1. (3 points) Prove or disprove: Let $G$ be a flow network where all the capacities are 1. If $G$ has at least two different integer-valued maximum flows $f$ and $f'$, then there are at least two different minimum cuts $(A, B)$ and $(A', B')$.

**Solution:** This is not true. For example consider the flow network with 4 nodes $s, a, b, t$, and edges $sa, ab, bt, at$, all with capacities 1. Note that there is unique min cut ($\{s\}, \{a, b, t\}$), however we can have the flow $f(sa) = f(ab) = f(bt) = 1$ and $f(at) = 0$, or $f'(sa) = f'(at) = 1$ and $f'(ab) = f'(bt) = 0$.

2. (3 points) We are given an $n \times n$ matrix $A$ with integer entries. We want to select the smallest number of rows and columns (in total) so that after removing these rows and columns all the remaining entries are strictly positive. Show that this problem be solved efficiently.

**Solution:** This is exactly the vertex cover problem in bipartite graphs that we have seen how to solve in the class. Consider nodes $r_1, \ldots, r_n$ representing the rows, and nodes $c_1, \ldots, c_n$ representing columns, and connect $r_i$ to $c_j$ if $A_{ij} \leq 0$. Now we want to remove the smallest set of vertices in this bipartite graph so that all the edges will be deleted.

3. (3 points) Give an efficient algorithm based on network flow techniques to solve the following problem. Justify that your algorithm is correct. The input is an undirected bipartite graph $G = (V, E)$ with parts $X$ and $Y$, where some of the edges are coloured red. We would like to find out whether we can colour more edges red (if necessary) so that every vertex is incident to exactly 10 red edges.
4. (3 points) We are given an \( n \times n \) matrix \( A \) with \( \pm 1 \) entries. We want to rearrange the rows of \( A \) and then the columns of \( A \) to obtain a new matrix \( B \) such that the sum of the diagonal entries of \( B \) is maximized. That is \( \sum_{i=1}^{n} B_{ii} \) is maximized. Show that this problem can be solved efficiently.

**Solution:** Note that every row remains a row (possibly reshuffled) and every column remains a column (possible reshuffled). Suppose that after doing the rearrangements, the initial rows are rearranged to \( \sigma_1, \ldots, \sigma_n \), and the initial columns are rearranged to \( \pi_1, \ldots, \pi_n \). Here \( \sigma_1, \ldots, \sigma_n \) and \( \pi_1, \ldots, \pi_n \) are both reorderings of \( 1, \ldots, n \). Note that

\[
\sum_{i=1}^{n} B_{ii} = \sum_{i=1}^{n} A_{\sigma_i \pi_i},
\]

so we want to select \( \sigma_1, \ldots, \sigma_n \) and \( \pi_1, \ldots, \pi_n \) such that we would have the largest possible number of \(+1\)’s in \( A_{\sigma_1 \pi_1}, \ldots, A_{\sigma_n \pi_n} \). Note that this equivalent to the bipartite matching problem (think of \( \sigma_i \) being matched to \( \pi_i \)). More precisely we construct a bipartite graph with vertices \( u_1, \ldots, u_n \) on the one side representing the rows, and the vertices \( v_1, \ldots, v_n \) on the other side representing the columns, and we put an edge \( u_i v_j \) if \( A_{ij} = +1 \). We solve the bipartite matching problem on this graph and if we find \( k \) edges in the matching, then we can have \( k \) positive numbers among \( A_{\sigma_1 \pi_1}, \ldots, A_{\sigma_n \pi_n} \), and the rest are going to be \(-1\)’s.

More precisely, if the found matching contains an edge \( u_ia v_b \), then we make sure that one of \( \sigma_i \pi_i \) is indeed \( ab \) (for example we can always take \((\sigma_1, \ldots, \sigma_n) = (1, \ldots, n)\), and then we set \( \pi_a = b \). The rest of \( \pi \) can be taken arbitrarily.

5. (3 points) Give an efficient algorithm based on network flow techniques to solve the following problem. Justify that your algorithm is correct.

We are given as input, positive integers \( a_1, \ldots, a_k \) and \( b_1, \ldots, b_t \) such that \( a_1 + \ldots + a_k = b_1 + \ldots + b_t = n \). Let \( G \) be the graph on \( n \) vertices that is formed by taking the disjoint union of \( k \) cliques, on \( a_1, \ldots, a_k \) vertices, respectively. We want to know if we can colour the vertices of \( G \) with \( t \) colours such that the first colour is used \( b_1 \) times, the second colour \( b_2 \) times, etc, so that no two adjacent vertices are coloured with the same colour.

**Solution:** To come up with a solution, this of this as distributing the vertices of each clique into different colours, and each colour can only take certain number of these vertices.

