NAME: _____

CSE 421
Introduction to Algorithms
Sample Midterm Exam Winter 2023

Anup Rao 1 Winter 2023

DIRECTIONS:

- Answer the problems on the exam paper.
- Justify all answers with proofs, unless the facts you need have been proved in class or in the book.
- If you need extra space use the back of a page
- You have 50 minutes to complete the exam.
- Please do not turn the exam over until you are instructed to do so.
- Good Luck!

1	/45
2	/20
3	/25
Total	/90
Extra	/10

1. (45 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY you answer.

   (a) $2^{(\log n)^2} = O(n^{1.5})$.

   Solution: False. $2^{(\log n)^2} = n^{\log n}$, which is not $O(n^{1.5})$.

   (b) Suppose we are given a connected undirected graph with distinct edge weights. We greedily delete the heaviest edge that does not disconnect the graph, until we cannot delete any more edges. Then the result is guaranteed to be a minimum spanning tree of the original graph.

   Solution: True. Each edge that is deleted is the heaviest edge of a cycle, each edge that is kept is the lightest edge crossing a cut.

   (c) If the running time of an algorithm satisfies the recurrence $T(n) = 3T(n/6) + cn$, for some positive constant c, and $T(1) = 1$, then $T(n) = O(n)$.

   Solution: True. Since $3 < 6^1$, the master recurrence theorem says the solution is $O(n)$.

   (d) You are given $2^{\sqrt{\ell}}$ sets and promised that some k of them cover the set $\{1, 2, \ldots, \ell\}$. Then the set cover algorithm from class would find $O(k \log \ell)$ of the sets whose union is $\{1, 2, \ldots, \ell\}$.

   Solution: True. The total number of sets is irrelevant. The algorithm will find a cover with $O(k \log \ell)$ sets that cover all inputs.

(e) There is a polynomial time algorithm for computing the a tour in the Traveling Salesperson problem that is within a factor of 2 of optimal.

Solution: True. We used the MST algorithm to show this.

(f) You are given a directed graph whose edges have non-negative weights. There is an algorithm for finding the length of the shortest path between every pair of vertices in time $O(n^5)$.

Solution: True. You can use Disjkstra's algorithm starting from every vertex to achieve this.

(g) In class, we have seen a polynomial time algorithm for computing the optimal vertex cover of a graph.

Solution: False. Anup says he doesn't know of such an algorithm.

(h) The Fast-Fourier transform algorithm leads to an algorithm for multiplying numbers in $O(n)$ time.

Solution: False. The running time is $O(n \log n \log \log n)$, as stated in lectures.

(i) There is an algorithm for multiplying two $n \times n$ matrices in time $O(n^{2.999})$.
Solution: True. Strassen's algorithm does this.

2. (20 points) You are given a directed graph that does not have any cycles. Give a polynomial time algorithm that outputs the vertices in an order satisfying that if (a, b) is an edge, then a is output before b. HINT: Find a greedy algorithm for this problem.

Solution: We give a greedy algorithm.

First we claim that a directed acyclic graph must have a vertex with no in-edges.

To see this, suppose for the sake of contradiction that there is a directed graph on n vertices, with no cycles and yet every vertex has an outgoing edge. Then let $v_0, v_1, \ldots, v_n$ be a sequence in the graph such that (v_{i+1}, v_i) is always an edge. Namely, we walk backwards along edges of the graph for n steps. Since every vertex has an incoming edge, we can find such a long sequence by repeatedly picking an incoming edge of v_i. By the pigeonhole principle, some vertex must repeat in this sequence. But then, just as we saw in class, we obtain a cycle in the directed graph, this is a contradiction.

Then the greedy algorithm is as follows:

(a) Find a vertex a with no incoming edges, which must exist since the graph is acyclic.

(b) Output a, and delete it from the graph, to obtain a new acyclic graph.

(c) Repeat this until no more vertices are left.

The algorithm can be implemented to run in time $O(n(m + n))$. First make a pass on the adjacency list representation to add a list of all the incoming edges. Then in each step we need to scan all the vertices to find a vertex with no incoming edges, which takes time $O(n)$.

To prove that the algorithm is correct, suppose (a, b) is an edge of the graph. Then we claim that a is output before b. Indeed, this is because if b is about to be output, then b cannot have any incoming edges, so a must have been output first.

3. (25 points) A perfect matching of an undirected graph on $2n$ vertices is a set of n edges such that each vertex is part of exactly one edge. Give a polynomial time algorithm that takes a tree on $2n$ vertices as input and finds a perfect matching in the tree, if there is one.

   Solution: We give a greedy algorithm. Actually our algorithm will work whenever the input graph is acyclic.

   (a) Find a vertex of degree 1 in the graph. If the graph has no such vertex, output "No perfect matching".

   (b) Match the vertex to its unique neighbor, and delete these two vertices. If the number of vertices becomes 0, halt, otherwise go to the previous step.

   The running time of the above algorithm is at most $O(n^2)$, since we are iterating at most $n/2$ times.

   To see the proof of correctness, note that at every step of the algorithm, the graph we are working with is acyclic. This is true initially, and as the algorithm executes it only deletes edges, so it cannot introduce a new cycle.

   Now if the graph does not have a perfect matching, then the algorithm must necessarily output that, because it cannot terminate in any other way.

   We prove that the algorithm finds the perfect matching if it exists by induction on n. In the case $n = 1$, this is clear. If $n > 1$, then the graph must contain a vertex of degree at most 1, since it is acyclic, and the degree of each vertex has to be at least 1 because of the perfect matching. So, the algorithm will find a vertex of degree 1 and match it. The remaining graph also has a perfect matching (namely the matching obtained by deleting the matched edge). So, by induction the algorithm will find a perfect matching in the remaining graph.

4. (Extra Credit: 10 points) Is it possible that it is asymptotically faster to square an n-bit number than it is to multiply two n-bit numbers? Justify your answer.

   Solution: No, it is not possible. Because you can compute

   $$(x+y)^2 = x^2 + 2xy + y^2$$

   and so

   $$xy = (1/2) \cdot ((x+y)^2 - x^2 - y^2).$$

   So the product can be computed by computing three squares.