

图论—Kruskal 重构树

最大生成树将部分内容倒置即可

回顾：Kruskal

基本信息

1. 求解最小生成树
2. 时间复杂度： $O(m \log m)$
3. 更适合稀疏图

算法思想

1. 按照边权从小到大排序
2. 依次枚举每一条边，如果这一条边两侧不连通，则加入这条边

代码

▼ 点击查看代码

```
1  const int N = 200010;
2
3  int f[N];
4
5  struct Edge
6  {
7      int a, b, w;
8      bool operator<(const Edge &W) const { return w < W.w; }
9  } g[N];
10
11 int find(int x) { return x == f[x] ? x : find(f[x]); }
12
13 int main()
14 {
15     int n = rr, m = rr;
16
```

```

17     int a, b, w;
18     for (int i = 0; i < m; ++i)
19         a = rr, b = rr, w = rr, g[i] = {a, b, w};
20
21     sort(g, g + m);
22
23     for (int i = 1; i <= n; ++i)
24         f[i] = i;
25
26     int res = 0, cnt = 0;
27     for (int i = 0; i < m; ++i)
28     {
29         int a = find(g[i].a), b = find(g[i].b), w = g[i].w;
30         if (a != b)
31             f[a] = b, res += w, ++cnt;
32     }
33
34     cnt < n - 1 ? printf("impossible\n") : printf("%d\n", res);
35     return 0;
36 }

```

Kruskal 重构树

算法思想

在构建最小生成树的时候，设现在枚举到了一条要加入最小生成树的边 (u, v, w) ：

则在 Kruskal 重构树中，构建一个点权为 w 的虚点，编号为 t ，同时连边 (u, t) 、 (v, t) 。

主要性质

1. 重构树是一棵 [二叉树] ；
2. [子节点的点权] 小于 [父节点的点权] （即大根堆） ；
3. 最小生成树上 [两点之间的最大边权] 等于重构树上 [两点之间的最大边权] （即为重构树上两点 LCA 的点权） 。

