Started on	Sunday, 18 February 2024, 10:55 PM
State	Finished
Completed on	Monday, 19 February 2024, 12:25 AM
Time taken	1 hour 30 mins
Marks	30.00/30.00
Grade	**10.00** out of 10.00 (**100**%)

The previous challenges covered <u>Insertion Sort</u>, which is a simple and intuitive sorting algorithm with a running time of $O(n^2)$. In these next few challenges, we're covering a *divide-and-conquer* algorithm called <u>Quicksort</u> (also known as *Partition Sort*). This challenge is a modified version of the algorithm that only addresses partitioning. It is implemented as follows:

Step 1: Divide
Choose some pivot element, p, and partition your unsorted array, arr, into three smaller arrays: $left$, $right$, and $equal$, where each element in $left < p$, each element in $right > p$, and each element in $equal = p$.

Example
$$arr = [5, 7, 4, 3, 8]$$
In this challenge, the pivot will always be at $arr[0]$, so the pivot is 5.

arr is divided into $left = \{4, 3\}$, $equal = \{5\}$, and $right = \{7, 8\}$.
Putting them all together, you get $\{4, 3, 5, 7, 8\}$. There is a flexible checker that allows the elements of $left$ and $right$ to be in any order. For example, $\{3, 4, 5, 8, 7\}$ is valid as well.

Given arr and $p = arr[0]$, partition arr into $left$, $right$, and $equal$ using the *Divide* instructions above. Return a 1-dimensional array containing each element in $left$ first, followed by each element in $equal$, followed by each element in $right$.

Function Description

Complete the *quickSort* function in the editor below.

quickSort has the following parameter(s):

- *int arr[n]:* $arr[0]$ is the pivot element

Returns

- *int[n]:* an array of integers as described above

Input Format

The first line contains n, the size of arr.
The second line contains n space-separated integers $arr[i]$ (the unsorted array). The first integer, $arr[0]$, is the pivot element, p.

Constraints

- $1 \leq n \leq 1000$
- $-1000 \leq arr[i] \leq 1000$ where $0 \leq i < n$
- All elements are distinct.

Sample Input

```
STDIN       Function
-----       --------
5           arr[] size n =5
4 5 3 7 2   arr =[4, 5, 3, 7, 2]
```

Sample Output

```
3 2 4 5 7
```

Explanation

$arr = [4, 5, 3, 7, 2]$ *Pivot:* $p = arr[0] = 4$.
$left = \{\}; equal = \{4\}; right = \{\}$

$arr[1] = 5 > p$, so it is added to $right$.
$left = \{\}; equal = \{4\}; right = \{5\}$

$arr[2] = 3 < p$, so it is added to $left$.
$left = \{3\}; equal = \{4\}; right = \{5\}$

$arr[3] = 7 > p$, so it is added to $\boldsymbol{right}$.

$left = \{3\}; equal = \{4\}; right = \{5, 7\}$

$arr[4] = 2 < p$, so it is added to $\boldsymbol{left}$.

$left = \{3, 2\}; equal = \{4\}; right = \{5, 7\}$

Return the array $\{32457\}$.

The order of the elements to the left and right of 4 needs to match the order of the original list.

For example:

Input	Result
5 4 5 3 7 2	3 2 4 5 7

Answer: (penalty regime: 0 %)

