BFS & DFS
(그래프의 탐색 방법)

2019.05.14

16 오하늘

ARGOS

INDEX

ARGOS

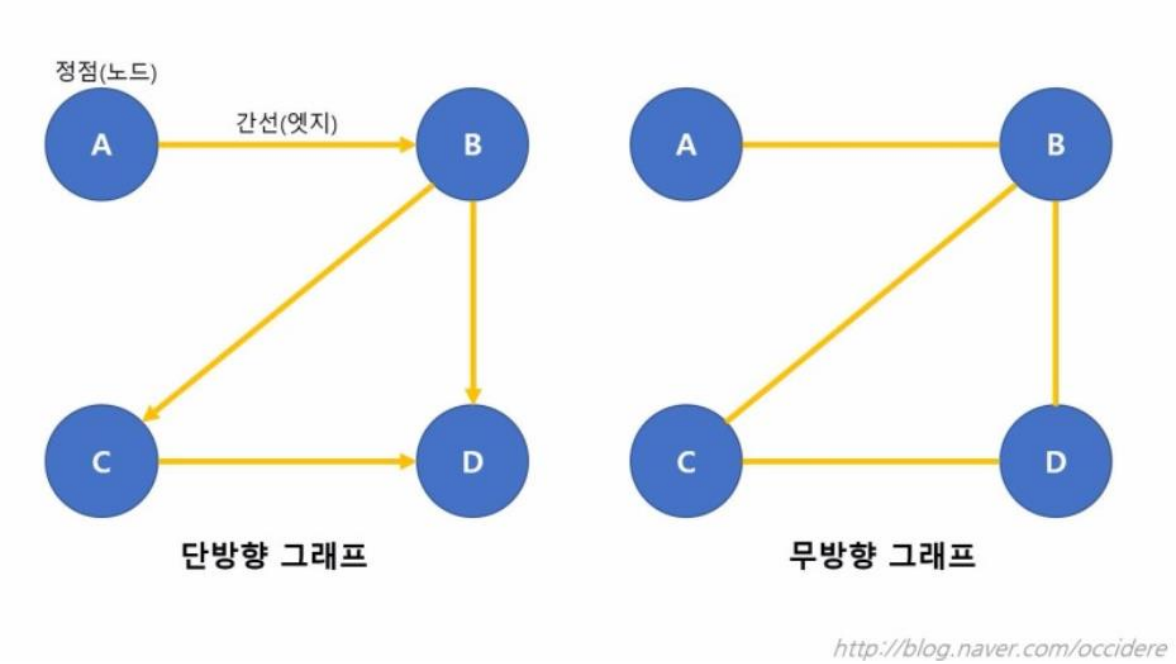
그래프

- 그래프의 정의
- 그래프의 표현
- 그래프의 탐색 기법

ARGOS

그래프의 정의

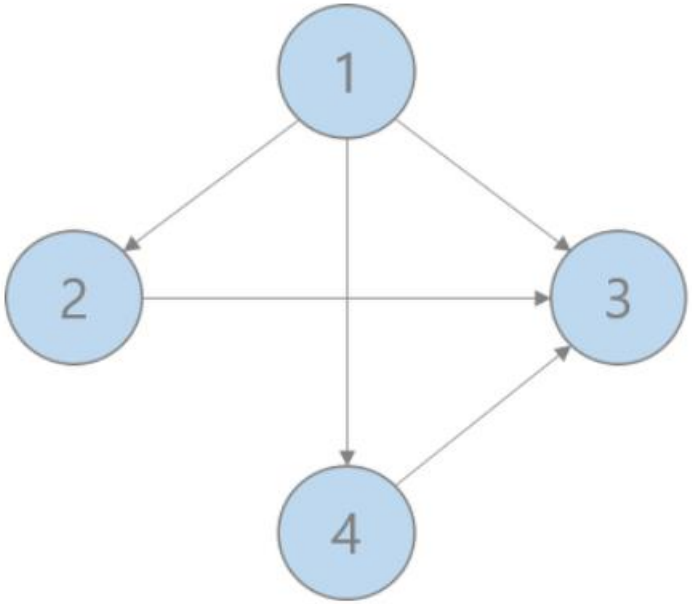
- **정점**(노드)와 **간선(**엣지)으로 이루어진 자료구조
- 간선의 방향 유무에 따라서 **단방향 그래프** 와 **무방향 그래프**(또는 양방향)



