Name: Ryan Evans

Student ID:

CS180 Spring 2010 - Midterm

Wednesday, April 28, 2010

You will have 110 minutes to take this exam. This exam is closed-book and closed-notes. There are 5 questions for a total of 100 points. **Please write your name and student ID on every page of your solutions. Please use separate pages for each question.**

Question	Points
1	18
2	18
3	20
4	7
5	17
Total	80

Class Average (mean): 50.2

Name: Ryan Evans

Student ID:

1. **[20 points]** Suppose we are given n men and n women along with their preference lists. Recall that a valid partnership is a pair (m, w) consisting of a man and a woman such that there is some stable matching which contains (m, w). Describe a polynomial time algorithm in high-level pseudocode to determine ~~if there is a~~ stable matching which simultaneously pairs each man with his best valid partner and each woman with her best valid partner. Prove that your algorithm is correct and give its running time (you may use the Gale-Shapley algorithm as a subroutine).

+18

The gale-shapley algorithm always favors the person being matched and always gives the least preference valid match to the person chosen for the person being matched.

In order to see if there is a matching which gives both genders their best valid partner, we need to run the algorithm twice. Once with the men being the choosers and once with the women being choosers.

Then we compare the two results and if they match. If they do, both genders best [illegible] will be [illegible]. If they are not equal then only one group can be matched with [illegible] best valid partner.

running time?

Name: Ryan Evans

Student ID:

+18

2. [**20 points**] We are given a set of n jobs, each with a processing time p_i and a weight w_i, where the weight represents the job's priority. Let C_i denote the completion time of job i. For example, if job j is the first job to be completed, then its completion time is $C_j = p_j$. If job j completes right after job i, then j's completion time is $C_j = C_i + p_j$. Design an efficient algorithm in high-level pseudocode to determine an ordering of the jobs so as to minimize the weighted sum of completion times $\sum_{i=1}^{n} w_i C_i$. Prove that your algorithm is correct and state its running time.

The optimal algorithm is to take all the jobs and find out what their ratio of weight to processing time is (w_i/p_i). Then sort the jobs from largest to smallest. This ensures that the weight per unit time is always decreasing. Then, if you run the jobs in this order, you will get the lowest weighted sum

Let us look at inversions to prove that our solution is correct. Take any two adjacent jobs and assume that job a runs before job b, but job b has a higher weight to time ratio (r_i) than a. So, $C_a < C_b$ and $r_a < r_b$. Then, total cost is $(x+p_a) \cdot r_a \cdot p_a + (x+p_a+p_b) \cdot r_b \cdot p_b$

$C_a = (x+p_a$ and $C_b = (x+p_a+p_b$

In normal english, this will always be more than if the inversion was reversed because a higher cost times more time will always be more than a lower cost times the same amount of time

Since this is a normal argument, it proves that all inversions are worse than if they were switched. So the optimal solution must have no inversions, so the optimal solution is the order from highest to lowest of w_i/p_i.

Prove cost after swap ≤ cost before swap

Name: Ryan Evans

Student ID:

3. **[20 points]** In the **Maximum Tri-Partition** problem, we are given an undirected graph $G = (V, E)$ along with an integer k. We are asked to determine whether there exist disjoint sets V_1, V_2, and V_3 whose union is V such that there are at least k edges of E whose endpoints are not in the same set V_i. Prove that this problem is **NP**-Complete.

+20

This problem is NP because to check the solution, we would only have to check each edge and keep a tally of how many edges straddled two sets. If that tally was greater than or equal to k, then we could verify that the solution was correct.

3 Color $\leq_p$ Tri Partition

We will use this problem to solve the three color problem for a map, we will construct a graph where each node stands for an area that needs to be colored and each edge connects two adjacent areas on the map. So each node whose border touches the border of another node is connected by an edge.

Then we pass this graph into tri partition with the value k equal to the total number of edges in the graph. If it succedes, that means we have three disjoint sets where each node is not touching any other node in the set. Thus, we can pick one color for each set and be guaranteed that no two adjacent areas have the same color.

Name: Ryan Evans

Student ID: S

+7

4. **[20 points]** You are given a minimum spanning tree T of an undirected graph $G = (V, E)$ with weights on the edges. Now, you have been told that the weight of a particular edge e in G has all of a sudden been increased, giving us a new graph G'. Describe, in high-level pseudocode, an algorithm to compute a minimum spanning tree T' in G' that runs in time $O(|V| + |E|)$. You may assume that all edge weights in G and G' are distinct.

[illegible]

The only edges we may need to change are those edges larger than e before its size increased.

~~So, for each edge larger than e, we need to see if it can now be~~

So, we need to remove e and all edges larger than e before its size was increased. Then we need to run the minimum spanning tree algorithm again starting at piece e (old size) and continuing as if it had left off there.

not linear.

If $e \notin T$, do nothing.

If $e \in T$, remove e. This creates a cut (segments the graph). Add the lightest edge that rejoins the ~~graph~~ spanning tree.

The two segments of the graph are S, V−S.

Name: Ryan Evans

Student ID: ______

5. (a) [**10 points**] For any problem $B \in \mathbf{NP}$, give an algorithm which solves B in time $O(2^{p(n)})$, where n is the size of the input and $p(n)$ is some polynomial which may depend on B.

+7

(b) [**10 points**] Consider the following decision version of the Minimum Spanning Tree problem, which we will call *D-MST*: Given an undirected graph G with weights on the edges and an integer k, is there a Minimum Spanning Tree of total weight at most k. A friend of yours comes to you and claims they have a proof that *D-MST* reduces to the Traveling Salesman Problem (i.e. *TSP*) in polynomial time. That is, they claim *D-MST* $\leq_p$ *TSP*. Your friend plans on writing up his solution very carefully and sending it to the most prestigious conference in computer science. Assuming your friend has not made any mistakes, do you feel that this result is important enough to get published? Explain your answer in detail.

+10

a. All NP problems are about finding some combination of values to satisfy a certain problem. Thus, to solve any NP problem in $2^{p(n)}$ time, we just have to come up with every possible combination of the inputs and give them to the verifier. Then all possible combinations will be checked and it will take $2^{p(n)}$ time to complete.

where do we get $p(n)$?

b. No, this is not important. We already know that minimum spanning tree can be solved in polynomial time or less without being reduced to anything else. There is already a deterministic algorithm to solve it. The way to solve the decision problem is to solve the minimum spanning tree then see if the total weight is less than K. Since this can already be done faster than TSP, there is no point to solving the problem using TSP.