University of Washington
Department of Computer Science and Engineering
CSE 421, Winter 2019

October 21, 2019

Solution to the Midterm Exam, Wednesday, February 13, 2019

NAME: _____

Instructions:

- Closed book, closed notes, no calculators
- Time limit: 50 minutes
- Answer the problems on the exam paper.
- If you need extra space use the back of a page
- Problems are not of equal difficulty, if you get stuck on a problem, move on.

1	/10
2	/10
3	/30
4	/10
5	/20
Total	/80

Problem 1. Stable Marriage (10 points):

Show that the Gale-Shipley Stable Marriage algorithm can take $\Theta(n^2)$ steps with appropriate choice of preference lists. Give preference lists and an ordering of the proposals that require $\Theta(n^2)$ steps. Explain why your example achieves the bound.

Hint: This can be done with all of the M's having the same preference lists, and all of the W's having the same preference lists.

Solution:

Give the M's the preference list $\{w_1, w_2, \ldots, w_n\}$, and the W's the preference list $\{m_1, m_2, \ldots, m_n\}$. During the i-th round, each unmatched M proposes to the i-th in its list. Exactly one new M is matched each round, so there are $n - i + 1$ in round i. The total number of proposals is

$$\sum_{i=1}^{n}(n - i + 1) = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}.$$

The order of the M's and W's preference lists *don't* matter in this argument - it just depends on on the lists being all the same.

Problem 2. Big Oh (10 points):

Let q, r, and s be positive constants. Prove that $qn^2 + rn + s$ is $O(n^2)$ using the formal definition of $O(\cdot)$.

Big $O(\cdot)$ definition: $f(n)$ is $O(g(n))$ if there exists $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq cg(n)$.

Solution:

Let $c = q + r + s$ and $n_0 = 1$. For $n \geq 1$ we have:

$$qn^2 + rn + s \leq qn^2 + rn^2 + sn^2 = (q + r + s)n^2 = cn^2.$$

Problem 3. True or False (30 points):

Determine if the following statements are true or false. Provide a short justification for each answer.

a) *True or false:* If G is a directed graph on n vertices where every vertex has out degree at least two, then G has a cycle. Justify your answer.

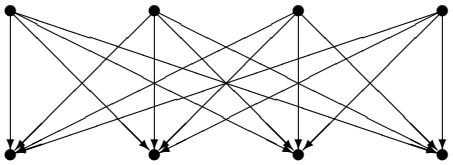
Solution:

True. Choose any vertex, and follow a path until there is a repeated vertex. Since the out degree of every vertex is at least one, there is always an edge to pick to lead to another vertex. A repeated vertex must always be found within n steps, hence, there is a cycle.

b) *True or false:* If G is a directed graph on n vertices with at least $2n$ edges, then G has a cycle. Justify your answer.

Solution:

False. The complete bipartite directed graph (or $K_{n,n}$) for $n \geq 4$. For $n = 4$, the graph has 8 vertices and 16 edges.



c) *True or false:* If G is a directed graph on n vertices, with distinct vertices r and s, where there is a path from r to every vertex in the graph, and there is a path from s to every vertex in the graph, then there is a cycle in the graph. Justify your answer.

Solution:

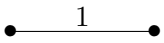
True.

The path from r to s followed by the path from s to r is a cycle.

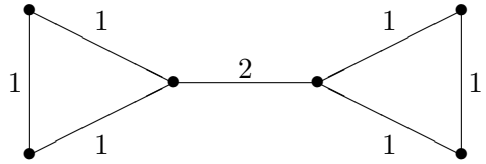
d) *True or false:* If G is an undirected graph with edge weights, and edge e has weight strictly greater than any other edge in the graph, then e cannot be in a minimum spanning tree for G. Justify your answer.

Solution:

False. A trivial counter example is a graph with a single edge:



A non-trivial example is where the most expensive edge is the only connection between two components:



e) *True or false:* If G is an undirected graph with edge weights, and edge e has weight strictly less than any other edge in the graph, then e must be in every minimum spanning tree for G. Justify your answer.

Solution:

False. This follows from the edge inclusion lemma. Suppose $e = (u, v)$ has cost less than any other edge. We use $\{u\}$ in the edge inclusion lemma as (u, v) is the minimum cost edge between $\{u\}$ and $V - \{u\}$.

f) *True or false:* If G is a undirected graph on n vertices with more than $n/2$ connected components, then at least one of the connected components is an isolated vertex. Justify your answer.

