3. 수 정렬하기

카운팅 정렬 [1, 3, 2, 2, 4, 4, 3, 4, 4, 2, 4]

① 원소 값들의 개수를 저장하는 카운팅 배열 만들기 [1, 4, 2, 5 ···]

② 카운팅 배열에 직전 요소들의 값 더해주기 [1, 5, 7, 12 ···] ↳ -1

③ 새로운 배열에 개수에 맞추어 값 추가해주기 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 12-1

Try 1 실패

```
x = int(input())
num_list = []

for i in range(x):
    num = int(input())
    num_list.append(num)

count = [0 for i in range(max(num_list))]
for i in num_list:
    count[i-1] += 1
for i, v in enumerate(count):
    if i > 0:
        count[i] += count[i-1]

result = [0 for i in range(len(num_list))]
for num in num_list:
    a = count[num-1]
    result[a-1] = num
    count[num-1] -= 1

for i in result:
    print(i)
```

→ input()으로 받아서 list에 추가 [1, 3, 2, 2, 4, 4, 3, 4, 4, 2, 4]

→ 각 숫자의 개수를 담는 list

→ 개수추가 [1, 4, 2, 5 ···] → [1, 5, 7, 12 ···]

→ 개수를 누적으로 바꿈

→ input() 담긴 리스트 길이 만큼의 list 생성

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

a = count[i-1] = 1

result[0] = 1

count[0] = 0

⇒ 메모리 초과

∴ result = [1, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4]

Try 2 실패

⇒ input()으로 입력을 받을경우 메모리 초과 발생가능 → sys 이용

```
1 import sys
2 n = int(sys.stdin.readline())
3 num_list = []
4
5 for i in range(n):
6   num_list.append(int(sys.stdin.readline()))
```

→ \n (개행문자 포함) int() 형 변환시 제거

⇒ 메모리 초과

Try 3 성공 (구글링)

```python
import sys

n = int(sys.stdin.readline())
num_list = [0] * 10001
  ↳ 개수를 담는 리스트
for _ in range(n):
    num_list[int(sys.stdin.readline())] += 1

for i in range(10001):
    if num_list[i] != 0:
        for j in range(num_list[i]):
            print(i)
```

입력받는 숫자가 10,000보다 작거나 같은 자연수

★ 인덱스가 숫자 역할 ★

입력받은 값이 해당하는 인덱스에 (+1)

num_list의 0이 아닌 원소를 그 값만큼 index를 출력

입력 : 3 5 2 5 1

num_list = [0, 1, 1, 1, 0, 2, ···]

idx : 0, 1, 2, 3, 4, 5 ···

출력 : 1 2 3 5 5

⇒ for문에서 append()를 사용하면 메모리 재할당이 이뤄져 메모리를 효율적으로 사용하지 못한다고 함 ∴ append 없애기

⇒ Try 1은 리스트를 총 3개 사용 > Try 3에서는 idx 활용하여 사용하는 리스트를 총 1개로 끝냄

4. 통계학

```python
import sys
from collections import Counter

N = int(sys.stdin.readline())
num_list = []
for i in range(N):
    num_list.append(int(sys.stdin.readline()))
num_list.sort()

print(int(round(sum(num_list)/N,0)))
print(num_list[N//2])

count = Counter(num_list).most_common(2)
if len(count) ==1:
    print(count[0][0])
elif count[0][1] != count[1][1]:
    print(count[0][0])   최빈값 1개
else:
    print(count[1][0])   최빈값 2개

print(max(num_list)-min(num_list))
```

→ 평균

→ 중앙값

→ 최빈값

→ 범위

Counter : 값의 개수를 세서 반환 (개수)

ex) {1:2, 2:3, 3:4, 4:1}

.most_common(2)

⇒ 가장 빈도가 높은 2개 값 보여줌

{ 같은 값이 2개 이상이면 작은값부터

같은 값이 1개라면 2번째로 빈도 높은 값도 }

5. 단어정렬

```python
import sys
words = []
N = int(sys.stdin.readline())

for i in range(N):
    word = sys.stdin.readline().strip()
    if word  not in words:
        words.append(word)

words.sort(key = lambda x: (len(x), x))

for i in words:
    print(i)
```

① 길이순으로 ② 사전순으로

↳ \n 없애주는 역할

6. 좌표압축

Try 1 실패

```python
import sys

N = int(sys.stdin.readline())
locations = list(map(int, sys.stdin.readline().split()))

locations_set = list(set(locations))
loca_sorted = sorted(locations_set)

result = [0] * N
for i, v in enumerate(locations):
    result[i] = str(loca_sorted.index(v))

answer = " ".join(result)
print(answer)
```

중복된 값 제거

(set = {3, 6, 9 ··· }

순서 없음!

> 정렬 ⇒ 숫자의 위치 (index)가

1 숫자보다 작은 값들의 개수가 됨

locations = [1, 2, -2, 2, 5]

loca_sorted = [-2, 1, 2, 5]

result = [1, 2, 0, 2, 3]

⇒ 시간초과

Try 2 성공 (구글링)

```python
import sys

N = int(sys.stdin.readline())
locations = list(map(int, sys.stdin.readline().split()))
loca_sorted = sorted(list(set(locations)))
result = {loca_sorted[i]: i for i in range(len(loca_sorted))}

for i, v in enumerate(locations):
    print(result[v], end = " ")
```

시간 줄이기

① list.index(x) 대신 dict[key] 사용

{-2: 0, 1: 1, 2: 2, 5: 3} 개수

② join 대신 print(end = " ") 사용