The University of Melbourne
School of Computing and Information Systems
COMP90038 Algorithms and Complexity

Assignment 1, Semester 1 2021

Deadline: Wednesday April 14, 9:00am
Marks available: 30 marks (15% of final assessment)

Objectives

To improve your understanding of the time complexity of algorithms and recurrence relations. To develop problem-solving and design skills. To improve written communication skills; in particular the ability to present algorithms clearly, precisely and unambiguously.

Problems

1. **[3 marks]**

   Use repeated substitution (or telescoping) to determine the asymptotic upper bound for the running time of an algorithm defined by the recurrence relation below:

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ 4T(\frac{n}{2}) + d\,n & \forall n > 1 \end{cases}$$

   where: c and d are constants. Assume that n is a positive value and is a power of 2.

2. **[6 marks]**

   In this question, you must fill in the table below.

   Assuming that there is always at least one negative value and at least one positive value in the given data structure (either an array or a linked list for the purpose of this question), what is the worst-case run time when searching for the first negative value (the *key*) in the data structure given the constraints listed in the 'Order of elements' column?

Data structure	**Order of elements**	**Time complexity**	**Brief justification**
Array	Random order		
Array	Sorted in ascending order		
Array	Sorted in descending order		
Linked list	Random order		
Linked list	Sorted in ascending order		
Linked list	Sorted in descending order		

3. **[9 marks]**

Consider an integer array A that contains n distinct elements.

A student in COMP90038 has developed an algorithm CheckSum that can be used to determine if there are two distinct numbers in the array that add up to a *key*, which is passed as an argument to the function.

```
function CheckSum(A[0..n-1], key)
    SelectionSort(A[0..n-1])                    ▷ function is available to use
    $i \leftarrow 0$
    $j \leftarrow n - 1$
    while i < j do
        if A[i] + A[j] = key then
            return True
        else
            if A[i] + A[j] < key then
                $i \leftarrow i + 1$
            else
                $j \leftarrow j - 1$
    return False
```

(a) (2 marks) Another student in COMP90038 suggested that the algorithm is incorrect, however, you are convinced that the algorithm is in fact correct. Present a clear and concise argument (maximum of 100 words) that you could share with the other student explaining why the CheckSum algorithm is correct.

(b) (2 marks) What is time complexity of the CheckSum algorithm? Justify your answer.

(c) (2 marks) In lecture 11, the MergeSort algorithm is introduced. MergeSort has a time complexity of $\Theta(n\,log\,n)$. If we replace SelectionSort with MergeSort in the CheckSum algorithm, will the time complexity of the CheckSum algorithm change? Justify your answer (maximum of 50 words).

(d) (3 marks) One of the tutors in COMP90038 suggested that the CheckSum algorithm can be modified to determine whether the array A contains three elements that sum up to zero.

Describe how you would change the algorithm listed above to meet this new requirement. Your description does not have to be in pseudocode. However, the expectation is that any text description will be less than 100 words.

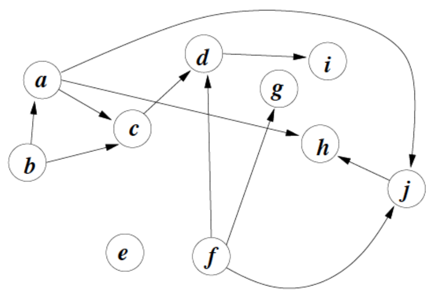
What time efficiency does the modified algorithm achieve?

4. **[3 marks]**

   Levitin suggests that a decrease-and-conquer approach can be used for the topological sorting problem. The idea is to repeatedly remove a source (a node with no incoming edges) and list the order in which such nodes were removed.

   Describe an algorithm for this version of topological sort. Full marks will be given for an algorithm that runs in time $O(|V| + |E|)$.

   Demonstrate your algorithm running using the following DAG.



5. **[9 marks]**

   (a) (5 marks) Design an algorithm which, given an array $D[0..n-1]$ ($n > 1$), constructs a graph with nodes $0, .., n-1$ such that node i has degree $D[i]$. Your algorithm should return the constructed graph using an adjacency list format. That is, your algorithm should return an array A whose elements are a list of nodes and corresponding edges consistent with the representation shown on the lecture slides: Lecture 7 - slide 13.

   You may assume that the function ADDEDGE(A, i, j), which adds node i into adjacency list $A[j]$ and node j into adjacency list $A[i]$, is available. Your algorithm should use this function when constructing the graph.

   It is important to note that it is possible that multiple graphs could be constructed satisfying the requirements listed above. However, your algorithm should just produce any one of the correct graphs. For example, given input array $D = [1, 2, 2, 1, 1, 3]$, it might produce this graph representation:

$$\begin{aligned}
A[0] &= 2 \\
A[1] &= 2 \rightarrow 5 \\
A[2] &= 0 \rightarrow 1 \\
A[3] &= 5 \\
A[4] &= 5 \\
A[5] &= 1 \rightarrow 3 \rightarrow 4
\end{aligned}$$

   Your algorithm must be written in unambiguous pseudocode. You should try to make your algorithm as efficient as you can.

   (b) (2 marks) Explain how your algorithm works (maximum of 100 words).

   (c) (2 marks) State the time complexity of your algorithm as a function of $|V|=n$ and $|E|=sum(D[i])/2$

Submission and evaluation

- You must submit a PDF document via the LMS. Note: handwritten, scanned images, and/or Microsoft Word submissions are not acceptable — if you use Word, create a PDF version for submission.
- Marks are primarily allocated for correctness, but elegance of algorithms and how clearly you communicate your thinking will also be taken into account. Where indicated, the complexity of algorithms also matters.
- Where a question asks for an explanation / justification / description, your response should be **written in English**. Your response should be clear, concise and adhere to the requested length when specified. Excessively long answers may be penalised.
- **Please write any pseudocode following the format suggested in the examples provided in this assignment specification, lecture slides and/or the textbook**. Take care with indentation, loops, if statements, initialisation of variables and return statements.

  Python code is **NOT** acceptable for this assignment.
- Make sure that you have enough time towards the end of the assignment to present your solutions carefully. Time you put in early will usually turn out to be more productive than a last-minute effort.
- You are reminded that your submission for this assignment is to be your own individual work. For many students, discussions with friends will form a natural part of the undertaking of the assignment work. However, it is still an individual task. You should not share your answers (even draft solutions) with other students. Do not post solutions (or even partial solutions) on social media. It is University policy that cheating by students in any form is not permitted, and that work submitted for assessment purposes must be the independent work of the student concerned.

  Please see https://academicintegrity.unimelb.edu.au

If you have any questions, you are welcome to post them on the LMS discussion board. You can also email the Head Tutor, Lianglu Pan <`lianglu.pan@unimelb.edu.au`> or the Lecturer, Michael Kirley <`mkirley@unimelb.edu.au`>. In your message, make sure you include COMP90038 in the subject header. In the body of your message, include a precise description of the problem.

Extension policy: obviously COVID has impacted on all students. We have carefully taken this issue into consideration when designing the questions and the time window available to attempt the assignment. It is in your best interest to complete the assignment by the due date so that there is ample time to complete the remaining assessment tasks in this subject.

Late submission will be possible, however **a late submission penalty of 3 marks per day may apply**.