

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 = nums[j] - nums[i]

diff < k
→ j++

$$= 1 - 1$$

$$= 0$$

Iteration : 2

$$K=2$$

$$i=0$$

$$j=2$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

$$= 3 - 1$$

$$= 2$$

nums	1	1	3	4	5
	0	1	2	3	4

$\text{diff} == K$

→ $\text{Ans} = (1, 3)$ 1st pair

$i++$

$j++$

Iteration : 3

$$K=2$$

$$i=1$$

$$j=3$$

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

nums	1	1	3	4	5
	0	1	2	3	4

$\text{diff} > K$
 $\rightarrow i++$

Iteration : 4


$$K = 2$$

$$i = 2$$

$$j = 3$$

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

nums	1	1	3	4	5
	0	1	2	3	4

$\text{diff} < K$
 $j++$

Iteration : 5

$$K=2$$

$$i=2$$

$$j=4$$

$$\begin{aligned} \text{diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 5 - 3 \\ &= 2 \end{aligned}$$

nums	1	1	3	4	5
	0	1	2	3	4

$$\text{diff} == K$$

→ Ans = (3, 5) 2nd pair
i++
j++

Iteration: 6

$K=2$

$i=3$

$j=5$

nums	1	1	3	4	5
	0	1	2	3	4

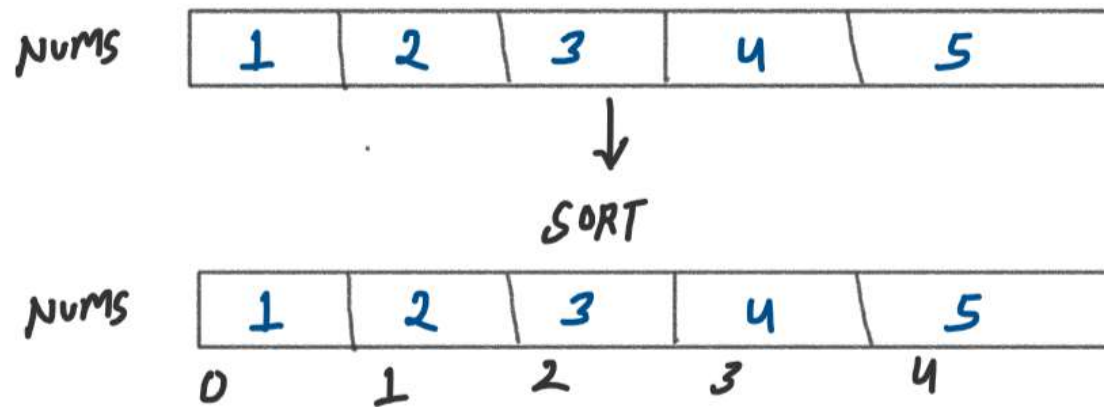
$j=5$
 $size=5$ } Run java... ($j < size$)

Return number of pairs = 2 Output

DRY
Example 2:

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

Output: 4



Iteration: 01

k = 1

i = 0

j = 1

$$\begin{aligned} \text{diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 2 - 1 \\ &= 1 \end{aligned}$$

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

Iteration: 02

$K = 1$

$i = 1$

$j = 2$

$$\begin{aligned} \text{diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 3 - 2 \\ &= 1 \end{aligned}$$

nums	1	2	3	4	5
	0	1	2	3	4

$\text{diff} == K$
 $\rightarrow \text{Ans} = (2, 3)$ 2nd pair
 $i++$
 $j++$

Iteration: 03

$K = 1$

$i = 2$

$j = 3$

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

nums

1	2	3	4	5
0	1	2	3	4

$\text{diff} == K$
 $\rightarrow \text{ANS} = (3, 4)$ and pair
 $i++$
 $j++$

Iteration: 04

$$K = 1$$

$$i = 3$$

$$j = 4$$

$$\begin{aligned} \text{diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 5 - 4 \\ &= 1 \end{aligned}$$

nums

1	2	3	4	5
0	1	2	3	4

$\text{diff} = K$
 $\rightarrow \text{ANS} = (4, 5)$ not pair
 $i++$
 $j++$

Iteration: 05

$i = 4$
 $j = 5$

nums

1	2	3	4	5
0	1	2	3	4

$size = 5$
 $j = 5$ } For $j = 5 \dots (j < size)$

Total pairs = 4 output

DRY

Example 3:

