Subtask B → Dijkstra : Pre compute and store

Minimum priority queue

Use treeset

Q.extractMin()

Q.decreaseKey(v)

Subtask C:

Need to have a new graph that captures the path (Common technique for graph questions)

Special case 3 since the transformed new is DAG

Augmenting the graph\*\*

1. Shortest path but all edges have the same value: BFS, DFS
2. Shortest Path but edges have value either 0 or 1: Instead of pq, use normal queue and put all the zero in front and all the one behind.
3. Shortest path and you can teleport to a node 2 edges away, all edges have weight = 0: Use new graph that contains old neighbours and node 2 edges away then use DFS, BFS on new graph
4. Shortest path but you have a lot of starting sources: Build a new graph with only 1 source → This new source connects to all the old sources with edge weight 0 then run Dijkstra on this new graph
5. Shortest path but you have a lot of destinations: Build a new graph with only 1 destination → Similar to point 4
6. Shortest path but the cost of an edge doubles every even move: Have new graph like subtask C but even move will double the weight

Bellman’s ford

* Relax all edges V-1 times
* Can we stop algorithm pre-emptively? Yes. Stop if no relax at all
* Dijkstra relax the smallest estimate to the source but bellman’s ford relax edges randomly.

So worst case for bellman’s ford is e.g 0 <- 1 <- 2 graph and it relax (1,0) then (2,1), it relaxes in reverse direction from dest to start

Online quiz 2:

1. Cayley’s formula