Minimum Possible value of |ai + aj – k| for given array and k.

You are given an array of n integer and an integer K. Find the number of total unordered pairs {i, j} such that absolute value of (ai + aj – K), i.e., |ai + aj – k| is minimal possible where i != j.

**Examples:**

Input : arr[] = {0, 4, 6, 2, 4},

K = 7

Output : Minimal Value = 1

Total Pairs = 5

Explanation : Pairs resulting minimal value are :

{a1, a3}, {a2, a4}, {a2, a5}, {a3, a4}, {a4, a5}

Input : arr[] = {4, 6, 2, 4} , K = 9

Output : Minimal Value = 1

Total Pairs = 4

Explanation : Pairs resulting minimal value are :

{a1, a2}, {a1, a4}, {a2, a3}, {a2, a4}

[**Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.**](https://ide.geeksforgeeks.org/)

A **simple solution** is iterate over all possible pairs and for each pair we will check whether the value of (ai + aj – K) is smaller then our current smallest value of not. So as per result of above condition we have total of three cases :

1. *abs( ai + aj – K) > smallest*: do nothing as this pair will not count in minimal possible value.
2. *abs(ai + aj – K) = smallest*: increment the count of pair resulting minimal possible value.
3. *abs( ai + aj – K) < smallest*: update the smallest value and set count to 1.

* C++
* Java
* Python3
* C#
* PHP

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // CPP program to find number of pairs  and minimal  // possible value  #include<bits/stdc++.h>  using namespace std;    // function for finding pairs and min value  void pairs(int arr[], int n, int k)  {      // initialize smallest and count      int smallest = INT\_MAX;      int count=0;        // iterate over all pairs      for (int i=0; i<n; i++)          for(int j=i+1; j<n; j++)          {              // is abs value is smaller than smallest              // update smallest and reset count to 1              if ( abs(arr[i] + arr[j] - k) < smallest )              {                  smallest = abs(arr[i] + arr[j] - k);                  count = 1;              }                // if abs value is equal to smallest              // increment count value              else if (abs(arr[i] + arr[j] - k) == smallest)                  count++;          }            // print result          cout << "Minimal Value = " << smallest << "\n";          cout << "Total Pairs = " << count << "\n";  }    // driver program  int main()  {      int arr[] = {3, 5, 7, 5, 1, 9, 9};      int k = 12;      int n = sizeof(arr) / sizeof(arr[0]);      pairs(arr, n, k);      return 0;  } |

**Output:**

Minimal Value = 0

Total Pairs = 4

An**efficient solution**is to use a self balancing binary search tree (which is implemented in set in C++ and TreeSet in Java). We can find closest element in O(log n) time in map.

* C++

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C++ program to find number of pairs  // and minimal possible value  #include<bits/stdc++.h>  using namespace std;    // function for finding pairs and min value  void pairs(int arr[], int n, int k)  {      // initialize smallest and count      int smallest = INT\_MAX, count = 0;      set<int> s;        // iterate over all pairs      s.insert(arr[0]);      for (int i=1; i<n; i++)      {          // Find the closest elements to  k - arr[i]          int lower = \*lower\_bound(s.begin(),                                   s.end(),                                   k - arr[i]);            int upper = \*upper\_bound(s.begin(),                                   s.end(),                                   k - arr[i]);            // Find absolute value of the pairs formed          // with closest greater and smaller elements.          int curr\_min = min(abs(lower + arr[i] - k),                             abs(upper + arr[i] - k));            // is abs value is smaller than smallest          // update smallest and reset count to 1          if (curr\_min < smallest)          {              smallest = curr\_min;              count = 1;          }            // if abs value is equal to smallest          // increment count value          else if (curr\_min == smallest )              count++;          s.insert(arr[i]);        }        // print result        cout << "Minimal Value = " << smallest <<"\n";      cout << "Total Pairs = " << count <<"\n";  }    // driver program  int main()  {      int arr[] = {3, 5, 7, 5, 1, 9, 9};      int k = 12;      int n = sizeof(arr) / sizeof(arr[0]);      pairs(arr, n, k);      return 0;  } |

**Output:**

Minimal Value = 0

Total Pairs = 4

**Time Complexity :** O(n Log n)