chapter6. 너비 우선 탐색

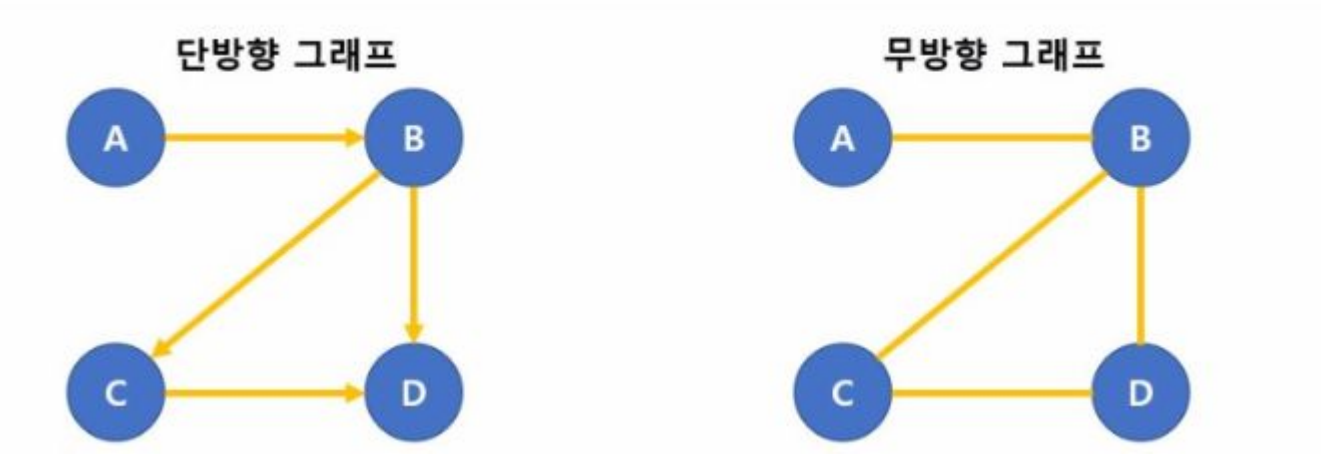
- Hello Coding 그림으로 개념을 이해하는 알고리즘 -

TABLE OF CONTENTS

1. 그래프

그래프란 연결의 집합을 모형화한 것입니다.

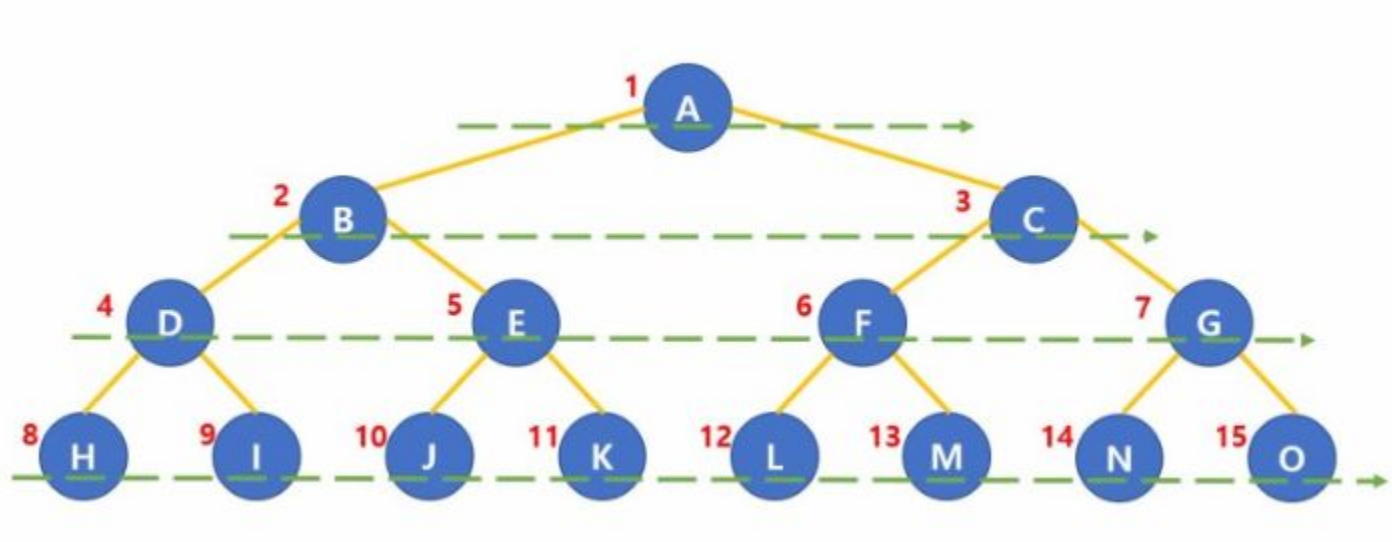
그래프는 정점(node)과 간선(edge)로 이루어집니다.



