The University of Melbourne
COMP20007: Design of Algorithms
Semester 1 Final Exam 2020

Writing Time Three hours.

Instructions to Students

- This exam counts for 70% of your final grade if you have not taken the end-of-semester test or if the mark in the end-of-semester test is lower than the mark in this exam. The exam counts for 50% and the end-of-semester test for 20% of your final grade if your score in the end-of-semester test is higher than your score in this exam.
- You must complete the exam by 6pm AEST. You will be given at most 30 minutes to scan and upload the exam. Submissions after 6:30pm AEST will not be accepted (this does not apply for students with alternative exam arrangements).
- This exam has 8 questions with a total of 70 marks.
- For all questions you **must** show **all** working and provide sufficient **justification**. A correct answer that does not show your working will result in 0 marks.
- The test is open book, which means you may only use course materials provided via the LMS or the text book but must not use any other resource including the Internet. You must not communicate with other students.
- Solutions must be written on separate pieces of paper with pen or pencil and not be digitally created. You must write your solution to each question on a new sheet of paper and clearly indicate which question you are answering on each page.
- You must indicate which question is answered on which page via Gradescope.
- You must use a Scanner app on your smartphone to scan your solutions, email them to your laptop and then submit a PDF file to Gradescope via Canvas.
- A Gradescope guide to scanning your test can be found here:

  `https://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf`
- If you require technical assistance, you can submit an enquiry via ask.unimelb or call 13 MELB (13 63 52).

Question 1 [10 marks]

(a) Explain whether or not the following statement is correct:

$$\frac{n^2 \cdot \sqrt{n^3}}{n^3+1} \in \Omega(n)$$

(b) Explain whether or not the following statement is correct:

$$\prod_{i=1}^{n}(i+n) \in O(n^{2n})$$

(c) Explain whether or not the following statement is correct:

$$\sum_{i=0}^{n} 3^i \in \Theta(3^n)$$

(d) Consider an algorithm A, whose runtime is dependent on some "size" variable n of the input. Explain the difference between the two statements below, and give an explicit example of an algorithm for which one statement is true but the other is false.

1. *The worst case time complexity of A is* n^2.
2. *A is* $O(n^2)$.

(e) Give an example of an algorithm (with a clear input type) which has a Big-Oh (O) and Big-Omega (Ω) bound on its time complexity, but no Big-Theta (Θ), and explain why. Make sure to state the actual O and Ω bounds you are claiming for your algorithm.

Common facts and notation

We know from the lectures that for $0 < \varepsilon < 1 < c$: $1 \prec \log n \prec n^{\varepsilon} \prec n^{c} \prec n^{\log n} \prec c^{n} \prec n^{n}$.

Remember also that for $a > 0$: $\sum_{i=0}^{n}(a^i) = \frac{1-a^{n+1}}{1-a}$.

We remind you also of the following conventions:

$$\sum_{i=m}^{n}(s_i) = s_m + s_{m+1} + \ldots + s_{n-1} + s_n \quad \text{and} \quad \prod_{i=m}^{n}(p_i) = p_m \cdot p_{m+1} \cdot \ldots \cdot p_{n-1} \cdot p_n$$

Question 2 [7 marks]

(a) Your company participates in a competition and the fastest algorithm wins. You know of two different algorithms that can solve the problem in the competition.

- Algorithm 1 solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm 2 solves problems of size n by dividing them into 16 subproblems of size $n/4$, recursively solving each subproblem, and then combining the solutions in $\Theta(n^2)$ time.

Which algorithm is the faster one? Explain your answer in detail. You may use the Master Theorem to answer this question.

Master Theorem

For integer constants $a \geq 1$ and $b > 1$, and function f with $f(n) \in \Theta(n^d), d \geq 0$, the recurrence

$$T(n) = aT(n/b) + f(n)$$

(with $T(1) = c$) has solutions, and

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Note that we also allow a to be greater than b.

(b) The following algorithm is written by the head chef at PizzaMash to determine what the maximum profit they can receive by selling a long pizza which is L centimetres in length in smaller, variable length slices.

They know that the price they can sell an i centimetre slice for is $S[i]$.

```
function PizzaProfit(L)
    if L == 0 then
        return 0
    p ← 0
    for i in 1...L do
        p' ← S[i] + PizzaProfit(L − i)
        if p' > p then
            p ← p'
    return p
```

Unfortunately this algorithm is an exponential time algorithm.

Describe how you could use Dynamic Programming to improve the efficiency of this algorithm. Give both the time complexity and space complexity of the improved algorithm.

