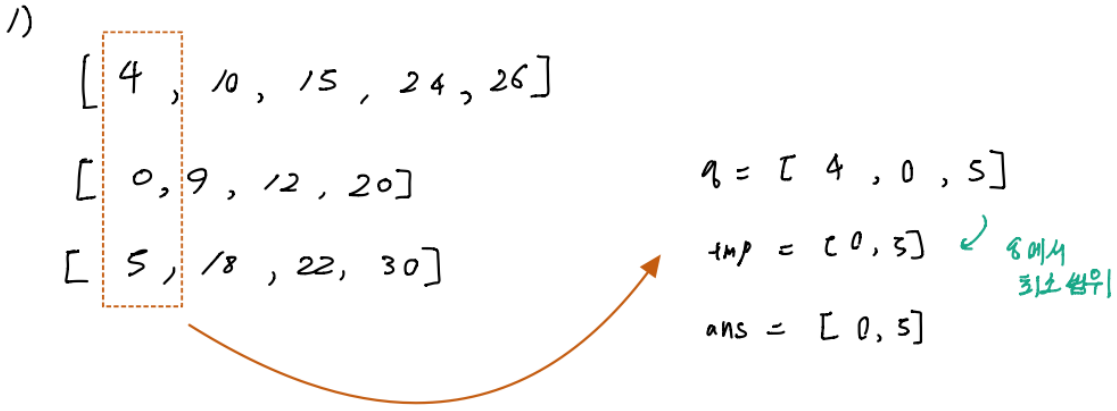
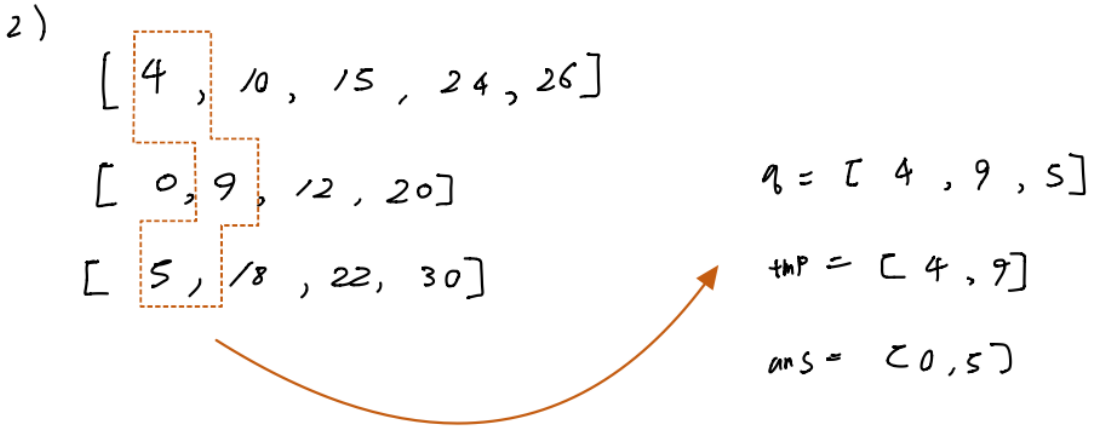
632. Smallest Range Covering Elements from K Lists

- 목표 : k개의 List의 원소를 적어도 1개 포함하는 최소 범위 찾기
- 사용하는 알고리즘 : 윈도우 슬라이딩(모든 원소를 봐야 하므로) + Heap
- 풀이 과정
- 1)

  1. 각 List 마다 최소의 값을 q에 넣어준다.
  2. 현재 q의 범위 값을 tmp로 저장한다.
  3. ans에 저장되어 있는 값이 없으므로 tmp를 ans로 저장한다.



- 2)
  1. q에서 가장 작은 값을 뺀다.
  2. 가장 작은 값이 있던 list에서 그 다음으로 작은 값을 q에 넣어준다
  3. 현재 q의 범위 값을 tmp로 저장한다.
  4. 기존 저장되어 있는 ans보다 범위가 작을경우에 ans로 저장한다.

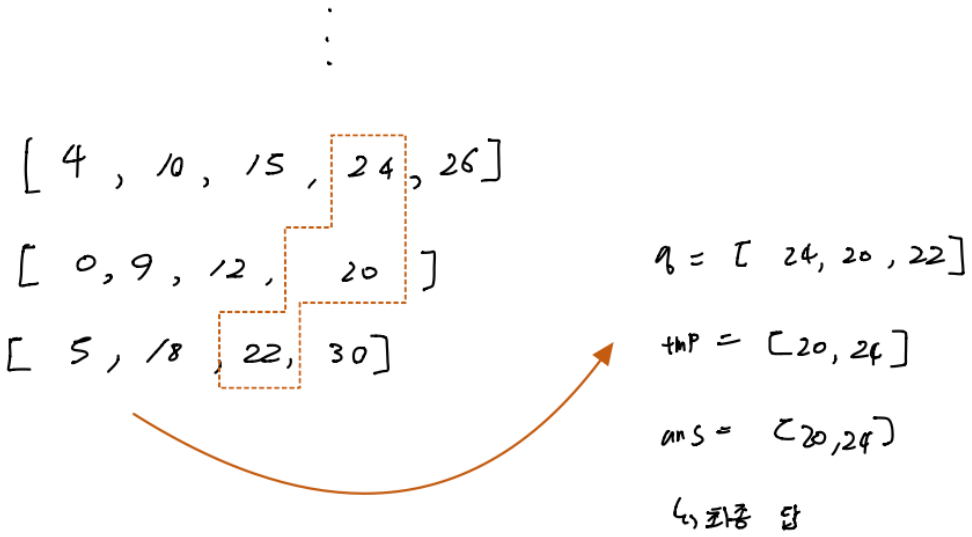


- 3)
  1. 위 같은 방법을 k개 리스트 중 마지막 원소가 나올 때 까지 반복한다.

     그 이유는 아래 예시를 보자

     q = [24, 20, 22] 에서 20은 두번 째 리스트의 마지막 원소이니 더 넣을 값이 없다

→ 그렇다면 22를 빼고 30을 넣어야하는데 q의 범위가 더 커지기 때문에 무의미하다



- 시간 복잡도

  O(KNlogK) - Heap을 사용 시

  1 ≤ K ≤ 3,500 & 1 ≤ N ≤ 50 (K : 리스트의 갯수, N : 리스트의 원소 갯수)

- Code

```
from heapq import heappush, heappop
class Solution:
    def smallestRange(self, nums: List[List[int]]) -> List[int]:
        q = []
        n = len(nums)
        max_num = -float('inf')
        for i in range(n):
            heappush(q, (nums[i][0], i, 0))
            max_num = max(max_num, nums[i][0])

        ans = (q[0][0], max_num)
        while True:
            num, nums_idx, idx = heappop(q)

            if idx == len(nums[nums_idx]) - 1:
                break
```

```
            next_num = nums[nums_idx][idx + 1]
            max_num = max(max_num, next_num)
            heappush(q, (next_num, nums_idx, idx + 1))

            if max_num - q[0][0] < ans[1] - ans[0]:
                ans = (q[0][0], max_num)
        return list(ans)
```