Problen #220: Contains Duplicate III

<https://leetcode.com/problems/contains-duplicate-iii/>

Solution:

1. As we iterate through the array, keep the subarray consisting of at most k elements to left sorted. Remove the left most element from the subarray and add the next element.

2. Copy the subarray and create a temp\_array. Look at the position where the current element of the array is inserted maintaining sorting order. Get the index of the current element.

3. Now that we have the index in the temp\_array, check the immediate neighbor to the left and to the right. If the index is the first element, there is only a neighbor to the right.

Also if the index is the last element, there is only a neighbor to the left. If the absolute difference between current element and neighbor <= t, return True.

4. If we go through the whole array and we are not able to find 2 indexes meeting the condition we want, return False.

Import bisect

class Solution:

def containsNearbyAlmostDuplicate(self, nums: List[int], k: int, t: int) -> bool:

n = len(nums)

j = 0

if n == 1:

return(False)

elif n == 2:

if k == 0:

return(False)

elif k == 1:

return(abs(nums[0] - nums[1]) <= t)

subarray = []

for i in range(1, n):

bisect.insort(subarray, nums[i-1]) # keep subarray sorted when inserting previous number

if i > k:

subarray.remove(nums[j])

j += 1

temp\_array = subarray[:] # make a temp array to insert current number & find its neighbors

bisect.insort(temp\_array, nums[i]) # keep array sorted when inserting the current number

idx = temp\_array.index(nums[i]) # find the index where current number would be placed

# Just compare the current number with its immediate neighbors only

if idx == 0:

if (temp\_array[idx + 1] - temp\_array[idx]) <= t:

return(True)

elif idx == len(temp\_array) - 1:

if (temp\_array[idx] - temp\_array[idx - 1]) <= t:

return(True)

else:

if (temp\_array[idx] - temp\_array[idx - 1]) <= t or (temp\_array[idx + 1] - temp\_array[idx]) <= t :

return(True)

return(False)