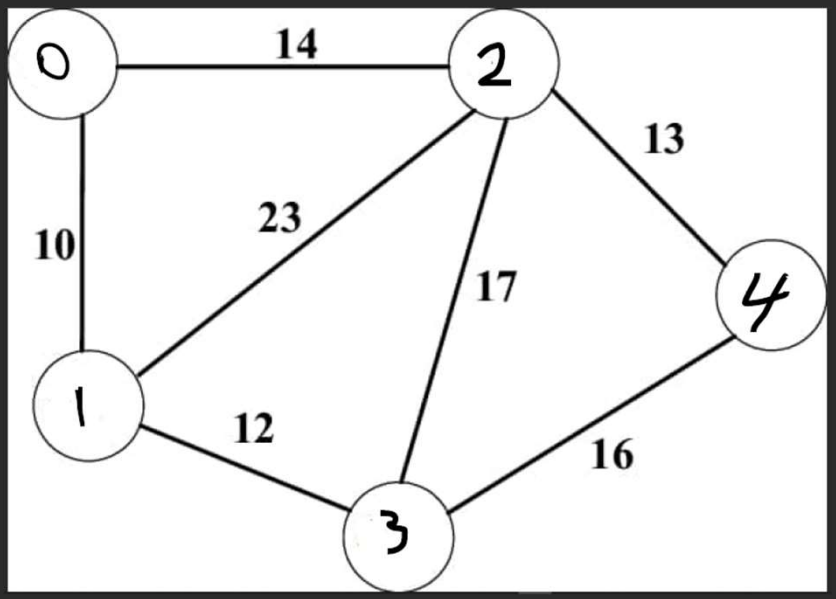
علی کریمی ۹۹۲۹۷۳۳

شرح پروژه پایانی ساختمان داده

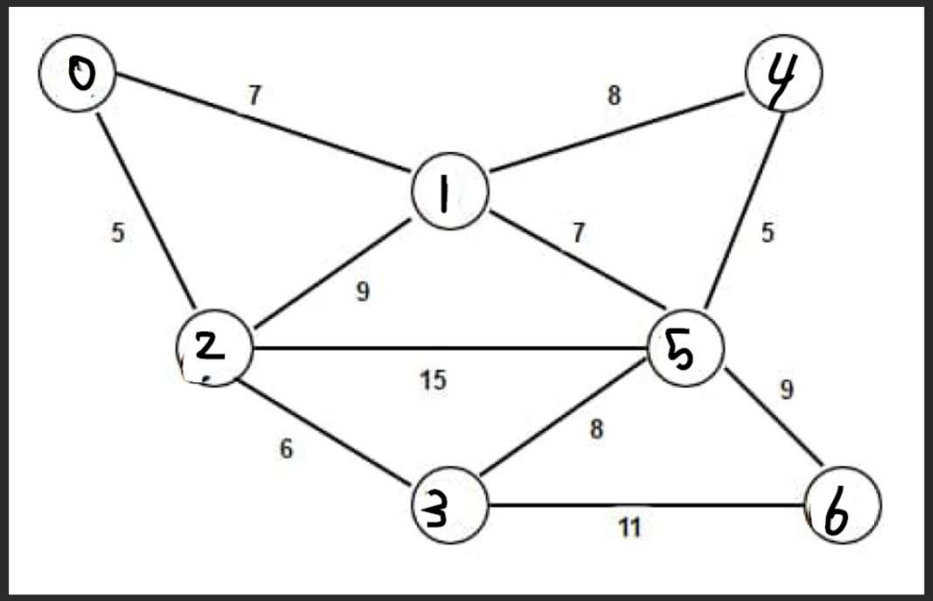
شماره پروژه: پروژه کراسکال

ابتدا مثال ها را بررسی میکنیم



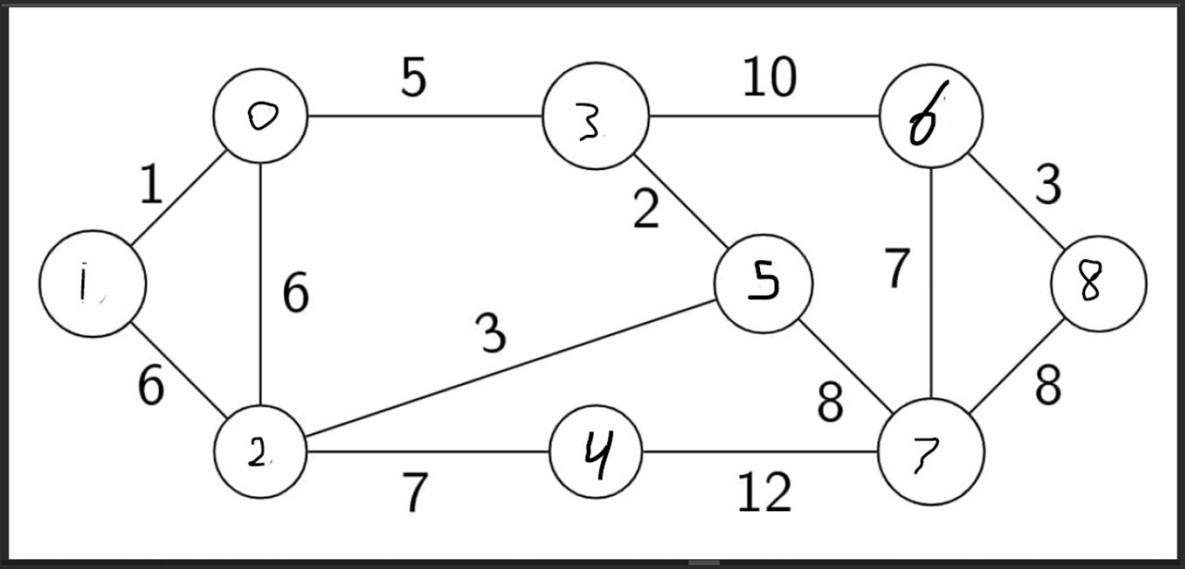

```
Run:  Main
        Structure\project\Ali_Karimi_kr
Enter weighted edges line by line and
0 1 10
0 2 14
1 2 23
1 3 12
2 3 17
3 4 16
2 4 13
end
---> MST:
0 1 10
1 3 12
2 4 13
0 2 14
MST cost : 49

Process finished with exit code 0
```




```
Project
Run: Main
Enter weighted edges line by line and En
0 1 7
0 2 5
1 2 9
1 4 8
4 5 5
5 6 9
6 3 11
5 3 8
2 3 6
2 5 15
1 5 7
end
---> MST:
0 2 5
4 5 5
2 3 6
0 1 7
1 5 7
5 6 9
MST cost : 39

Process finished with exit code 0
```



