Design and Analysis of Algorithms

Practice-sheet: **Shortest Paths**

1. (**Ponder over it ...**) Let $G = (V, E)$ be a directed graph with real edge weights but no negative cycle. Suppose $L : V \rightarrow R$ be a mapping satisfying the following equalities and inequalities.

   (a) $L(s) = 0$

   (b) For each $v \in V \backslash \{s\}$,

   - $L(v) \leq L(u) + wt(u, v)$ for each edge (u, v)
   - $L(v) = L(u) + wt(u, v)$ for at least one edge (u, v).

   Prove that $L(v) = \delta(s, v)$, i.e., $L(v)$ stores the distance from s to each vertex v in the graph.

2. (**Making Bellman-Ford algorithm space efficient ?**)

   Recall from the lecture on all-pairs shortest paths that we could reduce the space required by overwriting on the same distance matrix. Can we do it for Bellman-Ford algorithm as well ? In other words, consider the following algorithm.

```
L(s) ← 0;
for v ∈ V\{s} do
    L(v) ← ∞
end
for i = 1 to n − 1 do
    foreach (u, v) ∈ E do
        if L(u) + wt(u, v) < L(v) then
            L(v) ← L(u) + wt(u, v)
        end
    end
end
```

   Does $L(v)$ store distance from s to v in the graph at the end of the above algorithm ? Prove or give a counterexample.

3. (**Retrieving shortest path from the Bellman-Ford algorithm**)

   Recall the Bellman-Ford algorithm discussed in the class. Augment the algorithm suitably to compute a data structure of $O(n)$ size using which we can report the shortest path from source s to any vertex v efficiently (in time of the order of the number of edges in the shortest path from s to v).

4. (**Is Bellman-Ford algorithm really needed ?**)

   Consider the problem of shortest paths in directed graph with real weights but no negative cycle. An attentive student argues that there is no need of Bellman-Ford algorithm. We could just transform the given graph with some negative edges into a graph where all edge weights are positive and then apply Dijkstra's algorithm. The transformation is the following :

   Let w be the weight of the least weight negative edge in the given graph. Add $|w|$ to the weight of each edge. Let G' be the resulting graph.

   The student claims that shortest path from s to v in G is identical to the shortest path from s to v in G' for each vertex v. Do you agree with the student ? Either provide a proof or provide a counterexample.

5. (**Correct or Incorrect ?**)

   Which of the following statements is (are) correct ?

   (a) For every directed graph, Dijkstra's algorithm will fail to compute distance from a given source if there is any negative edge present in the graph.

   (b) There exist a directed graph with some negative edge weights such that Dijkstra's algorithm will fail to compute the distance from a given source in the graph.

   (c) There exists a directed graph with some negative edge weights on which Bellman-Ford algorithm may fail to compute distance from a given source vertex.

6. (**Verifying the distances from source ?**)

   Let $G = (V, E)$ be a directed graph with real edge weights but no negative cycle. Let $s \in V$ be a designated source vertex in the graph. You are given an array A storing some real numbers. We need to verify whether $A[i]$ is the distance from s to i for each $i \in V$ in the graph. Design the most efficient algorithm for this task..

Only for fun (not for the exam)

Refer to Problem 4 in this practice sheet. Though it was a failed attempt to make the edge weights non-negative while preserving the shortest paths from the source, one still feels that there might be an alternate way to make edge weights non-negative and still preserving the shortest paths from the source. Indeed, there is such a way. In fact, it leads to an $O(mn + n^2 \log n)$ time algorithm for all-pairs shortest paths in directed graphs with real edge weights but no negative cycle. Notive that this is much better than Floyd Warshal algorithm. The following is the sketch of this brilliant algorithm. Fill in the details suitably. Without loss of generality, assume that the graph is strongly connected.

1. Pick any arbitrary vertex s. Execute Bellman-Ford algorithm to compute distance from s.
2. Using the distance from s, transform the edge weights of the graph so that each edge weight is non-negative. Most importantly, the shortest path in the original graph are still the shortest paths in the transformed graph.
3. Execute Dijkstra's algorithm from each $v \in V \backslash \{s\}$.