PREPARE NEW CERTIFY COMPETE Search 210554M_CSE_21

All Contests > In21-CS2023-Lab5 > Lily's Homework

Lily's Homework

locked

Problem | Submissions | Leaderboard | Discussions

Submitted 2 minutes ago • Score: 35.00

Status: Accepted

✓	Test Case #0	✓	Test Case #1	✓	Test Case #2
✓	Test Case #3	✓	Success 0.02s	✓	Test Case #5
✓	Test Case #6	✓	Download	✓	Test Case #8
✓	Test Case #9	✓	Test Case #10	✓	Test Case #11

Submitted Code

nguage: C++

Open in editor

```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();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}

```