COL 351

TUTORIAL SHEET 10

1. **[KT-Chapter7]** Consider the following problem. You are given a flow network with unit-capacity edges: it consists of a directed graph $G = (V, E)$, a source s and a sink t, and $u_e = 1$ for every edge e. You are also given a parameter k. The goal is delete k edges so as to reduce the maximum $s - t$ flow in G by as much as possible. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in the graph $G_0 = (V, E \setminus F)$ is as small as possible. Give a polynomial-time algorithm to solve this problem.

   Solution: First observe that by removing any k edges in a graph, we reduce the capacity of any cut by at most k, and so, the min-cut will reduce by at most k. Therefore, the max-flow will reduce by at most k. Now we show that one can in fact reduce the max-flow by k. To achieve this, we take a min-cut X and remove k edges going out of it. The capacity of this cut will now become $f - k$, where f is the value of the max-flow. Therefore, the min-cut becomes $f - k$, and so, the max-flow becomes $f - k$.

2. **[KT-Chapter7]** In a standard $s - t$ maximum flow problem, we assume edges have capacities, and there is no limit on how much flow is allowed to flow through a node. In this problem, we consider the variant of the maximum flow and minimum cut problems with node capacities. Let $G = (V, E)$ be a directed graph, with source s, sink t , and non-negative node capacities u_v for each $v \in V$. Given a flow f in this graph, the flow though a node v is defined as $\sum_{e \in \delta^-(v)} f_e$. where $\delta^-(v)$ denotes the edges coming into v. We say that a flow is feasible, if it satisfies the usual flow-conservation constraints and the node-capacity constraint: the flow through a node v cannot exceed c_v. Give a polynomial-time algorithm to find a $s - t$ maximum flow in such node-capacitated network. Define an $s - t$ cut for node-capacitated networks, and show that the analog of the Maximum Flow Min Cut theorem holds true.

   Solution: We construct a new graph H where for every vertex v in G, we have two vertices v_i and v_o. There is an edge from v_i to v_o of capacity c_v. If the graph G has an edge (u, v), then we add an edge (u_o, v_i) in H of infinite capacity. Now notice that any flow in H entering v_i must go through the edge (v_i, v_o) and so the total amount of flow entering v_i (or leaving v_o) cannot exceed c_v. Now it is easy to check that a flow of value v in G results in a flow of the same value in H and vice-versa.

3. Suppose in a directed graph G, there are k edge-disjoint paths from s to t and from t to u. Are there k edge disjoint paths from s to u ?

   Solution: Yes. The proof is easy once we use the max-flow min-cut theorem. Let X be an cut which contains s and does not contain u. What is the capacity of X ? If X does not contain t, then X is an s-t cut, and so, its capacity must be at least k. If

X contains t, then it is a t-u cut, and so, its capacity must be at least k. Thus, any s-u cut has capacity at least k. The max-flow min-cut theorem now shows that there must be k edge disjoint paths from s to u.

4. **[KT-Chapter7]** In sociology, one often studies a graph G in which nodes represent people, and edges represent those who are friends with each other. Let's assume for purposes of this question that friendship is symmetric, so we can consider an undirected graph. Now, suppose we want to study this graph G, looking for a close-knit group of people. One way to formalize this notion would be as follows. For a subset S of nodes let $e(S)$ denote the number of edges in S, i.e., the number of edges that have both ends in S. We define the cohesiveness of S as $e(S)/|S|$. A natural thing to search for would be the set S of people achieving the maximum cohesiveness. Give a polynomial time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness at least α. Give a polynomial time algorithm to find a set S of nodes with maximum cohesiveness.

   Solution: Let $G = (V, E)$. Construct a directed graph H, which has one vertex x_v for every vertex $v \in V$, one vertex y_e for every edge $e \in E$, and two special vertices s and t. We have edges (s, y_e) of capacity 1, (x_v, t) of capacity α. Further, if $e = (u, v)$ is an edge in G, then we have directed edges (y_e, x_u) and (y_e, x_v) of infinite capacity. Now we claim that there is a set S of cohesiveness at least α iff there is an s-t cut in H of capacity at most $|E|$. Let us see why.

   Let A be an s-t cut in H, and A_e denote the vertices of type y_e in A and A_v denote the vertices of type x_v in A. First of all, note that if $y_e \in A$, where $e = (u, v)$, then both $x_u, x_v \in A$. Thus, $e(A_v) \geq |A_e|$. Now, the capacity of this cut is $|E| - |A_e| + |A_v|\alpha \geq |E| - e(A_v) + |A_v|\alpha$. Thus, if min-cut is at most $|E|$, then $|e(A_v)|/|A_v| \geq \alpha$, and so, A_v is the desired set. Conversely, if A_v is a set of vertices of cohesiveness at least α, then consider the cut A consisting of A_v and y_e where e has both end-points in A_v. The capacity of this cut is $|E| - e(A_v) + |A_v|\alpha \leq |E|$, and so, the min-cut is at most $|E|$.

   Finally, we can find the set of maximum cohesiveness by binary search on α. Note that cohesiveness of any set is a fraction of the form $\beta/n!$, where β lies between 0 and $n \cdot n!$. Thus we can perform a binary search on β – time take will be $\log(n!)$ which is $O(n \log n)$.

5. You are given an $n \times n$ matrix A with real entries. You would like to round each of the entries x in the array to either $\lfloor x \rfloor$ or $\lceil x \rceil$ such that the row and column sums don't change. Give an efficient algorithm which either achieves this rounding or declares that no such rounding is possible.

   Solution: First assume that all row sums and column sums are integral. We frame this as a max-flow problem with lower and upper bound on edge capacities. First set up a bipartite graph as follows: the vertices on left hand side, L, correspond to the rows of A, and those in the right hand side, R, correspond to columns of A. For every pair $i \in L, j \in R$, we have a edge from i to j. This edge has capacity in the range $(\lfloor A_{ij} \rfloor, \lceil A_{ij} \rceil)$. Now, we add a vertex s, and edges from s to every vertex in L. The (upper bound) capacity of each such edge is equal to the corresponding row sum of A.

Similarly we add a vertex t, and for every $j \in R$, add an edge (j, t) of capacity equal to the corresponding column sum. Now find a max-flow. If its value is equal to the sum of all the entries in A, then we can have such a rounding (and vice versa).