DESIGN ANALYSIS AND ALGORITHM

PRACTICAL NO 7

027_Abhishek_ Ojha

Experiment No:- 7 **Date of Experiment: - 23 October 2021**

Program: -Write a program to Implement Krushal's Algorithm.

Algorithm

Krushal's Algorithm

1. Begin

2. Create the edge list of given graph, with their weights.

3. Sort the edge list according to their weights in ascending order.

4. Draw all the nodes to create skeleton for spanning tree.

5. Pick up the edge at the top of the edge list (i.e. edge with minimum weight).

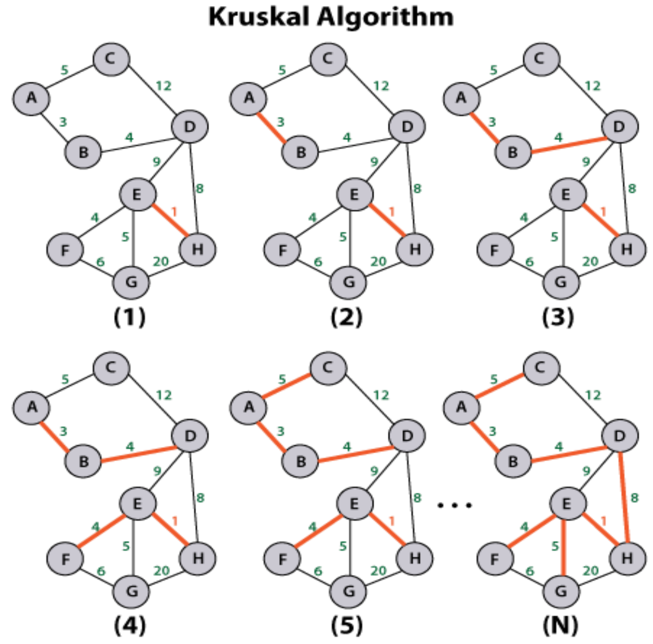
6. Remove this edge from the edge list.

7. Connect the vertices in the skeleton with given edge. If by connecting the vertices, a cycle is created in the skeleton, then discard this edge.

8. Repeat steps 5 to 7, until n-1 edges are added or list of edges is over.

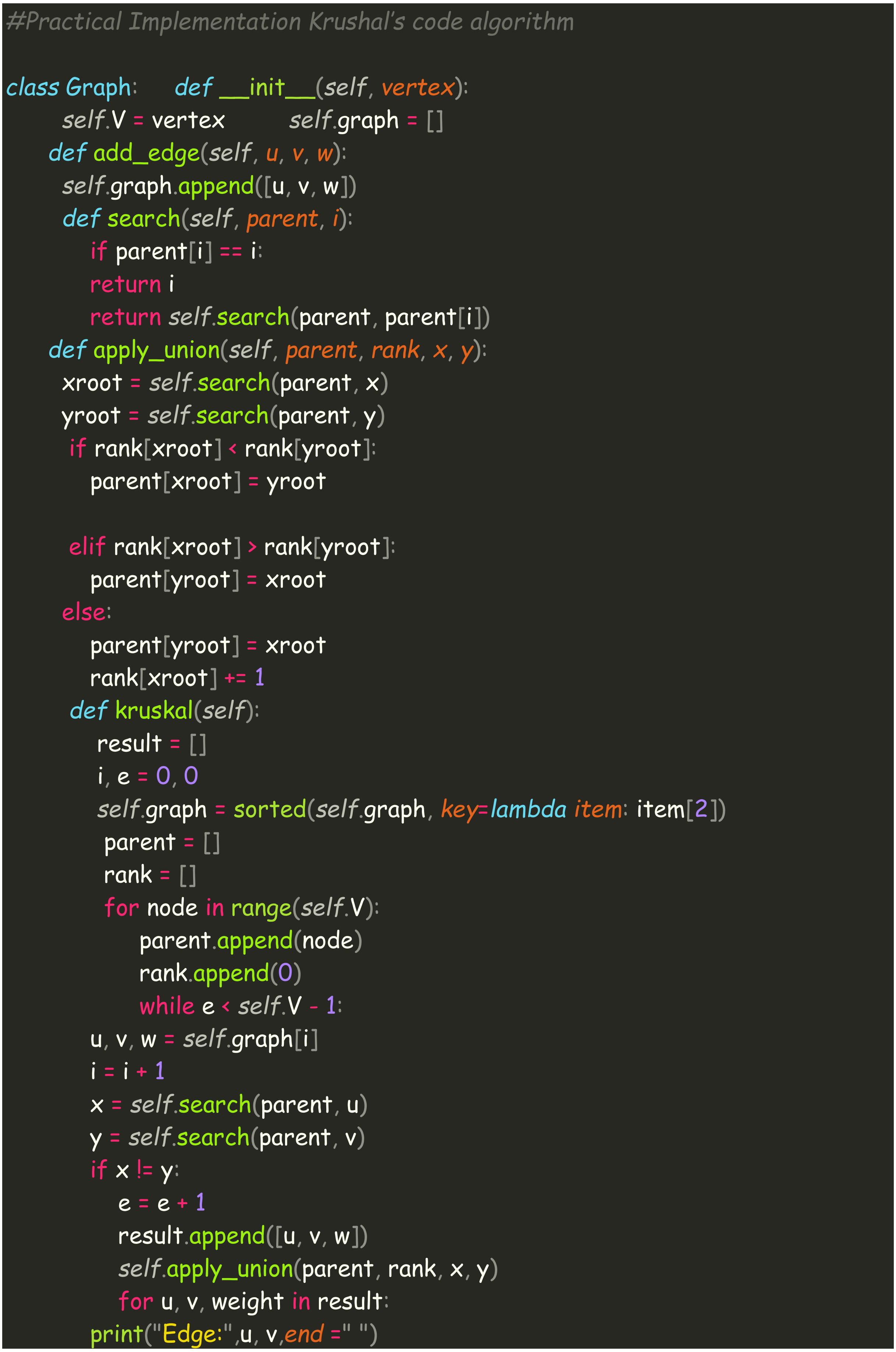
9. Return.

Fig:

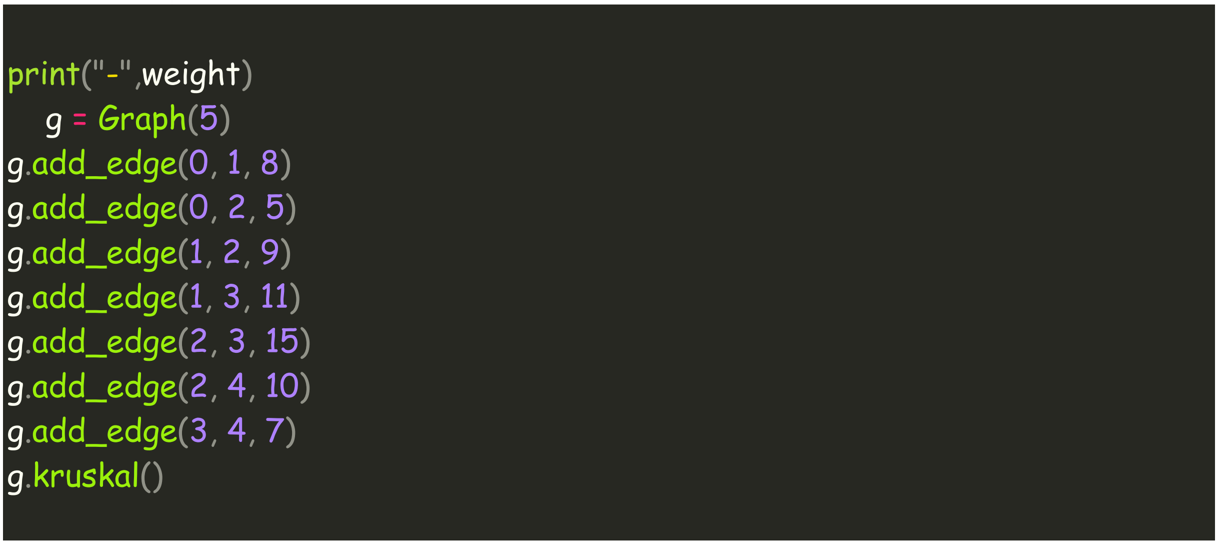


(1) (2) (3)

(4) (5) (N)

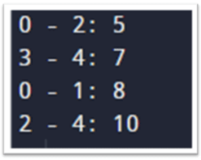


```
#Practical Implementation Krushal's code algorithm

class Graph:     def __init__(self, vertex):
    self.V = vertex        self.graph = []
   def add_edge(self, u, v, w):
    self.graph.append([u, v, w])
    def search(self, parent, i):
      if parent[i] == i:
      return i
      return self.search(parent, parent[i])
   def apply_union(self, parent, rank, x, y):
    xroot = self.search(parent, x)
    yroot = self.search(parent, y)
     if rank[xroot] < rank[yroot]:
      parent[xroot] = yroot

     elif rank[xroot] > rank[yroot]:
      parent[yroot] = xroot
    else:
      parent[yroot] = xroot
      rank[xroot] += 1
     def kruskal(self):
      result = []
      i, e = 0, 0
      self.graph = sorted(self.graph, key=lambda item: item[2])
       parent = []
       rank = []
       for node in range(self.V):
         parent.append(node)
         rank.append(0)
         while e < self.V - 1:
      u, v, w = self.graph[i]
      i = i + 1
      x = self.search(parent, u)
      y = self.search(parent, v)
      if x != y:
        e = e + 1
        result.append([u, v, w])
        self.apply_union(parent, rank, x, y)
        for u, v, weight in result:
      print("Edge:",u, v,end =" ")
```



```
print("-",weight)
   g = Graph(5)
g.add_edge(0, 1, 8)
g.add_edge(0, 2, 5)
g.add_edge(1, 2, 9)
g.add_edge(1, 3, 11)
g.add_edge(2, 3, 15)
g.add_edge(2, 4, 10)
g.add_edge(3, 4, 7)
g.kruskal()
```

Output:



```
0 - 2: 5
3 - 4: 7
0 - 1: 8
2 - 4: 10
```

The time complexity for Krushal's Algorithm is O(E log V)

Conclusion: Successfully Implemented the Krushal's Algorithm.