Week 12

2024-09-19

2225 합분해

dp[val][cnt]

→ cnt 개의 숫자를 를 써서 val 을 만드는 경우의 수

정답은 dp[n][k] 를 구하면 됨

점화식은

dp[val][cnt] = Σ(0 ~ val) dp[val – i][cnt – 1]

2225 합분해

탈출 조건은 cnt == 0 일때

val == 0 이라면 가능한 경우고

val != 0 인 경우 가능하지 않음

```
int go(int val, int cnt) {
    if(cnt == 0) {
        if(val == 0) return 1;
        return 0;
    }

    int &ret = dp[val][cnt];
    if(ret != -1) {
        return ret;
    }
    ret = 0;
    for(int i = 0; i <= val; i++) {
        ret += go(val - i, cnt - 1);
        ret %= mod;
    }
    return ret;
}
```

2250 트리의 높이와 너비

특정 노드의 x 축 위의 좌표를 구하려면 inorder 방식으로 순회를 하면 됨

깊이정보도 가지고 있는 채로 inorder 순회를 하며 해당 깊이의 가장 작은 값과 큰 값을 갱신해줌

2250 트리의 높이와 너비

루트 번호를 안 알려줘서 1 번으로 했는데 틀림 ;;;

루트를 직접 구해야 한다

주어지는 입력이 노드 번호와 그 노드의 왼쪽 오른쪽 '자식' 노드 번호가 주어지기 때문에 자식으로 한 번도 등록되지 않은 노드가 루트가 됨

(그냥 연결이 아님 !!)

그냥 연결이라면 루트를 찾기 까다롭다 .

4 번 규칙인 '노드가 배치된 가장 왼쪽 열과 오른쪽 열 사이엔 아무 노드도 없이 비어있는 열은 없다 .' 를 활용해야 하는데 루트가 하나만 나오는지도 모르겠고 모든 노드를 루트라고 생각하고 탐색을 해 봐야 한다 .

2250 트리의 높이와 너비

깊이정보도 가지고 있는 채로 inorder 순회를 하며 해당 깊이의 가장 작은 값과 큰 값을 갱신해줌

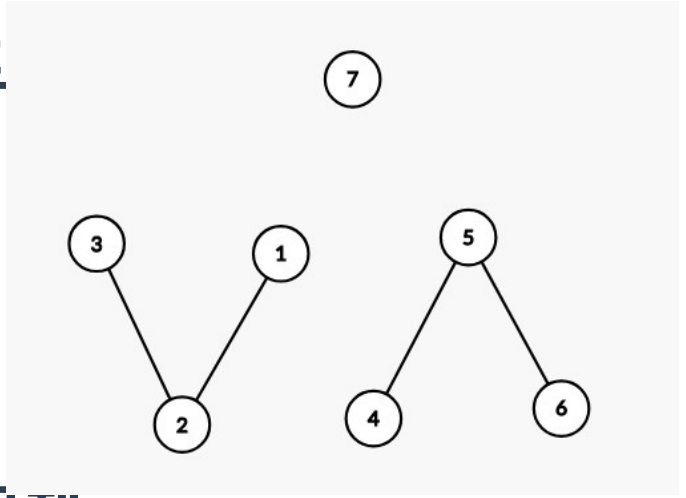
```
void go(int now, int depth) {
    if(now == -1) return ;
    go(g[now][0], depth + 1);
    dep[depth][0] = min(dep[depth][0], cnt);
    dep[depth][1] = max(dep[depth][1], cnt);
    cnt += 1;
    go(g[now][1], depth + 1);
}
```

4803 트리

트리는

특정 그래프 집합 내의 모든 노드가 연결되어 있고 싸이클이 없어야 함

오른쪽 그림은 3 개의 트리로 구성된 포레스트



주어진 그래프에서 트리가 몇 개 있는지 찾는 문제

4803 트리

입력으로 주어지는 간선 정보를 양방향으로 연결

→ DFS 를 이용해서 찾기

go 함수는 트리 집합인 경우 true 를 반환하고

그렇지 않은경우 false 를 반환함

```
bool go(int now, int p) {
    visited[now] = true;
    bool f = true;
    for(int next : g[now]) {
        if(next == p) continue;
        if(!visited[next]) f &= go(next, now);
        else f = false;
    }
    return f;
}
```

```
int cnt = 0;
for(int i = 1; i <= n; i++) {
    if(!visited[i] && go(i, 0)) {
        cnt += 1;
    }
}
```

