1. Phân tích bài toán

- Dễ thấy bài toán có thể đưa về dạng đồ thị cây: Mỗi đỉnh là 1 nhân viên và các con trực tiếp của đỉnh này là các nhân viên được nhân viên trên quản lý. Đỉnh gốc là đỉnh 1, tức CEO. Bài toán từ việc loại bỏ nhân viên thành loại bỏ các đỉnh của cây.

2. Thuật toán

- Giải sử tại cây con gốc u, ta đã giải được bài toán cho các cây con gốc v, để giải bài toán cho cây con gốc u, gọi $C_i = w_i + size_i - 1$ là độ phức tạp tại cây con i với $size_i$ là số lượng đỉnh con trực tiếp của i, ta sẽ chọn ra các cây con i trong các cây con gốc v của u sao cho:

$$C_u + \Sigma C_i < m$$

- Để tối ưu, ta sẽ chọn tối đa k đỉnh con bé nhất sao cho thỏa mãn bất đẳng thức trên, sau đó gán $C_u + \Sigma C_x$.
- Từ đây ta có thuật toán tham lam sau: Chạy thuật toán dfs, với đỉnh u đang xét, từ các đỉnh v đã dfs, ta tính toán lại $C_u = C_u + \Sigma_{i=1}^{k} C_i$ với $i = 1..k$ là k đỉnh có C bé nhất trong các đỉnh con của u và k là lớn nhất có thể.
- Có thể thấy thuật toán tham lam trên là đúng:
  1. Thứ tự chọn các đỉnh là không quan trọng.
  2. Xóa đỉnh chỉ làm cho độ phức tạp cộng việc của đỉnh cha lớn hơn(trừ các đỉnh lá có $w = 0$ nhưng ta có thể bỏ qua trường hợp này.)
  3. Xét nút lá, nếu nút lá này ta không thể xóa(nghĩa là khi xóa nút này ta đã phạm điều kiện) thì ta có thể coi như không tồn tại nút lá này và ta có thể gỡ nó khỏi cây(nhưng không tính vô đáp án), vì vậy việc chọn các phần tử C bé nhất và cộng vào nút cha là 1 phép xóa tối ưu.

3. Cài đặt

```cpp
#include<bits/stdc++.h>
using namespace std;
const int nmax = 696969;

int n, m;
long long c[nmax+1];
vector<int> a[nmax+1];

void input(){
  cin >> n >> m;
```

```cpp
  for(int i=1;i<=n;++i)
    cin >> c[i],
    c[i]-=1;
  for(int i=2, par;i<=n;++i)
    cin>>par,
    c[par]++,
    a[par].push_back(i);
}

int ans = 0;


void dfs(int u){
  vector<long long> s;
  for(int &v:a[u])
    dfs(v),
    s.push_back(c[v]);


  sort(s.begin(), s.end());

  for(auto &x:s)
    if(c[u] + x >= m) break;
    else c[u]+=x, ++ans;
}

void solve(){

  input();
  dfs(1);
  cout<<ans;

}

int main(){
  ios_base::sync_with_stdio(false);
  cin.tie(0),cout.tie(0);

  solve();
  return 0;
}
```