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

Assignment 1, Semester 2, 2017

Released: 9 August. Deadline: 31 August at 23:00

Objectives

To improve your understanding of recurrence relations, graph algorithms and search, as well as time complexity. 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 present algorithms clearly, precisely and unambiguously.

Four Challenges

1. Consider the recurrence relation $T(n) = 2T(n/2) + 1$, with $T(1) = 1$.

   (a) Use the Master Theorem to find a correct Θ expression for $T(n)$.

   (b) Use telescoping to give an exact closed-form definition of $T(n)$.

   (c) Suppose somebody presents you with the following argument, to show that $T(n) \notin \Theta(n)$.

   > If we had $T(n) \in \Theta(n)$ then $T(n) \leq k \cdot n$ for some constant k. But, substituting into the recurrence relation, we get $2 \cdot k \cdot n/2 + 1 \leq k \cdot n$. And that in turn is clearly false for all k. Hence we must have $T(n) \notin \Theta(n)$.

   Do you agree with this argument? Why or why not?

2. DNA nucleobases are `A` (adenine), `C` (cytosine), `G` (guanine) and `T` (thymine). Consider the problem of counting, in a DNA string s, the number of sequences (substrings) that start with an `A` and end with a `C`. For example, there are four such sequences in `TACAAGCGA`.

   (a) Using pseudo-code, give a brute-force algorithm to solve this problem.

   (b) Design a method that solves the problem with a single left-to-right scan of the string s and present it using pseudo-code. Briefly explain why the complexity is linear in $|s|$.

3. Given two nodes v and u in a connected, undirected graph $G = \langle V, E \rangle$, their *distance* is the length of the shortest path between v and u (here each edge is of length 1). The *remoteness* of a node is its distance to the node furthest away, that is, $rem(v) = \max\{dist(v, u) \mid u \in V\}$. A *hub* in G is a node which has minimal remoteness, compared to all other nodes. That is, v is a hub iff $rem(v) = \min\{rem(u) \mid u \in V\}$. For example, b is a hub in the graph on the left (below), whereas b, d and e all are hubs in the graph on the right. Note that a connected graph always has at least one hub.

(0,-0.2)(6,1.3) nodesep=1pt,linecolor=black,arrows=-

(0,1)A(0,1)a (1,1)B(1,1)b (2,1)C(2,1)c (0,0)D(0,0)d (1,0)E(1,0)e (2,0)F(2,0)f

(4,1)A1(4,1)a (5,1)B1(5,1)b (6,1)C1(6,1)c (4,0)D1(4,0)d (5,0)E1(5,0)e (6,0)F1(6,0)f

AB AD BC BD BE BF DE

A1B1 A1D1 B1C1 B1D1 B1E1 D1E1 E1F1

(a) Using pseudo-code, design a function that, given a connected undirected graph $\langle V, E\rangle$ and a node $v \in V$, determines all distances from v, that is, for each $u \in V$, gives $dist(v, u)$. Assume that nodes are labelled 1 to n, so that your function can return an array *dist* with $dist[i]$ giving i's distance to v. The algorithm should run in time that is linear in the size of the graph.

(b) Design an algorithm that, given a connected undirected graph $\langle V, E\rangle$, identifies a hub (any hub) in the graph. More precisely, use pseudo-code to define a function HUB that takes $\langle V, E\rangle$ as input and returns a node which is a hub. Aim for an $O(n^2)$ algorithm, where n is the size of the graph.

4. Let $\langle V, E\rangle$ be a directed graph. A node $v \in V$ is an *attractor* iff v is a sink, and moreover, for every node $u \in V$ with $u \neq v$, $(u, v) \in E$. Note that a graph can have at most one attractor. We want an algorithm that will identify an attractor in an input graph $\langle V, E\rangle$ if there is one. Assume that a graph is represented as an adjacency matrix A, with the nodes in V labelled 1 to n. The function ATTRACTOR$(A[.,.], n)$ should return i if node i is an attractor, and 0 if the graph has no attractor.

(a) Give an $O(n^2)$ algorithm for the problem, using pseudo-code.

(b) (Harder.) Give an $O(n)$ algorithm for the problem, using pseudo-code. Maximum marks will (only) be given for a solution that is both readable, intelligible, carefully explained and analysed, and, importantly, runs in linear time.

Submission and evaluation

You will probably find some of the problems harder than others. All should be solved and submitted by students individually. The deadline is 31 August at 23:00. Late submission will be possible, but a late submission penalty will apply: a flagfall of 2 marks, and then 1 mark per 12 hours late.

Submit a PDF document via the LMS. This document should be no more than 1 MB in size. For those who want to use the assignment to practise with LaTeX, I will leave the LaTeX source for this document in the content area where you find the PDF version. If you produce an MS Word document, it must be exported and submitted as PDF, and satisfy the space limit of 1 MB. We also accept *neat* hand-written submissions, but these must be scanned and provided as PDF, and the scanner resolution must be set low enough to meet the 1 MB size limit.

Your solution will count for 10 of the 30 marks allocated outside the written exam. Questions 1 and 4 are worth 2 marks each. Questions 2 and 3 are worth 3 marks each, evenly divided over the sub-problems. We expect your work to be neat—parts of your submission that are difficult to read or decipher will be deemed incorrect. 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.

Make sure that you have enough time as the deadline approaches, to present your solutions carefully. Careful presentation is often more time consuming than solving the problems. Explain your algorithms and write neat pseudo-code. Cormen *et al.* (available as an e-book in the library) provide some guidelines for pseudo-code (pages 20–22).

Time you put in early will usually turn out to be more productive that a last-minute, frantic effort. If you get stuck, email Yuan or Harald a precise description of the problem, bring it up at a lecture, or use the LMS discussion board. The COMP90038 LMS discussion forum is both useful and appropriate for this; soliciting help from sources other than the above will be considered cheating and will lead to disciplinary action. For more detail about the School's expectations to academic integrity, click the "Academic Integrity" menu item on the subject's LMS site.

Harald Søndergaard
9 August 2017