

Tree & Graph

트리 기초, 그래프 기초, dfs, bfs

Dosawas 2024-05-18

목차

차례

- 6회차 리뷰
- Graph
- Tree
- DFS
- BFS
- 0-1 bfs 문제

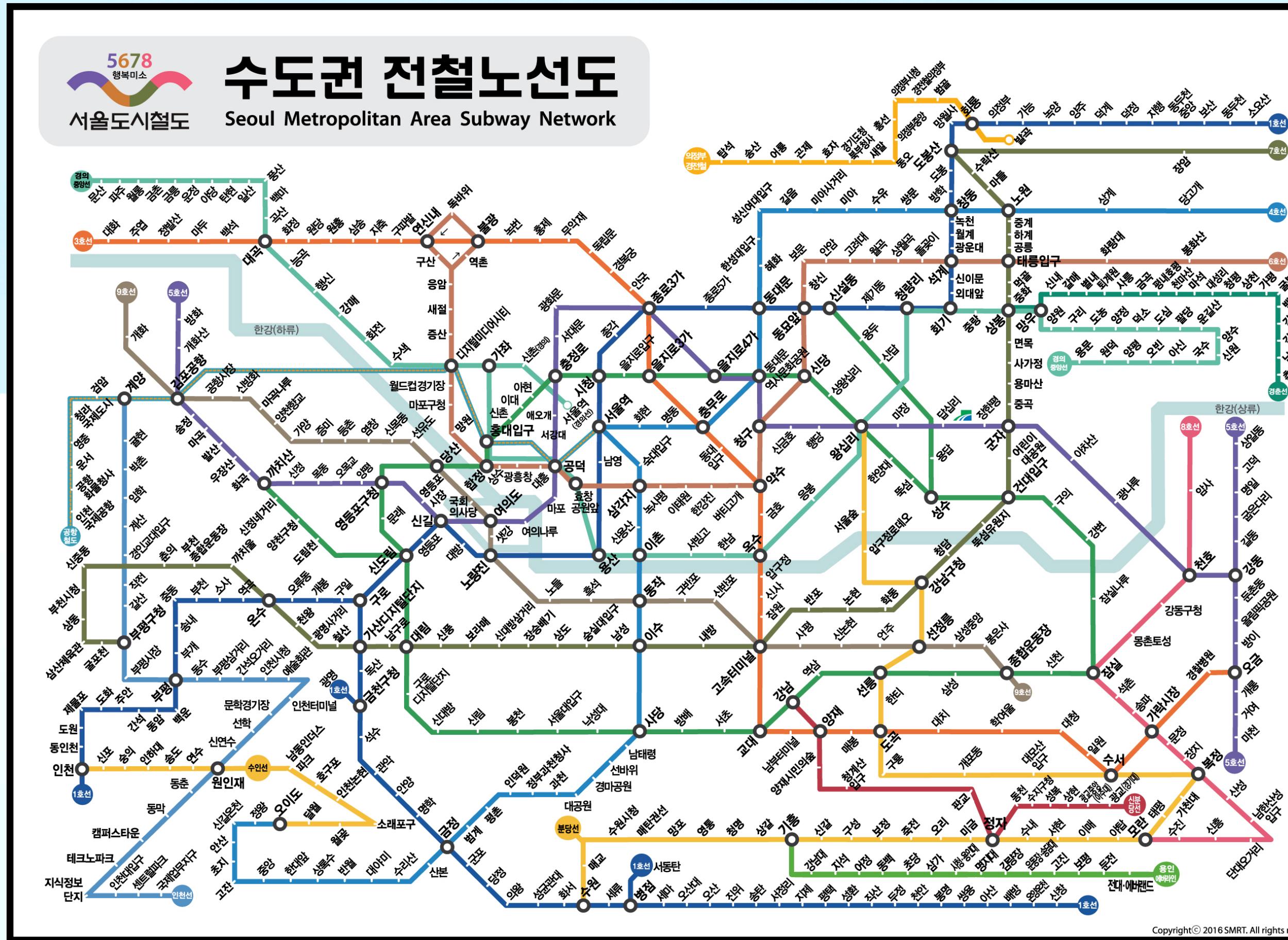
6회차 리뷰

- 소수 판정 : 에라토스테네스의 체, 소인수분해(factorization)
- 확장 유클리드 호제법
- 이항계수 :
 - DP로 이항계수 구하기
 - 모듈러역원을 이용한 이항계수 구하기
- 깃허브에 자기가 자주 쓰는 도구 저장하기

Graph

그래프도 자료구조다.

그래프는 객체들간의 상호관계를 나타낼 때 효과적이다

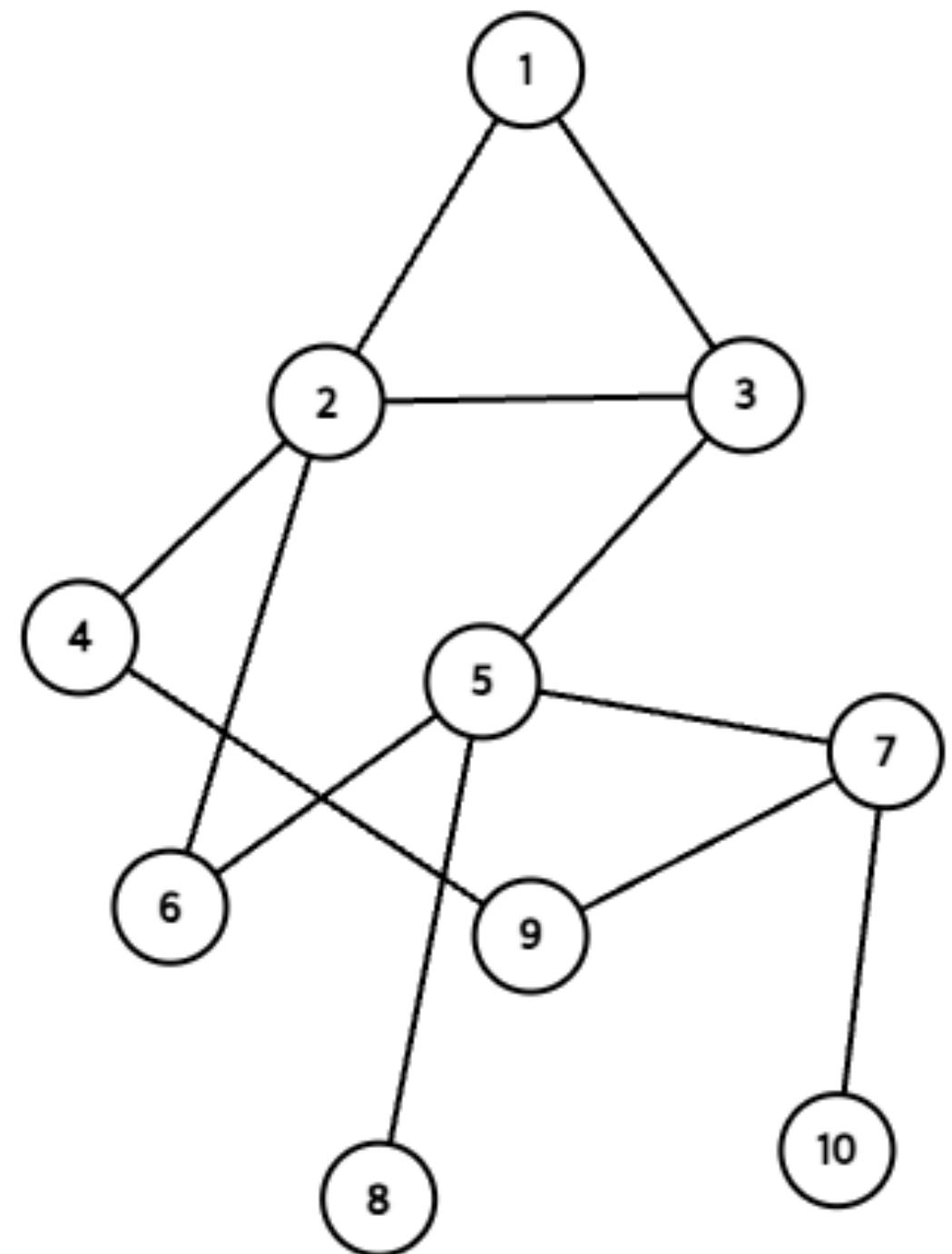


출처 : tvN

Graph

그래프도 자료구조다.

$G(V, E)$: Graph는 Vertex와 Edge로 이루어져있다.



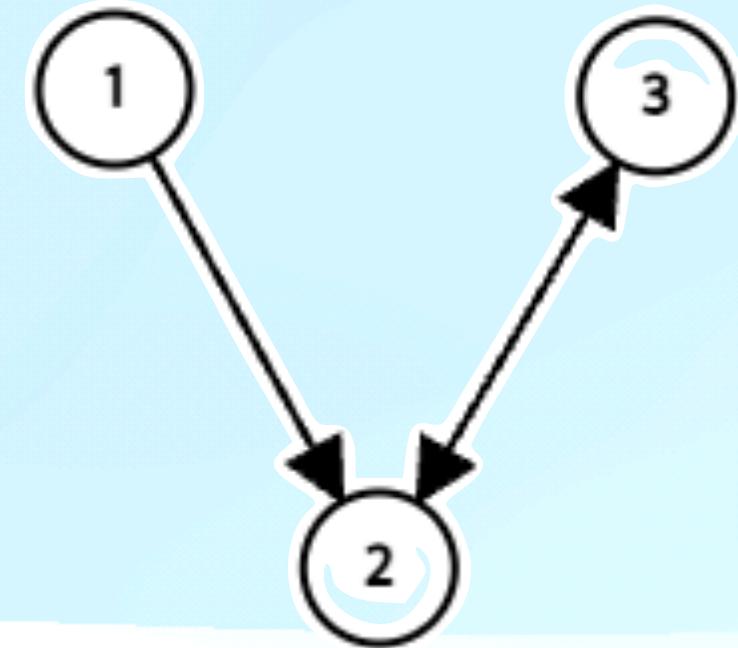
용어 :

