CMPSCI 311: Introduction to Algorithms **Spring 2018**

Homework 1

Released 1/24/2018 Due 11:59pm 2/7/2018 in Gradescope

Instructions. You may work in groups, but you must individually write your solutions yourself. List your collaborators on your submission.

If you are asked to design an algorithm as part of a homework problem, please provide: (a) the pseudocode for the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

There are four questions, each worth 25 points total.

Submission instructions. This assignment is by 11:59pm on 2/7/2018 in Gradescope. Please submit a pdf file. You may submit a scanned handwritten document, but a typed submission is preferred.

1. **Stable Marriages.**

   (a) **Stable Marriages: K&T Ch 1, Ex 5.** Consider a version of the stable matching problem where there are n students and n colleges as before. Assume each student ranks the colleges (and vice versa), but now we allow ties in the ranking. In other words, we could have a school that is indifferent two students s_1 and s_2, but prefers either of them over some other student s_3 (and vice versa). We say a student s *prefers* college c_1 to c_2 if c_1 is ranked higher on the s's preference list and c_1 and c_2 are not tied.

      i. **Strong Instability.** A strong instability in a matching is a student-college pair, each of which prefer each other to their current pairing. In other words, neither is indifferent about the switch. Does there always exist a matching with no strong instability? Either give an example instance for which all matchings have a strong instability (and prove it), or give and analyze an algorithm that is guaranteed to find a matching with no strong instabilities.

      ii. **Weak Instability.** A weak instability in a matching is a student-college pair where one party prefers the other, and the other may be indifferent. Formally, a student s and a college c with pairs c' and s' form a weak instability if either

      - s prefers c to c' and c either prefers s to s' or is indifferent between s and s'.
      - c prefers s to s' and s either prefers c to c' or is indifferent between c and c'.

      Does there always exist a perfect matching with no weak instability? Either give an instance with a weak instability or an algorithm that is guaranteed to find one.

2. **Big-Oh and Asymptotics.**

   (a) **Big-Oh.** For each function $f(n)$ below, find the smallest *integer* constant c such that $f(n) = O(n^c)$, or indicate that no such constant exists. All logarithms are with base 2.

      i. $f(n) = \frac{1}{2}n^2$.
      ii. $f(n) = n(\log n)^3$
      iii. $f(n) = \sum_{i=0}^{\lceil \log n \rceil} \frac{n}{2^i}$.
      iv. $f(n) = \sum_{i=1}^{n} i^3$.
      v. $f(n) = 2^{(\log n)^2}$

   (b) **Asymptotics. K&T Ch 2, Ex 6.** Given an array A consistent of n integers, you'd like to output a two-dimensional $n \times n$ array B in which $B[i, j] = A[i] + A[i + 1] + \ldots + A[j]$ for each $i < j$. For $i \geq j$ the value of $B[i, j]$ can be left as is.

For $i = 1, 2, \ldots, n$
  For $j = i + 1, \ldots, n$
    Add up $A[i] + A[i + 1] + \ldots + A[j]$.
    Store in $B[i, j]$.

i. What is the running time of this algorithm as a function of n? Specify a function f such that the running time of the algorithm is $\Theta(f(n))$.

ii. Design and analyze a faster algorithm for this problem. You should give an algorithm with running $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.

3. **Disrupting Communication Networks.** Suppose we have an undirected connected graph $G = (V, E)$ representing a communication network (e.g., the vertices denote routers and the edges denote links). We'd like to disrupt communication in this network by removing a node so that the resulting in a highly fragmented network. Precisely, we'd like to identify a node $v \in V$ whose removal splits the graph into as many different connected components as possible. In this question, we'll design a linear time $O(|V| + |E|)$ algorithm for this problem.

   Let $f(v)$ denote the number of connected components in the graph once the vertex v is removed.

   (a) How can you use a DFS tree starting at some $r \in V$ to calculate $f(r)$ efficiently?

   (b) Suppose $v \in V$ is a node in the resulting DFS tree, $v \neq r$, and that no descendant of v has any non-tree edge to any ancestor of v. How could you calculate $f(v)$ from the DFS tree in an efficient way?

   (c) For each node in the DFS tree, let $d(v)$ denote the depth of v in the DFS tree (so r has $d(r) = 0$, r's children have depth 1, etc.) How can we compute $d(v)$ for all vertices $v \in V$ in linear time. Your algorithm should output an array `d` with `d[v]` $= d(v)$.

   (d) If w is a node in the DFS tree, let $\text{up}(w)$ denote the depth of the shallowest node y such that $\{x, y\} \in E$ is a graph edge and either $x = w$ or x is a descended of w. Let $\text{up}(w) = \infty$ if no such edge exists.

   If v is a non-root node in the tree, which children $w_1, \ldots, w_k$, how can we compute $f(v)$ as a function of $k, \text{up}(w_1), \ldots, \text{up}(w_k)$ and $d(v)$.

   (e) Describe how to compute $\text{up}(v)$ for each vertex $v \in V$ in linear time.

   (f) Describe how to compute $f(v)$ for each vertex $v \in V$ in linear time.

4. **Graphs.**

   (a) **K&T Ch3.Ex12.** You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who've lived there over the past two hundred years. From these interviews, they've learned about a set of n people (all of them now deceased), whom we'll denote $P_1, P_2, \ldots, P_n$. They've also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

   - For some i and j, person P_i died before person P_j was born; or
   - For some i and j, the life spans of P_i and P_j overlapped at least partially.

   Naturally, they're not sure that all these facts are correct; memories are not so good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.

   Give an efficient algorithm to do this: either it should produce proposed dates of birth and death for each of the n people so that all the facts hold true, or it should report (correctly) that no such dates can exist — that is, the facts collected by the ethnographers are not internally consistent.

5. **(0 points).** How long did it take you to complete this assignment?