AQA Computer Science A-Level

4.3.6 Optimisation algorithm

Advanced Notes

Specification:

4.3.6.1 Dijkstra's shortest path algorithm

Understand and be able to trace Dijkstra's shortest path algorithm

Be aware of applications of shortest path algorithm

Optimisation Algorithms

An optimisation algorithm finds the best possible solution to the problem posed. The only one you must be aware of is Dijkstra's (pronounced dyke-struh's) algorithm, which finds the shortest path from a starting node to every other node in a network.

These points may be modelled as nodes in a weighted graph. You will not be asked to recall the steps of the algorithm in an exam, although you may be asked to identify it and state its purpose and/or trace the code.

Dijkstra's Algorithm

As mentioned above, Dijkstra's algorithm is used to find the shortest path between two nodes in a graph. If you take maths for A level, you may have already come across this algorithm.

However, in maths Dijkstra you will normally only be asked for the finished path, whereas computer science Dijkstra will require you to have a full understanding of the code, often leading to filling in a dry run table.

Dijkstra's algorithm similar to the breadth-first search algorithm, but keeps track of visited nodes with a priority queue rather than a standard queue.

Applications of Dijkstra's algorithm

Dijkstra's algorithm is heavily used in computer systems that need to calculate shortest paths. This includes satellite navigation systems that display the shortest or fastest route from a starting point to a chosen destination.

Routers in networks often employ Dijkstra's algorithm to find the shortest path when routing packets within networks.

Algorithm

An **algorithm** is a finite **sequence of instructions** that can be followed to complete a task and that **always terminates**.

Synoptic Link

Graphs can be used as **visual representations** of **complex relationships**. **Weighted Graphs** have values assigned to each **edge**.

Graphs are covered in **Graphs** under **Fundamentals of Data Structures**.

Synoptic Link

Tracing code refers to a **human** following the **algorithm** - often used for **dry run testing.**

Code tracing is covered in **Abstraction and Automation** under **Theory of Computation**

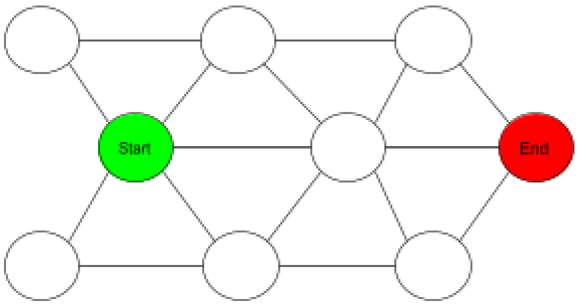
Synoptic Link

Routers are covered under **the Internet** in **fundamentals of communication and networking**.

Dijkstra's Algorithm Overview:

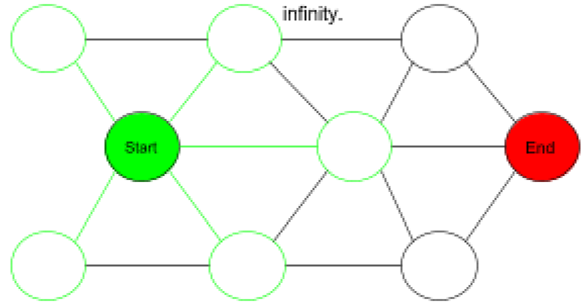
Step 1

Select Start and End nodes



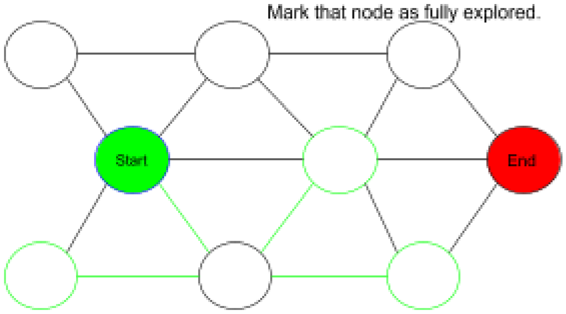
Step 2

Make note of the distances of the nodes from the start position. At this point, only the nodes connected to start will have a value, the others will have a distance of infinity.



