$ time.	
8.	MSB Radix sort on an array of n integers requires $O(n \log n)$ time.	

Question 2 [5 marks]

Complete the following table with worst case big-Oh running times for the operations on the named data structures. You should assume that there are n elements in the data structure at the time of the call to the operation. Be as precise as possible: that is, use Θ if possible and include all lower order terms involving n if needed.

1	Insert an element into a heap assuming $O(1)$ time for key comparisons.	
2	Remove an element from a sorted array. You may assume the index of the element is known.	
3	Find an item in a union-find-by-rank tree that uses path compression. You may assume an $\Theta(1)$ time map from the element id to the node in the tree exists.	
4	Find an item in a union-find data structure based on linked lists where each node includes a pointer to the head of the list. You may assume an $\Theta(1)$ time map from the element id to the node in list exists.	
5	Decrease the key of an element in a heap. You may **not** assume that a mapping exists from the element id into the heap structure.	

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 3.1 [3 marks]

Write a description of an efficient function that performs a Topological Sort on a directed graph $G(V, E)$; the function should return the vertices in order so that G is linearised. Your function should first check that G is a DAG, and if it is not simply exit reporting failure to sort. You may assume that you have access to a library function that performs DFS as outlined in the book and lectures. I am expecting about 3 or 4 lines of description: there is no need to go into low level details involving how G is stored and no need for C code.

Question 3.2 [3 marks]

Analyse your algorithm's worst case running time from Question 3.1. You can assume that $|V| = n$ and $|E| = m$.

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 4 [6 marks]

The following is Dijkstra's algorithm for solving the Single Source All Shortest Paths problem on a graph $G(V, E)$ [reproduced from the online draft version of Dasgupta et al. without permission.]

