Single Source Shortest Paths Guide
By Cole Frederick

Use the book to help you answer questions in this guide. These questions are NOT graded – they are for your study purposes only.

- Goal: build a shortest paths tree rooted at start node (like BFS)
- Reduction from Single Destination Shortest Paths
- Covers Single Pair shortest path
- Optimal substructure of a shortest path

Cycles?
- Positive
- Zero weight
- Negative

“Relaxing” edges
- Consider edge (u, v)
  - If current shortest distance to u, plus weight of edge is < current shortest distance to v
  - Update distance to v (and record u as v’s new predecessor)

Bellman-Ford Algorithm

Steps:
- Relax every edge, that’s one “round”
- Do a total of n rounds, you’ll either have a shortest paths tree or a negative cycle

Why it works:
After the first round, we know the shortest paths to all nodes of length one or zero. After the second, we know the shortest paths of length two, one or zero. Etc. How long could a cycle-less path be? ____________. After that many rounds (if there are no negative cycles) we’ll have the correct shortest paths tree. Do one more round, if any distances change (any edge relaxes), there must be a negative cycle.

Notes:
- Order of relaxation?
- Strategy of algorithm?
Shortest Paths in a DAG

See order of relaxation above*

Steps:
- Topologically sort, then consider nodes in sorted order
- For each node, relax all outgoing edges

Why does it work?
- ___________________________________

What's the runtime?
- ___________________________________

Dijkstra’s Algorithm

Correct only for graphs with non-negative edge weights

Similarities to Prim’s:
- Data structure to organize vertices: __________________________
- Strategy of algorithm: ____________________________

Differences from Prim’s:
- Key in priority queue represents
  - In Prim’s: __________________________________________
  - In Dijkstra’s: _________________________________________

Runtime: