Assignment No. 3.

◉ Implement greedy search algorithm for any of the following applicatn:-

kruskals minimal spanning tree algorithm

= Aim:- Implement greedy search algorithm for kruskal's minimal spanning tree algorithm.

= Theory:- what is a greedy algorithm ?

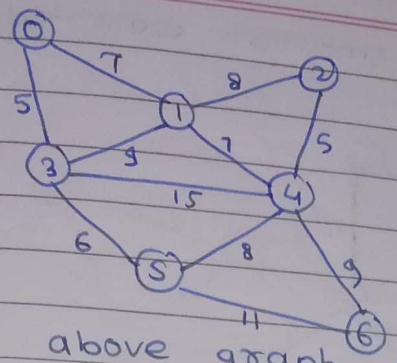
- In greedy algorithm a set of resources are recursively divided based on maximum, immediate availability of that resources at any given stage of execution.
- To solve a problem based on greedy approach
  1) scanning the list of items.
  2) optimizatn.

= characteristics of greedy algorithm :-

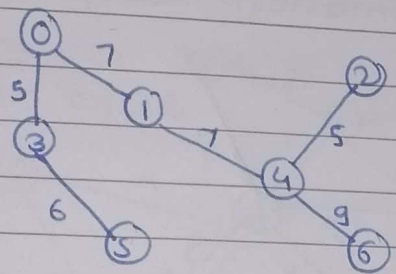
- There is an ordered list of resource, with cost or value attributions. These quantify constraints on a system.
- You will take the maximum quantify of resource in the constraints applies.
- for example, in an activity scheduling problem

= kruskals algorithm for finding minimum spanning tree:-

- given a connected & weighted undirected graph construct a minimum spanning tree.
- A minimum spanning tree is a spanning tree of connected, undirected graph.



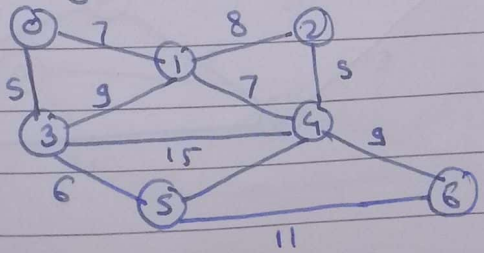
- consider above graph. Its minimum spanning tree will be following tree with exactly n-1 edges where n = total no. of vertices.



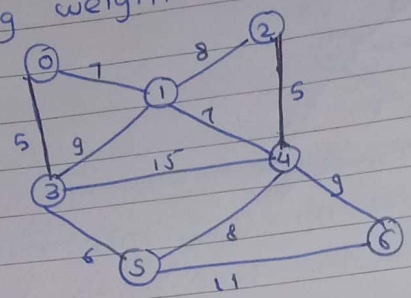
- we can use kruskal's minimum spanning tree algorithm, a greedy algorithm to find spanning tree for connected weighted graph.
- Let G = (V, E) be given graph initially our MST contains only vertices of given graph with no edges. The goal is to add minimum weight edge to our MST.

Sort all edges in graph G in order of their increasing weights; repeat V-1 times.

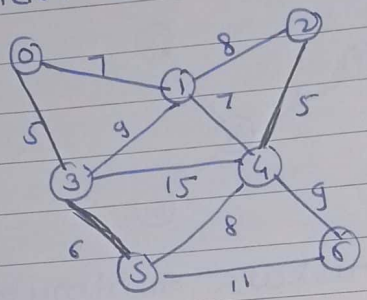
- Let's illustrate this by taking example of above graph. we start by considering smallest weighted 0-3 having weight 5



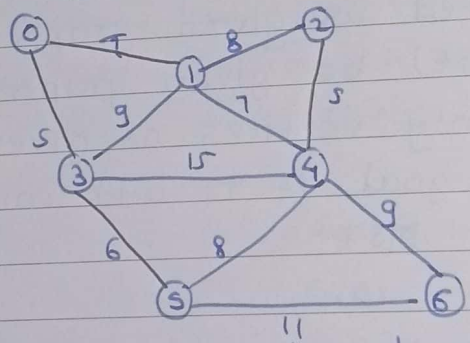
- we next consider smallest weighted edge 2-4 also having weight 5.