그래프의 표현

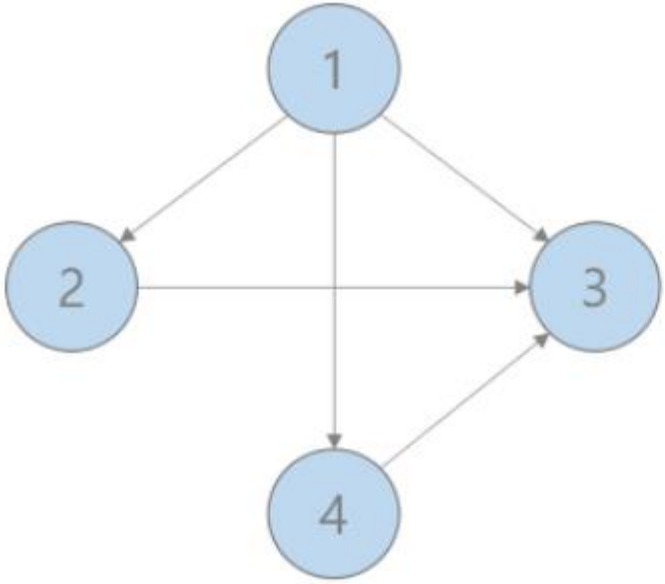
- 크게 **인접 행렬** 그래프와 **인접 리스트** 그래프
- 일반적으로 인접행렬 방식을 많이 사용하며, 경우에 따라 인접 리스트 방식

