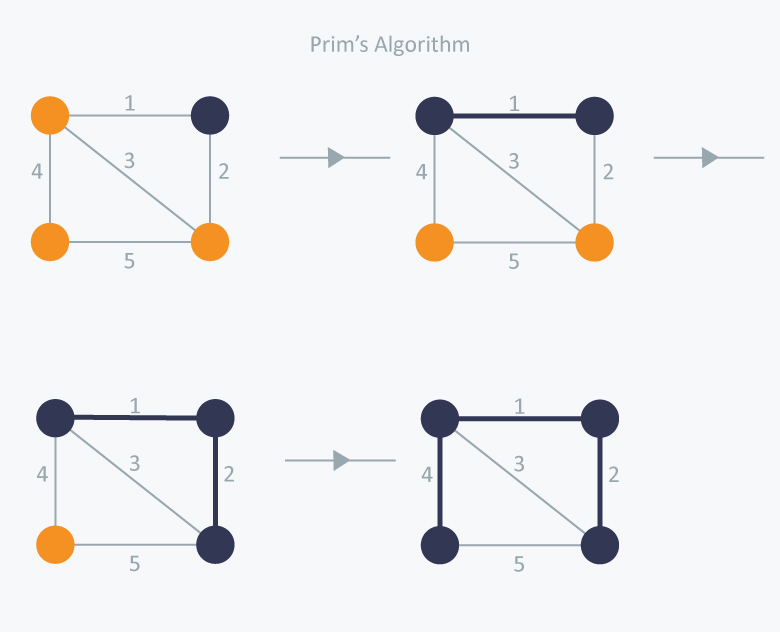
**Prim’s Algorithm:**

**Prim’s Algorithm also use Greedy approach to find the minimum spanning tree. In Prim’s Algorithm we grow the spanning tree from a starting position.**

**Algorithm Steps:**

* **Maintain two disjoint sets of vertices. One containing vertices that are in the growing spanning tree and other that are not in the growing spanning tree.**
* **Select the cheapest vertex that is connected to the growing spanning tree and is not in the growing spanning tree and add it into the growing spanning tree. This can be done using Priority Queues. Insert the vertices, that are connected to growing spanning tree, into the Priority Queue.**
* **Check for cycles. To do that, mark the nodes which have been already selected and insert only those nodes in the Priority Queue that are not marked.**

**Consider the example below:**

****

**In Prim’s Algorithm, we will start with an arbitrary node (it doesn’t matter which one) and mark it. In each iteration we will mark a new vertex that is adjacent to the one that we have already marked.**

**As a greedy algorithm, Prim’s algorithm will select the cheapest edge and mark the vertex. So we will simply choose the edge with weight 1.**

**In the next iteration we have three options, edges with weight 2, 3 and 4. So, we will select the edge with weight 2 and mark the vertex. Now again we have three options, edges with weight 3, 4 and 5. But we can’t choose edge with weight 3 as it is creating a cycle.**

**So we will select the edge with weight 4 and we end up with the minimum spanning tree of total cost 7 ( = 1 + 2 +4).**

**COMPLEXITY ANALYSIS OF SERIAL PRIMS ALGORITHM:**

**Prim's algorithm**finds a minimum spanning tree for a weighted undirected graph . It finds a subset of the edges that forms a tree which includes every vertex, where the total weight of all the edges in the tree is minimized.

1. MST-PRIM(G, w, r)
2. **for** each u ∈ G.V
3. u.key ← ∞
4. u.π ← NIL
5. r.key ← 0
6. Q ← G.V
7. **while** Q ≠ Ø
8. u ← EXTRACT-MIN(Q)
9. **for** each v ∈ G.Adj[u]
10. **if** v ∈ Q **and** w(u, v) < v.key
11. v.π ← u
12. v.key ← w(u, v)
13. Here you can see that, time required for one call to EXTRACT-MIN(Q) = O(log V) [using min priority queue]. The while loop at line 6 is executing total V times. So EXTRACT-MIN(Q) is called V times. So total time required for EXTRACT-MIN(Q) = O(VlogV).
14. the for loop at line 8 is executing total 2E times ( as total length of all the adjacency lists=2E for undirected graph ) . Time required for executing line 11=O(log v) [using DECREASE\_KEY operation on the min heap]. Line 11 is executing total 2E times. So total time required to execute line 11=O(2Elog V) = O(Elog v).
15. The for loop at line 1 will be executed for V times. Using BUILD\_HEAP procedure, to perform lines 1 to 5, it will require O(V) times

Total time complexity of MST-PRIM =

time required for executing 1 + time required for executing 2 + time required for executing 3 =

Total : O( VlogV + ElogV + V) = O(ElogV)