1. Vertex - 정점, Edge - 간선
2. Path - 경로
 1. Simple path - 한 정점을 한번만 지남
 2. Cycle - 시작한 정점으로 다시 돌아옴

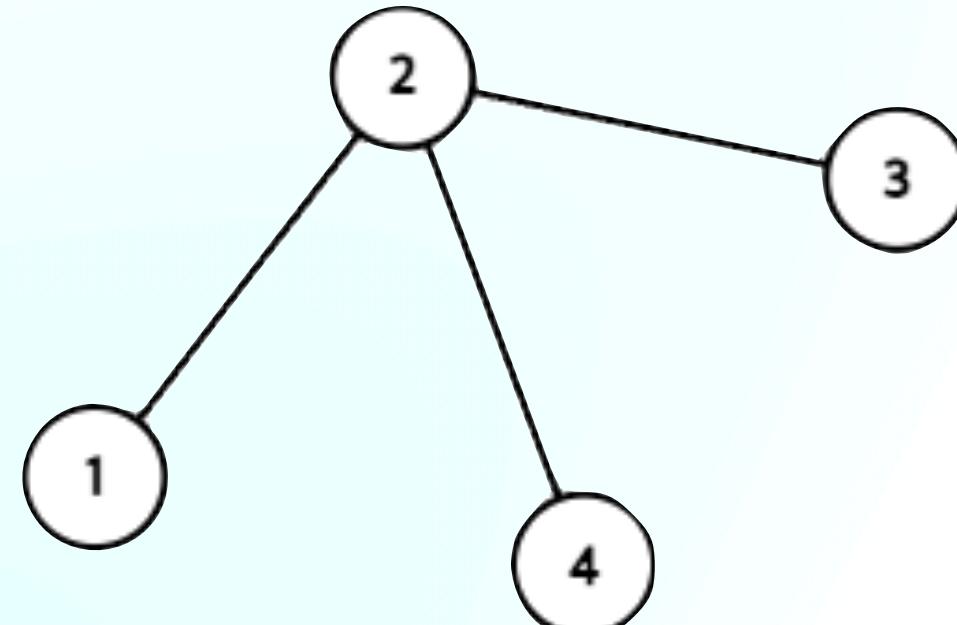
Graph의 종류

여러가지 그래프

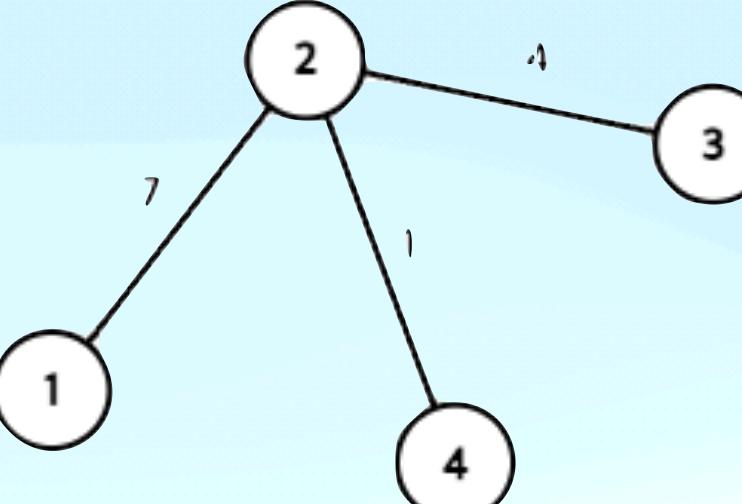
1. Directed graph



2. Undirected graph

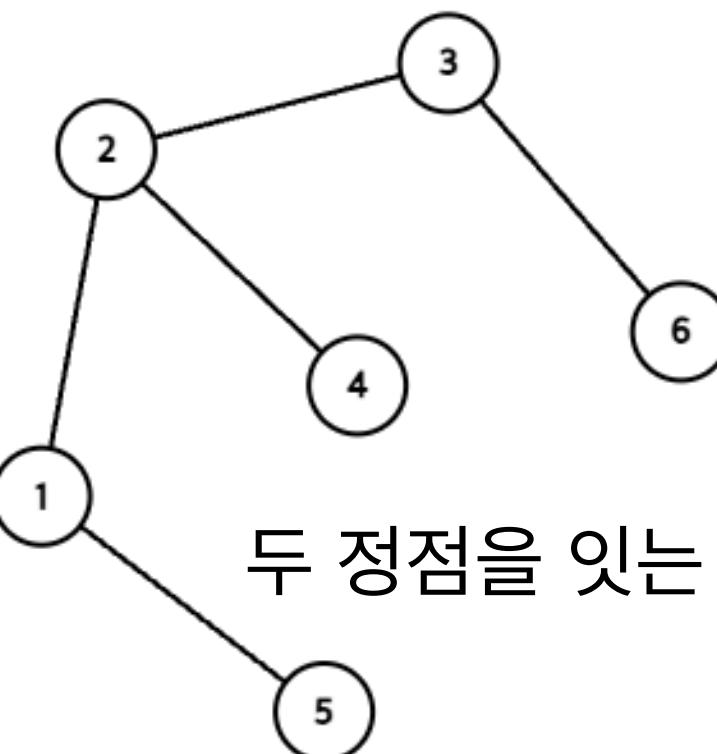


3. Weighted graph



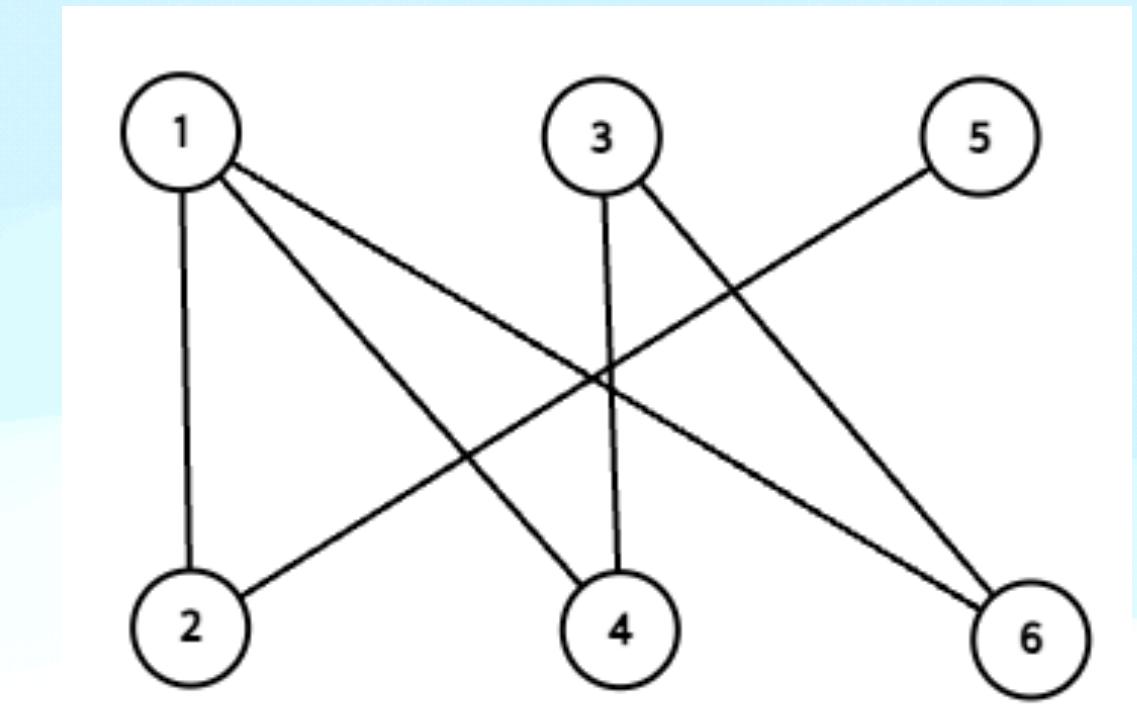
거리, 호감도 등

4. Tree(Unrooted tree)



두 정점을 잇는 방법이 딱 하나밖에 없다.

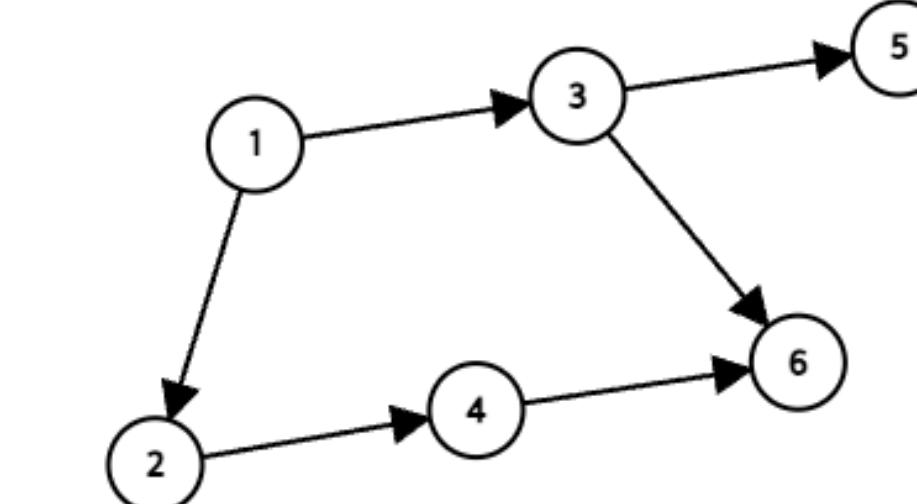
5. Bipartite graph(이분그래프)



하트시그널, 나는 솔로 짹궁 정하기?

여자끼리 선택 불가, 남자끼리 선택 불가

6. Directed acyclic graph(DAG)

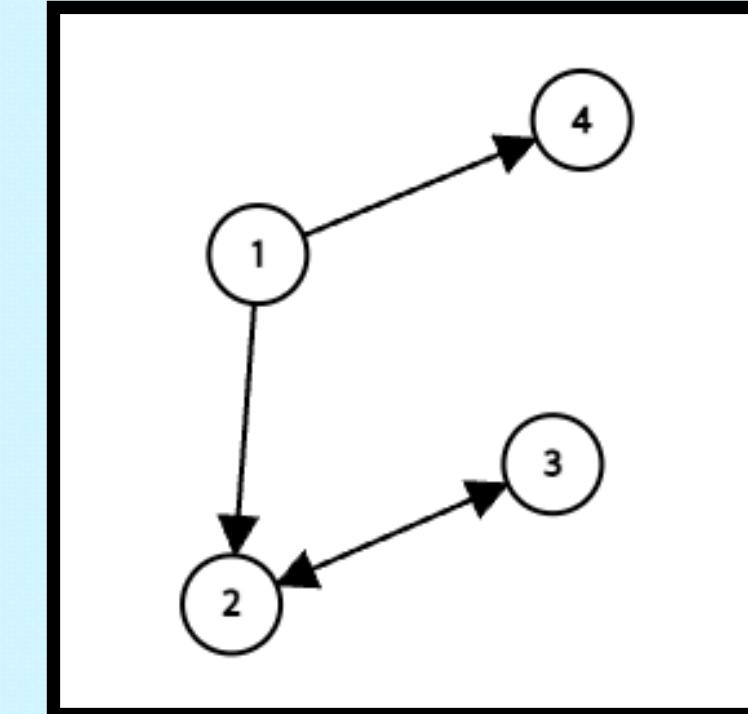


방향이 있는데 사이클이 없는 그래프

Graph 구현

인접 행렬과 인접 리스트 표현

- 인접 행렬 : 그래프의 연결관계를 2차원 배열로 표현



장점 : i에서 j로 가는길이 있는지
빠르게 확인 가능

i / j	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	1	0	0
4	0	0	0	0

단점 : 길이 많이 없는데 인접행렬을
쓰면 메모리 손실이 크다..

```
const int NODE_MAX = 101;
int adj [NODE_MAX] [NODE_MAX];
// i에서 j로 가는 길이 있는가?
// 가중치를 넣어도 됨
```

