ELL781: Re-Major Test

Sumeet Agarwal

January 10, 2022

Maximum marks: 35

Instructions:

- **Please clearly indicate the question number, and part number if applicable, at the start of each response; and correspondingly, indicate the page numbers corresponding to each question when uploading your answer script onto Gradescope.**
- **Please read all questions carefully.**
- **Please ensure that your responses are to-the-point and that you write only what is asked for on the answer script you submit.**
- **While the exam is open-book, all your answers must be written entirely in your own words, without any copying from anywhere.**
- **Please try to be clear and careful with all mathematical notation, so that there is no ambiguity in the expressions/formulae you write down. Try to stick to the kind of notation used in class as far as possible.**

Questions

1. Please write out the below pledge on your answer script, and add your signature underneath to indicate your assent. This will be needed for your responses to be graded.

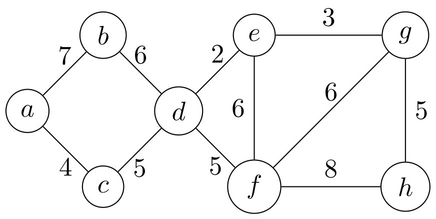
   I pledge to attempt all questions in this test on my own, without seeking assistance from anyone or copying from anywhere.

2. Consider a currency system (let its unit be called a 'dollar') which has notes/coins of the following denominations: 1, 2, 4, 8, 16, 32, 64, 128.

   (a) What combination of denominations would you use (as per the typical greedy strategy) in order to make a payment of 300 dollars? **[1]**

(b) Will the greedy strategy always be optimal for this set of denominations, in terms of using the fewest possible number of notes/coins for a given amount? Prove/justify your answer. (Hint: think about the number of times each denomination can possibly be chosen.) **[3]**

3. Consider the below graph.



Show clearly the *sequence* in which edges will get added to the constructed spanning tree, when you obtain it in each of the following ways for the above graph. Wherever there is a choice, assume vertices/edges will be selected or visited in alphabetical order as per the vertex labelling. Also mention the total cost of each spanning tree.

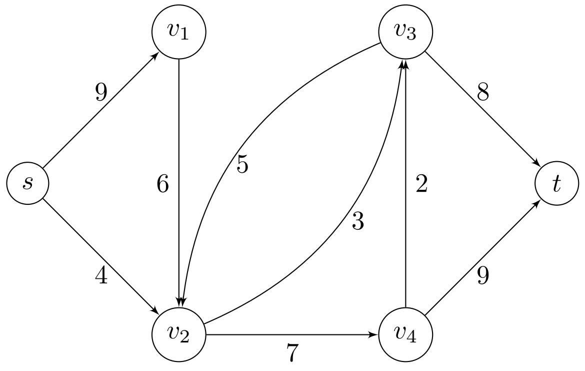
3.1. Using Kruskal's algorithm. **[2.5]**

3.2. Using Prim's algorithm. **[2.5]**

3.3. Using BFS, starting at vertex c. **[2.5]**

3.4. Using DFS, starting at vertex f. **[2.5]**

4. Consider the below flow network.



4.1. Show the path from source to sink that will be found by running BFS here. Clearly depict the process of finding this path, by showing the sequence in which vertices are visited and edges are traversed by BFS. (Hint: at any iteration, the set of edges traversed by BFS form a tree, so its working can be demonstrated by drawing this sequence of trees.) **[2.5]**

4.2. Draw the residual network obtained after using the path from (a) as an augmenting path. **[1.5]**

4.3. Now show the successive sequence of finding augmenting paths via BFS and obtaining the subsequent residual networks, until no more augmenting paths can be found. You need not show the working of BFS in detail for each augmenting path, but should clearly show what the augmenting path is and what the residual network is for each iteration. **[4]**

4.4. In the maximum flow finally obtained, what is the flow across the cut $(\{s, v_2\}, \{v_1, v_3, v_4, t\})$? What is the capacity of this cut? **[1]**

4.5. Is the cut just examined a minimum cut of the given flow network? If not, find a minimum cut. **[1]**

5. Consider the following decision problem:

   Given a weighted, directed graph where the weights represent edge lengths, does there exist any pair of nodes in the graph such that the shortest path between them has length no more than K?

   Which complexity classes from the set {P, NP, NP-complete, NP-hard} does this problem belong to, assuming P≠NP? Mention all that apply, and explain/justify your answer. **[4]**

6. Consider an undirected graph with 9 vertices and 16 edges (as usual, there are no self-loops), such that no vertex has degree greater than 4.

   (a) What is the smallest possible number of vertices in a vertex cover of such a graph? Explain your reasoning clearly, and draw an example graph with the mentioned properties which does indeed have a vertex cover of your determined size. **[4]**

   (b) Show clearly the working of the APPROX-VERTEX-COVER algorithm discussed in class on your example graph as drawn above. What is the size of the cover returned by it? Based just on this example, what can you say about the approximation ratio of this algorithm (*e.g.*, can you put an upper or lower bound on the ratio)? **[3]**