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

Assignment 2, Semester 2, 2019

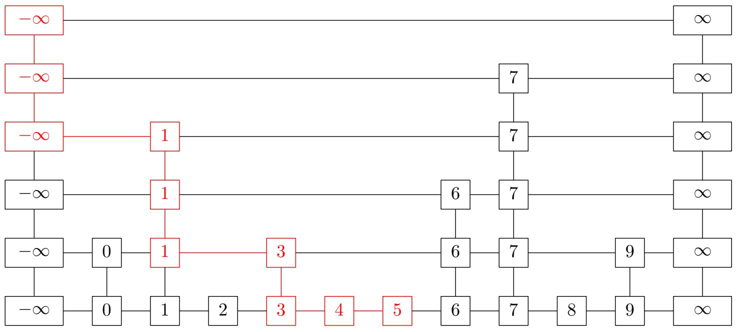
Released: Thursday September 26th. Deadline: Sunday October 20th at 23:59

Objectives

To improve your understanding of data structures and algorithms for sorting and search. To consolidate your knowledge of trees and tree-based algorithms. To develop problem-solving and design skills. To develop skills in analysis and formal reasoning about complex concepts. To improve written communication skills; in particular the ability to use pseudo-code and present algorithms clearly, precisely and unambiguously.

Problems

1. [5 Marks] A *skip list* is a randomised binary search structure, which has many of the desirable properties of a balanced tree. It is a sorted linked list with random shortcuts, as illustrated in the figure below. Search in a single-linked list of length n takes $\mathcal{O}(n)$. To speed it up, we construct another list containing about half of the items from the original, i.e., first, each item in the original list is duplicated with probability 1/2; then, all the duplicates are linked up; finally, we add a pointer from each duplicate back to the original. Sentinels are placed at the beginning and at the end of each list. We repeat recursively until the last sorted list contains sentinels only.



The search algorithm is presented below. We assume that each node has the fields $\{key, up, down, left, right\}$, where key represents the key value, and the remaining four are pointers to the adjacent nodes or NULL. Starting at the leftmost node L in the highest level, we scan through each level as far as we can without passing the target value x, and then proceed down to the next level. The search ends when we find the node with the largest key that is less or equal than x. Since each level of the skip list has about half of the nodes as the previous, the total number of levels is $\mathcal{O}(\log n)$, and the overall expected search time is $\mathcal{O}(\log n)$.

```
1: function SEARCH($x, L$)
2:     $u \leftarrow L$
3:     while $u.down \neq$ NULL do
4:         $u \leftarrow u.down$                 ▷ Drop a level down
5:         while $x \geq u.right.key$ do
6:             $u \leftarrow u.right$               ▷ Scan forward
7:     return $u$
```

(a) [2 Marks] Unscramble the routine INSERTNODE (x, u), which inserts a node with key x after u. This routine uses a function COINFLIP () that returns TRUE or FALSE with 1/2 probability, a function NEWNODE (x, u, v) which creates a new node with x as key after u and above v, and a function ISHEAD (u) that tests whether the u is the head of the list.

```
 1: $u \leftarrow u.up$
 2: while COINFLIP ()
 3: $v \leftarrow$ NEWNODE $(x, u, v)$
 4: if ISHEAD $(u)$ then
 5: $u \leftarrow u.left$
 6: function INSERTNODE $(x, u)$
 7: $v \leftarrow$ NEWNODE $(x, u,$ NULL$)$
 8: while $u.up =$ NULL and $u.left \neq$ NULL do
 9: $w \leftarrow$ NEWNODE $(-\infty,$ NULL$, u)$
10: NEWNODE $(\infty, w, v.right)$
11: return $v$
```

(b) [3 Marks] Now suppose that you are given two skip lists, one storing a set A of **unique** m keys, and another a set B of **unique** n keys, with $m > n$. There are **no repeats** between lists. Design a routine called MERGE (A, m, B, n) that merges B into A, storing the set $A \cup B$ in $\mathcal{O}(m + n)$ expected time. **Do not assume that every key in A is smaller than every key in B.** This routine should have about 15 lines of code.

