

Spencer Shimko  
CMSC 441  
HW 10

**1. 24.1-4 (Make your modifications as efficient as possible) Modify the Bellman-Ford algorithm so that it sets  $d[v]$  to  $-\infty$  for all vertices  $v$  for which there is a negative-weight cycle on some path from the source to  $v$ .**

For this modified algorithm we start by calling Initialize-Single-Source( $G, s$ ) [defined on pg. 585]. This initializes vertex attributes  $d[v]$  (upper weight bound) to  $+\infty$  (except for  $d[s]$  which is set to 0) and  $\pi[v]$  (predecessor) to nil for each vertex in  $G$ . We then make  $|V| - 1$  iterations through the graph each time relaxing each edge of the graph once. Then the path is checked for negative-weight cycles and all vertices that result in such a cycle are marked  $d[v] = -\infty$ . Now, only some of the vertices on the path from the source to  $v$  have been marked accordingly. Now, we could either perform a search that returns a path order result (BFS) and mark each predecessors of vertices who's  $d[v] = -\infty$  to also be  $-\infty$ . However, if we are using the book's Relax and Initialize-Single-Source algorithm's we have already built our path. Remember  $\pi[v]$ ? For each vertex who's  $d[v] = -\infty$  we can follow it's predecessor trail and set all of it's predecessor's  $d[v]$  to  $-\infty$  as well. An additional recursive function should be written to handle this in a efficient fashion. This function should not recurse if the vertex's predecessor  $d[v]$  to  $-\infty$  (efficiency).

(Note: the original algorithm was copied from page 588 in CLRS)

```
Bellman-Ford-New( $G, w, s$ )
Initialize-Single-Source( $G, s$ )
for  $i <- 1$  to  $|V[G]| - 1$ 
    do for each edge  $(u, v) \in E[G]$ 
        do Relax[ $u, v, w$ ]
for each edge  $(u, v) \in E[G]$ 
    do if  $d[v] > d[u] + w(u, v)$ 
         $d[v] = -\infty$ 
for each  $v$  such that  $d[v] = -\infty$ 
    do Follow-And-Mark-Pred( $v$ )
```

```
Follow-And-Mark-Pred( $v$ )
if  $\pi[v] \neq \text{nil}$  and  $d[\pi[v]] \neq -\infty$ 
do  $d[\pi[v]] = -\infty$ 
    Follow-And-Mark-Pred( $\pi[v]$ )
else
```

```
return
```

## 2. 24.3-4

We are given a directed graph  $G = (V, E)$  on which each edge  $(u, v) \in E$  has an associated value  $r(u, v)$ , which is a real number in the range  $0 \leq r(u, v) \leq 1$  that represents the reliability of a communication channel from vertex  $u$  to vertex  $v$ . We interpret  $r(u, v)$  as the probability that a channel from  $u$  to  $v$  will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between the two given vertices.

This problem is similar to that of the standard shortest path problems with one major difference, we are trying to maximize the function  $r(u, v)$  instead of minimize the function  $w(u, v)$ . So a simple way to convert this problem to a shortest path problem is take the inverse of the reliability value of an edge as the weight:

```
w(u, v) = 1 / r(u, v)
```

The problem can now be easily solved using Dijkstra's algorithm (note that since  $0 \leq r \leq 1$  we won't have negative weights). The most reliable connection between any two vertices  $(u, v) \in E$  is the shortest path as indicated by Dijkstra's algorithm. The reliability of any edge along this path is  $1/d[v]$  (remember we took the inverse of the reliability function  $r$  as the weight).

```
w(u, v)
  return (1 / r(u, v))
```

```
Dijkstra(G, w, s)
Initialize-Single-Source(G, s)
S <- nil
Q <- V[G]
while Q != nil
  do u <- Extract-Min(Q)
  S <- S ∪ {u}
  for each vertex v ∈ Adj[u]
    do Relax(u, v, w)
```

## 3. 24.3-7

Let  $G = (V, E)$  be a weighted, directed graph with weight function  $w : E \rightarrow \{0, 1, \dots, W\}$  for some nonnegative integer  $W$ . Modify Dijkstra's algorithm to compute the shortest paths from a given source vertex  $s$  in  $O((V + E) \lg W)$  time. (Hint: How many distinct shortest-path estimates can there be in  $V - S$  at any point in time?)

Dijkstra's algorithm (restated in problem 3 above) can be implemented using a min-priority queue with a min-heap (pg. 599). This results in Extract-Min and Decrease-Key operations having  $O(\lg V)$ . There are at most  $|V|$  Extract-Min operations and  $|E|$  Decrease-Key operations. This yields a runtime of  $O((V + E) \lg V)$ .

Following the hint, there are at most  $W$  shortest-path estimates in  $V - S$  at any time. This can be seen: any extracted vertex  $v$  will have neighbors  $n$ . Each neighbor's shortest-path estimate ( $d[n]$ ) is such that  $d[v] \leq d[n] \leq d[v] + W - 1$  (since  $w: E \rightarrow \{0, 1, \dots, W\}$ ). If we use a binary heap to store for storage (instead of a min-priority queue with a min heap) we can achieve  $|E|$  Decrease-Key operations in  $O(\lg W)$ .  $|V|$  Extract-Min operations can be performed in the same runtime. Remembering that there is at most  $W$  estimates in the queue we can plainly see that the total runtime is  $O((V + E) \lg W)$ .