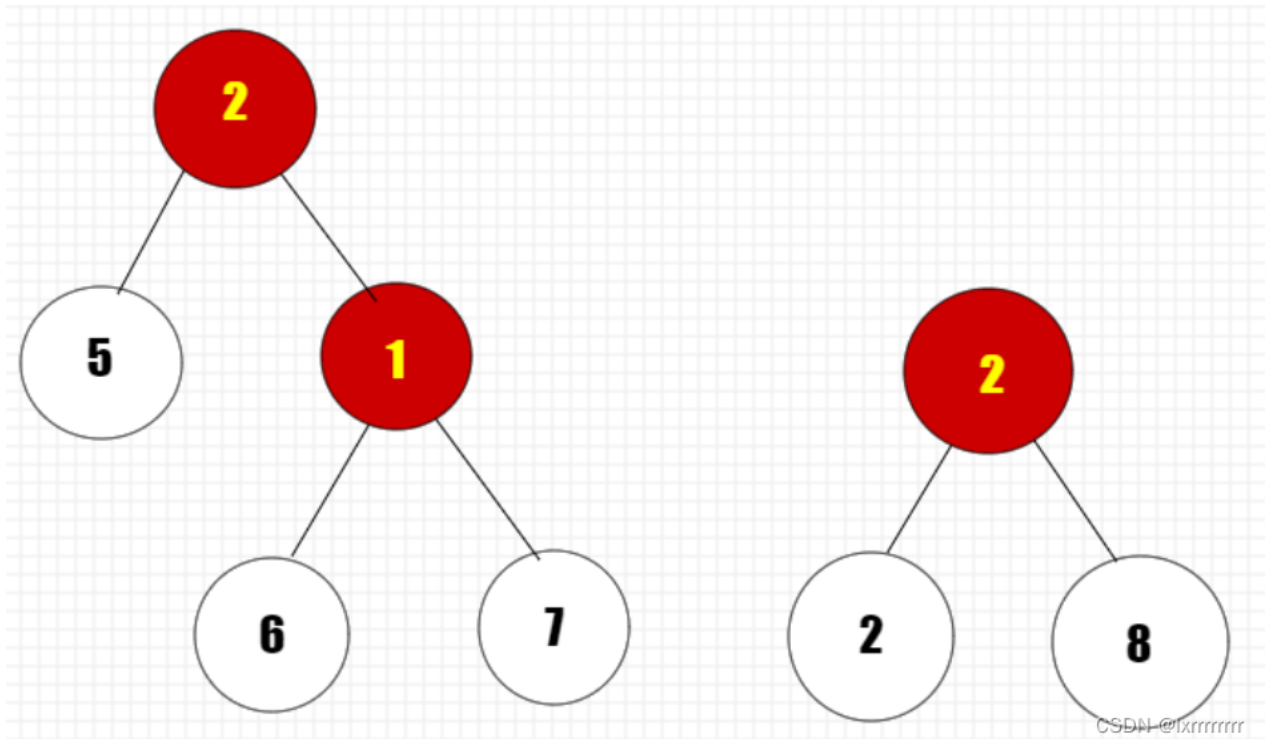
结论证明

最小生成树上两点间最大边权等于重构树上两点 LCA 的点权，证明：

1. 后加入的边权一定小于先加入的边权，所以重构树一定自上到下点权不减；

2. 两点在最小生成树上的路径的所有边一定都在重构树上两点之间；
3. 所以两点在最小生成树上之间的最长边权一定是重构树上两点 LCA 的点权。

如图：



其中红色的点表示虚点，中间的数字表示其点权；白色的点表示原有的点。

代码

```
1 // INPUT GRAPH
2 const int N = 2e5 + 10;
3 const int M = 2e5 + 10;
4
5 // NEW GRAPH
6 const int NN = N + M;
7 const int MM = M + M;
8
9 // 4LCA
10 const int K = 20;
11
12 // NODE, EDGE, QUERY
13 int n, m, q;
14
15 // INPUT GRAPH
16 struct e
```

```

17 {
18     int u, v, w;
19     bool operator<(const e &t) const { return w < t.w; }
20 } g[M];
21
22 // UNOIN
23 int f[NN];
24 int find(int x) { return x == f[x] ? x : f[x] = find(f[x]); }
25
26 // NEW GRAPH
27 int d[NN], cnt;
28 int h[NN], e[MM], ne[MM], idx;
29
30 // 4LCA
31 int depth[NN];
32 int up[NN][K];
33
34 // ADD TO NEW GRAPH
35 inline void _add(int u, int v)
36 {
37     e[idx] = v;
38     ne[idx] = h[u];
39     h[u] = idx++;
40 }
41
42 void add(int a, int b, int w)
43 {
44     d[++cnt] = w;
45     f[a] = f[b] = cnt;
46     _add(a, cnt), _add(cnt, a);
47     _add(b, cnt), _add(cnt, b);
48 }
49
50 // LCA INIT
51 void init(int u, int fa)
52 {
53     depth[u] = depth[fa] + 1;
54     for (int i = 1; i < K; ++i)
55         up[u][i] = up[up[u][i - 1]][i - 1];
56
57     for (int i = h[u]; i != -1; i = ne[i])
58     {
59         int v = e[i];
60         if (v == fa)
61

```

```

61         continue;
62         up[v][0] = u, init(v, u);
63     }
64 }
65
66 // KRUSKAL
67 int kruskal()
68 {
69     sort(g + 1, g + 1 + m);
70
71     for (int i = 1; i <= n * 2; ++i)
72         f[i] = i;
73
74     cnt = n;
75     memset(h, -1, sizeof h);
76
77     int res = 0;
78     for (int i = 1; i <= m; ++i)
79     {
80         int u = find(g[i].u), v = find(g[i].v), &w = g[i].w;
81         if (u == v)
82             continue;
83         res += w, add(u, v, w);
84     }
85
86     init(cnt, 0);
87     return res;
88 }
89
90 // LCA
91 int lca(int x, int y)
92 {
93     if (depth[x] < depth[y])
94         swap(x, y);
95
96     for (int i = K - 1; i >= 0; --i)
97     {
98         if (depth[up[x][i]] >= depth[y])
99             x = up[x][i];
100         if (x == y)
101             return x;
102     }
103
104     for (int i = K - 1; i >= 0; --i)
105

```

```

106         if (up[x][i] != up[y][i])
107             x = up[x][i], y = up[y][i];
108         return up[x][0];
109     }
110
111     int main()
112     {
113         n = rr, m = rr;
114
115         int a, b, w;
116         for (int i = 1; i <= m; ++i)
117             a = rr, b = rr, w = rr, g[i] = {a, b, w};
118
119         q = rr;
120
121         int res = kruskal();
122         while (q--)
123             printf("%d\n", d[lca(rr, rr)]);
124         return 0;
    }

```

Reference

- [1] <https://www.luogu.com.cn/blog/lizbaka/kruskal-chong-gou-shu>
- [2] https://blog.csdn.net/m0_61735576/article/details/124804973

本文来自博客园，作者：RainPPR，转载请注明原文链接：<https://www.cnblogs.com/RainPPR/p/kruskal-zhong-gou-shu.html>

合集：学习笔记

标签：算法 ， 学习笔记