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  vector<int> partition(vector<int>& inputArray, int pivotValue) {
7      vector<int> left, equal, right;
8
9      vector<int>::size_type index;
10     for (index = 0; index < inputArray.size(); index++) {
11         if (inputArray[index] < pivotValue) {
12             left.push_back(inputArray[index]);
13         } else if (inputArray[index] == pivotValue) {
14             equal.push_back(inputArray[index]);
15         } else {
16             right.push_back(inputArray[index]);
17         }
18     }
19
20     vector<int> result;
21     result.insert(result.end(), left.begin(), left.end());
22     result.insert(result.end(), equal.begin(), equal.end());
23     result.insert(result.end(), right.begin(), right.end());
24
25     return result;
26 }
27
28 int main() {
29     vector<int>::size_type arraySize;
30     cin >> arraySize;
31
32     vector<int> inputArray(arraySize);
33     for (vector<int>::size_type i = 0; i < arraySize; i++) {
34         cin >> inputArray[i];
35     }
36
37     int pivotValue = inputArray[0];
38
39     vector<int> sortedArray = partition(inputArray, pivotValue);
40
41     for (vector<int>::size_type i = 0; i < sortedArray.size(); i++) {
42         cout << sortedArray[i] << " ";
43     }
44
45     return 0;
46 }
```

	Input	Expected	Got	
✔	5 4 5 3 7 2	3 2 4 5 7	3 2 4 5 7	✔

Passed all tests! ✔

▸ **Show/hide question author's solution (Cpp)**

Marks for this submission: 10.00/10.00.

▸ **Show/hide question author's solution (Cpp)**

Marks for this submission: 10.00/10.00.

Whenever George asks Lily to hang out, she's busy doing homework. George wants to help her finish it faster, but he's in over his head! Can you help George understand Lily's homework so she can hang out with him?

Consider an array of n distinct integers, $arr = [a[0], a[1], \ldots, a[n-1]]$. George can swap any two elements of the array any number of times. An array is *beautiful* if the sum of $|arr[i] - arr[i-1]|$ among $0 < i < n$ is minimal.

Given the array arr, determine and return the minimum number of swaps that should be performed in order to make the array *beautiful*.

Example

$arr = [7, 15, 12, 3]$

One minimal array is $[3, 7, 12, 15]$. To get there, George performed the following swaps:

```
   Swap      Result
          [7, 15, 12, 3]
   3 7    [3, 15, 12, 7]
   7 15   [3, 7, 12, 15]
```

It took 2 swaps to make the array beautiful. This is minimal among the choices of beautiful arrays possible.

Function Description

Complete the *lilysHomework* function in the editor below.

lilysHomework has the following parameter(s):

- *int arr[n]:* an integer array

Returns

- *int:* the minimum number of swaps required

Input Format

The first line contains a single integer, n, the number of elements in arr. The second line contains n space-separated integers, $arr[i]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^9$

Sample Input

```
STDIN       Function
-----       --------
4           arr[]size n = 4
2 5 3 1     arr = [2, 5, 3, 1]
```

Sample Output

```
2
```

Explanation

Define $arr' = [1, 2, 3, 5]$ to be the beautiful reordering of arr. The sum of the absolute values of differences between its adjacent elements is minimal among all permutations and only two swaps (1 with 2 and then 2 with 5) were performed.

For example:

Input	Result
4 2 5 3 1	2
5 3 4 2 5 1	2

Answer: (penalty regime: 0 %)

Reset answer

```cpp
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   string ltrim(const string &);
6   string rtrim(const string &);
7   vector<string> split(const string &);
8
9   /*
10   * Complete the 'lilysHomework' function below.
11   *
12   * The function is expected to return an INTEGER.
13   * The function accepts INTEGER_ARRAY arr as parameter.
14   */
15
16  int lilysHomework(vector<int> arr_orig) {
17      int result = INT_MAX;
18      vector<int> sorted(arr_orig);
19      sort(sorted.begin(), sorted.end());
20      for (size_t rev = 0; rev < 2; rev++) {
21          int curSwap = 0;
22          if (rev) {
23              reverse(sorted.begin(), sorted.end());
24          }
25          vector<int> arr(arr_orig);
26          // value, position
27          unordered_map<int, size_t> valueToPosition;
28          for (size_t i = 0; i < arr.size(); i++) {
29              valueToPosition[arr[i]] = i;
30          }
31          for (size_t i = 0; i < arr.size(); i++) {
32              if (arr[i] == sorted[i]) {
33                  continue;
34              }
35              int currentElement = arr[i];
36              int sortedElement = sorted[i];
37              swap(arr[i], arr[valueToPosition[sortedElement]]);
38              curSwap++;
39              valueToPosition[currentElement] = valueToPosition[sortedE]
40              valueToPosition[sortedElement] = i;
41          }
42          result = min(result, curSwap);
43      }
44      return result;
45  }
46
47  int main()
48  {
49      string n_temp;
50      getline(cin, n_temp);
51
52      int n = stoi(ltrim(rtrim(n_temp)));
```

	Input	Expected	Got	
✔	4 2 5 3 1	2	2	✔
✔	5 3 4 2 5 1	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

Sorting is useful as the first step in many different tasks. The most common task is to make finding things easier, but there are other uses as well. In this case, it will make it easier to determine which pair or pairs of elements have the smallest absolute difference between them.

Example

$arr = [5, 2, 3, 4, 1]$

Sorted, $arr' = [1, 2, 3, 4, 5]$. Several pairs have the minimum difference of 1:
$[(1, 2), (2, 3), (3, 4), (4, 5)]$. Return the array $[1, 2, 2, 3, 3, 4, 4, 5]$.

Note
As shown in the example, pairs may overlap.

Given a list of unsorted integers, arr, find the pair of elements that have the smallest absolute difference between them. If there are multiple pairs, find them all.

Function Description

Complete the *closestNumbers* function in the editor below.

closestNumbers has the following parameter(s):

- *int arr[n]:* an array of integers

Returns
- *int[]:* an array of integers as described

Input Format

The first line contains a single integer n, the length of arr.
The second line contains n space-separated integers, $arr[i]$.

Constraints

- $2 \leq n \leq 200000$
- $-10^7 \leq arr[i] \leq 10^7$
- All $a[i]$ are unique in arr.

Output Format

Sample Input 0