Graph 구현

인접 행렬과 인접 리스트 표현

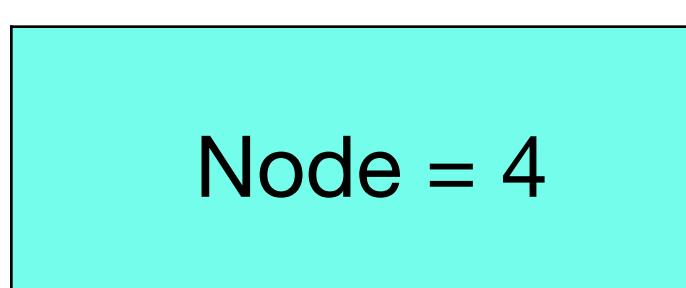
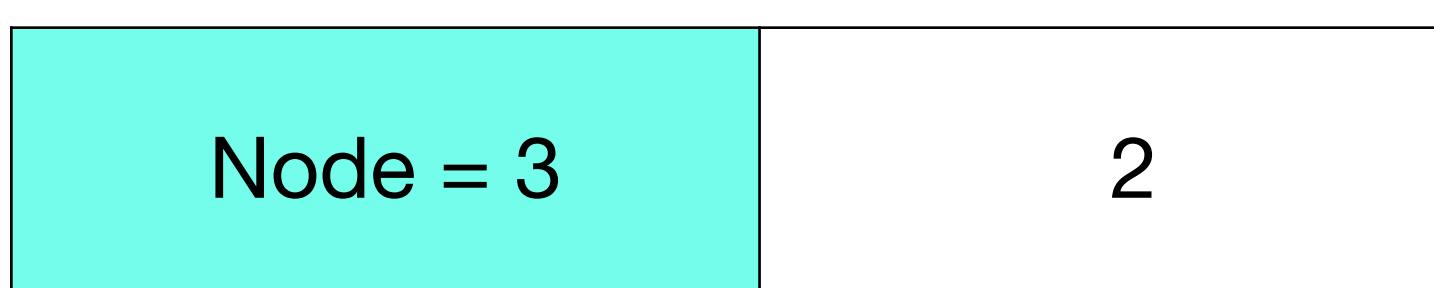
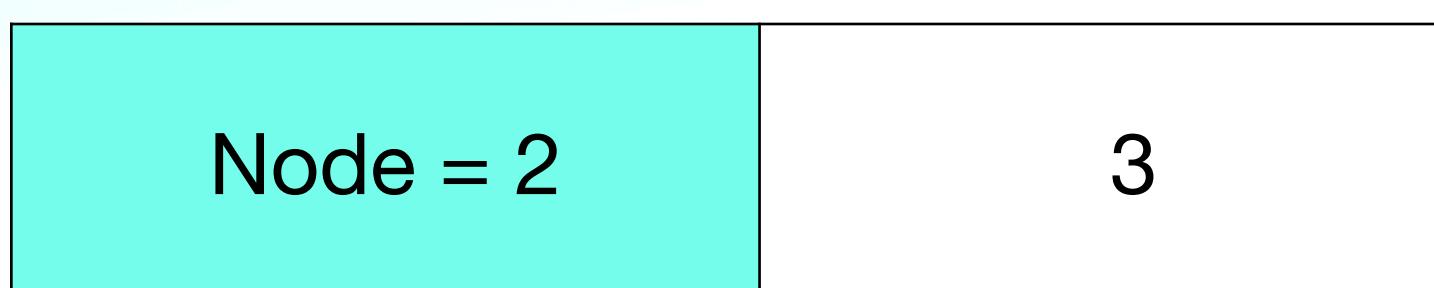
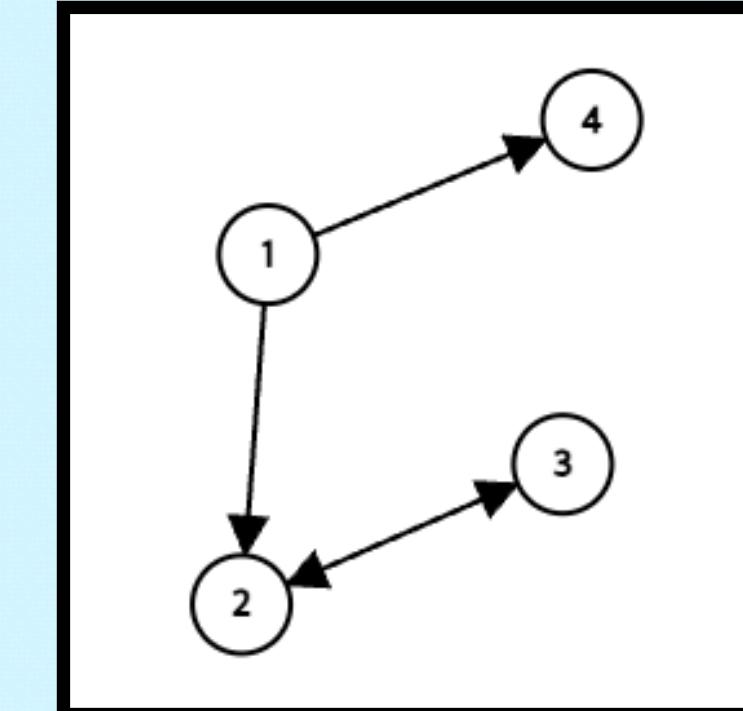
- 인접 리스트 : 각 노드마다 리스트를 만들어서 표현

장점 : 메모리를 아끼기 좋다

1에서 갈 수 있는곳이
몇개인지 쉽게 확인 가능

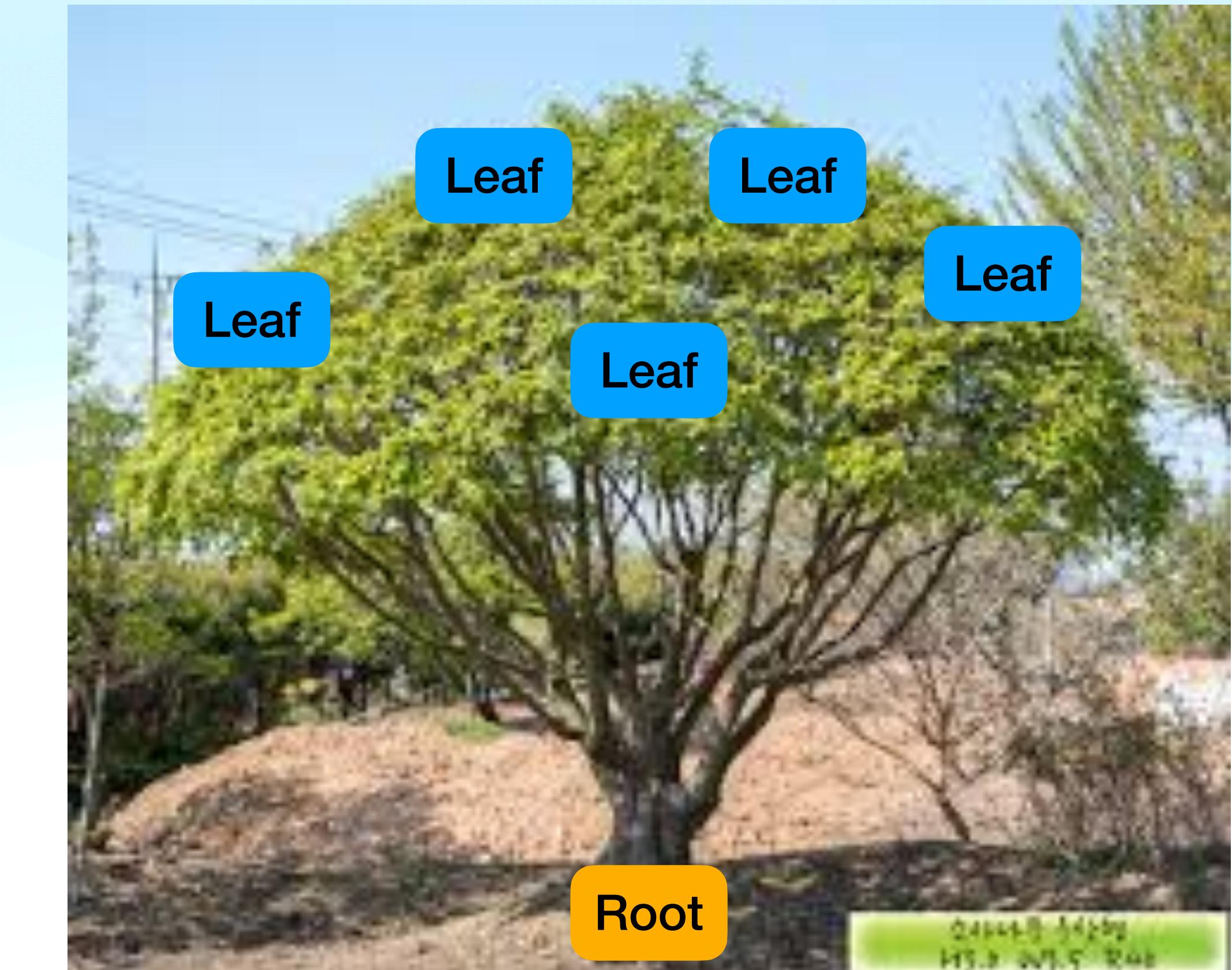
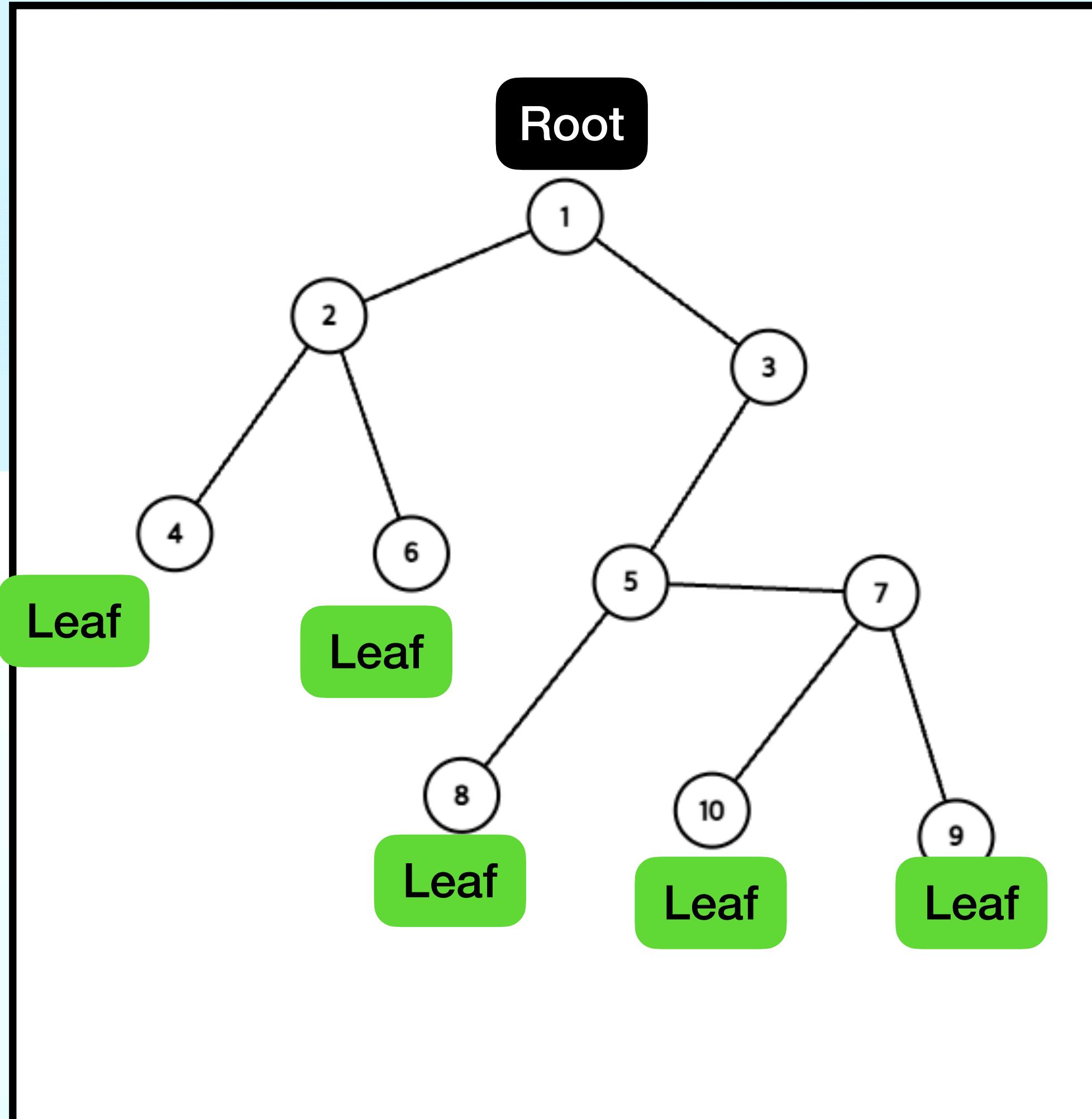
단점 : 구조변경이 힘들 수 있다.

```
vector<int> adj [NODE_MAX]; // 미리 만들어두기
vector adj = vector(NODE_MAX, vector<int>()); // 동적할당
// adj[i]에는 i에서 갈 수 있는 곳의 리스트가 저장되어있음
```



트리(Tree)

트리도 자료구조다. (사이클이 없이 모든 정점이 연결되어 있는 그래프)

