975: Odd Even Jump

arr = [10, 13, 12, 14, 15]

↑ final position

Find all the positions (index) and return the count, from which we can reach the end (final position)

We can reach the final index from any index by jumping. The jump is based on the rules:

1) If it's a odd number jump (1st, 3rd...) then jump to the next index that is >= to the current index.

current_index = i

jump to index j, where j > i

and arr[j] >= arr[i]

and arr[j] is smallest in all options.

10, 13, (12), 14, 15

The first jump from index 0 (arr[0] = 10) will be to 12, index = 2

2) If it's a even jump (2nd, 4th..) then vice-versa.

current_index = i

jump to j, where j > i

and arr[j] <= arr[i]

and arr[j] is the largest

arr = [1, 2, 3, 2, 1, 4, 4, 5]

The first jump here goes to 2 (index 1), and second jump from the (index 1), which is even numbered jump. ∴ Applying rule 2 we jump to index 5.

How to solve this?

At every index there can be two possible jumps, either jump would be even, or it would be odd.

∴ For every index we can store the index (next) for odd jump and even jump.

Eg:

hash-max:

[0:4, 1:3, 2:5, 3:5, 4:5, 5:6, 6:7, 7:-1]

hash-min:

[0:4, 1:3, 2:3, 3:4, 4:-1, 5:6, 6:-1, 7:-1]

Time complexity = O(N)

To calculate and store above hashes

We can use the logic of Next Greater element (NGE) using stack to create the above hashes.

Eg:- [10, 13, 12, 14, 15]
0 1 2 3 4

[10, 12, 13, 14, 15]

index array ⇒ [0 2 1 3 4]

sort the arr. with respect to index and run NGE on the index array.

Since the array is already sorted in ascending order the next greater element (for index array) is guaranteed to give the index with NGE.

to calculate hash-min do the same thing but sort the arr in reverse order.

After calculating this, we can use a for-loop or recursion to check which position (index) reaches the end.