Department of Computing and Information Systems

COMP20007 Design of Algorithms

Semester 1, 2016 Mid Semester Test

Name: STEVEN BIRD

Student Number: SAMPLE SOLUTIONS

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 11:10am and finish at 11:50am (40 minutes)*
- *The test is marked out of 40, 1 minute per mark.*
- *The test is worth 10% of your final mark for this subject (but your exam mark will be scaled up instead, if that results in a better final mark)*

- *This is a closed book test. 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) or another student's paper during the test will have their paper removed immediately, and will be referred to the School of Engineering 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 + − × / and memory access are all constant time.*

- *You may use the back of each page for any rough notes; markers will only look at the front side of any page.*
- *Please write legibly. Answer all questions.*
- *NB, −1 mark for an incorrect answer in Question 1 True/False (advised not to guess).*

Question 1. Circle T (True) or F (False) for each of these statements. You will gain one mark for a correct answer, and lose one mark for an incorrect answer. [7 marks, min 0 marks]

1. (T) / F $f(n) = 10n \log(10n)$ is in $\Theta(n \log n)$
2. (T) / F $f(n) = n^2$ is in $\Omega(\log n)$
3. T / (F) $f(n) = \sqrt{n}$ is in $\Theta(n)$
4. (T) / F 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. (T) / F An adjacency matrix representation of a graph with n vertices and m edges requires $\Theta(V^2)$ space
6. (T) / F Inserting a new item into a heap of n items requires $\Theta(\log n)$ time
7. T / (F) Consider an edge from u to v in a directed graph such that u is in strongly connected component X and v is in strongly connected component and Y != X. For any DFS on the graph, the pre number of u will be higher than the pre number of v

Question 2. A priority queue can be implemented in a variety of ways. State the efficiency of the given operations for each of the implementation methods. [10 marks]

	insert(x, pri)	deletemax()
array sorted by item_id	O(n)	O(n)
Array sorted by priority	O(n)	O(1)
linked list sorted by item_id	O(n)	O(n)
linked list sorted by priority	O(n)	O(1)
binary heap	O(log n)	O(log n)

Question 3. Consider the following (not very smart) algorithm. The input is an array, A, and some constant integer k. Assume a comparison based sort in Step 2. [6 marks]

```
my_divide_and_conquer(A, k)
    Let n be the number of elements in A
    If n <= k, return A
    Sort A                                   — O(n log n)
    Copy A[k..n-1] into a new array C        — O(n)
    Call my_divide_and_conquer(C, k)
```

a. Write down a recurrence relation that describes the running time of the algorithm [3]

$$T(n) = \begin{cases} O(1) & n \leq k \\ T(n-k) + O(n \log n) & \text{otherwise} \end{cases}$$

b. Establish a tight upper bound on the worst case running time of the algorithm, showing your working [3]

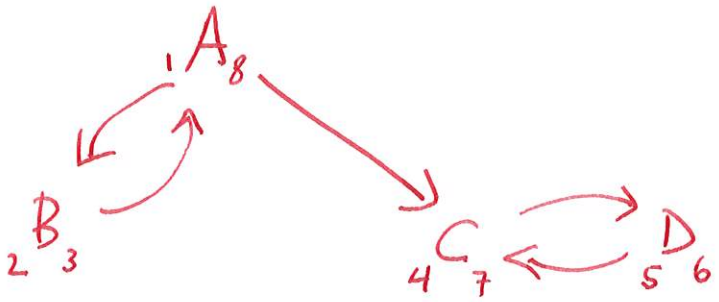
Worst case: $k=1$

$$T(n) = T(n-1) + O(n \log n)$$

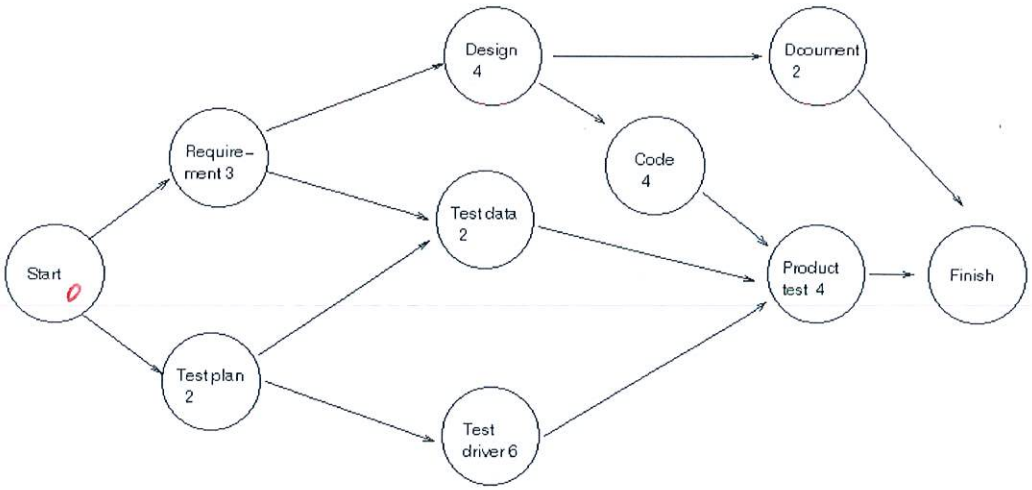
$$\in O(n^2 \log n)$$

Question 4. Kosaraju's algorithm for finding Strongly Connected Components (SCCs) uses the highest post numbered vertex from a DFS of the reverse of the input graph to find a sink SCC in the input graph. Why is it necessary to reverse the graph: why not just take the lowest post number in the original graph? (Hint: show a graph where this does not work; include pre and post numbers of DFS.) [4 marks]

The lowest post number may not be in a sink SCC, e.g.



Question 5. A software development company needs to determine the amount of time to complete a multi-component project. Some tasks can be undertaken in parallel. The tasks and dependencies can be represented using a graph (known as a PERT Chart), as illustrated below. [13 marks]

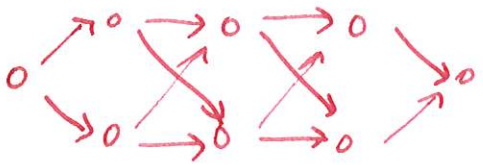


a. Write down a brute force algorithm which takes such a graph, with labelled, weighted nodes, and outputs the shortest amount of time required to complete the project. [2]
b. Analyse the efficiency of this algorithm. [3]

a) time(node)

```
max_incoming = 0
for each incoming node n:
    if max < time(n):
        max = time(n)
return max_incoming + time for this node
time("finish")
```

b.)



- adding 2 nodes doubles the number of paths

$2^{|V|/2}$ paths $\therefore O(2^{|V|})$

c. Devise a more efficient algorithm for doing the same task. [3]
d. Analyse the efficiency of this new algorithm. [3]

c) for each node n in topological_sort(nodes):

time[n] = max(incoming node times) + time for this node

time["finish"]

d) topological sort – $O(|V| + |E|)$

then for each node consider all incoming edges – $O(|V| + |E|)$ also

combined: $O(|V| + |E|)$

e. What property or properties of the graph are assumed by your algorithms? [2]

acyclic, unique finish node