Lily's Homework

locked

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.

Submissions: 191
Max Score: 35
Difficulty: Medium

Rate This Challenge:

More

C++

```cpp
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'lilysHomework' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

int lilysHomework(vector<int> arr) {
    vector<int> arr1=arr, ascArr=arr, descArr=arr;
    unordered_map<int,int>ascending,descending;
    int n = (int)arr.size();

    for(int i=0;i<n;i++)
        ascending[arr[i]]=i;
    for(int i=0;i<n;i++)
        descending[arr1[i]]=i;

    // doing quicksort to ascending order
    int end = n-1;
    int start = 0;
    int stack[end - start + 1];
    int top = -1;
    stack[++top] = start;
    stack[++top] = end;

    while (top >= 0)
    {
        end = stack[top--];
```

```
        start = stack[top--];

        int i = start, j = end;
        int tmp;
        int pivot = ascArr[(start + end) / 2];

        /* partition */
        while (i <= j)
        {
            while (ascArr[i] < pivot)
                i++;
            while (ascArr[j] > pivot)
                j--;
            if (i <= j)
            {
                tmp = ascArr[i];
                ascArr[i] = ascArr[j];
                ascArr[j] = tmp;
                i++;
                j--;
            }
        };

        /* push values to stack */
        if (i < end)
        {
            stack[++top] = i;
            stack[++top] = end;
        }
        if (start < j)
        {
            stack[++top] = start;
            stack[++top] = j;
        }
    }
    //-----------

    // doing quicksort in descending order
    int end2 = n-1;
    int start2 = 0;
    int stack2[end2 - start2 + 1];
    int top2 = -1;
    stack2[++top2] = start2;
    stack2[++top2] = end2;

    while (top2 >= 0)
    {
        end2 = stack2[top2--];
        start2 = stack2[top2--];

        int i = start2, j = end2;
        int tmp;
        int pivot = descArr[(start2 + end2) / 2];

        /* partition */
        while (i <= j)
        {
            while (descArr[i] > pivot)
                i++;
            while (descArr[j] < pivot)
                j--;
            if (i <= j)
            {
                tmp = descArr[i];
                descArr[i] = descArr[j];
                descArr[j] = tmp;
```

```
                i++;
                j--;
            }
        };

        /* push values to stack */
        if (i < end2)
        {
            stack2[++top2] = i;
            stack2[++top2] = end2;//
        }
        if (start2 < j)
        {
            stack2[++top2] = start2;
            stack2[++top2] = j;
        }
    }
    // -----------

    int swapsAsc=0,swapsDesc=0;
    for(int i=0;i<n;i++){
        if(arr[i]!=ascArr[i]){
            swapsAsc++;
            int temp=ascending[ascArr[i]];
            ascending[arr[i]]=temp;
            swap(arr[i],arr[temp]);
        }
    }
    for(int i=0;i<n;i++){
        if(arr1[i]!=descArr[i]){
            swapsDesc++;
            int temp=descending[descArr[i]];
            descending[arr1[i]]=temp;
            swap(arr1[i],arr1[temp]);
        }
    }
    return min(swapsAsc,swapsDesc);
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    string arr_temp_temp;
    getline(cin, arr_temp_temp);

    vector<string> arr_temp = split(rtrim(arr_temp_temp));

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        int arr_item = stoi(arr_temp[i]);

        arr[i] = arr_item;
    }

    int result = lilysHomework(arr);

    fout << result << "\n";

    fout.close();
```

```
169
170     return 0;
171 }
172
173 string ltrim(const string &str) {
174     string s(str);
175
176     s.erase(
177         s.begin(),
178         find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
179     );
180
181     return s;
182 }
183
184 string rtrim(const string &str) {
185     string s(str);
186
187     s.erase(
188         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
189         s.end()
190     );
191
192     return s;
193 }
194
195 vector<string> split(const string &str) {
196     vector<string> tokens;
197
198     string::size_type start = 0;
199     string::size_type end = 0;
200
201     while ((end = str.find(" ", start)) != string::npos) {
202         tokens.push_back(str.substr(start, end - start));
203
204         start = end + 1;
205     }
206
207     tokens.push_back(str.substr(start));
208
209     return tokens;
210 }
211
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code