Задача 1. Подсказка

Доктор Браун успя да превърне неговия DeLorean в машина на времето. Сега той се насочи към още по-голям проблем: най-дългата обща подредица. При дадени две редици числа A и B с дължини N и M, той иска да намери най-дългата (или една от най-дългите) редица C, такава че всички елементи на C участват в A и в B в същия ред (но не задължително последователно) както в C. Той успял да напише доста бавна програма, която ще работи доста дни докато намери решение. Той обаче се нуждае от отговор възможно най-скоро. Неговия първоначален план бил да остави програмата да работи и след няколко дена да изпрати резултата от нея назад във времето на сегашното си аз. Проблемът е, че пътуването във времето изисква огромно количество енергия, така че изпращането на пълното решение назад във времето ще бъде изключително скъпо.

Сега Докторът има нов план, но се нуждае от вщВашата помощ, за да я реализира. Той иска да изпрати кратка подсказка относно едно решение от бъдещето към настоящето и след това да използва тази подсказка да възстанови едно оптимално решение. Забележете, че не е задължително да възстанови същото оптимално решение като това изпратено в бъдещето. Вие трябва да напишете програма `hint.cpp`, която имплементира две функции: `genHint` и `solve`, които реализират плана на Доктора.

Детайли по имплементация

Вашата функция `genHint` трябва да има следния прототип:

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

Тя ще се извика само веднъж и ще получи като аргументи двете дадени редици и едно оптимално решение. Тя трябва да върне подсказка, която ще бъде подадена на другата функция.

Вашата функция `solve` трябва да има следния прототип:

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

Тя ще се извика само веднъж и ще получи като аргументи двете дадени редици и подсказката, която предишната функция е върнала. Тя трябва да върне едно оптимално решение.

Освен тези две функции вашата програма може да има всякакви помощни функции, класове, променливи и т.н. Все пак, тя **не трябва** да съдържа `main` функция и **трябва** да включва хедъра `hint.h`. **Забележете че на оценяващата система вашите две функции ще бъдат извикани в отделни инстанции на вашата програма, работещи в различни процеси**. Това означава, че Вие не може да споделяте глобални променливи между изпълнението на двете функции.

Локално тестване

Предоставен Ви е файлът `Lgrader.cpp`, които може да компилирате заедно с вашата програма, за да я тествате. Той ще чете N и M, последвани от A и B. След това ще прочете дължината на едно оптимално решение K и самото решение C. Ще отпечата дължината на подсказката от вашата `hint` функция, както и решението което вашата `solve` функция връща. Забележете, че за разлика от официалната оценяваща система локалният грейдър не пуска Вашите функции в отделни процеси.

Ограничения

$1 \leq N, M \leq 2 \times 10^5$
$0 \leq A_i, B_j < \min(N, M)$

Подзадачи

Подзадача	Точки	N, M
1	10	$\leq 10^4$
2	90	$\leq 2 \times 10^5$

Вашето решение ще получи точки за подзадача само ако премине успешно всички тестове в нея.

Оценяване

Оценката, която получавате за дадена подзадача зависи от максималната дължина L на подсказка, която Вашата програма е генерирала за тест в тази подзадача. Частта от точките, които ще получите за тази подзадача ще е:

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