Construct a flow network as in the following. We have a source \( s \) and a sink \( t \). The source is connected to \( k \) vertices \( u_1, \ldots, u_k \) with edges of capacities \( a_1, \ldots, a_k \) respectively. We also have nodes \( v_1, \ldots, v_t \) that are connected to \( t \) with edges of capacities \( b_1, \ldots, b_t \) respectively. Finally connect each \( u_i \) to each \( v_j \) with an edge of capacity 1. Now if MAX FLOW is \( m \) then there is such a colouring. Indeed if there is a flow of 1 on the edge \( u_i v_j \), then it means that we colour
one of the vertices of the $i$-th clique (that has $a_i$ vertices), with colour $j$. Note that colour $j$ is going to be used exactly $b_j$ times (because of the capacity of the edge $a_i b_j$).

6. (3 points) Give an efficient algorithm based on network flow techniques to solve the following problem. Justify that your algorithm is correct. In particular, explain how backup sets and flows correspond to each other.

Input: The coordinates of $n$ antennas on the plane, where the coordinate of the $i$-th antenna is of the form $(x_i, y_i)$ where $x_i$ and $y_i$ are integers.

Output: For each antenna $i$, select a backup set $B_i$ of five other antennas such that the antennas in $B_i$ are all within distances of at most 100 from the antenna $i$, and moreover in total no antenna belongs to more than 10 backup sets.

Solution: We put two nodes $u_i$ and $v_i$ for each antenna $i$, and put an edge $u_i v_j$ if $i$ is a potential backup for $j$. That is if $i \neq j$ and their distance is at most 100. Now we create a source and connect it to each $u_i$ with an edge of capacity 10, and we connect each node $v_j$ to a source $t$ with an edge of capacity 5. Now if the max flow saturates all the edges going to the sink, then it means that we can find the desired back up sets (if $f(u_i v_j) = 1$, then $i$ is in the backup set for $j$).

7. (3 points) Consider a graph where some of the edges have “directions”, but some are undirected edges. We want to assign direction to all the undirected edges such that in the resulting directed graph, the incoming degree of every vertex is equal to its outgoing degree. Note that we are not allowed to change the direction of the edges that have directions in the beginning.

Give an efficient algorithm based on network flow techniques to solve this problem. Justify that your algorithm is correct.

Solution: The idea is that every undirected edge $ab$ can contribute 1 to the indegree of $a$ or $b$, and we want to decide these contributions so that every vertex $a$ will in the end have indegree $\deg(a)/2$, where $\deg(a)$ is the number of edges (directed and undirected) incident to $a$. This is very similar to the baseball elimination problem.

Construct a network flow as in the following. For every vertex $a$ in the graph, put a vertex $v_a$ in the network, and for every edge $ab$ in the graph, put a vertex $v_{ab}$ in the network. If $ab$ has no direction, then add the edges $v_{ab} v_a$ and $v_{ab} v_b$, each with capacity 1 to the network. If $ab$ has already a direction, say from $a$ to $b$, then only add the edge $v_{ab} v_b$, and if the direction is from $b$ to $a$ then add $v_{ab} v_a$ instead. Add a source $s$ and connect it to each vertex $v_{ab}$ with an edge of capacity 1. Add a sink $t$, and for every vertex $v_a$, add the edge $v_a t$ with capacity $\deg(a)/2$.

Solution II: We could also construct the flow network in the following way: For every vertex $v_a$, we add the edge $v_a t$ with the capacity $\deg(a)/2$ minus the current indegree of $a$, and then remove all the initially directed edges from the graph and for undirected edges, we put a vertex $v_{ab}$ in the network and add the edges $v_{ab} v_a$ and $v_{ab} v_b$, each with capacity 1 to the network.

8. (Bonus problem: 4 points) Show that for every network flow, there is always a sequence of augmenting paths that leads to maximum flow, where none of these paths decreases the flow on any of the edges.

Solution: To prove this, we apply the following algorithm to decompose a maximum flow $f$ into a set of flow-paths:

- Repeat the following two steps until there is no such $s, t$-path:
- Find an $s, t$-path $P$ with positive flow, i.e., $f(e) > 0$ for all $e \in P$. 

3
• Let $\Delta$ be the minimum value of the flow $f$ on edges of $P$. Decrease the flow $f$ on each edge $e \in P$ by $\Delta$.

In every iteration the value of $f$ on at least one edge becomes 0. Hence the number of iterations is at most $m$. Using these paths as augmenting paths (with augmenting value $\Delta$) leads to the desired result, and note that these paths as augmenting paths never use the reversed edges, so they never decrease the flow on any edge.