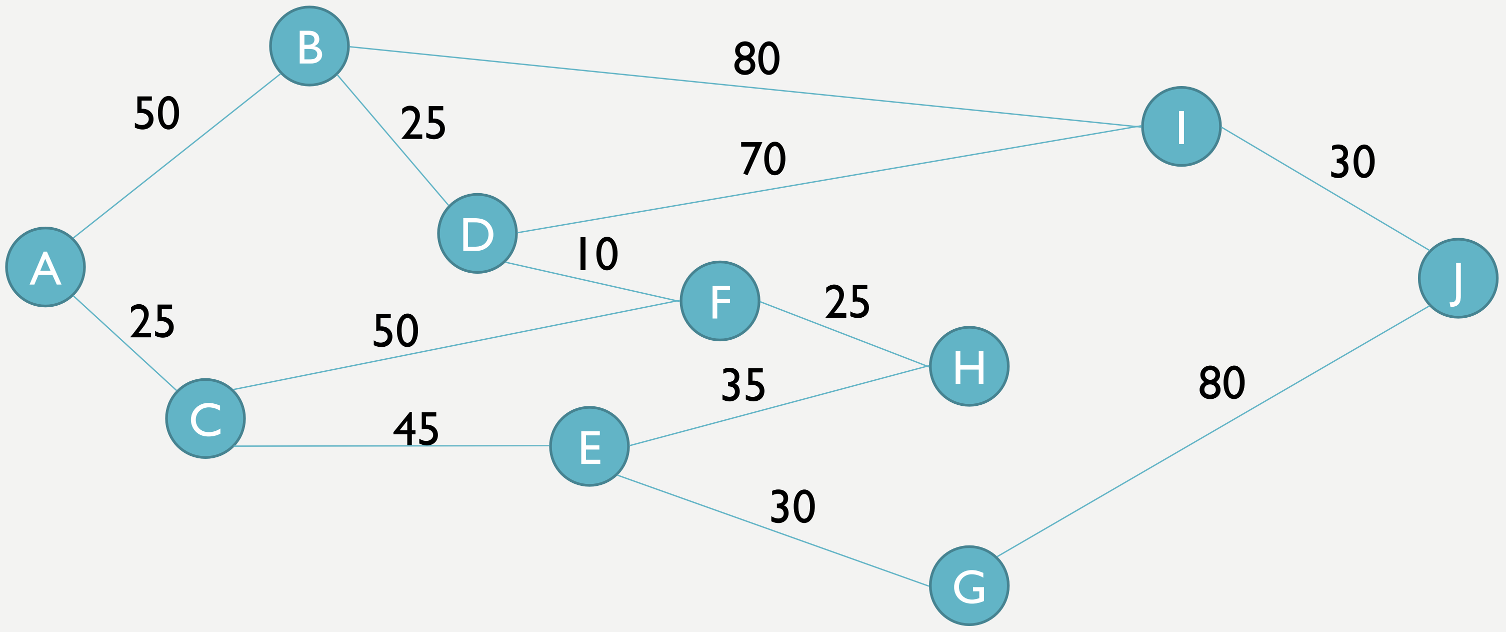
LEARNING OBJECTIVES

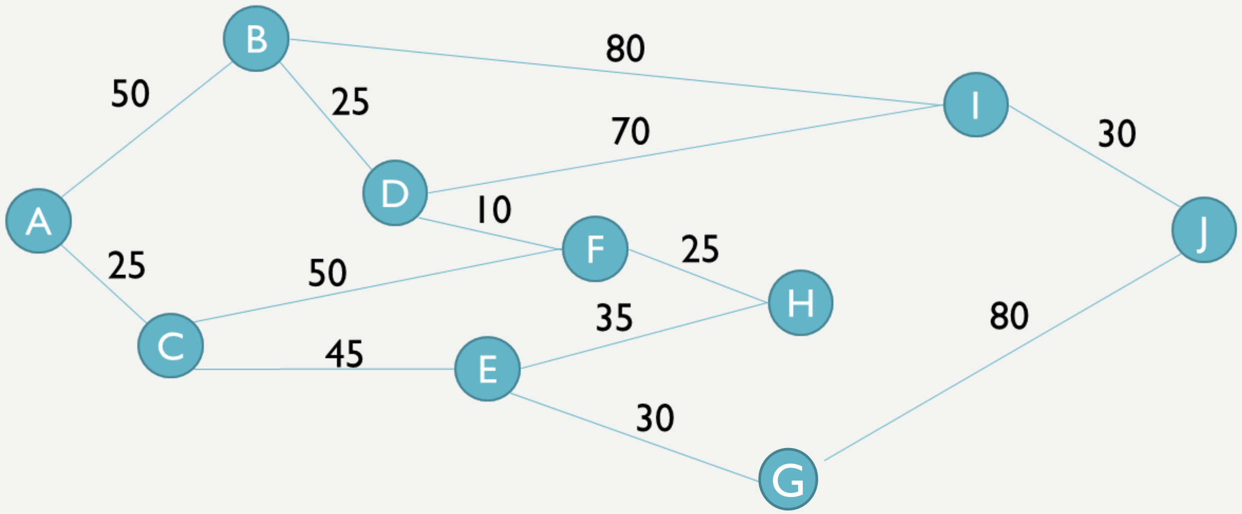
Dijkstra’s shortest path algorithm

DIJKSTRA'S ALGORITHM WORKED EXAMPLE

- Using the graph below, we shall use Dijkstra's algorithm to find the shortest path from A to J.



