BFS & DFS

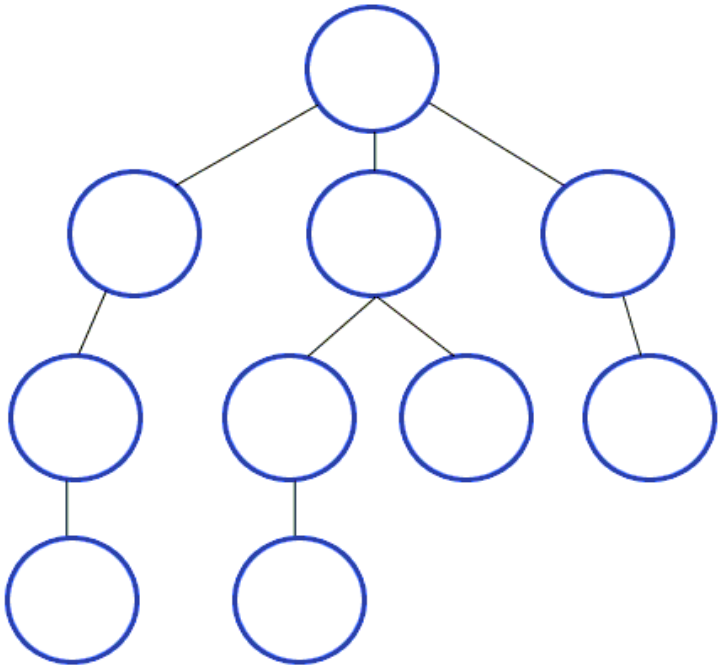
DFS&BFS

• **그래프**
정점(node)과 그 정점을 연결하는 간선(edge)으로 이루어진 자료구조

• **그래프 탐색**
하나의 정점으로부터 시작하여 차례대로 모든 정점들을 한 번씩 방문하는 것

• **그래프를 탐색하는 방법**
-깊이 우선 탐색(DFS)
-너비 우선 탐색(BFS)

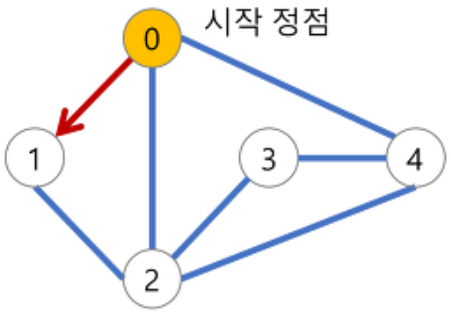
DFS(깊이 우선탐색)



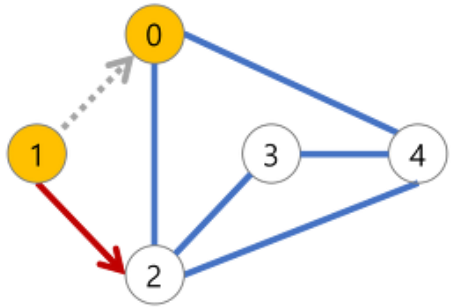
-루트 노드(혹은 다른 임의의 노드)에서 시작해서 다음 분기(branch)로 넘어가기 전에 해당 분기를 완벽하게 탐색
-Stack, 재귀 사용

