719. Find K-th Smallest Pair Distance

The **distance of a pair** of integers `a` and `b` is defined as the absolute difference between `a` and `b`.

Given an integer array `nums` and an integer `k`, return *the* `k`th *smallest* ***distance among all the pairs*** `nums[i]` *and* `nums[j]` *where* `0 <= i < j < nums.length`.

Example 1:

```
Input: nums = [1,3,1], k = 1
Output: 0
Explanation: Here are all the pairs:
(1,3) -> 2
(1,1) -> 0
(3,1) -> 2
Then the 1st smallest distance pair is (1,1), and its distance is 0.
```

Example 2:

```
Input: nums = [1,1,1], k = 2
Output: 0
```

Example 3:

```
Input: nums = [1,6,1], k = 3
Output: 5
```

Constraints:

- `n == nums.length`
- `2 <= n <= 10`4
- `0 <= nums[i] <= 10`6
- `1 <= k <= n * (n - 1) / 2`

719. Find K-th Smallest Pair Distance

```
/*
    Brute force (naive)- TLE
    n: size of input array
    mx: maximum number in input array
    Time complexity: O(n^2+mxlogmx)
    Space complexity: O(mx)
*/
typedef std::vector<int> vi;

class Solution {
    public:
        int smallestDistancePair(vi& nums, int k) {
            int n = nums.size();
            vi diffs;
            for(int i=0;i<n-1;++i){
                for(int j=i+1;j<n;++j){
                    diffs.push_back(abs(nums[i]-nums[j]));
                }
            }

            std::sort(diffs.begin(),diffs.end());

            return diffs[k-1];

        }
};
```

719. Find K-th Smallest Pair Distance

```
/*
    Linear search - TLE
    n: size of input array
    mx: maximum number in input array
    Time complexity: O(nlogn+mx^2)
    Space complexity: O(1)
*/
typedef std::vector<int> vi;
class Solution {
    public:
        int smallestDistancePair(vi& nums, int k) {
            int n=nums.size();
            std::sort(nums.begin(),nums.end());
            int mx=*std::max_element(nums.begin(),nums.end());

            /*
                All absolute difference are in range [0-mx],
                So, is the k-th smaller difference.
            */
            for(int diff=0;diff<=mx;++diff){
                /*
                    Using sliding window on the input array, count total number of pairs with
                    absolute difference <= diff
                */
                auto count=[&](void)->int{
                    int l=0,cnt=0;
                    for(int r=0;r<n;++r){
                        while(nums[r]-nums[l]>diff) l++;
                        cnt+=r-l;
                    }
                    return cnt;
                };

                int cnt=count();

                if(cnt>=k) return diff;
            }
            return -1; // Never reached
        }
};
```

719. Find K-th Smallest Pair Distance

```
/* Binary search - AC
    n: size of input array
    mx: maximum number in input array
    Time complexity: O(nlogn+nlogmx)
    Space complexity: O(1)
*/
typedef std::vector<int> vi;
class Solution {
    public:
        int smallestDistancePair(vi& nums, int k) {
            int n=nums.size();
            std::sort(nums.begin(),nums.end());
            int mx=*std::max_element(nums.begin(),nums.end());
            /*
                All absolute difference are in range [0-mx],
                So, is the k-th smaller difference.
            */
            int lo=0,hi=mx;
            while(lo<hi){
                int mid=(lo+hi)>>1;
                /*
                    Using sliding window on the input array, count total number of pairs with
                    absolute difference <= diff
                */
                auto count=[&](void)->int{
                    int l=0,cnt=0;
                    for(int r=0;r<n;++r){
                        while(nums[r]-nums[l]>mid) l++;
                        cnt+=r-l;
                    }
                    return cnt;
                };
                int cnt=count();
                if(cnt>=k) hi=mid;
                else lo=mid+1;
            }
            return hi; // or lo
        }
};
```