Design and Analysis of Algorithms (CS345A)

Practice-sheet : **Maximum Flow**

Note: This sheet has 11 problems. Do not be afraid of the huge length of the problems. If you read them carefully, you will find them close to real world applications and so very interesting. Also, please please do not look at the hint of a problem until you have spent sufficient time on the problem.

1. (**Flow fundamentals**)

   Suppose you are given a directed graph $G = (V, E)$ with a positive integer capacity c_e on each edge $e \in E$, a designated source $s \in V$, and a designated sink $t \in V$. You are also given an integer maximum (s, t)-flow in G, defined by a flow value f_e on each edge $e \in E$. Now suppose we pick a specific edge $e \in E$ and increase its capacity by one unit. Show how to find a maximum flow in the resulting capacitated graph in time $O(m + n)$, where m is the number of edges in G and n is the number of vertices in G.

   Hint: Work on the residual network. How much time does it take to build it for a given flow f ?

2. (**Blood bank problem**)

   We all know the basic rule for blood donation: A patient of blood group A can receive only blood of group A or O. A patient of blood group B can receive only blood of group A or O. A patient of blood group O can receive only blood of group O. A patient of blood group AB can receive blood of any group.

   Let s_O, s_A,s_B, s_{AB} denote the supply in whole units of the different blood types on hand. Assume that the hospital knows the projected demand for each blood type d_O,d_A,d_B, and d_{AB} for the coming week. Give a polynomial time algorithm to evaluate if the blood on hand would suffice for the projected need. You should formulate this problem as a max-flow problem, establish a relation between the two problems by stating a theorem, and then you should prove the theorem.

   Hint: Form a bipartite graph suitably, add edges and assign them capacities suitably. Add source and sink along with edges and their capacities suitably.

3. (**Mobile phone and base stations**)

   Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. There are n clients with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.

   For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a range-parameter r - a client can only be connected to a base station that is within distance r. There is

also a load parameter L - no more than L clients can be connected to any single base station.

Your goal is to design a polynomial time algorithm for the following problem. Given the position of a set of clients and a set of base stations, as well as the range and load parameters, decide whether every client can be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

Hint: Form a bipartite graph suitably, add edges and assign them capacities suitably. Add source and sink along with edges and their capacities suitably.

4. (**Max-damage to network**)

   You are given a flow network with unit capacity edges: It consists of a directed graph $G = (V, E)$, a source $s \in V$, and a sink $t \in V$; and $c_e = 1$ for every $e \in E$. You are also given a parameter k.

   The goal is to delete k edges so as to reduce the maximum-flow on 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 $G' = (V, E - F)$ is as small as possible subject to this. Give a polynomial time algorithm to solve this problem.

   Hint: For maximum damage, which cut should we pick ?

5. (**unique min-cut**)

   Let $G = (V, E)$ be a directed graph with source $s \in V$, sink $t \in V$ and nonnegative edge capacities $\{c_e\}$. Give a polynomial time algorithm to decide whether G has a unique minimum s-t cut (i.e., an s-t cut of capacity strictly less than that of all other s-t cuts.)

   Hint: Recall the proof we gave for Maxflow-Mincut theorem. Which mincut did we analyse ? What if we carried out analysis from side of t instead of s ?

6. (**Vertex disjoint paths**)

   There is a directed graph $G = (V, E)$ on n vertices and m edges. There are two vertices $s, t \in V$. Two paths from s and t are said to be vertex disjoint if they do not share any vertex except s and t Design a polynomial time algorithm to compute the maximum number of vertex disjoint paths from s to t.

   Hint: Make use of the algorithm for edge-disjoint paths after a suitable modification on G.

7. (**Applicants and jobs with multiple vacancies**)
   There are n applicants $A_1, \ldots, A_n$ and m jobs $J_1, \ldots, J_m$. For each $1 \leq i \leq m$, there is a positive integer v_i which denotes the vacancies in job J_i, that is, J_i can be assigned to at most v_i applicants. You are also given a set S of pairs of applicants and jobs such that a pair $(A_i, J_k) \in S$ means that applicant A_i is eligible for job J_k. Note that a job can be given to an applicant if and only if the applicant is eligible for that job. Moreover, an applicant can be assigned to at most one job. You need to design an

algorithm that assigns jobs to applicants so that the maximum number of applicants get employed. Instead of designing any arbitrary algorithm for this problem and provide its proof of correctness, you must do the following.

(a) Reduce the given problem to max-flow problem on a suitably defined graph. For this clearly describe the corresponding graph (vertices, edges, edge capacities).

(b) State and prove the theorem which connects the instance of the given problem with the instance of the max-flow problem.

Hint: We analysed this problem during bipartite matching. You will have to update capacities of certain edges in the graph we built.

