ELL781: Major Test

Sumeet Agarwal

November 21, 2021

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.

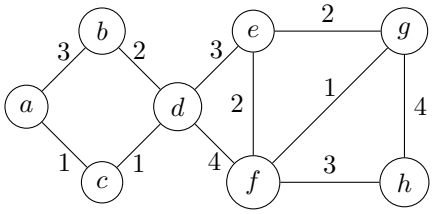
2. Consider an undirected graph with 8 vertices and 27 edges; assume there are no self-loops.

   2.1. What is the minimum number of colours needed for a valid colouring of this graph? Explain/justify your answer. **[3]**

2.2. Recall the greedy graph-colouring heuristic discussed in class. Will this heuristic always return the optimal colouring for the above graph, irrespective of the numbering/ordering of the vertices? Prove/justify either way. **[4]**

3. Prove the correctness of Prim's algorithm for graphs which have a *unique* MST. Make use of the greedy choice property (MST property) proved in class (you need not re-write that proof; just use the property in your correctness proof). Please write out your proof as clearly, formally, and concisely as you can. **[4]**

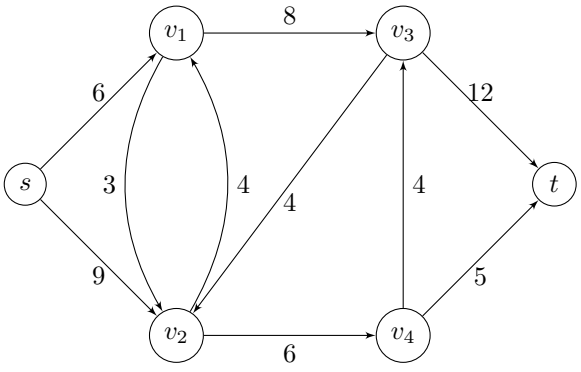
4. Consider the below graph.



4.1. Consider the edge (f, h). Use the MST property to show that this edge belongs to some MST. Write out your argument clearly (again, you do not need to prove the MST property itself, just use it). **[1]**

4.2. Show clearly the *sequence* in which edges will get added to the constructed MST, when you execute Kruskal's algorithm on the above graph. Wherever there is a choice, assume edges will be selected in alphabetical order as per the vertex labelling. **[2.5]**

5. Consider the below flow network.



5.1. 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]**

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

5.3. 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]**

5.4. 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]**

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

6. Consider the following decision problem:

   Given a weighted, directed graph where the weights represent edge capacities, does there exist any way of choosing a source vertex and a sink vertex in that graph such that the flow between them can be at least 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]**

7. Recall the algorithm APPROX-VERTEX-COVER, which maintains a set of uncovered edges, and at each iteration picks one edge from this set at random and adds both its endpoints to the cover set.

   7.1. A *matching* is a set of edges in a graph such that no two edges share a common vertex. A *maximal matching* is a matching to which no further edges can be added (whilst maintaining the matching property). Prove that, for any input graph, the set of edges picked by APPROX-VERTEX-COVER (which we had denoted as A in class) forms a maximal matching of the given graph. **[4]**

   7.2. Give an example of a graph for which the optimal vertex cover contains 4 vertices, but APPROX-VERTEX-COVER can never give the optimal solution, no matter what sequence of edges it may happen to pick. Explain/prove why this is so for your answer. **[2.5]**