# Day 12 - 완전 탐색 - (Breadth First Search)

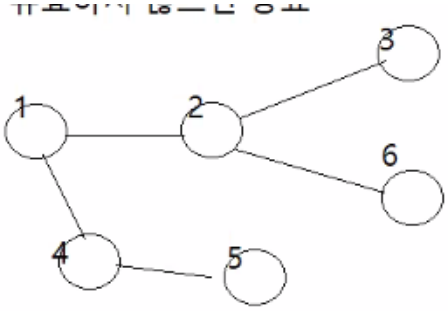
**완전 탐색**

1. **DFS**
2. **BFS**
    a. 가장 짧은 길 찾기 문제 혹은 최단 비용

**코드 작성 패턴**

1. 시작 부분을 큐에 삽입한다음 함수를 호출
2. 큐에서 데이터 뺀 다음, 해당 위치를 방문했다고 표시작업을 한다.
3. 다음 위치가 유효할 때 큐에 다음 위치를 넣은 후 2번 과정으로 돌아가고, 유효하지 않으면 종료

ex)




```
queue = []  --> 1번
while queue:
    <데이터> = queue.pop()
    <방문 처리 구문 코딩>
    queue.append()
```

**조금 더 깊게 들어가면:**

```
queue = []
queue.append(시작 위치)

while queue:
    현재 값 = queue.pop()
    배열에 현재 위치 방문 처리한다.
    방문에서 처리할 조건을 실행
    if 다음 위치를 방문할수 있는지:
        queue.push(다음 값)
```

**큐는 한쪽 방향으로만 데이터가 흘러간다.**

**단순 배열보다는 deque 라이브러리를 이용해서 큐를 만든다.**

**pop(), appendleft(), append(), popleft()**

- 방향을 통일해야 한다는 점에 유의

***탐색마다 무엇을 할 지, 어떤 경우에 다음 방문이 유효한지 판단 하는 것이 가장 중요.**

```
def solution(begin, target, words):
        q = deque()
        q.append([begin, 0])
        visited = [0] * len(words)   ⇒ 방문 않은곳은 0으로 초기화
```

**example)**

단어 begin 에서 target으로 변환하는 가장 짧은 순서를 찾는다.

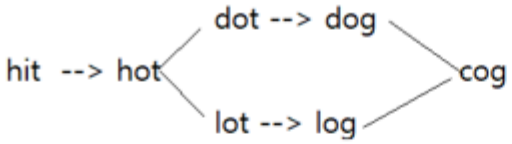
1. 한 번에 한 개의 알파벳만 바꿀 수 있으며 begin --------> target
2. 주어진 words에 있는 단어로만 변환할 수 있다.

ex.

| begin | target | words | 최소 변환 과정 | return |
|---|---|---|---|---|
| "hit" | "cog" | ["hot", "dot", "dog", "lot", "log", "cog"] | "hit" -> "hot" -> "dot" -> "dog" -> "cog" | 4 |
| "hit" | "cog" | ["hot", "dot", "dog", "lot", "log"] | x | 0 |

**Deque() 이용:**

begin 이 "hit" , target 이 :cog"




hit --> hot --> lot --> dog --> log --> dot --> cog

```
from collections import deque
def solution(begin, target, words):
        q = deque()
        q.append([begin, 0])
        visited = [0] * len(words) => 방문않은 곳은 0으로 초기화


while  q:
    word, count = q.popleft()
    if word == target :
        return count

    for i in range(len(words)):
        if not visited[i]:
            if _____________________________________
                q.append([words[i], count + 1)
                visited[i] = 1

        return 0
```

**파이썬 코드만으로 작성:**

```
def soultion(begin, target, words):
    q = deque()
    visited = {}

    for word in words:
        if is_valid(begin, word):
            q.append(word)
            visited[word] = 1

    while len(q) > 0:
            node = q.popleft()
            if node == target:
                return visited[node]
            for word in words:
                if word in visited or nor is_valid(node, word):
                        continue
                q.append(word)
                visited[word] = visited[node] + 1
    return 0

if __name__ == '_main__':
    begin = 'hit'
    target = 'cog'
    words = ['hot', 'dot','dof',.....]
    solutions = solution(begin, target, words)
    print(solutions)
```

**BFS 예시 문제:**

https://colab.research.google.com/drive/1-uTs5H0vyUE2ZMiC5FFbLNtwd50ghmBh#scrollTo=9QRKR6DvKYsH