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

Assignment 2, Semester 1 2019

Deadline: Monday May 27, 9:00am Marks available: 30 marks (15% of final assessment)

Objectives

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

1 Sorting and Hashing

Consider the function FindDuplicates($A[0..n-1]$) that returns a list of all duplicate items in the unsorted integer array A of size n. For example, given the array [3, 2, 4, 3], the function should return the value 3. For the array [1,2,3,5,5,5,6,6,8], the function should return an array (or list) [5,6].

In this task, you will develop two alternative solutions for the FindDuplicates($A[0..n-1]$) function. In your implementations, you may call any of the algorithms introduced in the lectures and/or 'generic' functions/operations/methods to help you develop your solution. That is, you do not have to write the 'standard' algorithms/functions – just call them with an appropriate comment.

(a) **[4 marks]** Write a pre-sorting based algorithm for the FindDuplicates($A[0..n-1]$) function. Your algorithm should strictly run in $O(n\ log\ n)$ time.

(b) **[4 marks]** Write a hashing based algorithm for the FindDuplicates($A[0..n-1]$) function. Your algorithm should run in $O(n)$ time.

2 Knapsack Problem

In a Knapsack problem, given n items $\{I_1, I_2, \cdots, I_n\}$ with weight $\{w_1, w_2, \cdots, w_n\}$ and value $\{v_1, v_2, \cdots, v_n\}$, the goal is to select a combination of items such that the total value V is maximized and the total weight is less or equal to a given capacity W.

$$V = \max \sum_{i=1}^{n} v_i x_i, \tag{1}$$

$$s.t. \sum_{i=1}^{n} w_i x_i \leq W. \tag{2}$$

In this question, we will consider two different ways to represent a solution to the Knapsack problem using

- an array with size n: $X = [x_1, x_2, \cdots, x_n]$
- linked list L that stores the indices of items that are included in a solution.

For example, if a solution only includes items I_2, I_3 and I_5, the binary array representation will be $[0, 1, 1, 0, 1, 0, \cdots, 0]$ and the linked list representation will be $head \rightarrow 2 \rightarrow 3 \rightarrow 5$.

(a) **[4 marks]** You decide to use a Monte Carlo-style simulation approach to investigate possible solutions to the Knapsack problem. In this approach, you randomly generate m feasible solutions $\{X_1, X_2, \cdots, X_m\}$ with objective values $\{V_1, V_2, \cdots, V_m\}$, where X_i can be represented using either an array or list as described above.

Write a function ComputeMean(*argument list*) to calculate the mean value of each variable x_j $(j = 1, 2, \cdots, n)$ in the given m samples: $\bar{x}_j = \frac{1}{m} \sum_{i=1}^{m} X_i[j]$.

Note: the format of the *argument list* will depend on the representation you have selected.

Full marks will be given if your algorithm runs in $O(\sum_{i=1}^{m} |L_i|)$ time, where $|L_i|$ denotes the size of the linked list representation of X_i.

(b) **[2 marks]** Briefly describe how a greedy algorithm (heuristic) could be developed for solving the Knapsack problem using the ComputeMean( ) function.

- You do not have to write an algorithm in pseudo code to answer part (b) of this question. We are expecting that you write a short paragraph or a short list of bullet points describing the important steps of the algorithm.
- You should also comment on whether such a greedy approach would be effective.

3 AVL Trees

Assume the following notation/operations on AVL trees. An empty AVL tree is denoted E. A non-empty AVL tree T has three attributes:

- The key $T.key$ is the root node's key.
- The left child $T.left$ is T's left subtree, which is an AVL tree (possibly E).
- The right child $T.right$ is T's right subtree, which is an AVL tree (possibly E).

(a) **[5 marks]** Write a function RangeCount(T, lo, hi) to count the number of nodes in an AVL tree with root T, where the key value is in the range $lo \leq key \leq hi$. Your algorithm should run in $O(n)$ time, assuming that n is the number of nodes in the AVL tree, T.

(b) **[3 marks]** Describe an alternative version of the RangeCount(T, lo, hi) function that runs in $O(log\ n)$ time.

- You do not have to write an algorithm in pseudo code to answer part (b) of this question. We are expecting that you write a short paragraph or a short list of bullet points describing the important steps of the algorithm.
- In your answer, you may augment the AVL tree notation/operations listed above. For example, you may include additional node attributes such as $T.bal$ indicating the balance factor of the AVL tree, or $T.sum$ indicating the sum of all key values in the tree with root T, and possibly even $T.parent$ that points to the parent node of current root, T. If you elect to do so, please make sure that you explain your representation.

4 Dynamic Programming

Consider the following problem based on the transformation of a sequence (or collection) of coloured disks.

Assume that you have a very large collection of disks, each with an integer value representing the disk colour from the range [0, c]. For example, the colour mapping might be:

0-red, **1**-yellow, **2**-blue, **3**-pink, ..., c-black.

For a given sequence of coloured disks e.g., (0,1,2,3,4), at each time step (or iteration) you are only allowed to perform one of three basic operations:

- remove one disk from the sequence – the disk you select can be any of the disks

  eg., (0,1,2,3,4) → (0,1,3,4)

- insert one disk at any position in the sequence – the disk you select to insert can be any colour in the range [0, c]

  e.g., (0,1,2,3,4) → (0,1,9,2,3,4)

- replace any single coloured disk in the sequence with a different coloured disk – the replacement disk can be any colour in the range [0, c]

  e.g., (0,1,2,3,4) → (0,1,2,3,9)

Note: after multiple operations (or iterations), the length of sequence of disks may be different to the length of the starting sequence. We suggest that you represent each sequence of disks as an array.

(a) **[6 marks]** Write a dynamic programming implementation for the function:

TRANSFORM($A[0..n-1]$, $B[0..m-1]$)

to calculate the minimum number of operations required to transform one given sequence of disks $A[0..n-1]$ into another $B[0..m-1]$. For example, $A[0,1,2,3,4] \rightarrow B[0,1,3,5]$ requires at least two operations – remove 2; replace 4 by 5.

Your function must be written in clear, unambiguous pseudo code.

(*Hint: consider approximate string search*)

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

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.
- Please write any pseudo code following the format suggested in the examples provided in the sample lecture slides and/or the textbook. Take care with indentation, loops, if statements, initialisation of variables and return statements. Cormen et al. (available as an e-book in the library) provide some guidelines for pseudo code (pages 20–22).
- 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.

Late submission will be possible, but **a late submission penalty will apply**: a flagfall of 4 marks, and then 2 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.