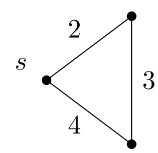
COSC 302: Analysis of Algorithms Lecture — Spring 2018
Prof. Darren Strash
Colgate University

Worksheet 8 — Single-Source Shortest Paths (with Solutions)

1. Give a weighted graph and a source vertex s such that the Prim-Jarník algorithm and Dijkstra's algorithm give different spanning subtrees when run from s.



2. Suppose you are given a shortest-paths tree from a source vertex s, which is stored in a predecessor array π. Give pseudocode to construct a shortest path from s to a given vertex v.

Algorithm 1 Construct shortest path.

proc CONSTRUCT-PATH(v, π)

path $= v$
while $\pi[v] \neq$ NIL
    path $= \pi[v]+$ path
    $v = \pi[v]$
return path

3. **(Previous exam question)** In this problem we will solve the minimum spanning tree problem with edge weight constraints. Suppose you are given a connected, undirected, weighted graph $G = (V, E, w)$.

   (a) Suppose that all edge weights are 3. Describe an algorithm to compute a minimum spanning tree in G in $O(V + E)$ time. Argue for, but do not formally prove, correctness of your algorithm.

   (To be discussed in exam review.)

   (b) Suppose that all edge weights are now 1 or 0. Describe an algorithm to compute a minimum spanning tree in G in $O(V+E)$ time. Formally prove that your algorithm is correct.

   (To be discussed in exam review.)

4. The Bellman-Ford algorithm may relax many edges that do not result in new shortest paths. For example, on graphs consisting of a single simple path $v_1 v_2 \cdots v_n$, Bellman-Ford will take $\Theta(n^2)$ time. Describe a change to Bellman-Ford that reduces the number of unnecessary edge relaxations. Your algorithm should have running time $O(n)$ on simple paths. What is the running time of your algorithm on arbitrary graphs?

   Solution: Intuitively, an edge (v, x) only needs to be relaxed if the value $d[x]$ changes. Then, similar to BFS, we can create a queue of vertices whose $d[\cdot]$ values have changed. However, we also need to make sure that we do not add multiple copies of a vertex to the queue.

Algorithm 2 Compute single-source shortest paths tree.

```
proc Bellman-Ford(G, s)
Q = ∅
inQueue[1..n] = [false, false, ..., false]
Q.enqueue(s)
inQueue[s] = true
while not Q.empty()
    v = Q.dequeue()
    inQueue[v] = false
    for x ∈ N(v)
        if d[x] > d[v] + w(v, x) then
            d[x] = d[v] + w(v, x)
            Q.enqueue(x)
            inQueue[x] = true
```

The running time is still $O(nm)$.