BFS / DFS

BFS, DFS 란?

그래프의 탐색 기법

목적 : 임의의 한 정점에서 시작하여 모든 정점을 방문

BFS : Breadth First Search, 너비 우선 탐색

DFS : Depth First Search, 깊이 우선 탐색

BFS

간선의 모든 가중치가 같을 때, 최단 거리를 구하는 알고리즘

시작 정점을 기준으로 가까운 정점을 먼저 방문한다. (방문 순서 -> 최단 거리에 영향)

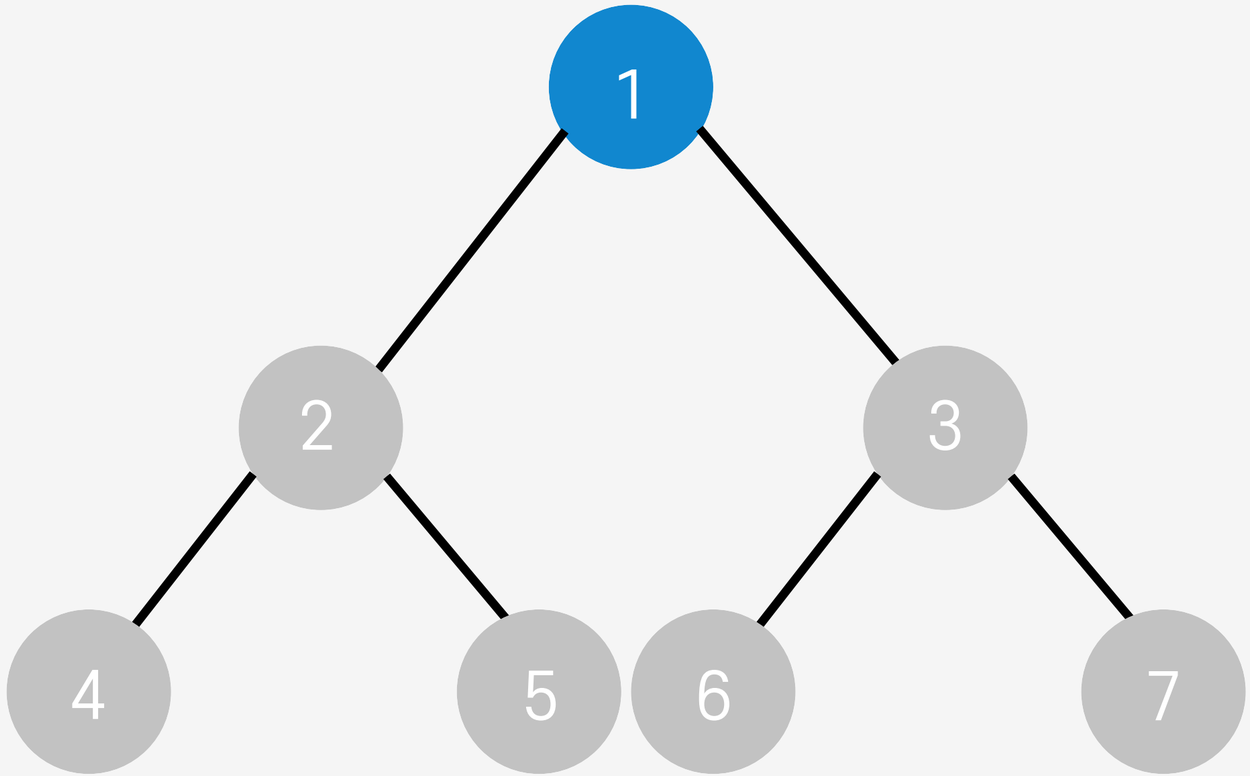
큐를 이용한다 = 선입선출의 원칙으로 탐색한다.