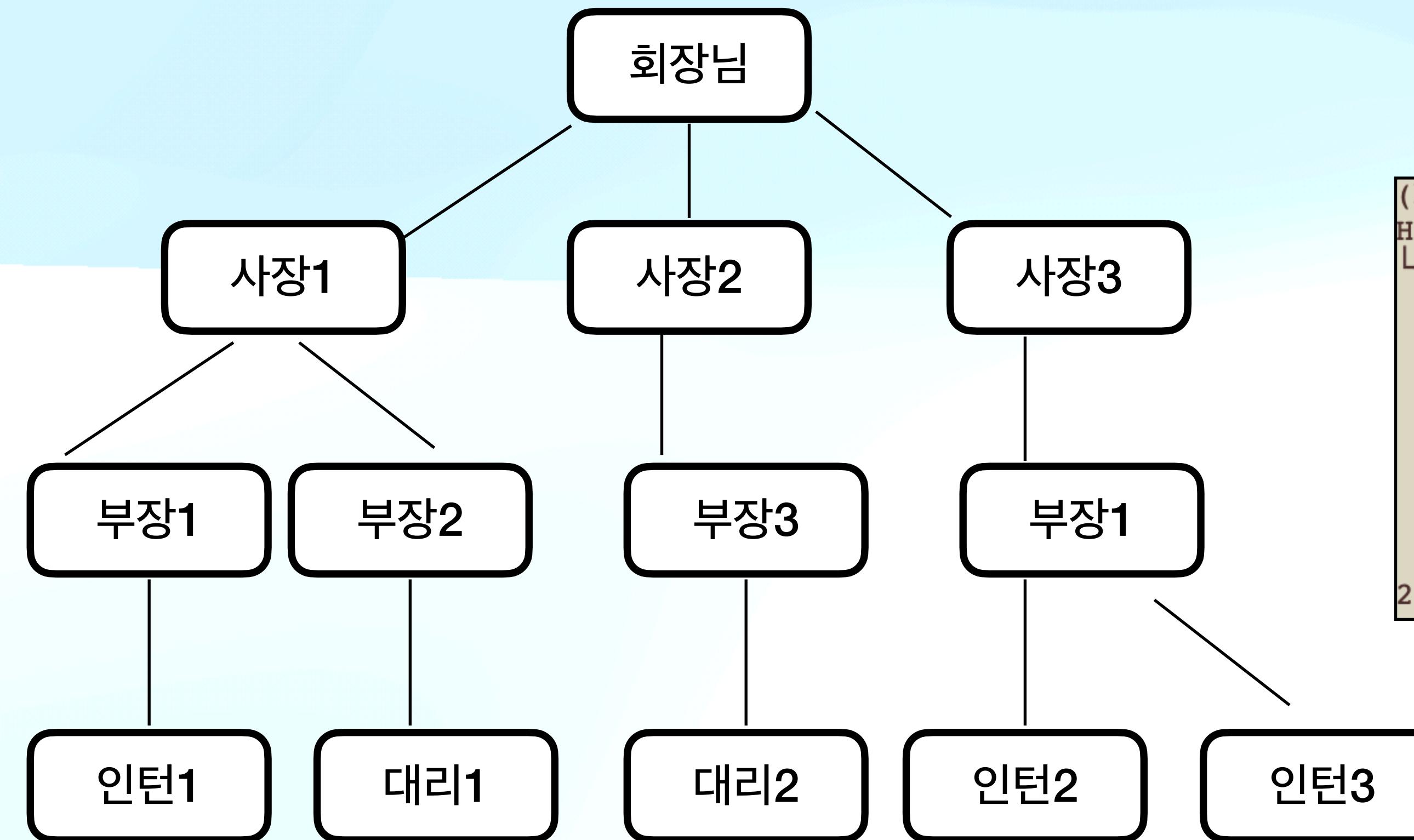


출처 : <https://www.google.com/imgres?q=%EB%82%98%EB%AC%B4&imgurl=https%3A%2F%2Fwww.treedb.co.kr%2Frbd%2Fhdd%2F2021%2F02%2F26%2F21c05c69f73d795c5a39d7e788596e40123203.jpg&imgrefurl=https%3A%2F%2Fwww.treedb.co.kr%2Frbd%2F%3Fr%3Dhome%26c%3D1%2F7%26uid>

트리(Tree)

트리도 자료구조다. (저번에 했던 set, pq 등과 연관)

트리는 계층관계를 나타낼 때 효과적이다



[회사의 계층구조]

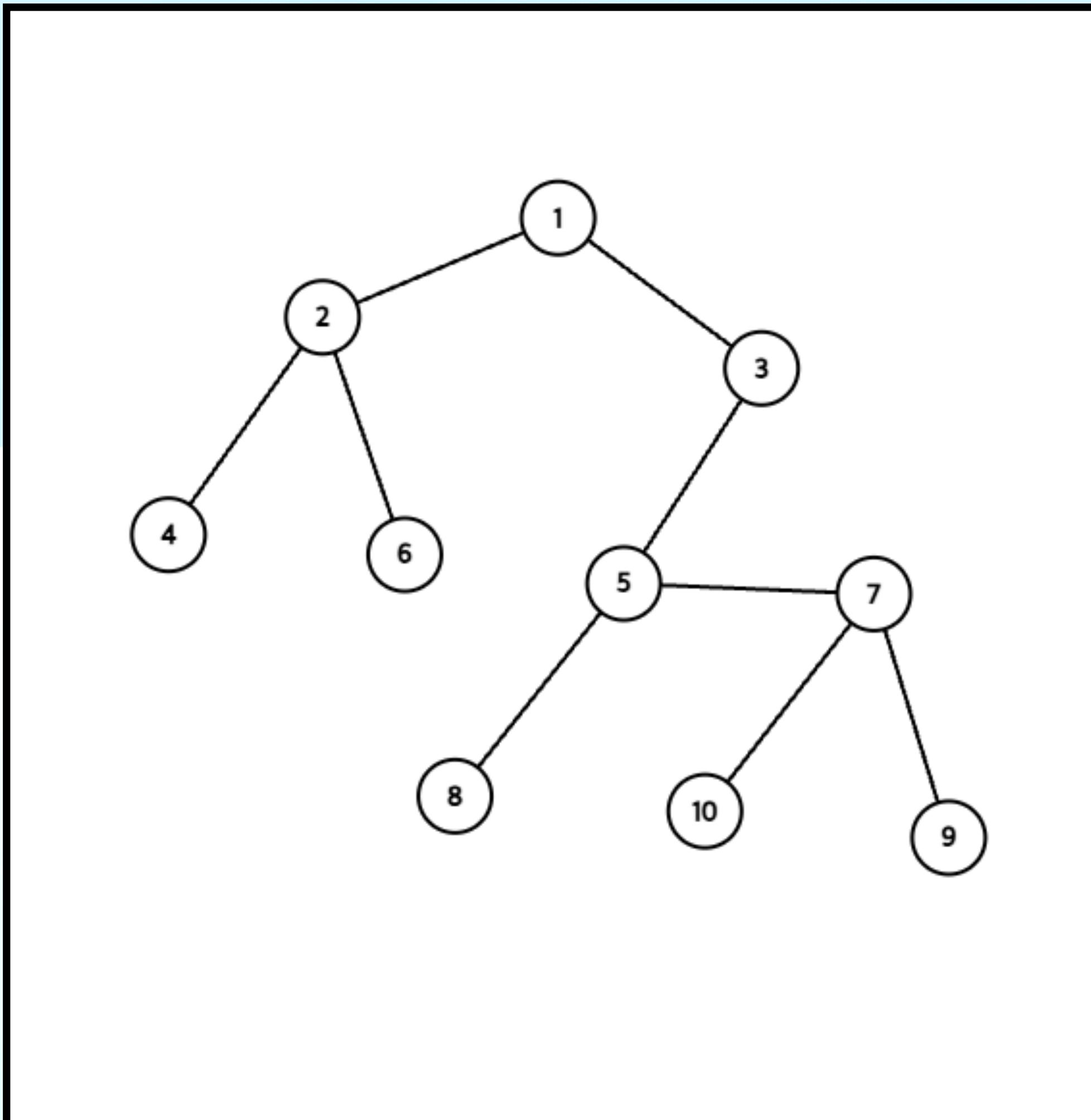
```
(base) dosawasseungjun@hanseungjun-ui-MacBookPro ~ % tree Hiarc_beginner
Hiarc_beginner
└── study_beginner
    ├── 1_강의 개관.pdf
    ├── 2_Bruteforce.pdf
    ├── 3_datastructure.pdf
    ├── 4_binarysearch.pdf
    ├── 5_dynamic programming.pdf
    ├── 6_Number Theory.pdf
    └── README.md

2 directories, 7 files
```

[tree 명령어]

트리(Tree)

트리도 자료구조다. (저번에 했던 set, pq 등과 연관)



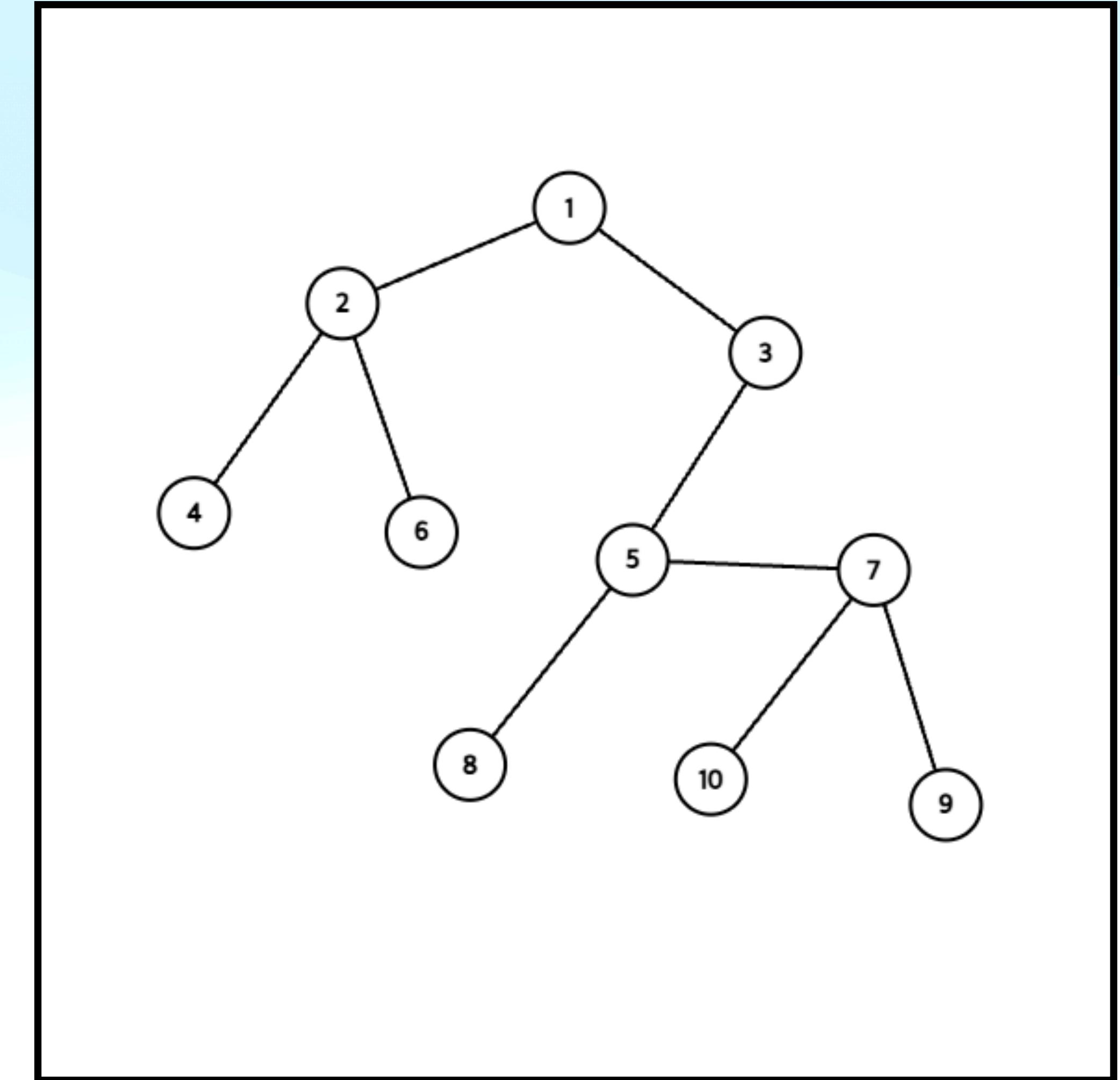
용어 :

1. Node (root node - 최상단 노드)
2. Parent, child
 1. 2는 4의 parent이다.
 2. 7의 children은 9와 10이다.
3. Edge
 1. 각 노드들은 edge로 연결되어 있다.
4. sibling(형제), ancestor(조상), descendant(후손)
5. Depth - root노드에서 최단거리
6. Height - tree에서 가장 깊은 depth

트리(Tree)

트리의 특성 1 - node의 개수가 n 이라면 간선의 개수는 $n-1$ 이다.