Solution:

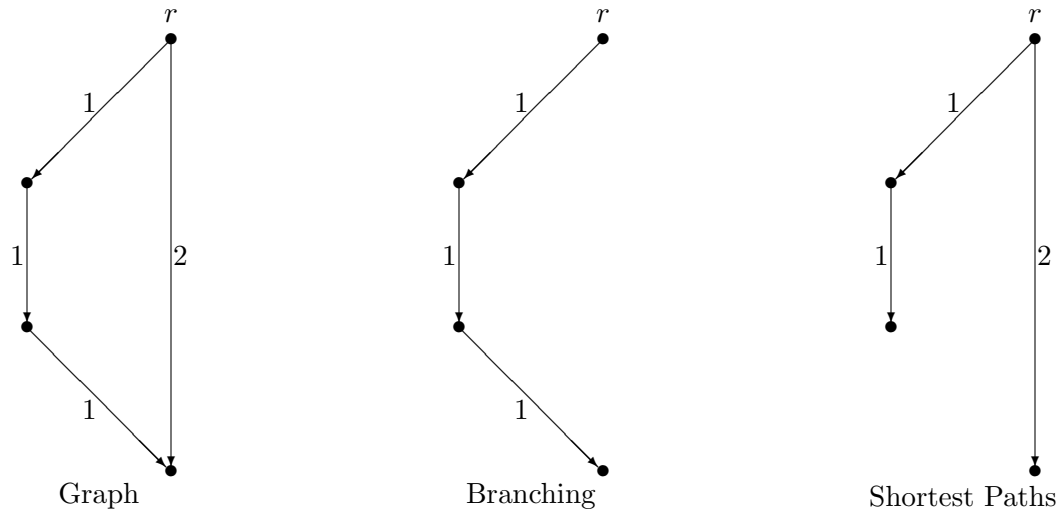
True. Suppose there are no isolated components, so each component has size at least 2. However, if we have more than $\frac{n}{2}$ components, we have more than $2 \cdot \frac{n}{2} = n$ vertices.

Problem 4. Minimum Weight Branching (10 points):

A branching is a rooted subtree in a directed graph where there is a path from the root r to every vertex in the graph. The *minimum branching problem* is: given a directed graph with weights on the edges and a specified vertex r, find a branching of minimum weight rooted at r.

Show that Dijkstra's shortest paths algorithm *does not* solve this problem. Specifically, give a graph where the shortest paths found by Dijkstra's algorithm do not form a minimum weight branching.

Solution:



(Note: This problem should have specified that the all edges have non-negative costs to avoid solutions with a malfunctioning Dijkstra's algorithm.)

Problem 5. One-Two Knapsack Problem (20 points):

The *Knapsack Problem* is: Given a collection of items $I = \{i_1, \ldots, i_n\}$ and an integer K where each item i_j has a weight w_j and a value v_j, find a subset of the items with weight at most K which maximizes the total value of the set. More formally, we want to find a subset $S \subseteq I$ such that $\sum_{i_k \in S} w_k \leq K$ and $\sum_{i_k \in S} v_k$ is as large as possible.

We define the *density* of d_j of item i_j to be $d_j = v_j/w_j$. A natural greedy algorithm for the knapsack problem is to consider the items in order of decreasing density, and place each item into the knapsack if there is still sufficient space for the item.

For this problem, we restrict the weights of the items to be either 1 or 2. For convenience, we assume the capacity K of the knapsack is an even number.

a) Given an example that shows that the greedy algorithm based on sorting items by density does not necessarily give an optimal solution, even if the weights are restricted to 1 and 2.

   Solution:

   Let $K = 2$, and we will have items $\{i_1, i_2, i_3\}$, where i_1 has weight 1 and value 2, i_2 has weight 1 and value 0, and i_3 has weight to and value 3.

   The greedy algorithm finds the solution $\{i_1, i_2\}$ with value 2, while the optimal solution is $\{i_3\}$ with value 3.

b) Describe an efficient algorithm that finds an optimal solution to the knapsack problem when the weights are restricted to 1 and 2.

Solution:

Step 1. Sort the items of size 1 by value and combine adjacent items into new items of weight 2. (If there is a left over item, make it an item of weight 2.)

Step 2. Select the largest $\frac{K}{2}$ items of weight 2 (including both the original items of weight 2, and the combined items from Step 1.

c) Provide a justification that your algorithm is correct.

Solution:

For convenience, we assume the values are distinct. This assumption can be removed by sorting equal value items in a consistent manner, such as using the item number as a secondary key.

The solution S constructed by the algorithm has the following properties:

1. Every item of weight 1 in S has greater value than than every item of weight 1 not in S.
2. Every item of weight 2 in S has greater value than than every item of weight 2 not in S.
3. Every item of weight 2 in S has greater value than than every pair of items of weight 1 not in S.
4. Every pair of items of weight 1 in S has greater value than item of weight 2 not in S.

Let T be a set of items different from S. Suppose $x \in T$ and $x \notin S$.

Case 1: Weight of $x = 1$ and there is a $y \in S$, $y \notin T$ with weight 1. In this case, y can be added to T and x removed to show T is not optimal.

Case 2: Weight of $x = 1$ and there is no $y \in S$, $y \notin T$ with weight 1. In this case, there is another $z \in T$ of weight 1 with $z \notin S$. An item of weight 2 can be added to T and x and z removed to show T is not optimal.

Case 3: Weight of $x = 2$. In this case either an item of weight 2, or two items of weight 1 can be added to T and x removed to show T is not optimal.

This shows that T is not optimal.