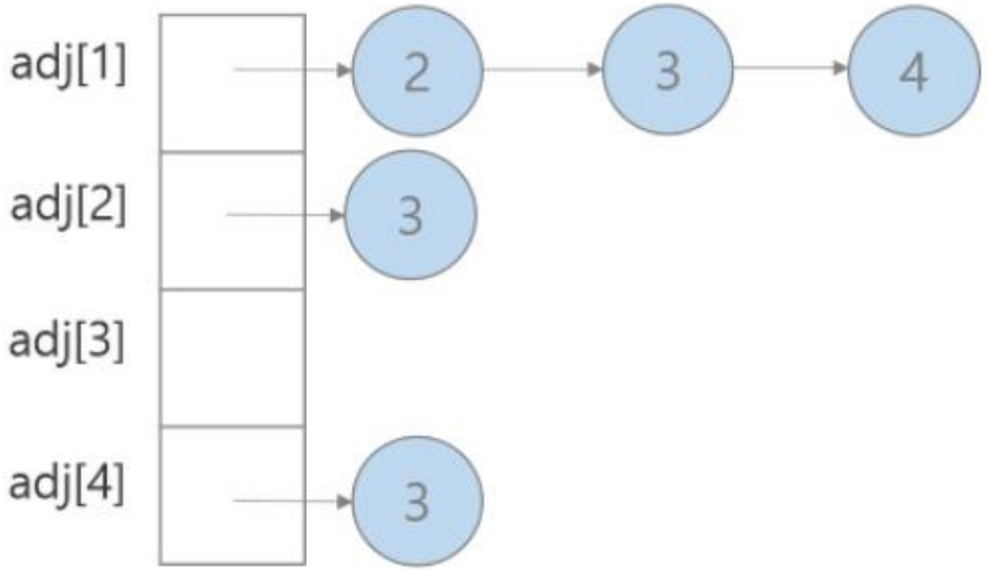
그래프의 표현 : 인접 행렬



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

그래프의 표현 : 인접 리스트





그래프의 탐색 기법

ARGOS

- 크게 **BFS탐색, DFS탐색**
- 이 외에도 Dijkstra, Floyd Washall 등등

- BFS, DFS 는 모든 정점을 한 번만 방문한다는 같은 목표를 가짐
  → 그 과정에서 탐색하는 방식의 차이가 있음

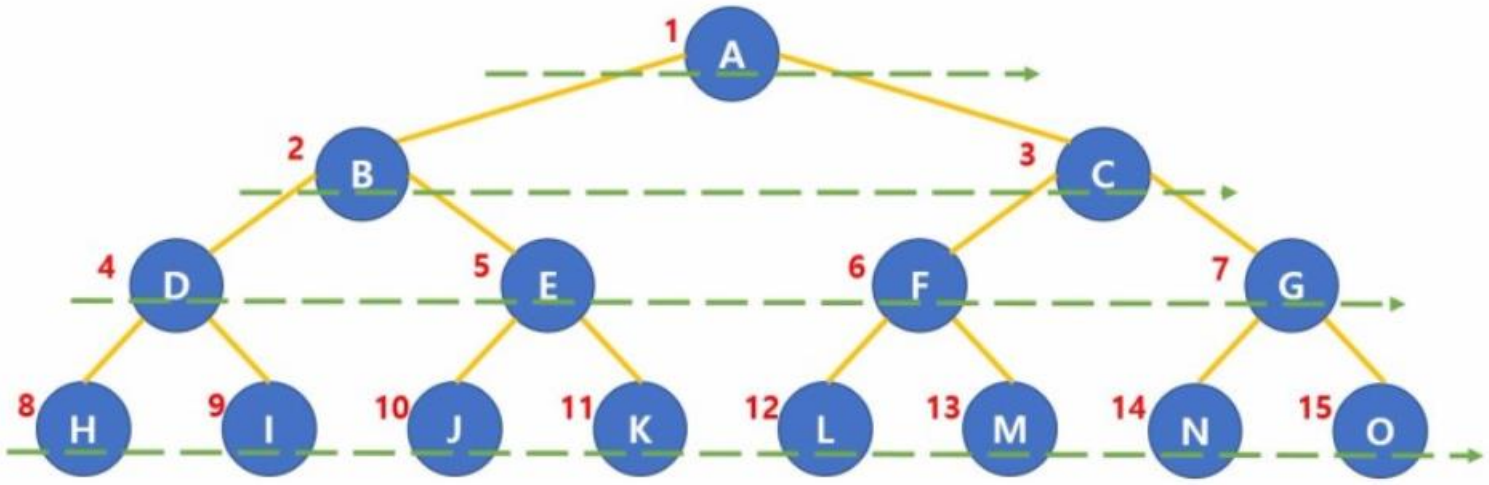
BFS 탐색

- Breadth First Search
- 예시

ARGOS

Breadth First Search

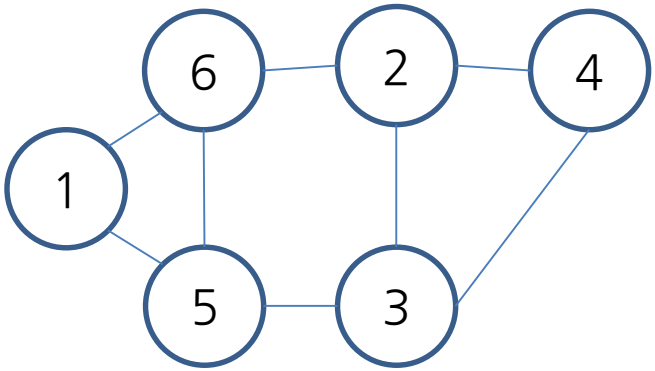
- **BFS (Breadth First Search ; 너비 우선 탐색)**
- 현재 정점과 연결된 정점들에 대해 우선적으로 탐색하는 방법



BFS 탐색 순서

http://blog.naver.com/occidere

Breadth First Search 예제 (1)



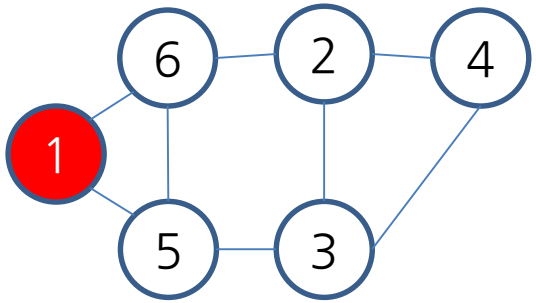
- 위 그래프를 1부터 BFS탐색 방법으로 탐색해보기
- 현재 위치에 인접한 모든 위치의 노드를 방문하고, 그 이웃 노드들의 또 다른 이웃 노드들을 방문하는 것은 그 다음에 진행하는 식

