I have read and agree to the collaboration policy. Lynne Diep.

Lynne Diep
Homework Heavy Grading
Collaborators: Sabrina

Homework 3-4
Due: May 22, 2017

a. Algorithm -

Initialize s = 0, all other vertices are set to ∞
start with smallest weight edge from source
Update vertex distances along this path until relaxing doesn't return a smaller weight
main a queue of all unrelaxed vertices → dequeue when relaxed
when finished, go to lowest weight unrelaxed vertex in queue
Update vertex distances along this path until relaxing doesn't return a smaller weight
continue until unrelaxed queue is empty (no more unrelaxed vertices)

Runtime: The time to sort |E| edges by weight is O(E log E), and the algorithm is based off of dijkstra which has a run time of O(v). Each of the passes take O(E) time. Thus the total runtime is O(E log E + V + E) = O(V + E log E).

Proof of Correctness:
Invariant: for each node in graph, d(u) is the length of shortest path
Base Case: S is set of nodes in which shortest path has already been found
S = 1
d(S) = 0
Induction Hypothesis: assume invariant is true for |s| = k ≥ 1

- Let V be next node added to S and let (u,v) be the chosen edge
- The shortest s-u path plus (u,v) is a s-v path of length $\pi(v)$
- Consider any s-v path p. We'll see that its no shorter than $\pi(v)$ ← original inequality for ours include property for y-v because the path always has increasing edges weights, y-v will be longer
- Let x-y be the first edge in p that leaves S, and let p' be the subpath to x

$$p' + (x,y) \text{ length} \underset{\text{(IH)}}{\geq} d(x) + l(x,y) \underset{\text{(def.)}}{\geq} \pi(y) \underset{\text{(our algorithm chose v instead of y)}}{\geq} \pi(v)$$

Thus finding the shortest path.

b. There are many cases, and each has their own algorithm
CASES:

1. all increasing sequences – same algorithm for part a
2. all decreasing sequences – same algorithm for part a but in decreasing order of weights

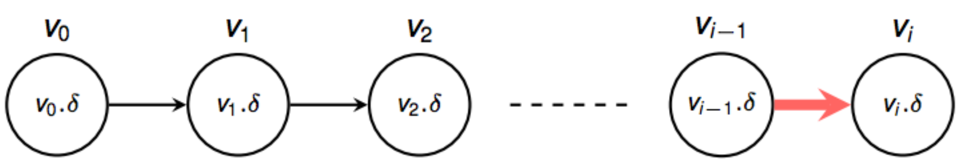
3. increase then decrease sequences – run part a algorithm, then run part a algorithm in decreasing order of weights
4. decrease then increase sequences – run part a algorithm in decreasing order of weights then run part a algorithm normally

Correctness:
We ensure the correctness by the path-relaxation property with the unique edge weights. The path-relaxation property states -

Path-Relaxation Property (Lemma 24.15)

If $p = (v_0, v_1, \ldots, v_k)$ is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = v_k.\delta$ (regardless of the order of other relaxation steps).

Proof:
By induction on i, $0 \le i \le k$
After the ith edge of p is relaxed, $v_i.d = v_i.\delta$
For i = 0, we have s.d = s. δ = 0 by initialization
The value of s.d never changes after that
Inductive step: $i - 1 \rightarrow i$
Assume $v_{i-1}.d = v_{i-1}.\,\delta$ and relax (v_{i-1}, v_i)



Thus the correctness of the algorithm.

Runtime: O(V + E log E)
The time to sort |E| edges by weight is O(E log E), and the algorithm is based off of dijkstra which has a run time of O(v). Each of the passes take O(E) time. Thus the total runtime is O(E log E + V + E) = O(V + E log E).