SOLUTIONS TO CSE 331 SAMPLE MID-TERM-I

Please do not read anything into the kind of problems in the sample mid-term. Overall, the mid-term will be harder than this sample mid-term (but still easier than the homeworks). The main purpose of this sample mid-term was to give you an idea of the format of questions. Also you can get an idea of how much detail is expected from your answers in the exam from the solutions below.

1. ($8 \times 5 = 40$ points) Each of the questions below have two parts. For the first part, you need to give a justification for the statement and is worth 2 points. For the second part, answer True or False and briefly JUSTIFY your answer. A correct answer with no or totally incorrect justification will get you 1 out of the total 4 points. **An incorrect answer *irrespective* of the justification will get you** 0 **out of** 4 **points**. (Recall that a statement is true only if it is logically true in all cases while it is is false if it is not true in some case).

   (a) Consider an arbitrary instance of the stable marriage problem with n men and n women.

   (**Part 1**) Argue why the following statement is **TRUE**.

   There are $n! = n \times (n-1) \times \ldots \times 1$ many possible *perfect* matchings.

   Solution. See Problem 1 in HW0.

   (**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

   There are at most $n!$ *stable* matchings for the instance.

   True. Since every stable matching is a perfect matching (but not necessarily the other way around), this part follows from part 1.

   (b) Let $f(n) = \log\log n$ and $g(n) = 10^{10^{10^{10^{10^{10}}}}}$.

   (**Part 1**) Argue why the following statement is **TRUE**.

   $f(n)$ is $\Omega(g(n))$.

   Solution. $f(n)$ grows with n while $g(n)$ does not increase with n at all.

   (**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

   $f(n)$ is $O(g(n))$.

   False. $g(n)$ is a constant (albeit a big one) and does not increase with n. However, $f(n)$ does increase with n (though slowly). Thus, for some large enough n, $g(n) \le f(n)$ and thus, $g(n)$ is $O(f(n))$ (and not the other way round).

   (c) Let $f(n) = n^n$ and $g(n) = 2^{400n}$.

   (**Part 1**) Argue why the following statement is **TRUE**.

$$f(n) = 2^{n \cdot \log_2 n}.$$

   Solution. This follows since $n = 2^{\log_2 n}$, which implies that $n^n = \left(2^{\log_2 n}\right)^n = 2^{n \cdot \log_2 n}$.

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

$f(n)$ is $\Omega(g(n))$.

True. From part 1, $f(n) = 2^{n\log_2 n}$ and since $n\log_2 n \ge 400n$ for $n \ge 2^{400}$, $f(n) \ge g(n)$ for every $n \ge 2^{400}$, which by definition implies that $f(n)$ is $\Omega(g(n))$.

(d) Let $a_1, \ldots, a_n$ be n integers.

(**Part 1**) Argue why the following statement is **TRUE**.

Let $a_i \in \{0,1\}$ for each $i \in [n]$. Then the n numbers can be sorted in $O(n)$ time.

Solution. See Q1 on HW 0.

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

Let $-\sqrt{n} \le a_i \le \sqrt{n}$ for every $i \in [n]$. Then the n numbers can be sorted in $O(n)$ time.

True. Consider the following algorithm (which is a variant of the solution to Q1 on HW 0). Do a scan through $a_1, \ldots, a_n$ and count the number of $i \in [n]$ such that $a_i = j$ for every $-\sqrt{n} \le j \le \sqrt{n}$. Let this number be n_j. Then finally, output n_j j's in increasing order of $j = -\sqrt{n}, \ldots, \sqrt{n}$. (This part is not needed to give full credit but here is an additional observation for your benefit: Since this needs to maintain an array of pointers of size $O(\sqrt{n}) = O(n)$, this is overall $O(n)$ time.)

(e) Consider the BFS algorithm with its input graph G in adjacency list format.

(**Part 1**) Argue why the following statement is **TRUE**.

The input size for BFS is $\Theta(n+m)$.

Solution. This is because the adjacency list representation takes $\Theta(m+n)$ space (as we saw in class when we compared the adjacency matrix and adjacency list representations).

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

BFS is a linear time algorithm. (Recall that an algorithm is a linear time algorithm if it runs in time $O(N)$ on inputs of size N.)

True. Recall that we have shown that BFS runs in time $O(m+n)$ when the graph is represented in the adjacency list format. Further, from part 1, $N = \Theta(n+m)$, which implies the run time is $O(N)$.

(f) For any graph, recall that running the BFS algorithm implicitly computes a BFS tree. (Note: BFS tree is *not* rooted.)

(**Part 1**) Argue why the following statement is **TRUE**.

A BFS tree can be computed (explicitly) in $O(m+n)$ time.

Solution. See the BFS pseudocode in the textbook for the algorithm.

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

Every graph has a a unique BFS tree for it.

False. Consider the cycle on four vertices: v_1, v_2, v_3, v_4 such that (v_i, v_{i+1}) for $1 \le i \le 3$ and (v_4, v_1) are the edges. Consider a BFS run that starts from v_1. Note that v_3 can be discovered from either v_2 or v_4 and each choice leads to a different BFS tree.

(g) Recall that a directed graph is strongly connected if and only if every pair of vertices have directed paths from one to the other.

(**Part 1**) Argue why the following statement is **TRUE**.

A directed graph has at most n^2 edges in it.

Solution. Since there are n^2 pairs of vertices, any directed graph has at most n^2 edges.

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

Any directed graph on n vertices with at least $n-1$ edges is strongly connected.

False. Consider the following DAG on three vertices A, B, C with directed edges $(A, B), (B, C), (A, C)$. In this graph C is not connected to A but it has $n = 3$ edges.

(h) Recall that any graph can be represented in adjacency matrix format.

(**Part 1**) Argue why the following statement is **TRUE**.

Adjacency matrix takes $\Theta(n^2)$ space.

Solution. This was stated in the lecture were we compared adjacency matrix and adjacency list representation of graphs.

(**Part 2**) Is the following statement true or false? Also remember to briefly JUSTIFY your answer.

There is an $O(n^2)$ time algorithm that for any graph on n vertices given in its adjacency matrix, converts it into its adjacency list representations.

True. In short here is the algorithm: go through the matrix row by row and for the vertex u corresponding to the current row, add a list of vertices w such that the entry for (u, w) is a 1. Each row takes $O(n)$ time and there are n rows, which makes for a total running time of $O(n^2)$.