Metal Gezinti

March 15, 2021

Tam çözüme başlamadan önce öncelikle herhangi bir düğümü başlangıç kabul edelim ve o düğüm için cevabı hesaplayalım. Başlangıç kabul ettiğimiz düğümün indisini x kabul edelim ve ağacımızın kök düğümünü de x yapalım. Kök düğümden başlayan ve sürekli artan bir şekilde ilerlediğimiz patikalar bizim için önemli. Eğer bir patikada azalan iki komşu var ise bu durum bizim sorumuz için kabul edilebilir durum değildir. Okuyucu çizerek ve inceleyerek bunu rahatlıkla görebilir. Daha sonra öyle bir y değeri bulalım ki her $z(x \leq z \leq y)$ düğümüne bu bulduğumuz patikaları kullanarak ulaşılabilir olsun. Bulduğumuz en büyük y değeri için cevabımız $y - x + 1$ olacaktır.

Ancak bizim bu hesabı her başlangıç düğümü için yapmamız gerekiyor. Üstteki çözümü her başlangıç noktası için ayrı ayrı yaptığımız zaman süre limitinden dolayı tam çözüme ulaşamıyoruz. Daha hızlı bir çözüm için düğümleri azalan sırada ağacımıza ekleyerek her eklediğimiz düğüm için o anki düğümün cevabını hesaplayıp tüm cevabımıza ekleyeceğiz.

Her düğüm eklendiğinde elimizde bir orman oluşacak ve ormanımızdaki her ağacın kök düğümü o ağacın en küçük elemanı olacak. Herhangi bir x düğümünü şuanki ormanımıza eklediğimizi düşünelim. x düğümüne komşu tüm düğümleri gezerek iki farklı durum üzerinden inceleme yapalım. x düğümüne komşu olan herhangi bir y düğümünü ele alalım.

- Eğer y düğümü kendi ağacının kök düğümü ise: Düğümleri azalan sırada incelediğimiz için $x \leq y$ olduğunu biliyoruz o halde; y düğümünden ulaştığımız tüm düğümlere x düğümünden de ulaşılabilir.
- Eğer y düğümü kendi ağacın kök düğümü değilse: O zaman y düğümünün ağacındaki kök düğümüne hiçbir zaman artan sırada ulaşamayacağız, çünkü y'nin bulunduğu ağaçtaki en küçük indisli düğüm kök düğümdür. Cevabımız dolayısıyla en fazla y ağacının kök düğümünün indisinden bir eksik olabilir.

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

int n,mx;
long long ans;
set <int> roots;
vector <int> dad;
vector <vector <int>> ed;

int find_dad(int x) {
        if (x == dad[x])
                return x;
        return dad[x] = find_dad(dad[x]);
}

void f(int x) {
        for (int u : ed[x]) {
                int v = find_dad(u);
                if (u != v)
                        mx = min(mx,v-1);
                dad[v] = x;
                roots.erase(v);
        }

        int res = mx;
        if (roots.empty() == false)
                res = min(res,*roots.begin() - 1);
        roots.insert(x);
        ans += res - x + 1;
}

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

        cin >> n;

        ed.resize(n+1);
        dad.resize(n+1);

        for (int i = 0 ; i < n - 1 ; i++) {
                int u,v;
                cin >> u >> v;
                if (u > v)
                        swap(u,v);
                ed[u].push_back(v);
        }

        for (int i = 1 ; i <= n ; i++)
                dad[i] = i;

        mx = n;
        for (int i = n ; i >= 1 ; i--)
                f(i);
        cout << ans << endl;
}
```