Sample Solution to Problem Set 5

(The solutions to the exercises from the book are taken from those provided by the authors.)

1. (12 points) Planning a company party

We are given with the tree structure of the company hierarchy and the conviviality ratings of each person. Let $\textsc{Total}_T$ be the sum of conviviality ratings of the guests selected from the tree rooted at T in the optimal solution. In the tree, if T is invited, then none of its children (employees whom he supervises) is invited. However, if T is not invited then some of its children may or may not be invited. Let $\textsc{Total}_{T(i)}$ be the sum of the ratings of the guest selected from tree rooted at T, when T is invited. Let $\textsc{Total}_{T(ni)}$ be the sum of the ratings of the guests selected from the tree rooted at T, when T is not invited.

Now,

$\textsc{Total}_T = \max(\textsc{Total}_{T(i)}, \textsc{Total}_{T(ni)})$

Also,

$\textsc{Total}_{T(i)} = \sum_{x \in children[T]} \textsc{Total}_{x(ni)} + \textsc{Rating}_T$

$\textsc{Total}_{T(ni)} = \sum_{x \notin children[T]} \textsc{Total}_x$

where $children[T]$ is the set comprising of children of node T in tree

Base Cases:

$\textsc{Total}_{l(i)} = \textsc{Rating}_l$

$\textsc{Total}_{l(ni)} = 0$

where l is a leaf in the tree.

Now, we can compute the value of optimal solution in bottom-up fashion with the help of recurrence relation and the base cases. For each node y in the tree we need to calculate $\textsc{Total}_{y(i)}$ and $\textsc{Total}_{y(ni)}$ starting from the leaves up to the root of the tree. $\textsc{Total}_y$ is selected to be maximum of $\textsc{Total}_{y(i)}$ and $\textsc{Total}_{y(ni)}$. It takes $O(n)$ time to compute these values where n is the number of nodes in the tree. We seek the value of $\textsc{Total}_T$ where T is the root of the tree. Once the $\textsc{Total}_y$ is computed for all the nodes y, we build the optimal solution (guest list) starting from the root T and either selecting it or not depending on the values of $\textsc{Total}_{T(i)}$ and $\textsc{Total}_{T(ni)}$ and following the recurrence relation.

2. (12 points) Chapter 6, Exercise 17, page 327.

(a) Consider the sequence 1,4,2,3. The greedy algorithm produces the rising trend 1,4, while the optimal solution is 1,2,3.

(b) Let $\textsc{Opt}(j)$ be the length of the longest increasing subsequence on the set P[j], P[$j+1$],....,P[n], including the element P[j]. Note that we can intialize $\textsc{Opt}(n)$=1, and $\textsc{Opt}(1)$ is the length of the longest trend, as desired.

Now, consider a solution achieving $\textsc{Opt}(j)$. Its first element is $\mathrm{P}[j]$, and its next element is $\mathrm{P}[k]$ for some $k > j$ for which $\mathrm{P}[k] > \mathrm{P}[j]$. From k onward, it is simply the longest increasing subsequence that starts at $\mathrm{P}[k]$; in other words, this part of the sequence has length $\textsc{Opt}(k)$, so including $\mathrm{P}[j]$, the full sequence has length $1 + \textsc{Opt}(k)$. We have thus justified the following recurrence.

$$\textsc{Opt}(j) = 1 + \max_{k>j:\mathrm{P}[k]>\mathrm{P}[j]} \textsc{Opt}(k)$$

The values of Opt can be built up in order of decreasing j, in time $O(n-j)$ for iteration j, leading to a total running time of $O(n^2)$. The value we want is $\textsc{Opt}(1)$, and the subsequence itself can be found by tracing back through the array of Opt values.

3. (12 points) Chapter 7, Exercise 7, page 417.

We build the following flow network. There is a node v_i for each client i, a node w_j for each base station j, and an edge (v_i,w_j) of capacity 1 if client i in within range of base station j. We then connect a super-source s to each client nodes by an edge of capacity 1, and we connect each of the base station nodes to a super-sink t by an edge of capacity L.

We claim that there is a feasible way to connect all clients to base stations if and only if there is an s-t flow of value n. If there is a feasible connection, then we send one unit of flow from s to t along each of the paths s, v_i,w_j,t, where client i is connected to base station j. This does not violate the capacity conditions, in particular on the edges (w_j,t), due to the load constraints. Conversely, if there is a flow of value n, then there is one with integer values. We connect client i to base station j if the edge (v_i,w_j) carries one unit of flow, and we observe that the capacity condition ensures that no base station is overloaded.

The running time is the time required to solve a max-flow problemon a graph with $O(n+k)$ nodes and $O(nk)$ edges.

4. (12 points) Chapter 7, Exercise 8, page 418.

(a) To solve this problem, construct a graph $G = (V, E)$ as follows: Let the set of vertices consist of a super-source node, four supply nodes (one for each blood type) adjacent to the source, four demand nodes and a super sink node that is adjacent to the demand nodes. For each supply node u and a demand node v, construct an edge (u,v) if type v can receive blood from type u and set the capacity to ∞ or the demand for type v. Construct an edge (s, u) between the source s and each supply node u with the capacity set to the available supply of type u. Similarly, for each node v and the sink t, construct an edge (v,t) with the capacity set to the demand for type v.

