Algorithm 4월 1주차

BFS

Table of content

부대복귀

[프로그래머스] 부대복귀

- 전체 노드 개수 n
- 출발지 번호 배열 sources
- 목적지 번호 destination
- 연결 간선 표시 배열 roads
- 각 source 에서 destination까지 도착하는 데 걸리는 시간을 반환

문제 설명

강철부대의 각 부대원이 여러 지역에 뿔뿔이 흩어져 특수 임무를 수행 중입니다. 지도에서 강철부대가 위치한 지역을 포함한 각 지역은 유일한 번호로 구분되며, 두 지역 간의 길을 통과하는 데 걸리는 시간은 모두 1로 동일합니다. 임무를 수행한 각 부대원은 지도 정보를 이용하여 최단시간에 부대로 복귀하고자 합니다. 다만 적군의 방해로 인해, 임무의 시작 때와 다르게 되돌아오는 경로가 없어져 복귀가 불가능한 부대원도 있을 수 있습니다.

강철부대가 위치한 지역을 포함한 총지역의 수 `n` , 두 지역을 왕복할 수 있는 길 정보를 담은 2차원 정수 배열 `roads` , 각 부대원이 위치한 서로 다른 지역들을 나타내는 정수 배열 `sources` , 강철부대의 지역 `destination` 이 주어졌을 때, 주어진 `sources` 의 원소 순서대로 강철부대로 복귀할 수 있는 최단시간을 담은 배열을 return하는 solution 함수를 완성해주세요. 복귀가 불가능한 경우 해당 부대원의 최단시간은 -1입니다.

n	roads	sources	destination	result
3	[[1, 2], [2, 3]]	[2, 3]	1	[1, 2]
5	[[1, 2], [1, 4], [2, 4], [2, 5], [4, 5]]	[1, 3, 5]	5	[2, -1, 0]

[프로그래머스] 풀이법 - BFS

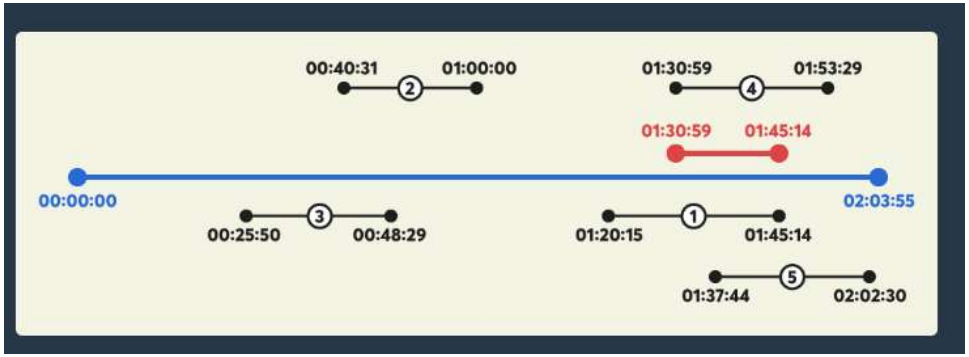
- 일단 최단 거리 소요 시간을 묻는 문제이므로 다익스트라 아니면 BFS로 푼다.
- 가중치가 없으므로 BFS로 풀면 됨 -> queue를 사용한다.
- Destination부터 출발해서 각 노드까지 도착하는 시간을 기록한다. (distance 배열에 기록)
- Source 부터 출발하여 기록하면 너무 많은 루트가 반복됨

```python
from collections import defaultdict, deque
def solution(n, roads, sources, destination):
    answer = []
    distance = [-1] * (n + 1)
    links = defaultdict(list)
    for i in range(len(roads)):
        links[roads[i][0]].append(roads[i][1])
        links[roads[i][1]].append(roads[i][0])
    q = deque()
    distance[destination] = 0
    q.append(destination)
    while q:
        now = q.popleft()
        for node in links[now]:
            if distance[node] == -1:
                distance[node] = distance[now] + 1
                q.append(node)
    for s in sources:
        answer.append(distance[s])
    return answer
```

부대복귀

[프로그래머스] 광고삽입

- 동영상에 광고를 넣는데 PIP 형태로 재생한다. (본 영상과 광고가 동시에 재생)
- 전체 동영상 재생시간 play_time과 광고 재생시간 adv_time이 주어짐
- 시청자들이 보는 구간이 기록됨 logs로 주어짐
- 가장 많은 시청자들에게 광고가 보여지게 하는 광고 시작 시간을 반환하는 문제



play_time	adv_time	logs	result
"02:03:55"	"00:14:15"	["01:20:15-01:45:14", "00:40:31-01:00:00", "00:25:50-00:48:29", "01:30:59-01:53:29", "01:37:44-02:02:30"]	"01:30:59"

