그래프 탐색 알고리즘

유튜브 주소 : https://youtu.be/uLQvUXHFRz0

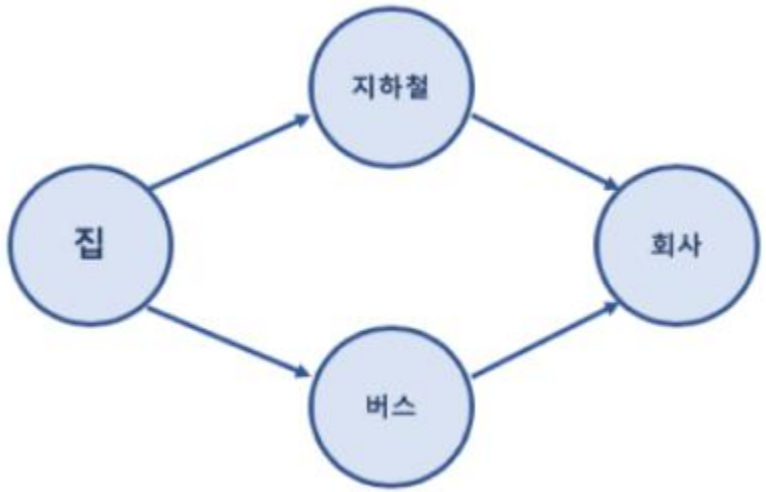
그래프 개요

DFS(깊이 우선 탐색)

BFS(너비 우선 탐색)

그래프 개요

- 그래프
  - 실제 세계의 현상이나 사물을 표현하기 위해 사용
  - 정점(Vertex) or 노드(Node) and 간선(Edge)으로 표현



그래프 개요

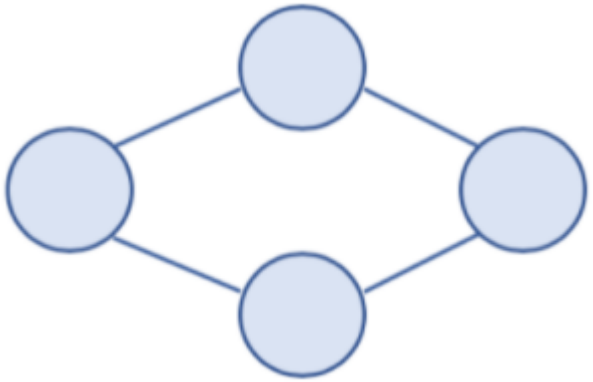
- 그래프와 트리의 차이
  - 트리는 그래프의 한 종류

	그래프	트리
방향성	방향 그래프, 무방향 그래프 모두 존재	방향 그래프만 존재
사이클	사이클 존재	사이클 존재 X
루트 노드	루트 노드 존재 X	루트 노드 존재
부모/자식 관계	부모 자식 존재 X	부모 자식 관계 존재

