OCR Computer Science A Level

2.3.1 Path Finding Algorithms

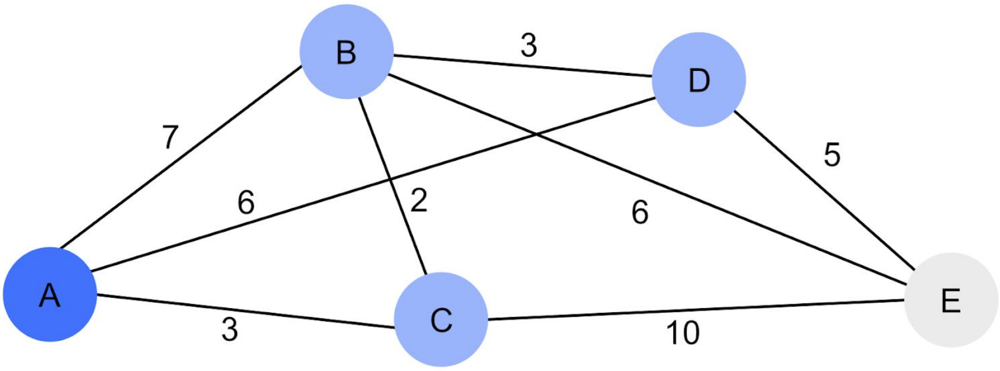
Concise Notes

Specification:

- Dijkstra’s shortest path algorithm
- A* algorithm

Dijkstra’s algorithm

- Dijkstra’s algorithm finds the shortest path between two nodes in a weighted graph.
- Graphs are used as an abstraction for real life scenarios.
- Nodes and edges can represent different entities.
- Implemented using a priority queue, with the smallest distances being stored at the front of the list.
- Below is a step-by-step example of using Dijkstra’s algorithm to find the shortest path between A and E.



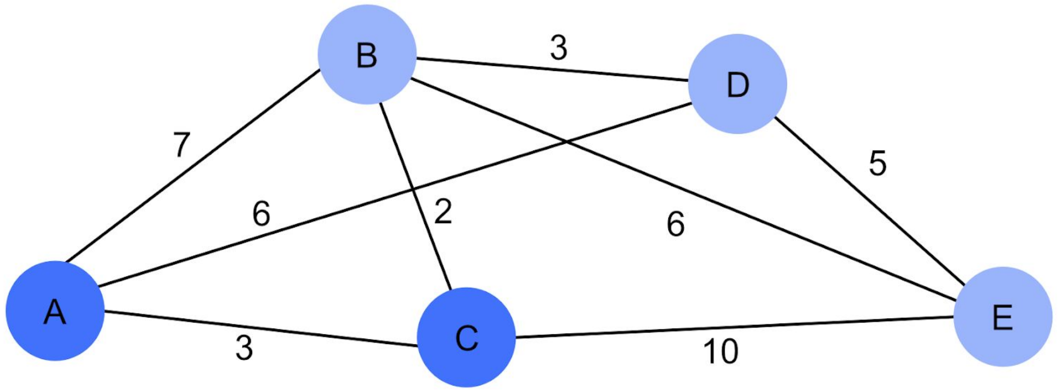
Priority Queue

C = 3	D = 6	B = 7	E = ∞	

Step 1

- Starting from the root node (A), add the distances to all of the immediately neighbouring nodes (B, C, D) to the priority queue.
- The table below shoud be used to keep track of visited nodes and distances.
- Begin by filling in the ‘Node’ column with the nodes connected to the root node.
- The ‘From’ column should contain the node that you are travelling from.
- The ‘Distance’ column contains the distance between these two nodes.
- The ‘Total distance’ is the sum of the distances from the root node to that particular node.
- The node which is the shortest distance away from the root node is highlighted and selected for the next stage.

Node	From	Distance	Total distance
B	A	7	7
C	A	3	3
D	A	6	6



AC = 3

Priority Queue

B = 5	D = 6	E = 13		

Step 2

- Remove the node the shortest distance away, from the front of the queue.
- Now traverse all of the nodes connected to the removed node C.
- If the total distance passing through the removed node to the neighbouring node is smaller than the distance currently stored with this node, update this value to the smaller distance.
- C has two neighbours: B and E. The shortest total cost of travelling to E so far is 13, as it is less than the infinite value previously allocated to E.
- Travelling from A to C to B adds up to a total cost of 5, while travelling from A to B has a cost of 7. B's cost value is therefore updated to 5.
- We can ignore nodes we have visited, such as B.
- We can remove routes that are not the shortest way to get to a particular node.

