Zeynep'in Ağacı Çözüm

Kısaca sorudan bahsetmek gerekirse bize 1 root olacak şekilde ağaç veriliyor. Her node'un bir değeri var. Bizden istenen ise 1'in de içinde bulunduğu en büyük **Binary Search Tree (BST)**'nin size'ını ekrana yazdırmak.

Çözüm için bir fonksiyon dizayn ediyorum öyle ki bu rekürsif fonksiyon bana verdiğim parametrelerden (min,max) aralığında bulunan alt node'ları kullanarak oluşturulabilecek en büyük BST size'ını verecek.

Bu fonksiyon'u doğru bir şekilde yazabilirsek bizim cevabımız fonk(node=1 , min=0 , max=10^9) fonksiyonunun geri döndüğü değer olacaktır.

Bu fonksiyonu her seferinde güncellerken bütün çocuklarıma çağırıp benden küçük eşit olan node'ların en büyüğü ve benden büyük node'ların en büyüğünün toplamının 1 fazlasını döndürüyorum.

Dikkat etmemiz gereken ilk nokta 1 eklemeyi unutmamalıyız çünkü çocuklarımdan hiç cevap gelmese bile ben kendimi saydığımda size'im 1 olmuş oluyor.

Bir diğer nokta ise her fonksiyonu çağırdığımda min ve max değerlerini doğru güncellemeli ve cevaba eklerken eklediğim node'un o değerler arasında kalıp kalmadığıma çok dikkat etmeliyim.

Çözüm Kodu:
Dil: C++

```
int dfs(int x,int par,int mn,int mx){
	int sol=0;
	int sag=0;
	for(int i=0;i<v[x].size();i++){
		int go = v[x][i];
		if(go == par ) continue;
		if(a[go] < mn or a[go] > mx) continue;

		if(a[go] <= a[x])
			sol = max(sol, dfs(go,x,mn,a[x]));

		else
			sag = max(sag, dfs(go,x, a[x]+1,mx));

	}
	return sol+sag+1;
}

int main(){
	cin >> n;
	for(int i=1;i<=n;i++)
		cin >> a[i];
	for(int i=1;i<n;i++){
		cin >> x >> y;
		v[x].push_back(y);
```

```
                v[y].push_back(x);
        }
        cout << dfs(1,-1,0,mod) << endl;
}
```