8. (**Circulation with lower bound on flow**)

Recall the circulation problem which we solved by reducing to max-flow problem. We shall now extend this problem further.

There is a flow network $G = (V, E)$ with source, sink $t \in V$ and nonnegative edge capacities $\{c_e\}$. Each vertex v has a demand d_v which is a real number. In addition each edge has a nonnegative number ℓ_e. Design a polynomial time algorithm to determine if there exists a circulation $f : E \to R$ such that

(a) For each vertex v, $f_{in}(v) - f_{out}(v) = d_v$.

(b) For each edge e, $\ell_e \leq f(e) \leq c_e$.

Hint: Reduce this problem to an instance of circulation problem without any lower bound on edges.

9. (**Application with lower bound on flow**)
There is an airline which has to serve n flights per day. Each flight i has four parameters: <origin, destination, departure-time, arrival-time>. One of the biggest factor of running the airline is the number of carriers (airplanes) that it requires to serve all n flights. It can be observed that a single airplane can serve multiple flights. In particular, if there are flights i and j with parameters $< s_i, t_i, d_i, a_i >$ and $< s_j, t_j, d_j, a_j >$ such that $t_i = s_j$ and $a_i < d_j$, then the airplane serving flight i can also serve flight j. In this manner, a single airplane can serve multiple flights. Design a polynomial time algorithm to compute the least number of airplanes needed to serve all n flights.

Hint: The title gives a hint. Form the graph suitably. Try to design algorithm to determine if k number of airplanes are sufficient to serve all the flights, then iterate from $k = n$ to $k = 1$.

10. (**Locating faults in a network**)
You have been called in to help some network administrators diagnose the extent of a failure in their network. The network is designed to carry traffic from a source node s to a target node t, so that we will model the network as a directed graph $G = (V, E)$, in which the capacity of each edge is 1 and in which each node lies on at least one path from s to t.

Now when everything is running smoothly in the network, the maximum s-t flow in G has value k. However, the current situation (and the reason you are here) is that

an attacker has destroyed some of the edges in the network, so that there is now no path from s to t using the remaining (surviving) edges. For reasons that we won't go into here, they believe the attacker has destroyed only k edge, the minimum number needed to separate s from t (i.e., the size of a minimum s-t cut); and we shall assume they are correct in believing this.

The network administrators are running a monitoring tool on node s, which has the following behavior. If you issue the command *ping*(v), for a given node v, it will tell you whether there is currently a path from s to v. (So *ping*(t) reports that no path currently exists; on the other hand, *ping*(s) always reports a path from s to itself.) Since it is not practical to go out and inspect every edge of the network, they would like to determine the extent of the failure using this monitoring tool, through judicious use of the *ping* command.

So here is the problem you face: Give an algorithm that issues a sequence of *ping* commands to various nodes in the network and then reports the full set of nodes that are not currently reachable from s. You could do this by pinging every node in the network, of course, but you would like to do it using many fewer pings (given the assumption that only k edges have been deleted). In issuing this sequence, your algorithm is allowed to decide which node to ping next based on the outcome of earlier ping operations.

Give an algorithm that accomplishes this task using only $O(k \log n)$ pings.

Hint: You will have to make use of a problem we discussed during Lectures under the topic "Applications of maximum flow". You will also have to use binary search. This is one of the most interesting problems.

11. (**Cell phone connectivity**)
You can tell that cellular phones are work in rural communities, from the giant microwave towers you sometimes see sprouting out of corn fields and cow pastures. Let us consider a simplified model of a cellular phone network in a sparsely populated area.

We are given the locations of n base stations specified as points $b_1, \ldots, b_n$ in the plane. We are also given the location of n cellular phones specified as points $p_1, \ldots, p_n$ in the plane. Finally we are given a range parameter $\Delta > 0$. We call the set of cell phone fully connected if it is possible to assign each phone to a base station in such a way that

- Each phone is assigned to different base station, and
- If a phone at p_i is assigned to a base station at b_j, then the straight-line distance between the points P_i and b_j is at most Δ.

Suppose the owner of the cell phone decides to go for a drive, travelling continuously for a total of z units of distance due east. As the cell phone moves, you may have to update the assignment of phones to base stations (possibly several times) in order to keep the set of phones fully connected.

Give a polynomial time algorithm to decide whether it is possible to keep the set of phones fully connected at all times during the travel of this one cell phone. You should assume that all other phones remain stationary during this travel. If it is

possible, you should report a sequence of assignments of phones to base stations that will be sufficient in order to maintain full connectivity; if it is not possible, you should report a point on the travelling phone's path at which the full connectivity can not be maintained.

Hint: You will have to solve multiple instances of bipartite matching problem.