BFS 과정

현재 위치

방문 x

방문 o

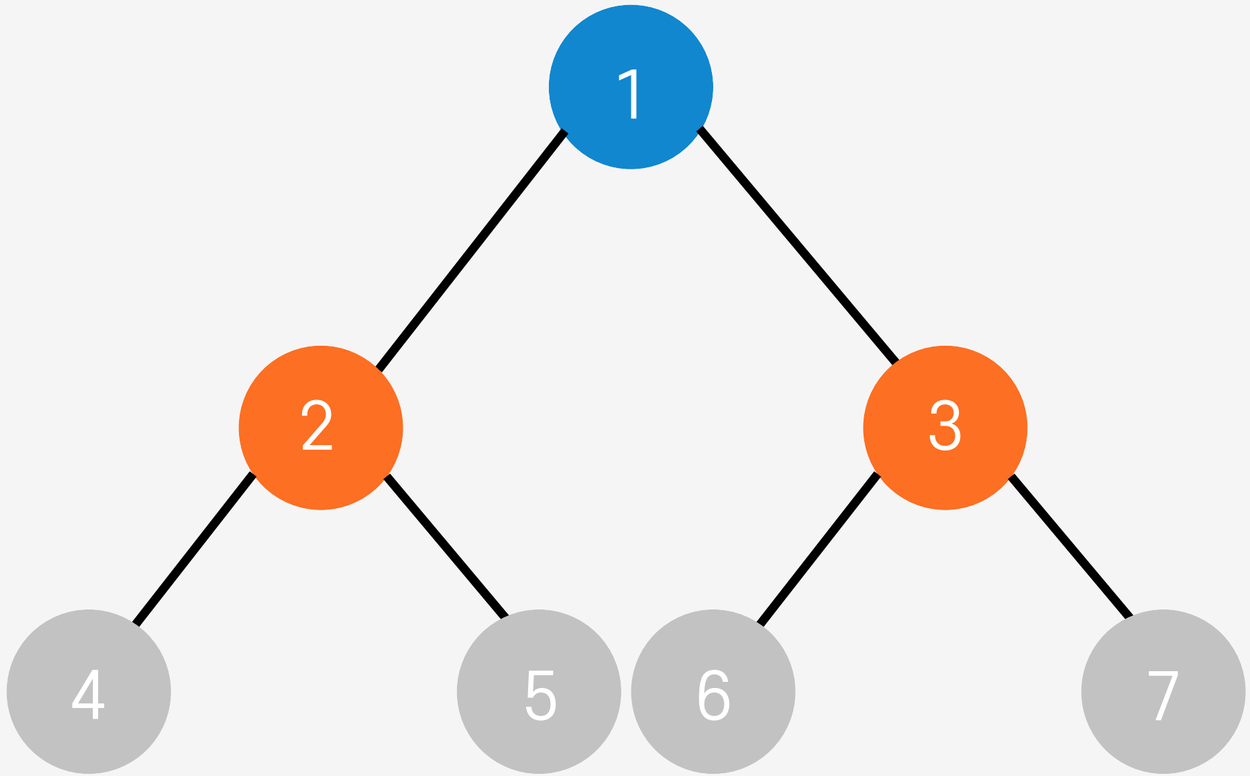


BFS 과정

현재 위치

방문 x

방문 o