[프로그래머스] 풀이법 – (1) 시간 변환

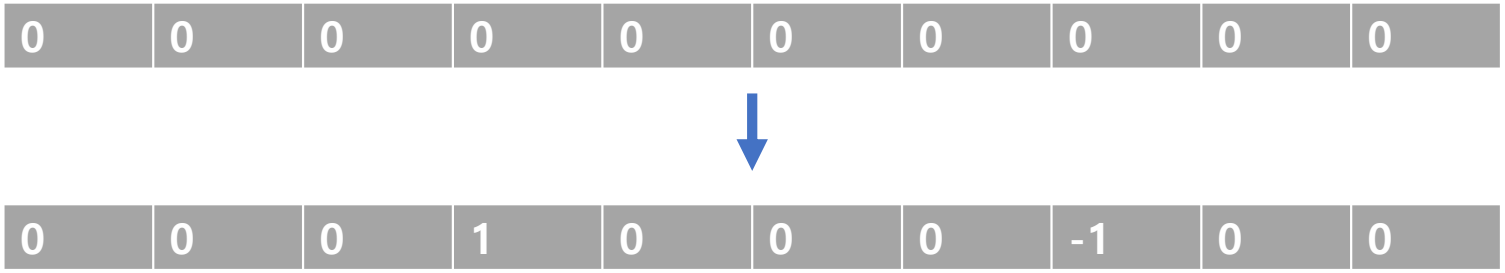
- 해당 문제 풀이에는 시간 변환 문법 + 부분합 알고리즘이 사용된다.
- 계산하기 쉽도록 시간을 정수값으로 변환해야 한다. (int())
- 정수값으로 변환해서 계산에 완료한 시간은 다시 답 양식에 맞추기 위해서 시간 형식으로 변환해야 한다. (str())

```python
def str_to_int(time):
    h, m, s = time.split(':')
    return int(h) * 3600 + int(m) * 60 + int(s)

def int_to_str(time):
    h = time // 3600
    h = '0' + str(h) if h < 10 else str(h)
    time %= 3600
    m = time // 60
    m = '0' + str(m) if m < 10 else str(m)
    time %= 60
    s = '0' + str(time) if time < 10 else str(time)
    return h + ":" + m + ":" + s
```

[프로그래머스] 풀이법 – (2) Prefix-sum

- 우선 일반적인 prefix-sum 문제처럼 주어진 logs의 시작점과 끝점을 하나의 배열에 반영해서 작성한다.
- 시작점은 +1로, 끝점은 -1로 기록한다.

```
def solution(play_time, adv_time, logs):
    play = str_to_int(play_time)
    adv = str_to_int(adv_time)

    # record start, end point to prepare prefix-sum
    view_count = [0 for _ in range(play + 1)]
    for log in logs:
        start_time, end_time = log.split("-")
        start = str_to_int(start_time)
        end = str_to_int(end_time)
        view_count[start] += 1
        view_count[end] -= 1
```



[프로그래머스] 풀이법 – (2) prefix-sum

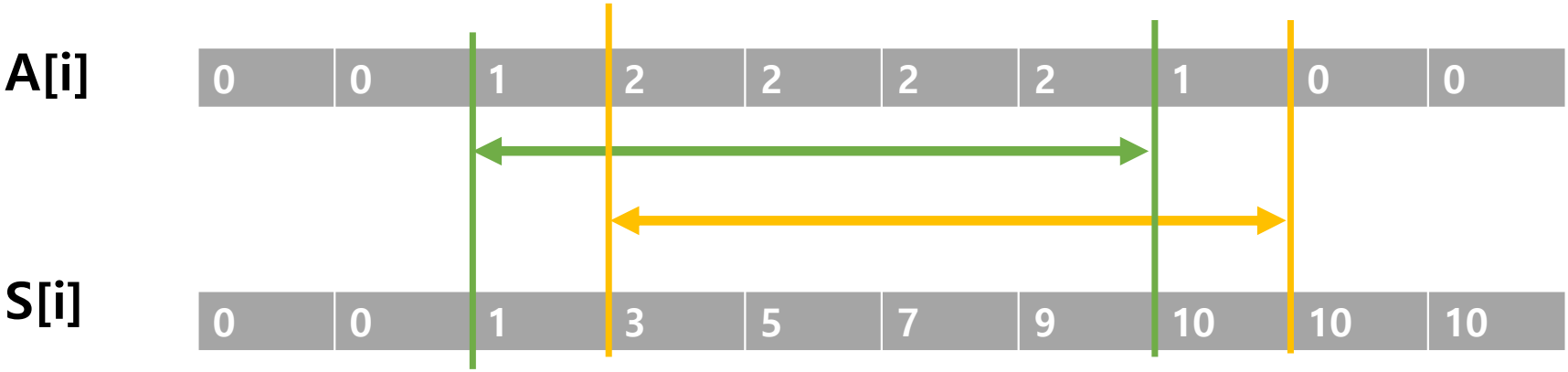
- 반복문을 두 번 돌려서 시청 누적 기록을 작성해야 한다
- 아래 수식을 사용한다고 보면 됨

$S[r] = A[1] + \ldots + A[r]$

$S[l-1] = A[1] + \ldots + A[l-1]$

입니다. 따라서, $S[r] - S[l-1] = A[l] + \cdots + A[r]$이 됩니다.

```python
# make prefix-sum
# A[i]: sum of view
for i in range(1, len(view_count)):
    view_count[i] += view_count[i - 1]

# S[i]: cumulative value of sum of view
for i in range(1, len(view_count)):
    view_count[i] += view_count[i - 1]
```



[프로그래머스] 풀이법 – (2) prefix-sum

- 굳이 누적 기록을 사용하는 이유: 한 방에 끝내기 위해
- 조회수를 기록한 A[i]를 사용하여 광고 시작점을 정하려면 조회수가 높은 구간의 시작점과 끝점을 정해서 검사하고 또 그것이 광고 시간만큼 긴지 체크해야 한다.

```
# S[r] - S[l] = A[l-r]: specific interval view
adv_start = 0
most_view = view_count[adv - 1]
for i in range(adv, play):
    if most_view < view_count[i] - view_count[i - adv]:
        most_view = view_count[i] - view_count[i - adv]
        adv_start = i - adv + 1

return int_to_str(adv_start)
```

- 하지만 누적 기록을 사용하면 두 지점의 S[i] 차이를 구하면 특정 구간의 조회수를 구할 수 있다.
- 특정 지점에서 S값에서 adv 앞당긴 지점에서 S값을 빼면 그 구간 조회수가 나오는데 애초에 adv를 앞당긴 지점이기 때문에 당연히 광고를 실을 수 있다. 이 구간 조회수를 Most_view로 기록하면 반복문 한 방에 처리할 수 있다.