HW 01: K-Diff Pairs in An Array (Leetcode-532)

Example 1:
Input: nums = [3,1,4,1,5], k = 2
Output: 2

Example 2:
Input: nums = [1,2,3,4,5], k = 1
Output: 4

Example 3:
Input: nums = [1,3,1,5,4], k = 0
Output: 1

Example 4:
Input: nums = [1,1,1,1,1], k = 0
Output: 1

Two Pointer Approach

Step 01: sort the array
Step 02: when diff of nums[i] and nums[j] equal to k then store different ans pair
i++ and j++
Step 03: when diff of nums[i] and nums[j] greater than k then
i++
Step 04: when diff of nums[i] and nums[j] less than k then
j++
Step 05: when i equal to j then i++ because i!=j
Step 06: return numbers of diff pair

DRY RUN

Example 1:

```
Input: nums = [3,1,4,1,5], k = 2
Output: 2
```

nums	3	1	4	1	5

↓

SORT

nums	1	1	3	4	5
	0	1	2	3	4

Iteration : 1

$K = 2$

$i = 0$

$j = 1$

diff < K

→ j++

diff = nums[j] - nums[i]

$= 1 - 1$

$= 0$

Itaration : 2

nums	1	1	3	4	5
	0	1	2	3	4

$K = 2$

$i = 0$

$j = 2$

diff == K

Ans = (1, 3) 1st pain

i++

j++

diff = nums[j] - nums[i]

$= 3 - 1$

$= 2$

Itaration : 3

nums	1	1	3	4	5
	0	1	2	3	4

$K = 2$

$i = 1$

$j = 3$

diff > K

$\hookrightarrow$ i++

$diff = nums[j] - nums[i]$

$= 4 - 1$

$= 3$

Itaration : 4

nums	1	1	3	4	5
	0	1	2	3	4

$K = 2$

$i = 2$

$j = 3$

diff < K

$j{+}{+}$

$$\begin{aligned} diff &= nums[j] - nums[i] \\ &= 4 - 3 \\ &= 1 \end{aligned}$$

Iteration : 5

nums	1	1	3	4	5
	0	1	2	3	4

$K = 2$

$i = 2$

$j = 4$

$diff = nums[j] - nums[i]$

$= 5 - 3$

$= 2$

$diff == K$

$\hookrightarrow$ Ans = (3, 5) 2nd pair

$i++$

$j++$

Itaration : 6

$K = 2$

$i = 3$

$j = 5$

nums

1	1	3	4	5
0	1	2	3	4

$j = 5$
size $= 5$ } Rok jaao .... ($j <$ size)

Return number of pairs $= 2$ Output

DRY RUN

Example 2:
Input: nums = [1,2,3,4,5], k = 1
Output: 4

NUMS	1	2	3	4	5

↓

SORT

NUMS	1	2	3	4	5
	0	1	2	3	4

Iteration: 01

K = 1
i = 0
j = 1

diff = nums[j] - nums[i]
= 2 - 1
= 1

diff == K

ANS = (1, 2) 1st pair
i++
j++

Itmation: 02

Nums

1	2	3	4	5
0	1	2	3	4

K = 1

i = 1

j = 2

diff = nums[j] - nums[i]

= 3 - 2

= 1

diff == K

ANS = (2, 3) 2nd pair

i++

j++

Ituation: 03

NUMS

1	2	3	4	5
0	1	2	3	4

K = 1

i = 2

j = 3

diff = nums[j] - nums[i]

= 4 - 3

= 1

diff == K

ANS = (3, 4) 3rd pair

i++

j++

Iteration: 04

Nums

1	2	3	4	5
0	1	2	3	4

K = 1

i = 3

j = 4

diff = nums[j] - nums[i]

= 5 - 4

= 1

diff == K

ANS = (4, 5) just pair

i++

j++

Itmation: 05

Nums

1	2	3	4	5
0	1	2	3	4

$i = 4$
$j = 5$

size = 5
j = 5 } Rok jaan --- (j < size)

Total pairs = 4 output

DRY RUN

Example 3:

Input: nums = [1,3,1,5,4], k = 0

Output: 1

NUMS	1	3	1	5	4

↓ SORT

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 01

$K = 0$

$i = 0$

$j = 1$

$diff = nums[j] - nums[i]$

$= 1 - 1$

$= 0$

Diff == K

ANS = (1, 1) 1st pair

i++

j++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 02

$K = 0$

$i = 1$

$j = 2$

$diff = nums[j] - nums[i]$

$= 3 - 1$

$= 1$

diff > K

i++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 3

$K = 0$

$i = 2$

$j = 2$

diff = nums[j] - nums[i]

(i == j)

j++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 4

$K = 0$

$i = 2$

$j = 3$

$diff = nums[j] - nums[i]$

$= 4 - 3$

$= 1$

diff > K

i++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 5

$K = 0$

$i = 3$

$j = 3$

diff = nums[j] - nums[i]

(i == j)

j++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 6

$K = 0$

$i = 3$

$j = 4$

diff = nums[j] - nums[i]

$= 5 - 4$

$= 1$

diff > K

i++

NUMS	1	1	3	4	5
	0	1	2	3	4

Iteration: 7

$K = 0$

$i = 4$

$j = 4$

diff = nums[j] - nums[i]

(i == j)

→ j++

Iteration: 8

$K = 0$

$i = 4$

$j = 5$

NUMS	1	1	3	4	5
	0	1	2	3	4

size = 5
j = 5 } Rok jaaoo ..... (size > j)

