Problem - K - Codeforces

简介

给定一棵树：
根节点处有无数支军队。不断地，可以使用一个单位地时间，将一支军队转移到相邻地节点。问至少要花费多少地时间，使得每一个节点都被军队遍历过。

solve

1. 考察最终的答案。最终每一支军队都将会分布在叶子节点处。

问几个问题：

探究解空间：

1. 一颗树的节点会被怎样遍历？

   考虑每一个儿子子树的遍历情况。

   1. 没有军队停留在树的叶子下：
   2. 至少一个军队进入：军队停留在叶子上。
   3. 如果至少一支军队停留，那么每一支军队都将停留在子树的叶子节点下。

状态定义：

$f_{u,0/1}$分别表示，有没有下派军队情况下，处理以u为根的子树的最小代价。

状态转移方程为：

$f_{u,\ 0} = \sum(f_{v,0} + 2)$

$f_{u,1}$ 若干$f_{v,1}$, $f_{v,0}$的组合。同时注意至少有一个$f_{v,1}$

code (dfs版本)

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const int oo = 0x0fffffff;
const int N = 1E6 + 10;

vector<int> e[N];
int dp[N][2];
int dep[N];


void dfs(int u) {
    dp[u][0] = 0;
    dp[u][1] = dep[u];
    for (auto v : e[u]) {
```

```
        dep[v] = dep[u] + 1;
        dfs(v);
        dp[u][1] = min({dp[u][1] + dp[v][0] + 2, dp[u][0] + dp[v][1] , dp[u][1] + dp[v][1]});
        dp[u][0] += dp[v][0] + 2;
    }
}


void work(int testNo)
{

    int n; cin >> n;
    for (int i = 1; i <= n; i++) e[i].clear();
    for (int i = 2; i <= n; i++) {
        int x;
        cin >> x;
        e[x].push_back(i);
    }
    dfs(1);
    cout << "Case #" << testNo << ": "  << min(dp[1][1] , dp[1][0])  << '\n';
}


int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    int t; cin >> t;
    for (int i = 1; i <= t; i++)work(i);
}
```

code2(常数优化版本)

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int oo = 0x0fffffff;
const int N = 1E6 + 10;

int dp[N][2];
int dep[N];
int fa[N];
void work(int testNo)
{

    int n; cin >> n;
    for (int i = 2; i <= n; i++) {
        cin >> fa[i];
        dep[i] = dep[fa[i]] + 1;
    }
    for (int i = 1; i <= n; i++) {
        dp[i][0] = 0;
```

```
        dp[i][1] = dep[i];
    }
    for (int v = n; v > 1; v--) {
        int u = fa[v];
        dp[u][1] = min({dp[u][1] + dp[v][1] , dp[u][1] + dp[v][0] + 2, dp[u][0] + dp[v][1]});
        dp[u][0] += dp[v][0] + 2;
    }
    cout << "Case #" << testNo << ": "  << dp[1][1]  << '\n';
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    int t; cin >> t;
    for (int i = 1; i <= t; i++)work(i);
}
```