Created by Matthew for CS2040 Tutorial Group 01

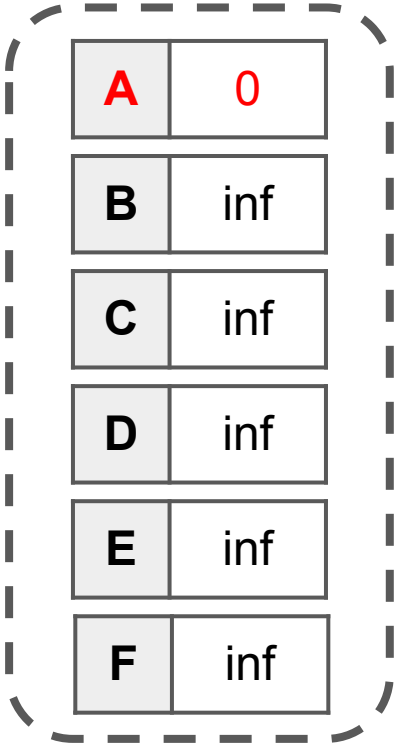
Test your implementation
here

Dijkstra's Algorithm

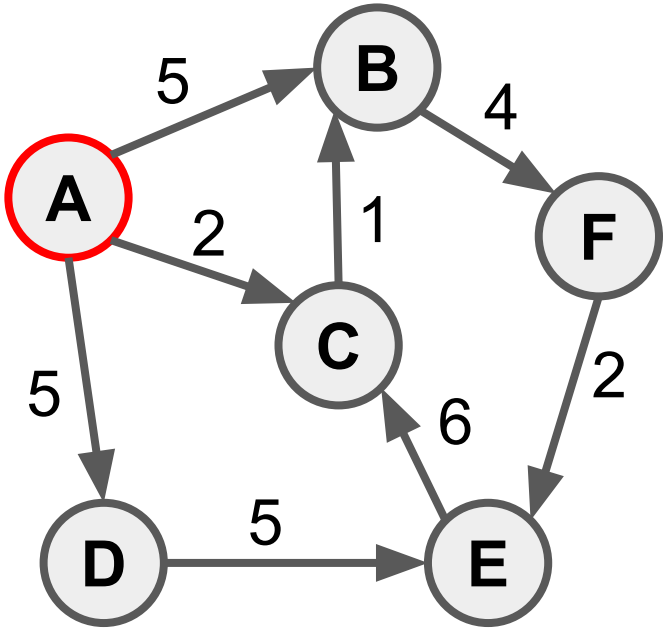
Single Source Shortest Path, Non-Negative Edges

Push all nodes into a priority queue with initial distance of infinity

The source node (A) will have distance 0



Priority Queue



Distance

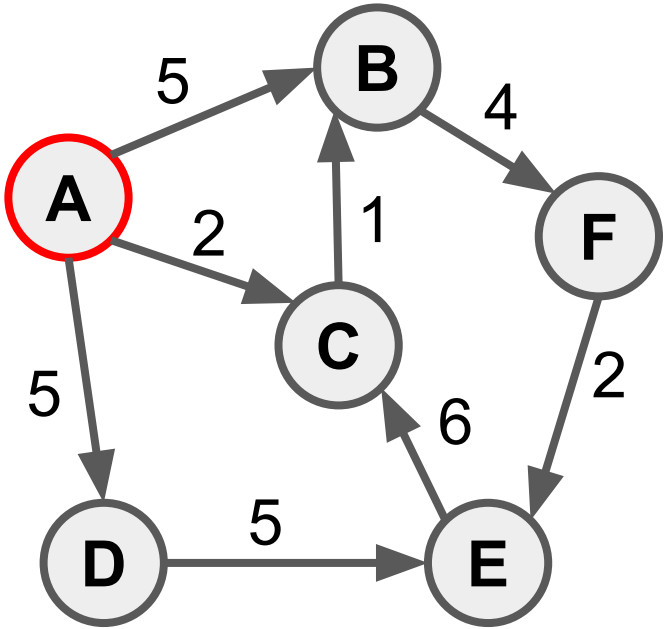
A	B	C	D	E	F
inf	inf	inf	inf	inf	inf

A	0

While priority queue is not empty, remove the cheapest distance and update the distance table

