알까기 4주차

BFS, DFS

그래프 순회

- 그래프의 각 정점을 방문하는 절차

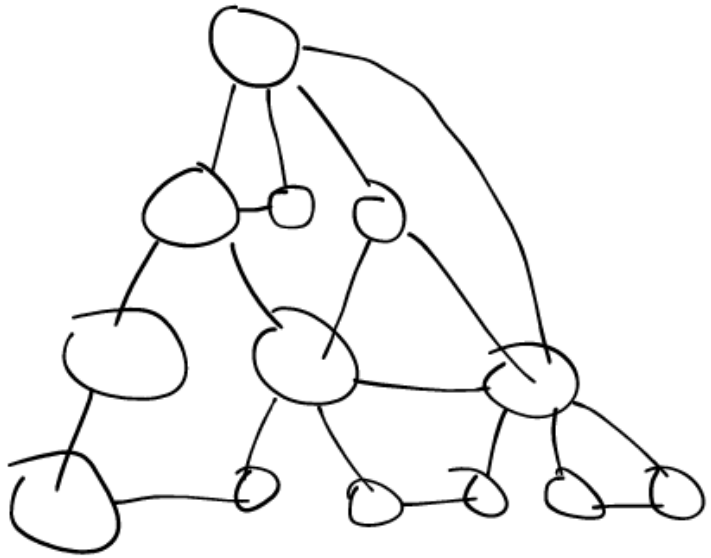
- 그래프 순회의 종류는 정점을 방문하는 순서에 따라 분류됨.

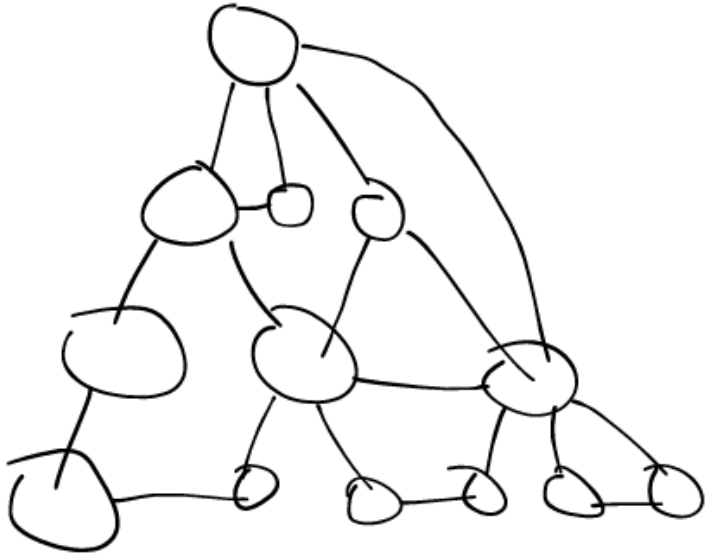
- 여러 번 방문하는 것을 방지하기 위해서 정점의 방문 여부를 관리하기도 함.
  - 일부 특별한 그래프에서는 명시적으로 정점의 방문 여부를 저장하지 않기도 함.
  - 이 경우에는 그래프의 구조가 정점을 중복으로 방문하지 않는 것을 보장함.

BFS, DFS

- **너비 우선 탐색, BFS** Breadth First Search
  - 한 정점과 인접한 정점들을 **모두 방문**하여 그래프를 순회

- **깊이 우선 탐색, DFS** Depth First Search
  - 한 정점에서 **깊이가 깊어지는 방향**으로 정점을 방문하여 그래프를 순회
  - 더 이상 나아갈 정점이 없다면 이전 정점으로 돌아옴





BFS

- 한 정점과 인접한 정점이 여러 개라면 어떤 정점을 우선해서 방문할까?
  - 따로 정해진 게 없다.
  - 아무 순서로 방문하면 됨.

- **Queue를 사용함.**

- 시작 정점을 Q에 넣음.
- Q가 빌 때까지 다음 과정을 반복
  - Q에서 정점 하나 u를 제거함.
  - u와 인접한 모든 정점 v에 대하여, v가 이전에 방문하지 않은 정점이면 Q를 v에 넣음.

BFS

```cpp
int N; // 정점 수

vector<vector<int>> graph(N); // 인접 그래프
vector<int> vis(N); // 정점 방문 여부
queue<int> Q; // 큐

Q.push(1);
vis[1] = 1;

while (!Q.empty()) {
    int u = Q.front(); Q.pop();
    vis[u] = 1;
    cout << u << " ";
    for (int v : graph[u]) {
        if (!vis[v]) Q.push(v);
    }
}
```

DFS

- 한 정점과 인접한 정점이 여러 개라면 어떤 정점을 우선해서 방문할까?
  - 따로 정해진 게 없다.
  - 아무 순서로 방문하면 됨.

