Антон Макеев

Яндекс. Тренировки по алгоритмам 2.0, занятие 7 (B)

9 окт 2021, 12:39:58
старт: 23 сен 2021, 12:00:00
начало: 23 сен 2021, 12:00:00

B. Таможня

Ограничение времени	3 секунды
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Идёт 2163 год. Мишу, который работает в отделении таможни при космодроме города Нью-Питер, вызвал в кабинет шеф.
Как оказалось, недавно Министерство Налогов и Сборов выделило отделению определённую сумму денег на установку новых аппаратов для автоматического досмотра грузов. Естественно, средства были выделены с таким расчётом, чтобы грузы теперь находились на таможне ровно столько времени, сколько требуется непосредственно на их досмотр.

В руках шефа каким-то образом оказались сведения о надвигающейся ревизии – список из N грузов, которые будут контролироваться Министерством. Для каждого груза известны время его прибытия, отсчитываемое с некоторого момента, хранимого в большом секрете, и время, требуемое аппарату для обработки этого груза. Шеф дал Мише задание по этим данным определить, какое минимальное количество аппаратов необходимо заказать на заводе, чтобы все грузы Министерства начинали досматриваться сразу после прибытия. Необходимо учесть, что конструкция тех аппаратов, которые было решено установить, не позволяет обрабатывать два груза одновременно на одном аппарате. Напишите программу, которая поможет Мише справиться с его задачей.

Формат ввода

На первой строке входного файла задано число N ($0 \le N \le 50\,000$). На следующих N строках находится по 2 целых положительных числа T_i и L_i – время прибытия соответствующего груза и время, требуемое для его обработки ($1 \le T_i \le 10^6$, $1 \le L_i \le 10^6$).

Формат вывода

В выходной файл выведите одно число – наименьшее количество аппаратов, которое нужно установить, чтобы не вызвать подозрений у Министерства.

Пример 1

Ввод	Вывод
3 3 2 4 2 5 2	2

Пример 2

Ввод	Вывод
5	3

```
13 4
15 1
11 5
12 3
10 3
```

Язык Swift 5.3

Набрать здесь | Отправить файл

```
import Foundation

struct CargoEvent: Comparable {
    enum Event: Int {
        case arrival = 1
        case departure = 0
    }
    let timePoint: Int
    let type : Event

    static func < (lhs: CargoEvent, rhs: CargoEvent) -> Bool {
        lhs.timePoint < rhs.timePoint ||
            (lhs.timePoint == rhs.timePoint && lhs.type.rawValue < rhs.type.rawValue)
    }
    static func == (lhs: CargoEvent, rhs: CargoEvent) -> Bool {
        lhs.timePoint == rhs.timePoint && lhs.type == rhs.type
    }

}

func readCargoEvents() -> [CargoEvent] {
    let n = Int(readLine()!)!
    var cargoEvents = [CargoEvent]()
    for _ in 0..<n {
        let input = readLine()!.components(separatedBy: " ").map { Int($0)! }
        cargoEvents.append(CargoEvent(timePoint: input[0], type: .arrival))
        cargoEvents.append(CargoEvent(timePoint: input[0]+input[1], type: .departure))
    }
    return cargoEvents
}

func findMaxCargosSimultaneously(during cargoEvents: [CargoEvent]) -> Int {
    let events = cargoEvents.sorted()
    var maxCargosSimultaneously = 0
    var currentCargosCnt = 0
    for event in events {
        if event.type == .arrival {
            currentCargosCnt += 1
```

Отправить

Предыдущая Следующая