


Graph

# Index	6
📅 CreatedAt	@September 28, 2022
👤 Person	A Ally Hyeseong Kim
⚙️ Status	TODO
☰ Tags	Graph Java Python
📅 UpdatedAt	@September 28, 2022

References

파이썬 알고리즘 인터뷰

2021 세종도서 학술부문 선정작. 현업과 실무에 유용한 주요 알고리즘 이론을 깊숙이 이해하고, 파이썬의 핵심 기능과 문법 까지 상세하게 이해할 수 있는 취업용 코딩 테스트를 위한 완벽

 <https://www.aladin.co.kr/shop/wproduct.aspx?ItemId=245495826>



References

- [1. Graph](#)
- [2. 오일러 경로](#)
- [3. 해밀턴 경로](#)
- [4. 그래프 순회](#)
- [5. Backtracking](#)

1. Graph

Graph Theory 에서 **Graph** 는 객체의 일부 쌍들이 연관되어 있는 객체 집합 구조이다.

2. 오일러 경로

오일러의 정리 는 모든 정점이 짝수 개의 차수(정점과 연결된 간선의 수)를 가지면 모든 다리를 한 번씩만 건너서 도달할 수 있다

| 는 정리이다.

- 오일러 경로: 모든 간선을 한 번씩 방문하는 유한 그래프

3. 해밀턴 경로

| **해밀턴 경로**는 각 정점을 한 번씩 방문하는 무향 혹은 유향 그래프 경로이다.

- 해밀턴 순환: 원래의 출발점으로 돌아오는 해밀턴 경로
 - Travelling Salesman Problem: 최단 거리의 해밀턴 순환을 찾는 문제

4. 그래프 순회

| **그래프 순회**는 그래프 탐색이라고도 하며 그래프의 각 정점을 방문하는 과정이다.

- Adjacency Matrix, Adjacency List로 구현할 수 있다.

4.1. Depth-First Search (DFS)

- 일반적으로 스택으로 구현하며, 재귀를 이용하면 더 간단하게 구현할 수 있다.
- 백트래킹을 사용할 수 있다.

4.1.1. 재귀로 구현

```
def recursive_dfs(v, discovered=[]):
    discovered.append(v)
    for w in graph[v]:
        if w not in discovered:
            discovered = recursive_dfs(w, discovered)
    return discovered
```

4.1.2. 스택으로 구현

```
def iterative_dfs(start_v):
    discovered = []
    stack = [start_v]
    while stack:
        v = stack.pop()
```

```

    if v not in discovered:
        discovered.append(v)
        for w in graph[v]:
            stack.append(w)
    return discovered

```

4.2. Breadth-First Search (BFS)

- 주로 큐로 구현한다.
- 그래프의 최단 경로를 구하는 문제 등에 사용된다.

4.2.1. Queue

```

def iterative_bfs(start_v):
    discovered = [start_v]
    queue = [start_v]
    while queue:
        v = queue.pop(0)
        for w in graph[v]:
            if w not in discovered:
                discovered.append(w)
                queue.append(w)
    return discovered

```

5. Backtracking

Backtracking 은 해결책에 대한 후보를 구축해 나아가다 가능성이 없다고 판단되는 즉시 후보를 포기(**Backtrack**)하여 정답을 찾아가는 알고리즘이다.

- 제약 충족 문제에 유용한 알고리즘이다.

5.1. 제약 충족 문제

제약 충족 문제 는 수많은 제약 조건을 충족하는 상태를 찾아내는 문제이다.