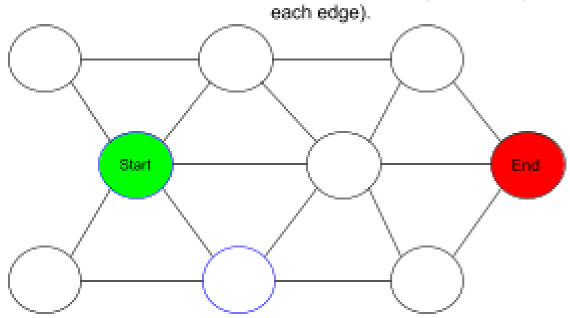
Step 3

Note the start node as fully explored. Choose the node closest to the start node, and update the table to reflect the shortest distance from A to each node. Mark that node as fully explored.



Step 4

Repeat step 3, always selecting the node the shortest distance from A which hasn't been fully explored. Continue until each node has been fully explored (and hence each edge).

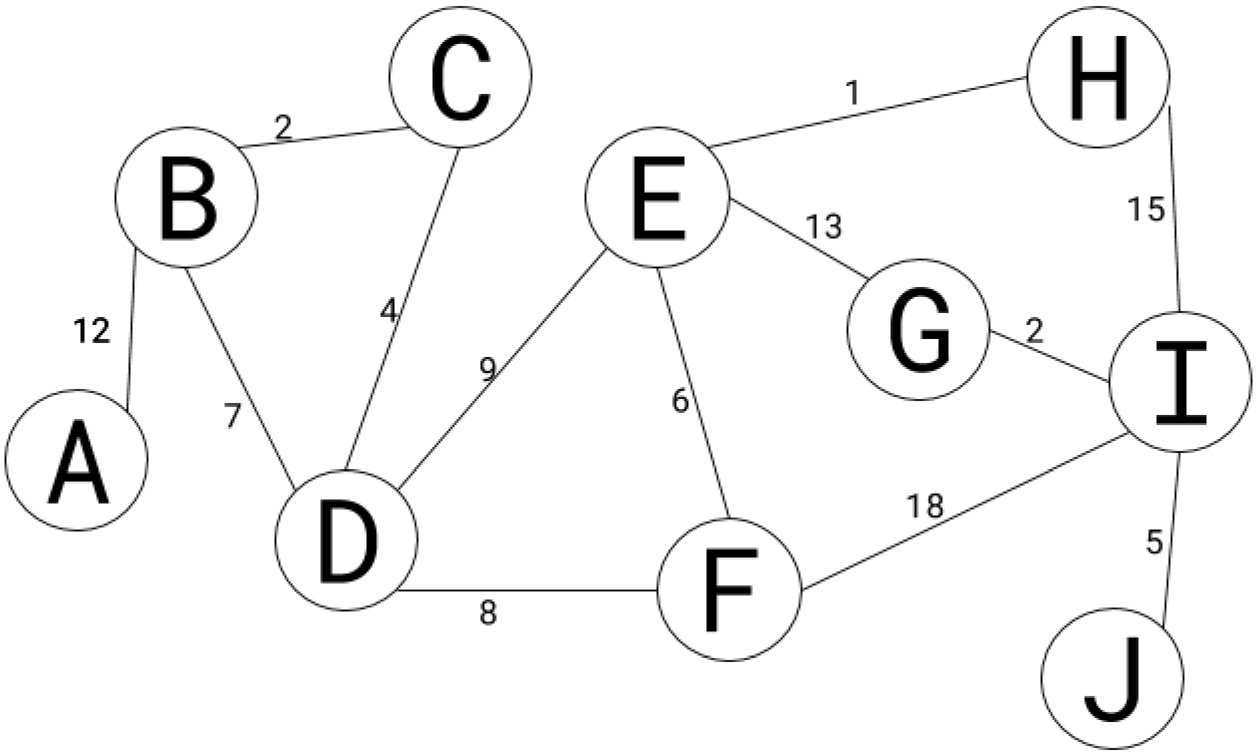


Dijkstra's Algorithm Example 1:

Edge

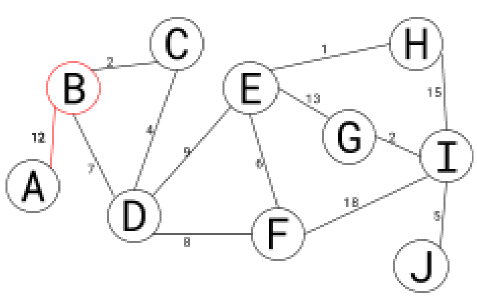
Edges/arcs are the connections between each node/vertex.

