**АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА**

**ТЪРКАЛЯЩО СЕ ТОПЧЕ**

Сигурно прави впечатление странността на конструкциите, които се разглеждат в задачата, а именно, че или , или . Случаят е изключен от задачата. Да не си помислите, че това се дължи на някакви особени физични закони! Просто авторите нямат добро решение за този случай☺. Ако някой реши задачата за този случай, авторите ще му бъдат безкрайно признателни, ако им съобщи идеята си.

**Идеи за решение**

Условията топчето да прескочи i-тия хълм са да:

1. Прескочи първите i-1 хълма
2. Е изпълнено неравенството

Ако неравенството е изпълнено за всяко i, очевидно имаме валидно решение. Нека пренапишем неравенството като:

Да положим .

Решение на задачата е такъв набор от височини, че всички стойности на F са неотрицателни.

Нека разгледаме изменението на F при промяна на височините на различни хълмове.

Очевидно е, че намаляване на височината на първия хълм, както и увеличаване на височината на който и да е друг, са операции, които не могат да присъстват в оптималното решение. Затова разглеждаме само две възможни промени:

*Повдигане на първия хълм с единица – “Операция A”*

Това води до увеличаване на Fi с ed за всяко i.

*Намаляване на хълм i>1 с единица – “Операция B” или “Операция Bi”*

Това води до увеличаване на Fi с eu, и увеличаване на Fj със (eu-ed) за всяко j>i.

Нека симулираме движението на топчето при началните височини и нека хълм номер **k** е първият, който топчето не може да прескочи. От начините по които се променя F, лесно можем да заключим, че няма смисъл да намаляваме хълм с номер по-малък от k ако е възможно да намалим k.

Единственият избор който трябва да направим е колко пъти да намалим текущия ‚блокиращ’ хълм и колко пъти да увеличим първият хълм. Този избор не е лесен и зависи от бъдещи блокиращи хълмове.

**Оптимален алгоритъм при фиксиран брой операции A**

Нека сме фиксирали, че операция А е изпълнена **a** пъти. В такъв случай Fi е увеличено с за всяко i. Използвайки новите стойности е лесно да конструираме алгоритъм, който намира минималния брой операции от тип B, които са нужни:

Започваме да симулираме движението на топчето отляво на дясно и всеки път когато открием ‘блокиращ’ хълм го намаляме с операция B докато можем да го прескочим. Тъй като броя операции А е фиксиран – нямаме ‘избор’ при блокиране. Алгоритъмът е със сложност O(n). Можем да използваме този алгоритъм като фиксираме всички възможни стойности за a, но това би било доста бавно – тъй като големините на хълмовете могат да са големи и съответно може отговорът също да е много голям.

**Решение за**

В този случай имаме, че:

От това следва, че операция A никога не е по-добрият избор. В такъв случай оптимално е да не я ползваме изобщо и е достатъчно да пуснем оптималния алгоритъм за 0 операции А. Решението има обща сложност O(n)

**Решение за**

Нека имаме дадено оптимално решение. Да вземем редица i1, i2, …, im , такава че в оптималното решение са извършени операции поне по веднъжи **m>1**. Да разгледаме как тези m операции изменят F:

* Fx не се променя за x<i1
* Fx се увеличава с за x=iy
* Fx се увеличава с за iy<x<iy+1

Лесно се вижда, че най-голямо увеличение претърпява im и то е със стойност:

Нека сега разгледаме алтернативно решение, в което заменяме тези m операции B с m операции А. Гореописаните промени не се състоят, а вместо тях F се променя просто като:

* Fx се увеличава с за всяко x.

*Твърдение: Алтернативното решение е не по-лошо от даденото оптимално.*

*Доказателство:*

Имаме:

Използвайки това:

Получаваме, че най-голямото възможно увеличение при първото решение е не по-голямо от увеличението на всяка от стойностите във второто решение. В такъв случай е ясно, че второто решение може да е само по-добро от първото.