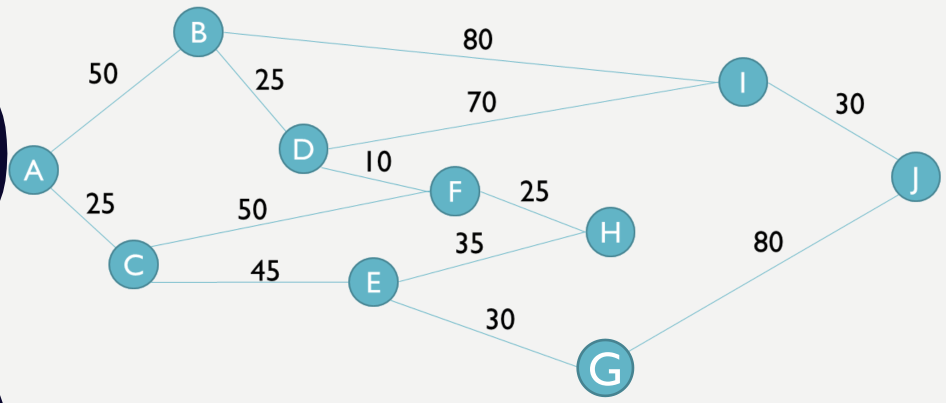
USING THE GRAPH BELOW, WE SHALL USE DIJKSTRA'S ALGORITHM TO FIND THE SHORTEST PATH FROM A TO J.
WE BEGIN WITH A AS THE "CURRENT NODE".



Node	Shortest distance from A	Previous Node
A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	
I	∞	
J	∞	

