Two Sum - using sorting

```
class Solution:
    def two_sum(self, nums: List[int], target: int) -> List[int]:

        A = []
        for i, num in enumerate(nums):
            A.append([num, i])

        A.sort()
        i = 0
        j = len(nums) - 1
        while i < j:
            cur = A[i][0] + A[j][0]
            if cur == target:
                return [min(A[i][1], A[j][1]),
                        max(A[i][1], A[j][1])]
            elif cur < target:
                i += 1
            else:
                j -= 1
```

return []

INFO: enumerate() function adds a counter as the key of the en. obj.

Step 1:

self -> solution instance

nums -> [3,4,5,6]

target -> 7

Step 2 {for i num in enumerate(nums)}

A -> [] - empty

i -> 0

num -> 3

Step 3: (A.append([num, i]))

A -> [[3,0]]

Step 4: (for i, num in enumerate(nums))

i -> 1

num -> 4

Step 5: (A.append([num, i]))

A -> [[3,0], [4,1]]

Step 6: for i, num in enumerate(nums)

i -> 2

num -> 5

Step 7: A.append([num, i])

A -> [[3,0], [4,1], [5,2]]

Step 8: for i, num in enumerate(nums)

i -> 3

num -> 6

Step 9: A.append([num, i])

A -> [[3,0], [4,1], [5,2], [6,3]]
(0, 1, 2, 3)

Step 10: i = 0, j = len(nums) - 1

i -> 0

j -> 3

Step 11: cur = A[i][0] + A[j][0]

cur -> A[0][0] + A[3][0]

=> 3 + 6 =>

cur -> 9

(A[0][0] -> value, index 0)

Step 12 : else: $j -= 1$

$j \rightarrow 2$

Step 13 : while $i < j$:

$cur = A[i][0] + A[j][0]$

$cur \rightarrow A[0][0] + A[2][0]$

$cur \rightarrow 3 + 5 \Rightarrow 8$

Step 14 : else : $j -= 1$

$j \rightarrow 1$

Step 15 : $cur = A[i][0] + A[j][0]$

$cur \rightarrow 7$

Step 16 : if $cur == target$

$7 == 7$ ✓

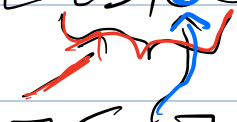
Step 17 : return [$\min(A[i][1], A[j][1])$,

$\max(A[i][1], A[j][1])$]

result $\rightarrow \min(A[0][1], A[1][1])$,

max (A[0][1], A[1][1])

A -> [[3,0], [4,1], [5,2], [6,3]]



min
A[0][1] -> 0
A[1][1] -> [4,1] -> 1
} min => 0

max between 0 and 1 => 1

result -> [0,1]