B	inf
C	inf
D	inf
E	inf
F	inf

Priority Queue



Only touch the distance table upon removal of pairs

Distance

A	B	C	D	E	F
0	inf	inf	inf	inf	inf

A	0

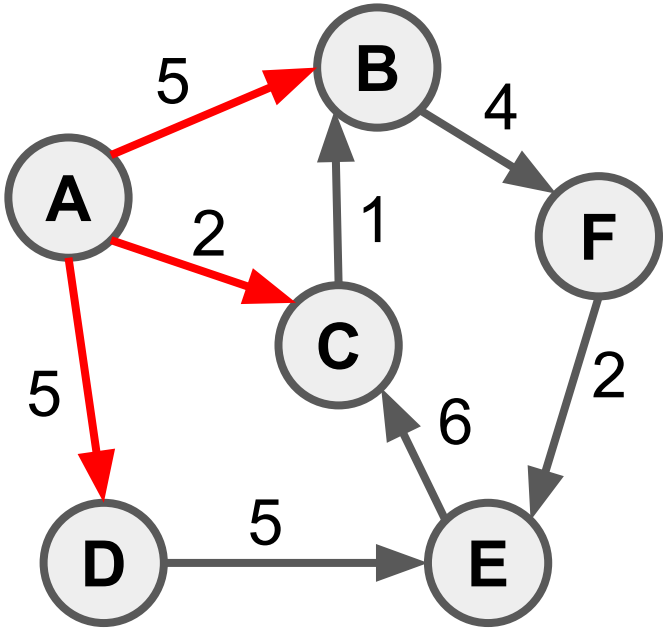
Relax all of A's neighbour (if applicable)

Eg. dist(A) + 5 < dist(B)

dist(B) is relaxed to 5

B	5
C	2
D	5
E	inf
F	inf

Priority Queue



Do not touch the distance table in this stage

Distance

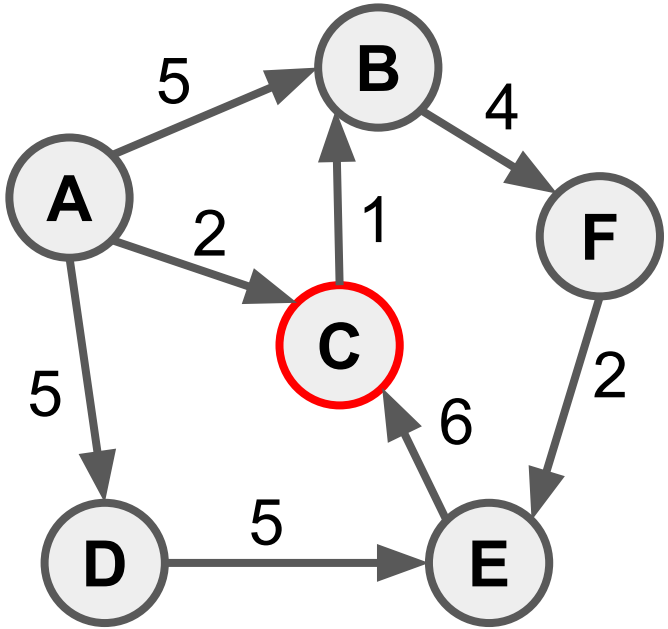
A	B	C	D	E	F
0	inf	inf	inf	inf	inf

C	2

Done. Remove the next cheapest node, C, and update the distance table

B	5
D	5
E	inf
F	inf

Priority Queue



Distance

A	B	C	D	E	F
0	inf	2	inf	inf	inf

Created by Matthew for CS2040 Tutorial Group 01

C	2

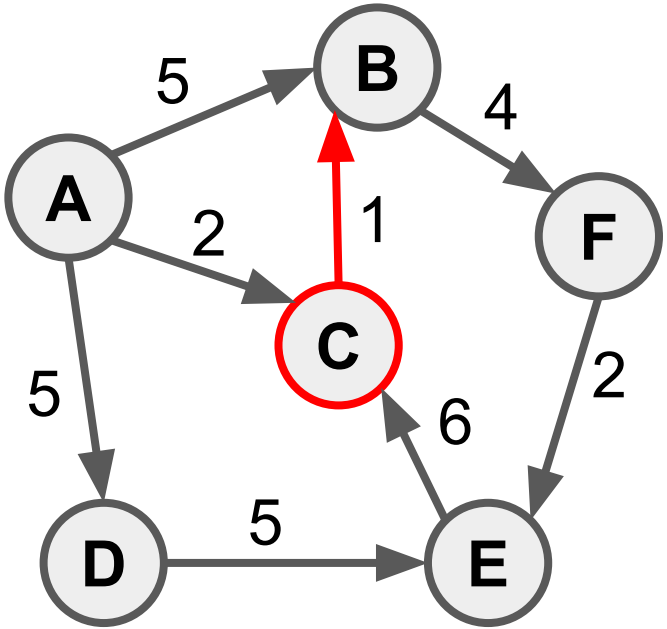
Relax all of C's neighbours

Eg. dist(C) + 1 < dist(B)

