

Task 1. Hint

Док Браун превратил свой Делориан в машину времени. Это было непросто, но теперь он хочет решить еще более сложную задачу: задачу о наибольшей общей подпоследовательности. Даны две последовательности целых чисел A и B с длинами N и M , он хочет найти наибольшую по длине последовательность C , такую, что все элементы C входят как в A , так и в B в том же порядке, что и в C (но не обязательно подряд). Если самых длинных последовательностей несколько, Док согласен найти любую из них. Он написал довольно-таки медленную программу, которая решает эту задачу, но есть проблема: программа работает несколько дней, а Доку нужен ответ как можно быстрее. Тогда Док придумал хитрый план: оставить программу работать несколько дней, а затем воспользоваться машиной времени и отправить себе в настоящее время решение задачи. К сожалению, проблема в том, что путешествия во времени – очень энергозатратная операция и отправить назад решение очень дорого.

Теперь у Дока новый план, но ему нужна ваша помощь, чтобы его реализовать. Он хочет отправить из будущего в настоящее небольшую подсказку к решению, а затем воспользоваться этой подсказкой, чтобы восстановить наибольшую общую подпоследовательность. Заметим, что его по-прежнему устроит любое оптимальное решение, не обязательно именно то, которое найдет его программа в будущем.

Напишите программу `hint.cpp`, реализующую две функции: `genHint` и `solve`, которая позволит Доку реализовать свой план.

Детали реализации

Функция `genHint` должна иметь следующую сигнатуру:

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

Она будет вызвана один раз и получит в качестве аргументов две заданных последовательности и оптимальное решение задачи о наибольшей общей подпоследовательности для них. Она должна вернуть подсказку, которая будет передана второй функции.

Функция `solve` должна иметь следующую сигнатуру:

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

Она будет вызвана один раз и получит в качестве аргументов две последовательности и подсказку, которую вернула первая функция. Она должна вернуть любую наибольшую общую подпоследовательность заданных последовательностей.

Кроме этих двух функций ваша программа может объявлять любые другие дополнительные функции, классы, переменные и так далее. Но, она **не должна** содержать функцию `main` и **должна** подключать заголовочный файл `hint.h`. **Обратите внимание, что при проверке ваши функции будут вызваны при разных запусках вашей программы.** Это означает, что они не смогут обмениваться какой-либо информацией через глобальные переменные.

Локальное тестирование

Для локального тестирования вам предоставляется файл `Lgrader.cpp`, который вы можете скомпилировать вместе со своей программой, чтобы ее тестировать. Он считывает со стандартного потока ввода числа N и M , а затем последовательности A и B . Затем он считывает длину оптимального решения

K и само оптимальное решение C . На вывод он отправляет длину подсказки, которую возвращает ваша функция `hint`, а затем решение, которое выдала ваша функция `solve`. Обратите внимание, что в отличие от официальной тестирующей системы локальный грейдер не пытается вызывать ваши функции в отдельных запусках.

Ограничения

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

Подзадачи

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

Ваше решение получит баллы за подзадачу только, если все тесты в этой подзадаче пройдены.

Scoring

Баллы, которые вы получите за подзадачу, зависят от максимальной длины L подсказки, которую ваше решение выдаст на тестах этой подзадачи. Доля баллов за подзадачу, которую вы получите, вычисляется по формуле:

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