1. Suppose that a maximum flow is already computed in a graph with integral capacities. Then the capacity of exactly one edge is increased by 1. Which of the following is true about the complexity of finding the new maximum flow?
   a) O(1), as the max flow will not change
   b) O(1), as the capacity of only 1 edge is increased by 1
   c) O(V+E)
   d) O(VE)

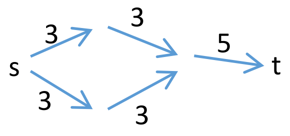
   (c) is true. This is because increasing the capacity of an edge by 1 can either make the maxflow remain the same or increase the maxflow by at most 1 (Can you prove this now that you know that maximum flow value = capacity of minimum cut? Just think whether the edge whose capacity is increased belongs to the minimum cut or not). So the new max flow will be found by at most one step of finding the augmenting path and augmenting capacity as the flows are integral, and if an augmenting path is found, it must increase the flow value by at least 1. This can be done in O(V+E) time. (Technically, O(VE) is also true, but we are looking for the tightest bound possible among the given ones).

2. 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 maximum flow f such that f(e) is even for all edges e.
   b) If all capacities are odd then there is a maximum 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 5 and the last edge must have 5 as it is a single edge to t. This must be achieved by only the combination of flows 2+3 or 3+2 in the upper and lower paths.

3. 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 the 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.

4. 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, with capacity of each edge = 1. Assign a flow = 1 on all edges from s and all edges into t, and assign flow = 1/k for all edges. Then this satisfies the capacity constraint and flow conservation constraint, so this is a feasible flow. Also, all edges from s are saturated, so this is a max flow. So ax flow value = n (the no. of nodes in each partition). Now since the capacities are all integers, by integral flow theorem, there exists a max flow with value n with all flows integral. So there exists a perfect matching.

5. The edge connectivity of an undirected connected graph is the minimum number of edges that has to be removed from the graph to make it disconnected? Given algorithm to find the edge-connectivity of a graph.

6. 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 exist for each company as total flow is k and each path can carry only 1).

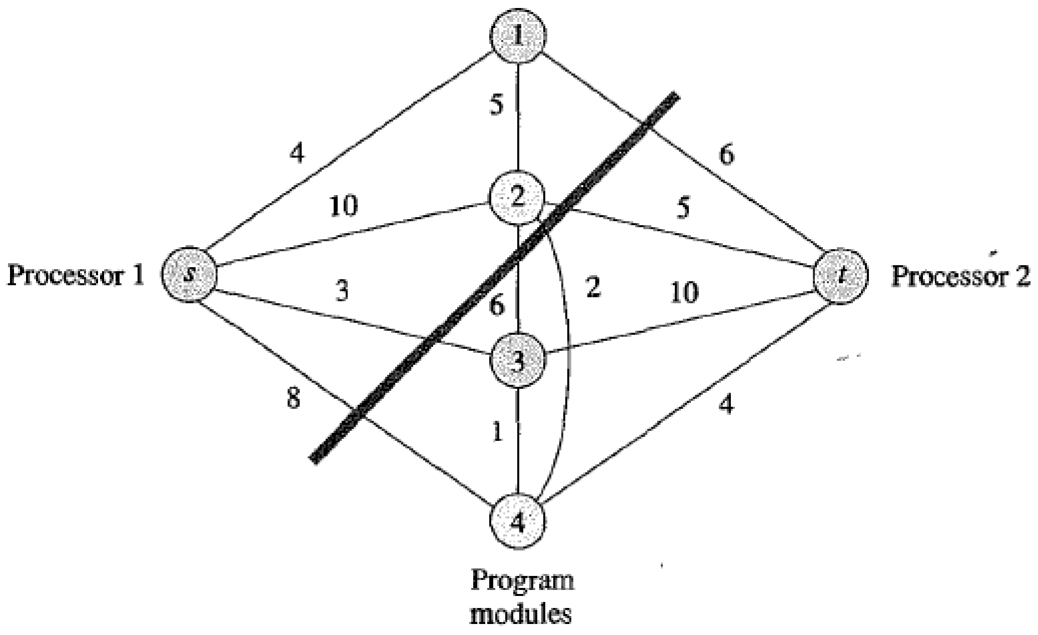
7. Consider a large task T that can be decomposed into n subtasks T_1, T_2, …T_n. The subtasks are to be performed on 2 machines M_1 and M_2. A subtask can be done on any of the two machines, but will incur different costs on the two machines. Specifically, subtask T_i incurs a cost C_i if it is done on machine M_1, and cost D_i if it is done on machine M_2. Also, some pairs of these subtasks need to communicate among themselves. If such a pair of subtasks T_i and T_j are run on the same machine the communication cost incurred is 0, otherwise, they incur a communication cost of P_{ij}. If a pair of subtasks do not need to communicate, they incur no communication cost irrespective of which machines they are run on. Find an assignment of the subtasks to the two machines so that the total cost of executing T is minimized. (Hint: Reduce the problem of finding the minimum cost of executing T to that of finding a minimum cut in a graph that you construct. First show the construction of the graph clearly stating what are the nodes and edges, and then argue why the minimum cut in this graph will give the minimum cost).

Create a graph as follows:

- Create one node X_i for each task T_i
- Create one node for M_1 and one node for M_2
- Add an edge between M_1 and each task node X_i with capacity D_i
- Add an edge between M_2 and each task node X_i with capacity C_i
- Add an edge between X_i and X_j if P_{ij} is not equal to 0.

Now find the maximum flow from M_1 to M_2 in this graph to find the minimum cut. Assign a task T_i to M_1 if it is on M_1's side of the cut, and to M_2 if it is on M_2's side of the cut.

Note that we reversed the capacity assignment on edges between M_1 and X_i (assigned D_i instead of C_i) and between M_2 and X_i (assigned C_i instead of D_i). You can also assign capacity C_i to (M_1, X_i) edge and D_i to (M_2, X_i) edge, but then you need to assign the tasks to the machine on the opposite side of the cut.



The picture shows an example graph and a cut with 4 tasks (marked 1 to 4) scheduled on two machines/processors, with $C_1 = 6$, $C_2 = 5$, $C_3 = 10$, $C_4 = 4$, $D_1 = 4$, $D_2 = 10$, $D_3 = 3$, $D_4 = 8$, $P_{24} = 2$, and all other P_{ij}'s = 0. Here Tasks 1 and 2 will be scheduled on Processor 1, and 3 and 4 on Processor 2.

Why is the minimum cut a minimum cost assignment? Firstly, all tasks will be on one or the other side of the cut, so all tasks are assigned to one of the machines. Also, (i) a cut will include exactly one of (M_1, X_i) or (M_2, X_i) edge, so every task is assigned to exactly one machine, and (ii) if two tasks i and j are assigned to different machines and P_{ij} is not equal to 0, the cut must include the edge between X_i and X_j (as they are on opposite sides of the cut, so the cut must cross the edge between them). Hence, the value of the cut will give the total assignment cost. This will be minimum when the cut is minimum.

8. Suppose that instead of a single capacity c(u,v) on each edge, each edge has a pair (l(u,v), c(u,v)) and the capacity constraint is changed such that the $l(u,v) <= f(u,v) <= c(u,v)$ (So each edge must carry some minimum flow on it). Can you find a feasible flow on this network? If yes, can you then find the maximum flow?