Week 12 Studio Sheet

(To be completed during the Week 12 studio 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.

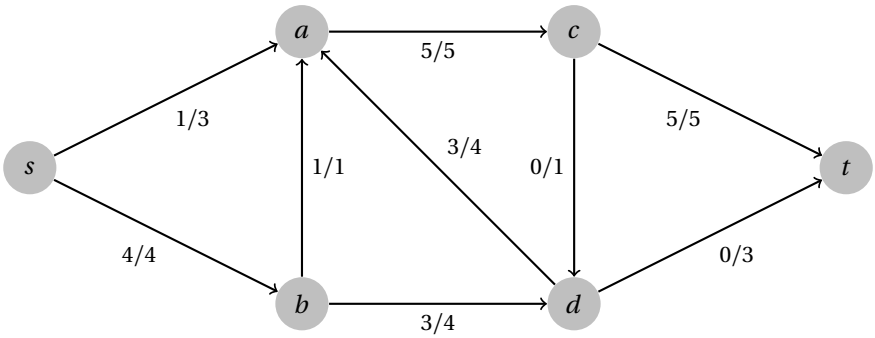
Instructions to Tutors:

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.

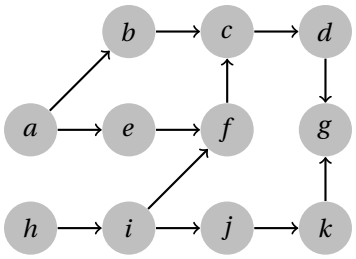


(a) Draw the corresponding residual network

(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.



Studio 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. Devise an algorithm for counting the number of paths between two given vertices s and t in a directed acyclic graph. Don't worry about the magnitude of the answer (i.e. assume that all arithmetic operations take constant time, despite the fact that the answer might be exponential in the input size.)

Problem 5. Complete the Ford-Fulkerson method for the network in Problem 1, showing the final flow network with a maximum flow.

Problem 6. Using your solution to Problem 5, 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 7. 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 8. 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 9. 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 10. 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.

Problem 11. Given a list of n integers $r_1, r_2, ..., r_n$ and m integers $c_1, c_2, ..., c_m$, we wish to determine whether there exists an $n \times m$ matrix consisting of zeros and ones whose row sums are $r_1, r_2, ..., r_n$ respectively and whose column sums are $c_1, c_2, ..., c_m$ respectively. Describe an algorithm for solving this problem by using a flow network.

Problem 12. Consider the problem of allocating a set of jobs to one of two supercomputers. Each job must be allocated to exactly one of the two computers. The two computers are configured slightly differently, so for each

job, you know how much it will cost on each of the computers. There is an additional constraint. Some of the jobs are related, and it would be preferable, but not required, to run them on the same computer. For each pair of related jobs, you know how much more it will cost if they are run on separate computers. Give an efficient algorithm for determining the optimal way to allocate the jobs to the computers, where your goal is to minimise the total cost.

Supplementary Problems

Problem 13. 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 14. 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 15. You are a radio station host and are in charge of scheduling the songs for the coming week. Every song can be classified as being from a particular era, and a particular genre. Your boss has given you a strict set of requirements that the songs must conform to. Among the songs chosen, for each of the n eras $1 \leq i \leq n$, you must play exactly n_i songs of that era, and for each of the m genres $1 \leq j \leq m$, you must play exactly m_j songs from that genre. Of course, $\sum n_i = \sum m_j$. Devise an algorithm that given a list of all of the songs you could choose from, their era and their genre, determines a subset of them that satisfies the requirements, or determines that it is not possible.

Problem 16. Recall the concept of a directed acyclic graph (DAG). A *path cover* of a DAG is a set of paths that include every vertex **exactly once** (multiple paths can not touch the same vertex). A minimum path cover is a path cover consisting of as few paths as possible. Devise an efficient algorithm for finding a minimum path cover of a DAG. [Hint: Use bipartite matching]

Problem 17. (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.

[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.

	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.

Problem 18. (Advanced) Consider a directed acyclic graph G where each edge is labelled with a character from some finite alphabet A. Given a string S over the alphabet A, count the number of paths in G whose edge labels spell out the string S. Your algorithm should run in $O((V+E)n)$, where n is the length of the string S.

