Trees : Level 2
Lesson 5

1. Largest Cycle in a tree
2. Nodes in a subtree
3. Directory Deletion
4. Population Outburst
5. Mancunian and Coloured tree
6. Gandhi Tree March

Largest Cycle in a Tree

Problem

You are given a tree of N nodes and N–1 edges. Now you need to select two nodes a and b in the tree such that the cycle that will be formed after adding an edge between the two nodes a and b, its length should be maximum. If there is more than one possible answer, you can output any of them.

Input

The first line contains an integer N as input. Next N–1 lines contain a pair of integers (a,b) that denote there is an edge between the two nodes a and b in the tree.

Output

In the output, you need to print two integers separated by space which denote the nodes between which you can add the edge so as to maximize the length of the cycle in the tree.

Constraints

$1 \leq N \leq 105$

Sample Input

```
7
1 2
1 3
2 4
2 5
3 6
3 7
```

Sample Output

```
4 6
```

```cpp
#include<bits/stdc++.h>
using namespace std;
void DFS(int N,map<int,vector<int>> &mp, int &a, int &dis, int d, int p)
{
    if(d>dis)
    {
        dis=d;
        a=N;
    }
    int k=mp[N].size();
    for(int i=0; i<k; i++)
    {
        if(mp[N][i]!=p)
            DFS(mp[N][i],mp,a,dis,d+1,N);
    }
}
int main()
{
    int N;
    cin>>N;
    map<int,vector<int>> mp;
    for(int i=0; i<N-1; i++)
    {
        int u,v;
        cin>>u>>v;
        mp[u].push_back(v);
        mp[v].push_back(u);
    }
    int a=1,dis=0;
    DFS(1,mp,a,dis,0,-1);
    int b=a;
    dis=0;
    DFS(a,mp,b,dis,0,-1);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

Nodes in a subtree

Problem

You are given a rooted tree that contains N nodes. Each node contains a lowercase alphabet. You are required to answer Q queries of type u,c, where u is an integer and c is a lowercase alphabet. The count of nodes in the subtree of the node u containing c is considered as the answer of all the queries.

Input format

- First line: Two space-separated integers N and Q respectively.
- Second line: A string s of length N (where the ith character of s represents the character stored in node i).
- Next N–1 line: Two space-separated integers u and v denoting an edge between node u and node v.
- Next Q lines: An integer u and a space-separated character c.

Output format

For each query, print the output in a new line.

Constraints

$1 \leq N, Q \leq 10^5$

$1 \leq u,v \leq N$

Note: It is guaranteed that the input generates a valid tree.

Sample Input

```
3 1
aba
1 2
1 3
1 a
```

Sample Output

2

```
#include<bits/stdc++.h>
using namespace std;
vector<vector<int>> freq;
void freq_calc(int u, int p, map<int,vector<int>> &mp,
string &S)
{
      freq[u][int(S[u-1]-'a')]++;
      int l=mp[u].size();
      for(int i=0; i<l; i++)
      {
            if(mp[u][i]!=p)
            {
                  freq_calc(mp[u][i],u,mp,S);
                  for(int j=0; j<26; j++)
                        freq[u][j]+=freq[mp[u][i]][j];
            }
      }
}
int main()
{
    int N, Q;
    string S;
    cin >> N >> Q;
    cin >> S;
    map<int,vector<int>> adj;
    for(int i = 0; i < N - 1; i ++)
    {
          int u, v;
          cin >> u >> v;
         // Process the Inputs of node details
          adj[u].push_back(v);
          adj[v].push_back(u);
    }
    freq.assign(N+1,vector<int>(26,0));
```

```
    freq_calc(1,-1,adj,S);
    while(Q --)
    {
        int u;
        char c;
        cin >> u >> c;
        // Process queries
        cout<<freq[u][c-'a']<<"\n";
    }
    return 0;
}
```

Directory Deletion

Problem

You are given a directory tree of N directories/folders. Each directory is represented by a particular id which ranges from 1 to N. The id of the root directory is 1, then it has some child directories, those directories may contain some other ones and it goes on. Now you are given a list of directory id's to delete, you need to find the minimum number of directories that need to be deleted so that all the directories in the given list get deleted.

Note that if you delete a particular directory, all its child directories will also get deleted.

Input

The first line of input contains an integer N that denotes how many folders are there.

The next line contains N space separated integers that where the ith integer denotes the id of the parent of the directory with id i . Note that the first integer will be −1 as 1 is the id of root folder and it has no parent. Rest of the integers will not be −1 .

The next line contains an integer M that denotes how many directories you need to delete.

The next line contains M space separated integers that denote the ids of the directories you need to delete.

Output

Print the minimum number of directories that need to be deleted.

Constraints

$1 \leq N \leq 105$

1≤id of parent≤N

1≤M≤N

1≤id to be deleted≤N

Sample Input

7
-1 1 1 3 3 4 4
3
2 3 4

Sample Output

2

