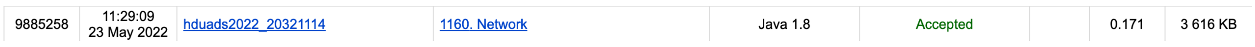
Laboratory work # 8

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1160. Networks

Screenshot from Timus:



9885258	11:29:09 23 May 2022	hduads2022_20321114	1160. Network	Java 1.8	Accepted		0.171	3 616 KB

Explanation of algorithm:

1. Use the CMP algorithms to sort the node array.
2. Find the minimum spanning tree.

Computational complexity of algorithm:

$$O(N \log N)$$

Source code:

```
import java.io.*;
import java.util.ArrayList;

public class Networks {

    StreamTokenizer in;
    PrintWriter out;
    static int M, N;
    ArrayList<Node> Q1 = new ArrayList<>();
    ArrayList<Node> Q2 = new ArrayList<>();
    int[] P = new int[2000];

    public static void main(String[] args) throws IOException {
        new Networks().run();
    }

    int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;
    }

    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
```

```java
        out = new PrintWriter(System.out);
        solve();
        out.flush();
    }

    int CMP(Node n1, Node n2) {
        return n1.getW() < n2.getW() ? -1 : 1;
    }

    int find(int x) {
        if (P[x] == x) {
            return x;
        } else {
            return P[x] = find(P[x]);
        }
    }

    void prepare() throws IOException {
        N = nextInt();
        M = nextInt();
        for (int i = 0; i < M; i++) {
            Node n = new Node(nextInt(), nextInt(), nextInt());
            Q1.add(n);
        }
    }

    void solve() throws IOException {
        int ans = 0;
        prepare();
        Q1.sort(this::CMP);
        for (int i = 0; i <= N; i++) {
            P[i] = i;
        }
        for (int i = 0; i < M; i++) {
            int x = find(Q1.get(i).getU());
            int y = find(Q1.get(i).getV());
            if (x != y) {
                ans = Q1.get(i).getW();
                Q2.add(Q1.get(i));
                P[x] = y;
            }
        }
        out.println(ans);
        out.println(Q2.size());
        for (Node node : Q2) out.println(node.getU() + " " + node.getV());
    }
}

class Node {
    private final int u, v, w;

    public Node(int u, int v, int w) {
        this.u = u;
        this.v = v;
        this.w = w;
    }

    public int getU() {
        return u;
    }

    public int getV() {
        return v;
    }
    public int getW() {
        return w;
    }
}
```