1) Consider the following matching problem. There are m students $s_1, s_2, \ldots, s_m$ and a set of n companies $C = \{c_1, c_2, \ldots, c_n\}$. Each student can work for only one company, whereas company c_j can hire up to b_j students. Student s_i has a preferred set of companies $\Lambda_i \subseteq C$ at which he/she is willing to work. Your task is to decide whether there is an assignment of students to companies such that all of the above constraints are satisfied and each student is assigned. Formulate this as a network flow problem and describe any subsequent steps necessary to arrive at the solution. Prove correctness.

Solution:

Construction of flow network: Represent each student and each company as a separate node. Add one source node s and a sink node t for a total of $m+n+2$ nodes. Add the following directed edges with capacities:
i. $s \rightarrow s_i$ with capacity 1 unit, for each $1 \leq i \leq m$,
ii. For each $1 \leq i \leq m$, $s_i \rightarrow c$ with capacity 1 unit for all nodes $c \in \Lambda_i$.
iii. $c_j \rightarrow t$ with capacity b_j units, for each $1 \leq j \leq n$.

Constructing the solution: Run any max-flow algorithm on the above network to get the realization of edge flows under a max flow configuration (lets call it O for brevity). If an edge $s_i \rightarrow c_j$ shows a non-zero flow in this configuration, assign student s_i to company c_j. Repeat this process for each edge between the student nodes and the company nodes to get the final assignment.
Proof of correctness: We have to show that the solution so obtained satisfies all constraints of the problem.
i. Since all outgoing edges from s_i are to the node set Λ_i, it is impossible for s_i to have a flow to a node outside Λ_i in configuration O.
ii. Since the only outgoing edge from c_j is of capacity b_j, configuration O cannot have more than b_j incoming edges of non-zero flow to node c_j. Thus, not more than b_j students can get assigned to company c_j.
iii. As s_i has a single incoming edge and multiple outgoing edges of capacity 1, configuration O cannot have more than one outgoing edge from s_i with non-zero flow. Hence, s_i can get assigned to at most one company.

We have proved that our mapping from configuration O to an assignment does not violate any of the constraints except possibly that all students might not be assigned. Note that this may happen in practice if it is impossible to assign all students while satisfying all of the given constraints. Thus, what we need to show is that if there exists a feasible assignment then configuration O will have each $s \rightarrow s_i$ edge carry a non-zero flow. Given a feasible assignment $\{(s_{i,\sigma(i)}), 1 \leq i \leq m\}$, by construction of the flow network, it is possible to set each edge $s_i \rightarrow c_{\sigma(i)}$ to carry 1 unit of flow. By feasibility of the assignment, company c_j gets no more than b_j incoming edges with non-zero flow, so the outgoing edge from c_j has enough capacity to carry away all incident flow on node c_j. Finally, since each s_i has an outgoing flow of 1 unit in this assignment, all $s \rightarrow s_i$ edges can be set to have 1 unit of flow, completing the proof.

2) Say you have a graph G and a collection of sources (s1, …sk) and sinks (t1, ….tk). You want to route a path from each source s_i to any one of the sink nodes, but you don't want the paths to share any edges, or share any sink nodes. Present a solution to this problem and describe how you determine whether a solution exits. Also provide complexity analysis.

Solution:

1. Add super source s connecting to si with capacity 1; add super sink t connecting to tj with capacity 1.
2. Assign capacity 1 to all other edges
3. Run max-flow from s to t, to check if we can get a flow value of k (k disjoint paths)
4. If so, we have k disjoint path, and we just choose the parts of the paths within the original network.

Extension, no share of nodes.

Split each node v in the network into two virtual nodes: v_in and v_out. The node v_in connects all the incoming edges to v, while v_out connects all the outgoing edges of v. Add an edge from v_in to v_out with capacity 1. Then we form a new network G'. Do the same procedure as in the previous subproblem.

After running max-flow, we merge v_in and v_out back into v, and see if there are k "strongly" disjoint edges.

3) You are given a flow network with source s, sink t and edge capacities. Further, for every vertex v in the network, you are given a non-negative number dv which is the vertex capacity. Design a polynomial time algorithm that finds a flow of maximum value that (in addition to satisfying the usual edge capacity constraints and flow conservation constraints) satisfies the constraint that for every vertex v, the flow into v is at most dv.

