Final Examination

CS 4104 (Spring 2021)

Assigned: May 3, 2021.
Due: submit PDF file containing your solutions on Canvas by 11:59pm on May 10, 2021.

Instructions

1. The Graduate Honor Code applies to this exam. In particular, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, the solutions to homeworks and the midterm examination, and the instructor. Do not use a search engine.

2. Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*

3. For every algorithm you describe, prove its correctness, and state and prove the running time of the algorithm. I am looking for clear descriptions of algorithms and for the most efficient algorithms and analysis that you can come up with. Except for one problem, I am not specifying the desired running time for each algorithm. I will give partial credit to non-optimal algorithms, as long as they are correct.

4. You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince me that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*

5. If you prove that a problem is $\mathcal{NP}$-Complete, remember to state the size of the certificate, how long it takes to check that the certificate is correct, and what the running time of the transformation is. All you need to show is that the transformation can be performed in polynomial time. Your transformation need not be the most efficient possible.

6. Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.

7. You must prepare your solutions digitally, i.e., do not hand-write your solutions. I prefer that you use LaTeX to prepare your solutions. However, I will not penalise you if you use a different system. To use LaTeX, you may find it convenient to download the LaTeX source file for this document from the link on the course web site. At the end of each problem are three commented lines that look like this:

   ```
   % \solution{
   %
   % }
   ```

   You can uncomment these lines and type in your solution within the curly braces.

Good luck!

Problem 1 (32 points) Let us start with some quickies. For each statement below, say whether it is true or false or fill in the blanks. You do not need to provide a rationale for your solution.

1. If A and B are problems such that $B \in \mathcal{P}$ and $A \leq_P B$, then $A \in \mathcal{NP}$.
   Solution: False.
2. In a directed graph, every edge has a positive integer weight that is bounded by the quantity w. The graph has n nodes and m edges. We can compute the shortest path between any two nodes in this graph in $O(w(n+m))$ time.
   Solution: True.
3. In a directed graph, let $d(x, y)$ denote the length of the shortest path between nodes x and y. If $e = (u, v)$ is an edge on a shortest path from s to t, then $d(s, t) = d(s, u) + d(u, t) - l(e)$, where $l(e)$ is the length of the edge e. Here, we are using the standard definition of the length of a path as the sum of the lengths of the edges in it.
   Solution: False.
4. If f is a maximum flow in a flow network and e is an edge such that the $f(e) = c(e)$, then e is an edge that crosses a minimum s-t cut.
   Solution: True.
5. If $\mathcal{P} = \mathcal{NP}$ and problem X is $\mathcal{NP}$-Hard, then there is a polynomial time algorithm to solve X.
   Solution: False.
6. Suppose that, by accident, the jobs input to `Greedy Balance` are sorted in non-decreasing order of running time. Then the makespan of the solution computed by the algorithm is at most 1.5 times the optimal makespan.
   Solution: False
7. Every edge in a flow network has a capacity of one. The value of the maximum flow in this network is the __ number of edge-disjoint paths __ from s to t. (Fill in the blanks with a phrase.)
   Solution: number of edge-disjoint paths
8. The second-most streamed show in the U.S. in 2020 is connected to the __ Stable Matching __ problem.
   Solution: Stable Matching

Problem 2 (8 points) In class, we developed an algorithm that when given an undirected, unweighted graph G with n nodes and an integer k, checks if G has a vertex cover of size k or less in $O(k2^k n)$ time. Note that this time is polynomial in n (but exponential in k). Since we reduced Independent Set to Vertex Cover in class, we also have an algorithm that can check if a graph G has an independent set of size at least l (for some integer $l > 0$) in time polynomial in n. Is this statement true or false? Give a brief explanation for your answer.

Solution: Since Independent Set $\leq_P$ Vertex Cover (they are both $\mathcal{NP}$-Complete), it is true that we also have an algorithm that can check if a graph G has an independent set of size at least l (for some integer $l > 0$) in time polynomial in n.

The size of the certificate would be the independent set of size l and it was shown in class how to perform the transformation in polynomial time.

Problem 3 (15 points) Consider the following problem: given an undirected, unweighted graph $G = (V, E)$ and a positive integer k, compute a spanning tree of G such that every node in the tree has degree $\leq k$, if such a tree exists. Extreme values of k are easy to handle. For example, if $k = 1$, then such a tree exists if and only G contains one edge (otherwise, the tree must have at least one vertex of degree two or more), Similarly, if $k \geq n - 1$, then any spanning tree of G will suffice. For a general (arbitrary) value of k, either develop a polynomial time algorithm for this problem or show that it is $\mathcal{NP}$-Complete.

