True False 잘 이용하기(2920)

```python
arr=list(map(int,input().split()))

ascending=True
descending=True

for i in range(len(arr)-1):
    if arr[i+1]>arr[i]:
        descending=False
    elif arr[i+1]<arr[i]:
        ascending=False

if ascending:
    print("ascending")
elif descending:
    print("descending")
else:
    print("mixed")
```

이런식으로 처음에는 True 로 사용했다가 리스트를 차근차근 진행해나가면서 반대의 경우가 존재한다면 False로 바꿔준다.

절반으로 나눠서 계속 찾기

```python
hap=sorted(hap)
start=0
end=len(hap)-1
while start<=end:
    mid = (start + end) // 2
    if hap[mid]<=M:
        result=hap[mid]
        start=mid+1
    elif hap[mid]>M:
        end=mid-1

print(result)
```

항상 start와 end를 바꿔줄때는 1을 더하고 빼는것을 까먹지 말자 !

1. 카드 중 3개씩 뽑는 모든 경우의 수는 $C(n,3)$이며, n은 최대 100입니다. $\frac{n(n-1)(n-2)}{3!}$ ⇒ 정수
2. 따라서 단순히 3중 반복문으로 모든 경우의 수를 확인하여 문제를 해결할 수 있습니다. = 1,000,000

파이썬은 1초에 2000만 정도 연산을 하는데 n이 최대 100일 경우는 n^3 이므로 최대 100만정도 밖에 안되므로 충분히 완전탐색으로 탐색 가능하다. (단순히 3중 반복문으로 가능하다)

카드를 차근차근 뽑을때는 3중 반복문 쓸때 (2798)

```python
N,M=map(int,input().split())
arr=list(map(int,input().split()))
hap=set()

for i in range(len(arr)):
    for j in range(len(arr)):
        if arr[i]==arr[j]:
            continue
        for k in range(len(arr)):
            if arr[i]==arr[k] or arr[j]==arr[k]:
                continue
            hap.add(arr[i]+arr[j]+arr[k])

hap=sorted(hap)
start=0
end=len(hap)-1
while start<=end:
    mid = (start + end) // 2
    if hap[mid]<=M:
        result=hap[mid]
        start=mid+1
    elif hap[mid]>M:
        end=mid-1

print(result)
```

```python
n, m = list(map(int, input().split(' ')))
data = list(map(int, input().split(' ')))

result = 0
length = len(data)

count = 0
for i in range(0, length):
    for j in range(i + 1, length):
        for k in range(j + 1, length):
            sum_value = data[i] + data[j] + data[k]
            if sum_value <= m:
                result = max(result, sum_value)

print(result)
```

굳이 위처럼 카드를 i , j ,k 를 할 때 일일히 0부터 할필요가 없다. 이미 선택됐으므로 앞에 경우는 건너뛰고 넘어가면 된다.

또한 나눠서 찾을 필요 없이 조건에 맞다면 다음식을 이용하여 더한값을 계산하면서 바로 구하는것도 가능하다.

result=max(result,sum_value)

for i in range(10,8) 이런식으로 앞에거가 크면 자동으로 아무런 실행을 하지않는다 따로 빼줄 필요가 없다 !

n,m=[10,9] 이런식으로 되어있으면 미지수와 리스트 데이터 개수가 같으면 자동으로 그 변수에 리스트 위치에 맞는값이 지정된다.

```
result=["1","2","3","4"]
print("".join(result))
```

```
1234
```

```
result=["1","2","3","4"]
print(" ".join(result))
```

```
1 2 3 4
```

```
result=["1","2","3","4"]
print("\n".join(result))
```

```
1
2
3
4
```

"사이에입력하고싶은값".join(리스트이름) 하면은 리스트 안에 있는 원소들을 각각 사이에 "사이에입력하고싶은값을 넣어서 출력시켜준다

```
n=int(input())
count=1
stack=[]
result=[]

for i in range(n):
    number=int(input())
    while count<=number:
        stack.append(count)
        result.append("+")
        count+=1
    if stack[-1]==number:
        stack.pop()
        result.append("-")
    else:
        print("NO")
        exit(0)

print("\n".join(result))
```

. 한번 넣은 숫자는 다시 못넣을 경우는 count로 1부터 하나씩 올리면된다. 예를들어 n 이 8이라면 8까지 넣고 난후에는 더이상 count가 9가 되어 아무리해도 number 가 9보다 커질수가 없어 while 문안에 들어갈수 없고 이미 꺼내진것들은 다빠졌기 때문에 남은것들만 쌓여 있는데 그중에서 스택에 마지막거랑 꺼내는것이 같다면 꺼낼수 있는것이고 만약에 같지 않다면 꺼내지 못하는것이기 때문에 no를 출력하게 된다.