dist(B) is relaxed to 3

B	3
D	5
E	inf
F	inf

Priority Queue



Distance

A	B	C	D	E	F
0	inf	2	inf	inf	inf

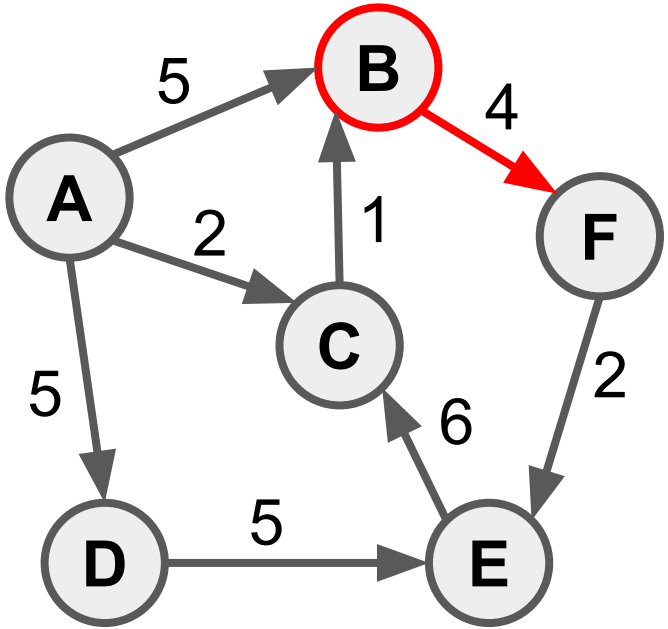
Created by Matthew for CS2040 Tutorial Group 01

