ASSIGNMENT 7

DUE: Wednesday November 13, 5 PM. NOTE that Crowdmark will list the due date as Friday November 15 but that ONLY applies if you are using a grace credit for this assignment. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies. When you are asked to design an algorithm you should: (1) describe the idea of your algorithm clearly in English; (2) give pseudocode; (3) argue correctness; (4) and analyze the run time.

1. [12 marks] Shortest simple path: Given a directed graph with weights on the edges and given vertices $s$ and $t$, find the minimum weight of a simple path from $s$ to $t$. Recall that a simple path is a path that does not repeat vertices.

In fact, this problem is NP-hard—no one knows a polynomial time algorithm for it.

(i) [2 marks] If $G$ has no negative-weight cycle then the Bellman-Ford algorithm will solve this problem. Give an example to show that the Bellman-Ford algorithm does not solve the shortest simple path problem if there are negative-weight cycles. Show step-by-step how the algorithm works on your example.

Use the following version of Bellman–Ford. It is the space-saving version with the addition of the line to return $d[t]$.

```
Algorithm 1: BellmanFord($G = (V,E), s, t, w$)

for all $v \in V$ do $d[v] \leftarrow \infty$;
$d[s] \leftarrow 0$;
for $i = 1, 2, 3, \ldots, n - 1$ do
    for all $(u, v) \in E$ do
        if $d[u] + w(u,v) < d[v]$ then
            $d[v] \leftarrow d[u] + w(u,v)$;
    return $d[t]$;
```

(ii) [10 marks] Design a branch-and-bound algorithm that solves the shortest simple path problem. Each configuration should include a simple path $s, u_1, u_2, \ldots, u_i$. You may also wish to store the weight of the path. To generate the children of this configuration, consider all edges leaving $u_i$.

Programming Assignment 2 will be to implement your algorithm.
2. [12 marks] More paths and cycles

(i) [6 marks] **Shortest paths with the most edges.** Consider a directed graph $G$ with non-negative edge weights. There may be more than one shortest path from $s$ to $t$ in $G$. Define the **Byzantine shortest path** from $s$ to $t$ to be a shortest path from $s$ to $t$ that has the maximum number of edges. Modify Dijkstra’s algorithm to find Byzantine shortest paths from a given source vertex $s$ to all vertices $v$ in $G$. Your algorithm should compute for each vertex $v \neq s$:

- $d(v) = \text{the minimum weight of a path from } s \text{ to } v,$
- $\ell(v) = \text{the maximum number of edges in a path from } s \text{ to } v \text{ of weight } d(v),$ and
- $\text{parent}(v) = \text{the vertex } u \text{ such that a Byzantine shortest path from } s \text{ to } v \text{ ends with the edge } (u, v).$

(ii) [6 marks] **Detecting negative-weight cycles.** The All Pairs Shortest Path Algorithm (or “Floyd–Warshall Algorithm”) takes as input a directed graph $G$ on $n$ vertices and returns an $n \times n$ matrix $D$. Pseudocode is given below for the non-space-saving version of the algorithm on vertex set $\{v_1, v_2, \ldots, v_n\}$. If $G$ has no negative-weight cycles then the output has the property that $D[u, v]$ is the minimum weight of a path from vertex $u$ to vertex $v$.

Suppose we run the algorithm on a graph $G$ that may have negative-weight cycles. Give a simple $O(n)$ time algorithm to test the output matrix $D$ to determine if $G$ has a negative-weight cycle. Note that you should not modify the algorithm. Prove your answer.

You may use these two results without proof:

- If $D[u, v] = k$ then there is a walk from $u$ to $v$ of weight $k$. (This is easy to prove by induction.)
- For our purposes it does not matter whether repeated vertices are allowed or not in a “cycle”—more precisely, if a directed graph $G$ has a walk from $u$ to $u$ of negative-weight then $G$ has a simple cycle of negative-weight.

<table>
<thead>
<tr>
<th>Algorithm 2: AllPairsShortestPaths($G = (V, E), w$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for all</strong> $u, v \in V \times V$ do <strong>$D_0[u, v]$</strong> $\leftarrow \infty;$</td>
</tr>
<tr>
<td><strong>for all</strong> $(u, v) \in E$ do <strong>$D_0[u, v]$</strong> $\leftarrow w(u, v);$</td>
</tr>
<tr>
<td><strong>for all</strong> $u \in V$ do <strong>$D_0[u, u]$</strong> $\leftarrow 0;$</td>
</tr>
<tr>
<td><strong>for</strong> $i = 1, 2, 3, \ldots, n$ do</td>
</tr>
<tr>
<td><strong>for all</strong> $v \in V$ do</td>
</tr>
</tbody>
</table>
| **$D_i[u, v]$** $\leftarrow D_{i-1}[u, v];$
| **if** $D_{i-1}[u, v_i] + D_{i-1}[v_i, v] < D_i[u, v]$ **then** |
| **$D_i[u, v]$** $\leftarrow D_{i-1}[u, v_i] + D_{i-1}[v_i, v];$
| **$D \leftarrow D_n;$** |