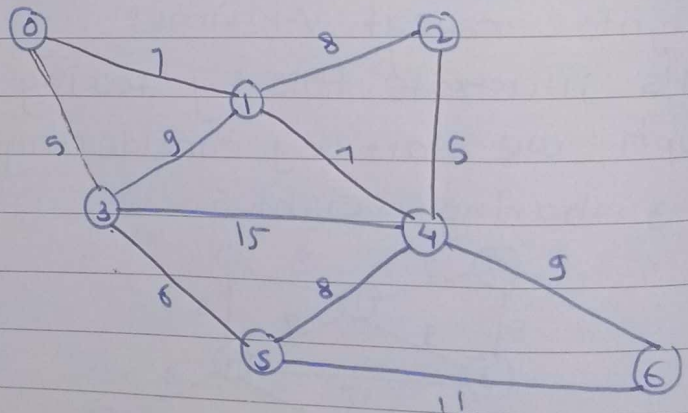
- next consider smallest weighted edge 3-5



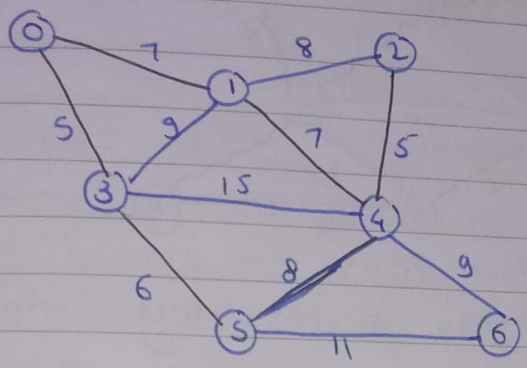
- smallest weighted edge 0-1 having wt 7.



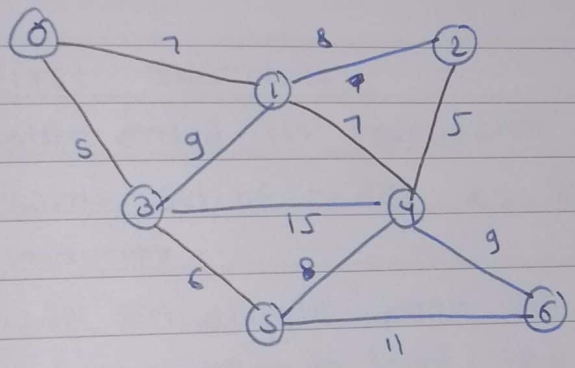
- next consider smallest wt edge 1-4



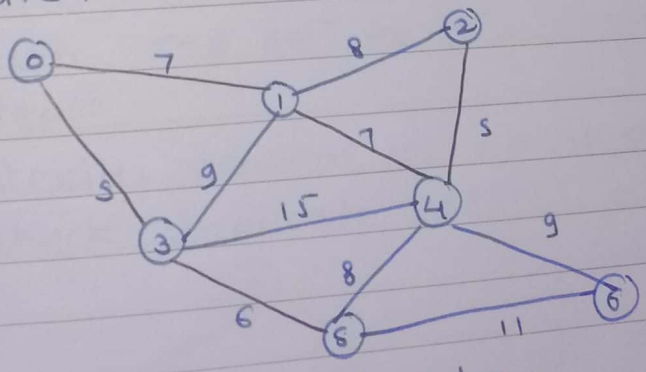
- next consider smallest wt 5-4 having wt 8.
  0-1-4-5-3-0



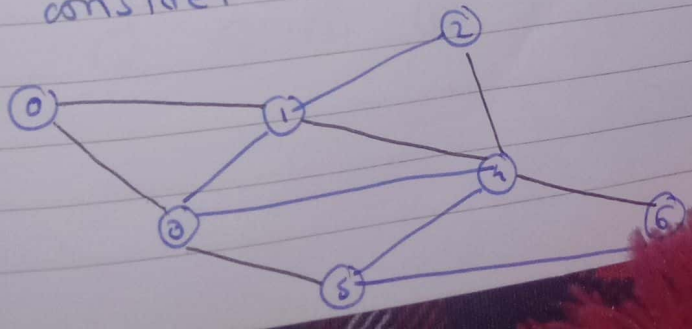
- next consider smallest wt edge 1-2 having wt 8 result in cycle 1-2-4-1