B	3

Remove B, relax F

D	5
E	inf
F	7

Priority Queue



Distance

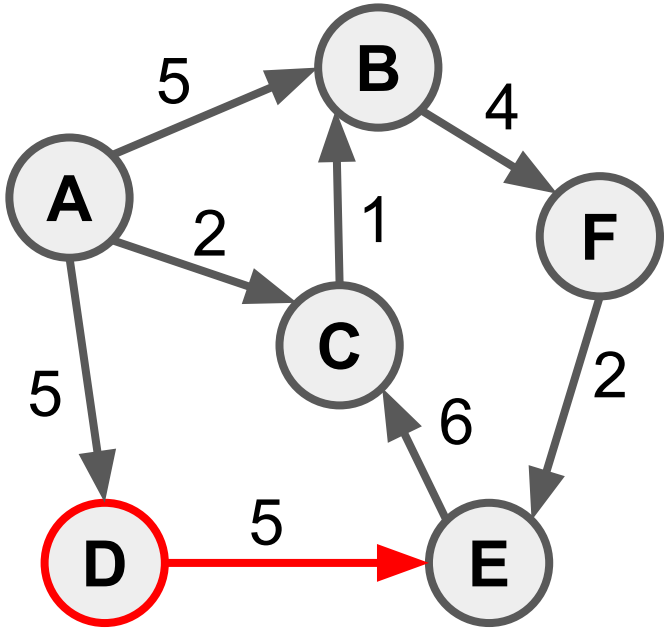
A	B	C	D	E	F
0	3	2	inf	inf	inf

D	5

Remove D, relax E

E	10
F	7

Priority Queue



Distance

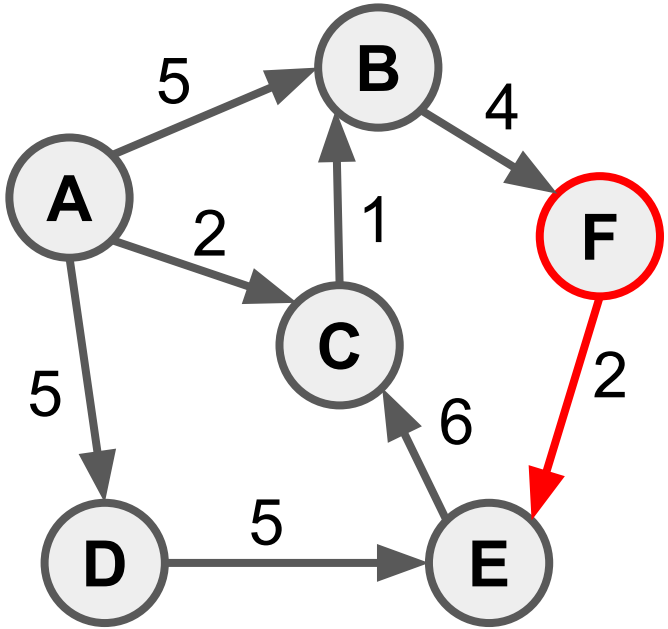
A	B	C	D	E	F
0	3	2	5	inf	inf

F	7

Remove F, relax E

E	9

Priority Queue



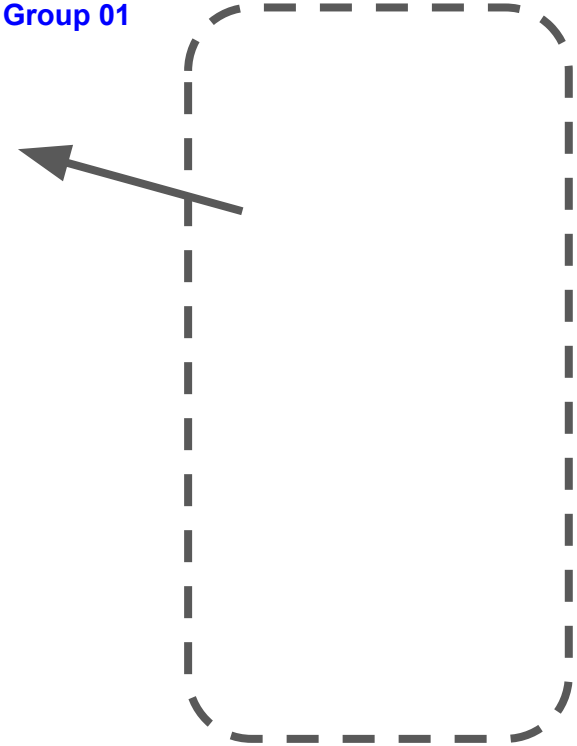
Distance

A	B	C	D	E	F
0	3	2	5	inf	7

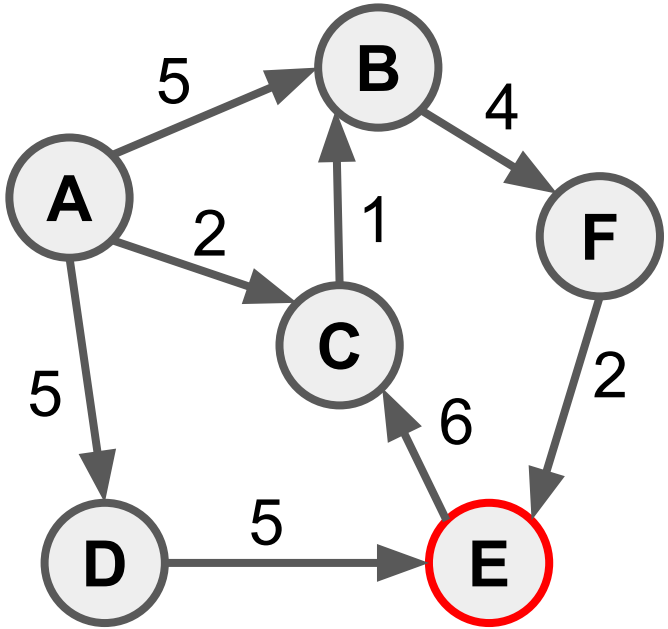
Created by Matthew for CS2040 Tutorial Group 01

