Тренировочный контест — бэкенд

D. Лей, лей, не жалей

✓ Полное решение </> OK Go 1.23.0

Исходный код

```
package main

import (
    "bufio"
    "fmt"
    "maps"
    "os"
    "slices"
    "sort"
    "strconv"
    "strings"
)

type Order struct {
    start int
    end   int
    cost  int
}

func NewOrder(data []string) *Order {
    return &Order{
        start: mustAtoi(data[0]),
        end:   mustAtoi(data[1]),
        cost:  mustAtoi(data[2]),
    }
}

type Request struct {
    start int
    end   int
    type_ int
}

func NewRequest(data []string) *Request {
    return &Request{
        start: mustAtoi(data[0]),
        end:   mustAtoi(data[1]),
        type_: mustAtoi(data[2]),
    }
}

func mustAtoi(s string) int {
    i, err := strconv.Atoi(s)
    if err != nil {
        panic(err)
    }
    return i
}

func buildPrefixsums(scan *bufio.Scanner, N int) (map[int]int, []int, map[int]int, []int) {
    start2cost := make(map[int]int)
```

```
    end2time := make(map[int]int)
    start2cost[0] = 0
    end2time[0] = 0

    for range N {
        scan.Scan()
        lineElements := strings.Split(scan.Text(), " ")
        order := NewOrder(lineElements)

        start2cost[order.start] += order.cost
        end2time[order.end] += order.end - order.start
    }

    s2cKeys := slices.Sorted(maps.Keys(start2cost))
    e2tKeys := slices.Sorted(maps.Keys(end2time))

    for i := 1; i < len(s2cKeys); i++ {
        start2cost[s2cKeys[i]] += start2cost[s2cKeys[i-1]]
    }
    for i := 1; i < len(e2tKeys); i++ {
        end2time[e2tKeys[i]] += end2time[e2tKeys[i-1]]
    }

    return start2cost, s2cKeys, end2time, e2tKeys
}

func main() {
    file, err := os.Open("input.txt")
    if err != nil {
        panic(err)
    }
    defer file.Close()

    scan := bufio.NewScanner(file)
    scan.Scan()
    N, err := strconv.Atoi(scan.Text())
    if err != nil {
        panic(err)
    }

    start2cost, s2cKeys, end2time, e2tKeys := buildPrefixsums(scan, N)

    // Пропускаем число запросов
    scan.Scan()

    for scan.Scan() {
        request := NewRequest(strings.Split(scan.Text(), " "))

        if request.type_ == 1 {
            leftIndex := sort.Search(len(s2cKeys), func(i int) bool { return s2cKeys[i] >= request.start })
            rightIndex := sort.Search(len(s2cKeys), func(i int) bool { return s2cKeys[i] > request.end })
            fmt.Printf("%d ", start2cost[s2cKeys[rightIndex-1]]-start2cost[s2cKeys[leftIndex-1]])
        } else {
            leftIndex := sort.Search(len(e2tKeys), func(i int) bool { return e2tKeys[i] >= request.start })
            rightIndex := sort.Search(len(e2tKeys), func(i int) bool { return e2tKeys[i] > request.end })
            fmt.Printf("%d ", end2time[e2tKeys[rightIndex-1]]-end2time[e2tKeys[leftIndex-1]])
        }
    }
}

```

Отличия от предыдущей посылки

Лог компиляции

№	Вердикт	Ресурсы	Баллы	
1	ok	3ms / 1.88Mb	—	⌄

№	Вердикт	Ресурсы	Баллы
2	ok	3ms / 2.00Mb	—
3	ok	3ms / 2.00Mb	—
4	ok	3ms / 2.00Mb	—
5	ok	3ms / 2.00Mb	—
6	ok	3ms / 2.00Mb	—
7	ok	3ms / 1.88Mb	—
8	ok	3ms / 2.00Mb	—
9	ok	3ms / 1.88Mb	—
10	ok	3ms / 2.00Mb	—
11	ok	3ms / 2.00Mb	—
12	ok	3ms / 2.00Mb	—
13	ok	3ms / 1.88Mb	—
14	ok	3ms / 2.00Mb	—
15	ok	0.832s / 29.78Mb	—
16	ok	0.604s / 20.23Mb	—
17	ok	0.526s / 5.03Mb	—
18	ok	0.507s / 5.03Mb	—
19	ok	0.614s / 18.50Mb	—
20	ok	0.574s / 5.03Mb	—
21	ok	0.56s / 5.03Mb	—
22	ok	0.83s / 29.78Mb	—
23	ok	0.839s / 29.27Mb	—
24	ok	0.805s / 29.78Mb	—
25	ok	0.804s / 29.27Mb	—
26	ok	0.824s / 30.30Mb	—
27	ok	0.803s / 29.78Mb	—
28	ok	0.799s / 30.30Mb	—
29	ok	0.812s / 29.78Mb	—
30	ok	0.827s / 28.75Mb	—