Here is a graph with nodes A to J. The numbers represent the minutes it would take travel along each edge.



Our task is to get from A to J in the shortest amount of time. The first step is to set up a table containing all the vertices.

V	A	B	C	D	E	F	G	H	I	J
A										

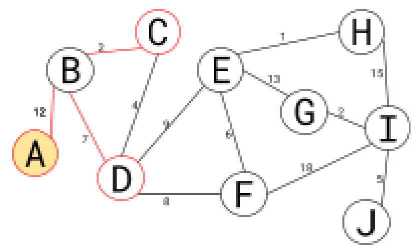


Starting from A, we list all the connections to other nodes. Where A is not connected to the node, we write ∞ because there is effectively no way to get to them - it would take an infinite time to reach them.

Looking at the graph we can see that A has only one edge incident on it. So we fill in the first line as thus:

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A

The subscripted A tells us that the weight is relative to A. The next step is to select the smallest number from the table- at first this would appear to be 0, but we have already investigated A (represented on the table by yellow shading), so this can be ignored. Instead, we notice that the smallest number is the 12 minutes it takes to get to B from A. Hence, the next line of the table filled in is that from B.



V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A			∞_A	∞_A	∞_A	∞_A	∞_A	∞_A

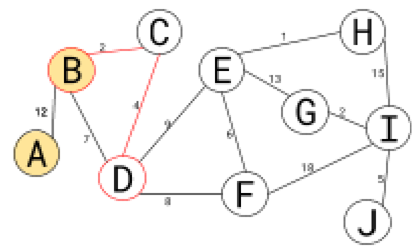
The most obvious parts of the table have been filled in. B has no connections to E F G H I or J, so the time to get to them is still infinite. The table is filled in with regards to A as we are trying to solve the shortest path between A and J, so the weight to B stays at 12. According to the graph, it takes 2 minutes to get from B to C (or vice versa). However, the table is relative to A. Therefore it takes 12 + 2 minutes to get to C from A via B; 14 is less than infinity so we have discovered a shorter path to the node C. We add this to the table as such:

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B		∞_A	∞_A	∞_A	∞_A	∞_A	∞_A

The subscript B tells us that B is the previous node visited. Similarly, D is 7 away from B and B is 12 away from A. The time it takes to get from A to D via B is 12 + 7. 19 < ∞, so 19 takes priority. B has now been investigated fully, so can be shaded yellow on the table.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A

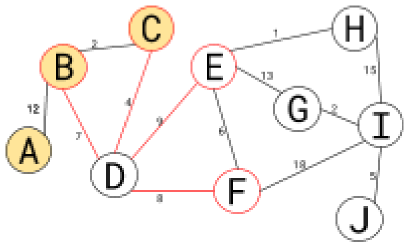
Now, we choose the smallest number in a blue column. Column C contains 14, which is smaller than 19 and infinity, so C is the next node to be explored. C has no connections to E F G H I or J so these remain unreachable. A and B have been fully explored so there shortest weight has already been discovered. The time it takes to get to C is still 14.



