ESO207A: Data Structures and Algorithms	
Homework *4a: Shortest Paths*	Due Date: 6 November 2017

Instructions.

1. Start each problem on a new sheet. For each problem, Write your name, Roll No., the problem number, the date and the names of any students with whom you collaborated.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A clear description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks.*

Problem 1. *CLRS 24.3-6* Given a directed weighted graph $G = (V, E)$, where each edge $(u, v) \in E$ has an associated value $r(u, v)$ which is a real number $0 \leq r(u, v) \leq 1$ that represents the *reliability* of the communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail and we assume that these probabilities are independent. Give an $O((|V| + |E|) \log(|V|)$ time algorithm to find the most reliable path from a given source vertex to all the other vertices in V. Argue the modelling of your problem carefully and hence the complexity of your algorithm. (You do not have to restate or redo any algorithm already done in the class).

Let $P = v_0, v_1, \ldots, v_k$ be a path from u to v such that $v_0 = u$ and $v_k = v$. (The path may be simple or not). Given the axioms of probability, the reliability of this path is

$$r(P) = r(v_0, v_1, \ldots, v_k) = \prod_{i=1}^{k} r(v_{i-1}, v_i)$$

Define $g(P) = -\ln r(P)$. Then, by taking ln on both sides, we have,

$$g(P) = \sum_{i=1}^{k} g(v_{i-1}, v_i)$$

Given a source vertex s and destination vertex v, the problem is to find the most reliable path from s to v. That is, we need,

$$\arg \max_{P \text{ is a path from } s \text{ to } v} r(P)$$

Since, the ln function is monotonic, the above answer is the same as replacing $r(P)$ by $-\ln r(P)$ and replacing the max operator by min operator. Thus, we are interested in

$$\arg \min_{P \text{ is a path from } s \text{ to } v} g(P)$$

Now $g(P)$ is the sum of the $g(r(e))$ for edges e on the path. Hence, we can define the edge-weights of every edge $e = (u, v) \in E$ as

$$w(u, v) = g(r(u, v)) = -\ln r(u, v)$$

Since, $0 \leq r(u, v) \leq 1$, we have, $0 \leq w(u, v) \leq \infty$, that is, edge weights are non-negative. Hence, we can solve the problem of finding the single source most reliable paths to all reachable vertices by using Dijkstra's algorithm for shortest paths in directed graphs using $w(u, v) = -\ln r(u, v)$ as the edge weights. This runs in time $O((|V| + |E|) \log |V|)$ time.

Problem 2. Let $G = (V, E)$ be a given weighted, directed graph with non-negative edge weights. You are given a table of claimed shortest path distances $u.d$ and predecessor $u.\pi$ for each $u \in V$. Can you check in $O(|V| + |E|)$ time whether the d and π values correspond to some shortest path tree? Is the condition of non-negative edge weights really necessary?

Assume $G = (V, E)$ be a weighted directed graph with non-negative edges. Consider the following algorithm that attempts to relax all edges in E once in some order. Find a vertex s with $s.\pi = NIL$, and call it the source vertex. We will design an algorithm as a test with two steps, if both steps pass, then we can claim that the d and π values correspond to some shortest path tree.

Step 1: Follow the edges of G_π and check if the relaxation is tight, that is, do the following

```
1.  for each vertex v ∈ V − {s}
2.      u = v.π
3.      if v.d ≠ u.d + w(u, v)
4.          return "Relaxation is not tight"
5.  return "Relaxation is tight"
```

Suppose the algorithm returns "*Relaxation is not tight*". Then, there is an edge $(u, v) \in E$ and it is given that $u = v.\pi$ and $v.d \neq u.d + w(u, v)$. If $v.d > u.d + w(u, v)$, then assuming $u.d$ is a correct shortest path distance from s to u, then, this shortest path to u followed by edge (u, v) is a path from s to v and so $v.d \leq u.d + w(u, v)$, which is a contradiction. Otherwise, suppose $v.d < u.d + w(u, v)$. Again, assuming d attributes are correct shortest distances, this means, the shortest path to v is *not* (the shortest path to u, followed by the edge (u, v)). Hence, u cannot be $v.\pi$ in the given distances and the shortest path tree.

Conversely, the tightness of relaxation also shows the following obvious property: $\delta(s, v) \leq v.d$, for every vertex v. The proof is by induction on the number of hops separating v from s in G_π.

Step 2: Suppose we run a BFS from s and each time an edge (u, v) is explored, check the relaxation condition, that is, $v.d \leq u.d + w(u, v)$. If $v.d > u.d + w(u, v)$, i.e., an improvement in $v.d$ is now possible as $u.d + w(u, v)$, then clearly, the given distances, either $u.d$ or $v.d$, or both, is not the shortest path distance. The algorithm returns "*Incorrect Relaxation*" in this case. If this case does not occur until the termination of BFS, the algorithm returns "*Relaxation is consistent*" .

