Week 12 Tutorial Sheet

(To be completed during the Week 12 tutorial class)

Objectives: The tutorials, in general, give practice in problem solving, in analysis of algorithms and data structures, and in mathematics and logic useful in the above.

Instructions to the class: Aim to attempt these questions before the tutorial! It will probably not be possible to cover all questions unless the class has prepared them in advance. There are marks allocated towards active participation during the class. You **must** attempt the problems under **Assessed Preparation** section **before** your tutorial class and give your worked out solutions to your tutor at the start of the class – this is a hurdle and failing to attempt these problems before your tutorial will result in 0 mark for that class even if you actively participate in the class.

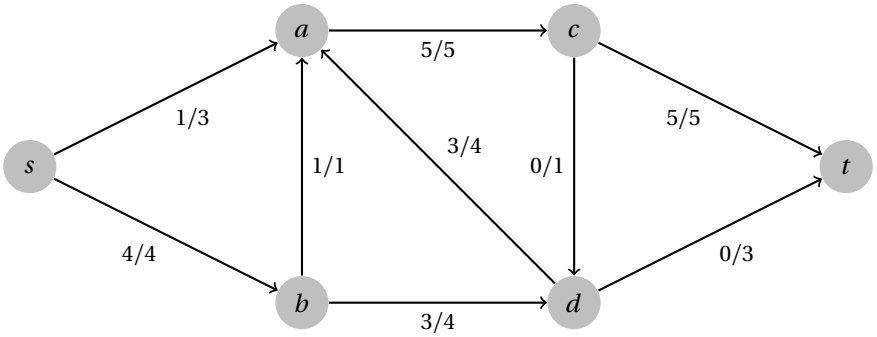
Instructions to Tutors:

1. The purpose of the tutorials is not to solve the practical exercises!
2. The purpose is to check answers, and to discuss particular sticking points, not to simply make answers available.

Supplementary problems: The supplementary problems provide additional practice for you to complete after your tutorial class, or as pre-exam revision. Problems that are marked as **(Advanced)** difficulty are beyond the difficulty that you would be expected to complete in the exam, but are nonetheless useful practice problems as they will teach you skills and concepts that you can apply to other problems.

Assessed Preparation

Problem 1. Consider the following flow network. Edge labels of the form f/c denote the current flow f and the total capacity c.

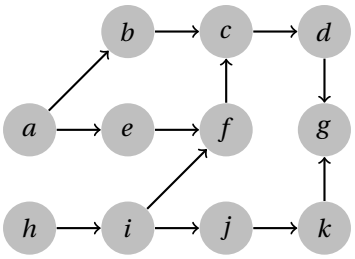


(a) Draw the corresponding residual network

(b) Identify an augmenting path in the residual network and state its capacity

(c) Augment the flow of the network along the augmenting path, showing the resulting flow network

Problem 2. Show the steps taken by Kahn's algorithm for computing a topological order of the following graph.



Tutorial Problems

Problem 3. Devise an algorithm for determining whether a given directed acyclic graph has a unique topological ordering. That is, determine whether there is more than one valid topological ordering.

Problem 4. Complete the Ford-Fulkerson method for the network in Problem 1, showing the final flow network with a maximum flow.

Problem 5. Using your solution to Problem 4, list the vertices in the two components of a minimum $s-t$ cut in the network in Problem 1. Identify the edges that cross the cut and verify that their capacity adds up to the value of the maximum flow.

Problem 6. Let G be a flow network and let f be a valid flow on G. Prove that the net outflow out of s is equal to the net inflow into t.

Problem 7. Consider a variant of the maximum network flow problem in which we allow for multiple source vertices and multiple sink vertices. We retain all of the capacity and flow conservation constraints of the original maximum flow problem. As in the original problem, all of the sources and sinks are excluded from the flow conservation constraint. Describe a simple method for solving this problem.

Problem 8. A Hamiltonian path in a graph $G = (V, E)$ is a path in G that visits every vertex $v \in V$ exactly once. On general graphs, computing Hamiltonian paths is NP-Hard. Describe an algorithm that finds a Hamiltonian path in a directed acyclic graph in $O(V + E)$ time or reports that one does not exist.

Problem 9. Consider a directed acyclic graph representing the hierarchical structure of n employees at a company. Each employee may have one or many employees as their superior. The company has decided to give raises to m of the top employees. Unfortunately, you are not sure exactly how the company decides who is considered the top employees, but you do know for sure that a person will not receive a raise unless all of their superiors do.

Describe an algorithm that given the company DAG and the value of m, determines which employees are guaranteed to receive a raise, and which are guaranteed to not receive a raise. Your algorithm should run in $O(V^2 + VE)$ time.

Supplementary Problems

Problem 10. Consider a variant of the maximum network flow problem in which vertices also have capacities. That is for each vertex except s and t, there is a maximum amount of flow that can enter and leave it. Describe

a simple transformation that can be made to such a flow network so that this problem can be solved using an ordinary maximum flow algorithm[1].

Problem 11. Two paths in a graph are *edge disjoint* if they have no edges in common. Given a directed network, we would like to determine the maximum number of edge-disjoint paths from vertex s to vertex t.

(a) Describe how to determine the maximum number of edge-disjoint $s-t$ paths.

(b) What is the time complexity of this approach?

(c) How could we modify this approach to find *vertex-disjoint paths*, i.e. paths with no vertices in common

Problem 12. (Advanced) A useful application of maximum flow to people interested in sports is the *baseball elimination* problem. Consider a season of baseball in which some games have already been played, and the schedule for all of the remaining games is known. We wish to determine whether a particular team can possibly end up with the most wins. For example, consider the following stats.

	Wins	Games Left
Team 1	30	5
Team 2	28	10
Team 3	26	8
Team 4	20	9

It is trivial to determine that Team 4 has no chance of winning, since even if they win all 9 of their remaining games, they will be at least one game behind Team 1. However, things get more interesting if Team 4 can win enough games to reach the current top score.

	Wins	Games Left
Team 1	30	5
Team 2	28	10
Team 3	29	8
Team 4	20	11

In this case, Team 4 can reach 31 wins, but it doesn't matter since the other teams have enough games left that one of them must reach 32 wins. We can determine the answer with certainty if we know not just the number of games remaining, but the exact teams that will play in each of them. A complete schedule consists of the number of wins of each team, the number of games remaining, and for each remaining game, which team they will be playing against. An example schedule might look like the following.

		Games Remaining				
	Wins	Total	vs T1	vs T2	vs T3	vs T4
Team 1	29	5	0	2	1	2
Team 2	28	10	2	0	4	4
Team 3	28	8	1	4	0	3
Team 4	25	9	2	4	3	0

Describe an algorithm for determining whether a given team can possibly end up with the most wins. Your algorithm should make use of maximum flow.

[1] Do not try to modify the Ford-Fulkerson algorithm. In general, it is always safer when solving a problem to reduce the problem to another known problem by transforming the input, rather than modifying the algorithm for the related problem.