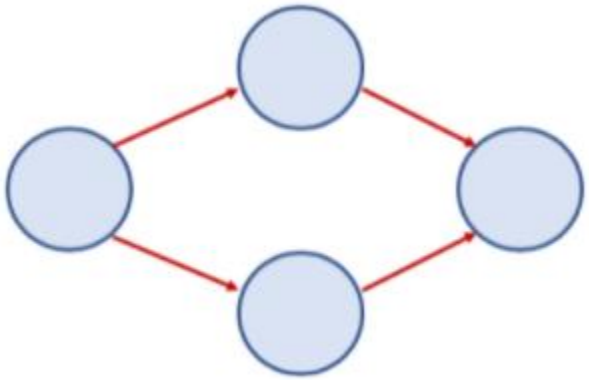
그래프 개요

- 그래프 종류

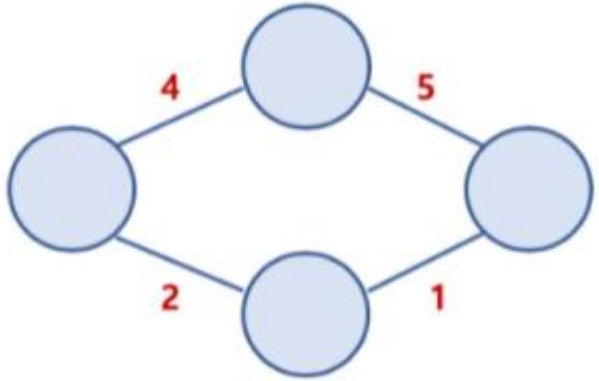
무방향 그래프



방향 그래프



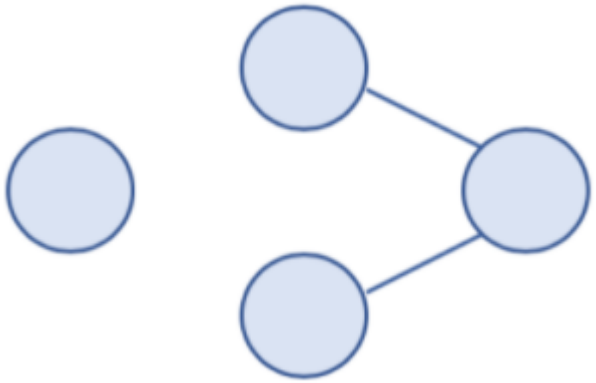
가중치 그래프



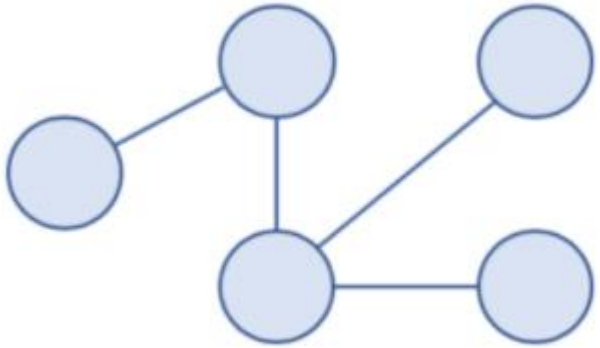
그래프 개요

- 그래프 종류

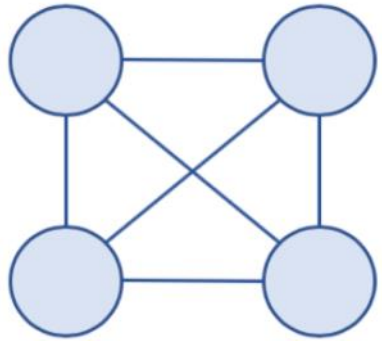
비연결 그래프



비순환 그래프



완전 그래프

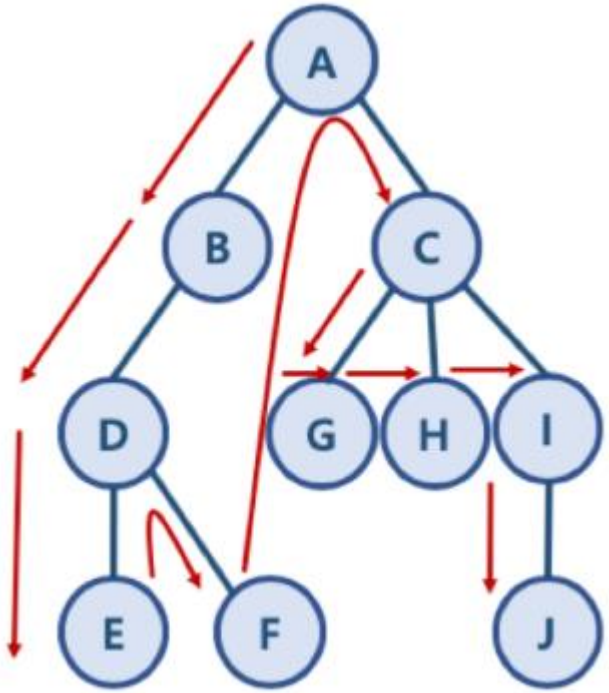


그래프 개요

- 그래프 탐색
  - 그래프의 가장 기본적인 연산
  - DFS(깊이 우선 탐색), BFS(너비 우선 탐색)
  - 하나의 노드에서 시작하여 차례대로 모든 노드를 한 번씩 방문

DFS(깊이 우선 탐색)

- 깊이 우선 탐색(DFS : depth-first search)
  - 밑으로 내려가며 탐색
  - 큐와 스택을 이용하여 구현



DFS(깊이 우선 탐색)

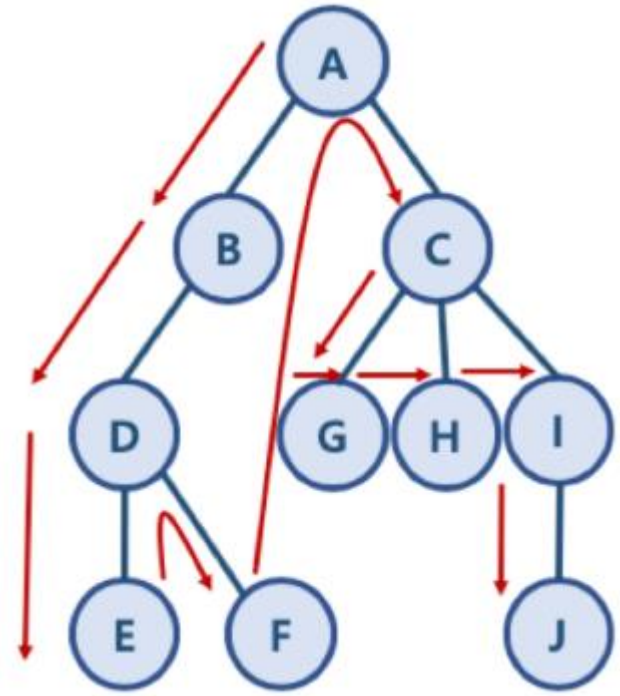
- 그래프 구현 코드

```python
graph = dict()

graph['A'] = ['B', 'C']
graph['B'] = ['A', 'D']
graph['C'] = ['A', 'G', 'H', 'I']
graph['D'] = ['B', 'E', 'F']
graph['E'] = ['D']
graph['F'] = ['D']
graph['G'] = ['C']
graph['H'] = ['C']
graph['I'] = ['C', 'J']
graph['J'] = ['I']

for i,j in graph.items():
    print(i,j)
```

```
C:\Users\minun\Desktop\Study\s
A ['B', 'C']
B ['A', 'D']
C ['A', 'G', 'H', 'I']
D ['B', 'E', 'F']
E ['D']
F ['D']
G ['C']
H ['C']
I ['C', 'J']
J ['I']
```