(a) (3.5 marks) What is the running time of the algorithm if G is stored as an adjacency matrix and the priority queue stored as a heap? Justify your answer by writing the cost of each line in the box to the right of the pseudo-code.

```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0

H = makequeue(V)   (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

Total	

(b) (2.5 marks) Without altering the format of G, can this running time be reduced? How and to what time complexity?

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Department of Computing and Information Systems
COMP20007 Design of Algorithms
Semester 1, 2013
Mid Semester Test
MARKING GUIDE

Instructions

- **Do not open this paper until instructed to do so.** You may read this page now.
- You may fill out your name and student number now.
- You must have your student card on display during this test.
- The test will start at 5:30pm and finish at 6:00pm.
- The total time allowed for this test is 30 minutes.
- This is a closed book exam. You should **not** have any study notes of any kind, including electronic (no calculators, phones, etc).
- Any student seen looking at their phone (or similar) during the test will have their paper removed immediately, and will be referred to the Engineering School for a breach of academic honesty.
- Throughout you should assume a RAM model of computation where input items fit in a word of memory, and basic operations such as $+ - \times /$ and memory access are all constant time.
- Answer all questions on this paper.
- Remember, -1 mark for an incorrect answer in Question 1 true/false (advised not to guess).

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 1 [8 marks, minimum 0 marks]

Answer True or False for each of these statements. You will gain one mark for a correct answer, and **lose** one mark for an incorrect answer.

1.	$f(n) = 10n\log(10n)$ is in $\Theta(n\log n)$	T
2.	$f(n) = n^2$ is in $\Omega(\log n)$	T - polynomial is bigger than log
3.	$f(n) = \sqrt{n}$ is in $\Theta(n)$	F - $n^{1/2} \not> c.n$
4.	If the running time of an algorithm is described by a recurrence relation $T(n) = aT(n/b) + O(n^d)$, and it is known that $d = \log_b a$, then $T(n)$ is in $O(n^d \log n)$.	T
5.	Any algorithm that runs in $\Omega(\log^* n)$ time can be described as efficient.	F - Some algs are $O(\log n)$, eg sift up.
6.	An adjacency matrix representation of a graph with n vertices and m edges requires $\Theta(V^2)$ space.	T
7.	Counting Sort on an array of n integers where the maximum element number K requires $\Theta(n+K)$ time.	T
8.	MSB Radix sort on an array of n integers requires $O(n\log n)$ time.	T - as it says on the front of the test, all inputs fit in $\log n$ bits.

Question 2 [5 marks]

Complete the following table with worst case big-Oh running times for the operations on the named data structures. You should assume that there are n elements in the data structure at the time of the call to the operation. Be as precise as possible: that is, use Θ if possible and include all lower order terms involving n if needed.

1	Insert an element into a heap assuming $O(1)$ time for key comparisons.	$O(\log n)$ sift up
2	Remove an element from a sorted array. You may assume the index of the element is known.	$O(n)$ to fill the hole
3	Find an item in a union-find-by-rank tree that uses path compression. You may assume an $\Theta(1)$ time map from the element id to the node in the tree exists.	$O(\log^* n$ OR $O(\log n)$
4	Find an item in a union-find data structure based on linked lists where each node includes a pointer to the head of the list. You may assume an $\Theta(1)$ time map from the element id to the node in list exists.	$\Theta(1)$. $O(1)$ no marks.
5	Decrease the key of an element in a heap. You may **not** assume that a mapping exists from the element id into the heap structure.	$O(n + \log n)$. n to find the element in the head, and $\log n$ to decrease its key. No marks for $O(\log n)$ or $O(n)$ alone.

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 3.1 [3 marks]

Write a description of an efficient function that performs a Topological Sort on a directed graph $G(V, E)$; the function should return the vertices in order so that G is linearised. Your function should first check that G is a DAG, and if it is not simply exit reporting failure to sort. You may assume that you have access to a library function that performs DFS as outlined in the book and lectures. I am expecting about 3 or 4 lines of description: there is no need to go into low level details involving how G is stored and no need for C code.

function TSort(G)
 Call DFS on G, recording the post-number of vertices.
 If a back-edge is discovered in the DFS, report failure and finish.
 Sort the vertices by increasing post-number.

1 Mark for calling DFS to get post-numbers

1 Mark for checking back-edges in DFS to detect cycles (or some other efficient mechanism)

1 Mark for sorting vertices by post number, or storing them in sorted order as they come out of the DFS

Question 3.2 [3 marks]

Analyse your algorithm's worst case running time from Question 3.1. You can assume that $|V| = n$ and $|E| = m$.

1 Mark for $O(n + m)$ or $O(V + E)$ for DFS. Θ also correct.

1 Mark for accounting for the time to check back edges: $O(V + E)$ time if done separately, or "no extra time" or $O(1)$ if folded into previous step.

1 Mark for sorting cost $O(n \log n)$, or $O(n + \max(V))$ if Counting Sort, or $O(n \log(\max(V)))$ if Radix sort.

Note: if the algorithm is wrong, but analysis is correct, can still get full marks here.

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)

Question 4 [6 marks]

The following is Dijkstra's algorithm for solving the Single Source All Shortest Paths problem on a graph $G(V, E)$ [reproduced from the online draft version of Dasgupta et al. without permission.]

(a) (3.5 marks) What is the running time of the algorithm if G is stored as an adjacency matrix and the priority queue stored as a heap? Justify your answer by writing the cost of each line in the box to the right of the pseudo-code.

```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0

H = makequeue(V)   (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

Total	

0.5 Mark for mentioning initialisations are $\Theta(n)$ or $O(n)$ in total.
0.5 Mark for cost of makequeue. Either $O(n)$ or $\Theta(n)$ or $O(n \log n)$.
0.5 Mark for "while" loops $\Theta(n)$ times or $O(n)$ or just n times
0.5 Mark for cost of deleteMin. $O(\log n)$.
0.5 Mark for "for all edges" being $O(n)$ or $\Theta(n)$ or n times.
0.5 Mark for decreasekey being $O(\log n)$.
0.5 Mark for totalling it all up to $O(n^2 \log n) = O(n + n \log n + n(\log n + n(\log n)))$. (not Θ.). Note that smarter totalling is possible. eg decreasekey is only called once per edge, so time could be $O(n^2 + m \log n)$.

Throughout using V rather than n is ok.

(b) (2.5 marks) Without altering the format of G, can this running time be reduced? How and to what time complexity?

0.5 mark Use an unsorted array for the priority queue
0.5 mark makequeue becomes/remains $O(n)$
0.5 mark deltemin becomes $O(n)$ as have to search for min
0.5 mark decreasekey becomes $O(1)$
0.5 mark making a total of $O(n^2)$

Downloaded by Lu Liu (liull3@student.unimelb.edu.au)