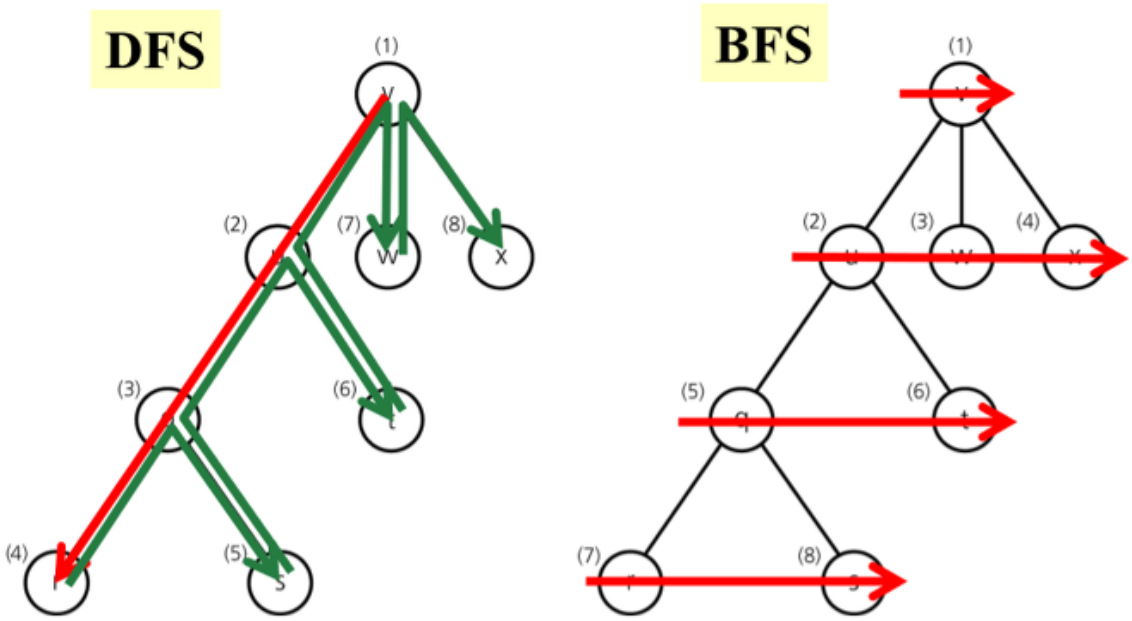
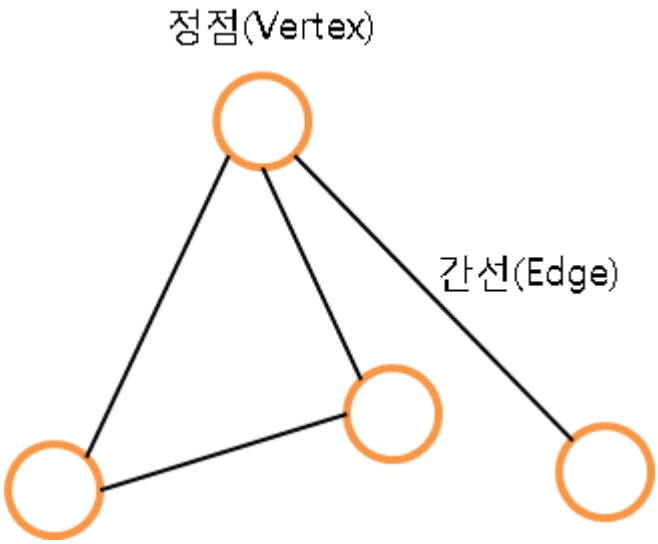
DFS & BFS



그래프 순회(Graph Traversals,그래프 탐색)



:그래프는 정점과 간선으로 이루어진 자료구조의 일종

: 그래프 순회는 그래프의 각 정점을 방문하는 과정을 이야기 한다.

:크게 DFS와 BFS알고리즘이 있다.

문제

<그림 1>과 같이 정사각형 모양의 지도가 있다. 1은 집이 있는 곳을, 0은 집이 없는 곳을 나타낸다. 철수는 이 지도를 가지고 연결된 집의 모임인 단지를 정의하고, 단지에 번호를 붙이려 한다. 여기서 연결되었다는 것은 어떤 집이 좌우, 혹은 아래위로 다른 집이 있는 경우를 말한다. 대각선상에 집이 있는 경우는 연결된 것이 아니다. <그림 2>는 <그림 1>을 단지별로 번호를 붙인 것이다. 지도를 입력하여 단지수를 출력하고, 각 단지에 속하는 집의 수를 오름차순으로 정렬하여 출력하는 프로그램을 작성하시오.

