Colorado CSCI 5454: Algorithms
Homework 1

Instructor: Bo Waggoner
Due (updated): by 11:59pm, September 5, 2019
Turn in electronically via Gradescope. LaTeX(LaTeX) is preferred, but not required.

Problem 1 (6 points)

Let $f(n) = \sum_{j=1}^{n} j^{10}$.

Part a (2 points) Show that $f(n) \in O(n^{11})$. *Hint: for each term in the sum,* $j \leq n$.

Solution. For each entry of the sum, $j \leq n$. So $f(n) \leq \sum_{j=1}^{n} n^{10} = (n)(n^{10}) = n^{11}$.

Part b (2 points) Show that $f(n) \in \Omega(n^{11})$. *Hint: only consider the second half of the terms.* We conclude that $f(n) \in \Theta(n^{11})$.

Solution. We have $f(n) \geq \sum_{j=\lceil n/2 \rceil}^{n} j^{10}$. In this sum, each term is at least $\left(\frac{n}{2}\right)^{10} = \frac{n^{10}}{2^{10}}$. So $f(n) \geq \sum_{j=\lceil n/2 \rceil}^{n} \frac{n^{10}}{2^{10}} \geq \left(\frac{n}{2} - 1\right)\left(\frac{n^{10}}{2^{10}}\right) = \Omega(n^{11})$.
(Shorter, acceptable solution:) Consider only the second half of the terms; there are $\Omega(n/2)$ terms, each of which is $\Omega(n^{10})$, so the total is $\Omega(n^{11})$.

Part c (2 points) Argue that $f(n) \in o(n^{12})$.

Solution. Because $f(n) \in O(n^{11})$, we know $f(n) \leq C \cdot n^{11}$ for all large enough n and some constant C. So $\lim_{n\to\infty} \frac{f(n)}{n^{12}} \leq \lim_{n\to\infty} \frac{C \cdot n^{11}}{n^{12}} = \lim_{n\to\infty} \frac{C}{n} = 0$.

Problem 2 (4 points)

In the course notes, it is proven that the Deferred Acceptance algorithm with n doctors and n hospitals (there referred to as men and women) always terminates within $O(n^2)$ rounds, where there is a single proposal in each round. Describe a family of instances where the algorithm takes $\Omega(n^2)$ rounds to terminate.

For any n, you should explain how to construct an input of size n and why the Deferred Acceptance algorithm requires $\Omega(n^2)$ rounds to terminate. You may specify exactly how the algorithm chooses the next proposal in each round in order to make the running time as bad as possible.

Solution. All of the doctors have the same preference ordering over hospitals. The hospitals can have any preference ordering. First, all of the doctors propose to the top hospital in any order (n rounds). It keeps only one of them and rejects the rest. Next, the remaining $n-1$ doctors propose to the second-best hospital in any order ($n-1$ rounds). It keeps only one and rejects the rest. This continues until the one remaining doctor proposes to the final hospital. The number of rounds is $n+(n-1)+\cdots+3+2+1=\sum_{j=1}^{n} j=\Omega(n^2)$. (Students may still receive full credit if they state the $\Omega(n^2)$ result without proof, but it is preferred to also argue this fact, which is proved in the same way as problem 1b above.)

Problem 3 (6 points)

We are given an undirected, unweighted graph $G=(V,E)$ and a particular edge $e=(u,v)$. We must give an algorithm to determine whether G has any cycle that contains e. For this problem, we assume all paths and cycles are *simple*, i.e. contain no repeated edges.

Part a (2 points) Prove that G has a cycle containing e if and only if, when we remove e from the graph, there exists a path from u to v. *Hint: First write down the precise definitions of path and cycle.*

Solution. Suppose we remove e from the graph and there exists a path from u to v. Call it $(u,w_1),\ldots,(w_k,v)$. Then there is a cycle in G that first takes this path, then follows the edge $e=(u,v)$. On the other hand, suppose there is a cycle containing e. The cycle must be of the form $(u,v),(v,w_1),(w_1,w_2),\cdots,(w_k,u)$, with no repeated edges. If we remove the edge $e=(u,v)$, we are left with all the remaining edges, which form a path from v to u.

Part b (4 points) Design a linear-time algorithm (i.e. running time $O(|V|+|E|)$) to determine whether G has any cycle containing a given edge e. You should argue both correctness and running time.

Solution. We first remove e from the graph. It takes linear time to make a copy of the input, excluding e. Then, starting at u, we execute a depth-first search. Initially all nodes are marked as univisited, and we mark them as visited once we reach them. The depth-first search takes linear time in the size of the input. If v is marked as visited, then there is a path from u to v not using e, so we return "yes". If v is not visited, there is no such path, so we return "no". By the previous subproblem, we know there is a cycle containing e in the original graph if and only if there is a path from u to v in the modified graph, proving correctness.