DIJKSTRA’S ALGORITHM -WORKED EXAMPLE

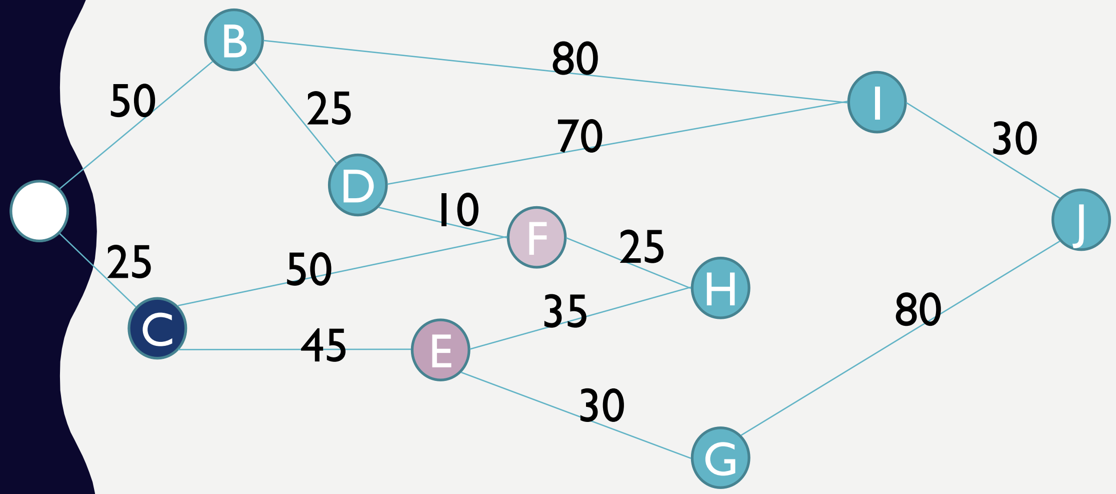
- So we begin at A as the current node.
- B becomes = 50 and C = 25
- We mark A as visited and make the node with the shortest time current node – in this case C



Node	Shortest distance from A	Previous Node
A (c)	0	
B	∞ 50	A
C	∞ 25	A
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	
I	∞	
J	∞	

So now C is the current node we look at distance to E and F (remember the distance is from A)

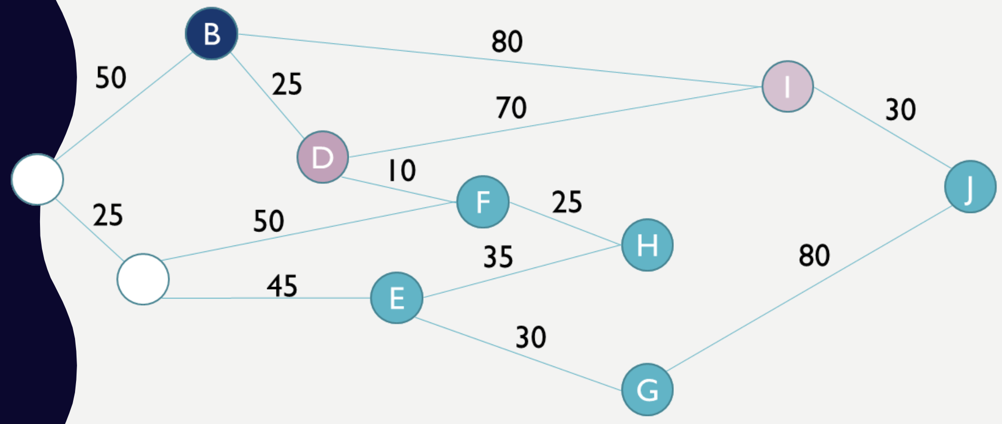
- E is marked as 70 (25 + 45) as distance from A
- F is marked as 75 (25 + 50) as distance from A
- The closest unvisited node is now B so it is marked as current node



Node	Shortest distance from A	Previous Node
A (v)	0	
B	∞ 50	A
C (c)	∞ 25	A
D	∞	
E	∞ 70	C
F	∞ 75	C
G	∞	
H	∞	
I	∞	
J	∞	

So now B is the current node we look at distance to I and D (remember the distance is from A)

