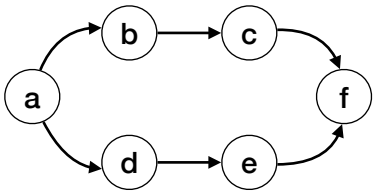
Prof. Salma Elmalaki
Fall 2022

Homework 2

Practicing Chapter 3 and 4

1. *(5 Points)* Chapter 3: Topological Order Consider the directed acyclic graph G in the figure below. How many topological orderings does it have? Explain how you get computed it. List the topological order.



2. *(25 Points)* **Chapter 4: Scheduling**

   You have n distinct jobs, labeled $J_1, J_2, ..., J_n$, which can be performed completely independently of one another. Each job consists of two stages: first it needs to be preprocessed on a supercomputer, and then it needs to be finished on one of a local PCs. Let's say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC.

   There are at least n PCs available on the premises, the finishing of the jobs can be performed fully in parallel–all the jobs can be processed at the same time. However, the supercomputer can only work on a *single* job at a time, so the system managers need to work out an order in which to feed the jobs to the supercomputer. As soon as the first job in order is done on the supercomputer, it can be handed off to a PC for finishing; at that point in time a second job can be fed to the supercomputer; when the second job is done on the supercomputer, it can proceed to a PC regardless of whether or not the first job is done (since the PCs work in parallel); and so on.

   (a) *(10 Points)* Design a schedule for the ordering of the jobs for the supercomputer that minimize the**completion time** of the schedule. The definition of the **completion time** of the schedule is the earliest time at which *all* jobs will have finished processing on the PCs.

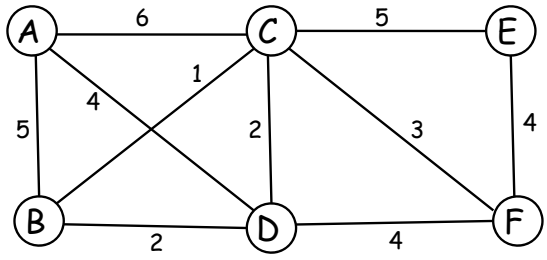
   (b) *(5 Points)* What is the running time of your algorithm?

   (c) *(10 Points)* Prove that the schedule that you designed is optimal.

   Hint: 1- Think of an order of the jobs with some rational explained and try to find a counter example until you find a good order. 2- You can use the exchange of argument to prove the optimality the same way that we did in for minimizing the max. lateness problem.

3. *(35 Points)* **Chapter 4 - Trees on Graphs (Midterm, Fall 2011).**

Consider the undirected graph shown in the following figure. It consists of six nodes A,B,C,D,E,F and nine edges with the shown edge costs.



(a) *(10 Points)* Run Dijkstra's algorithm to find the shortest paths from node A to all other nodes. (Show the final answer and briefly describe the intermediate steps.)

(b) *(10 Points)* Run an algorithm of your choice (*e.g.,* Kruskal, Prim, Reverse-Delete) and find a minimum spanning tree. (Show the final answer and briefly describe how you got there.)

(c) *(10 Points)* Is the minimum spanning tree of this graph unique? Justify your answer, *i.e.,* if the answer is yes, provide a proof; if the answer is no, provide a counter-example and explain why this is the case.

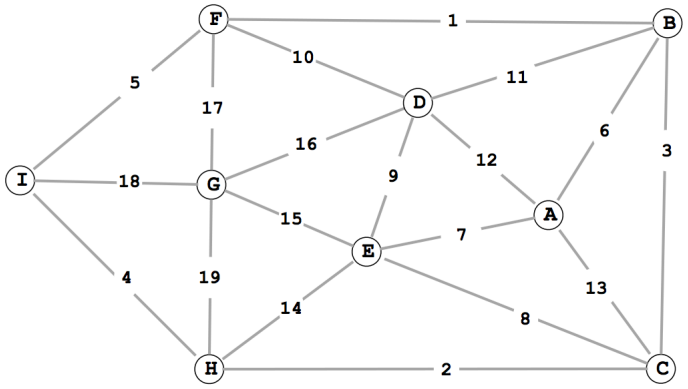
(d) *(5 Points)* Consider the average distance from A to all other nodes, first by following edges on the shortest path tree (a), let's call it d_{SPT}^{avg}; and then following edges on the minimum spanning tree found in (b), let's call it d_{MST}^{avg}. Which one is greater, d_{SPT}^{avg} or d_{MST}^{avg}? Does the same answer hold for any graph $G = (V, E)$ and node $A \in V$, or is it specific to this example?

4. *(20 points)* **Chapter 4 - (Midterm, Fall 2019)**.

Consider the undirected graph below.

(a) *(10 Points)* Use an algorithm of your choice to find a Minimum Spanning Tree (MST) for this graph. Show the final answer (draw the tree and write down its cost) and briefly describe how you got the answer (which algorithm you used and some intermediate steps).

(b) *(10 Points)* Is the MST for this graph unique? Justify your answer, i.e., if the answer is yes, provide a proof; if the answer is no, provide a second MST as a counter-example.



5. *(15 Points)* **Combinatorial Structure of Spanning Trees:** Let $\mathcal{G}$ be a connected graph, and $\mathcal{T}$ and $\mathcal{T}'$ two different spanning trees of $\mathcal{G}$. We say that $\mathcal{T}$ and $\mathcal{T}'$ are *neighbors* if $\mathcal{T}$ contains exactly one edge that is not in $\mathcal{T}'$, and $\mathcal{T}'$ contains exactly one edge that is not in $\mathcal{T}$.

   Now, from any graph $\mathcal{G}$, we can build a (large) graph $\mathcal{H}$ as follows. The nodes of $\mathcal{H}$ are the spanning trees of $\mathcal{G}$, and there is an edge between two nodes of $\mathcal{H}$ if the corresponding spanning trees are neighbors.

   Is it true that, for any connected graph $\mathcal{G}$, the resulting graph $\mathcal{H}$ is connected? Give a proof that $\mathcal{H}$ is always connected, or provide an example (with explanation) of a connected graph $\mathcal{G}$ for which $\mathcal{H}$ is not connected.