1. s is the unique vertex such that $s.\pi = NIL$
2. **if** $s.d \neq 0$ **return** FALSE
3. Run BFS from s
4. As an edge (u, v) is visited
5. **if** $v.d > u.d + w(u, v)$
6. **return** "*Triangle Inequality fails*"
7. **return** "*Triangle inequality holds*"

We have to show that when the algorithm returns "*Triangle Inequality holds*" , then the given table of values of $u.d$ and $u.\pi$ for each vertex $u \in V$, is the shortest path distance and forms a shortest path predecessor tree respectively.

We will first show that the d values correctly indicate the shortest path distance to each vertex from s. Let $G_{\pi'}$ be a shortest path tree produced by (conceptually) running Dijkstra's algorithm on the given graph. Suppose for sake of contradiction that there is a vertex v such that $v.d \neq \delta(s, v)$. Among all such vertices v with $v.d \neq \delta(s, v)$, choose a vertex v that is closest to s in terms of the number of hops from s in $G_{\pi'}$.

Clearly, v is not s, since, $s.d = 0$ and $\delta(s, s) = 0$. Let $u = v.\pi'$, that is, u is the parent of v in the graph. Therefore, $u.d = \delta(s, u)$. Hence, during the BFS of s, at some point the vertex u was visited and then the edge (u, v) was visited, and therefore, the algorithm found $v.d \leq u.d + w(u, v)$. However, since (u, v) is a tree edge in $G_{\pi'}$, $\delta(s, v) = u.d + w(u, v)$. Hence, $v.d \leq \delta(s, v)$. Thus, $v.d = \delta(s, v)$ (since we have from step 1 that $v.d \geq \delta(s, v)$). This contradicts the assumption that $v.d \neq \delta(s, v)$. In other words, for all vertices v reachable from s, $v.d = \delta(s, v)$.

So the d values correctly indicate the shortest path distance to each vertex from s. That the edges of G_π are correct follow from the tightness of the relaxation, namely, Step 1.

Step 2 can be simplified by replacing BFS by a single pass of checking relaxation for each edge $e \in E$ in some order.

1. **for** each edge $e = (u, v) \in E$ do
2. **if** $v.d > u.d + w(u, v)$
3. **return** FALSE
4. **return** TRUE

Does this work for graphs with negative edge-weights? Yes, if there are no negative cycles. The proof goes through without change.

Problem 3. *CLRS 24.1-4* Modify the Bellman-Ford algorithm so that it sets $v.d$ to $-\infty$ for all vertices v for which there is a negative-weight cycle on some path from the source to v.

Let C be a negative edge weight cycle that is reachable from s the source vertex. Then there exists at least one edge of C such that the triangle inequality is violated. Because, if not, we have

$$v.d \leq u.d + w(u, v), \quad \text{for each edge } (u, v) \in C$$

Now sum over all the edges of the cycle C. This gives

$$\sum_{(u,v)\in C} v.d \leq \sum_{(u,v)\in C} u.d + \sum_{(u,v)\in C} w(u, v)$$

Since, we are summing along a cycle C, every vertex appears exactly once as source of an edge and destination of its previous edge (u, v). Hence, $\sum_{(u,v)\in C} v.d = \sum_{(u,v)\in C} u.d$ leaving us with

$$0 \leq \sum_{(u,v)\in C} w(u, v)$$

that is, C is a non-negative weight cycle. But this is a contradiction since we assume C to be a negative weight cycle.

Thus, for each negative weight cycle, there is at least one edge (u, v) that violates triangle inequality. So, after the $|V| - 1$ passes of Bellman-Ford, make an additional pass over the edges, checking for triangle inequality and do a DFS (reachability) from each vertex u such that (u, v) violates triangle inequality, setting $w.d = -\infty$ for each vertex w reachable from u.

Note: Clearly, all negative cycles reachable from the source vertex are discovered, since, one vertex of each such negative cycle is discovered. DFS from this vertex discovers the remainder of the cycle and all the vertices reachable from it.

Note 2. Suppose (u, v) fails the test for triangle inequality. Then, it must be the case that u is reachable from the source s via a negative cycle. Why? If not, then the argument of Bellman-Ford would show that both $u.d$ and $v.d$ equal $\delta(s, u)$ and $\delta(s, v)$ respectively. Hence, it is possible for u to be not on a negative cycle, but reachable via a negative cycle, and (u, v) violating the triangle inequality.

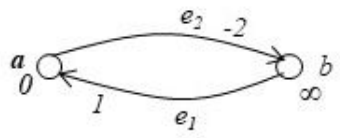
Problem 4. *CLRS 24.1-6* Given a weighted, directed graph $G = (V, E)$ with a negative weight cycle. Give an efficient algorithm ($O(|V|\,|E|)$ time) to list the vertices of *one such cycle*. Prove that your algorithm is correct.

