정렬

K번째 수

- array가 [1,5,2,6,3,7,4], i=2, j=5, k=3 이라면
  1. array를 i번째부터 j번째까지 자른다 → [5,2,6,3]
  2. 자른 배열을 정렬 → [2,3,5,6]
  3. 정렬한 배열에서 k번째 숫자를 뽑아 answer에 넣는다
- commands는 [i,j,k]들로 이루어진 배열

```
def solution(array, commands):
    answer = []

    for command in commands:
        tobesorted = array[command[0]-1:command[1]]
        tobesorted.sort()
        answer.append(tobesorted[command[2]-1])
    return answer
```

▼ Which sort does python use?

- Python’s default sort uses Tim Sort, which is a combination of both merge sort and insertion sort

⭐ 가장 큰 수 ⭐

- 주어진 수들을 이어붙여 만들수 있는 가장 큰 수 구하기

```
주어진 수 : [6,10,2]
만들 수 있는 가장 큰 수 : 6210

# 각 수는 1000을 넘지 않음
```

- 삽..질..

```
def swap(arr, idx1, idx2):
    temp = arr[idx1]
    arr[idx1] = arr[idx2]
    arr[idx2] = temp

def isLarge(str1, str2):
    for i in range(0, max(len(str1), len(str2))):
        j= i
        k=i
        if i >= len(str1):
            j = -1
            k = i
        elif i >= len(str2):
            j = i
            k = -1

        if int(str1[j]) > int(str2[k]):
            return True
        elif int(str1[j]) < int(str2[k]):
            return False
        elif int(str1[j]) == int(str2[k]):
            continue

def solution(numbers):
    answer = ''
    strarr = []
    for num in numbers:
        strarr.append(str(num))

    for i in range(1, len(strarr)):
        for j in range(i, 0, -1):
            if isLarge(strarr[j-1], strarr[j]):
                swap(strarr,j-1,j)
            else:
                break

    print(strarr)
    for i in range(len(strarr)-1, -1, -1):
        answer += strarr[i]
    return answer
```

- 정답
  - 4자리가 안되는 수에 대해서는 원래 수를 4자리까지 반복해서 붙인 후 비교하는 것이 핵심 (3→3333, 30 → 3030, 303 → 3033)
  - str상태에서 sort()
  - [0,0,0,0] 인 경우 “0000”이 아닌 “0”을 출력해야함

```
def solution(numbers):
    answer = ''
    temp = []
```

```
    sum = 0

    for num in numbers:
        extendedNum = (str(num)*4)[:4]
        temp.append((extendedNum, len(str(num))))

    temp.sort(reverse=True)
    for num in temp:
        sum += int(num[0])
        answer += num[0][:num[1]]

    if sum == 0:
        return '0'
    return answer
```

- [**979**, 67] → 97997
- [978, **97**] → 97978
- [977, **97**] → 97977
- [976, **97**] → 97976

H-Index

- 정렬없이 풀어도 통과되긴 함
  - 아마 $O(N^2)$???

```
def solution(citations):
    answer = 0
    crntcite = 1
    count = 0
    while True:
        for item in citations:
            if crntcite <= item:
                count += 1
        if count < crntcite:
            break
        count = 0
        crntcite += 1
    answer = crntcite-1
    return answer
```

- 계수정렬(의 변형) 이용 : $O(N)$

```
citations : [3,0,6,1,5]
count : [1,1,0,1,0,1,1]
# 일반 계수정렬과 다르게 count의 맨 뒤 인덱스부터 더함
```

```
sum : [5,4,3,3,2,2,1]

sum에서 index와 sum[index]가 달라지는 지점을 찾는다
```

```
def solution(citations):
    answer = 0
    count = [0]*(max(citations)+1)
    sum = [0]*(max(citations)+1)

    for citation in citations:
        count[citation] += 1

    sum[-1] = count[-1]
    for i in range(len(count)-2, -1, -1):
        sum[i] = sum[i+1] + count[i]

    h_idx = 0
    while h_idx < len(sum):
        if sum[h_idx] < h_idx:
            break
        h_idx += 1
    answer = h_idx - 1
    return answer
```