

# Kruskal算法求最小生成树

给定一个  $n$  个点  $m$  条边的无向图，图中可能存在重边和自环，边权可能为负数。

求最小生成树的树边权重之和，如果最小生成树不存在则输出 `impossible`。

给定一张边带权的无向图  $G = (V, E)$ ，其中  $V$  表示图中点的集合， $E$  表示图中边的集合， $n = |V|$ ， $m = |E|$ 。

由  $V$  中的全部  $n$  个顶点和  $E$  中  $n - 1$  条边构成的无向连通子图被称为  $G$  的一棵生成树，其中边的权值之和最小的生成树被称为无向图  $G$  的最小生成树。

## 输入格式

第一行包含两个整数  $n$  和  $m$ 。

接下来  $m$  行，每行包含三个整数  $u, v, w$ ，表示点  $u$  和点  $v$  之间存在一条权值为  $w$  的边。

## 输出格式

共一行，若存在最小生成树，则输出一个整数，表示最小生成树的树边权重之和，如果最小生成树不存在则输出 `impossible`。

## 数据范围

$1 \leq n \leq 10^5$ ,

$1 \leq m \leq 2 * 10^5$ ,

图中涉及边的边权的绝对值均不超过 1000。

## 输入样例：

```
4 5
1 2 1
1 3 2
1 4 3
2 3 2
3 4 4
```

## 输出样例：

```
6
```

y总代码：

```

1  #include <iostream>
2  #include <algorithm>
3
4  using namespace std;
5
6  const int N = 200010;
7
8  int n, m;
9  int p[N];
10
11 struct Edge
12 {
13     int a, b, w;
14
15     bool operator< (const Edge &w) const
16     {
17         return w < W.w;
18     }
19 } edges[N];
20
21 int find(int x)
22 {
23     if (p[x] != x) p[x] = find(p[x]);
24     return p[x];
25 }
26
27 int main()
28 {
29     scanf("%d%d", &n, &m);
30
31     for (int i = 0; i < m; i++)
32     {
33         int a, b, w;
34         scanf("%d%d%d", &a, &b, &w);
35         edges[i] = {a, b, w};
36     }

```

```

    sort(edges, edges + m);

    for (int i = 1; i <= n; i++) p[i] = i;

    int res = 0, cnt = 0;
    for (int i = 0; i < m; i++)
    {
        int a = edges[i].a, b = edges[i].b, w = edges[i].w;

        a = find(a), b = find(b);
        if (a != b)
        {
            p[a] = b;
            res += w;
            cnt++;
        }
    }

    if (cnt < n - 1) puts("impossible");
    else printf("%d\n", res);

    return 0;
}

```

```

#include <cstring>
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 100010, M = 200010, INF = 0x3f3f3f3f;

int n, m;
int p[N];

struct Edge
{
    int a, b, w;

    bool operator< (const Edge &w) const
    {
        return w < W.w;
    }
} edges[M];

```

```

int find(int x)
{
    if (p[x] != x) p[x] = find(p[x]);
    return p[x];
}

int kruskal()
{
    sort(edges, edges + m);

    for (int i = 1; i <= n; i++) p[i] = i;    // 初始化并查集

    int res = 0, cnt = 0;
    for (int i = 0; i < m; i++)
    {
        int a = edges[i].a, b = edges[i].b, w = edges[i].w;

        a = find(a), b = find(b);
        if (a != b)
        {
            p[a] = b;
            res += w;
            cnt++;
        }
    }

    if (cnt < n - 1) return INF;
    return res;
}

int main()
{
    scanf("%d%d", &n, &m);

    for (int i = 0; i < m; i++)
    {
        int a, b, w;
        scanf("%d%d%d", &a, &b, &w);
        edges[i] = {a, b, w};
    }

    int t = kruskal();

    if (t == INF) puts("impossible");
    else printf("%d\n", t);

    return 0;
}

```

作者: yxc

链接: <https://www.acwing.com/activity/content/code/content/48773/>

来源: AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