2. [4 Marks] Consider two sets of integers, $X = [x_1,\ x_2,\ \ldots,\ x_n]$ and $Y = [y_1,\ y_2,\ \ldots,\ y_n]$. Write two versions of a FINDUNCOMMON(X, Y) algorithm to find the uncommon elements in both sets. Each of your algorithms should return an array with the uncommon elements, or an empty array if there are no uncommon elements.

You may make use of any algorithm introduced in the lectures to help you develop your solution. That is, you do not have to write the 'standard' algorithms – just use them. Therefore, you should be able to write each algorithm in about 10 lines of code. **You must include appropriate comments in your pseudocode.**

(a) [2 Marks] Write a pre-sorting based algorithm of FINDUNCOMMON(X, Y). Your algorithm should strictly run in $\mathcal{O}(n \log n)$.

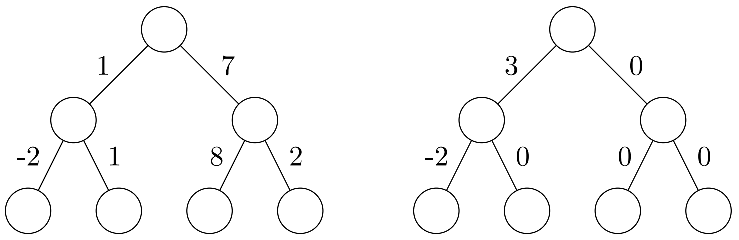
(b) [2 Marks] Write a Hashing based algorithm of FINDUNCOMMON(X, Y). Your algorithm should run in $\mathcal{O}(n)$.

3. [6 Marks] Klutzy Pty Ltd's CEO has asked you to help organise its end of the year *splurge*. Everybody invited, **one of which must be the CEO**, has to attend. Employees at Klutzy are organised into a strict hierarchical structure resembling a binary tree, where the root node represents the CEO. The data scientists at human resources have assigned to each employee an *awkwardness* score, α.

An $\alpha > 0$ indicates that the staff member and its supervisor dislike each other, whereas an $\alpha < 0$ indicates that they actually like each other. If the guest list does not include an employee and its supervisor, then the added awkwardness is zero. To help you model Klutzy Pty Ltd assume that each node has the following fields: (a) L is a pointer to the left child or NULL if there is no left child; (b) R is a pointer to the right child or NULL if there is no right child; and (c) *alpha* is the awkwardness score of the inviting the employee and its supervisor. Provide a method that finds the score of the least awkward party, i.e., minimise the overall awkwardness score. For example, your algorithm should return '-1' and '0' for the two trees in the figure below. You must provide:

(a) [2 Marks] A description of the optimisation problem to be solved, including an equation of the recursive relationship.

(b) [4 Marks] The pseudo-code of your method, in about 15 lines of code.



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.

- We expect your work to be neat — parts of your submission that are difficult to read or decipher will be deemed incorrect. 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.

- Number of lines are given as an **indication only** and should not be considered as the actual length of the code. Correct solutions could have a few lines more or less depending on your notation, but not many more (if you see yourself writing ten more extra lines, then you are probably doing something wrong).

- 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 or the discussion board. 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 *so long as you do not reveal details about your own solutions*. You can also email the Head Tutor, Lianglu Pan (lianglu.pan@unimelb.edu.au) or the Lecturer, Andres Munoz-Acosta (munoz.m@unimelb.edu.au). In your message, make sure you include COMP90038 in the subject line. In the body of your message, include a precise description of the problem.

Late Submission and Extension

Late submission will be possible, but a penalty will apply: a flag-fall of 2 marks, and then 1 mark per 12 hours late. Extensions will only be awarded in extreme/emergency cases, assuming appropriate documentation is provided. Simply submitting a medical certificate on the due date will not result in an extension.