DFS

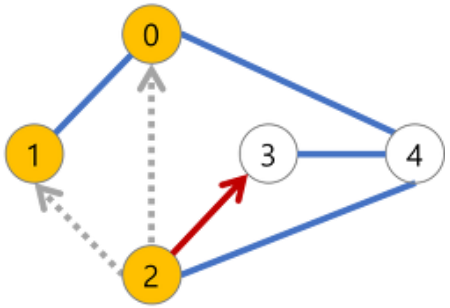
(1) 정점 1 방문



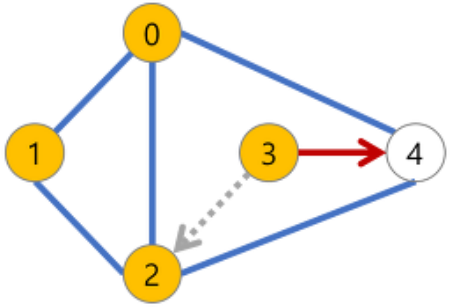
(2) 정점 2 방문



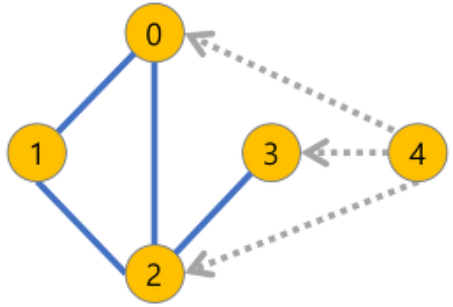
(3) 정점 3 방문



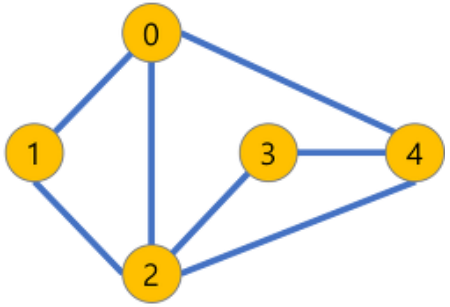
(4) 정점 4 방문



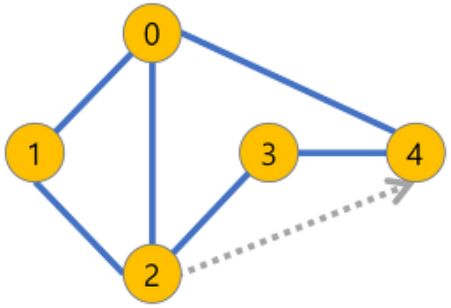
(5) 정점 3으로 backtracking
(다시 돌아와서 탐색하지 않은 정점이 있는지 확인)



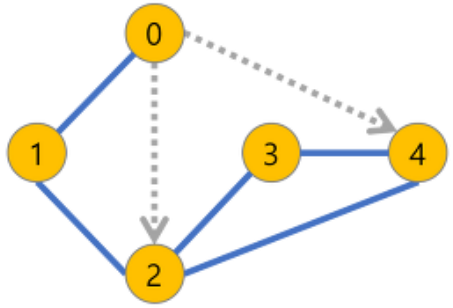
(6) 정점 2로 backtracking



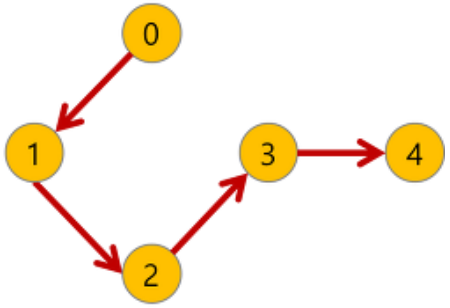
(7) 정점 1로 backtracking



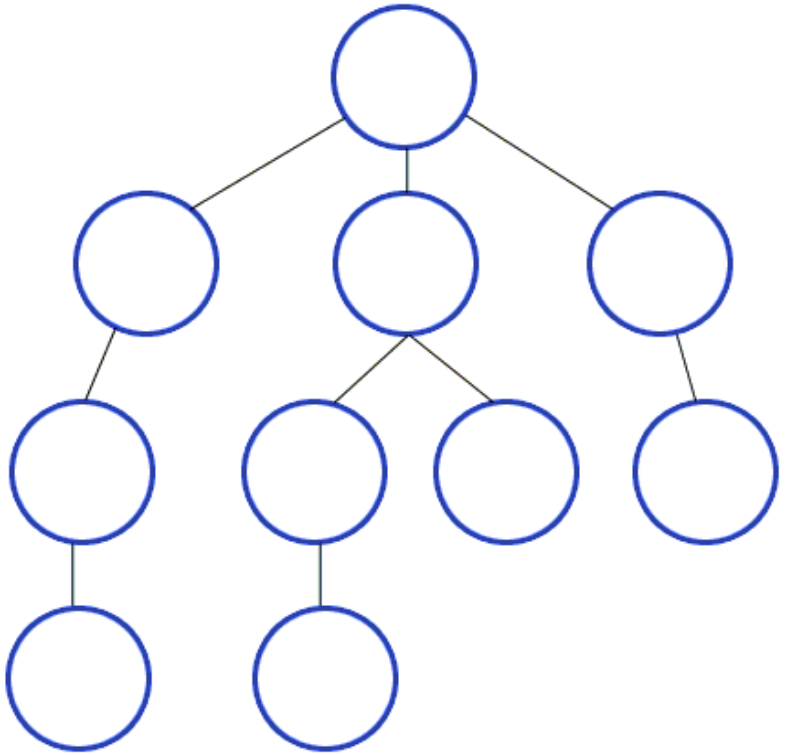
(8) 정점 0으로 backtracking(탐색 종료)



(9) 탐색 결과(방문 순서: 0,1,2,3,4)



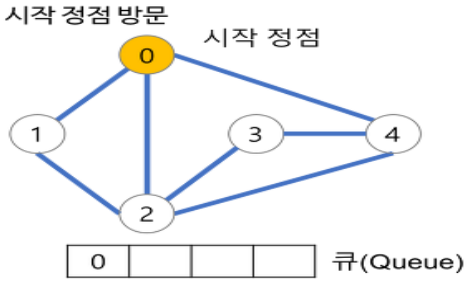
BFS(너비 우선탐색)



