GRAPH PART-3.

MA CAY

SHORTEST PATH ALGORITHM.

→ To calculate the shortest path b/w the vertices of a graph G. These algorithm includes :→

- MINIMUM SPANNING TREE
- DIJKSTRA'S ALGORITHM

} uses adjacency list to find shortest path.

- WARSHALL'S ALGO.] — use adjacency matrix to do the same.

MINIMUM SPANNING TREES

A spanning tree of a connected, undirected graph G is a sub-graph of G which is a tree that connect all the vertices together.

A graph G has many diff. spanning tree.

We can assign weights to each edge, and use it to assign a weight to a spanning tree by calculating the sum of weight of the edges in that spanning tree.

A minimum spanning tree (MST) → is defined as the spanning tree with wt. less than or equal to the weight of every other spanning tree.

Properties.

- Possible multiplicity → There can be multiple minimum spanning tree of the same weight. Particularly, if all the weights are the same, then every spanning tree will be minimum.

- Uniqueness → When each edge in the graph is assigned a diff. weight, then there will be only one unique min. spanning tree.

- Minimum-cost subgraph → If the edges of a graph are assigned non-negative weights, then a minimum spanning tree is in fact the minimum-cost subgraph or a tree that connects all vertices.

- Cycle Property → If there exists a cycle C in the graph G that has a weight larger than that of other edges of C, then this edge cannot belong to an MST.

- Usefulness → MST can be computed quickly & easily to provide optimal solution.

- Simplicity → min. spanning tree of a weighted graph is nothing but a spanning tree of the graph which comprises of n-1 edges of minimum total weight.

Note :: for any unweighted graph, any spanning tree is a minimum spanning tree.

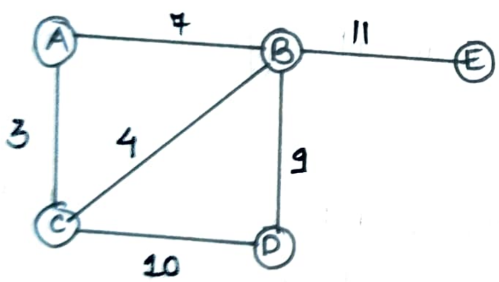
PRIMS ALGORITHM.

MALAY

- Prims algo. is a GREEDY ALGO.
- used to find or form a minimum spanning tree for a connected weighted undirected graph.
- for this, the algorithm maintains three sets of vertices which can be given as below :→

  - Tree Vertices ⟶ vertices that are part of MINIMUM SPANNING TREE.
  - Fringe Vertices ⟶ Vertices that are currently not a part of T, but are adjacent to some tree ~~vertices~~ vertex.
  - Unseen Vertices ⟶ Vertices that are neither tree vertices nor fringe vertices fall under this category.

- Running Time - O(E log V) → where E is the no. of edges and V is the no. of vertices in the graph.



Step 1: Choose a starting vertex A.

Step 2: Add the fringe vertices (that are adjacent to A). The edges connecting the vertices and fringe vertices are shown with dotted lines.

Step 3: Select an edge connecting the tree vertex and the fringe vertex that has the min. wt and add the selected edge & the vertex to the minimum spanning tree T.

Since the edge connecting A and C has less weight, add C to the tree. Now C is not a fringe vertex but a tree vertex.

Step 4: Add the fringe vertices (that are adjacent to C).

Step 5: Select an edge connecting the tree vertex and the fringe vertex that has the minimum weight and add the selected edge and the vertex to the min. spanning tree T.

Since the edge connecting C and B has less weight, add B to the tree.