단방향 그래프 기준 A,B,C,D가 정점이고 화살표가 간선입니다. 바로 이어진 정점을 이웃이라고 합니다.
B는 A의 이웃, D는 B,C의 이웃, C는 B의 이웃입니다.

2. 너비 우선 탐색

너비 우선 탐색이란 현재 정점들과 이어진 정점들(즉, 이웃부터)에 대해 우선적으로 넓게 탐색하는 방식입니다.



두 정점 사이의 최단경로 혹은 임의의 경로를 찾고 싶을 때 너비우선탐색 알고리즘을 사용합니다.

3. 큐(대기열)

정류장에 사람들이 줄을 서서 기다리고 있습니다. 맨 처음 줄을 선 사람부터 버스에 올라탈것입니다.

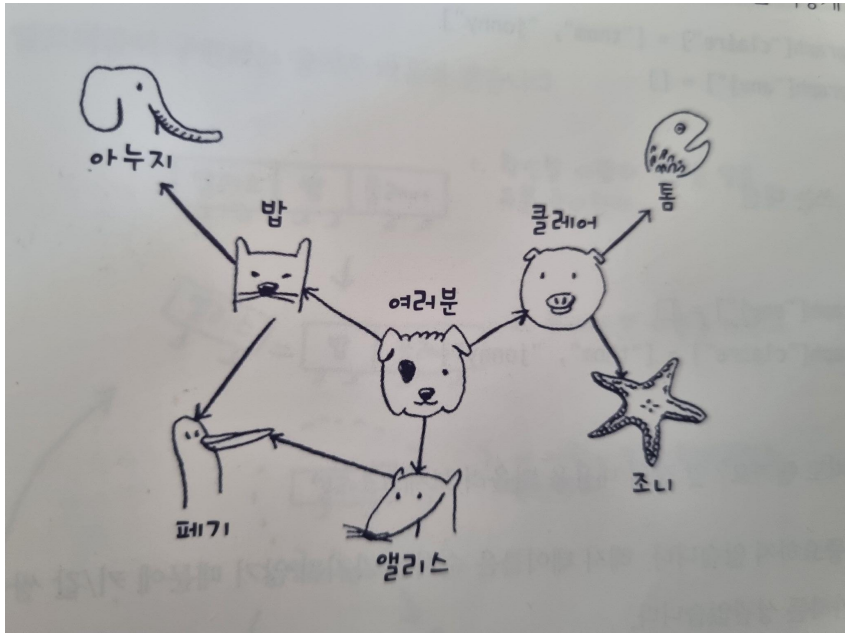
스택이랑 정 반대의 자료구조입니다. (*스택,큐 둘다 원소에 임의로 접근할 수 없다는 공통점이 있음.*)

큐를 **선입 선출 자료구조**라고도 하며 스택은 **후입 선출 자료구조**입니다.

큐는 리스트와 비슷하지만 차이점은 큐는 앞뒤에서 자료를 넣고 뺄 수 있습니다

너비우선탐색은 *큐 자료구조를 사용*합니다.

4. 망고 판매상 찾기



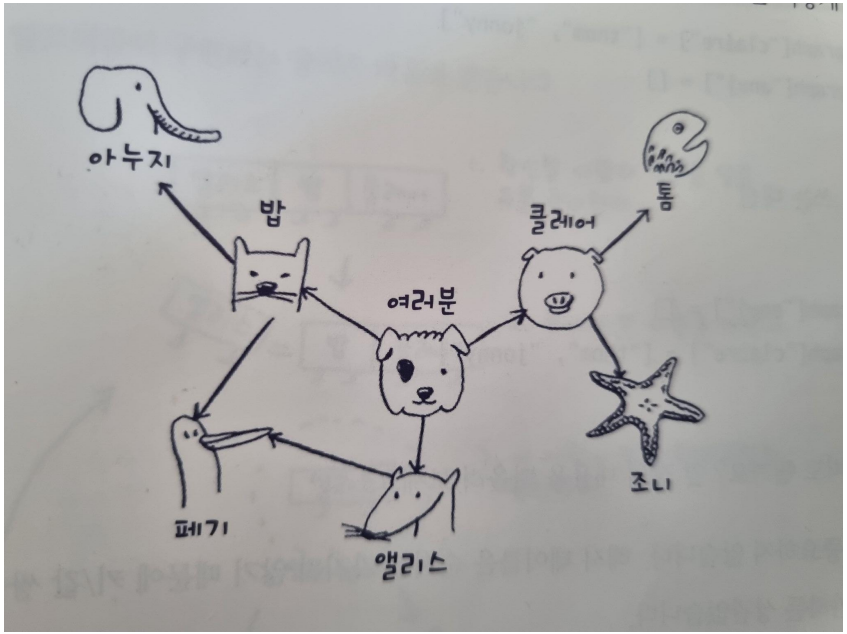
왼쪽 사진은 인물 관계도 입니다.

