**Task 1**

1. I use **Adjacency List to** implement graph in my program, because **Adjacency List is more memory efficient in compare to Adjacency matrix.**

****

|  |  |
| --- | --- |
| **Node** | **Adjacency List** |
| **0** | **1,7** |
| **1** | **0,2,7** |
| **2** | **1,3,5,8** |
| **3** | **2,4** |
| **4** | **3,5** |
| **5** | **2,3,4,6** |
| **6** | **5,7,8** |
| **7** | **0,1,6,8** |
| **8** | **2,6,7** |

|  |
| --- |
| **0** |
| **1** |
| **2** |
| **3** |
| **4** |
| **5** |
| **6** |
| **7** |
| **8** |

1

7

\0

2

0

7

\0

8

\0

5

3

1

4

\0

2

4

3

2

5

\0

3

8

\0

7

5

6

\0

8

\0

6

1

0

7

\0

6

2

**Task 1**

**2)**

Step1 > sort all the edges in non-decreasing order of their weight

**Unsort:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Edge | 0,1 | 0,7 | 1,7 | 1,2 | 7,6 | 7,8 | 2,8 | 8,6 | 2,3 | 2,5 | 6,5 | 5,3 | 3,4 | 5,4 |
| Wt | 4 | 8 | 11 | 8 | 1 | 7 | 2 | 6 | 7 | 4 | 2 | 14 | 9 | 10 |

**Sort:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Edge | 7,6 | 8,2 | 6,5 | 0,1 | 2,5 | 8,6 | 2,3 | 7,8 | 0,7 | 1,2 | 3,4 | 5,4 | 1,7 | 3,5 |
| Wt | 1 | 2 | 2 | 4 | 4 | 6 | 7 | 7 | 8 | 8 | 9 | 10 | 11 | 14 |

Step2> pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed include this edge else, discard it.

Repeat step 2 until there are (V-1) in the spanning.

1)pick edge7-6

1

2) pick edge8-2

2

1

3) pick edge6-5

2

1 2

4) pick edge0-1

4 2

1 2

5) pick edge2-5

4 2

4

1 2

6) pick edge7-6 : this edge make a cycle discard it.

7) pick edge7-6

7

4 2

4

1 2

8) pick edge7-8: this edge make a cycle discard it.

9) pick edge0-7

7

4 2

4

8

1 2

10) pick edge1-2: this edge make a cycle discard it.

11) pick edge3-4

7

4 9

2 4

8

1 2

**Task 1**

**3)**

KRUSKAL(G):

1 A = ∅

2 foreach v ∈ G.V:

3 MAKE-SET(v)

4 foreach (u, v) in G.E ordered by weight(u, v), increasing:

5 if FIND-SET(u) ≠ FIND-SET(v):

6 A = A ∪ {(u, v)}

7 UNION(u, v)

8 return A

**Task 1**

**4)** advantages and dis-advantages of KRUSKAL algorithm

**Advantages**

1. Easy to understand.
2. Give good result for large number of vertices and edges.

**Disadvantages**

1. Difficulty of checking whether arcs form cycles makes it slow and hard to program.
2. Same weight may increase the complexity.