```
#include<bits/stdc++.h>
using namespace std;
void dfs(int N, map<int,vector<int>> &mp, bool del[], int
&count)
{
      if(del[N]==true)
      {
            count++;
            return;
      }
      int k=mp[N].size();
      for(int i=0; i<k; i++)
      {
            dfs(mp[N][i],mp,del,count);
      }
}
int main()
{
   int N;
   cin>>N;
   map<int,vector<int>> mp;
   int u;
   cin>>u;
   for(int i=2; i<=N; i++)
```

```
24.     {
25.         cin>>u;
26.         mp[u].push_back(i);
27.     }
28.     int count=0;
29.     int m;
30.     cin>>m;
31.     bool del[N+1];
32.     for(int i=0; i<=N; i++)
33.         del[i]=false;
34.     for(int i=0; i<m; i++)
35.     {
36.         cin>>u;
37.         del[u]=true;
38.     }
39.     dfs(1,mp,del,count);
40.     cout<<count;
41. }
```

Population Outburst

Problem

A new species is trying to rule the planet. This species is creating their own population outburst to dominate other species. It all started with 1 single member of the species. The population increases in treelike fashion abiding by few rules as listed below.

- Single member is able to reproduce by itself.
- A new member is added to the population every minute.
- Every member is associated with integral name.
- Multiple members can share a common name.
- Every member has it's own reproduction capacity, that is maximum number of children it can reproduce.
- A member can start to reproduce only if all members older than it have exhausted their reproduction capacity.
- Level 0 in family tree of this species comprise of single member at the start of multiplication.
- Integral name of single member at the start is 0.
- The population grows level wise, where number of members at level *i* is dependent on reproduction capacity of members at prior level.

Given the integral name of new member and it's reproduction capacity that is added to the population, you have to find it's parent, level at which it is added and it's ascending age wise rank among siblings.

Input:

First line of the input contains 2 integers, N,RC0, representing number of minutes we will be examining the population increase and reproduction capacity of member at epoch. Next *N* line contains 2 integers each, IDi,RCi, representing integral name and reproduction capacity of new member born at time *i*.

Output:

N lines, each line containing 3 integers, P,L,C, representing integral name of the parent, level at which it is added and it's ascending age wise rank among siblings.

Note :

It will always be possible to reproduce a new child or in other words, through out the given time, there exists atleast one member which can still accomodate new child.

Constraints:

$1 \leq N \leq 106$

$-109 \leq IDi \leq 109$

$0 \leq RCi \leq 109$

Sample Input

```
9 2
1 2
2 1
5 2
4 1
10 0
15 0
21 0
23 1
42 100
```

Sample Output

```
0 1 1
0 1 2
1 2 1
1 2 2
2 2 1
5 3 1
5 3 2
4 3 1
```

23 4 1

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
      int N,RC;
      cin>>N>>RC;
      queue<pair<int,int>> q;
      q.push({0,RC});
      int t=0,level=0;
      while(t<N)
      {
            int l=q.size();
            level++;
            while(l-- && t<N)
            {
                  int rank=0;
                  int name=q.front().first;
                  int capacity=q.front().second;
                  q.pop();
                  while(capacity-- && t<N)
                  {
                        rank++;
                        int child;
                        cin>>child>>RC;
                        t++;
                        q.push({child,RC});
                        cout<<name<<" "<<level<<"
  "<<rank<<endl;
                  }
            }
      }
      return 0;
  }
```

Mancunian and Colored Tree

Problem

After a hectic week at work, Mancunian and Liverbird decide to go on a fun weekend camping trip. As they were passing through a forest, they stumbled upon a unique tree of N nodes. Vertices are numbered from 1 to N.Each node of the tree is assigned a color (out of C possible colors). Being bored, they decide to work together (for a change) and test their reasoning skills. The tree is rooted at vertex 1. For each node, they want to find its closest ancestor having the same color.

Input format

The first line contains two integers N and C denoting the number of vertices in the tree and the number of possible colors.

The second line contains $N-1$ integers. The i th integer denotes the parent of the $i+1$ th vertex.

The third line contains N integers, denoting the colors of the vertices. Each color lies between 1 and C inclusive.

Output format

Print N space-separated integers. The ith integer is the vertex number of lowest ancestor of the ith node which has the same color. If there is no such ancestor, print 1 for that node.

