Problem Set 5 Solutions

This problem set is due **at 11:59pm** on **Wednesday, March 22, 2017.**

EXERCISES (NOT TO BE TURNED IN)

Bipartite Matching

- Do Exercise 26.3-4 in CLRS (pg. 669)
- Do Problem 26-3 in CLRS (pg. 693)

Linear Programming

- Do Exercise 29.9-1 in CLRS (pg. 783)
- Do Exercises 29.4-2 in CLRS (pg. 810)

Problem 3-1. Volcano Escape [50 points]

(a) [25 points] After a public announcement that the tropical paradise of Jamuca is "probably safe", tourists are rushing to fly there to vacation. In order to satisfy all the travel requests, the local airport has asked you to help them plan flights.

There are n travel agencies a_i, where $1 \leq i \leq n$. Agency i has p_i planes reserved for tourists, each of which carries only a single person. Each agency operates only on a given subset of days $D_{a,i} \subseteq \{M, T, W, Th, F, S, Su\}$. Each plane can carry only a single tourist.

There are $m > n$ tourists t_i for $1 \leq i \leq m$ who want to buy flights. Each tourist can only fly on a given subset of days $D_{t,i} \subseteq \{M, T, W, Th, F, S, Su\}$.

Find, for each agency, a list of pairs (t_j, d_j) which tells which tourists that agency should fly, and the day it should fly them. Your schedule must satisfy the constraints described above.

Solution: Pre-Solution Steps We're being asked to match people to agencies, which suggests that this is a bipartite matching problem. However, we have the additional constraints that agencies can be matched to multiple people, and the days must overlap. All that remains is to determine how to represent these last two constraints in a flow network.

Solution We create a flow network with three layers: agencies, days of the week, and people. Edges from the source to agency i has capacity p_i and edges from people to the sink have capacity 1. We include edges from agencies to days only if the agency operates on that day, and include edges from days to people only if the person can fly on that day. We can then solve the problem with any max-flow algorithm. The graph has $|V| = m+n+9 = O(n+m)$ and $|E| = m+7m+7n+n = O(n+m)$. Edmonds-Karp runs in $O(V^2E)$ time, which yields a runtime of $O((n+m)^3)$. Ford-Fulkerson can do it in $O(f^*E) = O(m(n+m))$ time.

(b) [25 points] Unbelievably, the volcano on Jamuca is erupting! As the boiling lava creeps ever closer, the residents ask you to help them escape as quickly as possible.

There are n islanders x_i for $1 \leq i \leq n$ and $m > n$ boats b_j for $1 \leq j \leq m$. When the volcano erupts, islander i and boat j are at a distance $d_{i,j}$ from each other. Each islander moves one unit of distance per unit time, and moves in a straight line. Each boat can hold at most one islander, and an islander is considered safe as soon as they reach a boat.

Design an algorithm to decide, for each islander, which boat they should take in order the minimize the time required for all islanders to escape. Once the islanders have decided on boats, they all move simultaneously toward the ones they picked. Your algorithm should output the set of choices that minimizes the time required for the *last* islander to reach a boat.

Solution:

Pre-Solution Steps How could we find an escape plan if the allotted time s were fixed? We could treat the problem as a standard bipartite matching. We create n nodes for the islanders and m nodes for the ships. We create a source on the side of the islands and a sink on the side of the boats: we connect the source to each islander with an edge of capacity 1, and we connect each boat to the sink with an edge of capacity 1. Then, for each islander i and boat i we add an edge of capacity 1 if $d_{i,j} \leq t$.

Now we can apply any max flow algorithm to this network. Once we have a max flow, the magnitude of the flow is exactly the maximum number of islanders who can safely escape and each islander is assigned to a boat by an edge with nonzero flow.

We can treat this a subroutine for solving the rest of the problem. The key insight here is that if all villages can actually reach a boat within a fixed time t, this time is an upper bound on the maximum time it would take any islander to reach a boat in the ideal solution.