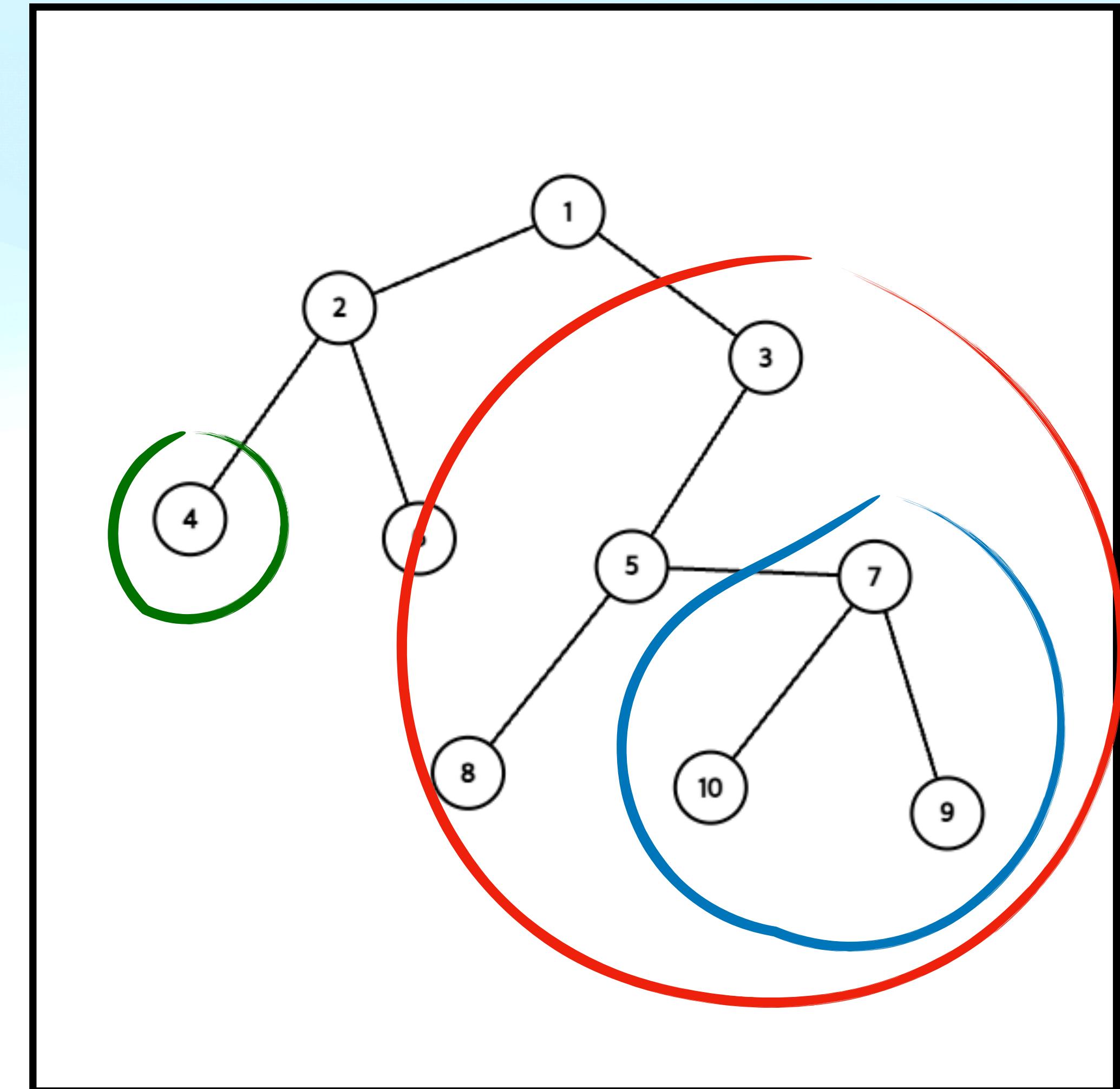
- edge의 개수가 $n-1$ 보다 작다면?
 - 연결되지 않은 node가 무조건 생긴다.
- edge의 개수가 $n-1$ 보다 크다면?
 - 사이클이 무조건 생긴다.



트리(Tree)

트리의 특성 2 - 어떤 노드 t 와 그의 자손들로 구성된 트리도 tree가 된다.

- 그림에서 3을 root로 하는 서브트리는?
 - 빨간색 원
- 그림에서 7을 root로 하는 서브트리는?
 - 파란색 원
- 그림에서 4를 root로 하는 서브트리는?
 - 초록색 원
- 이는 트리의 재귀적 속성을 보여준다.



트리(Tree)의 표현

가장 간단하게 트리를 표현하기

트리도 그래프니까 인접행렬이나 인접 리스트로 표현 가능

구조체로 노드 만들기

```
struct TreeNode{  
    string label;  
    TreeNode * parent;  
    vector<TreeNode * > children;  
};
```

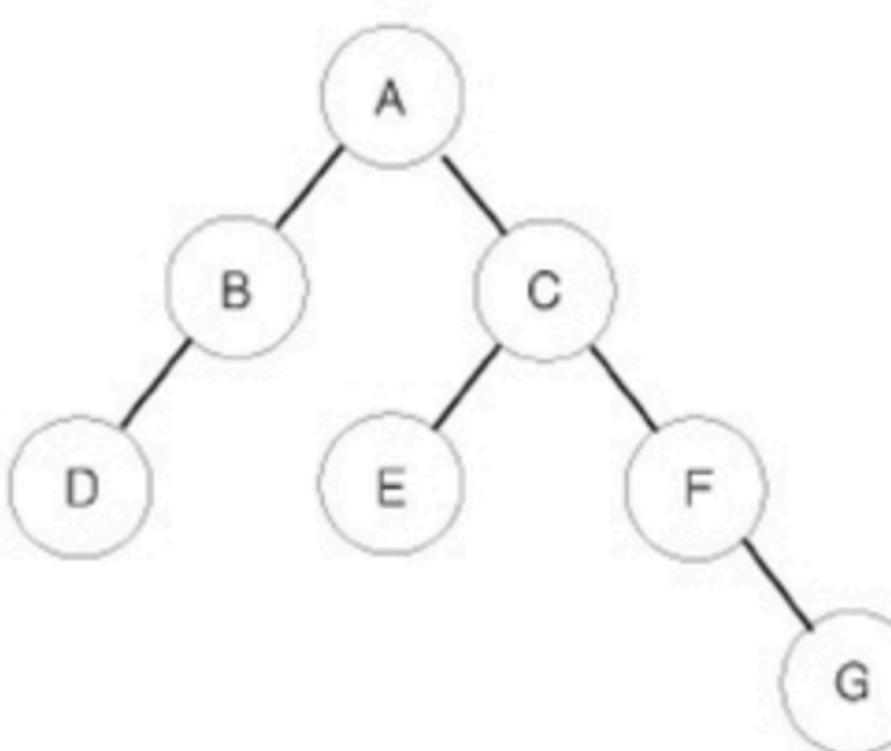
저런 노드를 만들어 두는 연습을 해두면
나중에 trie라는 자료구조를 배울때 도움이 됩니다.

트리(Tree)의 순회

1991 - 트리 순회(Preorder, inorder, postorder)

문제

이진 트리를 입력받아 전위 순회(preorder traversal), 중위 순회(inorder traversal), 후위 순회(postorder traversal)한 결과를 출력하는 프로그램을 작성하시오.



예를 들어 위와 같은 이진 트리가 입력되면,

- 전위 순회한 결과 : ABDCEFG // (루트) (왼쪽 자식) (오른쪽 자식)
- 중위 순회한 결과 : DBAECFG // (왼쪽 자식) (루트) (오른쪽 자식)
- 후위 순회한 결과 : DBEGFCA // (왼쪽 자식) (오른쪽 자식) (루트)

가 된다.

트리(Tree)의 순회

Preorder, inorder, postorder

트리의 재귀적인 특징을 잘 이용한 문제

```
43 int main(){
44     fast_io
45     int n; cin >> n;
46     for(int i=0;i<n;i++){
47         char p, c1, c2;
48         cin >> p >> c1 >> c2;
49         tree[p - 'A'].label = p;
50         if(c1 != '.') tree[p-'A'].left = &tree[c1-'A'];
51         if(c2 != '.') tree[p-'A'].right = &tree[c2-'A'];
52     }
53
54     preorder(&tree[0]); cout << '\n';
55     inorder(&tree[0]); cout << '\n';
56     postorder(&tree[0]);
57 }
```

```
12 struct TreeNode{
13     char label;
14     TreeNode * left,* right;
15 };
16
17 vector<TreeNode> tree(26);
18
19 void preorder(TreeNode* root){
20     if(root == NULL) return;
21
22     cout << root -> label;
23     preorder(root -> left);
24     preorder(root -> right);
25 }
26
27 void inorder(TreeNode* root){
28     if(root == NULL) return;
29
30     inorder(root -> left);
31     cout << root -> label;
32     inorder(root -> right);
33 }
34
35 void postorder(TreeNode* root){
36     if(root == NULL) return;
37
38     postorder(root -> left);
39     postorder(root -> right);
40     cout << root -> label;
41 }
```

Depth-First Search(DFS)

깊이 우선 탐색

- 미로를 탐색할 때 내가 갈림길을 만나면 표시를 하고 한 길로 쭉 가다가 막히면 다시 갈림길로 돌아온다.
- 깊이 들어갔다가 나온다.

```
12 const int V_MAX = 1e5+1;
13 bool vst[V_MAX];
14 vector<int> adj[V_MAX];
15
16 void dfs(int here){
17     vst[here] = true;
18     for(int nxt : adj[here]){
19         if(vst[nxt]) continue;
20         dfs(nxt);
21     }
22 }
```

Breadth-First Search(BFS)

너비 우선 탐색

- 바이러스가 확산되는 모습을 상상해보자
- 큐를 이용하여 구현한다.

```
12 const int V_MAX = 1e5+1;
13 bool vst[V_MAX];
14 vector<int> adj[V_MAX];
15
16 vector<int> bfs(int here){
17     queue<int> q;
18     vst[here] = true;
19     q.push(here);
20     while(!q.empty()){
21         int now = q.front();
22         q.pop();
23         for(int nxt : adj[now]){
24             if(vst[nxt]) continue;
25             q.push(nxt);
26             vst[nxt] = true;
27         }
28     }
29 }
```

DFS vs BFS

공부 시작

국어

수학

영어

국어 모의고사

수학 모의고사

영어 모의고사

국어 오답정리

수학 오답정리

영어 오답정리

깊이 우선탐색 :

국어 -> 국어모의고사 -> 국어오답정리 ->
수학 -> 수학모의고사 -> 수학오답정리 ->
영어 -> 영어모의고사 -> 영어오답정리

너비 우선탐색 :

국어 -> 수학 -> 영어
국어모의고사 -> 수학모의고사 -> 영어모의고사 ->
국어오답정리 -> 수학오답정리 -> 영어오답정리

그래프 순회

- DFS나 BFS는 그래프를 순회할때 씁니다.
- 그럼 그래프 순회는 어디가 쓸지 예시로 생각해봅시다.
 - 1. 홍대입구역에서 부산역까지 지하철을 타고 갈 수 있을까?
 - 홍대입구역에서 부터 그래프 순회를 해서 부산역에 visit했는지 확인한다.
 - 2. 코로나 바이러스가 퍼졌다. 우리가 만난 사람의 연결관계가 주어졌을 때 누구누구가 걸렸을 위험이 있을지 확인하는 방법
 - 첫번째 걸린 사람부터 그래프 순회를 해서 바이러스에 걸렸을 가능성이 있는 사람을 골라냄

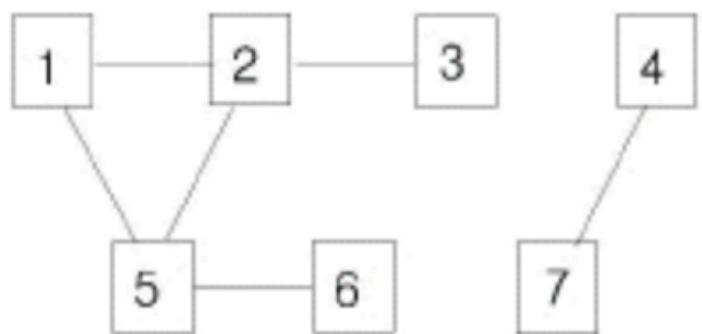
그래프 순회

2606 - 바이러스

문제

신종 바이러스인 웜 바이러스는 네트워크를 통해 전파된다. 한 컴퓨터가 웜 바이러스에 걸리면 그 컴퓨터와 네트워크 상에서 연결되어 있는 모든 컴퓨터는 웜 바이러스에 걸리게 된다.

