График

locked

Problem Submissions Leaderboard Discussions

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

На всеки студент във ФМИ всеки ден се налага да взема много тежки решения - дали да отиде на лекция по Структури от Данни и Алгоритми или в зала 100 на бордови игри. Тъй като това е много тежък проблем и засяга всички нас, от вас се иска да напишете програма, която го решава веднъж завинаги.

В графика на един студент има много събития, които за простота ще бъдат с една и съща важност (лекциите не са толкова приятни колкото бордовите игри, но спестяват време през сесията). Освен това всеки студент би искал да отиде на максимален брой събития, защото така живота му става по-лесен или по-приятен. В рамките на едно събитие е включено и пътуването до него. Следователно ако едно събитие завършва в даден момент, а друго събитие започва в абсолютно същото време, студентът може да отиде и на двете

Input Format

На първия ред на входа се въвежда едно число N - броя събития. Следват N реда, описващи всяко събитие. Събитията се описват с 2 числа Bi и Ti – **началото** на поредното събитие спрямо някакъв начален момент и **продължителността** му измерена в същата мерна единица.

Constraints

0 ≤ N ≤ 1 000 000

0 ≤ Bi + Ti ≤ INT_MAX

0 ≤ Bi, Ti

Output Format

Отпечатайте едно число - възможно най-големия брой събития, на които може да се отиде.

Sample Input 0

```
5
1 2
2 3
3 1
4 2
5 2
```

Sample Output 0

```
3
```

Explanation 0

Може да се отиде на първото, третото и четвъртото събитие и така общия брой събития е 3. Не е възможно да отиде на повече от 3 събития.

Submissions: 119
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;
int eventsPossible(pair<int,int>* event,int eventNumber){
    if(eventNumber==0){
        return 0;
    }
    int counter=1;
    for(int i=0;i<eventNumber-2;i++){
        for(int j=i+1;j<eventNumber-1;j++){
        if(event[i].second<=event[j].first){
            counter++;
            i=j;
            j++;
        }


    }
    }
    return counter;
}
void merge(pair<int,int>* originalArray, int start, int mid, int end) {
    pair<int,int>* mergeArray=new pair<int,int>[end+1];
    int leftIndex = start;
    int rightIndex = mid + 1;
    int sortedIndex = start;

    while (leftIndex <= mid && rightIndex <= end) {
        if (originalArray[leftIndex].second <= originalArray[rightIndex].second) {
            mergeArray[sortedIndex] = originalArray[leftIndex];
            leftIndex++;

        }
        else {
            mergeArray[sortedIndex] = originalArray[rightIndex];
            rightIndex++;

        }

        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(pair<int,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 eventNumber;
    cin>>eventNumber;
    pair<int,int>* event=new pair<int,int>[eventNumber];
    int* startTime=new int[eventNumber];
    int eventTime;
    int* endTime=new int[eventNumber];
    for(int i=0;i<eventNumber;i++){
        cin>>event[i].first;
        cin>>eventTime;
        event[i].second=event[i].first+eventTime;

    }
    mergeSort(event,0,eventNumber-1);

    cout<<eventsPossible(event,eventNumber);

    return 0;
}
```

Line: 1 Col: 1

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