异象石问题

赵涵铮

2023 年 2 月 19 日

1 异象石

1.1 题目描述

给定有 n 个节点的树，m 次操作，每次操作：

1. 选定一个点作为异象点
2. 删除一个异象点
3. 输出联通所有异象点的路径之和

1.2 数据范围

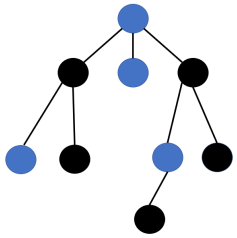
$1 \leq N \leq 10^5, 1 \leq M \leq 10^5, 1 \leq X \leq Y \leq N$, 数字不超过 C/C++ 的 int 范围。

1.3 解题思路

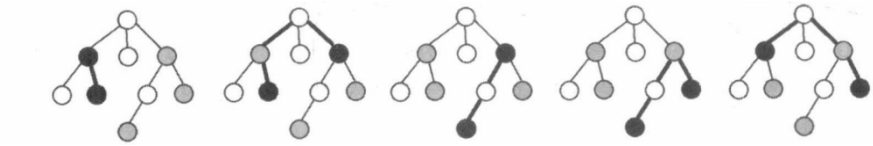
根据题意，本题是对一棵树进行维护，由于树上两点只有唯一路径，所以是不需要考虑最短路径等问题的，只需要考虑如何维护临近的两点。

我们可以先通过 dfs，得到每一个点的时间戳，然后按照时间戳顺序，将异象点的节点排成首尾相连的一圈，累加两点之间的路径长度，最终得到的结果恰好是答案的两倍（每条边恰好经过两次）

以此图为例，假设黑色的点为异象点：



按照时间戳顺序，以此选择两个点，加粗表示两者路径，通过五次计算，加粗的边恰好被覆盖了两次：



为了维护好这个异象点集合，我们可以使用数据结构 set，便于增删点，用 ans 来记录序列相邻点的路径长度之和（包括序列首尾）。

接下来，我们需要解决如何计算任意两点距离的问题。设 $path(x,y)$ 表示树上两点的路径长度，设 $d[x]$ 表示 x 到根节点的路径长度，可知：

$$path(x,y) = d[x] + d[y] - 2 * d[LCA(x,y)] \tag{1}$$

这样，就可以通过 LCA 算出 $path(x,y)$，同时，我们可以通过一次 dfs 预处理好 d 数组。

现在，让我们再次回顾一下全过程：若一个节点 x 出现了异象石，则根据时间戳，将节点 x 插入 set 中，它前后分别是 l，r，则

$$ans - path(l,r) + path(l,x) + path(x,r)$$

若一个节点的异象点被删除，就类似的更新 set。值得一提的是，set 的插入和删除复杂度都是 $O(\log N)$，时间复杂度为 $O((N+M)\log N)$

1.4 参考代码

Listing 1: 异象石代码

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define rep(i,a,b) for(int i = a ; i <= b ; i++ )
4 #define per(i,a,b) for(int i = a ; i >= b ; i-- )
```

```
5  typedef long long ll;
6  const int N = 1e5+7;
7
8  int n;
9  struct Node {
10         ll num,val;
11 };
12 vector<Node>tree[N];
13 int cnt = 1;
14 ll pos[N],d[N],dfn[N];
15 void dfs(int root) {
16         for(auto &son:tree[root]) {
17                 if(dfn[son.num])
18                         continue;
19                 dfn[son.num] = ++cnt;
20                 pos[ dfn[son.num] ] = son.num;
21                 d[son.num] = d[root] + son.val;
22
23                 dfs(son.num);
24         }
25         return ;
26 }
27
28 int dp[N][21],depth[N];
29 void init(int root) {
30         rep(i,1,n) {
31                 depth[i] = 0x3f3f3f3f;
32         }
33         queue<int>q;
34         q.push(root);
35         depth[root] = 1;
36         while(q.size()) {
37                 auto x = q.front();
38                 q.pop();
39                 for(auto &son:tree[x]) {
40                         if(depth[son.num] > depth[x]+1) {
41                                 depth[son.num] = depth[x]+1;
42                                 q.push(son.num);
43
```

```
44                                  dp[son.num][0] = x;
45                                  rep(i,1,20) {
46                                          dp[son.num][i] = dp[ dp[son.num][i
                                                -1] ][i-1];
47                                  }
48                          }
49                  }
50          }
51  }
52
53  int lca(int a,int b) {
54          if(depth[a] > depth[b])
55                  swap(a,b);
56          // b is below a
57          per(i,20,0) {
58                  if(depth[dp[b][i]] >= depth[a]) {
59                          b = dp[b][i];
60                  }
61          }
62          if(a==b)
63                  return a;
64          per(i,20,0) {
65                  if(dp[a][i] != dp[b][i]) {
66                          a = dp[a][i];
67                          b = dp[b][i];
68                  }
69          }
70          return dp[a][0];
71  }
72
73  ll path(int x,int y) {
74          return d[x]+d[y]-d[lca(x,y)]*2;
75  }
76
77  ll ans = 0;
78  set<int>s;
79  void work(ll flag,int x) {
80          auto it = s.find(x);
81          auto l = it;
```

```
82          if(l==s.begin()) {
83                  l = s.end();
84          }
85          l--;
86
87          auto r = it;
88          r++;
89          if(r==s.end())
90                  r = s.begin();
91
92          // 1 8 9
93          int ll = pos[*l], rr = pos[*r];
94          ans -= path(ll,rr)*flag;
95          ans += (path(ll,pos[x])+path(pos[x],rr))*flag;
96  }
97
98  int main() {
99          scanf("%d",&n);
100         rep(i,1,n-1) {
101                 int x,y,z;
102                 scanf("%d%d%d",&x,&y,&z);
103                 tree[x].push_back({y,z});
104                 tree[y].push_back({x,z});
105         }
106         init(1);
107         pos[1] = 1,dfn[1] = 1;
108         dfs(1);
109         int m;
110         scanf("%d",&m);
111         rep(i,1,m) {
112                 char ch;
113                 cin >> ch;
114                 if(ch=='?') {
115                         printf("%lld\n",ans/2);
116                 } else {
117                         int x;
118                         scanf("%d",x);
119                         if(ch=='+') {
120                                 s.insert(dfn[x]);
```

```
121                                 work(1,dfn[x]);
122                         } else {
123                                 work(-1,dfn[x]);
124                                 s.erase(dfn[x]);
125                         }
126                 }
127         }
128 }
```