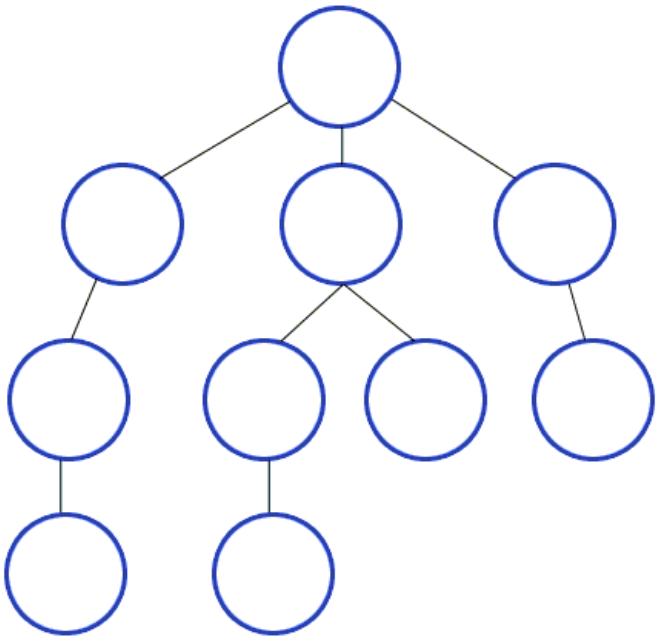
0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

<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>

DFS(Depth-first Search, 깊이 우선 탐색)



: 현재 정점에서 갈 수 있는 점들까지 들어가면서 탐색하는 방법

: 주로 스택으로 구현하거나 재귀로 표현하며 **백트래킹**에도 뛰어난 효용을 보인다

: 일반적으로 BFS에 비해 널리 사용된다

: Sudo code (재귀구조로 표현한 DFS에 대한 sudo code)

```
DFS(G,v)
  label v as discovered
  for all directed edges from v to w that are in G.adjacentEdges(v) do
    if vertex w is not labeles as dicovered then
      recursively call DFS(G,v)
```

장점.

-BFS에 비해 저장공간의 필요성이 적다

-찾아야 하는 노드가 깊은 단계에 있을 경우 BFS보다 유리하다

단점.

-답이 아닌 경로가 매우 깊다면 그 경로에 빠질 위험이 있다

→이러한 문제를 해결하기 위해 깊이제한(depth bound)를 두기도 함

-답의 경로가 다수인 경우 탐색이 답을 찾으면 거기서 탐색을 끝내기 때문에 도출된 답이 최단 경로일 것이라는 보장이 없다

*백트래킹(Backtracking)

: DFS보다 좀 더 광의의 의미

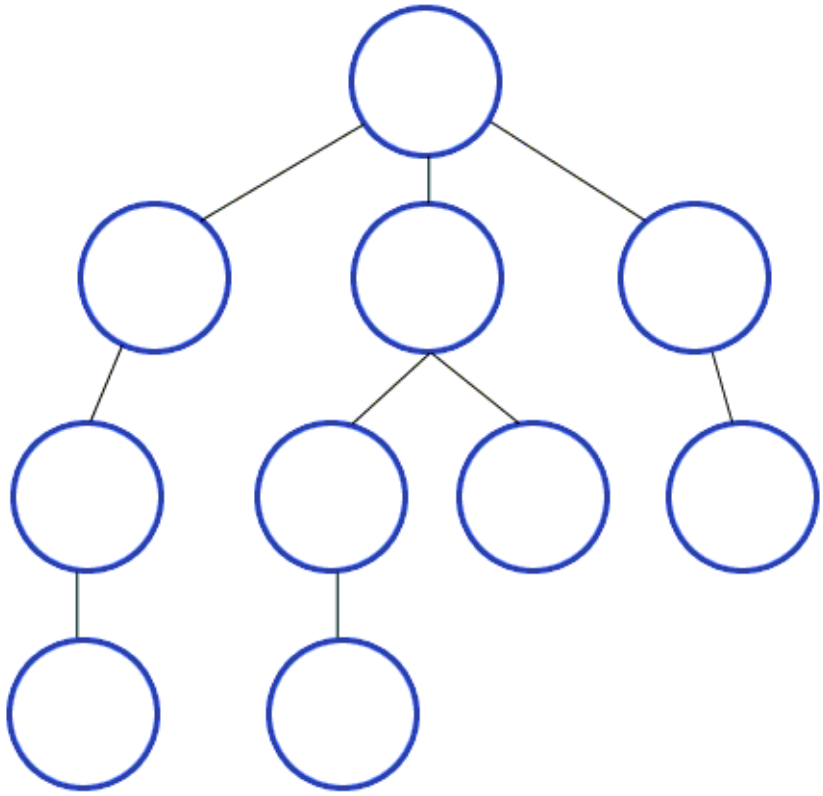
: 해결책에 대한 후보를 구출해 나아가다 가능성이 없다고 판단되는 즉시 후보를 포기(=백트랙)해 정답을 찾아가는 범용적인 알고리즘으로 “제약 충족 문제”에 특히 유용

