acwing 4908 饥饿的牛

贝茜是一头饥饿的牛。

每天晚上，如果牛棚中还有干草的话，贝茜都会吃掉其中的一捆。

初始时，牛棚中没有干草。

为了让贝茜不被饿死，农夫约翰制定了 N 个给贝茜送干草的计划。

其中第
i 个计划是在第
d_i 天的白天给贝茜送去
b_i 捆干草。

这些计划互不冲突，保证 $1 \le d_1 < d_2 < \ldots < d_N \le T$。

请你计算，贝茜在第 $1 \sim T$ 天中有多少天有干草吃。

输入格式

第一行包含两个整数
N 和
T。

接下来
N 行，每行包含两个整数
d_i, b_i。

输出格式

输出贝茜在第 $1 \sim T$ 天中有干草吃的天数。

数据范围

$1 \le N \le 10^5$,
$1 \le T \le 10^{14}$,
$1 \le d_i \le 10^{14}$,
$1 \le b_i \le 10^9$。

时/空限制: 1s / 64MB

总通过数: 3512

总尝试数: 12063

来源:
USACO 2023 February Contest Bronze

算法标签

输入样例1:

```
1 5
1 2
```

输出样例1:

```
2
```

样例1解释

两捆干草在第
1 天早上被送到了牛棚，所以贝茜第
1, 2 天有干草吃。

输入样例2:

```
2 5
1 2
5 10
```

输出样例2:

```
3
```

样例2解释

两捆干草在第
1 天早上被送到了牛棚，所以贝茜第
1, 2 天有干草吃。

10 捆干草在第 5 天早上被送到了牛棚，所以贝茜第 5 天有干草吃。

输入样例3:

```
2 5
1 10
5 10
```

输出样例3:

```
5
```

本题本人的初始思路是使用差分离散化存储的方式，接着遍历所有日期并判断是否符合要求。时间复杂度为 `O(T)`，指数级为14次。

```cpp
//
//  main.cpp
//  4908-饥饿的牛
//
//  Created by MacBook Pro on 2023/7/14.
//

#include <iostream>
#include <map>
using namespace std;

int N,T;
```

```
// 非可行算法--差分离散化后模拟导致复杂度过高
//unordered_map<int, int> nums;   //存储差分数组
//
//int main(int argc, const char * argv[]) {
//    //差分离散化
//    cin>>N>>T;
//    for(int i=0;i<N;i++){
//        int index;
//        cin>>index;
//        cin>>nums[index];
//    }
//    int ans=0;
//    int day=0;
//    for(int i=1;i<=T;i++){
//        if(nums.find(i)!=nums.end()){
//            day+=nums[i];
//        }
//        if(!day){
//            ans++;
//        }
//        else{
//            day--;
//        }
//    }
//    cout<<T-ans<<endl;
//    return 0;
//}

int main(){
    cin>>N>>T;
    int nums=0;
    int day_before=1;
    long long int ans=0;
    for(int i=0;i<N;i++){
        int di,bi;
        scanf("%d %d",&di,&bi);
```

```
            int duration=di-day_before; //上一次喂干草的历时（不包含di）
            if(duration<=nums){
                nums-=duration;
                ans+=duration;
            }
            else{
                ans+=nums;
                nums=0;
            }
            day_before=di;
            nums+=bi;
        }
        int duration=T+1-day_before; //上一次喂干草的历时（不包含di）
        if(duration<=nums){
            ans+=duration;
        }
        else{
            ans+=nums;
        }
        cout<<ans<<endl;

    }
```

该种方法时间复杂度过高，该进为遍历所有输入情况，并将本次输入和上次输入之间的时间计算中间结果。注意需要处理末尾的情况。

思路和 `@zzc` 相同 Code：

```
    int main(){
        cin>>N>>T;
        int nums=0;
        int day_before=1;
        long long int ans=0;
        for(int i=0;i<N;i++){
            int di,bi;
```

```
        scanf("%d %d",&di,&bi);
        int duration=di-day_before; //上一次喂干草的历时（不包含di）
        if(duration<=nums){
            nums-=duration;
            ans+=duration;
        }
        else{
            ans+=nums;
            nums=0;
        }
        day_before=di;
        nums+=bi;
    }
    int duration=T+1-day_before; //上一次喂干草的历时（不包含di）
    if(duration<=nums){
        ans+=duration;
    }
    else{
        ans+=nums;
    }
    cout<<ans<<endl;

}
```

代码提交状态： Accepted