CS256-01: Algorithm Design and Analysis

Assignment 1 (due 9/24/2020)

Instructor: Sam McCauley

You must use this LaTeX template for solutions: Overleaf link to template.

LaTeX typesetting is worth 5 points on this assignment.

Time Complexity

Problem 1 (10 points)**.** Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $f(n)$ and $g(n)$ are such that $f(n)$ is $O(g(n))$, then $g(n)$ must immediately follow $f(n)$ in your list seperated by a comma. If two functions have asymptotically the same order, that is, $f(n) = \Theta(g(n))$, then indicate that explicitly and place them at the same index in your list. For full credit, you must give a brief justification for the ordering of *each adjacent pair*.

To be clear: your solution should be an ordering of (a), . . . (k). You should also label every adjacent pair in the ordering; first briefly explaining why your solution is accurate, and also stating if the adjacent pair has asymptotically the same order. Thus, overall, your solution should consist of 1 ordering, as well as 9 explanations.

Note. All logs are base 2. You may find that sometimes instead of comparing $f(n)$ and $g(n)$ directly, it is easier to compare $\log(f(n))$ and $\log(g(n)))$. As $\log(x)$ is a strictly increasing function for $x > 0$, $\log(f(n)) < \log(g(n))$ implies $f(n) < g(n)$.

(a) $\log n$

(b) $\log \sqrt{n}$

(c) $\sqrt{\log n}$

(d) $\log(\sqrt{3^n})$

(e) $\sqrt{\log(3^n)}$

(g) $4^{\log n}$

(h) 5^n

(i) $n(\log n)^3$

(j) $n^{\frac{2}{\log n}}$

(k) 2^{n^2}

Solution.

□

Stable Matchings

We saw in class that the Gale-Shapley algorithm runs in $O(n^2)$ time in the worst case. Let's analyze the algorithm further. First, we will show that there exist inputs on which Gale-Shapley takes $\Omega(n^2)$ time; this means that the worst-case running time of the Gale-Shapley algorithm is $\Theta(n^2)$. Then, we will look at the *best-case* running time, and show that there are some lucky instances where only $O(n)$ time is required.

Problem 2 (10 points)**.** For any n, consider the following input: For all $1 \leq i \leq n$, the preferences for h_i are $s_1, s_2, \ldots, s_n$. For all $1 \leq i \leq n$, the preferences for s_i are $h_1, h_2, \ldots, h_n$.

Show that for any n, the Gale-Shapley Stable Matching algorithm runs in $\Omega(n^2)$ time on this input.

Technical clarification: For this question, we will assume that the list of free hospitals is implemented using a queue, exactly as we saw in class, and assume that the queue begins with all hospitals in order $h_1, \ldots, h_n$ (so h_1 will be the first hospital removed from the queue).

Solution.

□

Problem 3 (10 points)**.** Give a set of inputs[1] on which the Gale-Shapley algorithm runs in $O(n)$ time. Briefly explain why Gale-Shapley only requires $O(n)$ time on this input.

[1] A "set of inputs" means one input for each n, as in Problem 2.

Graphs and Traversals

Problem 4 (10 points)**.** (KT 3.9) There is a natural intuition that two nodes that are far apart in a communication network, i.e. separated by many hops, have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here is one that involves the susceptibility of paths to the deletion of nodes.

Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$.

(a) Prove that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. (In other words, show that the graph obtained from G by deleting v contains no path from s to t.)

(b) Give an algorithm with running time $O(m + n)$ to find such a node v.

Solution.

□

Problem 5 (10 points)**.** An Euler tour[2] of a graph G is a circuit (that is, a path that begins and ends at the same vertex), through G that traverses every edge of G exactly once.

(a) Prove that a connected undirected graph G has an Euler tour if and only if every vertex has even degree.
(b) Design and analyze a linear time algorithm to compute an Euler tour of a given graph, or report that no such tour exists.

Solution.

□

[2]Named after Leonhard Euler (1707-1783) who solved part (a) in 1735.

Problem 6 (10 points)**.** The diameter of a graph G is the "longest shortest path", that is, $diam(G) = \max\{dist(u, v) : u, v \in V\}$, where $d(u, v)$ is the length of the shortest path from u to v in G.

Until recently, there was no known method for computing the diameter of a graph that didn't first compute the shortest path between all pairs of nodes. When graphs are dense, all-pairs shortest paths is fairly expensive, so some people have explored algorithms that can more quickly *estimate* the diameter of the graph.

Develop a linear-time algorithm[3] that, given a graph G, returns a diameter estimate that is always within a factor of 1/2 of the true diameter. That is, if the true diameter is d, show that your algorithm returns a value k where $d/2 \leq k \leq d$.

Hint: Let's say I know the distance from x to y and the distance from y to z. What can I say about the distance from x to z?

Fun fact: This approximation factor is not far from optimal for a linear-time algorithm: there is a conditional lower bound showing that approximating the diameter to a factor better than 3/2 requires $\Omega(n^2)$ time. See: Liam Roditty and Virginia Vassilevska Williams. "Fast approximation algorithms for the diameter and radius of sparse graphs." STOC 2013.

Solution.

$\square$

[3]Remember: Linear time for a graph $G = (V, E)$ means time $O(n + m)$, where $n = |V|$ and $m = |E|$.

Bonus Feedback Question

Question is optional with bonus points for answering. Feel free to add a descriptive answer.

Problem 7 (2 point)**.** This problem set was:

(a) Just right amount of challenging, hits a good balance!
(b) Too challenging, and not in a good way.
(c) On the easy side for now
(d) Other (please specify)

Acknowledgments

Cite your sources and collaborators here. (Make sure this section starts on a new page and is the last page of the submission)