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

Assignment 2, Semester 1 2022

Deadline: Sunday May 29, 23:59
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. **[10 marks]**

   Alex loves skiing and, in order to keep gaining speed, they prefer to always ski downwards. Alex collected the measured altitude of an area that they plan to go next month, represented using an array of size n by n. An example with $n = 4$ is given below:

   $$\begin{bmatrix} 4 & 12 & 15 & 20 \\ 6 & 28 & 23 & 11 \\ 9 & 33 & 50 & 43 \\ 18 & 22 & 47 & 10 \end{bmatrix}$$

   Alex can start skiing from any cell and move to an adjacent cell on the right, left, up or down (no diagonals), anytime as needed. They will always ski towards an adjacent cell with a strictly lower altitude. In the above example, one possible skiing path is $50-23-15-12-4$. Of course, $50-33-28-23-15-12-4$ is another one, and in fact the longest possible one.

   (a) Write a function $LongestSkiingPath(M[0..n-1][0..n-1])$ in pseudocode, which takes a 2-D array representing for the measured altitude of an area as input and calculates the number of cells involved in the longest path, using Dynamic Programming. Using the above example, your algorithm should return 7.

   (b) What's the time complexity of your algorithm? Briefly justify your answer.

   Note: Partial marks will be awarded to working but less efficient implementations.

2. **[15 marks]**

Drug discovery is a costly and time consuming process that usually involves experimentally testing molecules (drug candidates) in a wet-lab in order to assess their efficacy. More than often, *combinations of molecules* could lead to better efficacy.

The intuition behind using combination of molecules is that, if a molecule A has good efficacy, and another molecule B also does, one might expect them to be as good or better together than individually (even though this might not always hold).

Additionally, the number of possible combinations of molecules grows quite quickly with the number of molecules (and testing molecules costs a lot of money!), meaning we need to be clever while designing ways to evaluate sets of molecules.

This process involves automation of the wet-lab tests and you have been assigned with the task of *defining a strategy to optimise combinations of molecules* against their antiviral effect.

You receive a very long list of n molecules as input and is also given access to the automation robot that will do the wet-lab tests, through a programming interface, via the function $test\_wet\_lab\_robot(mol[1...k])$. It receives a list of one or more molecules as input and returns a positive number denoting the efficacy of the molecule(s) (the larger this value, the better!).

Your job is to develop algorithms that optimises molecule efficacy, given a set of molecules, using the robot as little as possible.

(a) Describe with your own words (and in pseudocode) two different greedy approaches to optimise the combination of molecules, one with linear time complexity ($O(n)$) and the other with quadratic time complexity ($O(n^2)$), where n is the number of input molecules. Remember that the most costly operation is the function $test\_wet\_lab\_robot$.

(b) Do any of your approaches always find the optimal solution? (e.g., does this problem have optimal substructure?) Briefly justify.

3. **[5 marks]**

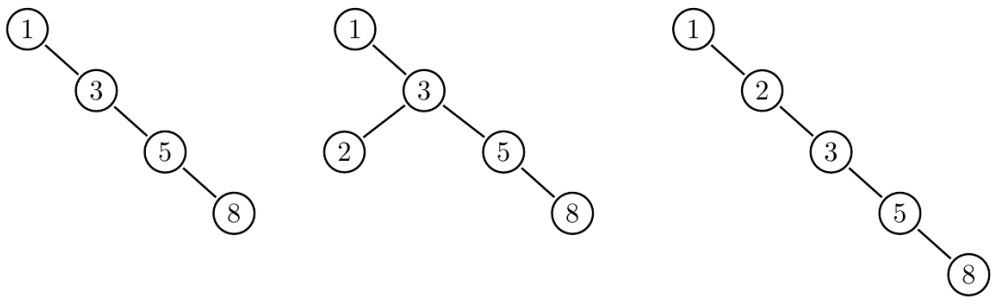
You are a hacker that recently got hired by a cybersecurity company. The team needs to be preemptive against any adversarial attacks. Your role is to develop such attacks so the defense team can prepare mechanisms to avoid them.

In this problem, you were able to inject code into the algorithm for inserting an element into a Binary Search Tree. Your goal is to **force the BST to always degrade to a "stick" (to the right), or more specifically, a linked list**. The algorithm to insert an element into the BST is as follows.

```
function BSTInsert(root, new)
    if new.value < root.value then
        if root.left = NULL then
            root.left ← new
            Stickify(new, root)
        else
            BSTInsert(root.left, new)
    if new.value > root.value then
        if root.right = NULL then
            root.right ← new
            Stickify(new, root)
        else
            BSTInsert(root.right, new)
```

Here *root* and *new* are nodes with fields *left* and *right* (which are pointers to other nodes) and a field *value*. If the BST is already a "right stick", then running the BSTInsert algorithm (with your Stickify function) should always leave the tree as a "right stick".

For example if we start with the tree on the left and insert 2 we will get the tree in the middle. After Stickify is run, we should get the tree on the right.



Your task is to write the pseudocode for the algorithm Stickify which takes the newly inserted node along with its parent and performs any necessary rotations to ensure the tree remains a "right stick". You may use RotateRight(*node*) and RotateLeft(*node*) without implementing these functions. In the above example, the RotateRight(3) should be called.

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.
- We've created a *Frequently Asked Questions* section on the LMS and will update it regularly to clarify doubts. The *Frequently Asked Questions* section also forms part of the task specification. We will NOT add anything into the *Frequently Asked Questions* section in the last 48 hours before the assignment is due.
- 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, Douglas Pires <`douglas.pires@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: We have carefully designed the questions so that it is adequate considering 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**.