예를 들어 7대의 컴퓨터가 <그림 1>과 같이 네트워크 상에서 연결되어 있다고 하자. 1번 컴퓨터가 웜 바이러스에 걸리면 웜 바이러스는 2번과 5번 컴퓨터를 거쳐 3번과 6번 컴퓨터까지 전파되어 2, 3, 5, 6 네 대의 컴퓨터는 웜 바이러스에 걸리게 된다. 하지만 4번과 7번 컴퓨터는 1번 컴퓨터와 네트워크상에서 연결되어 있지 않기 때문에 영향을 받지 않는다.



< 그림 1 >

어느 날 1번 컴퓨터가 웜 바이러스에 걸렸다. 컴퓨터의 수와 네트워크 상에서 서로 연결되어 있는 정보가 주어질 때, 1번 컴퓨터를 통해 웜 바이러스에 걸리게 되는 컴퓨터의 수를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에는 컴퓨터의 수가 주어진다. 컴퓨터의 수는 100 이하인 양의 정수이고 각 컴퓨터에는 1번 부터 차례대로 번호가 매겨진다. 둘째 줄에는 네트워크 상에서 직접 연결되어 있는 컴퓨터 쌍의 수가 주어진다. 이어서 그 수만큼 한 줄에 한 쌍씩 네트워크 상에서 직접 연결되어 있는 컴퓨터의 번호 쌍이 주어진다.

그래프 순회

1. 일단 1번 컴퓨터를 기준으로 순회를 한다.
2. 그럼 거기서 감염되는 컴퓨터를 체크한다.
3. 1번으로부터 감염된 컴퓨터를 구하는 것이므로 1을 빼서 정답을 구해준다.

```
12 const int V_MAX = 1e2 + 1;
13 bool vst[V_MAX];
14 vector<int> adj[V_MAX];
15 int V, E;
16
17 void dfs(int here){
18     vst[here] = true;
19     for (int nxt : adj[here]){
20         if (vst[nxt])
21             continue;
22         dfs(nxt);
23     }
24 }
25
26 int main(){
27     fast_io
28     cin >> V >> E;
29     for(int i=0;i<E;i++){
30         int u, v; cin >> u >> v;
31         adj[u].push_back(v);
32         adj[v].push_back(u);
33     }
34     dfs(1);
35     cout << count(vst, vst + V_MAX, 1) - 1;
36 }
```

그래프 순회

1707 이분 그래프

아까 배웠던 이분 그래프가
뭐였는지 생각해봅시다.

이분 그래프

성공



4 골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	98080	26840	16497	24.417%

문제

그래프의 정점의 집합을 둘로 분할하여, 각 집합에 속한 정점끼리는 서로 인접하지 않도록 분할할 수 있을 때, 그러한 그래프를 특별히 이분 그래프 (Bipartite Graph) 라 부른다.

그래프가 입력으로 주어졌을 때, 이 그래프가 이분 그래프인지 아닌지 판별하는 프로그램을 작성하시오.

입력

입력은 여러 개의 테스트 케이스로 구성되어 있는데, 첫째 줄에 테스트 케이스의 개수 K 가 주어진다. 각 테스트 케이스의 첫째 줄에는 그래프의 정점의 개수 V 와 간선의 개수 E 가 빈 칸을 사이에 두고 순서대로 주어진다. 각 정점에는 1부터 V 까지 차례로 번호가 붙어 있다. 이어서 둘째 줄부터 E 개의 줄에 걸쳐 간선에 대한 정보가 주어지는데, 각 줄에 인접한 두 정점의 번호 u, v ($u \neq v$)가 빈 칸을 사이에 두고 주어진다.

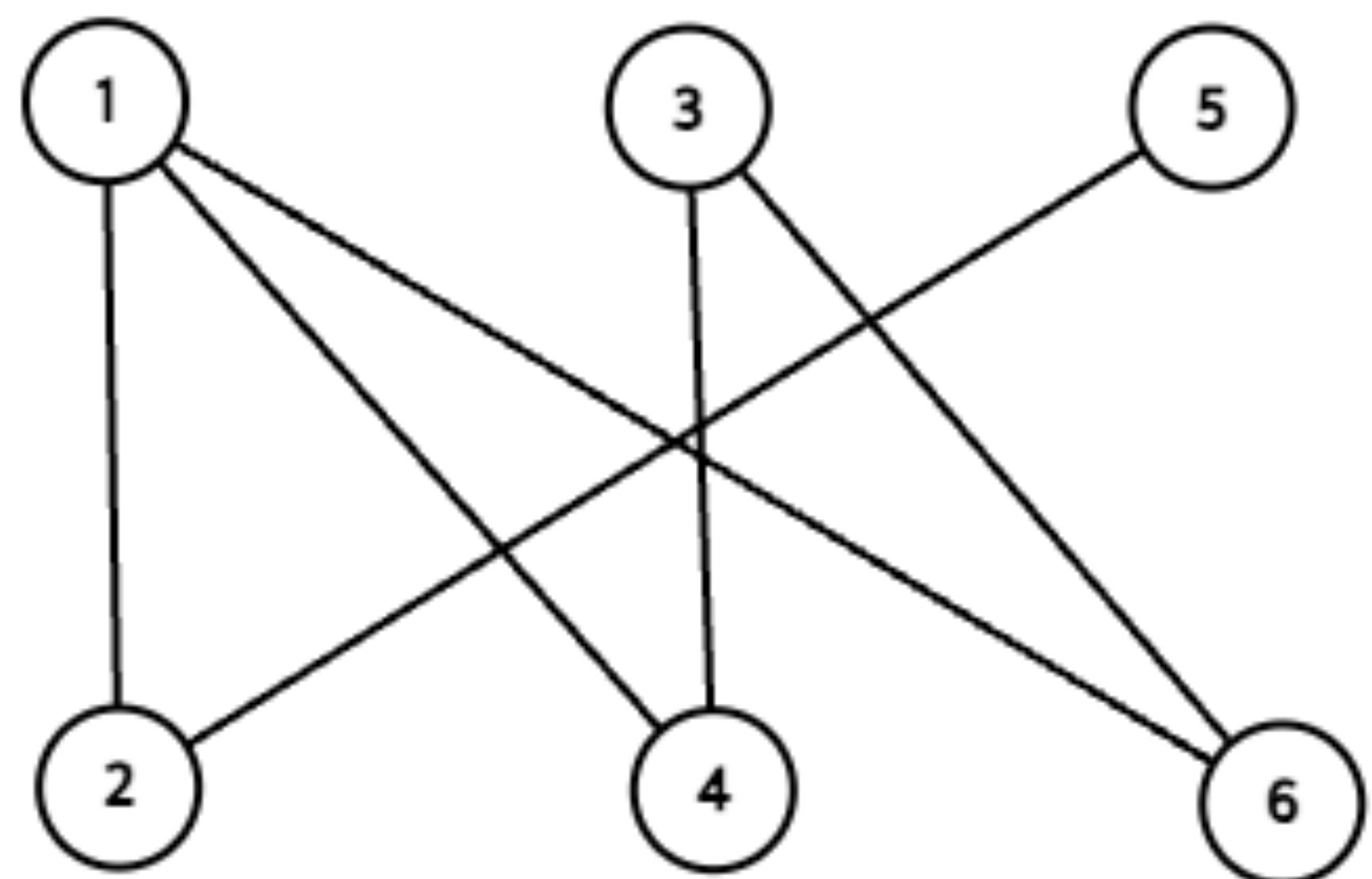
출력

K 개의 줄에 걸쳐 입력으로 주어진 그래프가 이분 그래프이면 YES, 아니면 NO를 순서대로 출력한다.

그래프 순회

1707 이분 그래프

<이분 그래프라는 것을 어떻게 확인할 수 있을까요??>



1, 3, 5에서 다음 칸으로 가면 무조건 2, 4, 6중 하나다.

하트시그널로 치면...

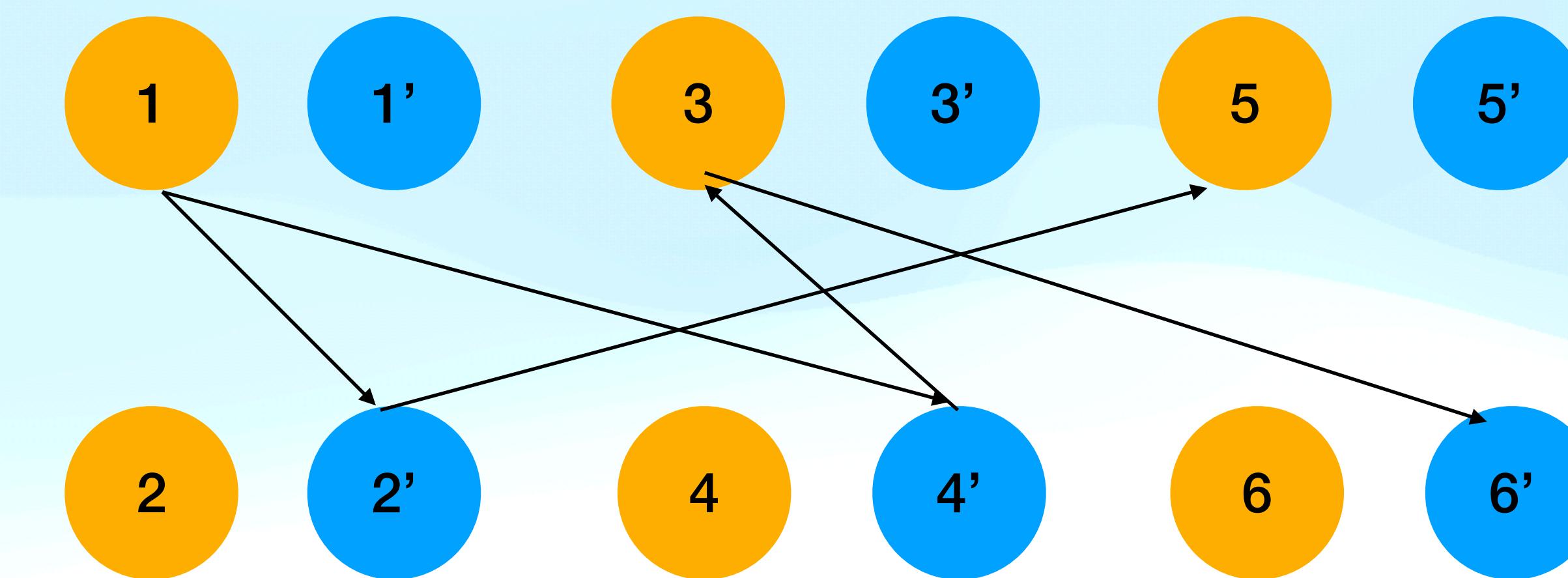
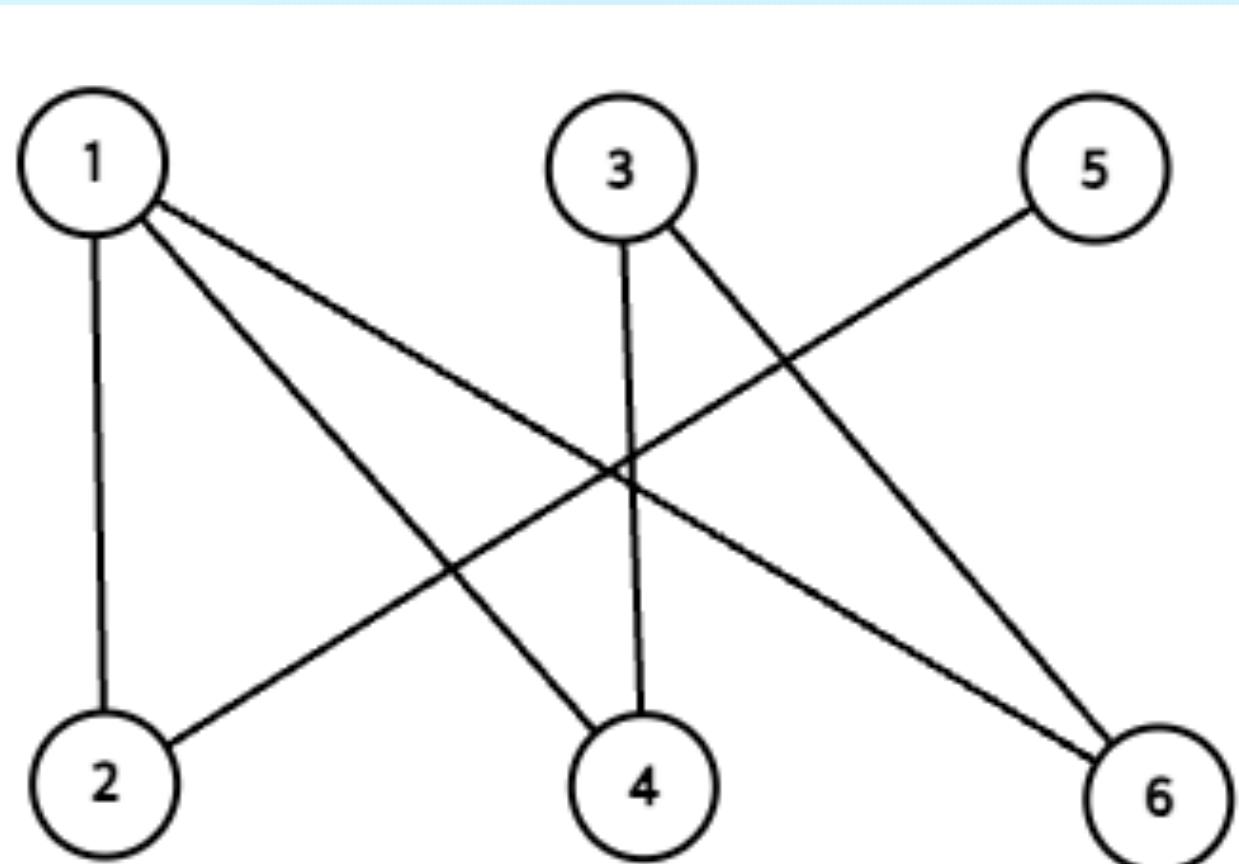
남자는 여자를 고를 수 밖에 없고

