НОИЗ`09 А6

Едно от нестандартните нововъведения в състезателното програмиране е появата на задачи, при които не се изисква да се решат бързо (за оптимално време), а да се решат „по някакъв начин” с ограничена памет.

Една прясна такава задача е задачата на Пешо ☺, дадена на втория ден на Националната олимпиада.

Ето условието:

Задача А6. Мутирали жаби

Мутиралите в столичния район жаби са изгубили целия си разсъдък. След години живот, затрупани в боклуци, те търсят пътя към спасението. Булевардът, на който са живели, сега е плътно засипан с внимателно пакетирани бали боклук. По дължината на булеварда има N бали, номерирани отляво надясно от 0 до $N-1$, с положителни височини $H_i (0 \leq i < N)$, като на всяка бала стои по една жаба. Изморена от забързания си живот, жабата на i-тата бала може да направи не повече от $J_i (0 \leq i < N)$ скока, като всеки скок е към най-близката бала вдясно, която е строго по-висока от текущата(хем е към покрайнините на града, хем по високо над боклуците). Жаба, у която са останали сили за поне още един скок след като не са останали по-високи бали вдясно, успява да скочи в един по-добър свят, който е толкова високо, че не намерихме достатъчно голямо число за височината му – тази височина ще бележим с „-1”. Напишете програма **`frogs`**, която да намери до каква максимална височина може да достигне всяка от жабите.

Вход:

От първия ред на стандартния вход е зададен броят на балите $N (0 < N < 10^6)$. Вторият ред съдържа N естествени числа $H_i (0 < H_i < 10^9)$, разделени с по един интервал. Третият ред съдържа N естествени числа $J_i (0 < J_i < N)$, също разделени с по един интервал.

Изход:

На единственият ред на стандартния изход изведете N естествени числа, съответстващи на максималната височина, до която може да достигне всяка от жабите, в реда им от ляво на дясно. Разделете всяко от числата с по един интервал, като не оставяте интервал преди първото и след последното.

Примерен вход:	**Примерен изход:**
8 3 1 4 5 6 2 3 8 1 2 1 3 4 2 1 2	4 5 5 -1 -1 8 8 -1

Ограничения: 1s, 15MB.

Първата идея, която идва наум е да се организира дърво чрез масив от бащите и да се обхожда. Тази идея, комбинирана със двоично търсене, би могла да реши задачата (както ми стана известно от *Гибона*). Но ограничението за паметта тук изиграва решаваща роля – тази памет означава, че освен два масива за височините и дължините на скоковете, ние можем да си позволим още най-много 1 масив с тази дължина. Като имаме предвид, че все пак някъде трябва да запишем отговорите, това означава, че трудно ще можем да решим задачата без презаписване на междинна информация в някои от входните масиви.

Ключово наблюдение, което решава ефективно задачата е, че не се нуждаем от пълното дърво във всеки момент – ако изчисляваме резултатите от дясно на ляво, това което ни трябва във даден момент е само най-левият път от корен към листо в дървото, който може да се държи в стек. Така общото решение на задачата, малко приличащо на алгоритъма за намиране на изпъкнала обвивка, е: като се движим от дясно на ляво, и поддържаме стек със най-левият път на дървото на скоковете, всяка бала я закачаме на съответното място в дървото чрез изключване на по-малките бали от нея.

Но това още не е краят. За да се поберем във паметта, имаме два варианта – първият е да се използва допълнителен масив за стек, и резултатите да се записват на мястото на масива със скоковете. Вторият е стекът да се организира от дясно на ляво в самият масив на скоковете, като резултатите се записват в нов масив.

Това е и решението, което трябва да работи напълно за задачата.

Още по-интересно е, чрез комбинация на двете идеи за намаляване на паметта, задачата може да се реши дори без допълнителен масив! За новото решение ще трябват само около 9 мегабайта памет, което е доста по-малко от петнайсетте от условието.

Идеята на новото решение е да организираме стек от текущите увеличаващи бали от дясно но ляво в един т входните масиви (на височините или на скоковете), и да пазим резултатите в другия!

Примерен код 1 – C++

```
// TASK: FROGS, 9MB!
#include <iostream>
using namespace std;

const int max_n = 1000010;

int h[max_n], j[max_n], ht;
int main(){
	int i, k, n;
	scanf("%d", &n);

	for(i = 0; i < n; i++)
		scanf("%d", h + i);
	for(i = 0; i < n; i++)
		scanf("%d", j + i);

	j[n-1] = -1;
	ht = i = n-2;
	while(i >= 0){
		// remove from stack
		while(ht < n-1 && h[ht+1] != -1 && h[i] >= h[ht+1])
ht++;
		// push to stack
		h[ht--] = h[i];
		if(j[i] >= n-1-ht) j[i] = -1;
		else j[i] = h[ht+j[i]+1];
		i--;
	}
	for(i = 0; i < n; i++)
		printf("%d ", j[i]);
	printf("\n");
	system("pause");
	return 0;
}
```