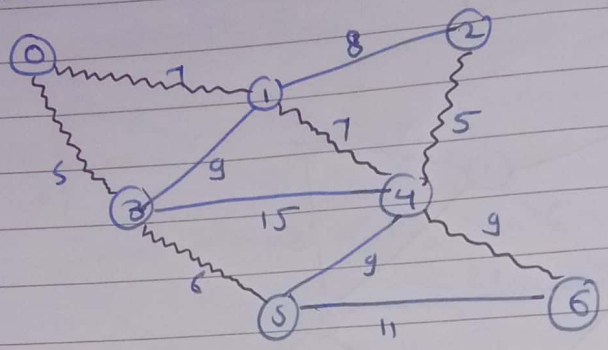
- next consider 3-1 edge. but result in 0-1-3-0 so discard it.



- finally consider 4-6 edge.



• MST is now connected.



= conclusion :- Thus successfully implemented kruskal's minimum spanning tree.

```java
import java.util.*;
class Edge{

int src, dest, weight;

public Edge(int src, int dest, int weight)

{

    this.src= src;
    this.dest = dest;
    this.weight = weight;
}

public String toString() {

return "(" +src + "." + dest +"." + weight + ")";
}
}
class DisjointSet
{
Map<Integer,Integer> parent = new HashMap<>();

public void makeSet(int n)
```

Main.java

```
24  public void makeSet(int n)
25  {
26      for(int i = 0; i<n; i++) {
27      parent.put(i,i);
28      }
29  }
30
31  private int find(int k)
32  {
33
34      if(parent.get(k)==k)
35      {
36          return k;
37
38      }
39      return find(parent.get(k));
40  }
41  private void union(int a,int b)
42  {
43       int x = find(a);
44
45      int y = find(b);
46
47  parent.put(x, y);
48
```

Main.java

Run

```
49  }
50
51  public static List<Edge> runKruskalAlgorithm(List<Edge> edges, int n) {
52      DisjointSet ds=new DisjointSet();
53
54  List<Edge> MST = new ArrayList<>();
55
56  ds.makeSet(n);
57
58  int index = 0;
59  Collections.sort(edges,Comparator.comparingInt(e-> e.weight));
60
61  while (MST.size() != n-1) {
62      Edge next_edge=edges.get(index++);
63
64   int x = ds.find(next_edge.src);
65
66  int y = ds.find(next_edge.dest);
67
68  if (x !=y)
69  {
70
71  MST.add(next_edge);
72  ds.union(x, y);
```

35°C
Mostly cloudy

Main.java

```
72  ds.union(x, y);
73
74  }
75
76  }
77
78  return MST;
79
80  }
81
82  }
83
84  class Main
85
86  {
87
88  public static void main(String[] args)
89
90  {
91
92  List<Edge> edges =Arrays.asList(new Edge(0, 1, 7), new Edge (1, 2, 8), new
        Edge(0, 3, 5), new Edge(1, 3, 9), new Edge(1, 4, 7), new Edge (2, 4, 5),
        new Edge(3, 4, 15), new Edge(3, 5, 6), new Edge(4, 5, 8),new Edge(4,6,9
        ),new Edge(5,6,11));
93
```

Main.java

```
87
88  public static void main(String[] args)
89
90  {
91
92  List<Edge> edges =Arrays.asList(new Edge(0, 1, 7), new Edge (1, 2, 8), new
        Edge(0, 3, 5), new Edge(1, 3, 9), new Edge(1, 4, 7), new Edge (2, 4, 5),
        new Edge(3, 4, 15), new Edge(3, 5, 6), new Edge(4, 5, 8),new Edge(4,6,9
        ),new Edge(5,6,11));
93
94
95  int n=7;
96
97  List<Edge>e=DisjointSet.runKruskalAlgorithm(edges, n);
98  System.out.println(e);
99
100 }
101 }
102
103
104
```

Run

Output

```
java -cp /tmp/js9jTvG5UL Main
[(0.3.5), (2.4.5), (3.5.6), (0.1.7), (1.4.7), (4.6.9)]
```

```
ew Edge (1, 2, 8), new
 7), new Edge (2, 4, 5),
4, 5, 8),new Edge(4,6,9


 n);
```