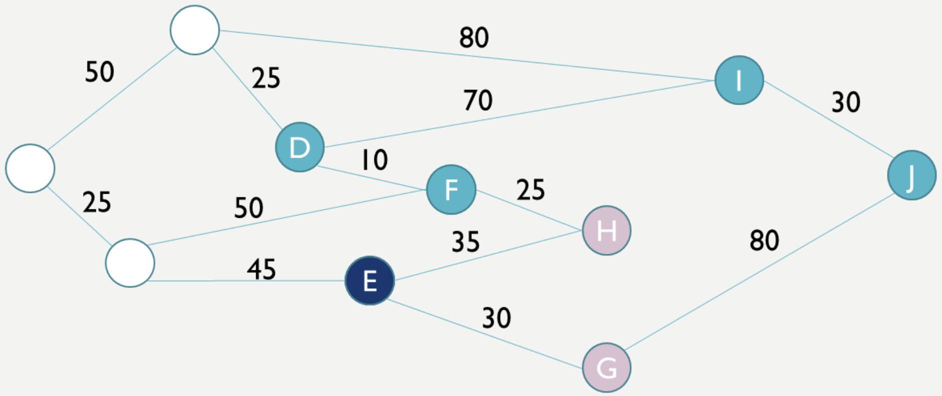
- D is marked as 75 (50 + 25) as distance from A
- I is marked as 130 (50 + 80) as distance from A
- B will be marked as visited and we move to E



Node	Shortest distance from A	Previous Node
A (v)	0	
B (c)	∞ 50	A
C (v)	∞ 25	A
D	∞ 75	B
E	∞ 70	C
F	∞ 75	C
G	∞	
H	∞	
I	∞ 130	B
J	∞	

So now E is the current node we look at distance to H and G (remember the distance is from A)

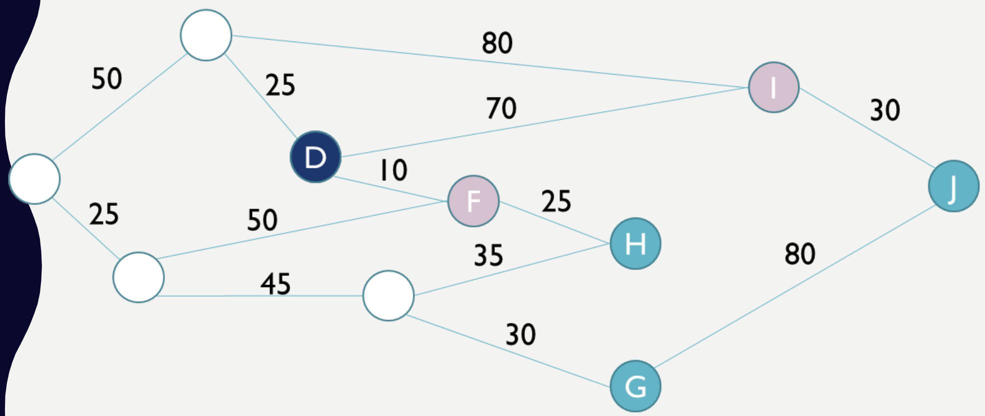
- G is marked as 100 (25 + 45 + 30) as distance from A
- H is marked as 105 (25 + 45 + 35) as distance from A
- E will be marked as visited and we can move to D or F as they are both the new shortest – so start at D



Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D	∞ 75	B
E (c)	∞ 70	C
F	∞ 75	C
G	∞ 100	E
H	∞ 105	E
I	∞ 130	B
J	∞	

So now D is the current node we look at distance to I and F (remember the distance is from A)

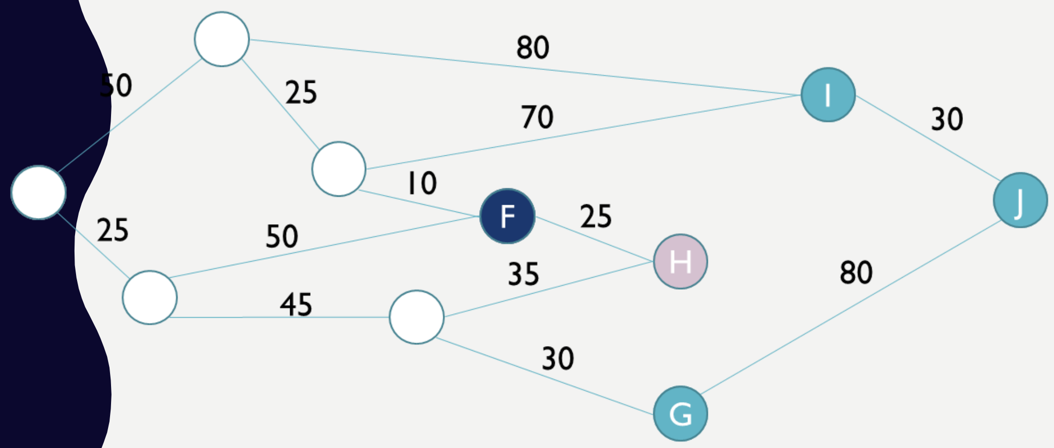
- I From A via D is 75 + 70 = 145 this is higher than current value of I so no update is done.
- F from A via D is 75 + 10 = 85 which is higher than current value so F is not updated either.
- F now becomes the current node



Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (c)	∞ 75	B
E (v)	∞ 70	C
F	∞ 75	C
G	∞ 100	E
H	∞ 105	E
I	∞ 130	B
J	∞	

So now F is the current node we look at distance to H as it is the only connected node. (remember the distance is from A)

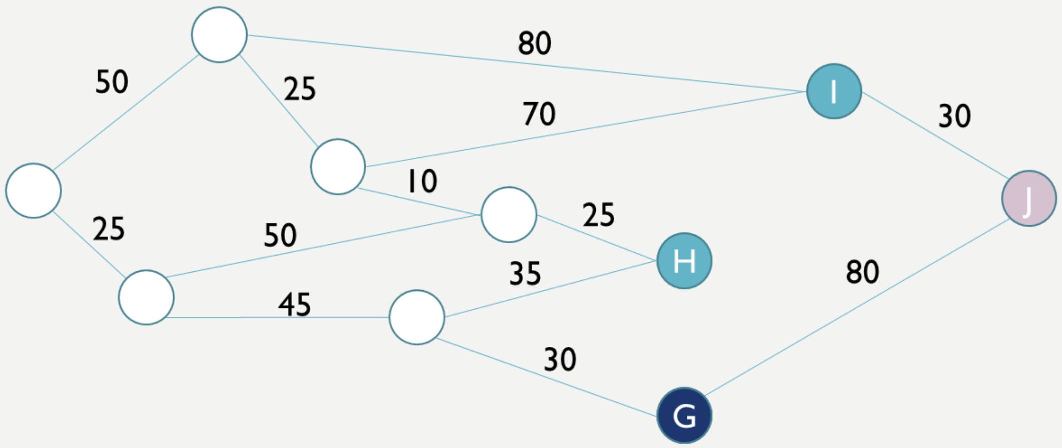
- H From A via F is 75 + 25 = 100 this value is lower than the current value of H so H is updated and the previous node is changed from E to F
- G or H could be the current node so again we start at the lowest alphabetically G



Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (v)	∞ 75	B
E (v)	∞ 70	C
F (c)	∞ 75	C
G	∞ 100	E
H	∞ ~~105~~ 100	~~E~~ F
I	∞ 130	B
J	∞	

So now G is the current node we look at distance to J as it is the only connected node. (remember the distance is from A)