Solution

We construct a new network G' from G : For each node v assign two nodes vin and vout and connect them with an edge with capacity of dv . Connect all v incoming and outgoing edges to vin and to vout , respectively with the same capacity. Now. Run Ford-Fulkerson algorithm to find out the maximum flow.

4) You are given a flow network with unit capacity edges. It consists of a directed graph G = (V, E), a source s and a destination t, both belong to V. You are also given a parameter k. The goal is to delete k edges so as to reduce the maximum flow in G as much as possible. Give an efficient algorithm to find the edges to be deleted. Prove the correctness of your algorithm and show the running time.

Solution:

If the minimum s-t cut has size no larger than k, then we can reduce the flwo to 0.

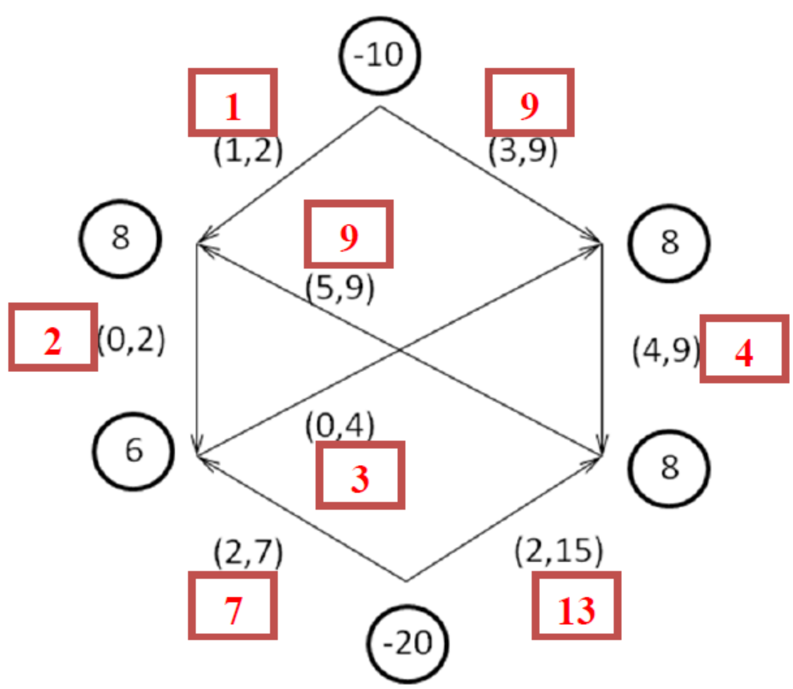
Otherwise, let f_{max} be the value of the maximum s-t flow (f_{max}>k). We identify a minimum s-t cut (A*,B*), and delete k of the edges out of A. The resulting subgraph has a maximum flow value of f_{max}-k.

Proof:

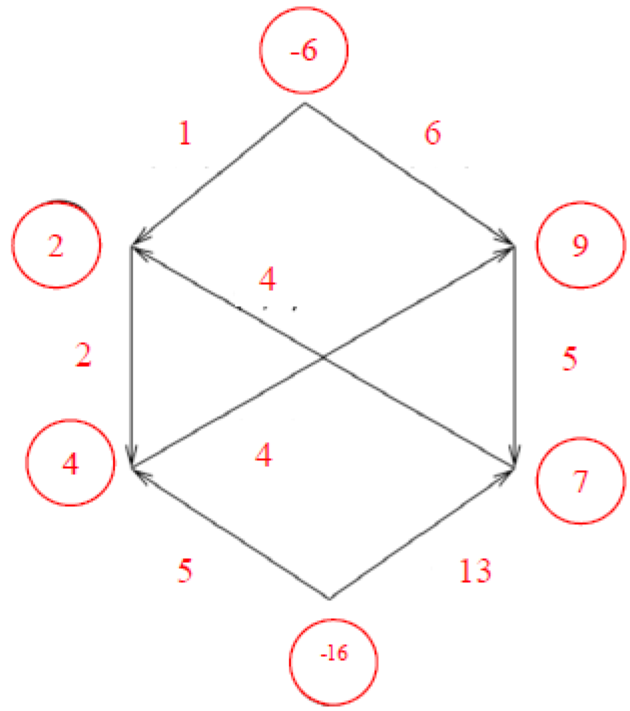
But we claim that for any set of edges F of size k, the subgraph G'=(V,E-F) has an s-t flow of value at least f_{max}-k.

