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

Assignment 2, Semester 1 2021

Deadline: Sunday May 30, 23:59
Marks available: 30 marks (15% of final assessment)

Objectives

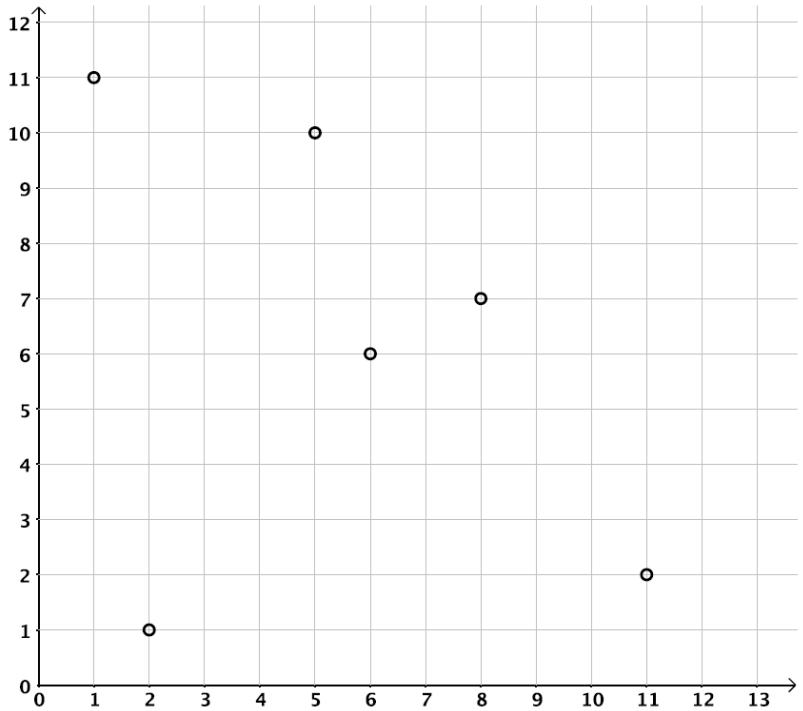
To improve your understanding of the time complexity of algorithms and recurrence relations. To develop problem-solving and design skills. To improve written communication skills; in particular the ability to present algorithms clearly, precisely and unambiguously.

Problems

1. **[10 marks]**

   Let S be a finite set of n distinct points in $\mathbb{R}^2$. We say that a point $P = (x, y) \in S$ *governs* another point $P' = (x', y') \in S$ iff $x' \leq x$ and $y' \leq y$. A point is called *pareto optimal* for S if it is not governed by any other point in the set S. Write pseudocode for an algorithm that prints all the *pareto optimal* points of S. Your algorithm must run in $\mathcal{O}(nlogn)$ time for full marks.

   You may assume that the input are two coordinate arrays X and Y. For example, with the input $X = [11, 5, 6, 1, 8, 2]$ and $Y = [2, 10, 6, 11, 7, 1]$, your algorithm should print $(5, 10)$, $(1, 11)$, $(11, 2)$ and $(8, 7)$ (not necessarily in that order).



2. **[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**.

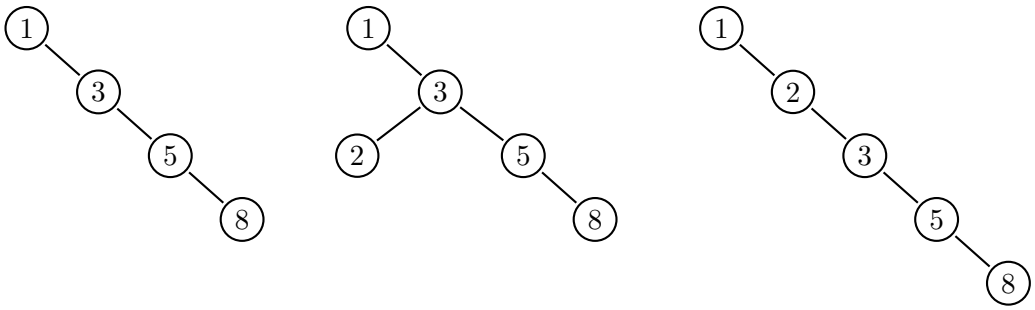
The algorithm to insert an element into the BST is as follows.

```
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.

3. **[5 marks]**

Consider the following hybrid sorting algorithm which runs standard MergeSort but when one of the subarrays reach size 10 or smaller, it calls InsertionSort on that array:

function HybridSort($A[0..n-1]$)
  if $n > 10$ **then**
    $B[0..\lfloor n/2 \rfloor - 1] \leftarrow A[0..\lfloor n/2 \rfloor - 1]$
    $C[0..\lceil n/2 \rceil - 1] \leftarrow A[\lfloor n/2 \rfloor..n-1]$
    HybridSort($B[0..\lfloor n/2 \rfloor - 1]$)
    HybridSort($C[0..\lceil n/2 \rceil - 1]$)
    Merge($B[0..\lfloor n/2 \rfloor - 1], C[0..\lceil n/2 \rceil - 1], A[0..n-1]$)
  else
    InsertionSort($A[0..n-1]$)

What is the worst case time complexity of HybridSort in terms of n? Justify your answer.

4. **[10 marks]**

Consider a two-player game. An array is given, containing n non-negative integers. Each player will take turn, to take one integer from either the beginning or end of the remaining array. This process is repeated until the array becomes empty. For both players, the goal is to maximise the sum of integers taken. Assume that both player will take their best move at each step.

(a) Provide an equation of the recursive relationship, for maximising the sum of integers taken. You may optionally explain it in a few sentences.

(b) Based on this recursive relationship, write the algorithm FirstPlayerSum($A[0..n-1]$) to calculate the sum of integers taken by the first player.

(c) State the time complexity of your algorithm.

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: 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**.