Kth Ancestor

by amititkgp

Problem Submissions Leaderboard Discussions Editorial

A tree of P nodes is an un-directed connected graph having $P-1$ edges. Let us denote R as the root node. If A is a node such that it is at a distance of L from R, and B is a node such that it is at at distance of $L+1$ from R and A is connected to B, then we call A as the parent of B.

Similarly, if A is at a distance of L from R and B is at a distance of $L+K$ from R and there is a path of length K from A to B, then we call A as the K^{th} parent of B.

Susan likes to play with graphs and Tree data structure is one of her favorites. She has designed a problem and wants to know if anyone can solve it. Sometimes she adds or removes a leaf node. Your task is to figure out the K^{th} parent of a node at any instant.

Input Format

The first line contain an integer T denoting the number of test cases. T test cases follow. First line of each test case contains an integer P, the number of nodes in the tree. P lines follows each containing two integers X and Y separated by a single space denoting Y as the parent of X. If Y is 0, then X is the root node of the tree. (0 is for namesake and is not in the tree).
The next line contains an integer Q, the number of queries.
Q lines follow each containing a query.

- $0\ Y\ X$: X is added as a new leaf node whose parent is Y . X is not in the tree while Y is in.
- $1\ X$: This tells that leaf node X is removed from the tree. X is a leaf in the tree.
- $2\ X\ K$: In this query output the K^{th} parent of X . X is a node in the tree.

Note

- Each node index is any number between 1 and 10^5 i.e., a tree with a single node can have its root indexed as 10^5

Constraints

$1 \leq T \leq 3$
$1 \leq P \leq 10^5$
$1 \leq Q \leq 10^5$
$1 \leq X \leq 10^5$
$0 \leq Y \leq 10^5$
$1 \leq K \leq 10^5$

Output Format

For each query of type 2, output the K^{th} parent of X. If K^{th} parent doesn't exist, output 0 and if the node doesn't exist, output 0.

Sample Input

```
2
7
2 0
5 2
3 5
7 5
9 8
```

```
8 2
6 8
10
0 5 15
2 15 2
1 3
0 15 20
0 20 13
2 13 4
2 13 3
2 6 10
2 11 1
2 9 1
1
10000 0
3
0 10000 4
1 4
2 4 1
```

Sample Output

```
2
2
5
0
0
8
0
```

Explanation

There are 2 test cases. The first test case has 7 nodes with 2 as its root. There are 10 queries

- 0 5 15 -> 15 is added as a leaf node to 5.
- 2 15 2 -> 2nd parent of 15 is 15->5->2 is 2.
- 1 3 -> leaf node 3 is removed from the tree.
- 0 15 20 -> 20 is added as a leaf node to 15.
- 0 20 13 -> 13 is added as a leaf node to 20.
- 2 13 4 -> 4th parent of 13 is 2.
- 2 13 3 -> 3rd parent of 13 is 5.
- 2 6 10 -> there is no 10th parent of 6 and hence 0.
- 2 11 1 -> 11 is not a node in the tree, hence 0.
- 2 9 1 -> 9's parent is 8.

the second testcase has a tree with only 1 node (10000).

- 0 10000 4 -> 4 is added as a leaf node to 10000.
- 1 4 -> 4 is removed.
- 2 4 1 -> as 4 is already removed, answer is 0.

Java 8

```
import java.io.*;
import java.util.*;

class CustomHeap{

    private Map<Integer, CustomHeapClass> map;

    public CustomHeap(){
        map  = new HashMap<Integer, CustomHeapClass>();
    }

    public void initiate(int parent, int child){

        if(parent == 0){
            CustomHeapClass heapClass = new CustomHeapClass(0,0);
            map.put(child,heapClass);
        }
        else{

            CustomHeapClass heapParentClass = map.get(parent);
            CustomHeapClass heapClass = new CustomHeapClass(parent,heapParentClass.getHeight() + 1);
            map.put(child,heapClass);
        }
    }

    public void addNode(int parent, int node){

        CustomHeapClass heapParentClass = map.get(parent);
        CustomHeapClass heapChildClass = new CustomHeapClass(parent, heapParentClass.getHeight() + 1);
        map.put(node, heapChildClass);
    }


    public void removeNode(int node){
        CustomHeapClass heapNodeClass = map.get(node);
        heapNodeClass.setParent(Integer.MAX_VALUE);
        map.put(node,heapNodeClass);
    }

    public int getKthParent(int node, int K){

        CustomHeapClass heapNodeClass = map.get(node);
        int i = 0;

        int initial = heapNodeClass != null ? heapNodeClass.height : Integer.MAX_VALUE;

        while(true)
        {

            //System.out.println(node + " node value  "  + heapNodeClass.height);

            if(heapNodeClass == null || heapNodeClass.height == Integer.MAX_VALUE || K > initial)
            {
                return 0;
            }
            else if(K == 0 || K == i)
            {
                return node;
            }
            else
            {
                node = heapNodeClass.parent;
                heapNodeClass = map.get(node);
            }

            i++;

        }
    }
}
```

```
71
72
73  class CustomHeapClass{
74
75      int parent,height;
76
77      public CustomHeapClass(int parent, int height){
78          this.parent = parent;
79          this.height = height;
80      }
81
82      public void setParent(int parent){
83          this.parent = parent;
84      }
85
86      public void setHeight(int height){
87          this.height = height;
88      }
89
90      public int getParent(){
91          return parent;
92      }
93
94      public int getHeight(){
95          return height;
96      }
97
98  }
99
100
101 public class Solution {
102
103     public static void main(String[] args) throws IOException{
104
105         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
106         int tst = Integer.parseInt(br.readLine());
107
108         for(int i = 0 ; i < tst ; i++){
109
110             int V = Integer.parseInt(br.readLine());
111
112             CustomHeap customHeap = new CustomHeap();
113
114             for(int edges = 0 ; edges < V  ; edges++){
115
116                 String[] num = br.readLine().split("\\s");
117
118                 customHeap.initiate(Integer.parseInt(num[1]), Integer.parseInt(num[0]));
119             }
120
121             int q = Integer.parseInt(br.readLine());
122
123             for(int que = 0 ; que < q ; que++){
124
125                 String[] num = br.readLine().split("\\s");
126
127                 if(Integer.parseInt(num[0]) == 0){
128                     customHeap.addNode(Integer.parseInt(num[1]), Integer.parseInt(num[2]));
129                 }
130                 else if(Integer.parseInt(num[0]) == 1){
131                     customHeap.removeNode(Integer.parseInt(num[1]));
132                 }
133                 else{
134                     System.out.println(customHeap.getKthParent(Integer.parseInt(num[1]),Integer.parseInt(num[2])));
135                 }
136             }
137         }
138     }
139 }
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code