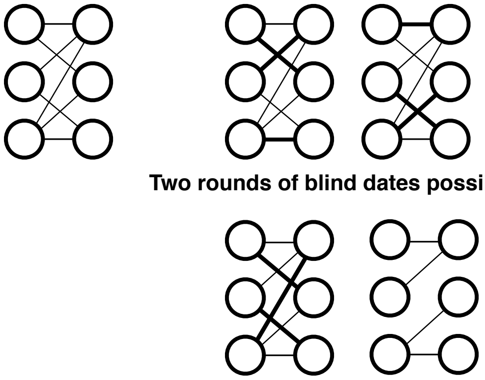
IE 512 Graphs, Networks, and Algorithms

Homework 5 Solution

Problem 5.1

Consider a network $N = (V, E)$, where the source is connected to all the male students, all female students are connected to the sink, and we add directed edges from a male student to a female student if the pair is compatible. On the edges between male and female students, we put capacity of one. On the edges connected to the sink, we put capacity of infinity. On the edges connected to the source, we put capacity of k. If we choose k and find the maximum flow, and the value of the maximum flow is kn, then we know that there are k perfect matchings in the problem. Therefore, we can start with $k = 1$ and increment k until the value of maximum flow is no longer kn.

Alternatively, we can think of a greedy algorithm that "peels off" perfect matchings one by one. Using any algorithm, either maximum bipartite matching algorithm using augmenting paths; or the above algorithm with $k = 1$, find a perfect matching on current graph. Then, delete all the edges in the matching, and repeat until there is no more perfect matching. The number of rounds this algorithm finds perfect matchings is the number of rounds we can have blind dates. This algorithm does not always give the right solution, and here is a counter example.

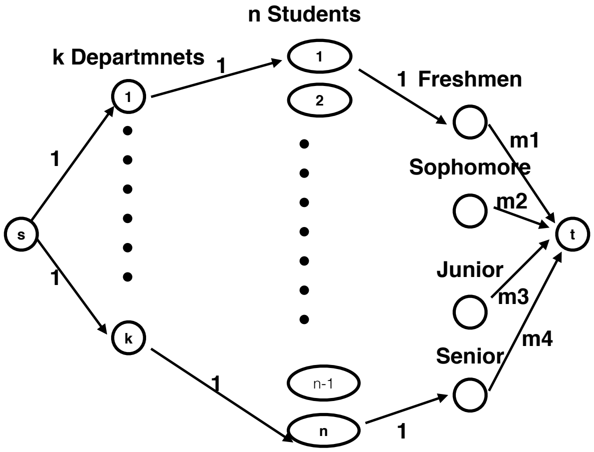


Two rounds of blind dates possible

Greedy algorithm might fail

In this example, there are two rounds of perfect matchings possible. But, a greedy algorithm can choose a sub-optimal matching in the first round, causing it to fail to find two perfect matchings. Therefore, this solution is worth 0.5 points.

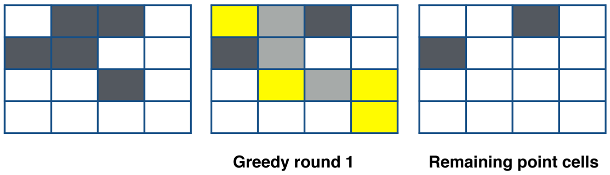
Problem 5.2 Consider the following graph. The source is connected to k departments with capacity one. Each department is connected to a subset of n students, that belong to the department with capacity one. Each student is connected to the respective group (freshmen, sophomore, junior, senior) with capacity one. Each group is connected to the sink with capacity m_i for group i. The maximum flow in this problem finds the set of students to be included in the committee.



Problem 5.3

(a) Consider the following max-flow min-cost problem. Connect the source to the cell on the top left with capacity one and the bottom right cell to the sink with capacity one. Each cell is connected to its right and below neighbors by a directed edge. On this network N, we put capacity one on every edge and cost per flow one on every edge, except for those edges that start (or end) at a point cell. The min-cost max-flow on this network N finds a flow of value one that minimizes the cost, i.e. goes through as many point cells as possible.

(b) Here is an example where greedy fails:


Greedy round 1

Remaining point cells

Greedy algorithm may require three paths, while the minimum number of paths is two

(c) (This is optional, and should not be graded.) Let M be the number of point cells. Consider the network from part (a), but this time put capacity of M on every edge. We formulate it as a max-flow problem with both upper and lower bounds on the flow (upper and lower capacities on each edges) without any cost on the edges.

Divide each point cell into two nodes, one for incoming edges and the other for outgoing edges. Then, connect those two nodes with a directed edge with upper capacity M and lower capacity 1. This means that we want a flow such that f_{ij} for that edge is at least one and at most M. We do this for every point cells. All other edges have lower capacity of zero. We want to find the minimum value flow such that makes the problem feasible: that is covers all the point cells. We can do it, for example, by chaining the capacity for the edge connecting the source to the upper left cell from one to M, incrementally, and looking for the first number that has feasible solution. To check feasibility, we can use standard algorithms for max-flow with lower and upper bounds on the flow (this was not covered in class).

Problem 5.4

(a) Run Ford-Fulkerson algorithm to find a maximum flow f. Consider the residual graph $R(G, f)$, and find a minimum cut (S, S^c) by letting S be the nodes reachable from s in $R(G, f)$. For one of the edge in $R(G, f)$ that cross over the cut, add an edge in the reverse direction (from S to S^c) with capacity one, and check if due to this added edge, we now have a path from s to t. Repeat it for all the edges in the cut, one by one. Note that you are not incrementally adding edges, but adding just one edge to the original $R(G, f)$ in each trial. If in all of those trials we get a path from s to t, then this implies that the minimum cut is unique.