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

Assignment 1, Semester 1 2022

Deadline: Friday April 8, 23:59 AEST
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. [7 Marks] Consider the following recursive function, which takes an unordered array of integers A, and an integer k, and returns True if k appears in A and False otherwise.

```
function ThirdsSearch(A[0..n − 1], k)
    if n == 0 then
        return False
    if n == 1 then
        return (A[0] == k)
    a ← ThirdsSearch(A[0..n/3 − 1], k)
    b ← ThirdsSearch(A[n/3..2 × n/3 − 1], k)
    c ← ThirdsSearch(A[2 × n/3..n − 1], k)
    return (a OR b OR c)
```

You may assume the input array is of length $n = 3^m$ for some positive integer m, so that the argument for each recursive call is an exact third of the size of the input array.

(a) Assume that the basic operation is the line **if** $n == \cdots$
Write down, and solve, a recursive formula $W(n) = \cdots$ which describes the number of basic operations taken by ThirdsSearch when processing an array of length n, for the worst case input. Make sure that you include the base case(s).

(b) Use O, Ω, Θ to bound the time complexity of ThirdsSearch, using $T(n)$ for its runtime. You must include an upper and lower bound. Full marks will be awarded for solutions with the tightest possible bounds.

(c) Mr. Clever suggested the following change to the algorithm listed above: if a is True then we can immediately return True without the need to evaluate b or c; similarly, we may check b before evaluating c. With this change, what would be the best case and worst case complexity of ThirdsSearch? Briefly justify your answers.

2. [6 Marks] We define two non-negative integers to be **close friends** if their octal representations (without leading zeros) are permutations of each other. For example, 519 (octal 1007) and 3592 (octal 7010) are **close friends** while 347 (octal 533) and 1579 (3053) are not. As a special case, an integer is considered to be a **close friend** of itself.

   Design an algorithm CLOSEFRIEND($num1$, $num2$) in pseudocode that takes two non-negative integers as input, the algorithm should return TRUE if the two integers are **close friends**, FALSE otherwise.

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

3. [8 Marks] In this task, we will implement a queue using only the stack ADT. You may assume:

   - in addition to the usual PUSH(x) and POP() operations, the stack ADT also comes with a SIZE() operation, which returns the number of elements currently stored;
   - SIZE(), PUSH(x) and POP() are constant-time operations.

   (a) Write pseudocode for the functions ENQUEUE(x) and DEQUEUE(). *Hint: you should not need more than three stacks.*

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

   (b) Use O, Ω and/or Θ to make the strongest possible claims about the complexity of your ENQUEUE(x) and DEQUEUE() functions respectively, considering both the best case and the worst case. Briefly justify your answers.

   (c) Suppose n elements are enqueued and dequeued in some order. What is the worst case complexity for this entire process (calling ENQUEUE(x) and DEQUEUE() n times)? Briefly justify your answer.

4. [9 Marks] Consider the 'traverse the maze problem' that we discussed in the first lecture. Suppose:

   - The maze is a $n \times n$ rectangle stored in a two-dimensional array $M[0..n-1][0..n-1]$, rows and columns are numbered from 0 to $n-1$;
   - $M[i][j] = 1$ indicates (i, j) is an accessible position, $M[i][j] = 0$ indicates we cannot stay or pass through (i, j);
   - The starting point is $(0, 0)$, the goal point is $(n-1, n-1)$;
   - We can traverse the maze by moving up/down/left/right each step, providing that we are moving into an accessible position.

   (a) Explain, in English, how we may model this problem using a graph. Your answer should at least include what is considered to be a vertex/edge in the graph, how to store the graph, etc.

   (b) Explain, in English, how to determine whether there is a path from the starting point to the goal point. Use Big-Oh notation to express the worst case complexity of your approach, in terms of n.

   (c) Explain, in English, how to find a shortest path (minimal number of movements) from the starting point to the goal point.

   NOTE: you may include labelled diagrams of the maze and/or graphs to support your arguments.

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.

- Write any pseudocode following the format suggested in the examples provided in this assignment specification, lecture slides and/or the textbook. **Be consistent with the syntax adopted**. 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.

  Submitted work for assessment must be original. Marks will be deducted if the work is found not to be original.

If you have any questions, you are welcome to post them on the LMS discussion board (Ed Discussion). 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**.