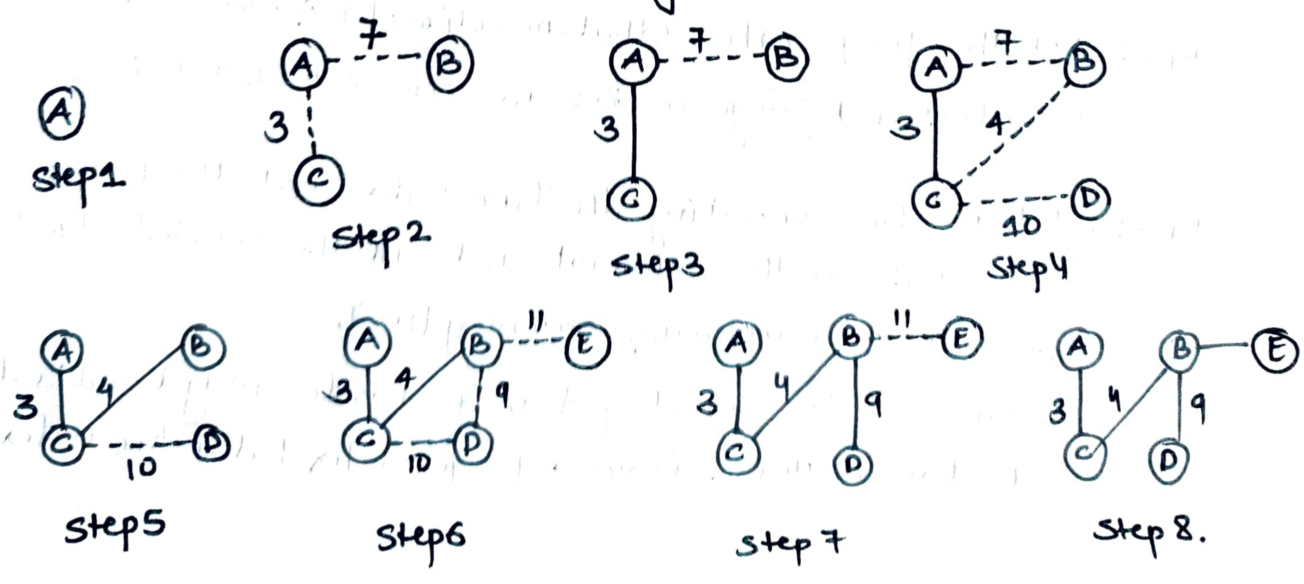
Now B is not a fringe vertex but a tree vertex.

Step 6:- Add the fringe vertices (that are adjacent to B).

Step 7:- Select an edge connecting the tree vertex and the fringe vertex that has the minimum weight and add the selected edge and the vertex to the minimum spanning tree T.

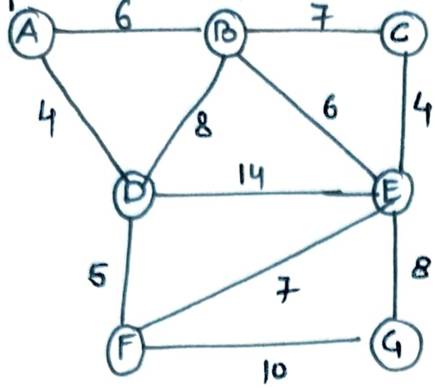
Since the edge connecting B and D has less weight, add D to the tree. Now D is not a fringe vertex but a tree vertex.

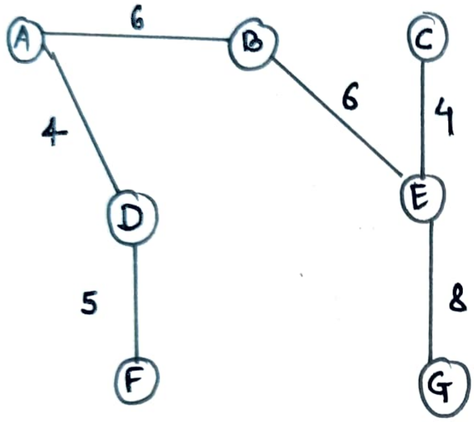
Step 8:- Note, now Node E is not connected, so we will add it in the tree because a minimum spanning tree is one in which all the n nodes are connected with n-1 edges that have minimum weight.



Q. Construct a minimum spanning tree of the graph. Start the Prim's algorithm from vertex D.

Malay





Step 1: Select a starting vertex

Step 2: Repeat Steps 3 and 4 until there are fringe vertices.

Step 3: Select an edge e connecting the tree vertex and fringe vertex that has minimum weight

Step 4: Add the select edge & the vertex to the minimum spanning tree T

[END OF LOOP]

Step 5: EXIT.

Malay

KRUSKAL'S ALGORITHM.

→ Kruskal's algorithm → is used to find the minimum spanning tree for a connected weighted graph.

The algo. aims to find a subset of the edges that forms a tree that includes every vertex.

The total weight of all the edges in the tree that includes every vertex.

The total weight of all the edges in the tree is minimized.

However, if the graph is not connected, then it finds a minimum spanning forests.

Note that a forest is a collection of trees. Similarly a minimum spanning forest is a collection of minimum spanning tree.

Kruskal's algo. is an example of a GREEDY ALGO., as it makes the locally optimal choice at each stage with the hope of finding the global optimum.

