NO

DATE : :

2023/03/23

Mountain Sort

```
Let recorder = dict  # to store and search found index in O(1)
Let counter = 0      # to record if all elements are found
Let result = []
While counter < array.length
    Let left = queue , l=0
    Let right = heap , r=0
    Let result = []
    For i = 0 to array.length    #scan array → ←
        If dict[i] == 1 and array[i] > l
            left.append(array[i])
            l = array[i] , counter++ , dict[i]=1
        If dict[array.length-i-1] == 1 and array[array.length-i-1] > r
            right.push(array[array.length-i-1])
            r = array[array.length-i-1] , counter++, dict[i]=1
    result = merge(result, left, right)
return result
```

Time Complexity : Best $O(n)$, AVG $O(n^2)$, Worst $O(n^2)$

Space Complexity : $O(n)$

First we scan the array from left to right, and from right to left ($O(n)$) to get two strictly increasing lists, them we use $O(n)$ time to merge three list (result, left, and right), repeat these steps until all element are sorted.

NO

DATE : :

array = [5, 1, 7, 3, 2, 6, 8, 4]

strictly increasing → (from right to left)

step 1: left = [5, 7, 8], right = [4, 6]

→ result = [4, 5, 6, 7, 8]

step 2: left = [1, 3], right = [2]

→ result = [1, 2, 3, 4, 5, 6, 7, 8]

The worst case

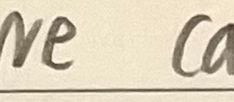
if the input array is like , then we can find 2 elements at most in 1 scan (index = 0 and array.length-1)

$T(n) = T(n-2) + n$

$= T(n-4) + n + n = T(n-6) + n + n + n = \cdots$

$= \frac{1}{2} n \cdot n = O(n^2)$

The best case

if the input array is like  ,  , or  (like a mountain), then we can find all elements at 1 scan

$T(n) = n + n = 2n = O(n)$