Projeto e Análise de Algoritmos - 2023.2

Aluno Maria Eduarda Carvalho Santos **Professor** Jan Mendonça
Projeto Trabalho 1 **Matrícula** 190092556 **Data** 22 / 09 / 2023/ **Sala** PAT 045

Trabalho 1 Análise Assintótica

Usuário do SPOJ: auntduda
URL: auntduda

1 Exercício 1 - MERGESORT

Simple. Sort the numbers on the standard input using the merge sort algorithm. Don't try to cheat by just calling your build in functions... I can see your source.

Input

On the standard input you will receive N ($1 \leq N \leq 100000$). Each number will fit in 32-bit integer

Output

Output the same integers in a sorted manner. Smallest to largest.

Example

Input	Output
7 3 2 5 4 3	2 3 3 4 5 7

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

using namespace std;

void merge(int *saida, int *auxiliar, int inicio, int meio, int fim){
    int i, j, k;
    i = inicio;
    j = meio + 1;
    k = inicio;
    while(i <= meio && j <= fim){
        if(auxiliar[i] < auxiliar[j]){
            saida[k] = auxiliar[i];
            i++;
        }
        else{
            saida[k] = auxiliar[j];
            j++;
        }
        k++;
    }

    while(i <= meio){
        saida[k] = auxiliar[i];
        i++;
```

```
        k++;
    }

    while(j <= fim){
        saida[k] = auxiliar[j];
        j++;
        k++;
    }
    //Copia os elementos que foram ordenados para o auxiliar
    for(int p = inicio; p <= fim; p++)
        auxiliar[p] = saida [p];
}



void mergeSort(int *saida, int *auxiliar, int inicio, int fim){
    if(inicio < fim){
        int meio = (inicio + fim) / 2;
        mergeSort(saida, auxiliar, inicio, meio);
        mergeSort(saida, auxiliar, meio + 1, fim);
        merge(saida, auxiliar, inicio, meio, fim);
    }
}

int main()
{
    int n; vector<int> a;

    while(cin >> n) a.push_back(n);

    int v[a.size()]; int ans[a.size()];

    for(int i=0; i<a.size(); i++) v[i] = a[i];

    mergeSort(ans, v, 0, a.size()-1);

    for(int i=0; i<a.size(); i++)
    {
        cout << ans[i] << " ";
    } cout << "\n";

    return 0;

}
```

2 Exercício 2 - INSERTIONSORT

Insertion Sort is a classical sorting technique. One variant of insertion sort works as follows when sorting an array a[1..N] in non-descending order:

```
for i <- 2 to N
    j <- i
    while j > 1 and a[j] < a[j - 1]
        swap a[j] and a[j - 1]
        j <- j - 1
```

The pseudocode is simple to follow. In the ith step, element a[i] is inserted in the sorted sequence a[1..i - 1]. This is done by moving a[i] backward by swapping it with the previous element until it ends up in it's right position.
As you probably already know, the algorithm can be really slow. To study this more, you want to find out the number of times the swap operation is performed when sorting an array.

Input

The first line contains the number of test cases T. T test cases follow. The first line for each case contains N, the number of elements to be sorted. The next line contains N integers a[1], a[2] ... a[N].

Output

Output T lines, containing the required answer for each test case.

Constraints

$1 \leq T \leq 5$

$1 \leq N \leq 100000$

$1 \leq a[i] \leq 100000$

Example

Input	Output
2 **5** **1 1 1 2 2** **5** **2 1 3 1 2**	**0** **4**

```
#include <iostream>

using namespace std;

long long merge(int arr[], int start, int mid, int end) {
    int aux_size = end - start + 1;
    int aux[aux_size];

    long long swaps = 0;

    int pos_aux, pos_firstarray = start, pos_secondarray = mid + 1;

    // compara os valores das duas arrays e coloca o menor primeiro na array auxiliar
```

```
    for (pos_aux = 0; pos_aux < aux_size; pos_aux++) {
        if (pos_firstarray > mid or pos_secondarray > end) break;
        if (arr[pos_firstarray] <= arr[pos_secondarray]) {
            aux[pos_aux] = arr[pos_firstarray];
            pos_firstarray++;
        }
        else {
            aux[pos_aux] = arr[pos_secondarray];
            pos_secondarray++;
            swaps += mid - pos_firstarray + 1;
        }
    }

    // coloca os elementos que tiverem sobrado de alguma array
    if (pos_firstarray > mid) {
        for (int i = pos_secondarray; i <= end; i++) {
            aux[pos_aux] = arr[i];
            pos_aux++;
        }
    }

    else {
        for (int i = pos_firstarray; i <= mid; i++) {
            aux[pos_aux] = arr[i];
            pos_aux++;
        }
    }

    // coloca os valores da array auxiliar na array principal
    int counter = 0;
    for (int i = start; i <= end; i++) {
        arr[i] = aux[counter];
        counter++;
    }

    return swaps;
}

long long mergesort(int arr[], int start, int end) {
    long long swaps = 0;
    if (start >= end) return 0;
    int mid = start + (end - start) / 2;
    swaps += mergesort(arr, start, mid);
    swaps += mergesort(arr, mid + 1, end);
    swaps += merge(arr, start, mid, end);

    return swaps;
}

int main() {
```

```
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int t; cin >> t;
    while (t--) {
        int n; cin >> n;
        int arr [n];

        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }

        cout << mergesort(arr, 0, n - 1) << '\n';
    }

    return 0;
}
```

3 Exercício 3 - BSEARCH1

You are given a sorted array of numbers, and followed by number of queries, for each query if the queried number is present in the array print its position, else print -1.

Input

First line contains N Q, number of elements in the array and number of queries to follow.

Second line contains N numbers, the elements of the array. Each number will be $-10^9 \leq ai \leq 10^9, 0 < N \leq 10^5, 0 < Q \leq 5 * 10^5$

Output

For each element in the query, print the elements 0 based location of its first occurence, if present, otherwise print -1.

Example

Input	Output
5 4 2 4 7 7 9 7 10 4 2	2 -1 1 0

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

using namespace std;

#define endl "\n"

int main() {
```

```
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);

    int n, count; cin >> n >> count;

    vector<int> a(n);

    for(int i=0; i<n; i++) cin >> a[i];

    while(count--)
    {
      int x; cin >> x;

    int ans=-1;

    int aux = lower_bound(a.begin(), a.end(), x) - a.begin();

    if(aux!=n) ans=aux;
    if(a[ans]!=x) ans=-1;

    cout << ans << endl;
    }

    return 0;
}
```