Solution: We will show that this k-DEGREE SPANNING TREE problem is $\mathcal{NP}$-Complete by showing how we can reduce it to a modified version of the Hamiltonian Cycle problem which we know to be $\mathcal{NP}$-Complete.

First, we know that if a graph has a Hamiltonian Cycle, then it is possible to delete any edge in the cycle to yield a Spanning Tree where each node in the Spanning Tree has a degree of $k \leq 2$. Furthermore, we can simply modify the Hamiltonian Cycle problem into a HAMILTONIAN PATH problem by removing the added constraint from the former problem that the resulting set of edges be a cycle, and changing this to constraint to be that the set of edges are a path.

Additionally, in our reduction to this Hamiltonian Path problem, we can reduce the remaining, general cases for $2 \leq k < n-1$ to the $k = 2$ HAMILTONIAN PATH problem. The modifications from Hamiltonian Cycle to HAMILTONIAN PATH do not make the problem easier (in terms of -Hardness), and the reduction from the presented k-degree Spanning Tree to Hamiltonian Path can be performed in polynomial time, so the result is that the presented problem is $\mathcal{NP}$-Complete.

We perform the reduction as follows:

HAMILTONIAN PATH $\rightarrow$ k-DEGREE SPANNING TREE

For each node in the graph G, we add $k-2$ more vertices and connect the nodes of V to as many of these added nodes as is permissible without violating the k-degree constraint nor entering the extreme domain of $k = 1$. Formally, the modified graph G' would be constructed with $V' = \{v_i, ..., v_{k-2} | v \in V\} \cup V$ and $E' = \{(v_i, v), ..., (v_{k-2}, v) | v \in V\} \cup E$.

From here, we know that if G has a Hamiltonian Path P, then the spanning tree T found on G' has a degree $\leq k$ since, by definition, the path P has degree ≤ 2 and we constructed G' such that no edges added were included on vertices with degree of 2.

k-DEGREE SPANNING TREE $\rightarrow$ HAMILTONIAN PATH

To show that we can revert from the spanning tree T found on G' back to the Hamiltonian Path P on G we simply remove all of the vertices we added to G' (and edges incident upon them) which are identifiable by the fact that they must have a degree of 1. Their deletion from T yields P, the Hamiltonian Path on G.

We can verify the certificate of the k-DEGREE SPANNING TREE problem, T, in polynomial time by checking that it is a spanning tree and that each node has a degree $\leq k$ via a modified BFS or DFS on T. The size of this certificate is proportional to $O((|V| + k - 2) + (|E| + k - 2)) = O(|V| + |E| + k)$ as that terms dictates how many additional nodes and edges are in G'.

Therefore, since the presented k-DEGREE SPANNING TREE problem can be reduced to an instance of the HAMILTONIAN PATH problem which we know to be $\mathcal{NP}$-Complete in polynomial time (and vice versa), the k-DEGREE SPANNING TREE is $\mathcal{NP}$-Complete.

Problem 4 (20 points) Given a directed graph $G = (V, E, l)$, where $l : E \rightarrow \mathbb{R}^+$ specifies a positive length for each edge in G, a source node s, and a sink node t, develop an algorithm to compute an integral flow[1] f such that the value $\nu(f) = 1$ and the quantity $\sum_{e \in E} l(e)f(e)$ is as small as possible. You can assume that there are no capacity constraints, i.e., each edge has unlimited capacity, but the flow along each edge cannot be negative. *Hint:* This is a "trick question" since it disguises another well-known problem in terms of flows.

Solution: The well-known problem being disguised here is simply a shortest path problem which we can solve using an algorithm like Dijkstra's to find the shortest $s-t$ path where the weight or distance of an edge is given by $l(e)$. Once this path is found, we can just send 1 unit of flow across this path to minimize the described quantity.

[1] A flow f is *intgeral* if $f(e)$ is an integer for every edge e in E.