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

Output: 1

Iteration: 01

$k = 0$

$i = 0$

$j = 1$

$\text{diff} = \text{nums}[j] - \text{nums}[i]$

$= 1 - 1$

$= 0$

nums

1	3	1	5	4
---	---	---	---	---

↓
sort

nums

1	1	3	4	5
---	---	---	---	---

0 1 2 3 4

$\text{diff} == k$

↪ $\text{ANS} = (1, 1)$ 1st pair
 $i++$
 $j++$

Iteration: 02

$$K = 0$$

$$i = 1$$

$$j = 2$$

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

nums	1	1	3	4	5
	0	1	2	3	4

diff > K
→ i++

Iteration: 0

$$K = 0$$

$$i = 2$$

$$j = 2$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

nums	1	1	3	4	5
	0	1	2	3	4

$(i == j)$
 $\rightarrow j++$

Iteration: 4

$$K = 0$$

$$i = 2$$

$$j = 3$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

$$= 4 - 3$$

$$= 1$$

nums	1	1	3	4	5
	0	1	2	3	4

Diff > K
→ i++

Iteration: 05

$$K = 0$$

$$i = 3$$

$$j = 3$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

nums	1	1	3	4	5
	0	1	2	3	4

$(i == j)$
 $\rightarrow j++$

Iteration: 6

$$K = 0$$

$$i = 3$$

$$j = 4$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

$$= 5 - 4$$

$$= 1$$

nums	1	1	3	4	5
	0	1	2	3	4

diff > K
→ i++

Iteration: 0 7

$$K = 0$$

$$i = 4$$

$$j = 4$$

$$\text{diff} = \text{nums}[j] - \text{nums}[i]$$

nums	1	1	3	4	5
	0	1	2	3	4

($i == j$)
→ $j++$

Iteration: 8

$$K = 0$$

$$i = 4$$

$$j = 5$$

nums	1	1	3	4	5
	0	1	2	3	4

$\left. \begin{array}{l} \text{size} = 5 \\ j = 5 \end{array} \right\} \text{Rok jaaaa... (size > j)}$

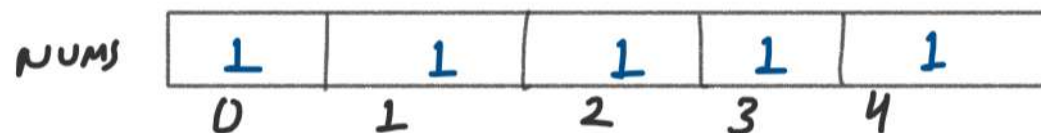
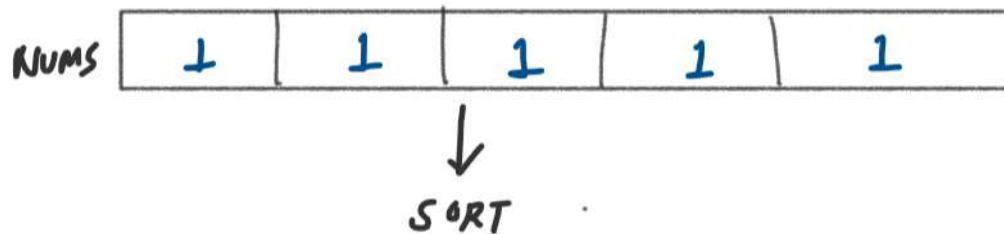
Total pairs = 1
Output

DRY RUN

Example 4:

Input: nums = [1,1,1,1,1], k = 0

Output: 1



Initialization:

$$k = 0$$

$$i = 0$$

$$j = 1$$

$$\text{Diff} = \text{nums}[j] - \text{nums}[i]$$

$$= 1 - 1$$

$$= 0$$

$$\text{Diff} == k$$

→ **ANS = (1, 1)** 1st pair

i++

j++

Iteration: 02

$$K = 0$$

$$i = 1$$

$$j = 2$$

$$\begin{aligned} \text{Diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 1 - 1 \\ &= 0 \end{aligned}$$

nums	1	1	1	1	1
	0	1	2	3	4

Diff == K
→ **ANS = (1, 1)** 2nd pair
i++
j++

Iteration: 03

$$K = 0$$

$$i = 2$$

$$j = 3$$

$$\begin{aligned} \text{Diff} &= \text{nums}[j] - \text{nums}[i] \\ &= 1 - 1 \\ &= 0 \end{aligned}$$

nums	1	1	1	1	1
	0	1	2	3	4

Diff == K
→ **ANS = (2, 1)** 3rd pair
i++
j++

Iteration: 04

$$K = 0$$

$$i = 3$$

$$j = 4$$

$$\text{Diff} = \text{nums}[j] - \text{nums}[i]$$

$$= 1 - 1$$

$$= 0$$

nums	1	1	1	1	1
	0	1	2	3	4

Diff == K
→ **ANS = (1, 1)** not pair
i++
j++

Iteration: 05

$K = 0$

$i = 4$

$j = 5$

nums

1	1	1	1	1
0	1	2	3	4

$$\left. \begin{matrix} \text{size} = 5 \\ j = 5 \end{matrix} \right\} \text{R \& R } j=0 \dots \dots (\text{size} > j)$$

Total unique different pair = 1
Output

```

// 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

Iteration: 01

K=2

i=0

target = nums[i] + K

SORT

1	1	3	4	5
0	1	2	3	4

$$\begin{aligned} \text{target} &= 1 + 2 \\ &= 3 \end{aligned}$$

1	3	4	5
1	2	3	4

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

iteration: 02

$K=2$

$i=1$

$target = nums[i] + K$

1	1	3	4	5
0	1	2	3	4

$$target = 1 + 2 \\ = 3$$

3	4	5
2	3	4

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

iteration: 03

$K=2$

$i=2$

$target = nums[i] + K$

1	1	3	4	5
0	1	2	3	4

$$target = 3 + 2 = 5$$

4	5
3	4

→ target Found then int pair (3,5) and $i++$

iteration: 04

$$K=2$$

$$i=3$$

$$\text{target} = \text{nums}[i] + K$$

1	1	3	4	5
0	1	2	3	4

$$\begin{aligned}\text{target} &= 4 + 2 \\ &= 6\end{aligned}$$

5

→ target⁴ not found and i++

iteration: 05

$K=2$

$i=4$

$target = nums[i] + K$

1	1	3	4	5
0	1	2	3	4

$$target = 5 + 2 \\ = 7$$

→ Target not found and $i++$

iteration: 6

$i = 5$
 $size = 5$ } $Run\ j = 0 \dots (i < size)$

Total unique diff pairs = 2

output

```

// 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);
    }
};

```