Now the problem reduces to searching for the tightest possible upper bound, which we can do efficiently using binary search. This is a very common approach for minimization problems when you know the upper bound must lie in some discrete sorted set. In this case, we know the upper bound on the maximum time must be equal to one of the distances $d_{i,j}$, so we just have to binary search over the possible distances.

Solution

Knowing this, we can find the fastest possible escape plan by performing a binary search on the length of time required for everyone to escape.

To implement our solution, we create an array with all of the times $d_{i,j}$ sorted in increasing order. We binary search through this array: at each step of the search, we construct the flow graph and find the maximum matching with t set to the median value of the subarray we are searching. If we find that everyone can escape in at most t time, we recurse on the lower half of the array. Otherwise, we recurse on the upper half. Once we have reached the base case, we know that t is the minimum possible value for everyone to escape, and we can thus use the matching we found with this last t as the best assigment of islanders to boats to solve our problem.

The number of edges is $O(mn) = O(m^2)$. The value of the max flow is bounded by $n = O(m)$, so if we use Ford-Fulkerson the runtime will be $O(E|f|) = O(m^3)$ for each round of matching.

We need to binary search over $O(nm) = O(m^2)$ different distances. Each step of the search takes $O(m^3)$ time using the subroutine defined in the pre-solution above; this yields a runtime of $O(m^3 \log m^2) = O(m^3 \log m)$.

Problem 3-2. Blamazon Prime [50 points]

Shopping giant Blamazon has just introduced Blamazon Prime, a service that allows people to request to have goods delivered to them regularly. Unfortunately, the producers and customers are

scattered everywhere and all of the roads have tolls. The CEO of Blamazon has asked you to help him find the most efficient way to deliver to everyone.

The city is modeled as a directed graph $G = \{V, E\}$. Each edge e has a cost $c(e)$ which you will incur for each unit delivered across it. Each vertex v has a demand $d(v)$, must be the net amount of goods entering it at any time. Vertices v with $d(v) > 0$ are producers, and those with $d(v) < 0$ are consumers. Your goal is to find for each edge an amount of goods to ship across that edge such that all edge demands and capacities are satisfied, and the total cost is minimized.

(a) [25 points] Express this problem as a linear program.

Solution:
Pre-Solution Steps The objective function is given in the problem as

$$\sum_{e \in E} c(e) \cdot f(e)$$

The demand is a generalization of the conservation constraints from regular max flow, and the capacities can be represented the same way they are in regular max flow.
Solution The full linear program is

$$\begin{aligned} &\min \sum_{e \in E} c(e) \cdot f(e) \\ \text{subject to } &\sum_{(u,v) \in E} f(u,v) - \sum_{(v,u) \in E} f(v,u) = d(v) \qquad \forall v \in V \\ &f(u,v) \geq 0 \end{aligned}$$

(b) [25 points] Calculate the dual of your linear program from part (a), and express it in standard form as inequality constraints. Furthermore, interpret what the variables and constraints mean in your own words.

Solution:
Pre-Solution Steps The matrix used in the matrix formulation of the primal is the incidence matrix of G. Therefore the dual will have E equations rather than V, and each will be the difference of two variables corresponding to vertices. Since we require strict equality in the primal, it is useful to use the asymmetric version of the dual.
Solution The primal can be written as

$$\begin{aligned} &\min c^T x \\ &\text{s.t. } Ax = b \\ &x \geq 0 \end{aligned}$$

where A is the incidence matrix of G. Therefore the dual is

$$\begin{aligned} &\max \sum_{v \in V} d(v) \cdot y_v \\ &\text{s.t. } y_{\text{in}} - y_{\text{out}} \leq c_e \qquad \forall e \in E \end{aligned}$$

where y_{in} is a variable assigned to the vertex where edge e starts, and y_{out} is a variable assigned to the vertex where edge e ends. Therefore the dual problem is to assign each node a value (or potential) such that the difference across each edge is no more than the cost, and the values of the nodes weighted by demand are maximized.