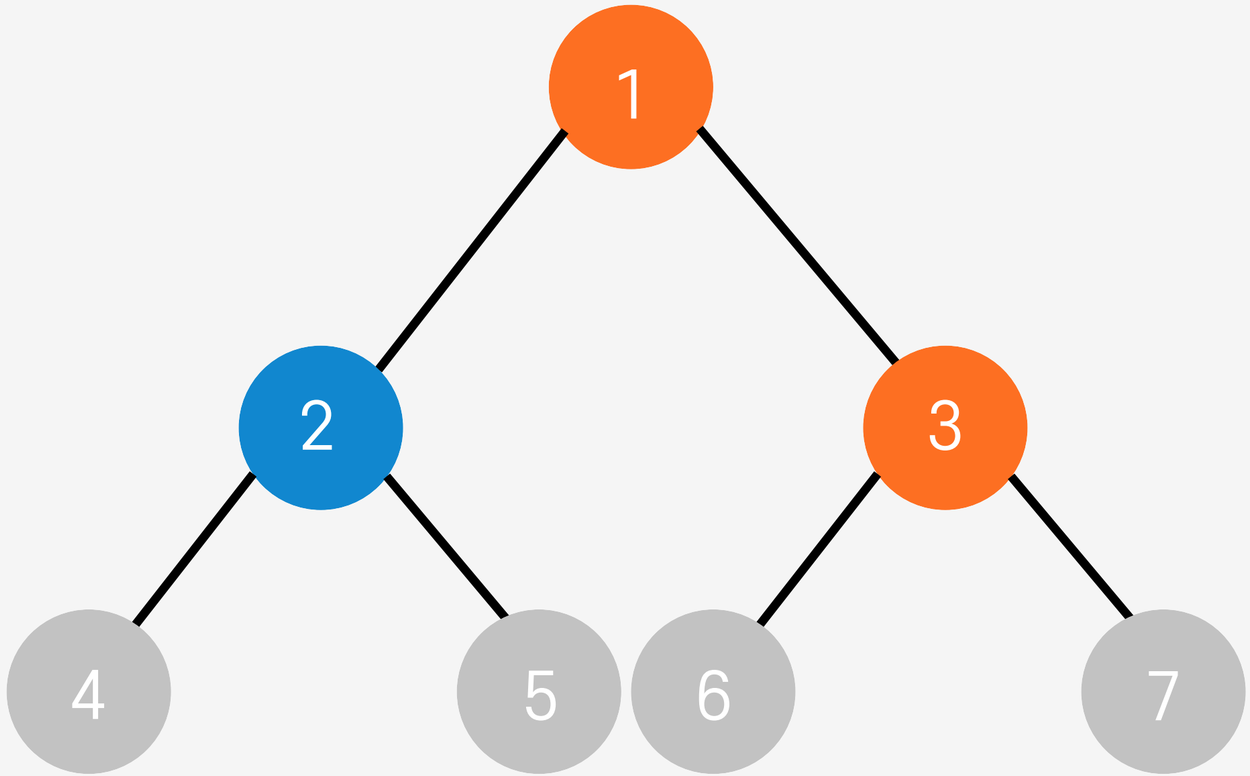


BFS 과정

현재 위치

방문 x

방문 o

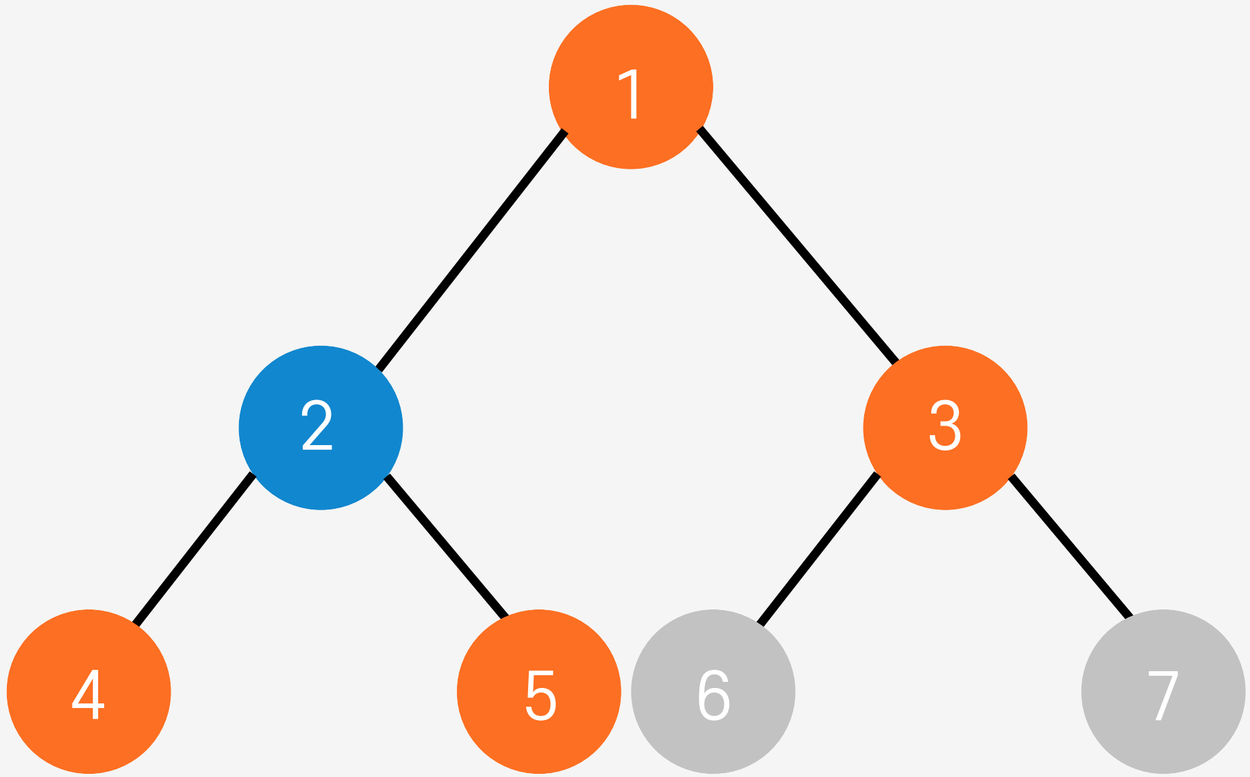


BFS 과정

현재 위치

방문 x