Algorithm 1 `integral_flow`$(G(V \cup \{s,t\}, E, l))$

Input: $G(V \cup \{s,t\}, E, l)$ the directed graph
Output: f the integral flow assignment that minimizes the quantity $\sum_{e \in E} l(e)f(e)$

```
// initialize a flow of 0 to every edge in the graph
```
1 $f(E) \leftarrow 0$
```
   // We assume we have an implementation of Dijkstra's which takes an additional target
   node parameter and returns the shortest s-t path
```
2 $P \leftarrow$ `Dijkstra's`(G, s, t, l)

3 **foreach** $e \in P$ **do**
4 | $f(e) \leftarrow 1$
5 **end**
6 **return** f

Proof of Correctness Claim: the presented algorithm minimizes the quantity$\sum_{e \in E} l(e)f(e)$. We can rely on the correctness of `Dijkstra's` to guarantee that the path P is the shortest possible length on the graph.

From here, it's clear that the quantity is simply the sum of the lengths of edges in the path since the flow value on each edge is just 1:

$$l(e)f(e) = \begin{cases} l(e) & \text{if } f(e) = 1 \\ 0 & \text{otherwise} \end{cases}$$

and we know that $\sum_{e \in P} l(e)$ is the minimum possible from the aforementioned correctness of `Dijkstra's`. Therefore, the sum over all edges is equal to $\sum_{e \in P} l(e)$.

Proof of Runtime Complexity: We know that the running time of `Dijkstra's` can be improved to $O(m \log n)$ by using a priority queue. In the worst case, G is just the path from $s - t$, so the loop in lines 3-4 could take at most $O(m)$, making the overall runtime $O(m + m \log n)$.

Problem 5 (25 points) Consider the proof that Hamiltonian Cycle $\leq_P$ Travelling Salesman that we developed to prove that the second problem is $\mathcal{NP}$-Complete. We started with an undirected, unweighted graph $G = (V, E)$ that was an input to the Hamiltonian Cycle problem. We created an input for Travelling Salesman as follows: for every node $i \in V$, we created a city c_i, and for every pair of cities v_i and v_j, we defined $d(v_i, v_j) = 1$ if (i, j) is an edge in E and $d(v_i, v_j) = 2$ if (i, j) is not an edge in E. Consider the following modification: in the second case, i.e., (i, j) is not an edge in E, we set $d(v_i, v_j) = c$, where c is a constant ≥ 2. Let us denote this input to Travelling Salesman as $I(G, c)$. Answer the following questions.

(a) (4 points) If G contains a Hamiltonian cycle, what is the length of the shortest travelling salesman tour in $I(G, c)$?

Solution: The length of the shortest travelling salesman tour in $I(G, c)$ would be $|V|$. If there is a Hamiltonian Cycle, then it means that there is a tour where, for each edge $e = (v_i, v_j), d(v_i, v_j) = 1$. Assuming that the travelling Salesman must return to the starting city to complete the tour, the length of the tour would be the number of edges traversed which is $|V|$ if there is a Hamiltonian Cycle.

(b) (4 points) If G does not contain a Hamiltonian cycle, what is the length of the shortest travelling salesman tour in $I(G, c)$?

Solution: If G does not contain a Hamiltonian Cycle, the minimum possible length of $I(G, c)$ would be at least $|V| - 1 + c$ since the next best case would be one where there is a Hamiltonian Path on G, but the last edge needed to return to the starting city and complete the tour would have have $d = c$.

(c) (11 points) Suppose there is an α-approximation algorithm for the Travelling Salesman problem that runs in polynomial time, i.e., for every input to this problem, this algorithm computes a travelling salesman tour whose cost is at most α times the cost of the optimal tour. Determine a value of c so that you can use this approximation algorithm to distinguish unambiguously between the two cases: G has a Hamiltonian cycle and G does not have a Hamiltonian cycle. Explain your approach.

Solution: Since we know that the length of the optimal tour is $|V|$ and the optimal approximation cost must be at most α times the cost of the optimal tour, it makes sense to set $c = \alpha|V|$. If G has a Hamiltonian cycle, then the cost the optimal tour remains $|V|$ as we determined in (a). Otherwise, if G does not have Hamiltonian cycle, then the optimal tour will include an edge with the cost of c making the total cost of the tour at least $|V| - 1 + \alpha|V|$ which we could quickly (after it's been calculated, that is) use to differentiate between the two cases.

(d) (6 points) What can you conclude from these arguments?

Solution: We can conclude that, since α-approximation algorithm for the Travelling Salesman runs in polynomial time, and we we also know that Travelling Salesman $\leq_P$ Hamiltonian Cycle.