E	9

Remove E



Priority Queue



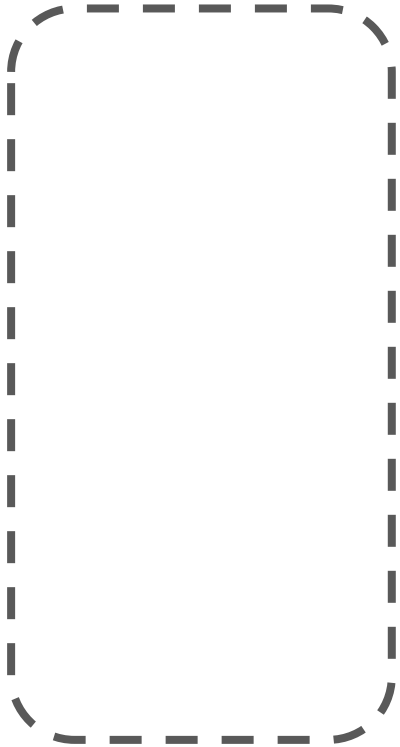
Distance

A	B	C	D	E	F
0	3	2	5	9	7

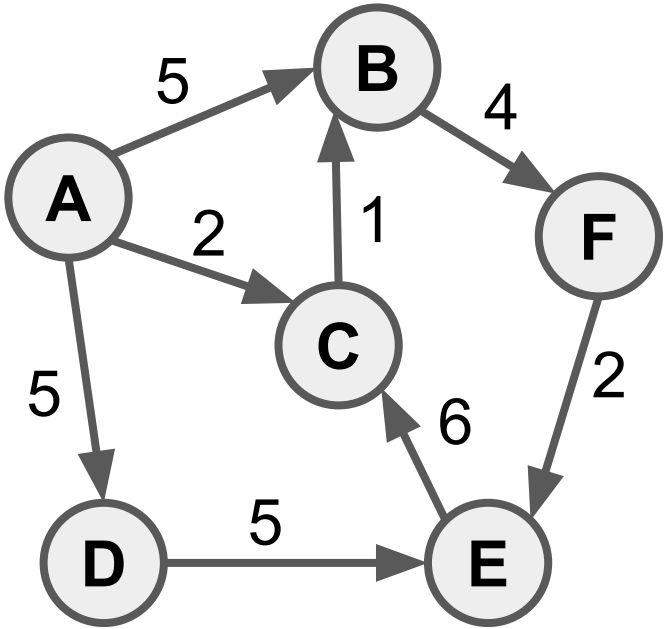
Created by Matthew for CS2040 Tutorial Group 01

Priority Queue is empty.

Algorithm is completed.



Priority Queue



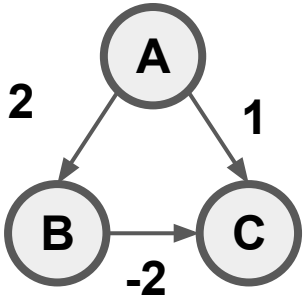
Distance

A	B	C	D	E	F
0	3	2	5	9	7

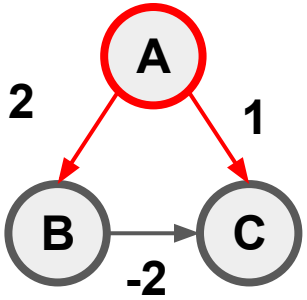
Dijkstra's Algorithm on Graphs with Negative Edges

Don't do it

Wrong Answer

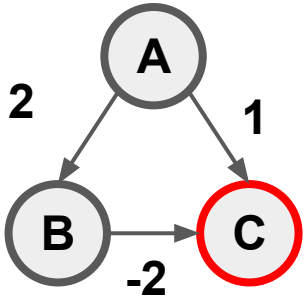


A	0
B	∞
C	∞



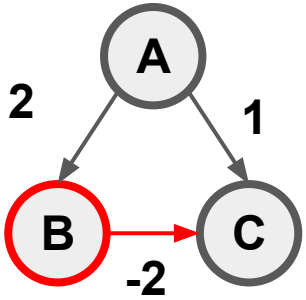
A	0
B	2
C	1

A's distance is correct



A	0
B	2
C	1

C's distance is correct



A	0
B	2
C	1

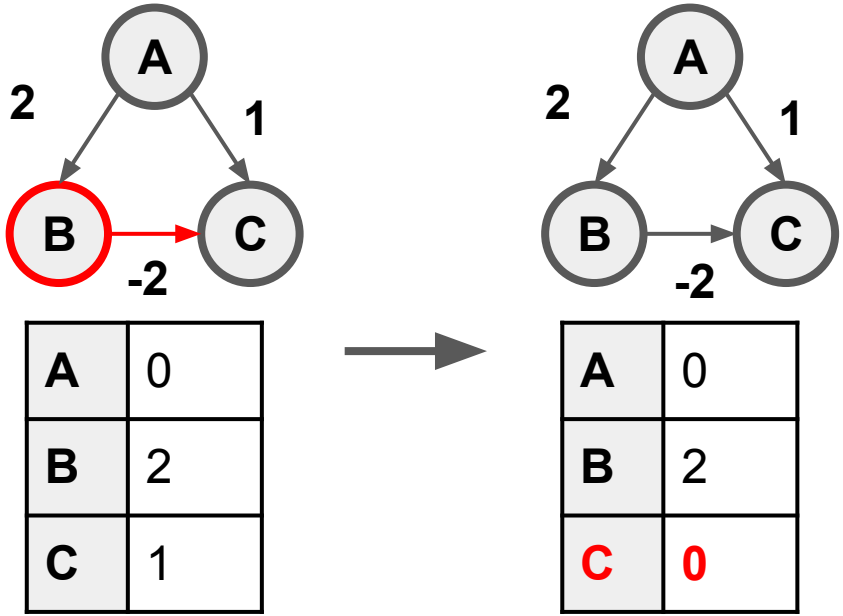
B's distance is correct

C's distance is assumed to be correct and will not be changed

Modified Dijkstra: “Patching” Original Dijkstra

Claim is: if the node can be further relaxed, but it has already been removed from the priority queue earlier, **re-insert it into the priority queue**

