탐욕법(Greedy)

체육복

```
n : 전체 학생 수
lost : 체육복을 도난당한 학생의 번호 리스트
reserve : 여벌 체육복을 갖고 있는 학생의 번호 리스트
answer : 체육복을 입을 수 있는 학생수의 최대값

* 체육복을 도난당한 학생은 여벌을 갖고 있는 학생에게 체육복을 빌린다
* 체육복은 인접한 번호의 학생에게만 빌려줄 수 있음 (3번 학생은 4번 또는 5번 학생에게서만 체육복을 빌릴 수 있음)
* 여벌 체육복을 가져온 학생이 체육복을 도난당했다면 다른 학생에게 빌려줄 수 없음
```

- 그리디 방식으로 lost의 가장 처음 학생부터 체육복을 빌려나감
  - 내 앞/뒷번호 학생이 모두 여벌을 가지고 있는 경우에는 앞 학생의 옷을 빌려야 최적으로 구할 수 있음
  - 그전에 먼저 lost랑 reserve를 sort해야함
    - lost = [4,2] , reserve = [3,5] 경우 4가 4의 앞번호 3을 먼저 가져가니까 2는 빌릴 사람이 없어짐 → lost = [2,4] reserve = [3,5] 로 정렬해놓고 시작해야 2가 3의 여벌을, 4가 5의 여벌 가져감
- lost랑 reverse에서 겹치는 애들은 제거하고 시작(도난당했는데 여벌이 있으면 그냥 여벌 입으면 됨)
  - for stud in reverse: 로 돌면서 stud가 lost에 있는 경우 reverse.remove()했는데 이러면 for문이 꼬임 → reserve_copy에다가 새로 저장

```
def solution(n, lost, reserve):
    answer = n - len(lost)
    lost.sort()
    reserve.sort()

    # 체육복 없는데 여벌도 없는애들 골라내기
    reserve_copy = []
    for stud in reserve:
        if stud in lost:
            lost.remove(stud)
            answer += 1
            continue
        reserve_copy.append(stud)

    for stud in lost:
```

```
        # 앞 번호 애한테 여벌이 있으면 앞번호 학생걸 빌린다
        if stud - 1 in reserve_copy:
            reserve_copy.remove(stud-1)
            answer += 1
        # 앞번호 애한테 여벌이 없는 경우에만 뒷번호 애 조사
        elif stud + 1 in reserve_copy:
            reserve_copy.remove(stud+1)
            answer += 1
    return answer
```

조이스틱

- 그리디 문제가 아님 (모든 테스트 케이스에서 최적합이 나오지 않음)
- 그리디로 하면 → 현재 위치에서 좌우를 살펴서 가장 가까이에 있는 바꿀 문자를 찾아감(반복)

```
def solution(name):
    answer = 0
    each_move = []
    for letter in name :
        move = ord(letter) - ord("A")
        if move < 14:
            each_move.append(move)
        else:
            each_move.append(26-move)

    crntidx = 0
    while True:
        answer += each_move[crntidx]
        each_move[crntidx] = 0
        if each_move.count(0) == len(each_move):
            break
        left = 1
        right = 1

        for i in range(1, len(each_move)):
            l_compareidx = crntidx - i
            if l_compareidx < 0 :
                l_compareidx = len(each_move) + l_compareidx

            if each_move[l_compareidx] == 0 :
                left += 1
            else:
                break

        for j in range(1, len(each_move)):
            r_compareidx = crntidx + j
            if r_compareidx >= len(each_move):
                r_compareidx = r_compareidx - len(each_move)

            if each_move[r_compareidx] == 0:
                right += 1
            else:
```

```
                break

        if left < right:
            answer += left
            crntidx = crntidx - left
            if crntidx < 0:
                crntidx = len(each_move) + crntidx
        else:
            answer += right
            crntidx = crntidx + right
            if crntidx >= len(each_move):
                crntidx = len(each_move) - crntidx

    return answer
```

큰 수 만들기

- 문제 : 문자열로 주어진 수 number에서 k개 수를 제거했을 때 얻을 수 있는 가장 큰 수를 리턴

```
# example
number = "1924" / k = 2    -----  return "94"
number = "1231234" / k = 3  ----  return "3234"
number = "4177252841" / k - 4 --  return "775841"
```

- 풀이
  - 리턴할 문자열의 길이 valid 정의
  - 리턴할 문자열의 각 자리수가 나올 수 있는 범위에서 가장 큰 수와 가장 큰 수의 인덱스를 찾아 저장
    - number = “1429” 면 리턴할 문자열의 길이는 2 (두자리 숫자 ab라고 가정)
    - a자리에 올 숫자는 “142” 중에서 찾는다.즉 4
      - a를 “1429”중에 찾으면 만약 a가 9이기 때문에 b를 찾을 곳이 없어짐
    - b자리에 올 숫자는 a자리에 올 4 뒤부터 찾는다. 즉 “29” 중에서 9
- 이렇게 풀었더니 2개에서 시간초과 뜸
  - 리턴할 문자열의 각 자리에 들어가는 문자를 찾을 때 9를 찾았다면 for loop를 끝내서 그 뒤에서 더 큰 숫자를 검사할 필요 없게 하니까 시간 초과 해결

```
def solution(number, k):
    answer = ''
    valid = len(number) - k
    crntidx = 0
```

```
    for i in range(valid,0,-1):
        crntmax = 0
        for index in range(crntidx,len(number)-i+1):
            if int(number[index]) > crntmax:
                crntmax = int(number[index])
                crntidx = index + 1
                # crntmax가 9일 경우 더 큰 수는 없기 때문에 그대로 answer에 추가하고 그 뒤는 탐색하지 않는다
                if crntmax == 9 : break
        answer += str(crntmax)
    return answer
```

구명보트

```
people : 사람들의 몸무게를 담은 배열
limit : 구명보트 1대가 감당할 수 있는 무게
return : 사람들을 모두 구출하기 위해 필요한 구명보트의 최솟값

* 각 구명보트는 최대 2명까지 탑승가능함

ex)
people = [70,50,80,50] , limit = 100 이면
구명보트는 총 3개 필요 (70/80/50,50)
```

- 아이디어
  - 과정 1) 몸무게가 limit/2 보다 큰 애들을 먼저 1인 1보트 함 (얘네끼리는 같은 보트를 탈 수 없기 때문)
  - 과정 2) 보트에 아직 못탄 애들을 이미 보트에 타고 있는 애들 중 가장 남은 capacity가 넓은 보트랑 먼저 비교(여기서 fit하지 않을 경우 그 다음 capacity가 넓은 보트에도 탑승하지 못할 것이기 때문)
  - 과정 3) 그러고도 보트에 아직 못 탄 애들은 걔네끼리 2인 1보트로 태우면 됨(모두 몸무게가 limit/2보다 같거나 작으므로)

```
import math
def solution(people, limit):
    answer = 0
    boat = [] # 과정 1에서 보트에 탈 애들
    # 과정 1
    people.sort()
    while len(people) > 0:
        if people[-1] > limit/2:
            boat.append(people[-1])
            answer += 1
            people.pop()
        else:
            break

    # 과정 2
```

```
    notsavedyet = len(people) # 아직 보트에 타지 못한 사람 수
    for i in range(len(people)-1, -1, -1):
        if len(boat) == 0:
            break
        if people[i] <= limit - boat[-1]: # 보트의 남은 capacity랑 비교
            boat.pop()
            notsavedyet -= 1

    # 과정 3
    answer += math.ceil(notsavedyet/2)
    return answer
```