Тъй като второто решение съдържа с m по-малко операции B, то значи започвайки с дадено оптимално решение, винаги можем да редуцираме броя операции B ако съществува описаната редица. Такава редица не съществува само ако броя операции B в решението е 0 или всичките операции B са направени на един и същ хълм (т.е. m=1). Ако имаме оптимално решение без използване на операции B – то може да бъде лесно намерено за линейно време, като просто сметнем колко операции А изисква всеки хълм за да бъде прескочен и вземем максимума от получените стойности. По-интересен е вторият случай:

***Решение когато всички операции от тип B са изпълнени на един и същ хълм.***

Нека **X** са броя операции А в оптималното решение, а **Y** са броя операции B. Да фиксираме хълма, на който са изпълнени всички операции B – нека това е хълм **K**.

Промяната на стойностите F спрямо началните е:

* Fi се увеличава с за i<K
* Fi се увеличава с за i=K
* Fi се увеличава с за i>K

Тъй като искаме всички стойности на F да са неотрицателни, то ни интересуват само най-малките стойности на F за всеки от горните три случая. Нека изберем такива **L** и **R**, че **първоначално**:

* за всяко , и
* за всяко , и

За да е валидно решението е нужно и достатъчно **след прилагане** на операциите да имаме:

Следователно, разглеждайки **първоначалните** стойности на F, получаваме три условия:

Трябва да изберем X, Y, така че горните условия да са изпълнени и X+Y да е минимално. Да си представим, че X е фиксирана стойност и да изведем условия за Y от второто и третото условие:

Тъй като X,Y>0, то при фиксирано X е оптимално да имаме:

В такъв случай имаме:

Нека положим:

Получаваме:

Ако разгледаме X като променлива то имаме максимум от три прави. Намирайки двете пресечните точки между тях можем да разделим равнината на 3 части и във всяка да намерим максимум на единствена права.

Единствените допълнителни ограничения, които трябва да спазим, са:

(от условие 1)

Тези ограничения не пречат на решението ни.

Така за O(1) можем да намерим оптималната стойност на X+Y при фиксиран хълм K. Пробвайки да фиксираме всеки хълм получаваме решение за O(n) при този случай на задачата. На всяка стъпка стойностите L и R могат да бъдат намирани за константно време чрез префиксни и суфиксни минимуми.

***Решение базирано на greedy***

Оказва се, че има по-лесен подход за решение чрез алчен алгоритъм.

Първото наблюдение, което трябва да направим, е, че върхът, който ще бъде намалян, е този с най-малка *F* стойност в началото. Така намираме нужните *FL, FK, FR* за *O(n)* и не е нужно да пробваме всеки хълм.

Второто наблюдение е, че следния подход ни дава оптимално решение.

1. Правим операция *A* докато *FL<0*
2. Правим операция *BK* докато
3. Правим операция *A* докато *FK<0* или *FR<0*

**Идея на решението**:

Първата част е очевидна – *FL* може да бъде променяно само чрез операция А. Нека след това си представим, че искаме да приложим всички операции *B* преди останалите операции *А* (редът на операциите никога няма значение). Нека *AK* е броя операции *A*, които ще са нужни, за да имаме *FK≥0*. Очевидно *AK*=. След операциите от тип *B*, броят операции от тип *A*, които са нужни ще е *MAX(AK, AR)* спрямо стойностите на *FK* и *FR*. Ако *AK>AR*, то няма как да е по-зле да направим още операции от тип *B*, тъй като всяка от тях със сигурност ще намали *AK* поне с 1. От друга страна, ако *AK<AR*, то със сигурност не трябва да правим повече операции от тип *B*, понеже операции от тип *А* ще намалят *AR* по-ефективно. В такъв случай правим операции от тип *B* докато не постигнем *AK≤AR*, след което довършваме с операции от тип *А*.

Трите стъпки могат да бъдат реализирани за *O(1)*, постигайки обща сложност *O(n).*

***Други (грешни) решения базирани на троично търсене***

Много привлекателна идея за състезателите навярно би била да направят троично търсене по броя операции *A*. Това наистина изглежда логично, тъй като колкото повече операции *А* правим, толкова по-малко операции *B* ще се налагат. Експериментално може да се види, че функцията наистина много прилича на такава, която подлежи на троично търсене, но за жалост не достатъчно. Решения базирани на тази техника могат да преминат тестовете ако успеят да “изгладят” функцията по различни начини, но това изобщо не е лесно без алгоритъмът да стане твърде бавен. Повечето такива решения се очаква да не преминат 5-та подзадача.

*Автор: Енчо Мишинев*