Now compute an (integer-valued) maximum flow on this graph. Since the graph has constant size, the scaling max-flow algorithm takes time $O(\log C)$, where C is the total supply, and the Preflow-Push algorithm take constant time.

We claim there there is sufficient supply for the projected need if and only if the edges from the demand nodes to the sink are all saturated in the resulting max-flow. Indeed, if there is sufficient supply, in which s_{ST} of type S are used for type T, then we can send a flow of s_{ST} from the supply node of type S to the demand node of type T, and respect all capacity conditions. Conversely, if there is a flow saturating all edges from demand nodes to the sink, then there is an, integer flow with this property; if it sends f_{ST} units of flow from the supply node for type S to the demand

node for type T, then we can use f_{ST} nodes of type S for patients of type T.

(b) Consider a cut containing the source, and the supply and demand nodes for B and A. The capacity of this cut is 50+36+10+3 = 99, and hence all 100 units of demand cannot be satisfied.

An explanation for the clinic administrators: There are 87 people with demand for blood types O and A; and there are only 86 such donors.

Note: We observe that part (a) can also be solved by a greedy algorithm; basically it works as follows. The O group can only receive blood from O donors; so if O group is not satisfied, there is no solution. Otherwise, satisfy the A and B groups using the leftovers from the O group; if this is not possible, there is no solution. Finally, satisfy the AB group using any remaining leftovers. A short explaination of correctness (basically following from above reasoning) is necessary for this algorithm, as it was with with the flow algorithm.

5. (12 points) Updating the maximum flow in a network

You are given a directed network G with n nodes and m edges, a source s, a sink t and a maximum flow f from s to t. Assume that the capacity of every edge is a positive integer. Describe an $O(m+n)$ time algorithm for updating the flow f in each of the following two cases.

(a) The capacity of an edge e increases by 1.

(b) The capacity of an edge e decreases by 1.

Both the parts can be solved by using the concepts of augmenting paths and the residual network.

(a) We first note that the increase in capacity of one edge can increase the max flow can increase by at most 1. This follows directly from the max-flow min-cut theorem and the fact that the min cut capacity can increase by at most 1. Therefore, either the max flow increases by 1 or remains the same.

Consider the residual network obtained from the original maximum flow f. Let G_f denote the residual network. Suppose we are given that the capacity of the edge (u, v) has increased by 1. We increase the capacity of the edge (u, v) in the residual network by 1. If (u, v) was not in the residual network, then we now add it and set its residual capacity to 1. Let G'_f denote the residual network thus obtained. We now run one iteration of the Ford-Fulkerson algorithm on G'_f. If we find an augmenting path from s to t in G'_f, then the flow increases by 1; otherwise, it remains the same.

The running time of this algorithm is the same as the running time of one iteration of the Ford-Fulkerson algorithm, which is $O(V + E)$.

(b) We first note that the decrease in capacity of one edge can decrease the max flow can increase by at most 1. This follows directly from the max-flow min-cut theorem and the fact that the min cut capacity can decrease by at most 1. Therefore, either the max flow decreases by 1 or remains the same.

Consider the residual network obtained from the original maximum flow f. Let G_f denote the residual network. Suppose we are given that the capacity of the edge (u, v) has decreased by 1. We consider two cases.

The first case is when (u, v) is in G_f. This means that the residual capacity of (u, v) was at least 1 before the decrease. In this case, (u, v) can still carry its original flow even if its capacity is decreased by 1. Therefore, the max flow value remains the same. To maintain the residual network, we reduce the residual capacity of (u, v) by 1; if its residual capacity becomes 0, we remove it from the graph.

The second case is when (u, v) is not in G_f. This implies that $f(u, v)$ equals $c(u, v)$. The capacity of (u, v) has decreased by 1, however. Therefore, we have to reduce the flow on (u, v) by 1. In order to maintain the flow conservation law, we have to reduce the flow on each edge along some path from s to t containing (u, v) by 1. We find such a path by traversing the residual network. We find a path from u to s and from t to v in the residual network. This together with the edge (v, u) in the residual network gives a path p from t to s in the residual network. For each edge (x, y) in the path p, we make the following changes: we increase $f(x, y)$ by 1 and decrease $f(y, x)$ by 1. Note that one of the effects of these changes is that $f(u, v)$ decreases by 1, thus satisfying the capacity constraint of (u, v). Furthermore, the net flow value has also decreased by 1. We now consider the new residual network G'. We run one more iteration of the Ford-Fulkerson algorithm. If we find an augmenting path from s to t in G', then the flow increases by 1, thus resulting in the same max flow as before; otherwise, it remains the same, thus resulting in an overall flow that has decreased by 1.

The above algorithm performs two searches in G_f, one from t to v and the other from u to s. Each of these takes $O(V+E)$ time. And finally, we perform one iteration of the Ford-Fulkerson algorithm, which takes $O(V + E)$ time. The total running time is thus $O(V + E)$.