

可持久化并查集

Persistent Disjoint Set Union

Idea: 所谓并查集，在实现上无非就是两个数组 `fa[]` 和 `sz[]`（按秩合并），只需要将这两个数组可持久化即可。把 y 合并到 x 上时，我们需要在某个历史版本上更改 `fa[findfa(y)]` 为 `findfa(x)`，并将 `sz[findfa(x)]` 加上 `sz[findfa(y)]`。而所谓可持久化数组，仍旧是可持久化线段树来实现。

Complexity: $O(n \lg^2 n)$

Code:

```
1  #include<algorithm>
2  #include<iostream>
3  #include<cstdio>
4
5  using namespace std;
6
7  const int N = 200005;
8  int n, m;
9
10 struct segTree{
11     int lson, rson;
12     int fa, sz;
13 }tr[N * 30];
14 int cnt, root[N];
15 void build(int id, int l, int r){
16     if(l == r){
17         tr[id].fa = l, tr[id].sz = 1;
18         return;
19     }
20     tr[id].lson = ++cnt, tr[id].rson = ++cnt;
21     int mid = (l + r) >> 1;
22     build(tr[id].lson, l, mid);
23     build(tr[id].rson, mid+1, r);
24 }
25 int queryID(int id, int l, int r, int pos){ // query the ID of pos'th leaf in tr[id]
26     if(l == r) return id;
27     int mid = (l + r) >> 1;
28     if(pos <= mid) return queryID(tr[id].lson, l, mid, pos);
29     else return queryID(tr[id].rson, mid+1, r, pos);
30 }
31 void modify(int cur, int pre, int l, int r, int pos, int fa){
32     tr[cur] = tr[pre];
33     if(l == r){ tr[cur].fa = fa; return; }
34     int mid = (l + r) >> 1;
35     if(pos <= mid){
36         tr[cur].lson = ++cnt;
37         modify(tr[cur].lson, tr[pre].lson, l, mid, pos, fa);
38     }
39     else{
40         tr[cur].rson = ++cnt;
41         modify(tr[cur].rson, tr[pre].rson, mid+1, r, pos, fa);
42     }
43 }
44 void add(int id, int l, int r, int pos, int val){
45     if(l == r){ tr[id].sz += val; return; }
46     int mid = (l + r) >> 1;
47     if(pos <= mid) add(tr[id].lson, l, mid, pos, val);
48     else add(tr[id].rson, mid+1, r, pos, val);
49 }
50
51 int findfa(int cur, int x){
52     int xid = queryID(cur, 1, n, x); // x's id in tr[cur]
53     return tr[xid].fa == x ? x : findfa(cur, tr[xid].fa);
54 }
55 void unionn(int cur, int pre, int x, int y){
56     tr[cur] = tr[pre]; // ATT!
57     x = findfa(pre, x), y = findfa(pre, y);
58     if(x == y) return;
59     int xid = queryID(pre, 1, n, x), yid = queryID(pre, 1, n, y);
60     if(tr[xid].sz < tr[yid].sz) swap(x, y), swap(xid, yid);
```

```

61     modify(cur, pre, 1, n, y, x);
62     add(cur, 1, n, x, tr[yid].sz);
63 }
64
65 int main(){
66     scanf("%d%d", &n, &m);
67     build(root[0] = 0, 1, n);
68     for(int i = 1; i <= m; i++){
69         int opt, a, b;
70         scanf("%d", &opt);
71         if(opt == 1){
72             scanf("%d%d", &a, &b);
73             root[i] = ++cnt;
74             unionn(root[i], root[i-1], a, b);
75         }
76         else if(opt == 2){
77             scanf("%d", &a);
78             root[i] = root[a];
79         }
80         else{
81             scanf("%d%d", &a, &b);
82             root[i] = root[i-1];
83             puts(findfa(root[i], a) == findfa(root[i], b) ? "1" : "0");
84         }
85     }
86     return 0;
87 }

```