

### Task 1. İp ucu

Doc Brown öz DeLorean avtomobilini zaman maşınına çevirməyi bacardı. İndi o daha çətin bir problemi həll etməyə çalışır: ən uzun ortaq alt ardıcılıq.  $N$  və  $M$  uzunluqlu  $A$  və  $B$  ardıcılığı verilib və o elə ən uzun  $C$  ardıcılığını tapmaq istəyir ki, onun içində gələn elementlər  $C'$ -də olduğu sırada həm  $A$  həm də  $B'$ -də olsun (yan yana olmalı deyillər). O, bir neçə gün işlədikdən sonra cavabı tapa biləcək bir proqram yazıb. Lakin ona cavab təcili lazımdır. Əvvəlcə o proqramı bir neçə gün işləməyə qoyub sonra geri qayıdıb, cavabı indiki zamana göndərmək istəyirdi. Lakin zamanda səyahət çox enerji alır və bütün cavabı göndərmək çox baha başa gələ bilər.

İndi doktorun yeni bir planı var, amma kodu sən yazmalısın. O, gələcəkdən indiyə bəzi ip ucları göndərəcək, sonra isə həmin ip uclarından istifadə edib optimal həlli tapacaq. Qeyd edək ki, optimal həll gələcəkdəki ilə eyni olmaq məcburiyyətində deyil.

hint.cpp proqramı yazaraq növbəti iki funksiyanın içini doldurmalısınız: *genHint* və *solve*, hansı ki bunlardan istifadə edərək doktor öz planını uğurla yerinə yetirəcək.

### İmplementasiya izahı

*genHint* funksiyanız növbəti formada olmalıdır:

```
std::vector<bool> genHint(const std::vector<int>& a, const std::vector<int>& b,
    const std::vector<int>& sol);
```

Bu funksiya yalnızca bir dəfə çağırılacaq və argument olaraq ona iki ardıcılıq və optimal həll veriləcək. O, bir hint (ip ucu) qaytarmalıdır, hansı ki həmin ip ucunu digər funksiya istifadə edəcəksiniz.

*solve* funksiyanız növbəti formada olacaq:

```
std::vector<int> solve(const std::vector<int>& a, const std::vector<int>& b,
    const std::vector<bool>& hint);
```

O, yalnızca bir dəfə çağırılacaq və argument olaraq ona iki ardıcılıq və digər funksiyanın qaytardığı ip ucu veriləcək. O, optimal bir cavab qaytarmalıdır.

Bu iki funksiya əlavə özünüzdən funksiyalar, class-lar, dəyişənlər və s. yarada bilərsiniz. Lakin, main funksiyası **olmamalıdır** və hint.h başlığı **istifadə olunmalıdır**. Qeyd edək ki, **qiymətləndirmə sistemi sizin funksiyalarınızı ayrı proseslər şəklində çağıracaq**. Yəni qlobal bir dəyişən yaradıb hər ikisində istifadə edə bilməyəcəksiniz.

### Lokal testing

Sizə Lgrader.cpp faylı verilib, hansı ki siz onu digər proqramlarla bir yerdə compile edib test edə bilərsiniz. Bu proqram  $N$  və  $M$  dəyişənlərini, ardınca isə  $A$  və  $B$  dəyişənlərini oxuyacaq. Bundan sonra optimal həllin uzunluğu olan  $K$  ədədini, daha sonra isə optimal həllin özü olan  $C$  ardıcılığını oxuyacaq. Proqram sizin *genHint* funksiyanızdan qayıdan ip ucunu və *solve* funksiyanızdan qayıdan cavabı çıxışa verəcək. Nəzərə alın ki rəsmi qiymətləndirmə sistemindən fərqli olaraq local grader sizin funksiyanızı ayrı proseslər şəklində işlətmir.

### Məhdudiyyətlər

$$1 \leq N, M \leq 10^5$$
$$0 \leq A_i, B_j < \min(N, M)$$

### Alt tapşırıqlar

Alt tapşırıq	Xal	$N, M$
1	10	$\leq 10^4$
2	90	$\leq 10^5$

Sizin həlliniz yalnızca alt tapşırıq üçün bütün testlər keçərsə tam xal alacaq.

### Xal sistemi

Alt tapşırıqdan alacağınız xal, sizin funksiyanızın qaytardığı ip ucunun uzunlu  $L$ -dən asılı olacaq. Alt tapşırıqdakı test üçün alacağınız xal aşağıdakı formul ilə hesablanacaq:

$$\min\left(\left(\frac{640}{L+1}\right)^{0.3}, 1\right)$$