V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B		∞_A	∞_A	∞_A	∞_A	∞_A	∞_A

The only difference now is D. The time it takes to get to D from C is 4 minutes according to the graph. As it takes 14 minutes to get to C from A via B, it must take 14 + 4 minutes to get to D from A via C. 18 < 19 so 18 takes priority.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A



Now, the smallest number in the table in an unexplored column is 18. D is next to be explored. D has no connections to G H I or J, so the time to reach them remains infinite. A B and C have been fully explored so they already have a minimum time. The weight from D to D is 0, so 18 (18 + 0) stays.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C			∞_A	∞_A	∞_A	∞_A

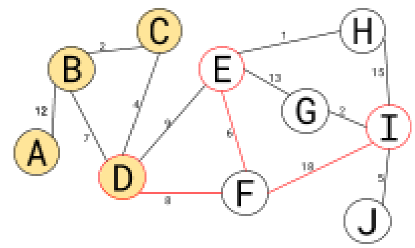
E is 9 from D. The shortest path to D from A takes 18 minutes, so it will take 27 minutes to get from A to E via D.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D		∞_A	∞_A	∞_A	∞_A

F is 8 away from D. D is 18 away from A, so F is 26 away from A via D.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A

The smallest number is now 26, so F is chosen to be next explored. There are still no connections to G H or J, so they remain infinite. A B C D and F have their shortest paths from A so these remain the same.



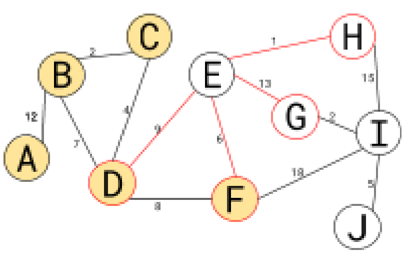
V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C		26_D	∞_A	∞_A		∞_A

E is 6 away from F. F is 26 away from A. Therefore E is 26 + 6 = 34 minutes away from A via F. However, this is a shortest path algorithm, and we have already discovered that E is 27 minutes away from A via D. 27 < 34, so 27 takes priority and is not replaced in the table.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A		∞_A

I is 18 away from F. 18 + 26 = 44. 44 < ∞, hence 44 takes priority.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A



The unexplored nodes are E, G, H, I and J. Out of those, E has the smallest value at 27, so E is next explored. Nodes A B C D F and E already have their shortest paths. The only node which remains unconnected is J, so the weight to it remains infinite.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D				∞_A

G is 13 away from E, which in turn is 27 away from A. Therefore, G is 40 away from A via E. 40 < ∞ so 40 is written in the table.

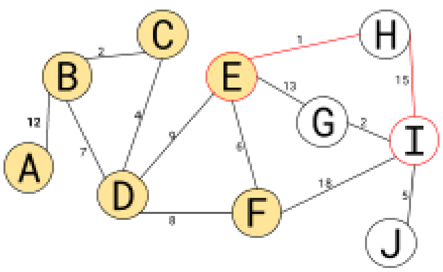
V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E			∞_A

H is 1 away from E, so the time between A and H is 27 + 1 = 28 minutes. 28 < ∞ so 28 is added to the table.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E		∞_A

Lastly E has no connection to I, so the 44 via F remains.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A



The smallest value in one of the blue columns is 28, column H. H will be the next node to be explored. The shortest path from A has already been found for nodes A B C D E F and H.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D		28_E		

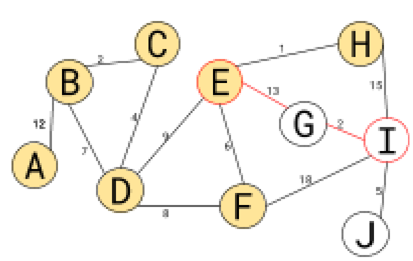
H does not have an edge incident on G or J, so their values remain the same.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E		∞_A

H is 15 from I. Therefore, I is 15 + 28 = 43 minutes away from A via H. 43 < 44, so 43 takes precedence and is added to the table.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	43_H	∞_A

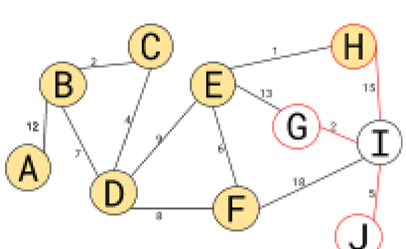
The smallest value is 40 in column G; G is the next node to be explored. Shortest paths have been found for A B C D E F H and G. J has yet to be discovered, so it remains infinite in the table.



