Nhóm 9

Phân tích, chứng minh và hiện thực thuật toán

Đề bài

Bạn đang làm ở văn phòng quản lý nhân sự trong một công ty gồm n người. Mỗi nhân viên có đúng một người quản lý (trừ nhân viên số 1, là CEO).

Nhân viên thứ i có khối lượng công việc w_i. Định nghĩa chỉ số phức tạp công việc của nhân viên i là:

$$C_i = w_i + \sum_{j \in \text{con của } i} 1$$

(khối lượng công việc cộng với số lượng nhân viên trực tiếp dưới quyền).

Bạn được giao nhiệm vụ **cắt giảm nhân sự**. Khi một nhân viên bị sa thải:

- Các nhân viên dưới quyền họ được chuyển cho quản lý trực tiếp của họ.
- Khối lượng công việc của người bị sa thải cũng được cộng dồn cho quản lý trực tiếp của họ.

Công ty quy định rằng chỉ số phức tạp công việc của mọi nhân viên **không được vượt quá** m. Bạn không thể sa thải CEO (nhân viên số 1).

Hãy tìm **số lượng nhân viên tối đa có thể sa thải** sao cho sau khi thực hiện, chỉ số phức tạp của mọi người đều $\leq m$.

Input:

- Dòng 1: hai số nguyên n, m.
- Dòng 2: n số nguyên $w_1, w_2, \ldots, w_n$.
- $n-1$ dòng tiếp: dòng thứ i chứa p_i – quản lý của nhân viên $i+1$.

Output: In ra một số nguyên duy nhất – số lượng nhân viên có thể sa thải tối đa.

Phân tích và hướng giải

Gọi cây quản lý với gốc là nhân viên 1 (CEO).

Với mỗi đỉnh x, ta cần xác định:

- $C(x)$ – chỉ số phức tạp của x sau khi đã xử lý toàn bộ cây con của nó.
- $cnt(x)$ – số lượng nhân viên bị sa thải tối đa trong cây con gốc x.

Chiến lược: Duyệt DFS hậu thứ tự – xử lý tất cả các con của x trước, rồi mới đến x.

Sau khi xử lý xong các con, ta có danh sách:

$$S(x) = \{(C(y), y) \mid y \text{ là con của } x\}$$

sắp xếp theo $C(y)$ tăng dần.

Hiện tại, chỉ số phức tạp của x là:

$$C_x = w[x] + |S(x)|$$

(vì mỗi con của x làm tăng phức tạp thêm 1).

Khi **xóa** một con y:

- Số con của x giảm 1 $\Rightarrow C_x$ giảm 1.
- Nhưng khối lượng công việc của y được cộng vào x, tức là $w[x]$ tăng thêm $C(y)$.

- Khi đó C_x mới $= C_x - 1 + C(y)$.

Do đó, để không vượt quá giới hạn m, ta chỉ được phép xóa y nếu:

$$C_x - 1 + C(y) \leq m.$$

Tham lam: Ta sắp xếp các con theo $C(y)$ tăng dần, vì khi chọn xóa những con có $C(y)$ nhỏ, chi phí tăng C_x là nhỏ nhất, giúp ta có thể xóa được nhiều con hơn.

Như vậy, tại mỗi đỉnh:

1. Khởi tạo $C_x = w[x]+$ số lượng con ban đầu.
2. Duyệt lần lượt các con theo $C(y)$ tăng dần:
   - Nếu $C_x - 1 + C(y) \leq m$, ta xóa y.
   - Cập nhật lại $C_x = C_x - 1 + C(y)$.

Thuật toán này đảm bảo tối đa hóa số lượng đỉnh bị xóa, vì mỗi lần xóa, ta chọn phương án có chi phí nhỏ nhất cho x — một lựa chọn tham lam địa phương tối ưu.

Chứng minh tính đúng đắn

Giả sử tại đỉnh x, ta chọn xóa các con $y_1, y_2, \ldots, y_k$.

Giả sử tồn tại hai con a, b với $C(a) > C(b)$ mà ta lại chọn xóa a mà không xóa b. Khi hoán đổi quyết định — xóa b thay vì a:

$$\Delta C_x = (C_x - 1 + C(b)) - C_x = C(b) - 1 < C(a) - 1 = \Delta C'_x.$$

Tức là việc xóa b "rẻ" hơn, làm C_x tăng ít hơn, nên giúp ta có khả năng xóa thêm các đỉnh khác. Vì vậy, chiến lược xóa theo $C(y)$ tăng dần là tối ưu (nguyên lý tham lam chuẩn).

Kết hợp với việc duyệt DFS từ lá lên gốc, đảm bảo rằng khi xét x, mọi cây con đã được xử lý tối ưu.

Độ phức tạp

Mỗi đỉnh được duyệt đúng một lần, các con của nó được sắp xếp:

$$O(n \log n)$$

là độ phức tạp tổng thể chấp nhận được với $n \leq 7 \times 10^5$.

Mã nguồn C++

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

const int N = 7e5 + 7;
vector<int> adj[N];
int w[N], n, m, C[N];
bool isDel[N];
int size_child[N];
```

```
10
11      void dfs(int x) {
12        for (auto y : adj[x]) dfs(y);
13        vector<pair<int,int>> vec;
14        for (auto y : adj[x]) vec.push_back({C[y], y});
15        sort(vec.begin(), vec.end());
16
17        int c = w[x] + (int)adj[x].size();
18        size_child[x] = adj[x].size();
19
20        for (auto [cy, y] : vec) {
21          if (c - 1 + cy <= m) {
22            c = c - 1 + cy;
23            size_child[x] += size_child[y] - 1;
24            w[x] += cy;
25            isDel[y] = true;
26          }
27        }
28        C[x] = c;
29      }
30
31      signed main() {
32        ios_base::sync_with_stdio(0);
33        cin.tie(0);
34
35        cin >> n >> m;
36        for (int i = 1; i <= n; i++) cin >> w[i];
37        for (int i = 1; i < n; i++) {
38          int p; cin >> p;
39          adj[p].push_back(i + 1);
40        }
41
42        dfs(1);
43        int res = 0;
44        for (int i = 1; i <= n; i++)
45        if (isDel[i]) res++;
46        cout << res;
47        return 0;
48      }
```