Breadth First Search 예제 (2)

현재 위치 이미 방문한 정점 방문 가능

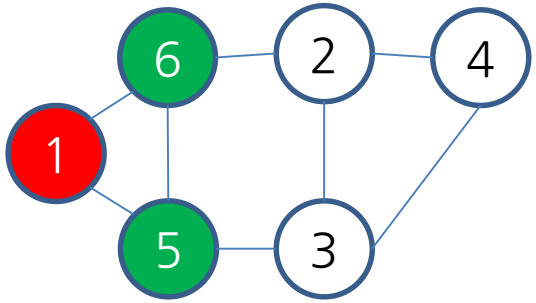
1.

순서						
큐	1					



2.

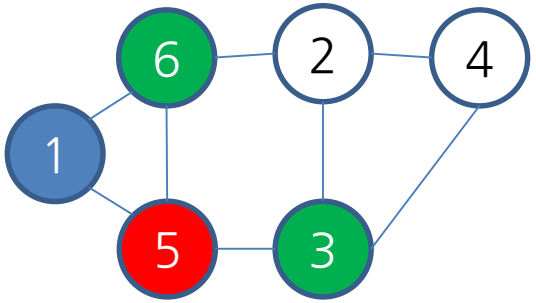
순서	1					
큐	5	6				



Breadth First Search 예제 (3)

3.

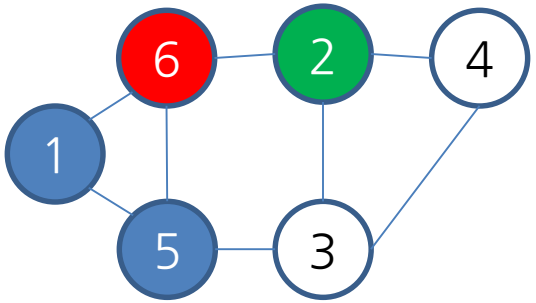
순서	1	5				
큐	6	3				



(여기서 6은 기존에 큐에 값이 있으므로 중복되지 않게 조심해야함)

4.

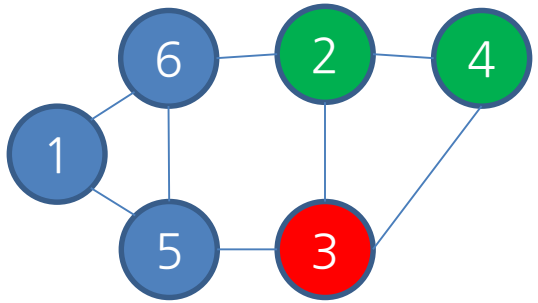
순서	1	5	6			
큐	3	2				



Breadth First Search 예제 (4)

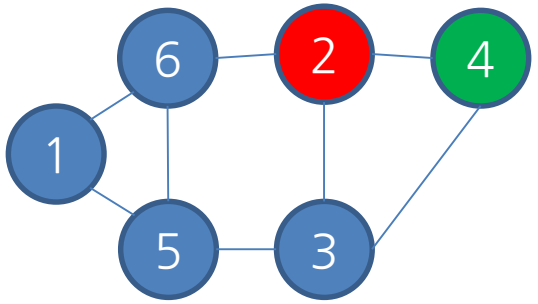
4.

순서	1	5	6	3		
큐	2	4				



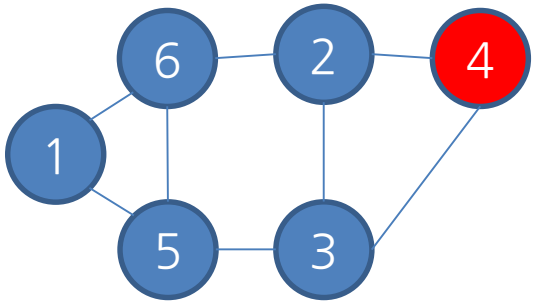
5.

순서	1	5	6	3	2	
큐	4					



6.