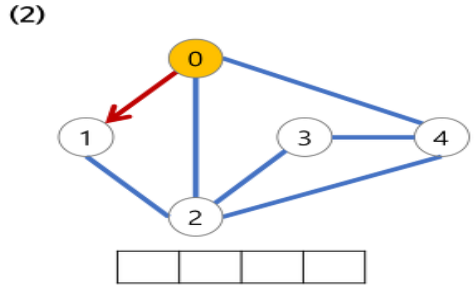
-루트 노드(혹은 다른 임의의 노드)에서 시작해서 인접한 노드를 먼저 탐색하는 방법
-Queue 사용

BFS

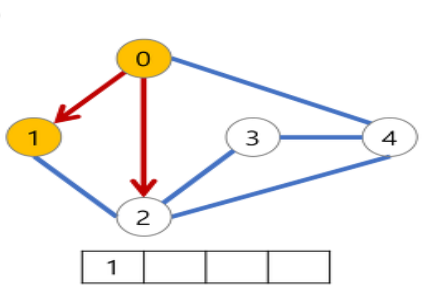
(1) 시작 정점 방문



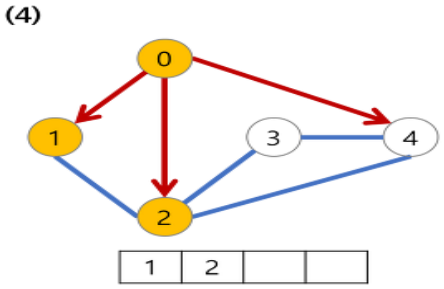
(2)



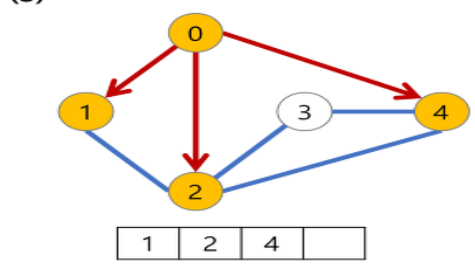
(3)



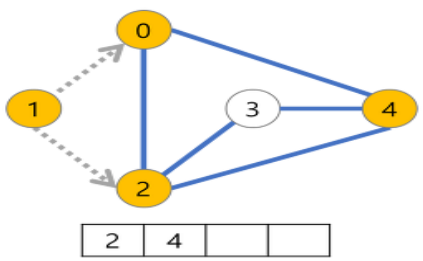
(4)



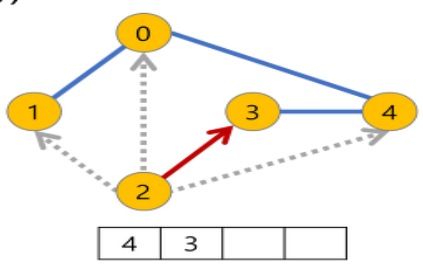
(5)



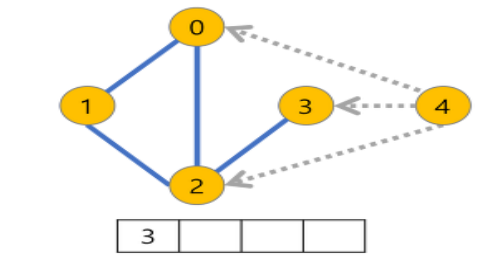
(6)



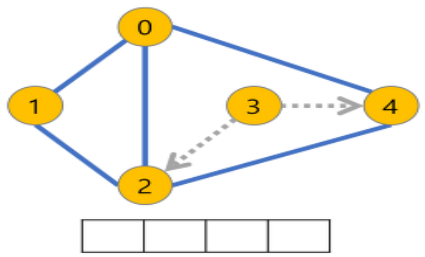
(7)



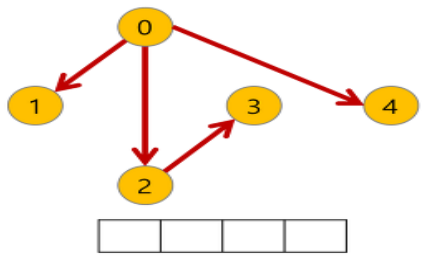
(8)



(9)



(10) 탐색 결과(방문 순서: 0,1,2,4,3)



시간 복잡도

	인접행렬	인접리스트
DFS	O(V^2)	O(V+E)
BFS	O(V^2)	O(V+E)

(V:노드 , E: 간선)

→ BFS / DFS 둘 다 인접행렬보다 인접리스트 방식으로 구현하는 것이 더 효율적이다

활용

• 그래프의 모든 정점을 방문하는 경우

-> DFS, BFS 중 편한 방법 사용

• 가중치가 같은 그래프의 최단거리 구해야 하는 경우

-> BFS를 사용

• 백트래킹 같이 특정 알고리즘을 사용하거나 검색 대상 그래프가 큰 경우

-> DFS 사용