그래프의 너비우선 탐색

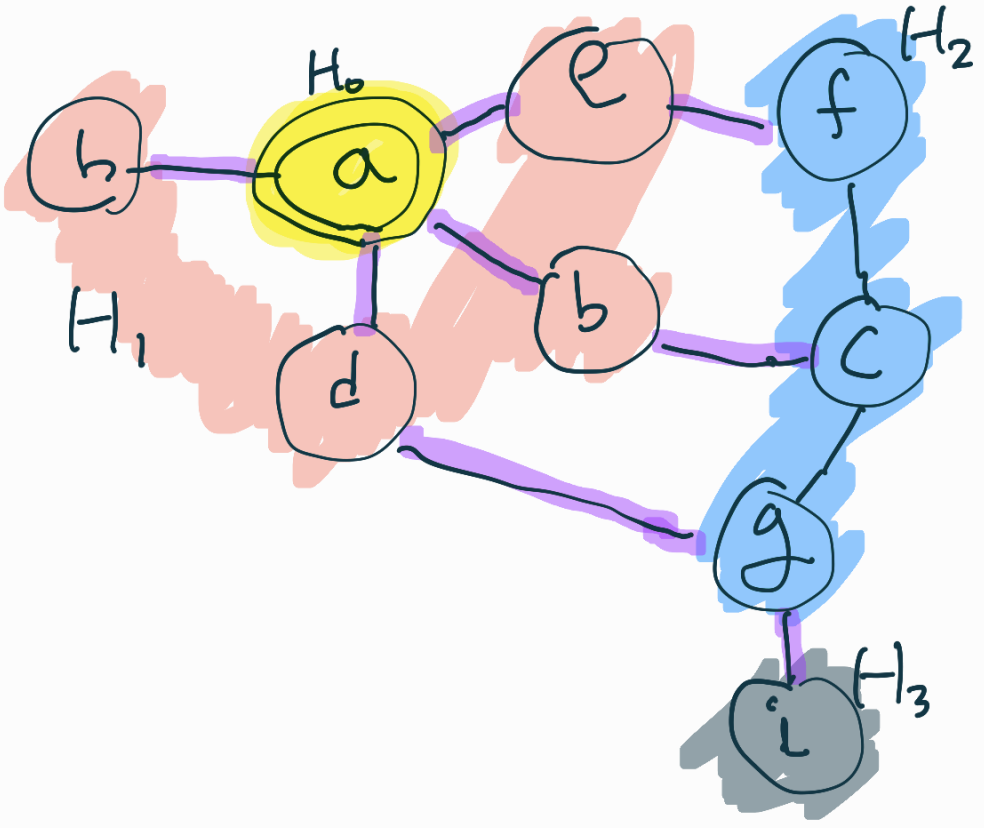
깊이우선 탐색과 함께 가장 널리 사용되는
그래프 탐색 알고리즘인 너비우선탐색에 대해 다룬다.

다익스트라, 최단거리알고리즘, 최소스패닝트리 알고리즘

의 기초

동작과정이 매우 이해하기 쉽다.

why? 시작점에서 가까운 정점부터 순서대로
방문하기 때문



사용 간선

$H_0 \rightarrow H_1 \rightarrow H_2 \rightarrow H_3$ 순으로 방문해 나간다.

(정점과 시작점 사이 경로가 여러개면 최단경로 사용)

너비우선탐색

- 각 정점의 방문시 마다 모든 인접정점을 검사한다.
  - if 처음보는 정점 발견?
    - ↳ "나여기 방문할거에요" 라고 따로 기록하기

인접한 정점을 모두 검사하고 나면?

여기서 꺼내서 방문한다

BFS의 방문 순서는 정점의 목록에서 부터 꺼내지에 따라 달라진다.

위를 구현하는 간단한 방법

방문예정목록 을 FIFO로 구현하면된다
(queue)

정점목록

BFS 코드

있어야 하는것
- 방문 예정 정점 큐
- 방문여부를 나타내는 배열
- 방문 순서를 저장할 배열

```
function bfs(start){
    const queue = [];
    const descovered = new Array(graph.length)
                        .fill(true);
    const order = [];
    discovered[start]= true;
    queue.push(start);
    while(!q.length===0){
        const here = queue.shift();
        order.push[here)
        for(let i=0; i<graph[here].length;i++)
        {
            const there = graph[here][i];
            if(!discovered[there]){
                queue.push(there);
                discovered[there]=true;
            }
        }
    }
    return order
}
```

다양하게 사용되는 DFS와 달리
BFS는 대개 최단경로 문제에서만
사용된다.

최단 경로 문제

두 정점을 연결하는 경로 中 가장
짧은 경로를 찾는 문제

네비게이션탐색 알고리즘을 수정해
시작점으로 부터의 거리 distance[]
을 계산하게 할 수 있다.

정점 V를 발견해 Queue에 넣을때
거리가 1 증가한다.

시작점 부터 다른 모든 정점까지의
최단 경로를 네비게이션 탐색 스타일대로 구현할 수 있다.

너비 우선 탐색 스패닝 트리을 계산하는 bfs 그리고
그를 활용 실제 최단 경로를 계산하는
shortest path() 의 구현)