**The Graph Data Structure:**

A graph is a data type, which is made nodes. These nodes are known as ‘vertices’. These vertices are joined by connections called edges. Unlike a tree data structure, a graph does not have a predictable/regular structure.

**Weighted Graph:**

* The edges have numerical ‘weights’
* The weight represents cost, distance or effort.
* For example, the distance on a map.

**Directed Graph:**

* A directed graph has one-way connections.
* You can only travel one way along the connection.
* The directions are shown by arrows.

**Dijkstra’s Algorithm:**

This is a way of computing the shortest path between two nodes on a graph.

1. Look for the shortest path from the start node.
2. Then the shortest path from that node and so on.
3. Work through all possible routes, shortest first.
4. Eventually one of the routes will reach the target node.
* We will be using this graph as an example.
* Let’s find the quickest route from A to B
* Let’s first draw a table for our working out

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Active |
| B |  |  |  |
| C |  |  |  |
| D |  |  |  |

* A is the ‘active node’ and the node we are starting from.

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | A | 6 |  |
| C | A | 3 |  |
| D |  |  |  |

* We then fill in the distances of the nodes that are connected to A.
* Node A is no longer being used, so we label it as ‘done’ because we will not be visiting it again.

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | A | 6 |  |
| C | A | 3 | Active |
| D |  |  |  |

* We will now pick the next node. We pick the node with the smallest value, which is C.
* C is now active.
* We now have to find out which nodes we can get from C

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | A | 6 |  |
| C | A | 3 | Active |
| D | C | 4 |  |

* You can get to D from node C.
* The total distance to D from A is 4. (3 + 1 = 4)
* We can get to B from C. However, the total distance from A, to C to B is 7 (3+4=7). This is longer than going from A straight to B, which is 6. We don’t use this route as it is longer.

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | A | 6 |  |
| C | A | 3 | Done |
| D | C | 4 | Active |

* We are now done with node C, so we make the status as done.
* Node D has the next smallest value, so we choose that as our active node.

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | A | 6 | Can Replace |
| C | A | 3 | Done |
| D | C | 4 | Active |

* The distance from D to B is 1. The distance from A to C to D is 4. 4+1 is 5.
* The distance from A to C to D to B is 5. This is shorter than A to B.
* We can now replace A to B with A – C – D – B

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Linked From: | Distance: | Status |
| A | - | 0 | Done |
| B | D | 5 |  |
| C | A | 3 | Done |
| D | C | 4 | Done |

* The shortest path to B is the route: A-C-D-B, with the distance 5

**In summary:**

1. The node with the smallest value becomes the active node
2. Find all nodes you can get to from the active node.
3. Calculate the total distance and only replace if the value is smaller.
4. The node is then done and not used again.
5. Repeat there are no more possible routes to the target node.