방문 o

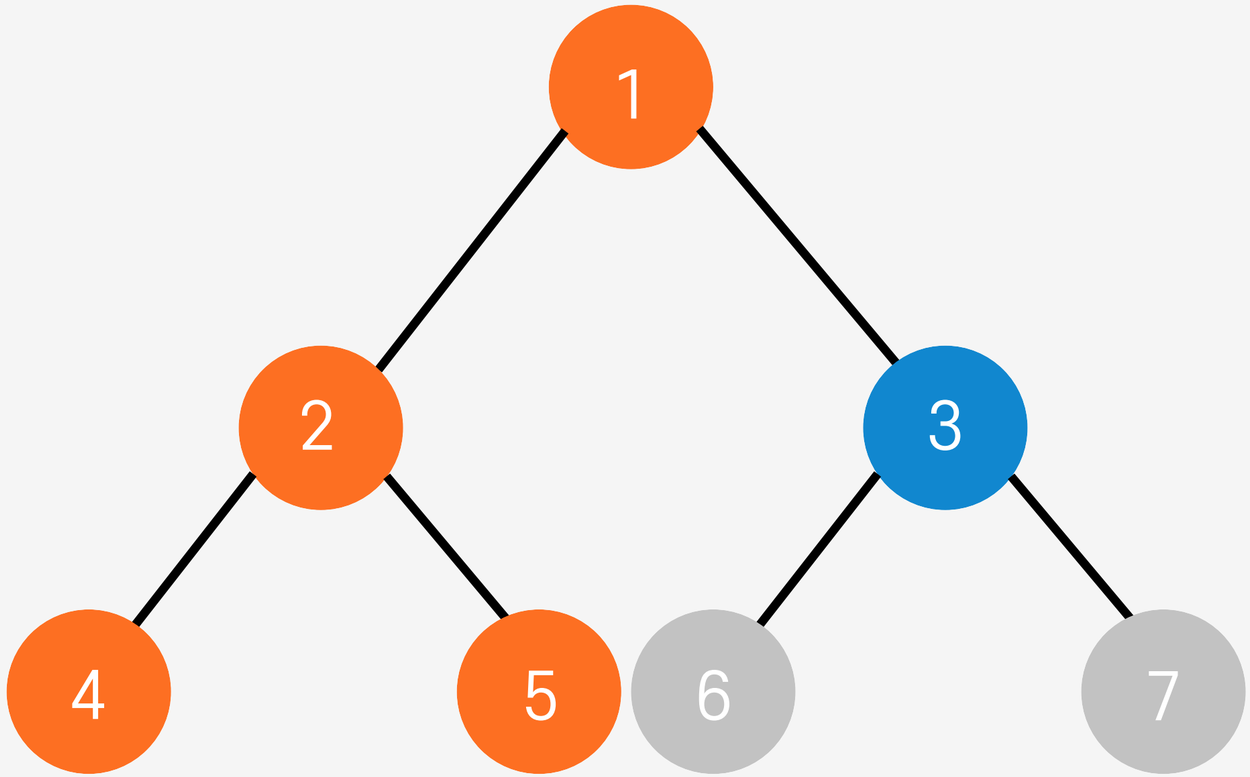


BFS 과정

현재 위치

방문 x

방문 o

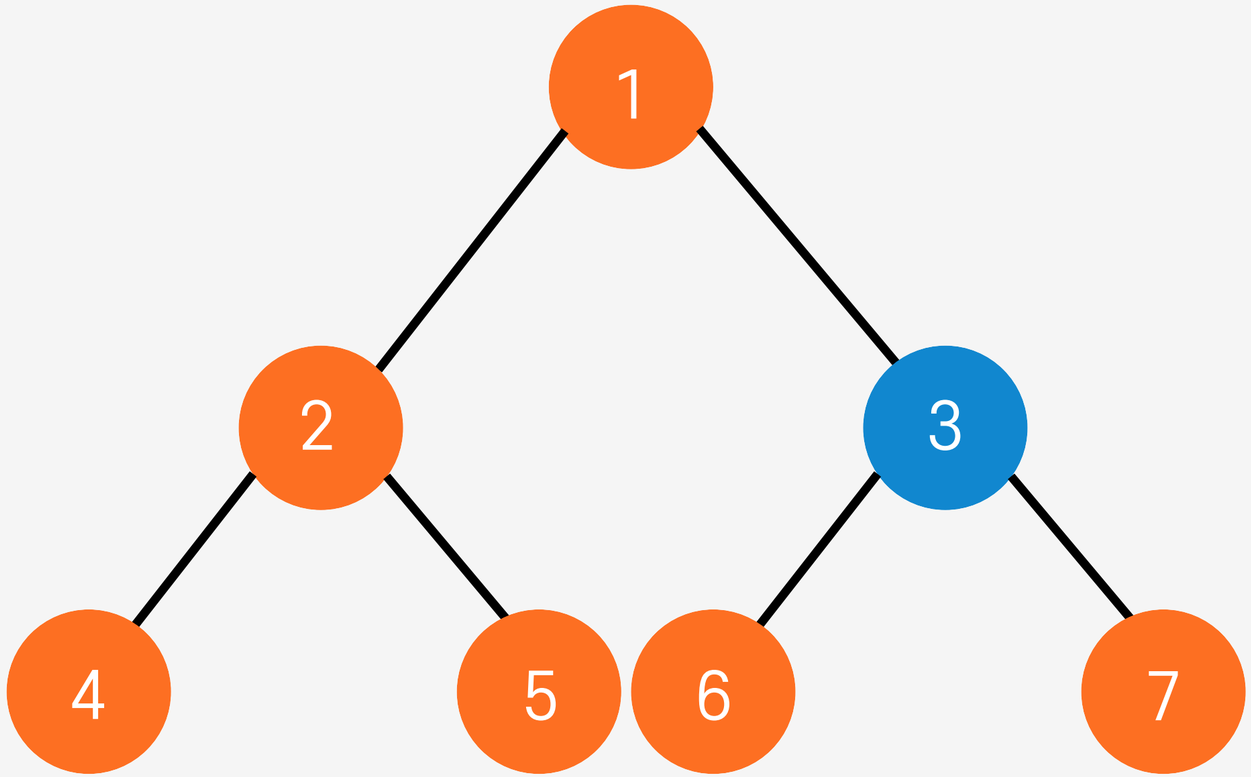