4803 트리

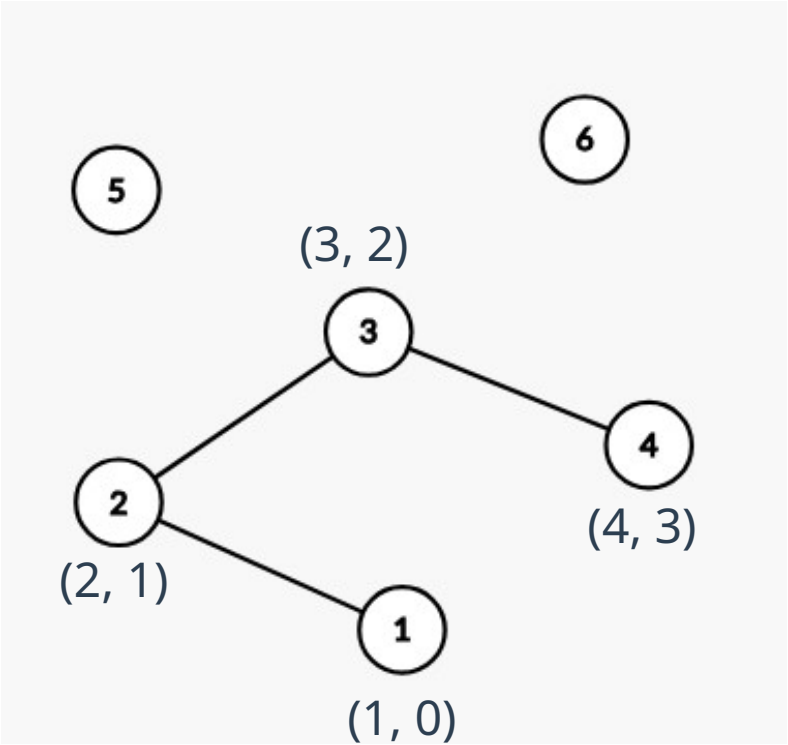
양방향 연결이기
때문에 부모쪽으로
이동할 수 있음

그래서 부모를 변수로
함께 넣어줌

```
bool go(int now, int p) {
    visited[now] = true;
    bool f = true;
    for(int next : g[now]) {
        if(next == p) continue;
        if(!visited[next]) f &= go(next, now);
        else f = false;
    }
    return f;
}
```

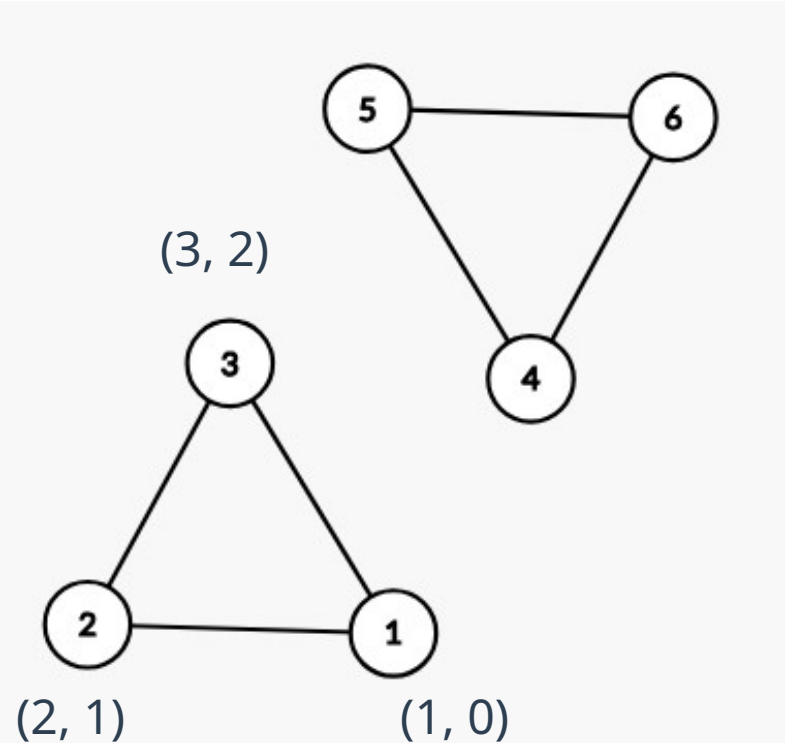
```
int cnt = 0;
for(int i = 1; i <= n; i++) {
    if(!visited[i] && go(i, 0)) {
        cnt += 1;
    }
}
```

4803 트리




```
bool go(int now, int p) {
    visited[now] = true;
    bool f = true;
    for(int next : g[now]) {
        if(next == p) continue;
        if(!visited[next]) f &= go(next, now);
        else f = false;
    }
    return f;
}
```

4803 트리




```
bool go(int now, int p) {
    visited[now] = true;
    bool f = true;
    for(int next : g[now]) {
        if(next == p) continue;
        if(!visited[next]) f &= go(next, now);
        else f = false;
    }
    return f;
}
```