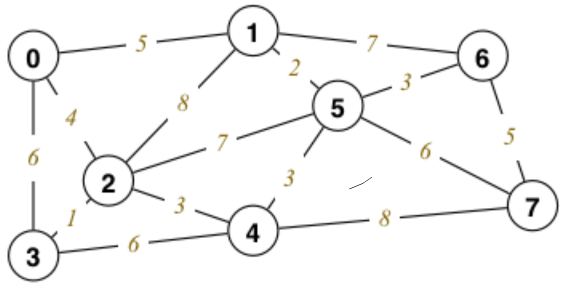
Q4.



Vertices	Distances	Predecessors
~~0~~, 1, 2, 3, 4, 5, 6, 7	0, ∞, ∞, ∞, ∞, ∞, ∞, ∞	−1, −1, −1, −1, −1, −1, −1, −1
1, ~~2~~, 3, 4, 5, 6, 7	0, 5, 4, 6, ∞, ∞, ∞, ∞	−1, 0, 0, 0, −1, −1, −1, −1
~~1~~, 3, 4, 5, 6, 7	0, 5, 4, 5, 7, 11, ∞, ∞	−1, 0, 0, 2, 2, 2, −1, −1
~~3~~, 4, 5, 6, 7	0, 5, 4, 5, 7, 7, 12, ∞	−1, 0, 0, 2, 2, 1, 1, −1
4, ~~5~~, 6, 7	0, 5, 4, 5, 7, 7, 12, ∞	−1, 0, 0, 2, 2, 1, 1, −1
~~4~~, 6, 7	0, 5, 4, 5, 7, 7, 10, 13	−1, 0, 0, 2, 2, 1, 5, 5
~~6~~, 7	0, 5, 4, 5, 7, 7, 10, 13	−1, 0, 0, 2, 2, 1, 5, 5
~~7~~	0, 5, 4, 5, 7, 7, 10, 13	−1, 0, 0, 2, 2, 1, 5, 5
$\emptyset$	0, 5, 4, 5, 7, 7, 10, 13	−1, 0, 0, 2, 2, 1, 5, 5

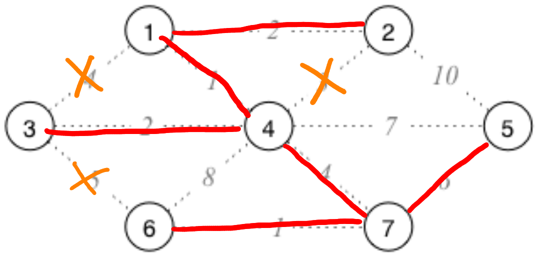
These are the minimum distances from 0 to every other vertex.

We use this to reconstruct the shortest paths.

Implicit assumption: all edge weights are positive.
You need a different algorithm if your graph has negative edge weights $\longrightarrow$ Bellman-Ford

(this isn't assessable!)

Q 5.



Iteration	Candidates	Choice
1	1→4, 6→7	1→4
2	6→7	6→7
3	1→2, 3→4	3→4
4	1→2	1→2
5	cycle ~~2→4~~, cycle ~~1→3~~, 4→7	4→7
6	cycle ~~3→6~~, 5→7	5→7.

Kruskal: pick smallest edge that avoids cycles at each step.

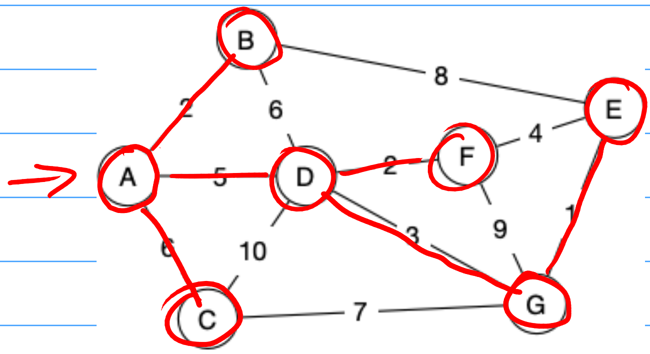
Unique edges considered: 9 (6 picked + 3 rejected)

For a graph G(V, E)

- Min #. of edges considered: V − 1
- Max #. of edges considered: E

What extra edge could force the worst case?

Q6.



Iteration	Candidates	Choice
1	AB(2), AC(6), AD(5)	AB
2	AC(6), AD(5), BE(8)	AD
3	AC(6), BE(8), CD(10), DF(2), DG(3)	DF
4	AC(6), BE(8), CD(10), DG(3), EF(4), FG(9)	DG
5	AC(6), BE(8), CD(10), EF(4), CG(7), EG(1)	EG
6	AC(6), CD(10), CG(7)	AC

Prim: Add smallest "loosely connected" edge at each step.