Nhóm 5

1. Chứng minh thuật toán

Thuật toán dựa trên ý tưởng **Dynamic Programming trên cây** kết hợp với **Greedy**. Cụ thể:

- Với mỗi nút u, ta định nghĩa $c[u]$ là giá trị ban đầu của nút cộng số con của nó: $c[u] = w[u] + \deg(u)$.
- Duyệt cây theo thứ tự **DFS từ lá lên gốc**.
- Tại mỗi nút u, trước tiên giải quyết tất cả các con của nó, sau đó sắp xếp con theo giá trị $c[v]$ tăng dần.
- Với mỗi con v, nếu tổng $c[u] + c[v] - 1 \leq m$, thực hiện **gộp** v **vào** u, tăng biến đếm ans và cập nhật $c[u]$.

Tính đúng đắn:

- Duyệt DFS từ lá lên gốc đảm bảo khi xử lý u, tất cả các con đã được tối ưu hóa.
- Sắp xếp con theo $c[v]$ tăng dần đảm bảo gộp được nhiều con nhất mà vẫn không vượt quá m.
- Cập nhật $c[u] = c[u] + c[v] - 1$ phản ánh đúng tổng chi phí khi gộp v vào u.

Như vậy, thuật toán đảm bảo tối đa hóa số lần gộp mà không vượt quá giới hạn m.

2. Cài đặt thuật toán

Listing 1: C++ Implementation

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

const int maxn = 696969 + 6;
vector<int> adj[maxn];
long long c[maxn];

int n, m, ans;
int w[maxn];

bool cmp(int a, int b){
    return c[a] <= c[b];
}

void solve(int u){
    for(int v : adj[u]){
        if(!adj[v].empty()) solve(v);
    }
    sort(adj[u].begin(), adj[u].end(), cmp);
    for(int v : adj[u]){
```

```
        if(c[u] + c[v] - 1 <= m){
            c[u] = c[u] + c[v] - 1;
            ans++;
        }
    }
}

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    cin >> n >> m;
    for(int i = 1; i <= n; i++) cin >> w[i];
    for(int i = 2; i <= n; i++){
        int pi; cin >> pi;
        adj[pi].push_back(i);
    }
    for(int i = 1; i <= n; i++) c[i] = w[i] + adj[i].size();
    solve(1);
    cout << ans;
    return 0;
}
```

3. Phân tích độ phức tạp thời gian

- Duyệt DFS toàn bộ cây: $O(N)$.
- Tại mỗi nút u, sắp xếp các con theo $c[v]$: $O(\deg(u) \log \deg(u))$.
- Tổng chi phí sắp xếp cho tất cả các nút: $\sum_u O(\deg(u) \log \deg(u)) \leq O(N \log N)$.
- **Tổng thời gian:** $O(N \log N)$.

4. Phân tích độ phức tạp không gian

- Lưu danh sách kề: $O(N)$.
- Mảng c, w: $O(N)$.
- **Tổng bộ nhớ sử dụng:** $O(N)$, với DFS chiếm $O(\log N)$ cho stack.