CS256: Algorithm Design and Analysis **Spring 2021**

Assignment 1 (due 03/03/2021)

Instructor: Shikha Singh *Solution template:* Overleaf, .tex file

Problem set advice. Make sure to review the problem set advice outlined on this handout before attempting the questions. Also, when asked to design and analyze an algorithm, you must prove that the algorithm is correct and analyze its running time.

Time Complexity

Problem 1 (15 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 appear after $f(n)$ in your ordering. Separate functions in your ordering with 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 that same location in your ordering. For full credit, you must give a brief justification for the ordering of *each adjacent pair*.

For example, suppose we are sorting the functions $\log_{100} n$, n^2, $n \log n$ and $\log_2 n$ in ascending order of growth, then the final ordering must be $\boxed{\log_{100} n = \Theta(\log_2 n),\ n \log n,\ n^2}$.

Thus, overall, your solution should consist of an ordering of all ten functions (where some functions may have the same asymptotic order), as well as 9 explanations.

Note. All log's 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.

Overall ordering:

□

Justifications: □

Stable Matchings

Problem 2 (10 points)**.** (a) Decide whether you think the following statement is TRUE or FALSE. If the statement is true, give a proof. If it is false, give a counterexample.

Consider an instance of the Stable Matching Problem in which there exists a hospital h and a student s such that h is ranked first on the preference list of s and s is ranked first on the preference list of h. Then, h and s must be matched to each other in every stable matching for this instance.

Solution.

□

(b) We saw in class that the Gale-Shapley algorithm runs in $O(n^2)$ time in the worst case. Now, 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 $\Theta(n^2)$.

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. 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.

□

Graphs and Traversals

Problem 3 (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.)

Solution.

□

(b) Give an algorithm with running time $O(m + n)$ to find such a node v. Describe your algorithm in prose and explain why it works correctly (use part **??**). Make sure your algorithmic description is clear and concise.

Solution.

□

Problem 4 (5 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[1] 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: An approximation factor of 2 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.[2]

Solution.

□

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

[2] See Liam Roditty and Virginia Vassilevska Williams. "Fast approximation algorithms for the diameter and radius of sparse graphs." STOC 2013.

Problem 5 (10 points)**.** An Euler circuit [3] 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 if a connected undirected graph G has an Euler circuit then every vertex in the graph must have an even degree.

Solution.

□

(b) In this part, we show that the reverse also holds, that is, if every vertex in a graph G has even degree, then the graph must have an Euler tour. We will give a constructive proof of this statement by designing an algorithm that finds the Euler tour.

Design and analyze a linear-time algorithm to compute an Euler circuit of a given graph G if every vertex in G has even degree, otherwise report that no such circuit exists. You must prove that your algorithm is correct.

Solution. □

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

Bonus Feedback Question

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

Problem 6 (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)

Acknowledgement

This is where you cite your collaborators and sources.