Run: Main

```
2 4 7
2 5 3
4 7 12
5 7 8
0 3 5
3 5 2
7 6 7
3 6 10
6 8 3
7 8 8
end
---> MST:
0 1 1
3 5 2
2 5 3
6 8 3
0 3 5
2 4 7
7 6 7
5 7 8
MST cost : 36

Process finished with exit code 0
```

راه حل را توضیح میدهیم

```java
public class Console {
    private static final Scanner SCANNER = new Scanner(System.in);
```

کلاس console صرفا برای ورودی و خروجی گرفتن است و خیلی فرقی ندارد که به اسم vertex ها عدد بدهیم یا حروف بزرگ انگلیسی اما ما در برنامه باید اسم vertex ها را عدد بدهیم

```java
public class Director {
    Console console;

    public Director() {
        console = new Console();
    }

    public void start() {
        Graph graph = getGraph();
        ArrayList<Edge> kruskalMST = graph.getKruskalMST();
        console.printMST(kruskalMST);
    }
```

در کلاس Director تمام کار های برنامه از قبیل خواندن و چاپ کردن و صدا زدن تابع getKruskalMST و ساخت یک آبجکت از Graph
که دارای یک فیلد console است و قسمت getGraph صرفا

منطق اصلی در getKruskalMST پیاده سازی شده است که ابتدا یک ArrayList برای mst درست میکنیم بعد node ها را مقدار دهی اولیه میکنیم

```java
public ArrayList<Edge> getKruskalMST() {
    ArrayList<Edge> mst = new ArrayList<>();
    initNodes();
    sortEdges();
    for (Edge edge : edges) {
        if (mst.size() >= nodes.length - 1) {
            break;
        }
        int xTop = findTopSimple(edge.x);
        int yTop = findTopSimple(edge.y);
        if (xTop != yTop) {
            mst.add(edge);
            mergeClouds(xTop, yTop);
        }
    }
    return mst;
}
```

Main | Edge | Node

```java
class Node {
    int next;
    int rank;

    public Node( int next,
                 int rank ) {
        this.next = next;
        this.rank = rank;
    }
}
```

سپس sortEdges میکنیم

چون کلاس Edges اینترفیس comparable را implement میکند
پس وقتی sort را فراخوانی میکنیم sort بر اساس وزن انجام میشود

```java
class Edge implements Comparable<Edge> {
    int x;
    int y;
    int weight;

    public int compareTo( Edge compareEdge ) {
        return this.weight - compareEdge.weight;
    }

    @Override
    public String toString() {
        return String.format("%d %d %d",
                             x,
                             y,
                             weight);
    }
}
```

```java
private void sortEdges() {
    Arrays.sort(edges);
}
```

```java
public ArrayList<Edge> getKruskalMST() {
    ArrayList<Edge> mst = new ArrayList<>();
    initNodes();
    sortEdges();
    for (Edge edge : edges) {
        if (mst.size() >= nodes.length - 1) {
            break;
        }
        int xTop = findTopSimple(edge.x);
        int yTop = findTopSimple(edge.y);
        if (xTop != yTop) {
            mst.add(edge);
            mergeClouds(xTop, yTop);
        }
    }
    return mst;
}
```

بعد تا زمانی که mst ما تمام درخت را شامل نشده روی iterate edges می کنیم و هربار که کمترین وزن را اضافه میکنیم و بعد اگر که در یک cloud نباشد یعنی xtop != ytop باشد آن edge را اضافه میکنیم و دو ابر را با هم merge میکنیم روش کار findTopSimple را توضیح میدهیم

```java
private int findTopSimple( int v ) {
    int top = v;
    while (top != nodes[top].next) {
        top = nodes[top].next;
    }
    return top;
}
```

متد findTopSimple از روش pathcompresion استفاده نمیکند ولی find top از path compression استفاده میکند یعنی سرعت کار بالا تر میرود

```java
private int findTop( int v ) {
    if (nodes[v].next == v) {
        return v;
    }
    int top = findTop(nodes[v].next);
    nodes[v].next = top;
    return top;
}
```

متد mergeCloud هم مانند توضیحات اسلاید است و از روش رنکینگ استفاده شده تا برنامه در O کمتری اجرا شود

```
private void mergeClouds( int xTop,
                          int yTop ) {
    Node x = nodes[xTop];
    Node y = nodes[yTop];

    if (x.rank < y.rank) {
        x.next = yTop;
    } else if (x.rank > y.rank) {
        y.next = xTop;
    } else {
        x.next = yTop;
        y.rank++;
    }
}
```