C can be further relaxed.
Relax it further and re-insert it into the priority queue

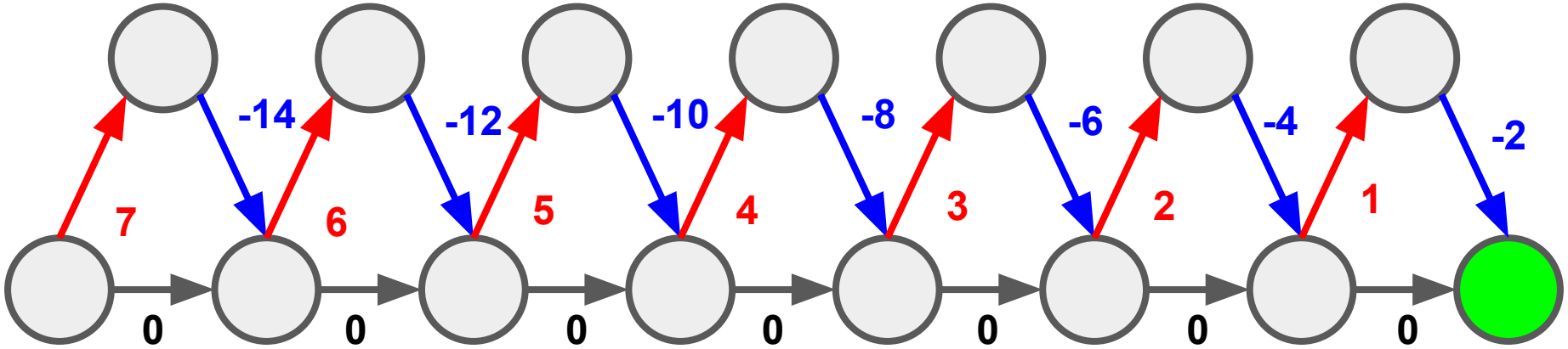


A	0
B	2
C	1

A	0
B	2
C	0

Modified Dijkstra “Killer”

Worst case time complexity: ~ O(2 ^ N)
Worse than Bellman Ford…
Just don’t do it...



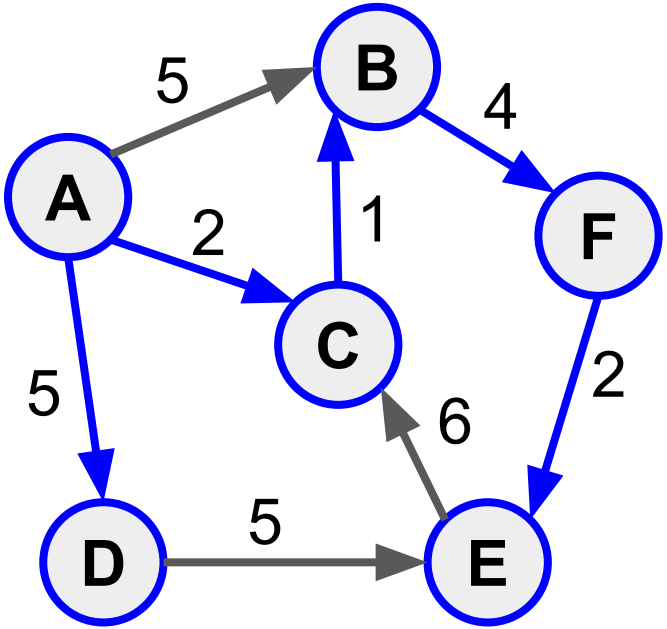
This node alone will be reinserted ~ 2 ^ 7 times

