

Завдання 1. Підказка

Доктору Брауну вдалося перетворити свій DeLorean на машину часу. Тепер він зосередився на ще більшій проблемі: найдовша спільна підпоследовність. Для заданих последовностей чисел A та B з довжинами N та M , він хоче знайти найдовшу (або одну з найдовших) последовність C , таким чином, що всі елементи C з'являються як в A так і в B в тому ж порядку (але не обов'язково последовно), як і в C . Йому вдалося написати дещо повільну програму, яка працюватиме багато днів, поки не знайде рішення. Однак йому потрібна відповідь якомога швидше. Його початковий план полягав у тому, щоб залишити свою програму запущеною, а потім за кілька днів надіслати її результат назад у часі до його теперішнього часу. Проблема в тому, що подорож у часі вимагає величезної кількості енергії, тому відправка повного рішення назад у часі була б надзвичайно дорогою.

Тепер у Доктора Брауна новий план, але для його реалізації йому потрібна ваша допомога. Він хоче надіслати коротку підказку про рішення з майбутнього до теперішнього часу, а потім використати цю підказку, щоб відновити оптимальне рішення за допомогою підказки. Зауважте, що це не обов'язково має бути таким самим оптимальним рішенням, як рішення майбутнього.

Ви повинні написати програму `hint.cpp` яка реалізує дві функції: `genHint` та `solve`, які досягають плану Доктора.

Деталі реалізації

Ваша функція `genHint` повинна мати наступний прототип:

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

Вона буде викликана лише один раз і отримає як аргументи дві задані последовності та оптимальне рішення. Вона повинна повернути підказку, яка буде надіслана вашій іншій функції.

Ваша функція `solve` повинна мати наступний прототип:

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

Вона буде викликана лише один раз і отримає як аргументи дві задані последовності та підказку, що повернула ваша інша функція. Вона повинна повернути оптимальне рішення.

Крім цих двох функцій, ваша програма може мати будь-які допоміжні функції, класи, змінні тощо. Проте, вона **не повинна** містити функцію `main` та **повинна** включати заголовний файл `hint.h`. **Зауважте, що в системі оцінювання ваші дві функції будуть викликатися в окремих екземплярах вашої програми, які виконуються в окремих процесах.** Це означає, що ви не зможете поділитися будь-яким глобальним станом між запусками двох функцій.

Локальне тестування

Вам надається файл `Lgrader.cpp`, який ви можете скомпілювати разом зі своєю програмою, щоб перевірити її. Він буде читати N та M , а потім A та B . Після цього він прочитає оптимальну довжину рішення K та оптимальне рішення C . Він виведе довжину підказки з вашої `hint` функції, а також рішення, яке повернула функція `solve`. Зауважте, що, на відміну від офіційної системи оцінювання, локальне оцінювання не запускає ваші функції в окремих процесах.

Обмеження

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

Підзавдання

Підзавдання	Бали	N, M
1	10	$\leq 10^4$
2	90	$\leq 10^5$

Ваше рішення отримає бали за підзавдання, лише якщо воно пройде всі в ньому тести.

Підрахунок балів

Оцінка, яку ви отримаєте за дане підзавдання, залежить від максимальної довжини підказки L , яку ваша програма згенерує для тесту в цьому підзавданні. Частина балів, яку ви отримаєте за підзавдання, становитиме:

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