

MaxDoubleSliceSum

```
# you can write to stdout for debugging purposes, e.g.  
# print("this is a debug message")
```

```
def solution(A):  
    A.pop(); A.pop(0);  
  
    left_arr, max_num = [0], 0  
    for l_i in range(1, len(A)):  
        max_num = max(0, max_num+A[l_i-1])  
        left_arr.append(max_num)  
  
    A.reverse()  
    right_arr, max_num = [0], 0  
    for r_i in range(1, len(A)):  
        max_num = max(0, max_num+A[r_i-1])  
        right_arr.append(max_num)  
    right_arr.reverse()  
  
    max_num = 0  
    for l_num, r_num in zip(left_arr, right_arr):  
        max_num = max(max_num, l_num+r_num)  
    return max_num
```

제목이 Double이 들어간 것처럼 두개의 max slice sum을 만들어서 이를 합해주어야 하는 문제.

풀이 방법 :

non-empty array A consisting of N integer is given.
A triplet (X, Y, Z) , such that $0 \leq X < Y < Z < N$, is
called a double slice.

The sum of double slice (X, Y, Z) is the total of
 $A[X+1] + A[X+2] + \dots + A[Y-1] + A[Y+1] + A[Y+2] + \dots + A[Z-1]$
 \Rightarrow 즉, $A[Y]$ 빼고 $A[X+1] \sim A[Z-1]$ 의 sum을 의미

ex)

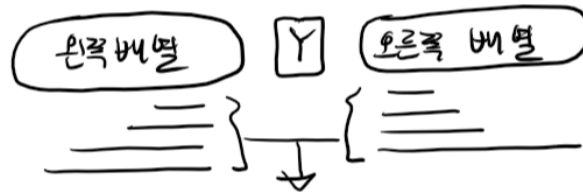
$A = [3, 2, 6, -1, 4, 5, -1, 2]$

X 가 0, Y 가 N 이라 한다면

$A[1] \sim A[N-1]$ 값만 고려되기에

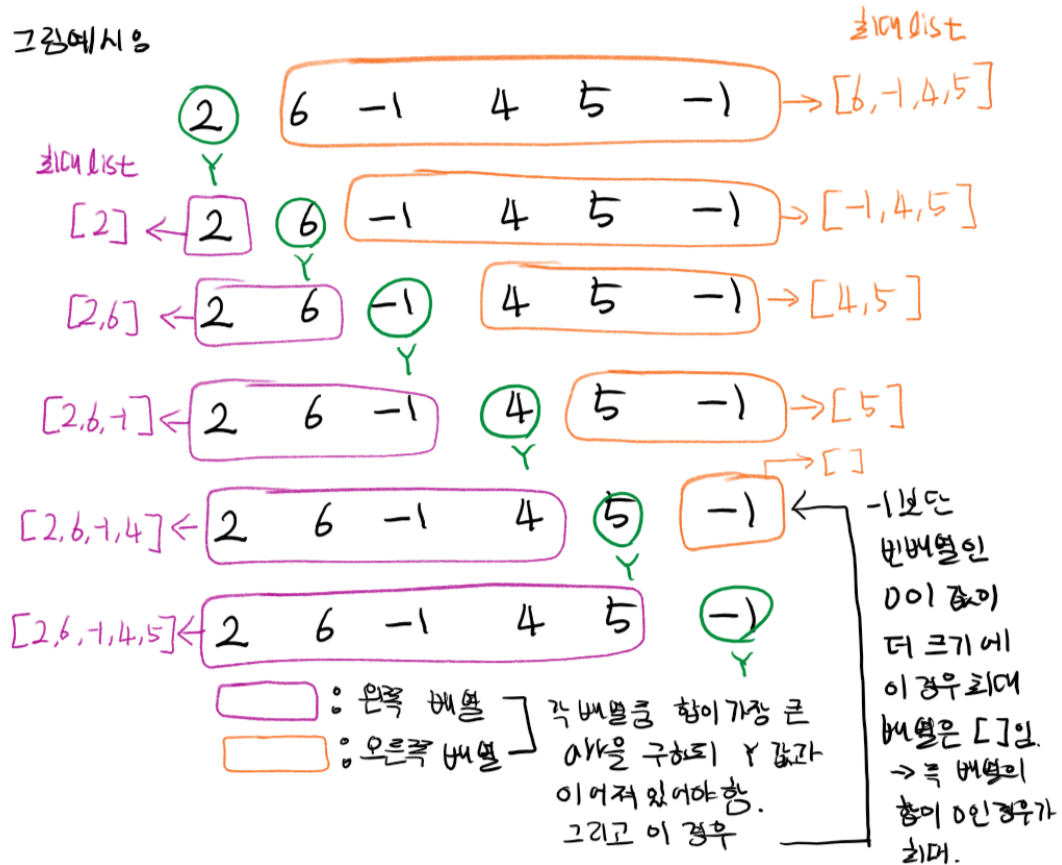
$A[0]$ 과 $A[N]$ 을 실제로 필요 X.

해결법: Y값을 기준으로 오른쪽 배열과 왼쪽 배열의 Max sum이 되는 배열을 찾는다



이런식으로 $Y-1$, $Y+1$ 이 무조건 포함되던가 아니면 빈 배열.

그림예시:



느낀 점 : Kadane's algorithm을 적용. [-1] 배열보다 [] 배열의 원소 값이 0으로 합이 최대가 되는데 이것을 고려하지 못하였음.