THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

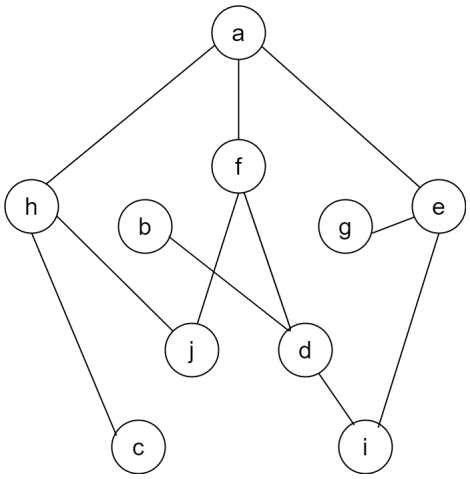
School of Information Technology and Electrical Engineering

Semester Two Final Examinations, 2020

COMP4500/7500 Advanced Algorithms & Data Structures

Question 1 [20 marks total]

Consider the following undirected graph, in which each node is labeled using one letter.



a. [10 marks] Draw a depth-first search tree for the above graph, starting from node a, as generated by the DFS algorithm given below.

DFS(G)

```
1  for each u ∈ G. V
2        u. colour = WHITE
3        u.parent = NULL
4  for each u ∈ G. V in alphabetical order
5        if u. colour == WHITE
6              DFS-VISIT(G, u)
```

DFS-VISIT(G, v)

```
1  v. colour = GREY
2  for each u ∈ G. Adj[v] in alphabetical order
3        if u. colour == WHITE
4              u.parent = v
5              DFS-VISIT(G, u)
6        v. colour = BLACK
```

b. [10 marks] Draw a breadth-first search tree for the above graph, starting from node $s = \texttt{a}$, as generated by the BFS algorithm given below.

BFS(G, s)

```
 1  for each u ∈ G.V
 2        u.colour = WHITE
 3        u.parent = NULL
 4  s.colour = GREY
 5  Q.initialise()
 6  Q.enqueue(s)
 7  while not Q.isEmpty()
 8        v = Q.dequeue()
 9        for each u ∈ G.Adj[v]
10              if u.colour == WHITE
11                    u.colour = GREY
12                    u.parent = v
13                    Q.enqueue(u)
14        v.colour = BLACK
```

Question 2 [15 marks total]

Given the following divide and conquer recursive algorithm, find the tight asymptotic bound on the complexity function $T(n)$, knowing that the function MERGE takes 3 parameters, each of them is an array of size $n/3$, and has an asymptotic tight bound of $\Theta(n)$. Justify your answers by using the master theorem. Show your working.

MY-DIVIDE-AND-CONQUER(W)

```
1  // W is an array of size n, and W^(0), W^(1),and W^(2) are arrays of size n/3
2  // X is an array of size n, and X^(0), X^(1),and X^(2) are arrays of size n/3
3  W^(0).initialise()
4  W^(1).initialise()
5  W^(2).initialise()
6  n = W.length
7  if n ≤ 1
8        return W
9  for i = 0 to n − 1
10       if i mod 3 == 0
11             W^(0).append(w_i)
12       else
13             if i mod 3 == 1
14                   W^(1).append(w_i)
15             else
16                   W^(2).append(w_i)
17 X^(0) = MY-DIVIDE-AND-CONQUER(W^(0))
18 X^(1) = MY-DIVIDE-AND-CONQUER(W^(1))
19 X^(2) = MY-DIVIDE-AND-CONQUER(W^(2))
20 X = MERGE(X^(0), X^(1), X^(2))
21 return X
```

Question 3 [15 marks] Given the recurrence

$$T(n) \quad = \quad T(\lfloor n/3 \rfloor) + T(\lfloor 2n/3 \rfloor) + cn$$

a. [5 marks] Make an informed guess of the asymptotic upper bound using any method you prefer. Provide justification for your guess and make sure it is as tight as possible.

b. [10 marks] Solve the provided recurrence using the substitution method to prove the suggested asymptotic upper bound you provided in part a of this question. Clearly describe your inductive assumptions and boundary conditions, and show your working.

Question 4 [25 marks]

This question involves performing an amortised analysis of a data structure.

