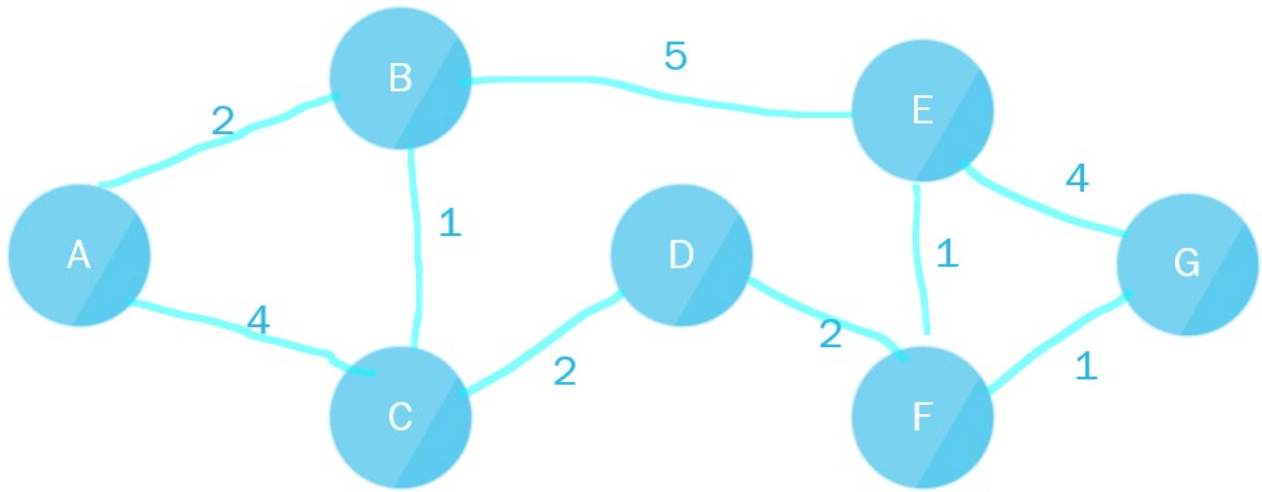
Playbook Notes: Dijkstras Shortest Path Algorithm

Dijkstras algorithm will allow you determine the shortest path from a starting node to each node in the graph. Using Dijkstra algorithm, we assume that all edges are weighted, possibly representing a distance or level of difficulty to the next node.

Assume the following graph, and we will start from node A.



Algorithm:

1. Mark all nodes as unvisited

2. Create two lists. One for visited nodes and one for unvisited nodes.

    Visited Nodes: NULL

    Unvisited Nodes: A, B, C, D, E, F, G

3. Assign all nodes a tentative distance value and allow for a previous node to be recorded.

Node	Shortest Distance	Previous Node
A	0	
B	Inf.	
C	Inf.	
D	Inf.	
E	Inf.	
F	Inf.	
G	Inf.	

A is set to 0 because A to A requires no distance. All other values are set to infinity.

Current Node is set to A.

While (current node != NULL)
{

4. For the current node, calculate the distance to all unvested neighbors.

    If the updated distance is shorter than the old, update the table.

5. Mark the node as visited

6. Choose the new current node from the unvisited nodes with the minimal distance.

}

Algorithm Trace:

Start with A, we look at the distance from A to B and the distance from A to C.

Since the current distance to each node is infinity, the new distances are shorter.

So, we update the table.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	4	A
D	Inf.	
E	Inf.	
F	Inf.	
G	Inf.	

Visited: A

Unvisited: B, C, D, E, F, G

Since B has the shortest distance of all the unvisited nodes, we will visit B.

Now we visit B. Examine B to E and B to C.

The distance from B to C is equal to 1. Add the distance from A to B (2) that equals 3.

Since three is less than the current value of 4, we update the table.

We also update E with 7 from B. (5 + 2 = 7)

Move B to the visited list.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	Inf.	
E	7	B
F	Inf.	
G	Inf.	

Visited: A, B

Unvisited: C, D, E, F, G

Since C has the shortest distance of all the unvisited nodes, we will visit C.

Now examine C. Examine C to D. The total distance to D is (2 + 1 + 2 = 5)

Since five is less than infinity, we update the table.

Move C to the visited list.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	5	C
E	7	B
F	Inf.	
G	Inf.	

Visited: A, B, C

Unvisited: D, E, F, G

Since D has the shortest distance of all the unvisited nodes, we will visit D.

Now examine D. Examine D to F. The total distance to F is (2 + 1 + 2 + 2 = 7)

Since seven is less than infinity, we update the table.

Move D to the visited list.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	5	C
E	7	B
F	7	D
G	Inf.	

Visited: A, B, C, D

Unvisited: E, F, G

Nodes E and F are tied at 7. So, let's pick F.

Now examine F. Examine F to E

The total distance F to E is (2 + 1 + 2 + 2 + 1 = 8). This is greater than 7. So, nothing is updated.

Examine F to G: (2 + 1+ 2 + 2 + 1 = 8). Since 8 is less than infinity, the table is updated.

Move F to the visited list.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	5	C
E	7	B
F	7	D
G	8	F

Visited: A, B, C, D, F

Unvisited: E, G

Node E has the smallest distance from the unvisited list, so next we will visit E.

Now examine E. Examine E to G

The total distance E to G is (2 + 5 + 4 = 11). This is greater than 8. So, nothing is updated.

Move E to the visited list.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	5	C
E	7	B
F	7	D
G	8	F

Visited: A, B, C, D, F, E

Unvisited: G

The only node left is G.

G has no neighbors, so we move G to visited and we are done.

Node	Shortest Distance	Previous Node
A	0	
B	2	A
C	3	B
D	5	C
E	7	B
F	7	D
G	8	F

We now have the shortest path to every other node starting from A.

If we wanted the shortest path from A to G, would just work the path backwards from G. .

A → B → C → D → F → G