순서	1	5	6	3	2	4
큐						



Breadth First Search

- 현재 정점과 연결된 정점들에 대해 우선적으로 넓게 탐색하는 방식
- 최종적으로 큐가 비어 있으면 탐색을 그만둠
- BFS를 가장 효과족으로 구현하는 방법은 큐를 이용하여 순환적 형태로 구현

- 인접행렬 또는 인접리스트를 통해 BFS를 구현하여 사용하면 됨

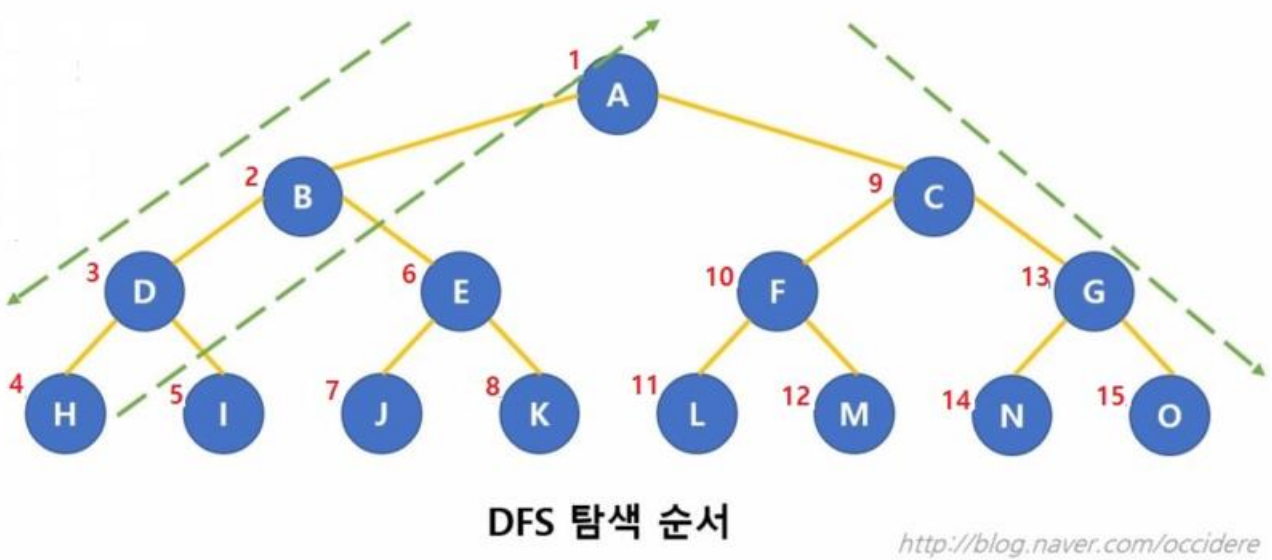
DFS 탐색

- Depth First Search
- 예시

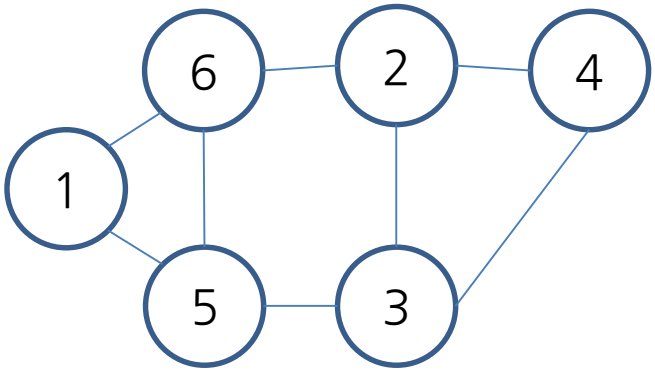
ARGOS

Depth First Search

- **DFS (Depth First Search; 깊이 우선 탐색)**
- 현재 정점에서 연결된 정점을 하나 골라 최대한 내려가며 탐색하는 방법



Depth First Search 예시 (1)



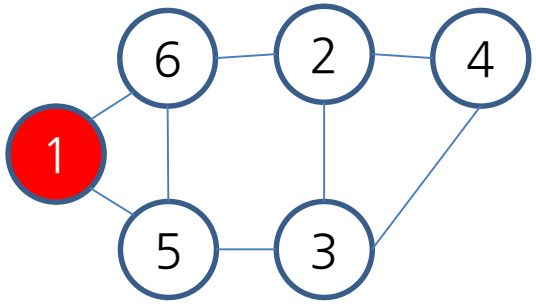
- 위 그래프를 1부터 DFS탐색 방법으로 탐색해보기
- 내가 지나간 곳을 계속 추적해야 하기 때문에 스택이 필요

Depth First Search 예시 (2)

현재 위치 이미 방문한 정점

1.