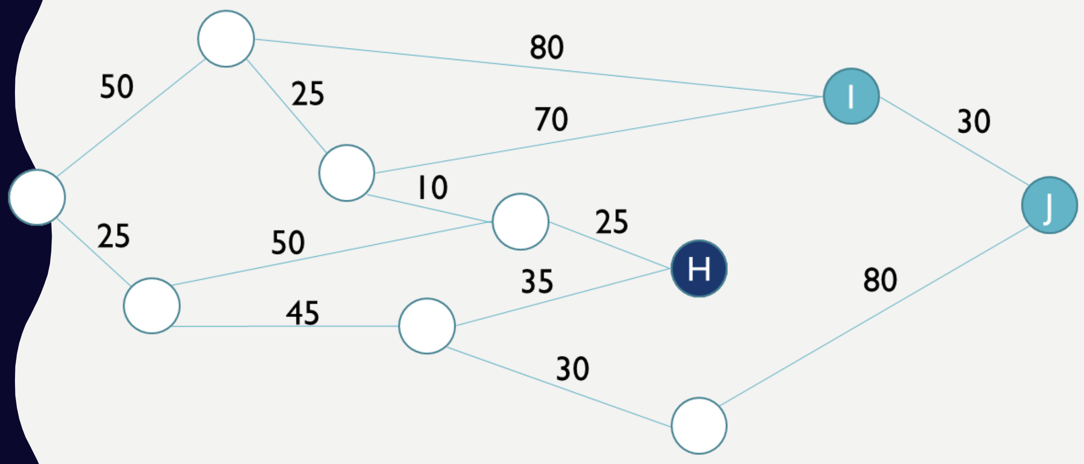
- J From A via G is 100 + 80 = 180 J is updates with previous node being G
- H is now the current node



Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (v)	∞ 75	B
E (v)	∞ 70	C
F (v)	∞ 75	C
G (c)	∞ 100	E
H	∞ ~~105~~ 100	~~E~~ F
I	∞ 130	B
J	∞ 180	G

As H has no connected nodes we simply set it to visited.

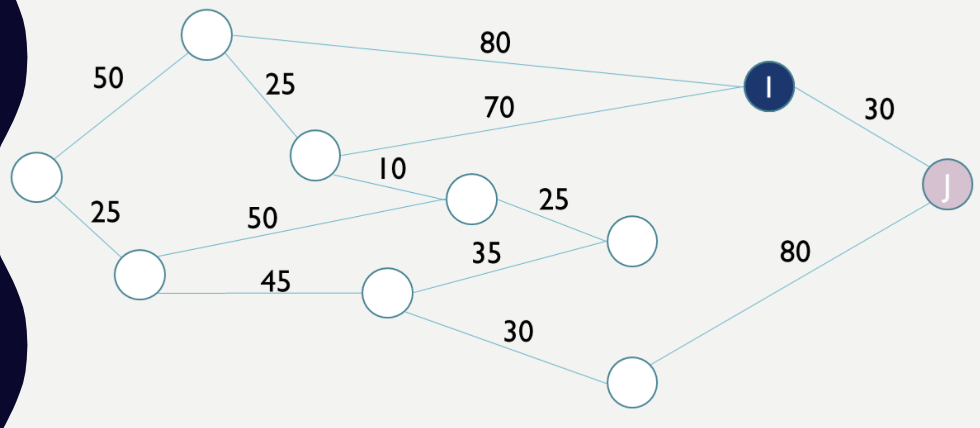
- I becomes the current node.



Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (v)	∞ 75	B
E (v)	∞ 70	C
F (v)	∞ 75	C
G (v)	∞ 100	E
H (v)	∞ ~~105~~ 100	~~E~~ F
I	∞ 130	B
J	∞ 180	G

With I as the current node we look at the distance to J via I.

- J via I is 130 + 30 = 160
- As 160 is lower than current value we update as follows.
- And we set J to current node.



