CSE 4304:Data Structure Lab-08
Graph Basic

Namisa Najah Raisa 210042112

November 2023

1 Problem B:Kefa and Park

In this problem I had to find the number of distinct leaves in the tree where Kefa can go to a restaurant without encountering more than 'm' consecutive vertices with cats on the path from house to the restaurant.

```
#include <iostream>
#include <vector>
using namespace std;
const int MAX = 100001;
vector<int> tree[MAX];
int cats[MAX];
int consecutive_cats[MAX];
int n, m, ans;
void dfs(int v, int parent, int total_cats)
{
    if (cats[v] == 1)
    {
        total_cats++;
        if (total_cats > m)
        {
            return;
        }
    }
    else
    {
        total_cats = 0;
    }
    if (tree[v].size() == 1 && v != 1)
    {
        ans++;
        return;
    }
```

```
    for (int u : tree[v])
    {
        if (u != parent)
        {
            dfs(u, v, total_cats);
        }
    }
}
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
    {
        cin >> cats[i];
    }
    for (int i = 1; i < n; i++)
    {
        int x, y;
        cin >> x >> y;
        tree[x].push_back(y);
        tree[y].push_back(x);
    }
    ans = 0;
    dfs(1, 0, 0);
    cout << ans << endl;
    return 0;
}
```

I solved this using DFS to traverse the tree.

'const int MAX=100001' refers to the maximum number of vertices in the tree.

'vector<int> tree[MAX]' refers to the tree structure.tree[i] contains the neighbours of vertex i.

'int cats[MAX]' refers to the information of the cats present at each vertex of the tree.

'int consecutive_cats[MAX];' is intended to keep tack of the consecutive cats encountered in the path from the root to the current vertex.

'int n,m,ans' refers to the vertices of the tree,maximum number of consecutive vertices with cats that Kefa is okay with,the final count of the the leaves where Kefa can go to a restaurant without encountering too many consecutive cats.

'void dfs(int v, int parent, int total_cats())'→ takes 3 parameters.'v' is the current vertex being visited,'parent' is the parent of the current ver-

tex,'total_cats is the number of consecutive cats encountered so far on the path from the root to the current vertex.

'if (cats[v]==1)' checks if the current vertex has a cat(represented by 1).If it has then the total_cats is incremented(indicates one more consecutive cat has been encountered).If it exceeds the number of allowed consecutive cats(>m) it returns from the current DFS branch.Kefa can't go on this path.

'else'→If the current vertex does not have a cat it resets the total_cats count to zero(indicating consecutive cats are only counted when encountered and are reset to zero when a non-cat vertex is visited.

'if (tree[v].size() == 1 && v != 1)'→This checks if the current vertex is a leaf node and if it's not the root.If these conditions are met,it means that 'v' is a restaurant Kefa can go to and 'ans' is incremented.Then it returns from the current DFS branch.No need to go further.

'for (int u : tree[v])'→This FOR loop iterates over all the neighbours 'u' of the current vertex 'v'.It checks if 'u' is the parent of 'v'.So the function doesn't backtrack to the vertex it came from in the DFS traversal.If 'u' is not the parent then the function is called recursively on 'u'.

The main function takes input of 'n' and 'm'.Then takes input of cats[i] from 1 to n(if the vertex has a cat or not).

'for (int i = 1; i < n; i++)'→This loops iterates from 1 to (n-1).It reads the tree's edges.Each edge is represented by a pair of integers 'x' and 'y'. 'tree[x].push_back(y); and tree[y].push_back(x);' add vertices 'x' and 'y' to the adjacency list of each other.This constructs the tree structure by creating the edges between the vertices.

]'ans=0' initializes ans to zero.

'dfs(1,0,0)'→It starts from the root(1),sets parent of the root to zero and initializes the total_cats to zero.

Finally the prints the ans as in the number of eligible restaurants for Kefa.

_____X_____