**Problem-3**

**Aim:**

You are given a tree with *N* vertices (numbered 11 through *N*) and a sequence of integers 1,2,…,*A*1​,*A*2​,…,*AN*​. You may choose an arbitrary permutation 1,2,…,*p*1​,*p*2​,…,*pN*​ of the integers 11 through *N*. For each valid *i*, you should assign the value *Api*​​ to vertex *i*.

Then, you should choose a vertex of the tree and −1*K*−1 times, perform one of the following operations:

* Move forward — move to a vertex which is adjacent to your current vertex. However, you must not move to the vertex in which you were immediately before the previus operation (if it exists).
* Turn around — stay in your current vertex. You may only do this if it is impossible to move forward. Since you do not move in this operation, you may move forward again afterwards.

In this process, you obtain a sequence of vertices 1,2,…,*v*1​,*v*2​,…,*vK*​ — the initial vertex and the vertices in which you were after each operation. Your *score* is ∑=1∑*i*=1*K*​*Apvi*​​​. What is the maximum possible value of this score?

1. **Problem Description :**

A tree is given having N vertices and a sequence of integers A1, … AN. We take arbitrary permutation of  
integers 1 to N and assign the value Api to the vertex I. Then we will choose a vertex and move forward and  
move backward (K-1) times. We will obtain a sequence of vertices v1,…,vk i.e. the initial vertex and the  
vertices in which you were after each operation. Then we will calculate the score using the formula and find its  
maximum possible value for this score.

1. **Algorithm :**

Take input n, k

Create vector of size n.

For Loop (1 to n)

For Loop (1 to n)

Input x and y

v[x].push\_back(y);

v[y].push\_back(x);

set r = 1;

for loop (1to n)

check if(v[i].size() == 1)

set r = i;

sort(array a in descending);

set ans = 0, s = 0;

for loop (0 to mn)

set s += a[i];

check if((k / mn) & 1)

ans = (k / mn - 1) \* s;

k = k % mn + mn;

else

ans = k / mn \* s;

k = k % mn;

if(k < 3)

ans += a[0] \* (k >= 1);

ans += a[1] \* (k == 2);

else for(int i = 0; i < mn && k > 0; i++)

ans += min(k, 2) \* a[i];

k -= 2;

Print ans

1. **Source Code for Experiment :**

#include <bits/stdc++.h>

using namespace std;

using ll = long long;

using ld = long double;

#define ft first

#define sd second

constexpr ll N = ll(3e5) + 5;

constexpr int MOD = int(1e9) + 7;

constexpr int inf = 0x3f3f3f3f;

int a[N], d[N], mn;

vector<int> v[N];

int dfs(int x, int p = -1){

if(v[x].size() == 1 && p != -1){

mn = min(mn, d[x]);

return 1;

}

int m1 = inf, m2 = inf;

for(auto& i : v[x]){

if(i == p) continue;

d[i] = 1 + d[x];

int k = dfs(i, x);

if(k < m1){

m2 = m1;

m1 = k;

}

else if(k < m2) m2 = k;

}

if(m2 != inf) mn = min(mn, m1 + m2);

return m1 + 1;

}

void solve(){

int n, k;

cin >> n >> k;

for(int i = 0; i < n; i++) cin >> a[i];

for(int i = 1; i < n; i++){

int x, y;

cin >> x >> y;

v[x].push\_back(y);

v[y].push\_back(x);

}

int r = 1;

for(int i = 1; i <= n; i++){

if(v[i].size() == 1) r = i;

}

mn = inf;

d[r] = 0;

dfs(r);

mn++;

sort(a, a + n, greater<int>());

ll ans = 0, s = 0;

for(int i = 0; i < mn; i++) s += a[i];

if((k / mn) & 1){

ans = (k / mn - 1) \* s;

k = k % mn + mn;

}

else{

ans = k / mn \* s;

k = k % mn;

}

if(k < 3){

ans += a[0] \* (k >= 1);

ans += a[1] \* (k == 2);

}

else for(int i = 0; i < mn && k > 0; i++){

ans += min(k, 2) \* a[i];

k -= 2;

}

cout << ans << '\n';

for(int i = 1; i <= n; i++) v[i].clear();

}

int main() {

int tt = 1;

cin >> tt;

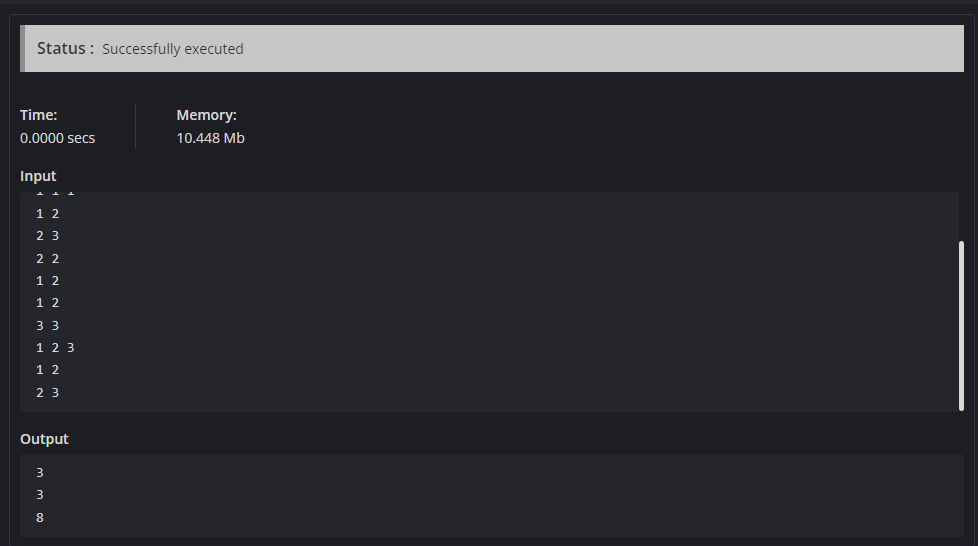
for(int i = 0; i < tt; i++) {

solve();

}

}

1. **Result/Output :**

****

**Learning outcomes (What I have learnt):**

1. Learnt about the linked list data structure in C++.
2. Learnt about the fast and slow pointers approach.
3. Learnt about how to find the middle node in linked list.
4. Learnt about how to reverse the linked list.
5. Learnt about how to check palindrome linked list.