Question 3 [9 marks]

Assume that the array A has n distinct elements. You may further assume that each element is an integer. Professor Swift has decided to develop an efficient algorithm *DuoSum* that returns for a given number called *key* if there are two distinct numbers in the array that add up to *key*. If such a duo (i.e., two elements) can be found, the algorithm returns $True$, and $False$ otherwise.

```
function DuoSum(A[1..n], key)
    MergeSort(A[1..n])
    i ← 1
    j ← n
    while i < j do
        if A[i] + A[j] = key then
            return True
        else if A[i] + A[j] < key then
            i ← i + 1
        else
            j ← j − 1
    return False
```

(a) Explain why the algorithm is correct, that is, the algorithm finds a duo for a given key if such a duo exists and reports $True$, and correctly reports $False$ otherwise.

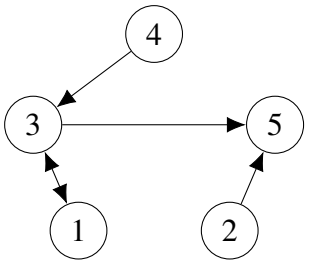
(b) Provide a detailed discussion for the time complexity of the algorithm.

(c) If we replace MergeSort with QuickSort, does the complexity of Professor Swift's algorithm change? Explain your answer.

(d) Professor Swift has an even better idea: she can modify her original algorithm *DuoSum* to determine if the array A contains three elements that sum up to zero. Describe the key steps to change her algorithm. What time efficiency does the modified algorithm achieve?

Question 4 [8 marks]

(a) Consider the following graph.



Use Warshall's algorithm to compute the transitive closure of this graph. You need to give the adjacency matrix after each step of the algorithm. Give your final answer as an adjacency matrix.

(b) A *source* in a directed graph is a node with no incoming edges. A *sink* is a node with no outgoing edges.

Assume that we know that every DAG has at least one sink. Use this fact to explain why every DAG must have at least one source.

(c) Consider the following graph algorithm which takes a DAG G as input.

```
function ComputeSomething(DAG G)
    L ← empty linked list
    S ← stack of all source nodes in G
    while S is non-empty do
        u ← Pop(S)
        InsertAtTail(L, u)
        for each outgoing edge (u, v) do
            remove edge (u, v) from graph G
            if v has no other incoming edges then
                Push(S, v)
    return L
```

Explain what the output of this algorithm is, and briefly explain why it works.

Question 5 [6 marks]

You are given an array A of integers in sorted order. However, you do not know the length n of the array. Assume that in our programming language arrays are implemented in such a way that you receive an out-of-bounds error message whenever you wish to access an element A[i] with $i > n$.

For simplicity we assume that the error message simply returns the value `INT_MAX` and that every value in the array is smaller than `INT_MAX`.

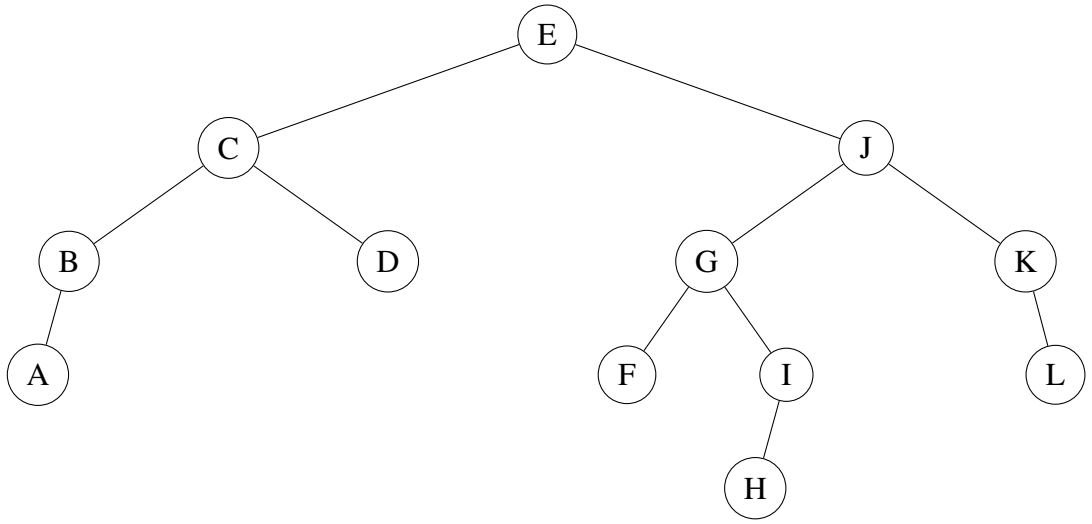
(a) Design an algorithm with $O(\log n)$ time efficiency that takes an integer *key* as input and finds a position in the array containing *key*, if *key* is an element of A.

