CSc445 Homework #4 Due: Thu Nov 5, 2015 MIDNIGHT

Instructions.

1. Solution could **not** be submitted by students in pairs.
2. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is **easily** readable.
3. If your solution is not clearly written, it might not be graded.
4. Prove the correctness of your answer. A correct answer without a proof might not be accepted.
5. If you have discussed the solution with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually **encouraged**. However, you must write your whole solution yourself.
6. All questions have same weight.
7. You could refer to a data structure studied in class, and just mention briefly their guaranties. For example "*It is known that a Red-Black tree could support the insert/delete/find operations on a set of n elements in time $O(\log n)$.*
8. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If you answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what's the effect (if any) on the running time and on other operations that the algorithm performs.
9. In general, a complete answer should contain the following parts:
   (a) High level description of the data structures (if needed). *E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...*
   (b) High level description of the algorithms
   (c) Proof of correctness (why your algorithm provides what is required).
   (d) A claim about the running time, and a proof showing this claim.

1. Let $G(V, E)$ be a given graph with arbitrary weights on its edges, containing no negative circles. Here $n = |V|$ is an arbitrary large number. Let $s \in V$ be the source vertex. Consider the array $d[1..n]$ used by the Bellman-Ford algorithm. Discuss which conditions should the input graph satisfy, so the following condition holds. After the first iteration of the outer loop of the algorithm (after one pass of all edges) for every vertex v in the graph, $d[v] = \delta(s, v)$.

   You do not need to modify the algorithm itself. Assume that the algorithm scan the edges in the order specifies by the input.

2. Let $G(V, E)$ be a given graph with arbitrary weights on its edges, without negative circles, with a source $s \in V$. Consider the array $d[1..n]$ used by Bellman-Ford algorithm.

   Show an example of a graph with n vertex, for which only after $n-1$ iterations of the outer loop of the Belman-Ford algorithm, it holds that

   $$d[v] = \delta(s, v), \quad \forall v \in V,$$

   but this property does not hold earlier during the algorithm.

3. You are given a set $P = \{p_1 \dots p_n\}$ of points in the plane, given to you by their x and y coordinates. Each points represents the location of an airport. You are also given a a list of edges E, of **pairs** of airports, where (p_i, p_j) is an edge of E if and only if there is a direct flight from p_i to p_j. (Note that not all pairs of airports are connected by an edge). Let $w(p_i, p_j)$ be the cost of this flight. You are also given $s, t \in P$. Assume s is the leftmost airport (most to the West) and t is the rightmost airport.

   We say that a route $s \to t$ is **monotone** if it consists of airports that are connected by edges of E, and when following this route, we move monotonically from left to right (West to East).

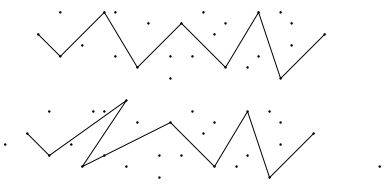

Figure 1: Top: A monotone path. Bottom: A path which is not monotone. The dots represents airports.

   Suggest an algorithm that in time $O(m + n \log n)$ finds the cheapest monotone route from s to t.

   Do not worry about the timing of the flights.

4. You are given **two** identical glass cups, and an access to skyscraper building. You have access to each window in each floor of the skyscraper. Your goal is to experimentally find what is the the highest floor K_{max} that such a cup could survive being dropped

from this floor to the ground. That is, a drop from the $(K_{max}+1)$'th floor would break this cup.

Assume that only the height influences whether the cup would survive the fall. So for example if it survived a fall from the 7'th floor, it would also survive a drop from any lower floor.

Note that if the cup survived, you have to go down to the ground level to fetch it. Suggest an algorithm for finding $K_{\max}$, while the number of times the surviving cup need to be fetched is only $O(\sqrt{n})$, where n is the number of floors in the skyscraper.

...and no, a broken cup could not be fixed, you could not buy new cups, etc.

5. Let $G(V,E)$ be a network, let $p(u,v)$ be a flow on this network (defined on every pair $u,v \in V$) and let $q(u,v)$ be another flow on this network (defined on every pair $u,v \in V$). We define

$$p'(u,v) = p(u,v) + q(u,v), \qquad \forall u,v \in V.$$

which of the constraints of a flow are satisfies by $p'(u,v)$, and which might be violated? Prove your answer. Refer to the slides about network flow to see the list of constrains.

6. Let $G(V,E)$ be a network, let $p(u,v) \geq 0$ be a legit flow on this network (defined on every pair $u,v \in V$). For some $u,v \in V$ such that $(u,v) \notin E$ and $(v,u) \notin E$. What is the flow $p(u,v)$?

7. Let $G(V,E)$ be a network, and let

$$\pi = (e_1, e_2 \ldots e_k)$$

be a path that lead from the source $s \in V$ to the sink $t \in V$. Here each e_i is an edge in E. Assume $p(u,v)$ is a legit positive flow on G. Let $\delta > 0$ be a small positive value. We change the positive flow $p(\cdot,\cdot)$ by setting $p'(u,v) = p(u,v) + \delta$ for every edge $e_i \in \pi$. Which of the constraints of flow are satisfies by $p'(u,v)$, and which might be violated ? Prove your answer.

8. Let $G(V,E)$ be an undirected bipartite graph. That is, V is the union of two disjoint sets A and B (given to you explicitly), and each edge of E connects a vertex of A to a vertex of B. Also assume that you have an access to an algorithm that finds a flow with maximum value on this network. Furthermore, if the capacities are all integer, the flow on each edge is an integer as well.

   Explain (and prove) how to use this algorithm to find a maximum-cadinality set $M \subseteq E$ of edges such that all the following conditions hold:

   (a) Each $b \in B$ is adjacent to ≤ 4 edges of M

   (b) Each $a \in A$ is adjacent to ≤ 2 edges of M

   (c) $|M|$ is as large as possible.

   You could assume that you know which vertex belongs to A, and which belongs to B.

9. Assume a set S of students are standing in the rectangle bounded by Campbell Av, Park Av, 6'th street and Speedway Blvd. There are many of them, and their locations are arbitrary, fixed, and could not be changed once teams are formed.

   Your goal is to design an algorithm that partition the students into *as many teams as possible*. Let L denote some length – say 100 yards. The rules for forming teams are

   (a) Each student could belong to at most one team

   (b) Students are not allowed to move.

   (c) In each team, one member should be on Speedway Blvd, and one on 6'th street.

   (d) In each team, each member s (excluding the ones on Speedway and 6'th street) has two other team-members s', s'' whose distance from s is $\leq L$, and in addition, s' is to the South of s and s'' is North of s.

   In other words, each team forms a *chain* of students, starting at Speedway Blvd, ends at 6'th street, and distances between consecutive members are all $\leq L$.

   Ignore locations of building — assume students could be placed.

   Assume you have access to an algorithm for finding maximum flow in a network. Show how you could use this algorithm for solving our problem. Furthermore, if the capacities are all integers, the flow on each edge is an integer as well.