

Algorithm 4월 3주차

BFS

Table of content

- 단어 변환 – BFS
- 보석쇼핑 – Hash, two pointer

BFS

[프로그래머스] 단어 변환

- 단어 변환에서 한 단계에 글자 하나만 달라야 변환에 성공한다.
- Words에 변환 단계에 징검다리로 사용할 수 있는 단어들이 주어짐
- Begin 단어에서 target 단어로 갈 때까지 필요한 단어 변환 횟수를 반환하는 문제

두 개의 단어 begin, target과 단어의 집합 words가 있습니다. 아래와 같은 규칙을 이용하여 begin에서 target으로 변환하는 가장 짧은 변환 과정을 찾으려고 합니다.

1. 한 번에 한 개의 알파벳만 바꿀 수 있습니다.
2. words에 있는 단어로만 변환할 수 있습니다.

예를 들어 begin이 "hit", target가 "cog", words가 ["hot","dot","dog","lot","log","cog"]라면 "hit" -> "hot" -> "dot" -> "dog" -> "cog"와 같이 4단계를 거쳐 변환할 수 있습니다.

두 개의 단어 begin, target과 단어의 집합 words가 매개변수로 주어질 때, 최소 몇 단계의 과정을 거쳐 begin을 target으로 변환할 수 있는지 return 하도록 solution 함수를 작성해주세요.

begin	target	words	return
"hit"	"cog"	["hot", "dot", "dog", "lot", "log", "cog"]	4
"hit"	"cog"	["hot", "dot", "dog", "lot", "log"]	0

풀이법 - BFS

- DFS로 시도했으나 실패.
- 최소 횟수를 구하는 것이므로 BFS(큐 자료구조)를 사용하는 것이 효율적
- 이미 사용한 단어는 사용하지 못하도록 visited를 사용한다.
- 매칭은 반복문을 활용해서 손쉽게 카운트하여 비교

```
visited = [ 0 for i in range(len(words))]  
q = deque()  
q.append([begin, 0])  
  
while q:  
    last, cnt = q.popleft()  
    if last == target:  
        return cnt  
    for j in range(len(words)):  
        diff = 0  
        if visited[j]:  
            continue  
        now = list(words[j])  
        prev = list(last)  
        for i in range(len(now)):  
            if prev[i] != now[i]:  
                diff += 1  
        if diff == 1:  
            visited[j] = 1  
            q.append([words[j], cnt + 1])
```

Hash, Two pointer

[프로그래머스] 보석쇼핑

- 보석쇼핑을 하는데 모든 종류의 보석을 담아야 한다.
- 시작 인덱스에서 끝 인덱스까지 모든 보석을 담는다. (중간에 빼먹을 수 없음)
- 가장 짧은 길이로 모든 보석을 담아야 한다.
- 그 때의 시작점과 끝점을 반환

1	2	3	4	5	6	7	8
DIA	RUBY	RUBY	DIA	DIA	EMERALD	SAPPHIRE	DIA

gems	result
["DIA", "RUBY", "RUBY", "DIA", "DIA", "EMERALD", "SAPPHIRE", "DIA"]	[3, 7]
["AA", "AB", "AC", "AA", "AC"]	[1, 3]
["XYZ", "XYZ", "XYZ"]	[1, 1]
["ZZZ", "YYY", "NNNN", "YYY", "BBB"]	[1, 5]

Sorting을 사용하면 시간초과

- Sort를 하면 시간초과가 발생한다.

Sorting을 사용하면 시간초과

- Sort를 쓰지 않기 위해 딕셔너리를 사용
- Answer를 양쪽 끝지점으로 지정하고 $\text{right} - \text{left}$ 와 비교하여 계속 업데이트한다.

```
def solution(gems):
    answer = [0, len(gems)]
    kind = set(gems)
    dic = {gems[0]:1}
    left, right = 0, 0

    while right < len(gems):
        if len(dic) < len(kind):
            right += 1
            if right == len(gems):
                break
            if gems[right] in dic:
                dic[gems[right]] += 1
            else:
                dic[gems[right]] = 1
        else:
            if (right - left) < answer[1] - answer[0]:
                answer = [left, right]
            if dic[gems[left]] == 1:
                del dic[gems[left]]
            else:
                dic[gems[left]] -= 1
            left += 1
```