*제약 충족 문제(Constraint Satisfication Problems)

: 수많은 제약 조건을 충족하는 상태를 찾아내는 수학문제를 일컫는다

: 스도쿠, 조합 최적화, 십자말 풀이 , 8퀸 문제가 대표적인 예

BFS(Breadth-first Search, 너비 우선 탐색)



: 현재 정점에 연결된 가까운 점부터 탐색하 방법

:주로 큐로 구현하며, 그래프의 최단 경로를 구하는 문제(다익스트라 알고리즘) 등에 사용된다

: Sudo code (큐를 이용한 반복 구조로 구현)

```
BFS (G, start_v)
  let Q be a queue
  label start_v as discovered
  Q.enqueue(start_v)
  while Q is not empty do
  v := Q.deqeue()
  if v is the goal then
    return v
  for all edges from v to w in G.adjacentEdges(v) do
    if w is not labeled as discovered then
      label w as discovered
      w.parent := v
      Q.enqueue(w)
```

: BFS는 재귀로 동작하지 않는다 (=재귀로는 구현 불가)

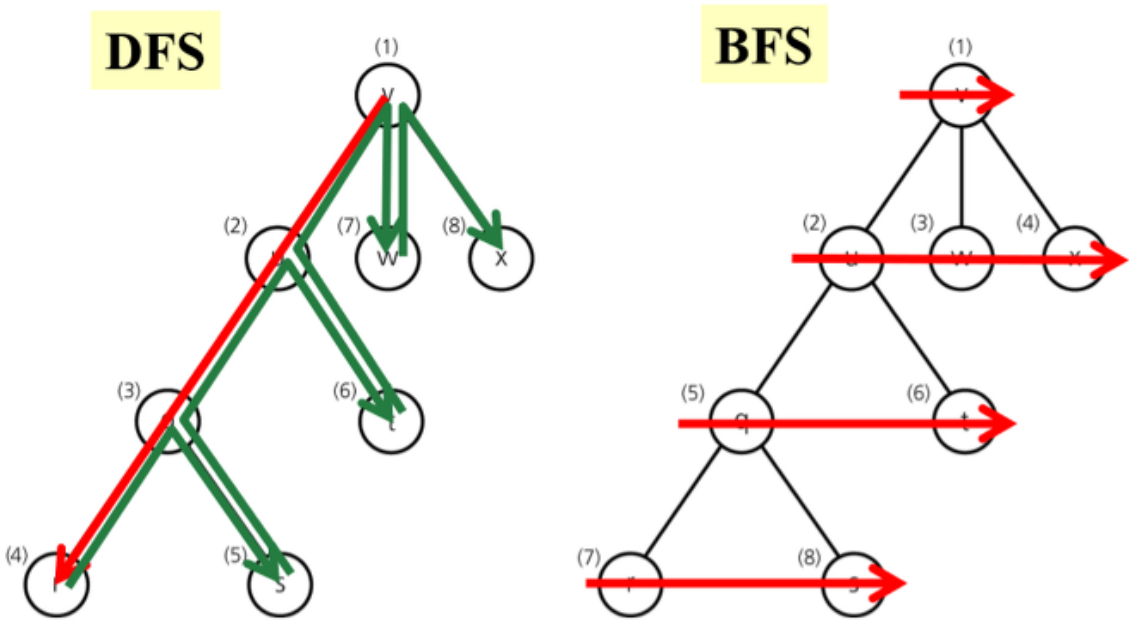
장점.

-너비를 우선으로 탐색하기 때문에 여러 개의 답이 존재할 경우에도 최단 경로임을 보장

-노드의 수가 적고 깊이가 얕은 해가 존재할 때 유리잠

단점.

-노드의 수가 많을 수록 필요없는 노드들까지 저장해야 하기 때문에 많은 저장공간을 사용



- 질문사항

Q. 왜 BFS를 큐로 구현해야 하는지?

A. BFS는 그래프 탐색을 한 후 어떤 노드를 방문했는지 여부를 반드시 검사해야 한다.

이때, 방문한 노드들을 차례로 저장하고 FIFO원칙에 따라 꺼낼 수 있는 자료구조인 큐가 이에 적합하기 때문이다.