A Robot is designed to complete optional tasks stored in its memory. New tasks are found regularly and added to the Robot tasks list T. The Robot memory, list T, has a maximum capacity of n tasks. Once a task is completed it is removed from the list T. However, if a new task is found while the Robot memory is full, then this Robot is programmed to remove old tasks to make space for storing newly discovered tasks. The operation NEW-TASK(x) inserts a new task x into list T.

The implementation uses an array T of size n, and an integer $task$ where $0 \leq task \leq n$. The array T is indexed from zero and the next free slot (if any) is at position $task$ within the array T.

NEW-TASK(x)

1 **if** $task == n$
2 REMOVE-OLD(T) // Remove $\lceil n/3 \rceil$ oldest tasks from T and make $task = n - \lceil n/3 \rceil$
3 $T[task] = x$
4 $task = task + 1$

If NEW-TASK(x) is called when the list T is full, (i.e. $task == n$), then $\lceil n/3 \rceil$ oldest tasks will be removed from T, then task x will be inserted into T. The removal is performed by operation REMOVE-OLD which will also update the value of $task$ to be $n - \lceil n/3 \rceil$.

a. [10 marks] Starting from an empty list T (i.e. $task == 0$), assume that the Robot has found m new tasks. How many times will the operation REMOVE-OLD be called? Provide an exact answer (not order of magnitude) in terms of m (the number of operations) and n (the size of T). Assume that the Robot didn't complete any operation during the time of founding the m new tasks.

b. [10 marks] Assuming that the procedure REMOVE-OLD(T) has a worst-case time complexity of $\Theta(n)$, use the *aggregate method* to determine an upper-bound on the worst-case time complexity of any sequence of m consecutive NEW-TASK(x) operations, starting from an empty collection (i.e. $task = 0$). Answer in terms of m (the number of operations) and n (the size of T). Make your bound as tight as possible, and show your working.

c. [5 marks] Based on your answer to part (b) of this question, what is the *amortised cost* per NEW-TASK operation, over a sequence of m consecutive NEW-TASK operations, starting from an empty collection?

Question 5 [25 marks total]

You have just been appointed as the manager of the first team at Redlands United Football Club. You have been assigned a budget of `B` (measured in thousands of dollars) to spend on new players to improve the performance of your team. Luckily, you have a smart associate who has already done some scouting and has provided you with some candidate players to help you spend your budget wisely.

Within your team, there are n positions that need to be filled, $p_1, p_2, ..., p_n$.

- Some positions need improvement more than others. The need for a player in position p_i is a real number `N[`i`]`, where for all i, $0 \leq$ `N[`i`]` ≤ 1.
- For each of these positions, your associate has scouted two candidate players to sign, player a and player b. You may only sign one player in each position.
- The cost of player a in position p_i is given by `Va[`i`]` (measured in thousands of dollars). If you choose to sign player a to position p_i, the value added to your squad is `N[`i`]` $\times$ `Va[`i`]`.
- The cost of player b in position p_i is given by `Vb[`i`]` (measured in thousands of dollars). If you choose to sign player b to position p_i, the value added to your squad is `N[`i`]` $\times$ `Vb[`i`]`.
- Since all the n positions need to be filled, it is safe to assume that $\sum_{i=1}^{n} min($`Va[`i`], Vb[`i`]`$) \leq$ `B`.

Your goal as manager is to maximise the value added to your squad by signing new players. You must however ensure that you remain within the budget, so the sum of the values of the signings you make must be less than or equal to `B`.

a. [10 marks] Let `Max-Added-Value`(i, b) be the maximum added value you can achieve by signing players in positions $p_{i+1}, ..., p_n$ with a remaining budget of b, where $0 \leq i \leq n$ and $0 \leq b \leq$ `B`. The solution to the overall problem is `Max-Added-Value`$(0,$ `B`$)$.
Give a recurrence defining `Max-Added-Value`(i, b) for $0 \leq i \leq n$ and $0 \leq b \leq$ `B`.
You do NOT have to give a dynamic programming solution, just the recurrence.
Be sure to define the base cases of the recurrence as well as the more general cases.

b. [10 marks] For the dynamic programming solution, indicate in what order the elements of the matrix `Max-Added-Value` corresponding to the recurrence should be calculated. As part of answering this question, give pseudocode indicating the evaluation order.

c. [5 marks] What is the time complexity of your dynamic programming algorithm? Justify your answer.

END OF EXAMINATION