V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	43_H	∞_A
G	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E		∞_A

The time between G and I is 2. Therefore, it takes 40 + 2 = 42 minutes to get to I from A via G. 42 < 43. The 42 takes precedence over the 43 and so is added to the table.

V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	43_H	∞_A
G	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	∞_A



I and J are the only nodes left unexplored. 42 is smaller than infinity, so I is selected for exploration. A B C D E F G H and I have a defined shortest path from A.

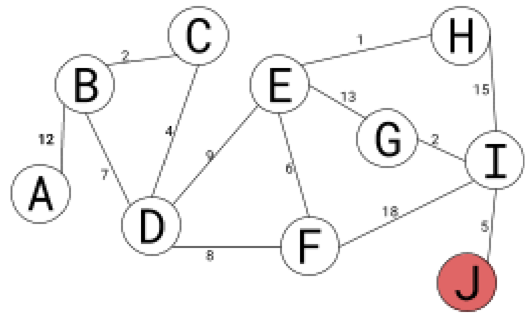
V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	43_H	∞_A
G	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	∞_A
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	

J is 5 away from I. Therefore J is 42 + 5 = 47 minutes away from A via I. 47 is smaller than infinity.

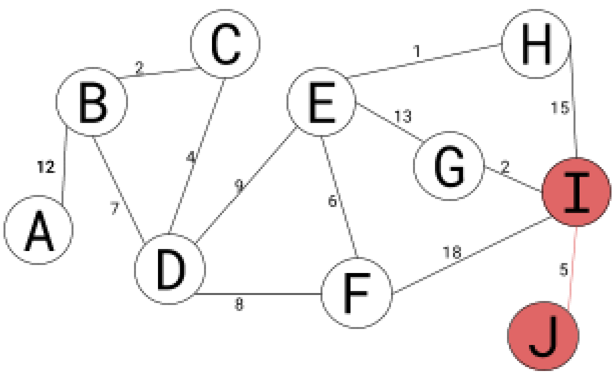
V	A	B	C	D	E	F	G	H	I	J
A	0_A	12_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
B	0_A	12_A	14_B	19_B	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
C	0_A	12_A	14_B	18_C	∞_A	∞_A	∞_A	∞_A	∞_A	∞_A
D	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	∞_A	∞_A
F	0_A	12_A	14_B	18_C	27_D	26_D	∞_A	∞_A	44_F	∞_A
E	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	44_F	∞_A
H	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	43_H	∞_A
G	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	∞_A
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

As J is the last node, all edges must have been explored. The table is complete.

From the table, we can work backwards to find the shortest path between A and J. We only need to use the bottom row. Firstly, we note that the path will be 47 in length.



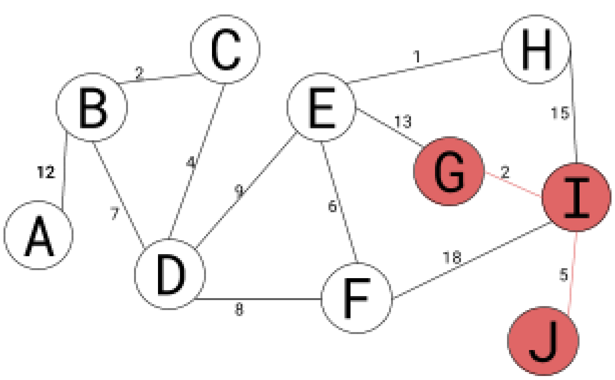
V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I



The letter accompanying this 47 is I. The node visited immediately before J is I. So now, we need the shortest path to I from A.

V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

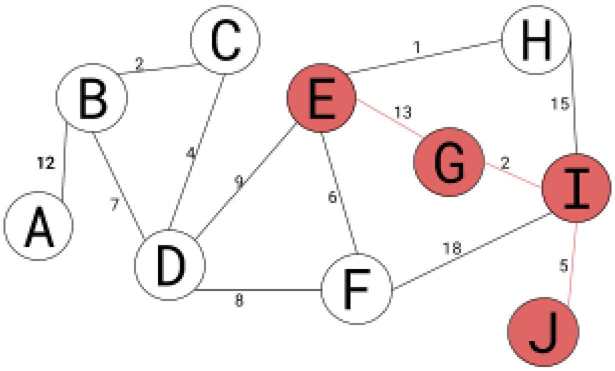
The shortest path to I has a length of 42. The node immediately preceding it is G.



