1647 - 도시분할계획

난이도	골드4

문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	13191	6631	4776	49.518%

문제

동물원에서 막 탈출한 원숭이 한 마리가 세상구경을 하고 있다. 그러다가 평화로운 마을에 가게 되었는데, 그곳에서는 알 수 없는 일이 벌어지고 있었다.

마을은 N개의 집과 그 집들을 연결하는 M개의 길로 이루어져 있다. 길은 어느 방향으로든지 다닐 수 있는 편리한 길이다. 그리고 각 길마다 길을 유지하는데 드는 유지비가 있다.

마을의 이장은 마을을 두 개의 분리된 마을로 분할할 계획을 가지고 있다. 마을이 너무 커서 혼자서는 관리할 수 없기 때문이다. 마을을 분할할 때는 각 분리된 마을 안에 집들이 서로 연결되도록 분할해야 한다. 각 분리된 마을 안에 있는 임의의 두 집 사이에 경로가 항상 존재해야 한다는 뜻이다. 마을에는 집이 하나 이상 있어야 한다.

그렇게 마을의 이장은 계획을 세우다가 마을 안에 길이 너무 많다는 생각을 하게 되었다. 일단 분리된 두 마을 사이에 있는 길들은 필요가 없으므로 없앨 수 있다. 그리고 각 분리된 마을 안에서도 임의의 두 집 사이에 경로가 항상 존재하게 하면서 길을 더 없앨 수 있다. 마을의 이장은 위 조건을 만족하도록 길들을 모두 없애고 나머지 길의 유지비의 합을 최소로 하고 싶다. 이것을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 집의 개수 N, 길의 개수 M이 주어진다. N은 2이상 100,000이하인 정수이고, M은 1이상 1,000,000이하인 정수이다. 그 다음 줄부터 M줄에 걸쳐 길의 정보가 A B C 세 개의 정수로 주어지는데 A번 집과 B번 집을 연결하는 길의 유지비가 C ($1 \le C \le 1,000$)라는 뜻이다.

출력

첫째 줄에 없애고 남은 길 유지비의 합의 최솟값을 출력한다.

예제 입력 1 복사

```
7 12
1 2 3
1 3 2
3 2 1
2 5 2
3 4 4
7 3 6
5 1 5
1 6 2
6 4 1
6 5 3
4 5 3
6 7 4
```

예제 출력 1 복사

```
8
```

풀이과정

크루스칼 알고리즘으로 구현한 MST(최소신장트리)로 마을을 구성한 뒤, 길의 유지비 중 가장 큰 길 하나만 지우면 된다.

💡 주요 포인트

1. 마을을 최소신장트리의 구조로 바꿔가면서, 유지비가 가장 큰 길을 갱신해가자.

마을을 MST로 바꿔가며 answer에 유지비를 누적해서 더해준 뒤, 해당 MST 중 유지비가 가장 큰 길을 answer에서 빼줄 것이다.

소스코드

```
package Graph.MST.BOJ1647;

import java.io.*;
import java.util.Arrays;
import java.util.StringTokenizer;

class Edge implements Comparable<Edge> {
    int from;
    int to;
    int cost;

    public Edge(int from, int to, int cost) {
        this.from = from;
        this.to = to;
        this.cost = cost;
    }

    @Override
    public int compareTo(Edge o) {
        return Integer.compare(cost, o.cost);
    }
}

public class Main {

    static long answer;
    static int N,M;
    static int[] group;
    static Edge[] edges;

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));

        StringTokenizer st = new StringTokenizer(br.readLine());

        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());

        edges = new Edge[M+1];
        group = new int[N+1];
        for(int i=1 ; i<=N ; i++)
            group[i] = i;

        int a,b,c;
        for(int i=1 ; i<=M ; i++) {
            st = new StringTokenizer(br.readLine());

            a = Integer.parseInt(st.nextToken());
            b = Integer.parseInt(st.nextToken());
            c = Integer.parseInt(st.nextToken());

            edges[i] = new Edge(a,b,c);
        }

        Arrays.sort(edges,1,M+1);
```

```
        int connectCount = 0;
        int maxCost = 0;
        for(int i=1 ; i<=M ; i++){
            if(find(edges[i].from) != find(edges[i].to)){
                union(edges[i].from, edges[i].to);
                connectCount++;
                answer += edges[i].cost;
                // 두 마을로 분리했을 때의 최소 비용 구하는 방법
                // 일단 최소신장트리로 만들고, 그 중 가장 큰 길을 없앤다.
                maxCost = Math.max(maxCost, edges[i].cost);
            }

            if(connectCount == N-1)
                break;
        }

        bw.write((answer-maxCost) + "\n");
        bw.flush();
        bw.close();
        br.close();
    }

    static void union(int a, int b) {
        int aGroup = find(a);
        int bGroup = find(b);

        group[aGroup] = bGroup;
    }

    static int find(int a) {
        if(a == group[a])
            return a;
        return group[a] = find(group[a]);
    }
}
```