Consider any cut (A,B) of G'. There are at least f edges out of A in G, and at most k have been deleted, so there are at least f-k edges out of A in G'. Thus, the minimum cut in G' has value at least f_{max}-k, and so there is a flow of at least this value.

5) Solve the following feasible circulation problem. Determine if a feasible circulation exists or not. If it does, show the feasible circulation. If it does not, show the cut in the network that is the bottleneck.



Solution:
First we eliminate the lower bound from each edge:



Then, we attach a super-source s* to each node with negative demand, and a super-sink t* to each node with positive demand. The capacities of the edges attached accordingly correspond to the demand of the nodes.
We then seek a maximum s*-t* flow. The value of the maximum flow we can get is exactly the summation of positive demands. That is, we have a feasible circulation with the flow values inside boxes shown as above.

6) An Edge Cover on a graph $G = (V; E)$ is a set of edges $X \subseteq E$ such that every vertex in V is incident to an edge in X. In the Bipartite Edge Cover problem, we are given a bipartite graph and wish to find an Edge Cover that contains $\leq k$ edges. Design a polynomial-time algorithm to solve it and justify your algorithm.

Solution 1:

Using circulation:

Algorithm:

Consider a bipartite graph with blue and red nodes: B={b1,b2….bn} and R={r1, r2….rm}.

For each edge (bi, rj), assign capacity range [0,1] to it: lower bound 0, upper bound 1;

Create super source s, connect it to each bi, with capacity range [1,+inf];

Create super sink t, connect each rj to t, with capacity range [1, +inf];

Connect from t to s by an edge with capacity range [0,k].

Claim: there is an edge cover (EC) with size no larger than k if and only if there is a feasible circulation in the above network.

Proof:

Two aspects:

1. EC size no larger than k ->feasible circulation: each edge (s, bi)'s flow and (rj, t)'s flow are at least 1 satisfying the lower bound, and the total flow over link (t,s) is no larger than k.
2. Feasible circulation -> EC size no larger than k: with the circulation feasible, each nodes bi and ri will connect at least 1 unit of flow edge; the total flow no larger than k will results in that the edge cover size is no larger than k.

   For details see the review lecture notes.

Solution 2:

Algorithm:

i) The goal of edge cover is to choose as many edges as possible which cover 2 nodes. You can find this subset of edges by running Bipartite Matching on the original graph, and taking exactly the edges which are in the matching. (Equivalently, you can set capacity of each edge in the graph as 1. Set a super source node s connecting each "blue" node with edge capacity 1 and a super destination t connecting each "red" node with edge capacity 1. Then run max-flow to get the subset of edges connecting 2 nodes in G)
ii) What remains is to cover the remaining nodes. Since you can only cover a single node (of those remaining) with each selected edge, simply choose an arbitrary incident edge to each uncovered node.
iii) Set the set of edges you choose in the above two steps as set X. Count the total number of edges in X and compare the size with k.

Proof:

It is obvious that X is an edge cover. The remaining part is to show that X contains the minimum number of edges among all possible edge covers.

Denote the number of edges we find in step i) as x1; denote the number of edges we find in step ii) as x2.

Then we have x1 * 2 + x2 = |V|.

Consider an arbitrary edge cover set Y. Suppose Y contains y1 edges, each of which is counted as the one covering 2 nodes. Suppose Y contains y2 edges, each of which is counted as the one covering 1 nodes. (We can ignore the edges covering zero nodes, because we can delete those edges from Y without affecting the coverage)

Then we have y1 * 2 +y2 = |V|.

Here x1 must be the maximum number of edges that covers 2 nodes in the bipartite graph, because we do bipartite matching in G (max-flow in G′ including s and t). Therefore, we have x1 >= y1.

Then we have:

x1+x2 = |V| - x1 = y1* 2 +y2 –x1 = y1 + y2 - (x1-y1) <= y1 + y2.

The above algorithm gives the minimum number of edges for covering the nodes.