Next, we need to find the shortest path to G from A.

V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

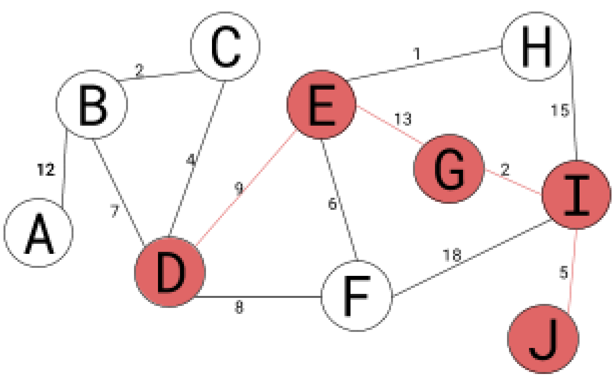
The node before G is E.



Now we look at column E in the table.

V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

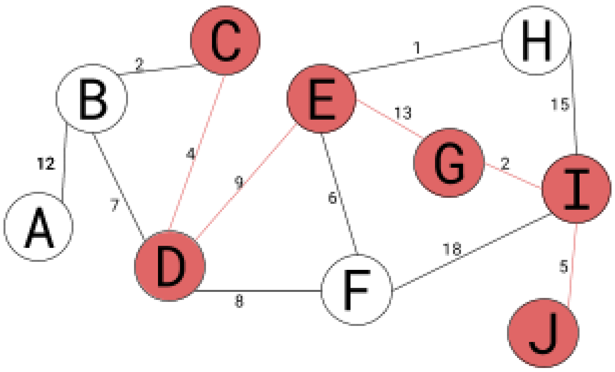
D is just before E.



Next we look at column D in the table.

V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

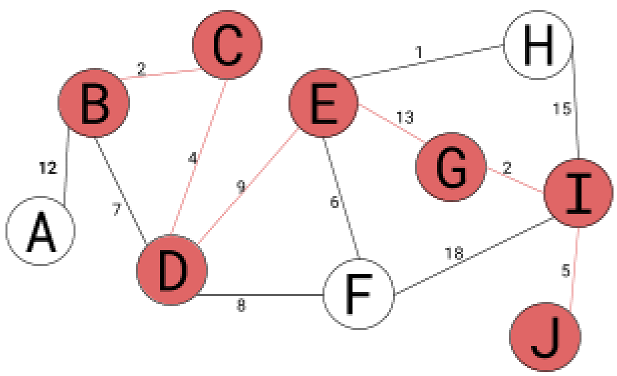
D follows on from C.



Now, we look at column C in the table.

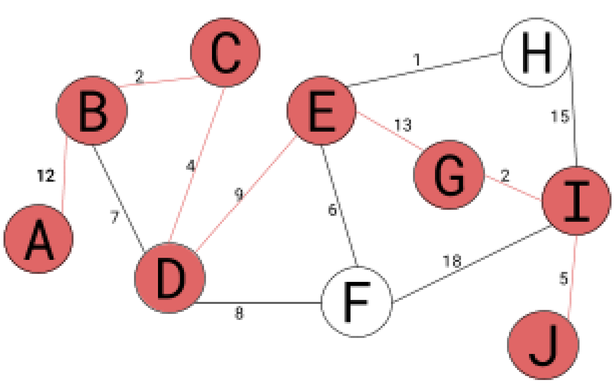
V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I

The node before C is B.



The column headed B is looked at next.

V	A	B	C	D	E	F	G	H	I	J
I	0_A	12_A	14_B	18_C	27_D	26_D	40_E	28_E	42_G	47_I



Before B is A.

The path is complete; the shortest path to get from A to J is A B C D E G I J and will take 47 minutes.