여자는 남자를 고를 수 밖에 없는 상황

즉, 짝수번째 도착하는 곳은 항상 여자 그룹,
홀수번째 도착하는 곳은 항상 남자그룹으로 일반화 가능

그래프 순회

1707 이분 그래프



순회를 할 때

노란색에서 출발하면 파란색으로,

파란색에서 출발하면 노란색으로 간다

모든 순회를 한 후에, 각 숫자를 기준으로 파란색 노란색 둘다 접근이 되어있으면
이분그래프가 아니다.

그래프 순회

1707 이분 그래프

```
12 const int V_MAX = 2e4+1;
13 vector<vector<int>> adj;
14 int vst[V_MAX][2];
15
16 void dfs(int here, int cnt){
17     vst[here][cnt] = true;
18     for(int nxt : adj[here]){
19         if(vst[nxt][cnt ^ 1]) continue;
20         dfs(nxt, cnt ^ 1);
21     }
22 }
23
24 void dfsAll(int v){
25     for(int i=1;i<=v;i++) if(!vst[i][0] && !vst[i][1]) dfs(i, 0);
26 }
```

이 문제가 조금 짜증나는게

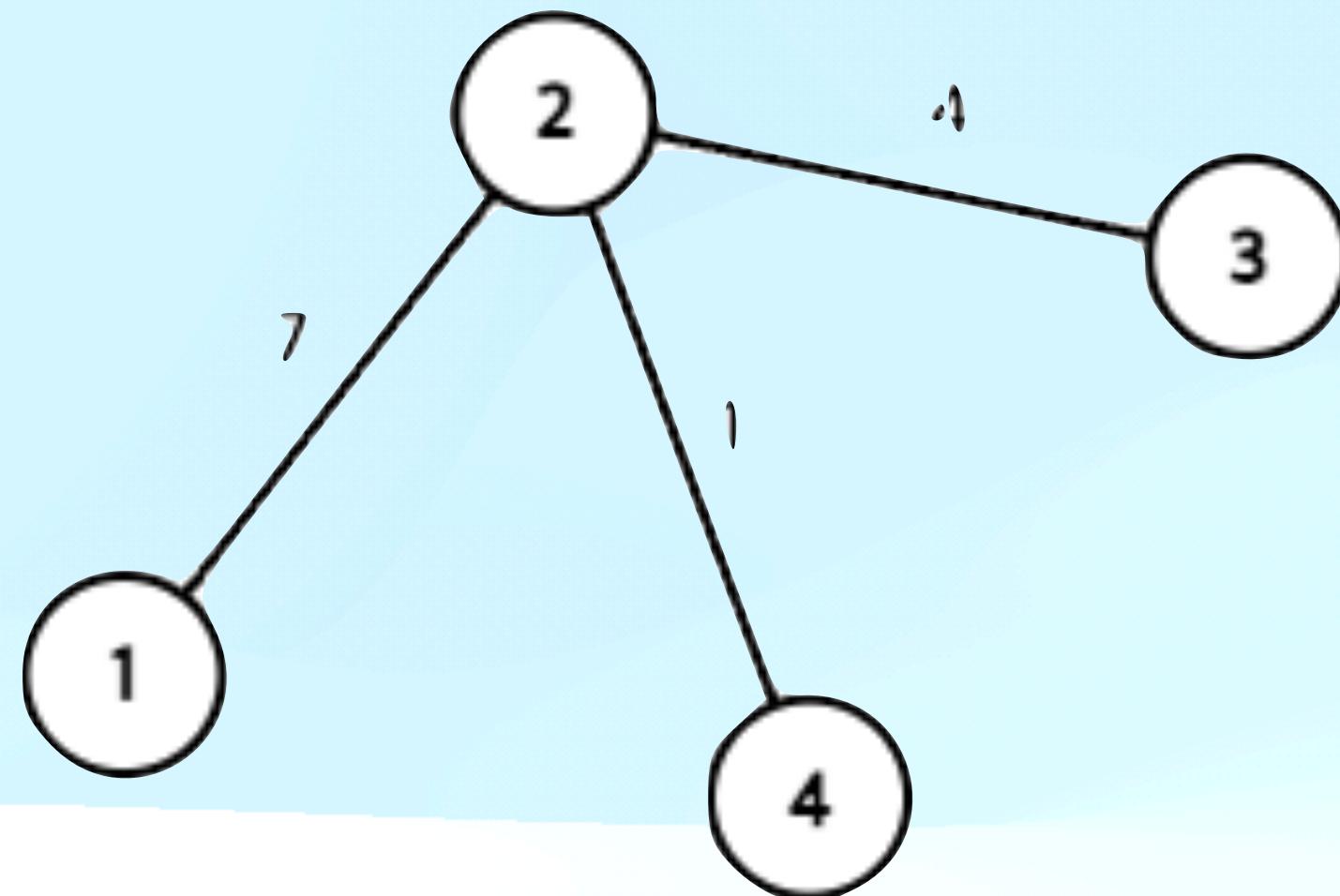
여러개의 그래프가 입력으로 들어올 수 있습니다.

즉 끊어진 여러개의 그래프의 경우도 고려해야 합니다.

또한 테스트 케이스마다 필요한 변수를 초기화하는 것을 잊지맙시다.

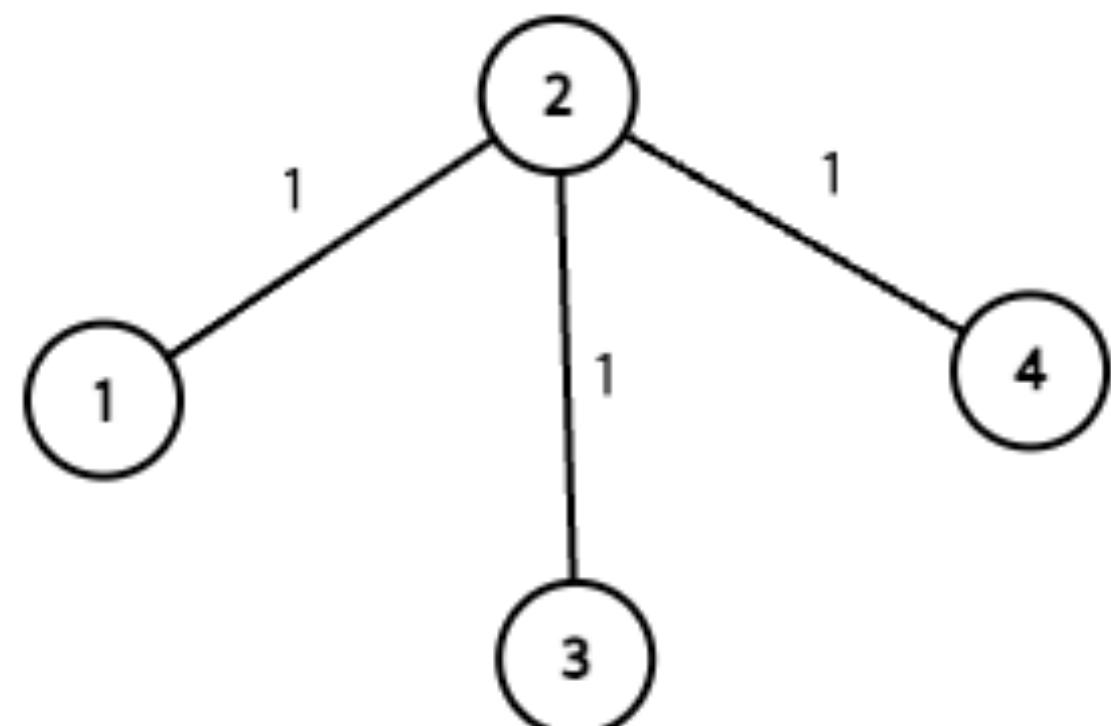
```
28 void solv(){
29     int v, e; cin >> v >> e;
30     adj = vector(v+1, vector<int>());
31     memset(vst, 0, sizeof(vst));
32     for(int i=0;i<e;i++){
33         int u, v; cin >> u >> v;
34         adj[u].push_back(v);
35         adj[v].push_back(u);
36     }
37     dfsAll(v);
38     bool isBipar = true;
39     for(int i=1;i<=v;i++){
40         if(vst[i][0] + vst[i][1] == 1) continue;
41         isBipar = false;
42     }
43     if(isBipar) cout << "YES\n";
44     else cout << "NO\n";
45 }
46
47 int main(){
48     fast_io
49     int tt; cin >> tt;
50     while(tt--) solv();
51 }
```