BFS 과정

현재 위치

방문 x

방문 o

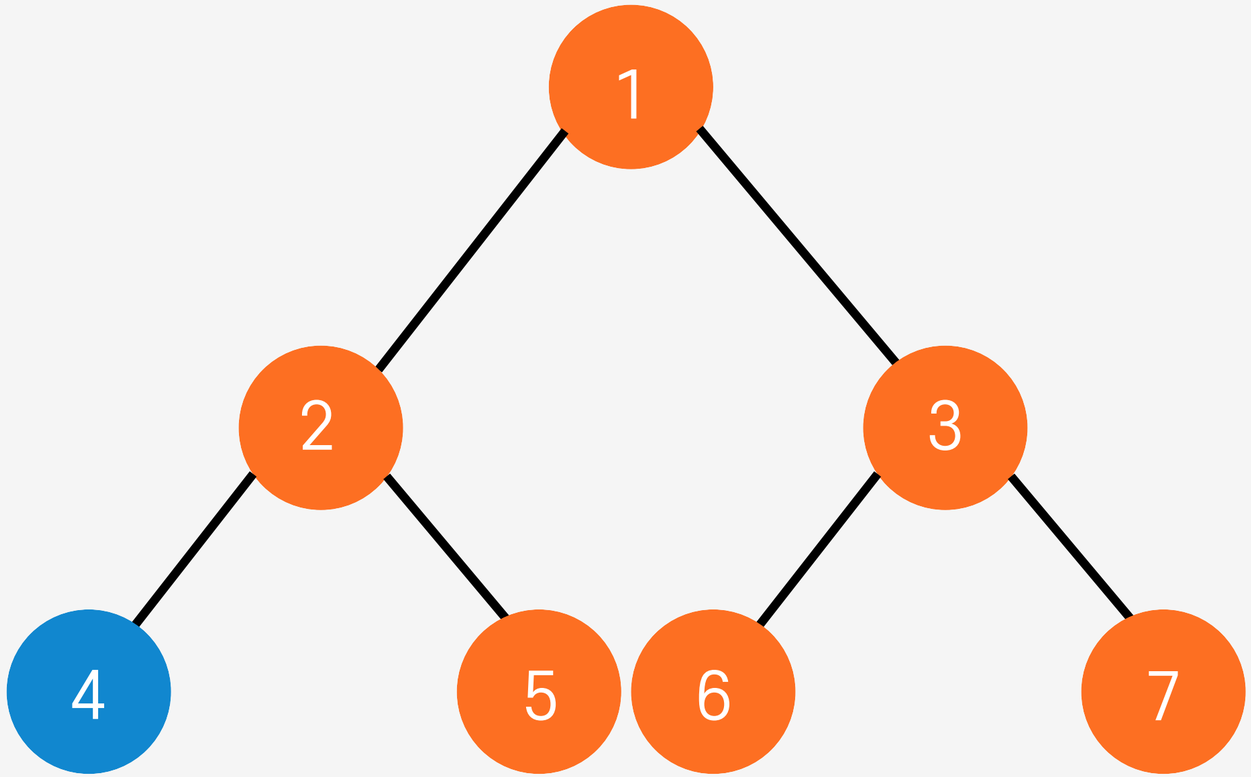


BFS 과정

현재 위치

방문 x

방문 o



BFS 구현

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

typedef long long ll;

bool check[100];
vector<int>v[100];