The Bellman-Ford algorithm runs $|V| - 1$ passes of relaxing all edges (in some order). Instead run $|V|$ passes of relaxing all edges, as follows.

```
1.  INITIALIZE_SINGLE_SOURCE(G, s)
2.  repeat |V| times
3.      for all edges e = (u, v) ∈ E
4.          if v.d > u.d + w(u, v)
5.              v.d = u.d + w(u, v)
6.              v.π = u
```

Now, we obtain the graph G_π defined by the predecessor edges $v.\pi$, for all vertices v for which $v.d$ is finite. Run DFS on G_π. If G_π is detected as acyclic, then s has no vertex reachable via a negative cycle. Otherwise, if G_π has a cycle, then that cycle is a negative weight cycle.

```
1.  Perform DFS on G_π consisting of set of vertices V_π = {u : u.d is finite }.
2.      if there is no backedge found, then return "No negative cycle reachable"
3.      else the backedge (v, u) together with stack vertices from u till v
            forms a negative wt. cycle
```



a is source vertex.
Initially $a.d = 0$, $b.d = \infty$
$a.\pi = NIL$, $b.\pi = NIL$
Bellman-Ford makes 2-1 = 1 pass over the edges in the order:

Relax(e_1) : No change to d or π values.

Relax(e_2): $b.d = -2$, $b.\pi = a$

The graph G_π

Now make one more pass over edges:

Relax(e_1) : $a.d = -2 + 1 = -1$. $a.\pi = b$
(a cycle appears in G_π

Relax(e_2): $b.d = -3$, $b.\pi = a$

The graph G_π

Figure 1: Running Bellman-Ford with $|V| - 1$ passes of relaxation may not give a cycle in G_π.

Remarks 1. If we run Bellman-Ford with $|V| - 1$ passes of relaxation, then, G_π may not have a cycle, see Figure 1.

To prove correctness, we will show the following two properties.

1. Suppose G_π has a cycle. Then, this cycle is a negative weight cycle.
2. Suppose G_π is acyclic. Then, G has no negative weight cycle reachable from s.

.

We first prove (1). Let $(v_0, v_1, v_2, \ldots, v_k = v_0)$ be a cycle C in G_π. Say that the relaxation of edge (u, v) *causes an update* if the condition $v.d > u.d + w(u, v)$ is true and this triggers the relaxation update: $v.d = u.d + w(u.v); v.\pi = u$.

In the sequence of relaxations executed by the algorithm, let the edge (v_{k-1}, v_k) be the last edge whose relaxation caused an update among all the edges of the cycle C. Note that this can be assumed without loss of generality (by renumbering the indices of the vertices of the cycle). Let t_j be the last instant at which the relaxation of the edge (v_{j-1}, v_j) caused an update, for $j = 1, 2, \ldots, k$. By assumption, $t_k > t_j$ for $j = 1, 2, \ldots, k-1$. Then, just after this instant t_j, when the relaxation update step is completed, we have, $v_j.d = v_{j-1}.d + w(v_{j-1}, v_j)$. At all time instants subsequent to t_j, $v_j.d$ does not change. However, it may be the case that $v_{j-1}.d$ changes after t_j (that is, $t_{j-1} > t_j$), and if so, $v_{j-1}.d$ can only reduce. Hence, at the instant just before t_k, we have,

$$v_j.d \geq v_{j-1}.d + w(v_{j-1}, v_j), \quad j = 1, 2, \ldots, k-1$$

Now, consider the instant just before the instant t_k. At this time,

$$v_k.d > v_{k-1}.d + w(v_{k-1}, v_k)$$

and the relaxation step is yet to be updated. Then, summing the inequations, we have,

$$\sum_{j=1}^{k} v_j.d > \sum_{j=1}^{k} v_{j-1}.d + \sum_{j=1}^{k} w(v_{j-1}, v_j)$$

Since, $\sum_{j=1}^{k} v_j.d = \sum_{j=1}^{k} v_{j-1}.d$, we have,

$$0 > \sum_{j=1}^{k} w(v_{j-1}, v_j) \ .$$

Hence, C is a negative weight cycle. This proves property (1).

We now consider property (2). Suppose G has a negative cycle $C = (v_0, v_1, \ldots, v_k = v_0)$ reachable from s. Then, there exists an edge (v_{i-1}, v_i) of this cycle that violates triangle inequality.Why? Suppose all edges satisfy triangle inequality, that is, $v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$, for $i = 1, 2, \ldots, k$. Summing over i, we get,

$$\sum_{i=1}^{k} v_i.d \leq \sum_{i=1}^{d} v_{i-1}.d + \sum_{i=1}^{d} w(v_{i-1}, v_i)$$

or that

$$\sum_{i=1}^{d} w(v_{i-1}, v_i) \geq 0$$

that is, the cycle C is a non-negative weight cycle. This is a contradiction.

This means the shortest path from s to v_i is via a negative cycle. Following the π pointers from v_i, that is, $v_i.\pi, v_i.\pi.\pi, \ldots$, is cyclic. In other words G_π has a cycle.