만약 8이 되기전에 5까지 넣고 5를 꺼냈는데 그 다음에 3이라는 숫자가 나온다면은 숫자가 더 작기때문에 더이상 넣지도 못하고 마지막 숫자도 같지 않기때문에 더이상 꺼내지 못하고 no를 출력하게 되는것이다.

숫자가 지금까지 넣은 숫자보다 크다면 더 넣으면 되지만 그거보다 작다면 그냥 꺼내는수밖에 없는데 꺼내는경우의 숫자가 stack 의 마지막거랑 같지않다면 꺼내지 못하는것이다.

인덱스와 데이터 한번에 저장하는 방법 (enumerate)

```
n_list=[[int(x),idx] for idx,x in enumerate(input().split())]
```

```
t=int(input())

for i in range(t):
    n,m=list(map(int,input().split()))
    data=list(map(int,input().split()))
    data2=[]
    for j in range(len(data)):
        data2.append([data[j],j])

    count=0
    while True:
        length = len(data2)
        max_value=data2[0][0]
        for l in range(1,length):
            if max_value<data2[l][0]:
                max_value=data2[l][0]

        while data2[0][0]!=max_value:
            data2.append(data2[0])
            del data2[0]
        count+=1

        if data2[0][1]==m:
            print(count)
            break
        del data2[0]
```

이렇게 일일히 찾을 필요없이 그냥

max함수 사용하면된다.

max(queue,key=lambda x:x[0])

x[0] 기준중에 제일 큰값요소니까

(,) 튜플형태로 뽑혀서 마지막에[0]을 써준것이다.

```
test_case = int(input())

for _ in range(test_case):
    n, m = list(map(int, input().split(' ')))
    queue = list(map(int, input().split(' ')))
    queue = [(i, idx) for idx, i in enumerate(queue)]

    count = 0
    while True:
        if queue[0][0] == max(queue, key=lambda x: x[0])[0]:
            count += 1
            if queue[0][1] == m:
                print(count)
                break
            else:
                queue.pop(0)
        else:
            queue.append(queue.pop(0))
```

이렇게 리스트를 형성한다음에

for idx, i in enumerate(list) 를 통하여 인덱스번호랑 리스트요소를 같이 저장할수 있다.

del queue[0] 말고 queue.pop(0) 이렇게 써도된다.

queue.append(queue.pop(0)) 을 통하여 0번째 데이터를 추가하면서 0번째데이터를 제거할수 있다.

```python
t=int(input())

for _ in range(t):
    data=list(input())
    length=len(data)
    result=[]
    save=[]
    for j in range(length):
        if not result and (data[j]=="<" or data[j]=="-"):
            continue

        elif not save and data[j]==">":
            continue

        elif data[j]=="<" and result:
            save.append(result.pop())

        elif data[j]==">" and save:
            result.append(save.pop())

        elif data[j]=="-" and result:
            result.pop()

        else:
            result.append(data[j])
    print("".join(result)+"".join(save[::-1]))
```

있으면 진행하고 없으면 진행 안하면 된다. 따로 continue 로 뺄 필요없이 조건문안에 조건문을 하나 더만들어줘서 편하게 생각할수도 있다.

```python
test_case = int(input())

for _ in range(test_case):
    left_stack = []
    right_stack = []
    data = input()
    for i in data:
        if i == '-':
            if left_stack:
                left_stack.pop()
        elif i == '<':
            if left_stack:
                right_stack.append(left_stack.pop())
        elif i == '>':
            if right_stack:
                left_stack.append(right_stack.pop())
        else:
            left_stack.append(i)
    left_stack.extend(reversed(right_stack))
    print(''.join(left_stack))
```

```python
a=[1,2,3,4]
b=[8,7,6,5]
a.extend(b)
print(a)
a.extend(reversed(b))
print(a)
print(b)
b.reverse()
print(b)
```

```
[1, 2, 3, 4, 8, 7, 6, 5]
[1, 2, 3, 4, 8, 7, 6, 5, 5, 6, 7, 8]
[8, 7, 6, 5]
[5, 6, 7, 8]
```

두개의 리스트를 합쳐줄때는 extend 사용

리스트를 뒤집어 줄때는 sorted sort 처럼 reversed 는 원본 유지 b.reverse는 원본자체를 뒤집는다.