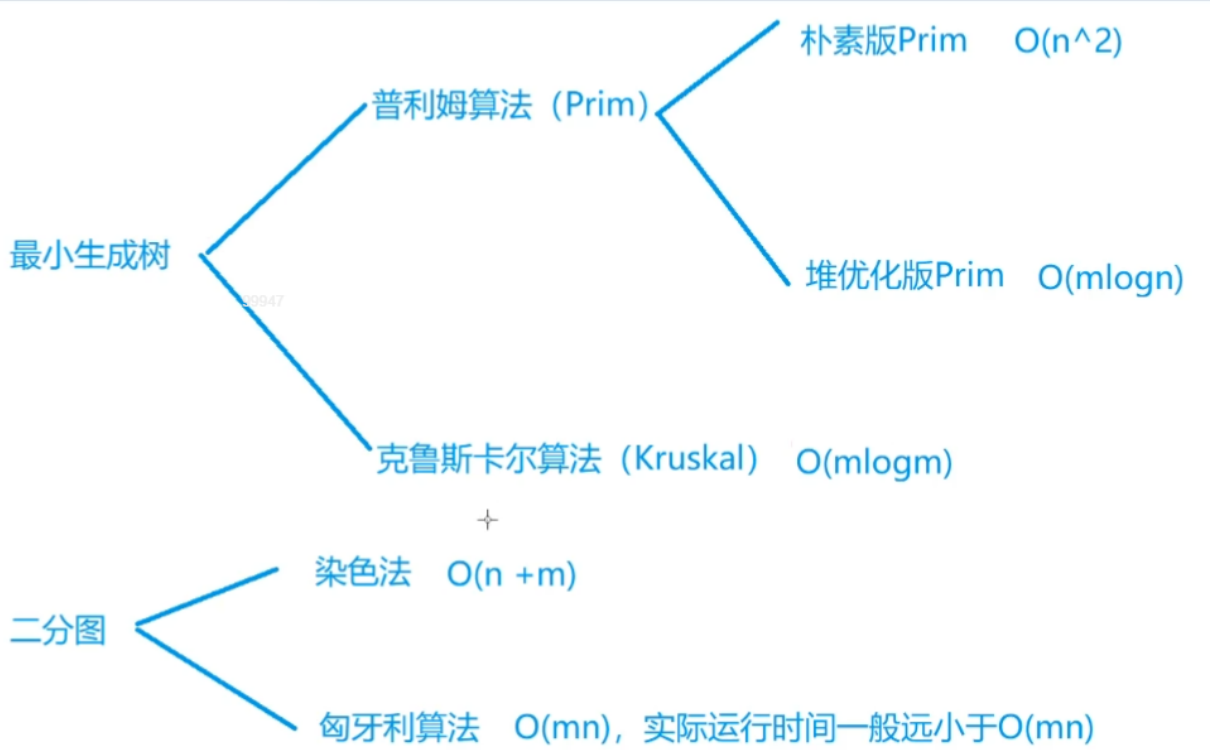

原理讲解视频：[最小生成树原理]

1.Prim算法（普里姆算法——稠密图）

枚举点

(21条消息) Prim算法（~详细整理，简单易懂，附最详细注释代码）_pursuit的博客-CSDN博客_prim算法

题目描述：

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。

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

$1 \le n \le 500$,
$1 \le m \le 10^5$,
图中涉及边的边权的绝对值均不超过 10000。

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
```

```
实现思路:
dis[n] -> 正无穷（把距离初始化为正无穷）
for(int i = 0; i < n; i++)
    找到集合外距离最近的点t; (和dijkstra算法一样的步骤)
    用t更新其他点到集合的距离;
    st[t] = true;
```

```cpp
#include<bits/stdc++.h>

using namespace std;

const int N = 510, INF = 0x3f3f3f3f;
int n,m;
int g[N][N];
int dis[N];
bool vis[N];

int prim()
{
    memset(dis, 0x3f3f3f3f, sizeof dis);

    int res = 0;
    for(int i = 0; i < n; i++){
        int t = -1;
        for(int j = 1; j <= n; j++){
            if(!vis[j] && (t == -1 || dis[j] < dis[t]))
                t = j;
```

```
        }

        //如果不是第一个点并且当前距离最近的点到集合的距离都是正无穷
        //说明当前图是不连通的，就不存在最小生成树
        if(i && dis[t] == INF) return INF;

        //如果不是第一条边，就把当前的距离加到答案中
        if(i) res += dis[t];
        vis[t] = true;

        for(int j = 1; j <= n; j++) dis[j] = min(dis[j], g[t][j]);
    }

    return res;
}

int main()
{
    cin >> n >> m;
    memset(g, 0x3f, sizeof g);

    while(m--){
        int a,b,c;
        cin >> a >> b >> c;
        g[a][b] = g[b][a] = min(g[a][b], c);
    }

    int ans = prim();

    if(ans == INF)  cout << "impossible";
    else cout << ans;

    return 0;
}
```

2.Kruskal算法（克鲁斯卡尔算法——稀疏图）

枚举边

```
实现思路:
1. 将所有边按权重从小到大排序（使用快排）
2. 枚举每条边 from,to,weight
    if from,to 不连通
    将这条边加入集合中（加边使用并查集）
```

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 1e5 + 10, M = 2*1e5 + 10, INF = 0x3f3f3f3f;
int n,m;
int pre[N];
struct Edge{
    int from,to,weight;

    //重载小于号，sort按权重排序
    bool operator < (Edge e){
        return weight < e.weight;
    }

}edge[M];

int find(int son)
{
    if(pre[son] == son) return son;
    return pre[son] = find(pre[son]);
}

int kruskal()
{
    int res = 0, cnt = 0;
    for(int i = 0; i < m; i++){
        int from = edge[i].from, to = edge[i].to, weight = edge[i].weight;

        from = find(from), to = find(to);

        //如果边form，to不连通，就将这条边加入到集合中
```

```
32          if(from != to){
33              pre[from] = to;
34              res += weight;
35              cnt++;
36          }
37      }
38
39      if(cnt < n - 1) return INF;
40      return res;
41  }
42
43  int main()
44  {
45      cin >> n >> m;
46      //初始化并查集
47      for(int i = 0; i < n; i++)  pre[i] = i;
48
49      for(int i = 0; i < m; i++){
50          int from,to,weight;
51          scanf("%d%d%d", &from, &to, &weight);
52          edge[i] = {from, to, weight};
53      }
54
55      //按权重从小到大排序边
56      sort(edge, edge + m);
57
58      int ans = kruskal();
59      if(ans == INF)  cout << "impossible";
60      else cout << ans;
61
62      return 0;
63  }
```