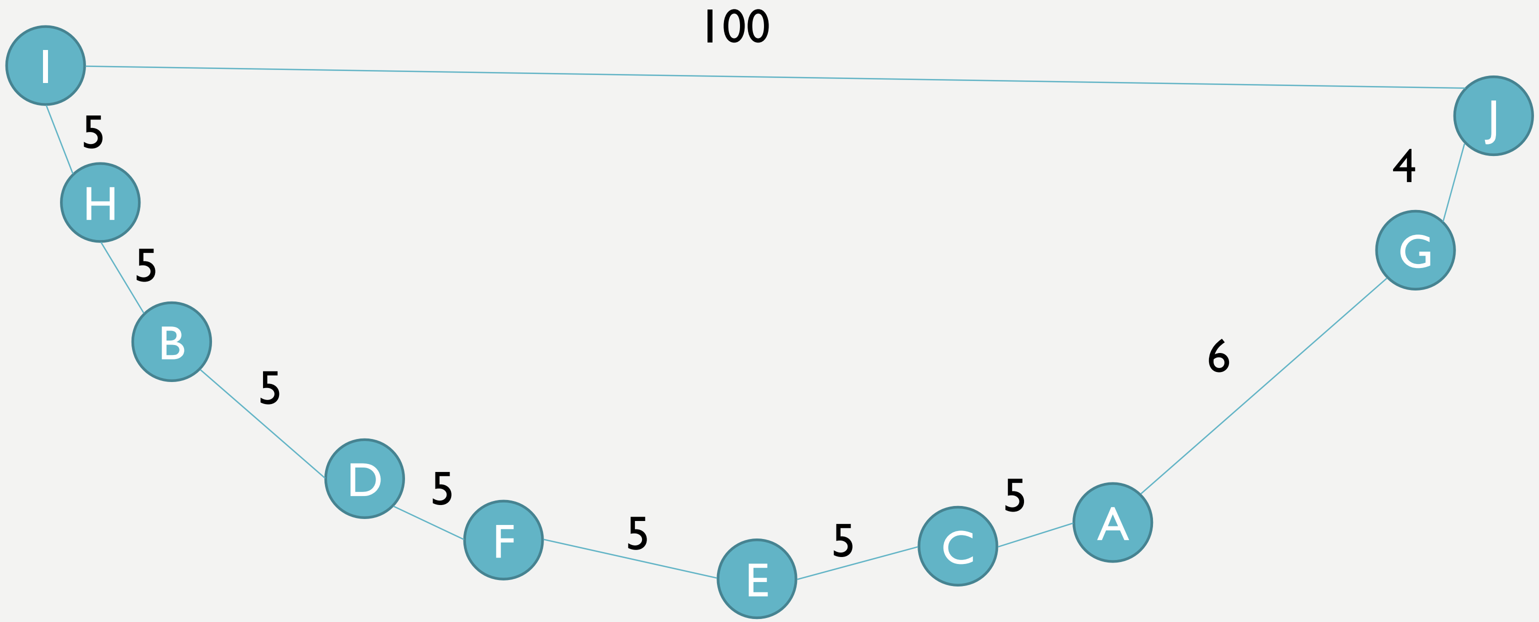
Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (v)	∞ 75	B
E (v)	∞ 70	C
F (v)	∞ 75	C
G (v)	∞ 100	E
H (v)	∞ ~~105~~ 100	~~E~~ F
I (c)	∞ 130	B
J	∞ ~~180~~ 160	~~G~~ I

- So we have now we know the time shortest time from A to J is 160.
- We now need to look at the path – Previous node to J is I, from I is B and B is A.
- So the shortest path is A, B, I J.

Node	Shortest distance from A	Previous Node
A (v)	0	
B (v)	∞ 50	A
C (v)	∞ 25	A
D (v)	∞ 75	B
E (v)	∞ 70	C
F (v)	∞ 75	C
G (v)	∞ 100	E
H (v)	∞ ~~105~~ 100	~~E~~ F
I (v)	∞ 130	B
J (c)	∞ ~~180~~ 160	~~G~~ I

DIJKSTRA'S ALGORITHM

- Of course that works so long as it's not obvious.



- Here you can see the shortest route is from A G to J – 45.
- But using Dijkstra's Algorithm, you would need to visit every node to work that out.

DIJKSTRA'S ALGORITHM

- The algorithm is as follows:

```
Mark the start node as a distance of 0 from itself and all other nodes as an
infinite distance from the start node.

WHILE the destination node is unvisited:

    Go to the closest unvisited node to A (initially this will be A itself) and
    call this the current node.

    FOR every unvisited node connected to the current node:
        Calculate the distance to the current plus the distance of the edge to
        unvisited
        If this distance is less than the currently recorded shortest distance,
        make it the new shortest distance.
    NEXT Connected node

    Mark the current node as visited.

ENDWHILE
```