깊이 우선 탐색

모든 정점을 특정 순서에 따라 방문하는 알고리즘
- 탐색

그래프 - 트리보다 복잡한 구조
↳ 탐색과정에서 얻어지는 정보가 굉장히 중요

탐색 과정에서 알 수 있는 정보
- 어떤 간선이 사용되었나?
- 어떤 순서로 정점들이 방문되었나?
↳ 그래프 구조 파악

널리 알려진 두가지 탐색 알고리즘

DFS
깊이우선 탐색

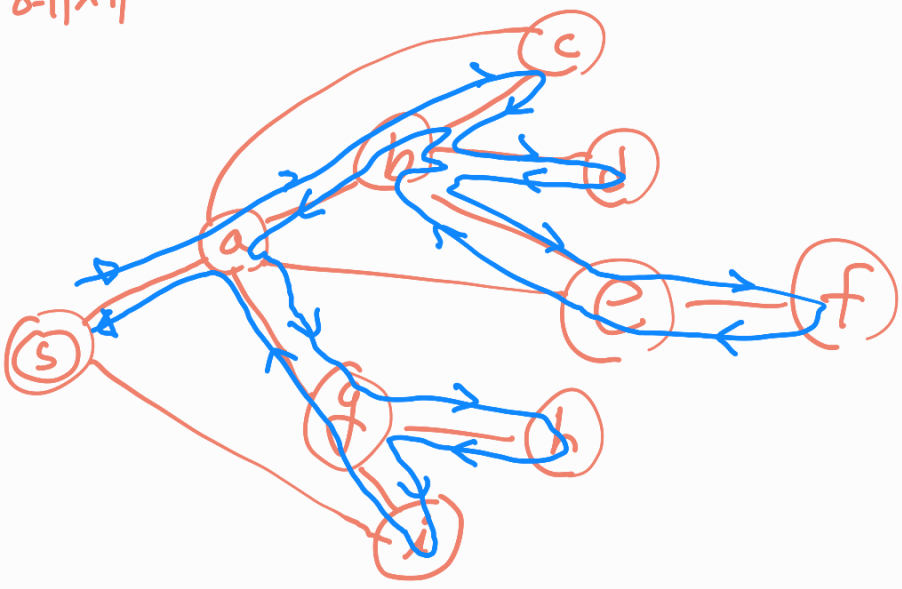
BFS
너비우선 탐색

DFS 에대한 설명, 문제, 해법

DFS 그래프의 모든 정점을 발견하는 가장 단순하고 고전적인 방법

인접한 간선을 하나씩 검사 → 방문X 정점으로 향하는 간선 무조건 따라감
↓
이 이상 갈곳이 없을 때 다시 왔던대로 돌아감

DFS 예제



DFS의 중요한 특성: 더 따라갈 간선이 없을경우 이전으로 돌아가야 함. ⇒ 지금까지 방문한 정점을 모두 저장해 두어야 한다.

⇓

재귀 호출을 이용하여 간단하게 할 수 있다.

dfs 코드

dfs()는 아직 방문하지 않은 정점으로 이어지는 간선이 있을경우 재귀 호출로 해당 정점을 방문한다.

Vector <vector<int>> adj ← 그래프 인접리스트

Vector <bool> visited ← 방문여부 (bool로 초기화)

```
dfs(int here){
    print "Dfs visits ${here}"

    visited[here] = true
```

인접 정점 순회

```
    for ( i=0; adj[here].size(); i++ ){
        int there = adj[here][i];
        if (!visited[there])
            dfs(there);
    }
}
dfsAll(){
    visited = bool로초기화
    for(i=0; i < adj.size; i++){
        if (!visited[i])
            dfs(i);
    }
}
```

dfs 의 시간복잡도

그래프의 표현 방식에 따라 달라진다.

1. 인접리스트 방식 $|V|+|E|$

→ dfs는 정점을 방문할때 1번씩 호출된다.

$|V|$번

dfs의 안에는 for 문이 존재하는데 이는 간선의 갯수만큼만증가함

2. 인접행렬 방식 $|V|^2$

→ dfs 정점방문 1번씩

$|V|$번 for문은 언제나 1번돈다.

DFS의 사용

1. 두점사이를 잇는 경로가 있는지 확인

dfs(u) 를하고 생성된 visited 를 참조하면 된다.

2. 연결된 부분집합의 개수 세기

서로 연결되지 않은 그래프가

있을경우

dfs로 손쉽게 연결된 그래프를

확인할 수 있다.

3. 위상정렬

별개의 필기로 기록한다.

4. 오일러 서킷

별개의 필기로 기록한다.