int main()
{
    queue<int>q;
    q.push(1); //시작 노드 삽입
    check[1] = true;
    while (!q.empty()){ //큐가 비어있을 때 -> 더 이상 갈 곳이 없을 때 종료
        int node = q.front(); //큐의 맨 앞 원소
        q.pop();
        for (int i = 0; i < v[node].size(); i++) {
            int next = v[node][i];
            if (check[next]) continue; //이미 방문한 노드라면 스킵
            check[next] = true; //방문 체크
            q.push(next);
        }
    }
    return 0;
}
```

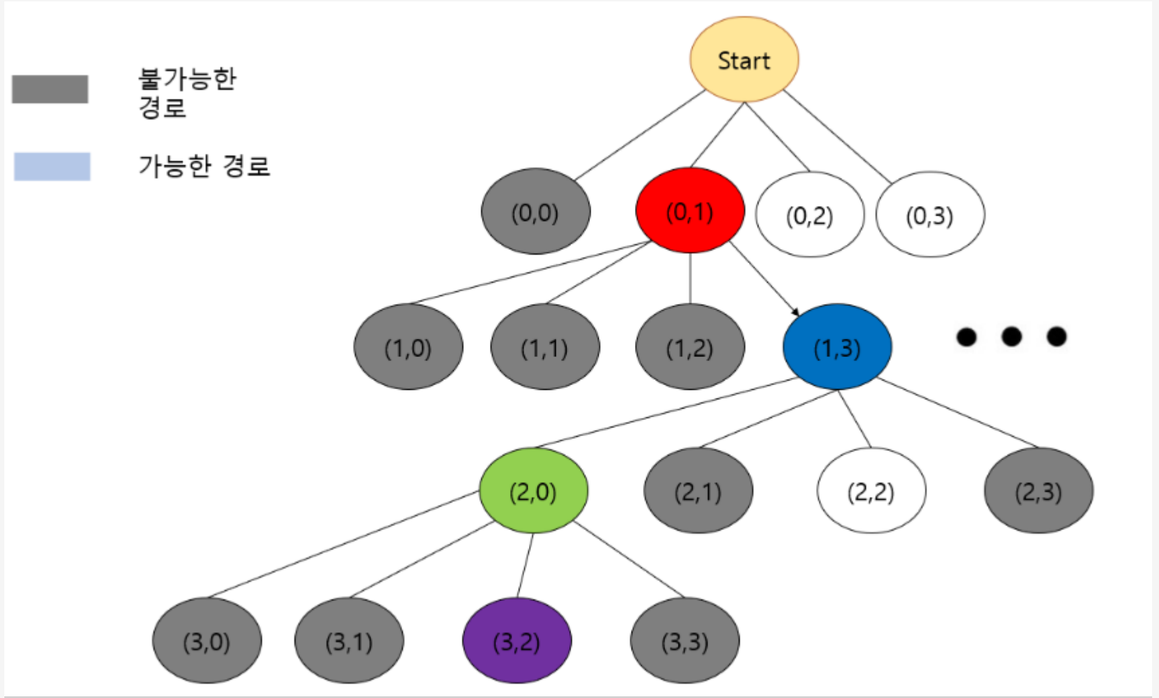
BFS 구현 - 최단 거리

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

typedef long long ll;

bool check[100];
vector<int>v[100];
int dist[100]; //dist[i] : 시작 노드 ~ i번째 노드까지의 최단 거리
int main()
{
    queue<int>q;
    q.push(1); //시작 노드 삽입
    check[1] = true; dist[1] = 0;
    while (!q.empty()){ //큐가 비어있을 때 -> 더 이상 갈 곳이 없을 때 종료
        int node = q.front(); //큐의 맨 앞 원소
        q.pop();
        for (int i = 0; i < v[node].size(); i++) {
            int next = v[node][i];
            if (check[next]) continue; //이미 방문한 노드라면 스킵
            check[next] = true; //방문 체크
            dist[next] = dist[node] + 1;
            q.push(next);
        }
    }
    return 0;
}
```

DFS

우리 본 적 있어요!

단, 최단 거리가 아닐 수 있다!



DFS

단, 최단 거리가 아닐 수 있다!

-> 최단 거리일 때 초기화 로직 필요

DFS 코드

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

typedef long long ll;

vector<int>v[100];
int dist[100]; //dist[i] : 시작 노드 ~ i번째 노드까지의 최단 거리

void dfs(int node) {
    for (int i = 0; i < v[node].size(); i++) {
        int next = v[node][i];
        if (dist[next] == -1 || dist[next] > dist[node] + 1) {
            dist[next] = dist[node] + 1;
            dfs(next);
        }
    }
}
int main()
{
    memset(dist, -1, sizeof(dist));
    dfs(1);
    return 0;
}
```

참고

플러드 필

연결된 영역(연결 요소)가 몇 개인지 찾는 알고리즘

0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

<그림 1>

0	1	1	0	2	0	0
0	1	1	0	2	0	2
1	1	1	0	2	0	2
0	0	0	0	2	2	2
0	3	0	0	0	0	0
0	3	3	3	3	3	0
0	3	3	3	0	0	0

<그림 2>

해결 방법

모든 정점(칸)에 대해 bfs 또는 dfs를 하되, 방문한 정점은 모두 체크해둔다.

만약 bfs할 정점이 체크되어 있으면 패스

총 bfs를 한 횟수 = 연결된 영역(연결 요소)의 개수