Electrical energy

locked

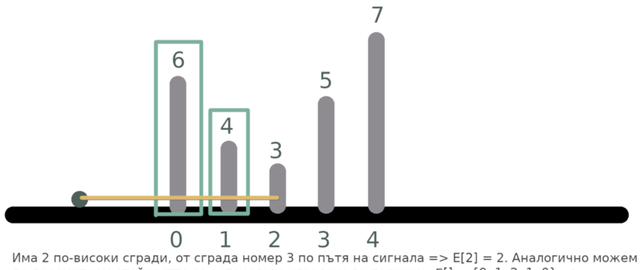
Problem | Submissions | Leaderboard | Discussions

За тази домашна работа не е разрешено използването на std::sort и други вградени методи за сортиране

В града в който живеете има редица от N на брой сгради. Всяка сграда има определена височкина - H[i].

Вие разполагате с предавател, който може да изпрати сигнал до сграда по ваш избор. За да изпратите сигнал до определена сграда ви трябва различно количество електрическа енергия, базирано на броя по-високи сгради от нея по пътя на сигнала. По-точно за да изпратите сигнал до сграда с пореден номер i, ви трябва електрическа енергия E[i] = броят по-високи сгради между вашият предавател и въпрсоната сграда.

Сега ви интересува колко общо електрическа енергия би ви трябвала, за да изпратите сигнал до всяка от N-те сгради в редицата. Намерете нужната енергия, ако знаете че предавателят ви се намира от лявата страна на редицата от сгради.



Има 2 по-високи сгради, от сграда номер 3 по пътя на сигнала => E[2] = 2. Аналогично можем да пресметнем стойността за останалите сгради и да получим: E[] = {0, 1, 2, 1, 0} =>
Σ E[i] = 4, което е и общото количество енергия, която ще ни е необходима.

Input Format

На първият ред на входа се въвежда числото N - брой на сградите в редицата.

Следват N числа H[0], ..., H[N - 1] - височините на сградите.

Constraints

0 <= N <= 100 000;

0 <= H[i] <= 100 000;

Output Format

Изведете 1 число - необходимата електрическа енергия, за да изпратите сигнал до всяка сграда.

Sample Input 0

```
5
6 4 3 5 7
```

Sample Output 0

```
4
```

Submissions: 131
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
long int counter=0;
void merge(int* originalArray, int start, int mid, int end) {
    int* mergeArray=new int[end+1];
    int leftIndex = start;
    int rightIndex = mid + 1;
    int sortedIndex = start;

    while (leftIndex <= mid && rightIndex <= end) {
        if (originalArray[leftIndex] <= originalArray[rightIndex]) {
            mergeArray[sortedIndex] = originalArray[leftIndex];
            leftIndex++;
        }
        else {
            mergeArray[sortedIndex] = originalArray[rightIndex];
            rightIndex++;
            counter+=mid-leftIndex+1;

        }

        sortedIndex++;
    }

    while (leftIndex <= mid) {
        mergeArray[sortedIndex] = originalArray[leftIndex];
        leftIndex++;
        sortedIndex++;
    }
    while (rightIndex <= end) {
        mergeArray[sortedIndex] = originalArray[rightIndex];
        rightIndex++;
        sortedIndex++;
    }

    for (int i = start; i <= end; i++) {
        originalArray[i] = mergeArray[i];
    }
    delete[] mergeArray;

}
void mergeSort(int* arr, int start, int end) {

    if (start < end) {
        int mid = (start + end) / 2;
        mergeSort(arr, start, mid);
        mergeSort(arr, mid + 1, end);
        merge(arr, start, mid, end);

    }
}

int main() {
    int buildingNumber;
    cin>>buildingNumber;
    int* buildings=new int[buildingNumber];
    int buildingSize;
    for(int i=0;i<buildingNumber;i++){
        cin>>buildings[i];
    }
mergeSort(buildings,0,buildingNumber-1);
   delete[] buildings;
   cout<<counter;
    return 0;
}
```

Line: 1 Col: 1

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