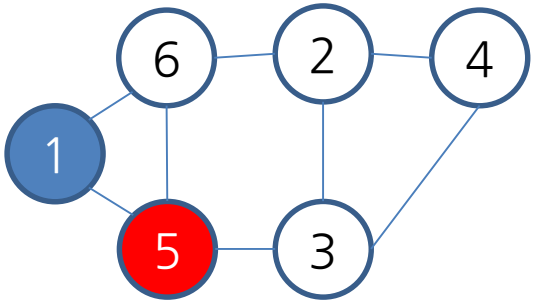
순서						
스택	1					



2.

순서	1					
스택	6	5				

순서	1	5				
스택	6					



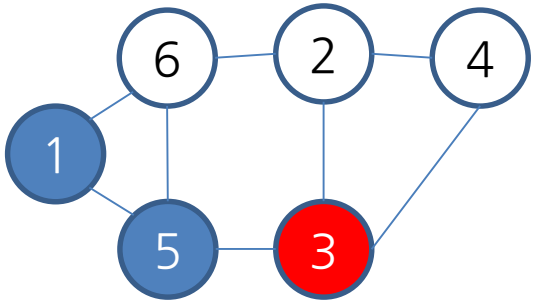
1의 자식노드인 6과 5가 stack에 push되고, 5를 방문하게 되면서 pop된다.

Depth First Search 예시 (3)

3.

순서	1	5				
스택	6	3				

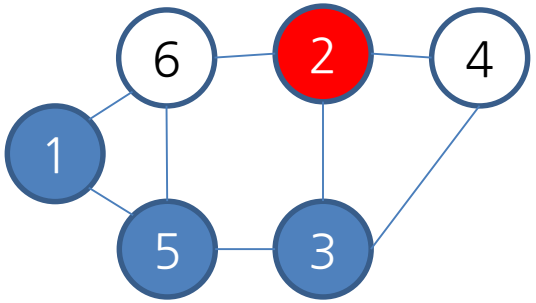
순서	1	5	3			
스택	6					



4.

순서	1	5	3			
스택	6	4	2			

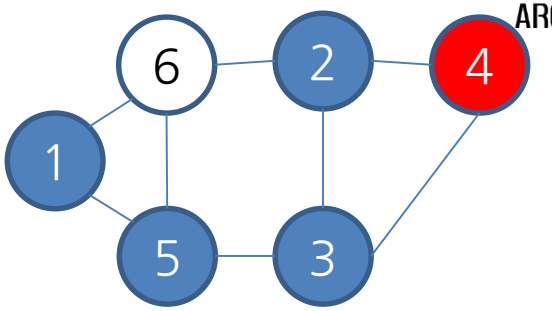
순서	1	5	3	2		
스택	6	4				



Depth First Search 예시 (3)

5.

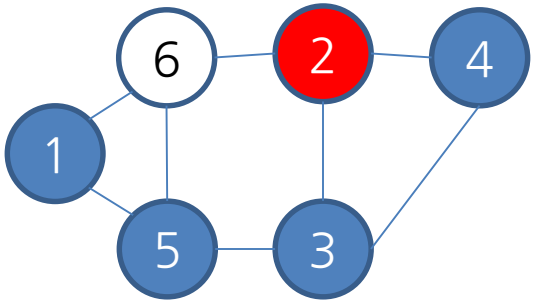
순서	1	5	3	2		
스택	6	4				



여기서 2의 자식노드로 stack에 push할 노드는 없으니 그냥 방문할 다음 노드에 방문하고 pop하게 된다.

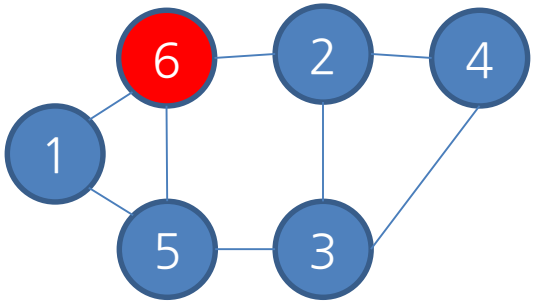
6.

