A Simple Tree Problem

Time Limit: 3 Seconds      Memory Limit: 65536 KB

Given a rooted tree, each node has a boolean (0 or 1) labeled on it. Initially, all the labels are 0.

We define this kind of operation: given a subtree, negate all its labels.

And we want to query the numbers of 1's of a subtree.

Input

Multiple test cases.

First line, two integer **N** and **M**, denoting the numbers of nodes and numbers of operations and queries.(1<=N<=100000, 1<=M<=10000)

Then a line with **N**-1 integers, denoting the parent of node 2..**N**. Root is node 1.

Then **M** lines, each line are in the format "o *node*" or "q *node*", denoting we want to operate or query on the subtree with root of a certain node.

Output

For each query, output an integer in a line.

Output a blank line after each test case.

Sample Input

3 2

1 1

o 2

q 1

Sample Output

1

题意：给你一颗树，并且有两种操作，一种是将以节点i为root的子树每个节点上的值取反，还有一种操作时

       查询以节点i为root的子树所有节点上的值得和。节点上的值不是1就是0，初始皆为0。

题解：把子树化为区间的就是裸的线段树了。我们可以先序遍历树，把以x为根节点的子树的用L[x],R[x]区间表示

         然后就是线段树区间更新了。

1. #include<cstring>
2. #include<cstdio>
3. #include<iostream>
4. #include<cmath>
5. #include<vector>
6. #include<algorithm>
7. #define lc idx<<1
8. #define rc idx<<1|1
9. #define lson l,mid,lc
10. #define rson mid+1,r,rc
11. #define N 100010
13. using namespace std;
15. int n,m;
16. vector<int>G[N];
17. int L[N],R[N],id;
19. struct Tree {
20. int st;
21. int one;
22. } tree[N<<2];
24. void init() {
25. for(int i=0; i<=n; i++)
26. G[i].clear();
27. id=1;
28. }
30. void dfs(int fa) {
31. L[fa]=id++;
32. for(int i=0; i<G[fa].size(); i++) {
33. dfs(G[fa][i]);
34. }
35. R[fa]=id-1;
36. }
38. void push\_up(int idx) {
39. tree[idx].one=tree[lc].one+tree[rc].one;
40. }
42. void push\_down(int idx,int m) {
43. if(tree[idx].st) {
44. tree[lc].st^=1,tree[rc].st^=1;
45. tree[idx].st=0;
46. tree[lc].one=m-m/2-tree[lc].one;
47. tree[rc].one=m/2-tree[rc].one;
48. }
49. }
51. void build(int l,int r,int idx) {
52. tree[idx].one=0;
53. tree[idx].st=0;
54. if(l==r)return;
55. int mid=(l+r)>>1;
56. build(lson);
57. build(rson);
58. }
60. void update(int l,int r,int idx,int x,int y) {
61. if(x<=l&&r<=y) {
62. tree[idx].st^=1;
63. tree[idx].one=r-l+1-tree[idx].one;
64. return;
65. }
66. push\_down(idx,r-l+1);
67. int mid=(l+r)>>1;
68. if(x<=mid)update(lson,x,y);
69. if(y>mid) update(rson,x,y);
70. push\_up(idx);
71. }
73. int query(int l,int r,int idx,int x,int y) {
74. if(x<=l&&r<=y)
75. return tree[idx].one;
76. push\_down(idx,r-l+1);
77. int mid=(l+r)>>1;
78. int ans=0;
79. if(x<=mid) ans+=query(lson,x,y);
80. if(y>mid)  ans+=query(rson,x,y);
81. return ans;
82. }
84. int main() {
85. //freopen("test.in","r",stdin);
86. while(~scanf("%d%d",&n,&m)) {
87. init();
88. int fa;
89. for(int i=2; i<=n; i++) {
90. scanf("%d",&fa);
91. G[fa].push\_back(i);
92. }
93. dfs(1);
94. build(1,n,1);
95. char c[3];
96. int rt;
97. while(m--) {
98. scanf("%s%d",c,&rt);
99. if(c[0]=='o') {
100. update(1,n,1,L[rt],R[rt]);
101. continue;
102. }
103. printf("%d\n",query(1,n,1,L[rt],R[rt]));
104. }
105. printf("\n");
106. }
107. return 0;
108. }