- **Stack을 사용함.**
  - 구현의 편의를 위해 보통 재귀 함수를 사용함.

- **재귀 함수** F(u)를 정의하자. (단, u는 정점)
  - u와 인접한 정점 v에 대하여, v를 방문하지 않았으면 F(v)를 호출한다.

- **F(시작 정점)**을 호출한다.

DFS

```
int N; // 정점 수
vector<vector<int>> graph(N); // 인접 리스트
vector<int> vis(N); // 방문 여부

void dfs(int u) {
    vis[u] = 1;
    cout << u << " ";
    for (int v : graph[u]) if (!vis[v]) dfs(v);
}


...

dfs(1);
```

BFS / DFS – 시간 복잡도

- 시간 복잡도 : O(V + E)
  - 정점을 O(V)번 방문함
  - 총 O(E)개의 간선을 거침 **handshaking lemma**

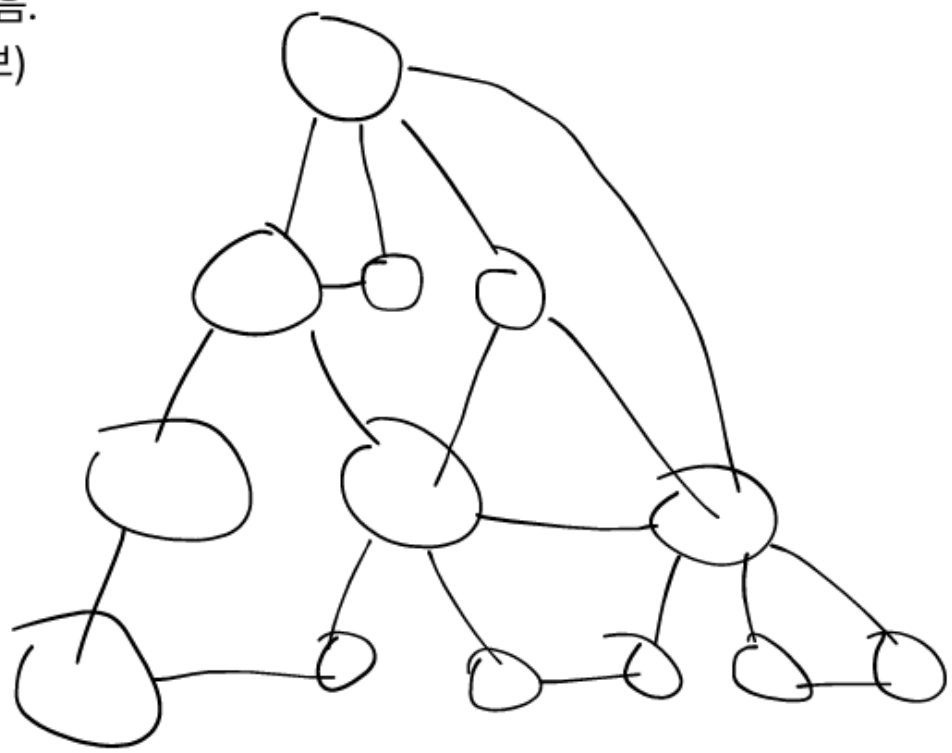
- 공간 복잡도 : O(V + E)
  - 인접 리스트 : O(V + E)
  - 방문 여부 배열 : O(V)

BFS – 최단경로

모든 간선의 가중치가 같은 그래프에서 BFS를 사용하여 최단 경로 그래프를 구할 수 있다.

단순히 시작 정점에서 BFS를 돌려주는 것으로 모든 정점과의 최단 경로를 구할 수 있음.
여기서 관리할 정보는 현재 정점이 시작 정점과 얼마나 멀리 떨어져 있는가. (거리 정보)

- 다음 정점으로 넘어갈 때, **현재 정점 거리 + 1**이 다음 정점의 최단 경로 거리가 된다.



DFS – 최단 경로,,?

- DFS도 최단 경로를 구하는 데 사용할 수 없을까?
  - 물론, 예시의 DFS 코드를 조금만 수정하면 DFS로도 최단 경로를 구할 수 있다.

- DFS는 정점 하나를 잡고 간선을 따라서 계속 이동한다.
- 최단 경로를 갱신하기 위하여 이미 방문한 정점을 다시 방문하여 처리해야 할 수도 있다.

- $O(V^2)$ 시간이 걸릴 수 있음.
  - BFS보다 비효율적이라서 일반적으로 사용하지 않음.

문제...in

- 1260번: DFS와 BFS (acmicpc.net)

이분탐색

정렬된 데이터에서 원하는 값을 찾아내는 알고리즘

대상 데이터의 중앙값과 찾고자 하는 값(target)을 비교하여

데이터의 크기를 절반씩 줄여나가는 방식입니다