(b) Explain in detail why your algorithm runs in $O(\log n)$.

Question 6 [11 marks]

(a) Draw a full binary tree where each node stores a key (character), and a pre-order traversal of the tree visits the nodes in the order "`a l m n x y z`" and a post-order traversal visits the nodes in the order "`m n l y z x a`".

(b) You are given the AVL Tree in the figure below. Assume that the nodes are sorted in alphabetical order.



Draw the resulting BST after node E is removed. To construct the new BST replace node E with an appropriate node from the left subtree of E.

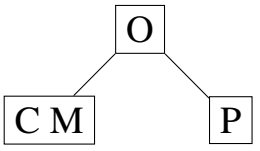
Do not rebalance the resulting tree.

Label each node in the resulting tree with its balance factor.

(c) Now rebalance the tree from the previous task. Draw a new tree for each rotation that occurs when rebalancing the tree. You do not need to label the trees with the balance factors.

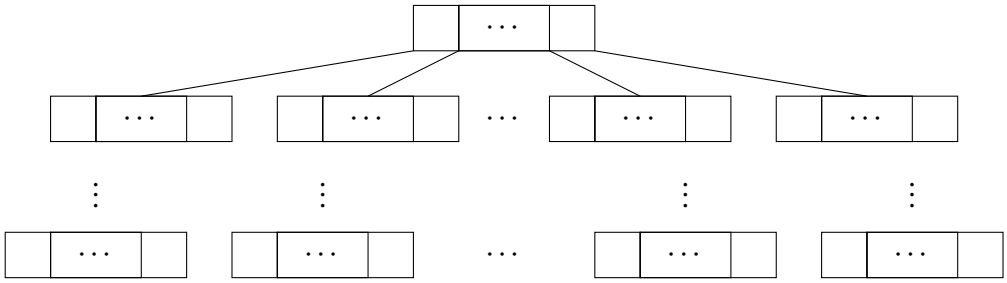
Hint: You may need to perform more than one rotation to rebalance the tree.

(d) We have the following 2-3 tree of characters:



Your task is to insert the characters `L`, `E`, `T`, `D`. Show the state of the 2-3 tree after each insertion.

(e) Consider a generalised 2-3 tree, in which each node contains an array of up to k elements in increasing order and an array of up to $k+1$ children. This data structure may look something like



Explain how you would look up a value in this generalised 2-3 tree. What is the worst case time complexity if the tree has n distinct elements? Give your answer in terms of n and k.

Question 7 [10 marks]

Consider a hash table with 9 slots. We are to insert strings into this hash table with the hash function:

$$h(s) = (\#\text{ of 'a's in the string } s) \mod 9$$

The hash table will use linear probing with a step size of 3.

(a) Show the state of the hash table after inserting the following strings:

`alabama`, `hat`, `canada`, `naan`, `car`

(b) Describe which slots must be checked to lookup `car` in the hash table.

(c) Describe which slots must be checked to lookup `japan` in the hash table.

(d) Is our hash function a good hash function? Explain why or why not.

(e) Illustrate how Horspool's algorithm would search for the pattern $P =$ `"baa"` in a text of 12 'a's, *i.e.,* $T =$ `"aaaaaabbbbbb"`. Exactly how many comparisons are performed?

(f) If we are to use Horspool's algorithm to search for the pattern $P =$ `"abcd"` in a text T of length 12, what is the maximum number of comparisons that may be performed? Provide an example of such a text T which will achieve this maximum number of comparisons.

Question 8 [9 marks]

(a) Show the state of the following array after performing a Hoare partition. Sort in ascending order and choose the first element to be the pivot.

$$\left[\, 5,4,8,1,2,9,7,3 \,\right]$$

(b) Assume that an algorithm called MFA (for Median Finding Algorithm) can find the median of an array of n elements in linear time. You decide to modify the QUICKSORT algorithm in the following way: each time PARTITION is called, the median of the partitioned array is found using the MFA. This median value is subsequently used as the pivot. Does this change the worst case complexity of QUICKSORT? Explain your answer.

(c) Consider the following hybrid sorting algorithm which runs INSERTIONSORT for arrays of small length (*i.e.,* $n \leq k$ for the threshold k), and MERGESORT otherwise.

```
function HybridSort(A[1...n])
    if n ≤ k then
        InsertionSort(A[1...n])
    else
        MergeSort(A[1...n])
```

What is the worst case time complexity of HYBRIDSORT in terms of n and k (give your answer as a Big-Theta expression). You must justify your solution.

(d) How large can k be (as a Big-Theta expression involving n) without making HYBRIDSORT slower than MERGESORT? You must justify your solution.

END OF EXAM