ALGO.

Step 1: Create a forest in such a way that each graph is a separate tree.

Step 2: Create a priority queue Q that contain all the edges of the graph.

Step 3: Repeat Step 4 and 5 while Q is NOT EMPTY.

Step 4: Remove an edge from Q.

Step 5: IF the edge obtained in Step 4 connects two diff. trees, then Add it to the forests (for combining two trees into one tree).

ELSE

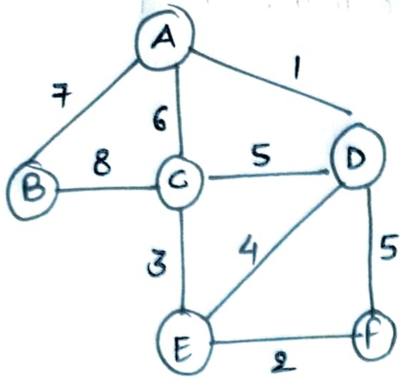
DISCARD the edge

Step 6: END:-

In the algo., we use a priority queue Q in which edges that have minimum weight takes a priority over any other edge in the graph. When the Kruskal's algorithm terminates, the forest has only one component & forms a ~~mst~~ MST of the graph.

RUNNING TIME = O(E log V), where E is the edges no. & V is the no. of vertices in the graph.

Malay



Apply Kruskal:-

Initialy, we have F = { {A}, {B}, {C}, {D}, {E}, {F} }.

MST = { }

Q = { (A,D), (E,F), (C,E), (E,D), (C,D), (D,F), (A,C), (A,B), (B,C) }

1

Step 1: Remove the edge (A,D) from Q & make the following changes :

F = { {A,D}, {B}, {C}, {E}, {F} }

MST = { A,D }

Q = { (E,F), (C,E), (E,D), (C,D), (D,F), (A,C), (A,B), (B,C) }

Step 2: Remove the edge (E,F) from Q & make the following changes

F = { {A,D}, {B}, {C}, {E,F} }

MST = { (A,D), (E,F) }

Q = { (C,E), (E,D), (C,D), (D,F), (A,C), (A,B), (B,C) }

Step 3: Remove the edge (C,E) from Q & make the following changes

F = { {A,D}, {B}, {C,E,F} }

MST = { (A,D), (C,E), (E,F) }

Q = { (E,D), (C,D), (D,F), (A,C), (A,B), (B,C) }

Step 4: Remove the edge (E, D) from Q & make the following changes:

F = {{A, C, D, E, F}, {B}}

MST = {(A, D), (C, E), (E, F), (E, D)}

Q = {(C, D), (D, F), (A, C), (A, B), (B, C)}

Step 5: Remove the edge (C, D) from Q. Note that this edge does not connect diff. trees, so simply discard this edge. Only an edge connecting (A, D, C, E, F) to B will be added to the MST.

F = {(A, D, C, E, F), {B}}

MST = {(A, D), (C, E), (E, F), (E, D)}

Q = {(D, F), (A, C), (A, B), (B, C)}

Step 6: Remove the edge (D, F) from Q. Note that this edge does not connect different trees, so simply discard this edge. Only an edge containing (A, D, C, E, F) to B will be added to the MST.

F = {A, C, D, E, F}, {B}}

MST = {(A, D), (C, E), (E, F), (E, D)}

Q = {(A, C), (A, B), (B, C)}

Step 7:- Remove the edge (A, C) from Q. Only an edge connecting (A, D, C, E, F) to B

F = {{A, C, D, E, F}, {B}}

MST = {(A, D), (C, E), (E, F), (E, D)}.

Q = {(A, B), (B, C)}.

Malay

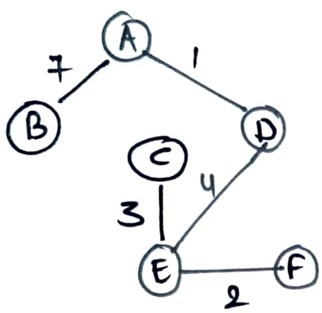
Step 8:- Remove the edge (A,B) from Q & make the changes

F = { A, B, C, D, E, F }

MST = { (A,D), (C,E), (E,F), (E,D), (A,B) }

Q = { (B,C) }.

Step 9: Algo continue until Q is empty.



F = { A, B, C, D, E, F }

MST = { (A,D), (C,E), (E,F), (E,D), (A,B) }

Q = { }.