Total pairs = 1

Output

DRY RUN

Example 4:
Input: nums = [1,1,1,1,1], k = 0
Output: 1

NUMS

1	1	1	1	1

↓

SORT

NUMS

1	1	1	1	1
0	1	2	3	4

Iteration: 0

$K = 0$

$i = 0$

$j = 1$

$Diff = nums[j] - nums[i]$

$= 1 - 1$

$= 0$

$Diff == K$

→ $ANS = (1, 1)$ 1st pair

$i++$

$j++$

Iteration: 02

NUMS

1	1	1	1	1
0	1	2	3	4

$K = 0$

$i = 1$

$j = 2$

$Diff = nums[j] - nums[i]$

$= 1 - 1$

$= 0$

$Diff == K$

$ANS = (1, 1)$ 2nd pair

$i++$

$j++$

Iteration: 03

NUMS

1	1	1	1	1
0	1	2	3	4

K = 0

i = 2

j = 3

Diff = nums[j] - nums[i]

= 1 - 1

= 0

Diff == K

ANS = (1, 1) 3rd pair

i++

j++

Iteration: 04

NUMS				
1	1	1	1	1
0	1	2	3	4

K = 0

i = 3

j = 4

Diff = nums[j] - nums[i]

= 1 - 1

= 0

Diff == K

ANS = (1, 1) not pair

i++

j++

Iteration: 05

NUMS				
1	1	1	1	1
0	1	2	3	4

$K = 0$

$i = 4$

$j = 5$

$size = 5$
$j = 5$ } $K \leq K$ ज़ाओ ..... $(size > j)$

Total unique Different pairs = 1

Output

```cpp
// HW 01: K-Diff Pairs in An Array (Leetcode-532)

/*
Two Pointer Approach
Step 01: sort the array
Step 02: when diff of nums[i] and nums[j] equal to k then store different ans pair
        i++ and j++
Step 03: when diff of nums[i] and nums[j] greater than k then
        i++
Step 04: when diff of nums[i] and nums[j] less than k then
        j++
Step 05: when i equal to j then i++ because i!=j
Step 06: return numbers of diff pair

Time Complexity: O(NlogN), where N is length of array nums
Space Complexity: O(1), no extra space used
*/
class Solution {
public:
    int twoPointerSol(vector<int>& nums, int k){
        sort(nums.begin(),nums.end());
        set<pair<int,int>> ans;
        int n = nums.size();
        int i = 0;
        int j = 1;

        while(j < n){
            if(nums[j] - nums[i] == k){
                ans.insert({nums[i],nums[j]});
                i++;
                j++;
            }
            else if(nums[j] - nums[i] < k){
                j++;
            }
            else{
                i++;
            }
            if(i==j){
                j++;
            }
        }
        return ans.size();
    }

    int findPairs(vector<int>& nums, int k) {
        return twoPointerSol(nums, k);
    }
};
```

Binary Search Approach

Step 01: sort the array

Step 02: apply binary search

DRY RUN

Example 1:
Input: nums = [3,1,4,1,5], k = 2
Output: 2

SORT

1	1	3	4	5
0	1	2	3	4

iteration: 01

K=2
i=0
target = nums[i]+K

target = 1 + 2
= 3

1	3	4	5
1	2	3	4

target found then 1st pair (1,3) and i++

iteration : 02

1	1	3	4	5
0	1	2	3	4

K = 2

i = 1

target = nums[i] + K

target = 1 + 2

= 3

3	4	5
2	3	4

target found then 2nd pair (1, 3) and i++

iteration : 03

1	1	3	4	5
0	1	2	3	4

K = 2

i = 2

target = nums[i] + K

target = 3 + 2
= 5

4	5
3	4

target found then print pair (3, 5) and i++

iteration : 04

1	1	3	4	5
0	1	2	3	4

K=2

i=3

target = nums[i] + K

target = 4 + 2
= 6

5
4

target not found and i++

iteration: 05

1	1	3	4	5
0	1	2	3	4

K=2

i=4

target = nums[i] + K

target = 5 + 2

= 7

↳ Target not found and i++

itmation: 6

$i = 5$
$size = 5$ } RUK jaoo ... (i < size)

Total unique diff pairs = 2

output

```cpp
// HW 01: K-Diff Pairs in An Array (Leetcode-532)

/*
Binary Search Approach
Step 01: sort the array
Step 02: apply binary search

Time Complexity: O(NlogN), where N is length of array nums
Space Complexity: O(1), no extra space used
*/
class Solution {
public:
    // Binary Search Approach
    bool bs(vector<int>& nums,int start,int target){
        int end = nums.size()-1;
        while(start<=end){
            int mid = start + (end-start)/2;
            if(nums[mid]==target){
                return true;
            }
            else if(nums[mid]>target){
                end = mid-1;
            }
            else{
                start = mid+1;
            }
        }
        return false;
    }
    int binarySortingSol(vector<int>& nums, int k){
        sort(nums.begin(),nums.end());
        set<pair<int,int>> ans;

        for(int i=0;i<nums.size();i++){
            bool target = bs(nums,i+1,nums[i]+k);

            if(target){
                ans.insert({nums[i],nums[i]+k});
            }
        }
        return ans.size();
    }

    int findPairs(vector<int>& nums, int k) {
        return binarySortingSol(nums, k);
    }
};
```