```
10
-20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854
```

Sample Output 0

```
-20 30
```

Explanation 0
(30) - (-20) = 50, which is the smallest difference.

Sample Input 1

```
12
-20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854 -520
-470
```

Sample Output 1

```
-520 -470 -20 30
```

Explanation 1
(-470) - (-520) = 30 - (-20) = 50, which is the smallest difference.

Sample Input 2

```
4
5 4 3 2
```

Sample Output 2

```
2 3 3 4 4 5
```

Explanation 2

Here, the minimum difference is *1*. Valid pairs are *(2, 3)*, *(3, 4)*, and *(4, 5)*.

Submissions:

196

Max Score:

25

Difficulty:

Easy

Rate This Challenge:

More

C++

For example:

Input	Result
10 -20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854	-20 30
12 -20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854 -520 -470	-520 -470 -20 30
4 5 4 3 2	2 3 3 4 4 5

Answer: (penalty regime: 0 %)

Reset answer

```cpp
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string ltrim(const string &);
6  string rtrim(const string &);
7  vector<string> split(const string &);
8
9  /*
10  * Complete the 'closestNumbers' function below.
11  *
12  * The function is expected to return an INTEGER_ARRAY.
13  * The function accepts INTEGER_ARRAY arr as parameter.
14  */
15
16  vector<int> closestNumbers(vector<int> arr) {
17      sort(arr.begin(), arr.end());
18      int minDiff = INT_MAX;
19      vector<int> res;
20      for (size_t i = 1; i < arr.size(); ++i) {
21          int diff = abs(arr[i] - arr[i - 1]);
22          if (diff < minDiff) {
23              minDiff = diff;
24              res.clear();
25              res.push_back(arr[i - 1]);
26              res.push_back(arr[i]);
27          } else if (diff == minDiff) {
28              res.push_back(arr[i - 1]);
29              res.push_back(arr[i]);
30          }
31      }
32      return res;
```

```cpp
33  }
34
35
36  int main()
37  {
38      string n_temp;
39      getline(cin, n_temp);
40
41      int n = stoi(ltrim(rtrim(n_temp)));
42
43      string arr_temp_temp;
44      getline(cin, arr_temp_temp);
45
46      vector<string> arr_temp = split(rtrim(arr_temp_temp));
47
48      vector<int> arr(n);
49
50      for (int i = 0; i < n; i++) {
51          int arr_item = stoi(arr_temp[i]);
52
```

	Input	Expected	Got	
✔	10 -20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854	-20 30	-20 30	✔
✔	12 -20 -3916237 -357920 -3620601 7374819 -7330761 30 6246457 -6461594 266854 -520 -470	-520 -470 -20 30	-520 -470 -20 30	✔
✔	4 5 4 3 2	2 3 3 4 4 5	2 3 3 4 4 5	✔

Passed all tests! ✔

▸ Show/hide question author's solution (Cpp)