0-1 BFS



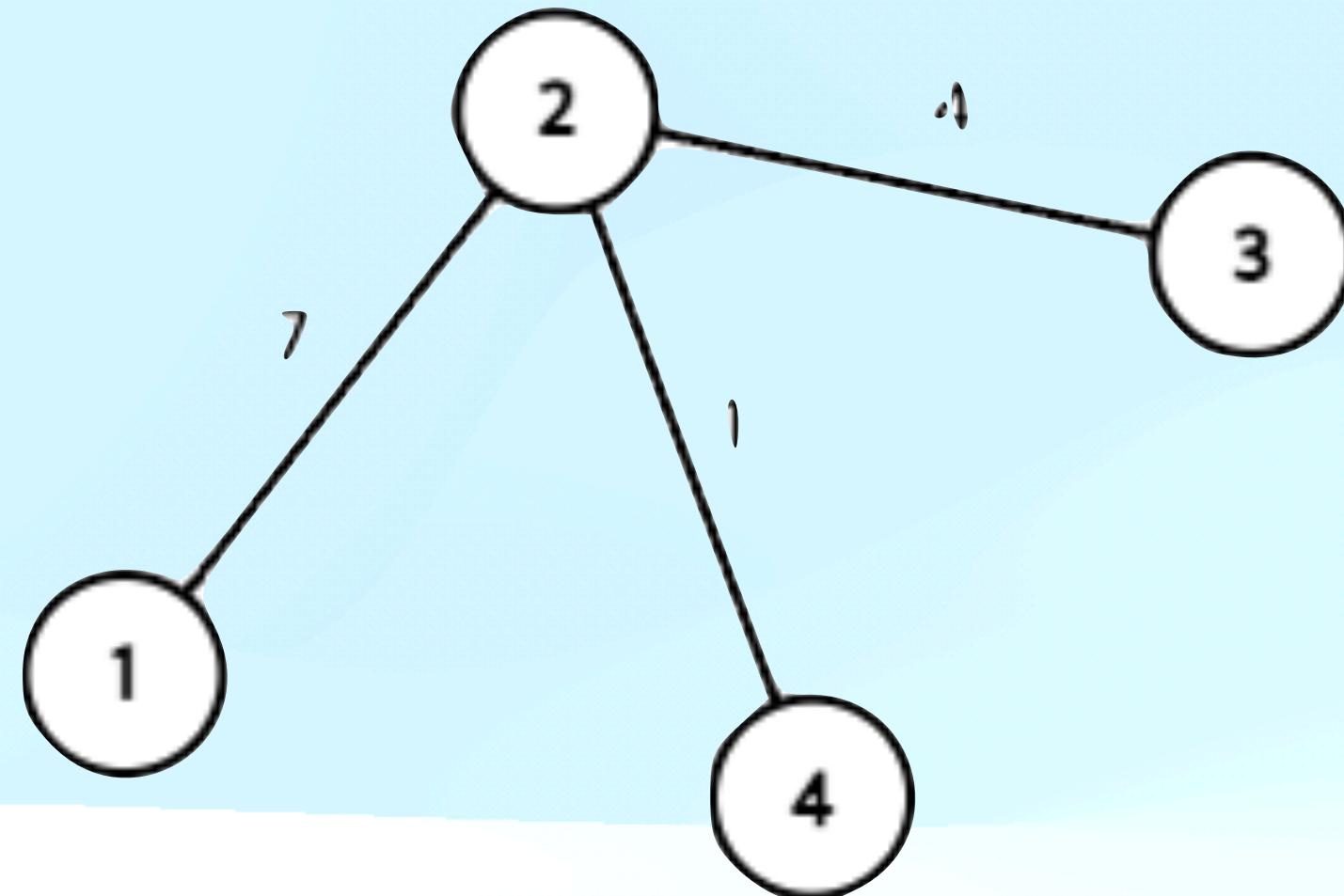
이렇게 가중치가 있는 경우
한 정점에서 다른 정점까지 거리가 궁금합니다.

이 경우 단순히 BFS로는 구하기 힘들 것입니다.
왜냐면 더 돌아가는 것처럼 보여도 더 가까운 경우가 생기니까요



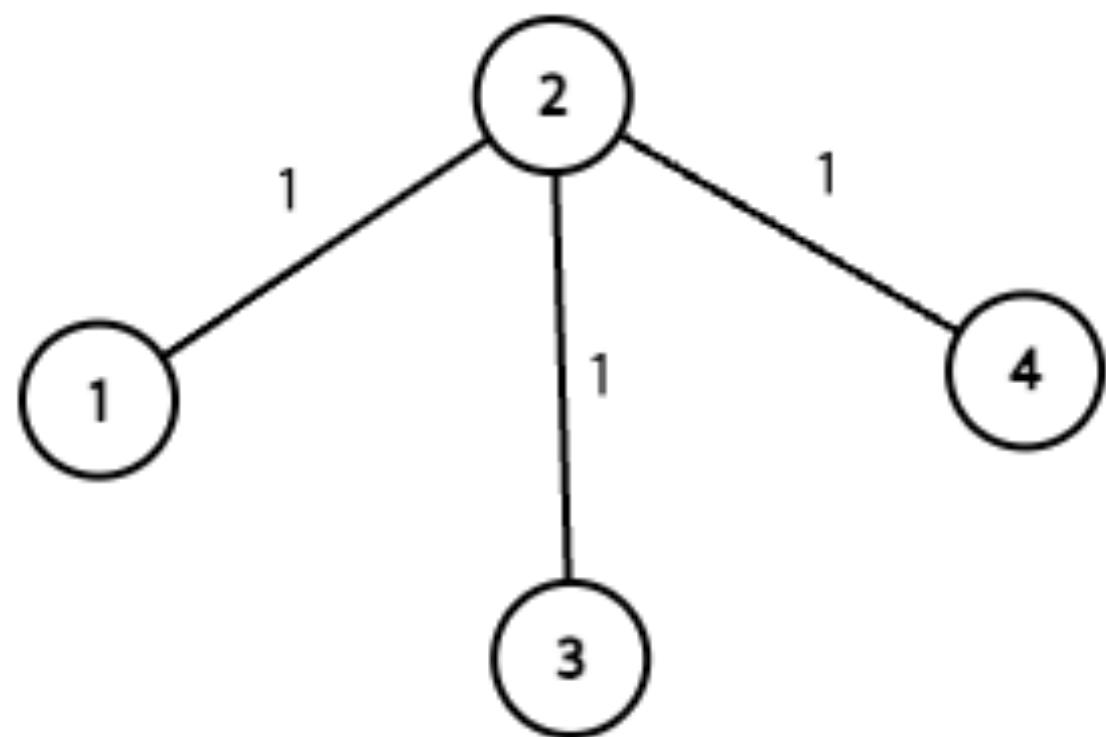
하지만 모든 가중치가 1이라면
최단 거리를 구할 수 있습니다.

0-1 BFS



이렇게 가중치가 있는 경우
한 정점에서 다른 정점까지 거리가 궁금합니다.

이 경우 단순히 BFS로는 구하기 힘들 것입니다.
왜냐면 더 돌아가는 것처럼 보여도 더 가까운 경우가 생기니까요



하지만 모든 가중치가 0 또는 1이라면
한 정점으로부터 최단 거리를 bfs를 응용해 구할 수 있습니다.

특히, 큐 대신 덱을 사용하는데요

왜인지 스스로 생각해봅시다.

0-1 BFS

13549 - 숨바꼭질 3 (이런 유형의 문제는 정말 많습니다!!)

숨바꼭질 3

성공



5 골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	106999	27544	18344	24.102%

문제

수빈이는 동생과 숨바꼭질을 하고 있다. 수빈이는 현재 점 $N(0 \leq N \leq 100,000)$ 에 있고, 동생은 점 $K(0 \leq K \leq 100,000)$ 에 있다. 수빈이는 걷거나 순간이동을 할 수 있다. 만약, 수빈이의 위치가 X 일 때 걷는다면 1초 후에 $X-1$ 또는 $X+1$ 로 이동하게 된다. 순간이동을 하는 경우에는 0초 후에 $2*X$ 의 위치로 이동하게 된다.

수빈이와 동생의 위치가 주어졌을 때, 수빈이가 동생을 찾을 수 있는 가장 빠른 시간이 몇 초 후인지 구하는 프로그램을 작성하시오.

입력

첫 번째 줄에 수빈이가 있는 위치 N 과 동생이 있는 위치 K 가 주어진다. N 과 K 는 정수이다.

0-1 BFS

13549 - 숨바꼭질 3 (이런 유형의 문제는 정말 많습니다!!)

숨바꼭질 3

성공

5 골드 V

시간 제한

2 초

메모리 제한

512 MB

제출

106999

정답

27544

문제

수빈이는 동생과 숨바꼭질을 하고 있다. 수빈이는 현재 점 $N(0 \leq N \leq 100,000)$ 에 있고, 동생은 점 $K(0 \leq K \leq 100,000)$ 에 있다. 수빈이는 걷거나 순간이동을 할 수 있다. 만약, 수빈이의 위치가 X 일 때 걷는다면 1초 후에 $X-1$ 또는 $X+1$ 로 이동하게 된다. 순간이동을 하는 경우에는 0초 후에 $2*X$ 의 위치로 이동하게 된다.

수빈이와 동생의 위치가 주어졌을 때, 수빈이가 동생을 찾을 수 있는 가장 빠른 시간이 몇 초 후인지 구하는 프로그램을 작성하시오.

입력

첫 번째 줄에 수빈이가 있는 위치 N 과 동생이 있는 위치 K 가 주어진다. N 과 K 는 정수이다.

이 문제가 0-1 bfs인 이유

모든 행동은 0초 혹은 1초 걸린다.★

즉, 가중치가 0이거나 1이다.

vst 배열 대신

dist 배열을 써서

맞힌 사람
18344 24.102%
가중치가 0이면 $dist[nxt] = dist[now]$

가중치가 1이면 $dist[nxt] = dist[now] + 1$
으로 해결해보자.

0-1 BFS

13549 - 숨바꼭질 3

1. 단순히 방문 처리로는 안된다.

1. 더 빠른 방법이 있으면 간신해준다.

2. $4 \rightarrow 3 \rightarrow 6$ 이
 $4 \rightarrow 5 \rightarrow 6$ 보다 이득이다.

3. 이를 방문처리로 풀려면
if문의 순서를 $-1 \rightarrow +1 \rightarrow 2^*$
순으로 트어야한다.

2. $2 * now$ 부분의 범위체크

1. 조건문 검사는 순서대로 한다.

```
12 const int V_MAX = 2e5+1;
13 const int INF = 1e9+1;
14
15 vector<int> bfs(int here){
16     deque<int> dq;
17     vector<int> dist(V_MAX, INF);
18     dq.push_back(here);
19     dist[here] = 0;
20     while(!dq.empty()){
21         int now = dq.front();
22         dq.pop_front();
23         if(now + 1 < V_MAX && dist[now + 1] > dist[now] + 1){
24             dq.push_back(now + 1);
25             dist[now + 1] = dist[now] + 1;
26         }
27         if (now - 1 >= 0 && dist[now - 1] > dist[now] + 1){
28             dq.push_back(now - 1);
29             dist[now - 1] = dist[now] + 1;
30         }
31         if(2 * now < V_MAX && dist[2 * now] > dist[now]){
32             dq.push_front(now * 2);
33             dist[now * 2] = dist[now];
34         }
35     }
36     return dist;
37 }
38
39
40 int main(){
41     fast_io
42     int N, K; cin >> N >> K;
43     vector<int> D = bfs(N);
44     cout << D[K];
45 }
```

마무리

Check list

- 그래프란 무엇인가?
 - Directed graph, undirected graph, DAG, tree 등으로 구분할 수 있는가?
 - 그래프를 인접행렬 또는 인접리스트로 나타낼 수 있는가?
- 트리는 무엇인가?
 - 트리를 구현할 수 있는가?
 - 코드로 트리의 순회를 할 수 있는가?
- DFS는 무엇인가?
- BFS는 무엇인가?
 - Bfs 를 응용하여 0-1 bfs 문제를 해결할 수 있는가?

마무리

추천문제

- 1991 트리 순회 - 트리 순회를 공부하는 정말 좋은 문제
- 2606 바이러스 - 그래프 순회는 왜 필요한가?
- 1707 이분 그래프
 - 이분 그래프가 무엇인지 아는가?
 - 끊어진 그래프가 있을 수도 있다.
- 13549 숨바꼭질 3
 - 0-1 bfs에 대해 배워보자
- 1068 트리 - 트리에서의 dfs
- 1260 DFS와 BFS - 개념문제

마무리

추천문제

- 1325 효율적인 해킹
- 16985 Maaaaaaaaaze - 매우 귀찮지만 코테 대비용으로도 괜찮은 문제
- 1261 알고스팟 - 0-1 bfs 가능
- 1936 소수 경로
 - 소수판정과 그래프 순회의 결합
- 2263 트리의 순회
 - 조금 어려운 문제 - 배열을 slice하는 스킬이 필요함
- 18251 내 생각에 A번인 단순 dfs 문제가 이 대회에서 E번이 되어버린 건에 관하여(Easy) - 좀 많이 어려울 수 있음 ㅎ ㅎ
 - inorder(중위 순회)를 이용하면 트리의 있는 모든 원소를 일자로 펼 수 있다.
 - 이 후 높이(depth)를 기준으로 범위를 정해서 높이가 범위 사이인 것만을 이용해 최댓값을 구한다.
 - 최댓값을 구할 땐, 양수인 경우 계속 최댓값을 갱신해 나가다가 중간 결과가 음수가 되면 0으로 둑 끊어버리면 될 것이다.