순서	1	5	3	2	4	
스택	6					



7.

순서	1	5	3	2	4	6
스택						



Depth First Search (2)

- 최종적으로 stack이 비어 있으면 탐색을 그만둠
- DFS탐색 원리를 응용한 알고리즘이 '백 트래킹(역추적)'
- **스택(stack)**을 이용해서 구현하거나 **재귀호출**을 사용할 수 있음
  - 지금까지 거쳐온 정점들을 모두 저장하기 위해 stack 사용
- 옆으로 넓은 그래프에 대해서 준수한 성능을 보이나
- 아래로 깊은 그래프의 경우 좋은 성능을 기대하기 힘듬

- 인접행렬 또는 인접리스트를 통해 DFS를 구현하여 사용하면 됨

Q & A

Thank You for Listening

Created By 오하늘

그 외의 탐색 기법 (최단경로 구하기)

- 최단 경로 구하는 문제
- Dijkstra Algorith
- Floyd-Warshall Algorithm

ARGOS

Dijkastra Algorithm

- 하나의 정점에서 다른 모든 정점들의 최단 경로를 구함
- 간선들은 **모두 양의 간선들을** 가져야함
- 궁금하신 분들께서는 구글님에게 물어보세요 ^^

Floyd-Warshall Algorithm

- 모든 최단 경로를 구하는 문제
- Dijkstra 와 달리 음의 가중치도 사용할 수 있음
- 모든 정점에 대한 경로를 계산하므로 거리를 저장할 자료구조는 2차원 배열
- 기본 개념 : 특정 경로 안에 무수히 많은 경로가 있을 때, 중간 정점들이 각각 최단이 된다면 이를 모두 이은 경로 또한 최단이 됨. (optimal substructure)

Floyd-Warshall Algorithm 원리 (1)

- 두 정점 v_a, v_b의 최단 경로가 p = 〈 v1, v_a, …., v_z , v2〉 일 때
- 최단 경로의 중간 정점 p의 양 끝 정점 v1과 v2를 제외한 {v_a, … v_z} 가 됨

→ 임의의 한 정점에 대해서 이러한 중간 정점 집합의 한 원소가 되는지 안되는지 확인하는 방식

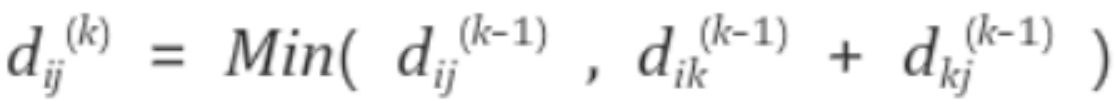
Floyd-Warshall Algorithm 원리 (2)

- v_i →v_j 의 최단 경로의 중간 정점 집합을 { v1, v2, ..., v(k-1) } 라고 하면,
- 어떤 새로운 정점 v_k가 중간 정점 집합에 포함돼서 더 비용이 적은 최단 경로를 만드느냐, 아니면 기존의 중간 정점 집합을 유지시키는 것이 최단경로인가를 판단!

$$l_{ij}^{(m)} = Min\left(l_{ij}^{(m-1)} , \underset{1 \le k \le n}{Min} \left\{ l_{ik}^{(m-1)} + w_{kj} \right\} \right)$$

[최단경로와 행렬곱의 재귀적 해법]

Floyd-Warshall Algorithm 원리 (3)



$$d_{ij}^{(k)} = Min(d_{ij}^{(k-1)} , d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

- 각 인접행렬 원소(가증한 중간경로들이 됨)에 대해서 i->j의 최단경로에 포함할 것인지 아닌지
- d_ij(k)는 k번째 (1, … , n)정점을 포함시킬지 말지 고민 전의 최단경로

참고한 사이트

ARGOS

- 그래프 https://m.blog.naver.com/occidere/220923695595
- 인접 행렬과 인접 리스트 https://www.leafcats.com/108
- BFS 탐색 기법 https://manducku.tistory.com/24?category=683258
- DFS 탐색 기법 https://manducku.tistory.com/23
- 플로이드 워셜 알고리즘 https://victorydntmd.tistory.com/108