<u>Solution Sketch for Tutorial-3</u>

1. Let G be a network with source s, sink t, and integer capacities. Prove or disprove the following statements:

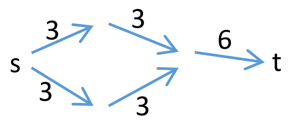
   (a) If all capacities are even then there is a maximal flow f such that f(e) is even for all edges e.

   (b) If all capacities are odd then there is a maximal flow f such that f(e) is odd for all edges e.

   [Solution Sketch]

   (a) is true because since capacities are all even, we can divide the capacity by 2 for all edges, which will still give integral capacity for all edges, and so there exists a maximum flow which will assign integral flow to all edges in this new graph. Now just multiply the flows along each edge by 2 to get the maximum flow in the original graph with f(u,v) even for all edges (u,v).

   (b) is false. Consider

   So all capacities are odd. But max flow is 6 and the last edge must have 6 as it is a single edge to t.

2. In some country there are n cities and m bidirectional roads between them. Each city has an army. Army of the i-th city consists of a_i soldiers. Now soldiers roam. After roaming each soldier has to either stay in his city or go to the one of neighboring cities by moving along at most one road. Check if it is possible that after roaming there will be exactly b_i soldiers in the i-th city.

   [Solution sketch] Model it as a maximum flow problem as follows. Let's build a flow network in following way. Make a source. Make a first group of vertices consisting of n vertices, each of them for one city. Connect a source with i-th vertex in first group with edge that has capacity a_i. Make a sink and a second group of vertices in the same way, but use b_i except for a_i for connecting the vertices with the sink. If there is a road between cities i and j or i = j, make two edges, first should be connecting the i-th vertex from first group to the j-th vertex from second group, and has infinity capacity. Second should be similar, but connect j-th vertex from first group to i-th vertex from second group. Then find a maxflow in this graph. If maxflow is equal to sum of a_i (which is equal to sum of b_i), then the answer is yes, otherwise no.

3. You have seen how to find edge-connectivity of a graph using max-flow. Suppose that we want to find not only the minimum number of edges that disconnect the graph, but also the exact set of edges. Can you modify the algorithm you studied to do that?

   [Solution sketch] Run the algorithm for finding edge connectivity. For the pair (u,v) that gives the minimum, traverse the final residual graph from u to find the set S of all nodes reachable from u in the residual graph. Then find all edges that go between S and V-S in the original graph, where V is the set of all nodes.

4. A number k of trucking companies, $c_1,...,c_k$, want to use a common road system, which is modeled as a directed graph, for delivering goods from source locations to a common target location. Each trucking company c_i has its own source location, modeled as a vertex s_i in the graph, and the common target location is another vertex t. (All these k +1 vertices are distinct.) The trucking companies want to share the road system for delivering their goods, but they want to avoid getting in each other's way while driving. Thus, they want to paths in the graph, one connecting each source s_i to the target t, such that no two trucks use a common road. We assume that there is no problem if trucks of different companies pass through a common vertex. Design an algorithm for the companies to use to determine k such paths, if possible, and otherwise return "impossible".

   [Solution Sketch] Model this as a max flow problem. Add a special source vertex s, with edges to all the individual sources s_i, each with capacity 1. Also associate capacity 1 with every other edge. Now find a max-flow from s to to. If the max-flow value is = k, then a solution is possible, else not. If max-flow = k, it is easy to find a path for each company (for each company I, follow edges from s_i with flow = 1. It is guaranteed to find such a path that reaches till t. Now change the flow on all the edges on this path to 0 and repeat for the next company. A path must exists for each company as total flow is k and each path can carry only 1).

5. Prove that every k-regular bipartite graph has a perfect matching (prove using notions of max flow).

   [Solution Sketch] Use the standard max-flow algorithm for finding matching on bipartite graph, but each edge (u,v) has weight 1/k. Weights of s to u and v to t = 1 as usual. What is the value of max-flow (and maximum matching size in the original graph) in this graph?

6. How fast can you find the maximum matching in a bipartite graph using maximum flow? (Use push-relabel in specific order)

   [Solution sketch] Use preflow push with push-relabels in a specific order. Suppose the two partitions are X and Y (the source is connected to X). Then from the initial preflow and h values, first relabel all nodes in X, then push from all nodes in X to nodes in Y, then relabel all nodes in Y, then push from all nodes in Y to t, then push back any excess flow in similar order (complete and verify the complexity).

7. Finding max-flow in a graph when each edge also has a lower bound that must be satisfied by the flow.

[Solution Sketch] This is a standard problem with solutions in many places on the net, the intuition towards the solution was discussed in class. To see the complete solution, see for example https://courses.engr.illinois.edu/cs498dl1/sp2015/notes/25-maxflowext.pdf