Constraints

- $1 \leq N \leq 100,000$
- $1 \leq C \leq 100,000$

Sample Input

5 4

1 1 3 3

1 4 2 1 2

Sample Output

-1 -1 -1 1 3

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. int main()
4. {
5.       int N,C;
6.       cin>>N>>C;
7.       int arr[N-1];
8.       for(int i=0; i<N-1; i++)
9.            cin>>arr[i];
10.      int col[N];
11.      for(int i=0; i<N; i++)
12.           cin>>col[i];
13.      cout<<-1<<" ";
14.      for(int i=2; i<=N; i++)
15.      {
16.           int j=arr[i-2];
17.           while(j!=1 && col[j-1]!=col[i-1])
18.                j=arr[j-2];
19.           if(col[j-1]==col[i-1])
20.                cout<<j<<" ";
21.           else
22.                cout<<-1<<" ";
23.      }
24.      return 0;
25.  }
```

Gandhi Tree March

Problem

Gandhijee is interested in building human tree. He defined a human node as follows :

- Person_Id = English alphabet {a...z} .
- Person_Chain_Definition = Person_Id (Person_Chain_Definition Person_Chain_Definition)

For example :

a(b(..) c(d(. e(..)) f(. .))) refers to a human tree having a as the root human, whose chain links are are b and c, (spaces here are for clarity, input file has no spaces).

Note : After every Person_Id there is necessarily an opening parenthesis (that starts the definition of the sub-tree under the person. For empty(NULL) human nodes, there is a '.' character.

Now, Gandhijee figure out the tree as follows :

Step 1 : He writes root human root id in column 0.

Step 2 : Then he writes left link's person id in 1 column to the left and right link's person id to 1 column to the right.

He continues writing the subtree, in a way that, for an id written in column C, it's left link's id is written in column C-1 and the right link's id is written in column C+1.

Now, Gandhijee wants to know the id's of people standing in a particular column C in lexicographically sorted order.

You are provided with column number C and the Tree definition string S. Gandhijee is very busy and hence he wants you to help him.

Input :

First line contains a number T, the number of test cases.

Each test case consists of a single line containing C, the column number for which you need to print the result, followed by a space followed by definition of the tree in the format given above, without any spaces.

Output :

For each test case, on a new line print the required id's in lexicographically sorted order, without spaces. If there are no id's in column C, print "Common Gandhijee!" (quotes for clarity).

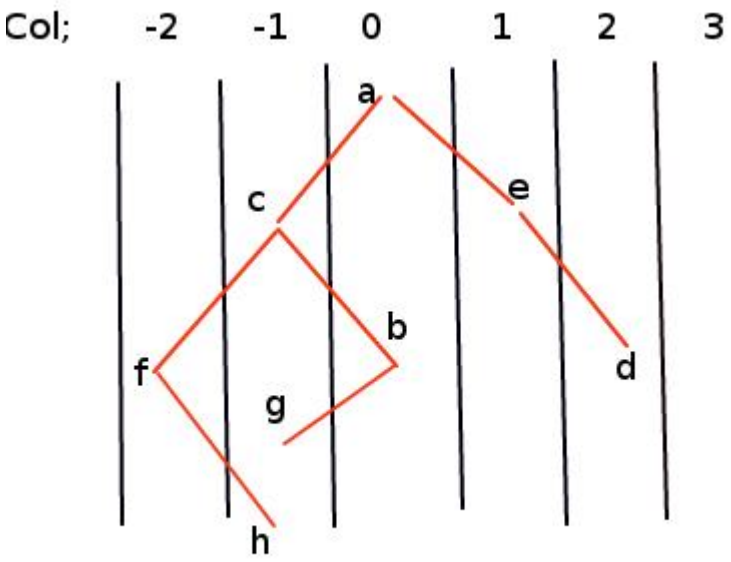
Constraints :

$1 \leq T \leq 100$

$1 \leq Len(S) \leq 10000$

$-10000 \leq C \leq 10000$

Image for given testcase.



Sample Input

2

-1 a(c(f(.h(..))b(g(..).))e(.d(..)))

3 b(c(..)a(..))

Sample Output

cgh

Common Gandhijee!

```
#include<bits/stdc++.h>
using namespace std;
void dfs(int c, string S, vector<char> &ans, int &i, int
col)
{
      if(S[i]=='.')
      {
            i++;
            return;
      }
      if(col==c)
            ans.push_back(S[i]);
      i+=2;
```

```
13.     dfs(c,S,ans,i,col-1);
14.     dfs(c,S,ans,i,col+1);
15.     i++;
16.     return;
17.   }
18.   int main()
19.   {
20.     int t;
21.     cin>>t;
22.     while(t--)
23.     {
24.          int c;
25.          string S;
26.          cin>>c>>S;
27.          vector<char> ans;
28.          int i=0;
29.          dfs(c,S,ans,i,0);
30.          int l=ans.size();
31.          if(l==0)
32.               cout<<"Common Gandhijee!\n";
33.          else
34.          {
35.               sort(ans.begin(),ans.end());
36.               for(int j=0; j<l; j++)
37.                    cout<<ans[j];
38.               cout<<"\n";
39.          }
40.     }
41.     return 0;
42.   }
```