An example for Dijkstra’s algorithm:

Djikstra’s algorithm is used to find the single source shortest paths.

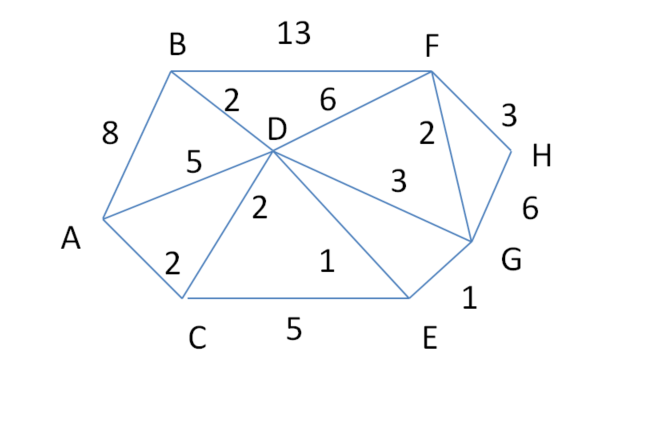
Djikstra and Prims are examples of Greedy Algorithms.

Recall that the **Floyd’s algorithm gave us the shortest paths, for every pair of nodes, in a directed or undirected graph weighted graph**. It assumed an adjacency matrix and had a complexity of 

Djikstra’s algorithm is also a shortest path algorithm for directed or undirected weighted graphs. **IT WOULD FIND ALL OF THE SHORTEST PATHS FROM A FIXED START NODE**. Its complexity would be the same as that of the Prim’s.

Example for Dijkstra:

Given graph:



Using my approach red values need not be considered again.

∞ would designate that the node is not in near vicinity.

Once we add a new node using the new node we would have compute the distance from the source to the destination node. If the distance is not less than the existing distance then it need not be updated. And also if a node is not reachable from a particular node then it need not be updated.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| v | A | B | C | D | E | F | G | H |
| A | 0 | 8 | 2 | 5 | ∞ | ∞ | ∞ | ∞ |
| A,C |  | 8 | 2 | 4(via C A to C to D with a cost of 2+2 ) | 7 | ∞ | ∞ | ∞ |
| A,C,D |  | 6 (A to C to D to B) |  | 4 | 5 (A to C to D to E) | 10 (A to C to D to F) | 7 (A to C to D to G) | ∞ |
| A,C,D,E |  | 6 (still 6 as adding e would increase the distance) |  |  | 5 | 10 | 7 (adding E would make it 8) | ∞ |
| A,C,D,E,B |  | 6 |  |  |  | 10 | 6 (A to C to D to E to G ) | 14 (A to D to F to H) |
| A,C,D,E,B,G |  |  |  |  |  | 8 | 6 | 11 |
| A,C,D,E,B,G,F |  |  |  |  |  | 8 |  | 10 |
| A,C,D,E,B,G,F,H |  |  |  |  |  |  |  | 10 |

PRIM’S Algorithm:

* Prim’s algorithm is an example of a greedy Algorithm.
* Used to create the Minimum spanning Tree.
* It constructs a sequence of subtrees , each adding a node together with an edge to a

node in the previous subtree. In each step it picks a closest node from outside the

tree and adds that.

In each iteration the tree would grow by one edge. The tree would grow to include the node from outside that has the smallest cost.