DFS(깊이 우선 탐색)

- DFS 구현 코드

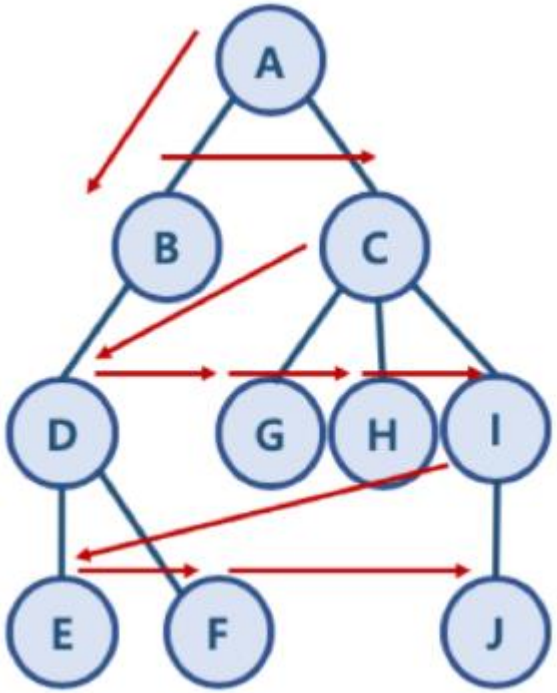



```python
def dfs(graph, start_node):
    visited, need_visit = list(), list()
    need_visit.append(start_node)

    while need_visit:
        node = need_visit.pop()
        if node not in visited:
            visited.append(node)
            need_visit.extend(graph[node])

    return visited

print(dfs(graph,'A'))
```

```
['A', 'C', 'I', 'J', 'H', 'G', 'B', 'D', 'F', 'E']
```

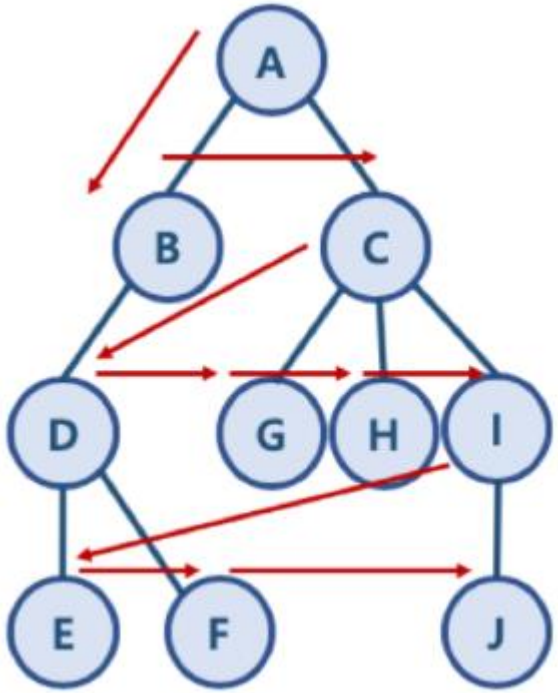
BFS(너비 우선 탐색)

- 너비 우선 탐색(BFS : breadth-first search)
  - 한 단계씩 내려가며 탐색
  - 큐 2개를 이용해 구현



BFS(너비 우선 탐색)

- BFS 구현 코드




```python
def bfs(graph, start_node):
    visited = list()
    need_visit = list()

    need_visit.append(start_node)

    while need_visit:
        node = need_visit.pop(0)
        if node not in visited:
            visited.append(node)
            need_visit.extend(graph[node])

    return visited

print(bfs(graph,'A'))
```

```
['A', 'B', 'C', 'D', 'G', 'H', 'I', 'E', 'F', 'J']
```

Q & A