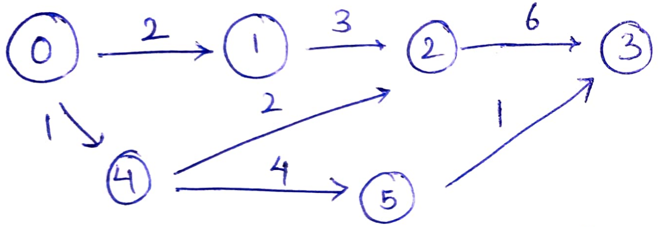
• **Shortest Path in Directed Acyclic Graph (Using DFS)**



src = 0

dist

{0, 2, 3, 6, 1, 5}

Approach: (1) Find Topological Sort using DFS or BFS

(2) Then, Pop out the elements from stack and check the distances with dist. array assign shortest values.

(3) Atlast return the dist array.

{NOTE : Adjacency list vector will have 2 values i.e. vertex value, weight; so for this we have add pair of int, int.}

```
• void shortestDist (V, adj, src) {
      Stack<int> st = getTopological sort(); ← To get the Topological sort.
      vector<int> dist(~~N~~, INT_MAX);
      dist [src] = 0;
      while (! st.empty()) {
          int node = st.top();
          st.pop();
          for (auto it : adj (node]) {
              if (dist (node] + it.second < dist (it.first]) {
                  dist (it.first] = dist [node] + it.second;
              }
          }
      }
      return dist;
  }
```

• **Minimum Spanning Trees (MST)**

↳ A Graph is said to be MST, only and only if the no. of nodes is 'n' and the no. of edges is 'n-1'.

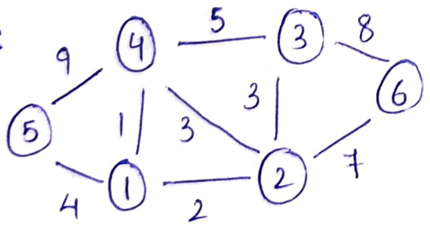
{NOTE: In MST, There is no cycle or loop.}

↳ A graph has to weighted.

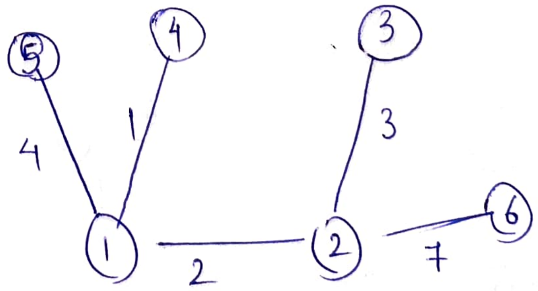
↳ All nodes should be connected directly or indirectly.

↳ ~~Node~~ cost for the graph should be minimum.

Example:



$\longrightarrow$



↳ To find the MST, there are 2 algo:

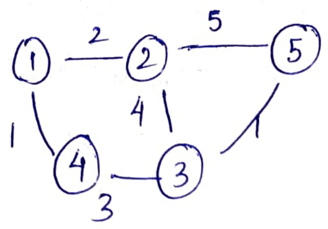
(i) Prim's Algorithm.

(ii) Kruskal's Algorithm.

Cost ⇒ 17

This is MST for the given graph.

• ~~Dija~~ Dijkstra's Algorithm



adj. list

1	(2,2), (4,1)
2	(1,2), (5,5), (3,4)
3	(2,4), (4,3), (5,1)
4	(1,1), (3,3)
5	(2,5), (3,1)

y, src = 1

1 - 1 → 0
1 - 2 → 2
1 - 3 → 4
1 - 4 → 1
1 - 5 → 5

Approach: (i) Here instead of using queue, we have to use priority queue (min heap) to get the minimum distance at the top of the PQ.
(ii) Pop out the element from the PQ and check for the minimum distances same as we did in (Shortest dist. in undirected graph) (unit weigh
(iii) If the distance of the adjacent nodes is change/minimize, we have to push that nodes in PQ and repeat the process until the PQ is not empty.

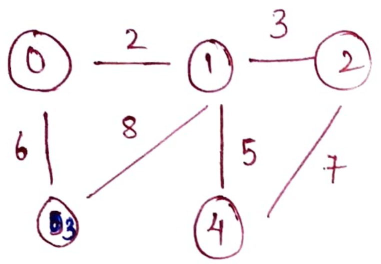
```
void djikstra (src, V, adj) {
    pq; ← min_heap (dist, val).
    dist (v+1, INT_MAX);
    dist [src] = 0;
    pq.push({0, src});
    while (! pq.empty()) {
        int wt = pq.top().front;
        int node = pq.top().second;
        for (auto it : adj[node]) {
            if (dist [node] + it.second < dist [it.first]) {
                dist [it.first] = dis[node] + wt;
                pq.push({ dist[it.frist], it.first });
            }
        }
    }
    return dist;
}
```

T.C → O((N+E) logN)
↳ O(NlogN)
S.C → O(N) + O(N)
↳ O(N).

Prim's Algorithm:

↓
Pick up an element and takes minimum of ~~that~~ all the adjacent nodes and mark that adjacent nodes as a part of MST.

• **Procedure :**



Key

	0	1	2	3	4
		2	3	6	5
	0	~~∞~~	~~∞~~	~~∞~~	~~∞~~

MST

	0	1	2	3	4
	T	T	T		
	~~F~~	~~F~~	~~F~~	F	F

Parent

	0	1	2	3	4
		0	1	0	1
	-1	~~-1~~	~~-1~~	~~-1~~	~~-1~~

Starting from 0,

$\boxed{\text{node} = 0}$

↓

(1)
(3)

} from Adjacency list, Just mark the min. edge and parent.

Now, check for the minimum edge in Key array.

$\boxed{\text{node} = 1}$

↓

(0) → We will not consider this, as this is already a part of MST

(3) → No need to consider this, 6 is already minimum (better than 8).

(4)

(2)

$\boxed{\text{node} = 2}$,

(1) → No need to consider.

(4) → No need to consider.

After doing this again & again we will have MST array have value True, which means MST is created.

• **Implementation of Prim's Algo.**

```
void prims(int n , vector<int> adj[]) {
   vector<int> parent(n,-1), mst(n,false), key(n, INT_MAX);
   priority queue pq ← min heap.
   pq.push({0,0});
   Key[0]=0;
   for(int c=0; c< N-1; c++) {
        int u = pq.top().second;
        pq.pop();
        mst[u] = true;
        ~~it~~ for(auto it : adj[u]) {
             int v = it.first;
             int wt = it.second;
             if (~~adj~~ mst[v]==false && wt < key[v]) {
                  key[v] = wt;    parent[v]=u;
                  pq.push({key[v], v});
             }
        }
   }
   for (int i=1; i<n; i++)
        cout << parent[i] <<"-" << i <<endl;
}
```

{ T.C → O(N log N)
  S.C → O(N) }