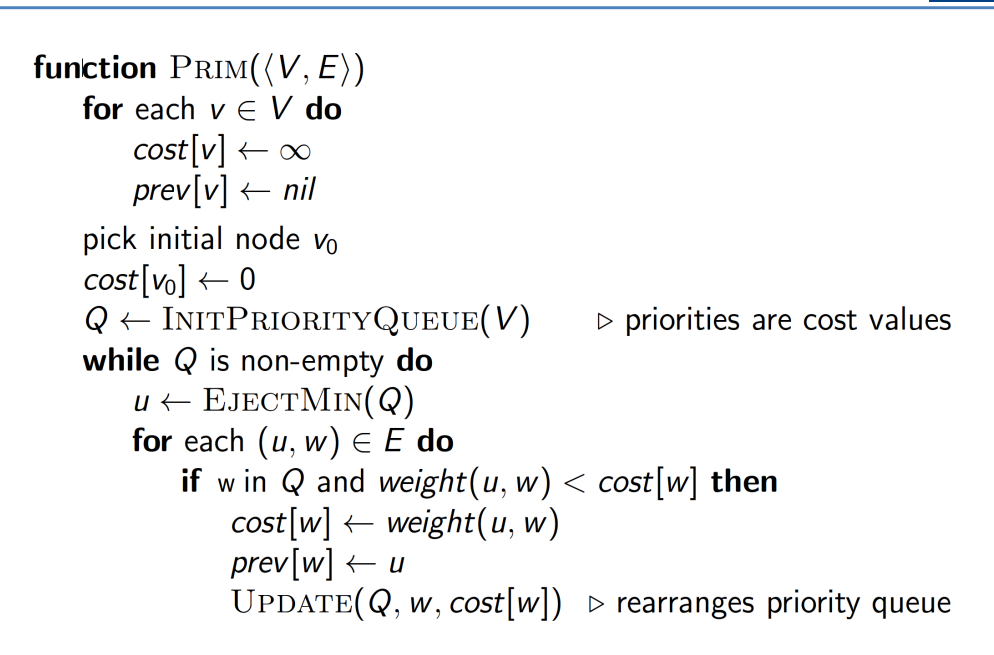
We would use a priority queue here, and organize the min-heap by edge cost.

The information about which nodes are connected in T can be captured by an array prevo od nodes indexed by V.

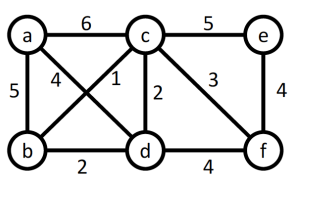
Namely, when (v, u) is included, this is captured by setting prev [u] = v.

Example for PRIM’S Algorithm:

The prim’s algorithm is given to be as:



Algorithm in Action:

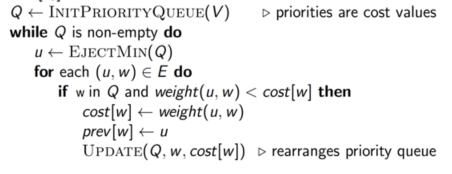


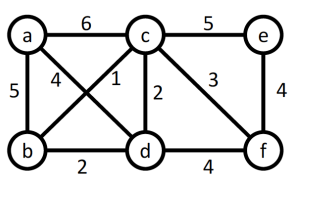


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tree T |  | a | b | c | d | e | f |
|  | Cost | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
|  | Prev | nil | nil | nil | nil | nil | nil |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tree T |  | a | b | c | d | e | f |
|  | Cost | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
|  | Prev | nil | nil | nil | nil | nil | nil |
|  |  |  |  |  |  |  |  |
|  | Cost | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
|  | Prev | nil | nil | nil | nil | nil | nil |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |





* Already added nodes need not be considered again.
* Need not consider distance from the previous nodes, at a particular distance from a particular node only needs to be considered. (Unlike Djijstra it is about finding the )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tree T |  | a | b | c | d | e | f |
|  | Cost | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
|  | Prev | nil | nil | nil | nil | nil | nil |
|  |  |  |  |  |  |  |  |
|  | Cost | 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
|  | Prev | nil | nil | nil | nil | nil | nil |
|  |  |  |  |  |  |  |  |
| a | Cost |  | 5 | 6 | 4 | ∞ | ∞ |
|  | Prev |  | a | a | a | nil | nil |
|  |  |  |  |  |  |  |  |
| a,d | Cost |  | 2 | 2 |  | ∞ | 4 |
|  | Prev |  | d | d |  | nil | d |
|  |  |  |  |  |  |  |  |
| a,d,b | Cost |  |  | 1 |  | ∞ | 4 |
|  | Prev |  |  | b |  | nil | d |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| a,d,b,c | Cost |  |  |  |  | 5 | 3 |
|  | Prev |  |  |  |  | c | c |
|  |  |  |  |  |  |  |  |
| a,d,b,c,f | Cost |  |  |  |  | 4 |  |
|  | Prev |  |  |  |  | c |  |
|  |  |  |  |  |  |  |  |
| **a,d,b,c,f,e** |  |  |  |  |  |  |  |