COL 351

TUTORIAL SHEET 9

1. 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 ehich 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).

2. **[KT-Chapter7]** You have a collection of n software applications, $\{1, \ldots, n\}$, running on an old system; and now you would like to port some of these to the new system. If you move application i to the new system, you expect a net (monetary) benefit of $b_i \geq 0$. The different software applications interact with one another; if applications i and j have extensive interaction, then the you will incur an expense if you move one of i or j to the new system but not both – let's denote this expense by $x_{ij} \geq 0$. So if the situation were really this simple, you would just port all n applications, achieving a total benefit of $\sum_i b_i$. Unfortunately, there's a problem. Due to small but fundamental incompatibilities between the two systems, there's no way to port application 1 to the new system; it will have to remain on the old system. Nevertheless, it might still pay off to port some of the other applications, accruing the associated benefit and incurring the expense of the interaction between applications on different systems. So this is the question: which of the remaining applications, if any, should be moved? Give a polynomial-time algorithm to find a set $S \subseteq \{2, \ldots, n\}$ for which the sum of the benefits minus the expenses of moving the applications in S to the new system is maximized.

   Solution: This can be solved by the min-cut algorithm. We first create an undirected graph as follows. It has a vertex for every software application – call these $v_1, \ldots, v_n$, where v_i corresponds to application i. We also have a special vertex t. If applications i and j have an associated value x_{ij}, then we have an edge between v_i and v_j of capacity x_{ij}. For every vertex v_i, $i \neq 1$, we have an edge (v_i, t) of capacity b_{v_i}. Now we argue that a v_1-t min-cut will give the desired solution. Indeed, let X be such a cut. So, $v_1 \in X, t \notin X$. What is the capacity of X? It is $\sum_{i,j:v_i \in X, v_j \notin X} x_{ij} - \sum_{i:v_i \notin X} b_{v_i} + \sum_v b_v$.

Note that it is exactly the expense minus benefit of moving the applications which are not in the set X (plus a term which is fixed). Thus, finding a min-cut is same as finding the set of applications for which expense minus benefit is minimized, or benefit minus expense is maximized.

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 $O(n \log n)$.