Created by Matthew for CS2040 Tutorial Group 01

Dijkstra Spanning Tree

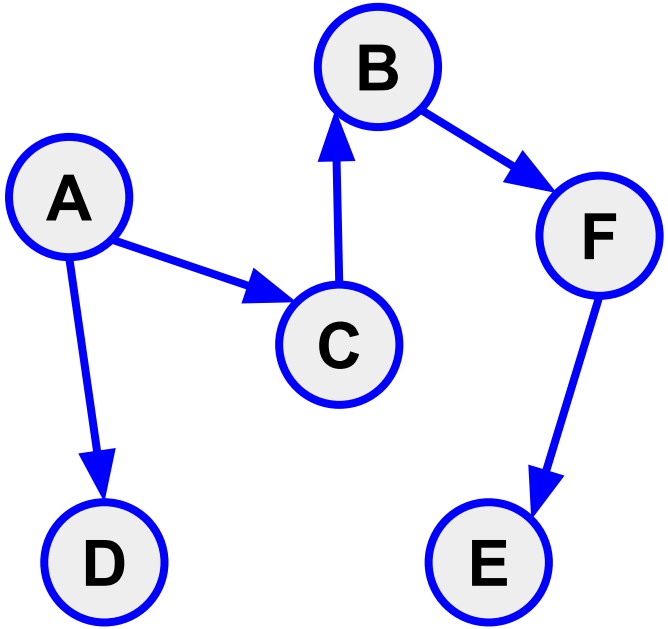
Dijkstra Spanning Tree

If we keep track of the predecessors of every node (the person who discovered that node with a lowest cost), we get a *Dijkstra Spanning Tree*



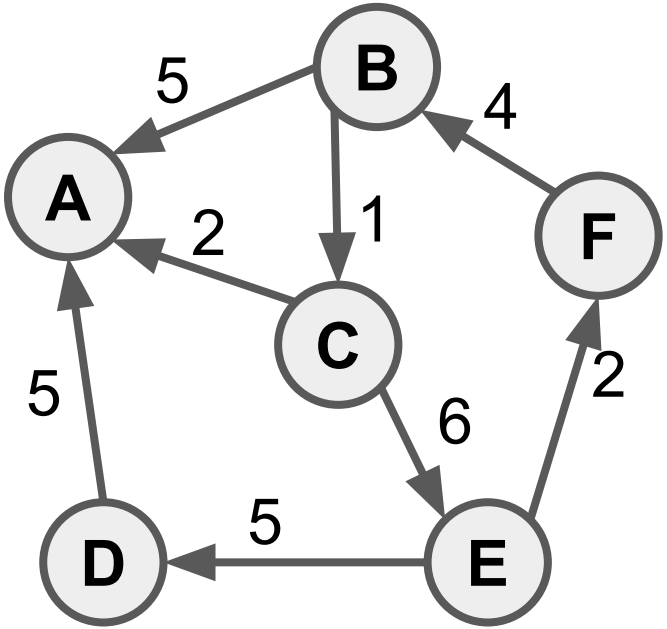
Dijkstra Spanning Tree

If we keep track of the predecessors of every node (the person who discovered that node with a lowest cost), we get a *Dijkstra Spanning Tree*



Application

Your house is located at node A. You will be dropped off at a random node. For every node, determine the next node you should move to in order to get to your house as quickly as possible? (shortest distance)

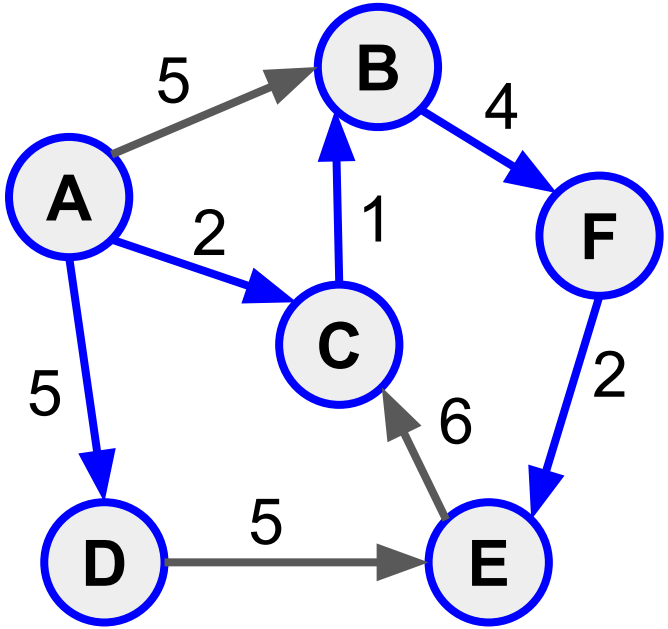


Application

Solution

Transpose the graph (Reverse all edges)