밥, 클레어, 앨리스는 여러분의 이웃입니다.

아누지는 밥의 이웃이고, 페기는 밥과 앨리스의 이웃입니다.

...

여러분의 지인중에서 망고판매상이 누군지 알아내려고 합니다.




```
graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []
```

왼쪽 관계를 표시하는 자료구조로 해시 테이블이 있습니다.

파이썬 코드는 위와 같습니다.

사람 이름이 “m”으로 끝나면 망고 판매상입니다. 바보같은 판단기준이지만 예로 든 것 뿐입니다.

```python
graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []

def person_is_seller(name):
  return name[-1] == "m"

from collections import deque
```

```python
def search(name):
  search_queue = deque()
  search_queue += graph[name]
  searched = []
  while search_queue:
    person = search_queue.popleft()
    if not person in searched:
      if person_is_seller(person):
        print(person + " is a mango seller!")
      else:
        search_queue += graph[person]
        searched.append(person)
  return False

search("you")
```

person_is_seller 함수는 망고판매상인지 확인하는 함수입니다.

```python
graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []

def person_is_seller(name):
    return name[-1] == "m"

from collections import deque
```

```python
def search(name):
    search_queue = deque()
    search_queue += graph[name]
    searched = []
    while search_queue:
        person = search_queue.popleft()
        if not person in searched:
            if person_is_seller(person):
                print(person + " is a mango seller!")
            else:
                search_queue += graph[person]
                searched.append(person)
    return False

search("you")
```

파이썬에서는 양방향 큐인 deque 함수를 사용할 수 있습니다.
search_queue = deque() 새 큐를 생성합니다.
search_queue += graph[“you”] 모든 이웃을 탐색 큐에 추가합니다.
searched = [] 이 배열은 이미 확인한 사람을 추적하기 위한 것입니다.

```python
graph = {}
graph["you"] = ["alice", "bob", "claire"]
graph["bob"] = ["anuj", "peggy"]
graph["alice"] = ["peggy"]
graph["claire"] = ["thom", "jonny"]
graph["anuj"] = []
graph["peggy"] = []
graph["thom"] = []
graph["jonny"] = []

def person_is_seller(name):
    return name[-1] == "m"

from collections import deque
```

```python
def search(name):
    search_queue = deque()
    search_queue += graph[name]
    searched = []
    while search_queue:
        person = search_queue.popleft()
        if not person in searched:
            if person_is_seller(person):
                print(person + " is a mango seller!")
            else:
                search_queue += graph[person]
                searched.append(person)
    return False

search("you")
```

while search_queue: 큐가 비어있지 않는 한 계속 실행합니다.
person = search_queue.popleft() 큐 가장 왼쪽원소를 끄집어내서 person에 할당.
if not person in searched: 이전에 확인하지 않은 사람만 확인
searched.append(person) 확인한 사람은 searched에 추가

5. 핵심 내용 정리

▶ 너비 우선 탐색은 A에서 B로 가는 경로가 있는지 알려줍니다.
▶ 만약 경로가 존재한다면 최단 경로도 찾아줍니다.
▶ 만약 X까지의 최단 경로를 찾는 문제가 있다면 그래프로 모형화 한 다음 너비 우선 탐색으로 문제를 풉니다.
▶ 방향 그래프는 화살표를 가지며 화살표 방향으로 관계를 가집니다.(rama → adit는 rama가 adit에게 돈을 빚지고 있다는 뜻입니다.)
▶ 무방향 그래프는 화살표가 없고 둘 간의 상호 관계를 나타냅니다. (rama - adit는 둘이 데이트 했다는 뜻)
▶큐는 선입선출입니다.
▶스택은 후입선출입니다.
▶탐색 목록에 추가된 순서대로 사람을 확인해야 합니다. 그래서 탐색 목록은 큐가 되어야 합니다. 그렇지 않으면 최단 경로는 구할 수 없습니다.
▶누군가를 확인한 다음에는 두 번 다시 확인하지 않도록 해야 합니다. 그렇지 않으면 무한 반복이 되어 버릴 수 있습니다.