Node	From	Distance	Total distance
~~B~~	~~A~~	~~7~~	~~7~~
~~C~~	~~A~~	~~3~~	~~3~~
D	A	6	6
B	C	2	5
E	C	10	13

Step 3

- Continue repeating Step 2 until the goal node has been reached.

Node	From	Distance	Total distance
~~B~~	~~A~~	~~7~~	~~7~~
~~C~~	~~A~~	~~3~~	~~3~~
D	A	6	6
~~B~~	~~C~~	~~2~~	~~5~~
E	C	10	13
~~D~~	~~B~~	~~3~~	~~8~~

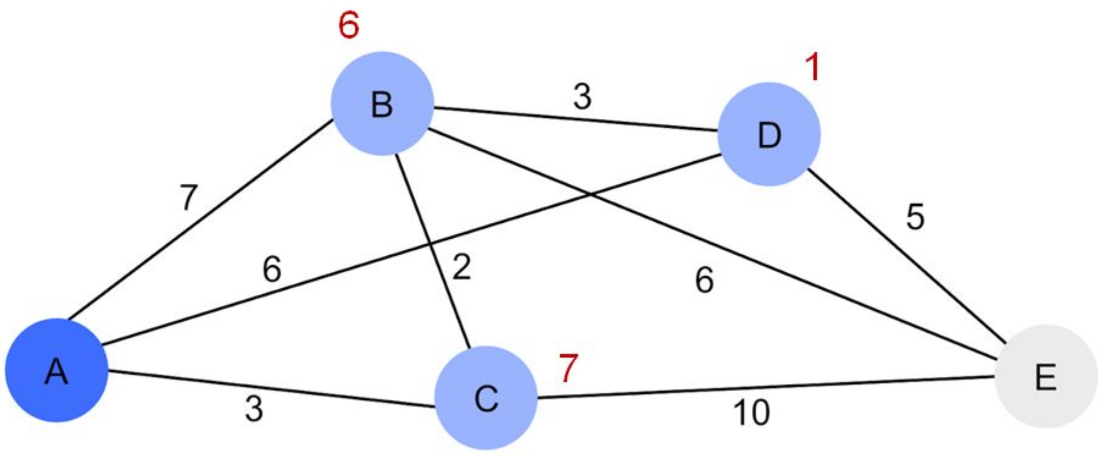
- Once again, we repeat this same process for node D.

Node	From	Distance	Total distance
~~B~~	~~A~~	~~7~~	~~7~~
~~C~~	~~A~~	~~3~~	~~3~~
D	A	6	6
~~B~~	~~C~~	~~2~~	~~5~~
~~E~~	~~C~~	~~10~~	~~13~~
~~D~~	~~B~~	~~3~~	~~8~~
E	D	5	11

- As we have now visited all of the nodes on the graph, we can confirm by tracing back through the table above that the shortest path is ADE.

A* algorithm

- TA general path-finding algorithm which is an improvement of Dijkstra's algorithm and has two cost functions:
    1) The actual cost between two nodes - the same cost as is measured in Dijkstra's algorithm.
    2) An approximate cost from node x to the final node, called a heuristic, which aims to make the shortest path finding process more efficient.
- The approximate cost is added onto the actual cost to determine which node is visited next.



Step 1

- The method used here is very similar to the method used in Dijkstra's algorithm,
- The difference is that the heuristic cost is added onto the actual cost to calculate the total cost.
- Again, the route with the lowest total cost is selected to traverse further.

Node	From	Distance	Heuristic	Total distance
B	A	7	6	13
C	A	3	7	10
D	A	6	1	7

Step 2

- The node D is then selected.
- Note that the heuristic cost of the previous node is not added on to the new distance.
- As 11 is the shortest total distance, the algorithm terminates.
- The shortest route is found to be ADE, at a cost of 11.

Node	From	Distance	Heuristic	Total distance
B	A	7	6	13
C	A	3	11	14
D	A	6	12	18
E	D	5	-	11

- The heuristics used here allow the shortest path to be found much quicker than when using Dijkstra's algorithm.
- How effective the A* algorithm is depends on the accuracy of the heuristics used.