Find the Dijkstra Spanning Tree



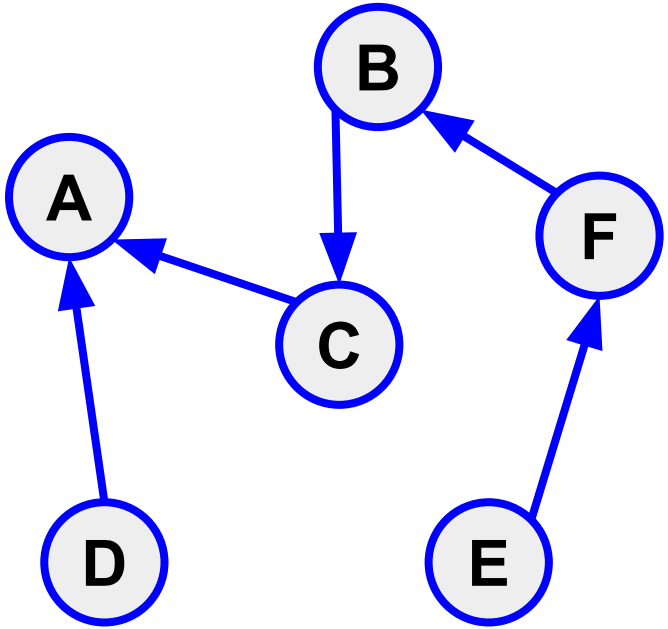
Application

Solution

Transpose the graph (Reverse all edges)

Find the Dijkstra Spanning Tree

The selected edges corresponds to the direction to travel to in the original graph



Meet in the Middle

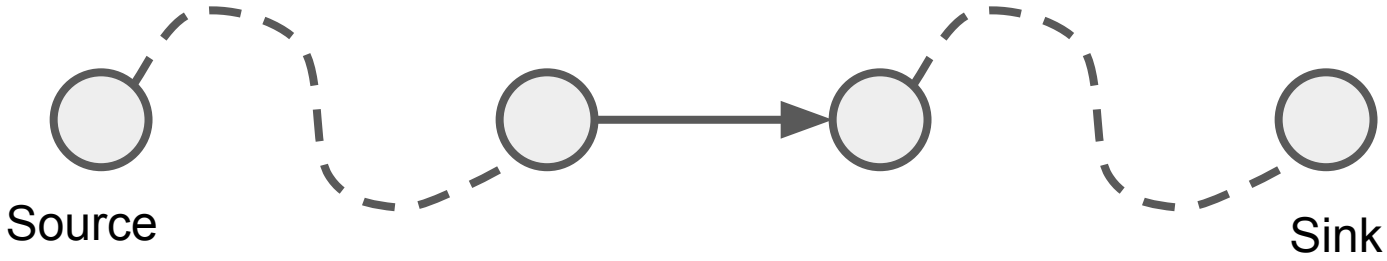
Meet in the Middle

Idea: Run Dijkstra's algorithm from the source (on the original graph) and from the sink (on the transpose of the graph)

Types of question this technique can answer:

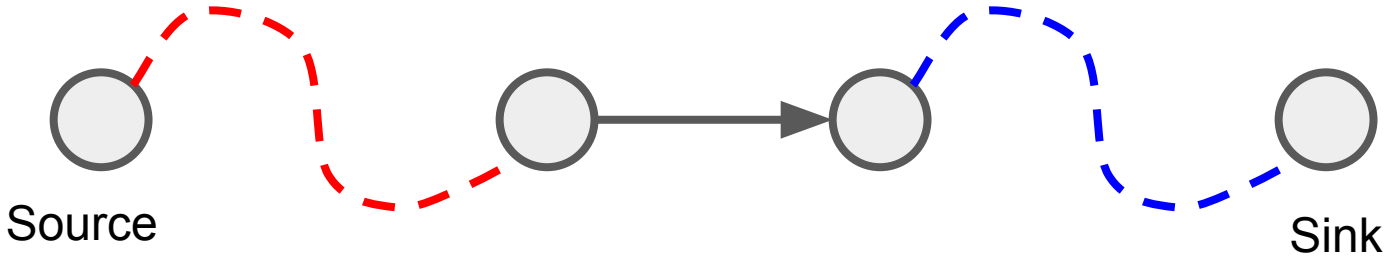
- Is an edge possibly part of a shortest path between 2 points?
- If a new edge is added into the graph, will the shortest path between 2 points become shorter?

Meet in the middle



If you were travelling from the source to the sink, and I force you to use this edge, how do you minimize the total distance travelled?

Meet in the middle



If you were travelling from the source to the sink, and I force you to use this edge, how do you minimize the total distance travelled?

- Take the shortest path from the source to one end of that edge
- Take the shortest path from the other end of that edge to the sink