Problem 19. (Advanced) Describe how an instance of the unbounded knapsack problem can be converted into a corresponding directed acyclic graph. Which graph problem correctly models the unbounded knapsack problem on this graph?

Problem 20. (Advanced) A problem closely related to the transitive closure problem mentioned in lectures is the *transitive reduction*. In a sense, the transitive reduction is the opposite of the transitive closure. It is a directed graph with the fewest possible edges that has the same reachability as the original graph. In other words, for all pairs of vertices u and v, there is a path between u and v in the transitive reduction if and only if there is a path between u and v in the original graph. Give an algorithm for computing the transitive reduction of a directed acyclic graph. Your algorithm should run in $O(V^2 + VE)$ time.

Problem 21. (Advanced) You are in charge of projects at a large company and need to decide for this year, which projects the company will undertake. Each project has a profit value associated with it. Some projects are worth positive profit, meaning you earn money from completing them. Some projects are worth negative profit, meaning they cost more money than they make. Projects have prerequisites with other projects, meaning that a project can only be completed once all of its prerequisites have been completed. The goal is to determine which projects to complete in order to make the maximum amount of profit, while ensuring that all prerequisite relationships are satisfied. This problem can be solved by reducing it to a minimum cut problem as follows:

- Create a flow network with a source vertex s and a sink vertex t
- Create a vertex for each project
- For each project x with positive profit p, add a directed edge from s to x whose weight is p
- For each project x with negative profit $-p$, add a directed edge from x to t whose weight is p
- For each project x that has y as a prerequisite, add a directed edge from x to y with weight ∞

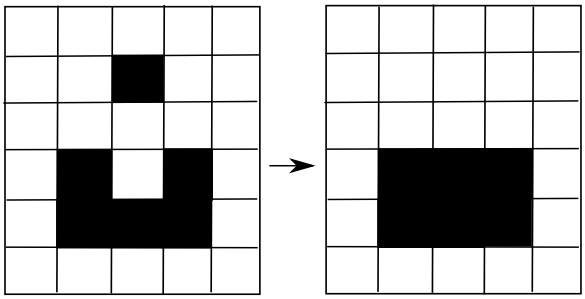
The minimum cut of this network can be used to determine the optimal set of projects to complete.

(a) Explain what the components of the minimum $s-t$ cut correspond to in the optimal solution

(b) What is the purpose of the infinite capacity edges?

(c) Explain how to compute maximum profit from the capacity of the minimum $s-t$ cut

Problem 22. (Advanced) You are the owner of a plot of land that can be described as an $n \times m$ grid of unit-square-sized cells. Each cell of land is either filled in, or a hole. The government has to decided to crack down on safety regulations and requires you to fence off the holes in the ground. Each unit of fencing will cost you $\$c_{\text{fence}}$. In order to reduce the amount of fencing required, you have the option to fill in some of the holes, or even dig some additional holes. It will cost you $\$c_{\text{dig}}$ to dig a new hole, or $\$c_{\text{fill}}$ to fill in an existing hole. For safety, there are no holes on the boundary cells of your land, and you are not permitted to make any. For example, in the following case (where black represents a hole), with $c_{\text{fence}} = 10$, $c_{\text{dig}} = 10$, and $c_{\text{fill}} = 30$, the cost to fence the land initially would be \$160. The optimal solution is to dig out the middle cell and fill in the topmost hole, yielding a total cost of \$140.



Describe how to determine the minimum cost to make your land safe by reducing it to a minimum cut problem.

Problem 23. (Extra Advanced) Consider a variant of the maximum flow problem in which we enforce lower bounds on the edge flows, i.e. we can require that an edge has at least a certain amount of flow. Formally, each edge now has a capacity c, and a demand d, and we require that $d(u,v) \leq f(u,v) \leq c(u,v)$. This problem can be solved by modifying the network and using an ordinary maximum flow algorithm.

(a) Describe how to determine whether a feasible flow exists, i.e. a flow the satisfies the demand

(b) Describe how to determine a maximum feasible flow, i.e. a maximum flow satisfying the demands

(c) Describe how to determine a minimum feasible flow, i.e. a minimum flow satisfying the demands