XXVIII Всеукраїнська олімпіада з інформатики

Другий тур

1. Шоколадка

<https://www.e-olymp.com/uk/problems/7390>

У Марічки та Зеника є прямокутна плитка смачного українського шоколаду. Плитка складається з однакових квадратиків, кожен з яких може бути або чорним, або білим.

Марічка проведе довільну (можливо, нульову) кількість повних горизонтальних (зліва направо) розрізів плитки, тоді як Зеник — довільну (можливо, нульову) кількість повних вертикальних (зверху вниз) розрізів. Зауважте, що розрізи можна проводити тільки між квадратиками плитки, а також що Зеник і Марічка можуть зробити різну кількість розрізів. Після проведення всіх розрізів плитка розпадеться на певну кількість прямокутних шматків, які наші герої певним чином розділять між собою.

Оскільки Зеник дуже любить чорний шоколад, він хоче, щоб після всіх розрізів кількість цілих шматків, які складаються виключно з чорних квадратиків, була якомога більшою. Марічка хоче цю саму кількість зробити якомога меншою. Зауважте, що героїв не цікавить, якого саме розміру будуть шматочки: головне — їхня кількість.

Щоб усе було чесно, Андрій окремо та незалежно запитає Марічку та Зеника, в яких саме місцях потрібно провести розрізи, а потім сам замість них проведе ці розрізи.

**Завдання**

Напишіть програму **chocolate**, яка за заданими розмірами плитки, а також типом кожного її одиничного квадратика визначить, скільки після всіх розрізань утвориться шматочків, що складаються виключно з чорних квадратиків, — за умови оптимальних дій обох героїв.

**Вхідні дані**

Перший рядок вхідного файла **chocolate.dat** містить два цілих числа і (, ): висоту та ширину плитки відповідно. Далі йдуть рядків по символів в кожному, які позначають тип відповідного одиничного квадратика шоколадки. Символ **B** позначає чорний квадратик, а символ **W** — білий.

**Вихідні дані**

Єдиний рядок вихідного файла **chocolate.sol** повинен містити одне ціле число — кількість шматків, які складатимуться виключно з квадратиків чорного шоколаду.

**Оцінювання**

Набір тестів складається з 2 блоків:

1. 40 % балів: ,.
2. 60 % балів: на вхідні дані не накладають додаткових обмежень.

**Приклад вхідних та вихідних даних**

|  |  |
| --- | --- |
| chocolate.dat | chocolate.sol |
| 3 4  BWBB  BBBW  BBBW | 2 |

1. Гурток

<https://www.e-olymp.com/uk/problems/7391>

У деякій школі країни Олімпія проводиться гурток з інформатики, в якому займаються досвідчених програмістів та новачків. Тренер будує заняття, орієнтуючись на роботу в парі досвідченого учня та новачка. Провівши тренування, тренер визначив ефективність співпраці -го досвідченого програміста з -м новачком, що виражається числом . Загальна ефективність роботи гуртка дорівнює сумарному показнику ефективності співпраці для всіх пар за умови, що кожен учень працюватиме в парі, причому тільки в одній. Тренер хоче періодично проводити ротації пар, тому його цікавить питання: чи при довільному розбитті на пари ефективність роботи гуртка буде однаковою.

**Завдання**

Напишіть програму **section**, що за інформацією про ефективність співпраці кожної пари з досвідченого учня та новачка визначатиме, чи ефективність роботи гуртка відрізнятиметься залежно від того, як учнів розбито на пари.

**Вхідні дані**

Перший рядок вхідного файла **section.dat** містить натуральне число () — кількість тестів у файлі. Далі йде опис різних гуртків: в окремому рядку записано натуральне число () — кількість досвідчених програмістів та новачків у гуртку; потім іде рядків по цілих чисел через пропуск: -те число в -му рядку дорівнює () — ефективність роботи в парі -го досвідченого програміста з -м новачком.

**Вихідні дані**

Вихідний файл **section.sol** має містити чисел, записаних по одному в рядку: -те з них () — сумарна ефективність -го гуртка, якщо вона не залежить від розподілу учнів на пари, або -1 в іншому випадку.

**Оцінювання**

Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

1. 30 % балів: для всіх гуртків.
2. 20 % балів: для всіх гуртків.
3. 50 % балів: для всіх гуртків.

**Приклад вхідних та вихідних даних**

|  |  |
| --- | --- |
| section.dat | section.sol |
| 2  4  1 2 3 4  5 6 7 8  9 10 11 12  13 14 15 16  4  1 2 3 4  5 6 7 8  8 10 11 12  13 14 15 16 | 34  -1 |

1. Перестановка

Уявімо, що на дошці записано деяку перестановку чисел від до , а в Петриковому зошиті — кілька впорядкованих пар чисел, причому кожне число в межах від до і числа в парі не можуть бути однаковими. Якщо на дошці поряд стоять два числа, які в тому ж порядку утворюють одну з пар у зошиті, будь-яке з цих двох чисел на дошці Петрик може витерти. При цьому Петрик вважає, що чи́сла, які розділяло витерте число, тепер стоять поряд.

Завдання

Напишіть програму permutation, що для заданого набору пар чисел, виписаних у Петриковому зошиті, будує приклад початкової перестановки чисел, яку хлопець зможе шляхом послідовних витирань звести до єдиного (довільного) числа, або встановлює, що такої перестановки не існує.

Вхідні дані

У першому рядку вхідного файла permutation.dat записано натуральні числа та (, ), де — кількість упорядкованих пар, записаних у зошиті Петрика. Наступні рядків містять по два числа — елементи відповідних пар. Кожне число натуральне, не перевищує та відмінне від іншого числа в парі. Серед заданих упорядкованих пар немає однакових.

Вихідні дані

У перший рядок вихідного файла permutation.sol слід вивести будь-який приклад перестановки чисел від до , яку Петрику вдасться звести до одного числа, а у другий рядок потрібно вивести послідовність з числа, які одне за одним витиратиме Петрик. У випадку, якщо потрібної перестановки не існує, у вихідний файл треба вивести єдине число 0.

Оцінювання

Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

1. 20 % балів: .
2. 35 % балів: .
3. 45 % балів: .

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| permutation.dat | permutation.sol |
| 4 4  1 4  4 3  1 2  3 4 | 1 3 4 2  3 4 1 |

**Пояснення.** У перестановці можемо витерти трійку, бо маємо пару . Далі послідовність чисел на дошці стає такою: . Тепер можемо витерти четвірку, бо маємо пару . На дошці залишається пара чисел . Оскільки ця пара також є в Петриковому зошиті, можемо витерти, наприклад, одиницю, залишивши на дошці єдине число (в даному випадку — ).

4. Рядковий автомат

Рядковий автомат «Гомер-2015» приймає на вхід рядок символів і повертає деякий рядок як результат. В автомата є певний набір інструкцій — список правил заміни. Кожне правило складається з підрядка, який потрібно замінити, і рядка, на який цей підрядок потрібно замінити. Автомат послідовно робить такі операції: шукає першу в списку заміну таку, що поточний рядок містить відповідний підрядок, міняє перше входження цього підрядка на відповідний рядок-заміну і повторює все спочатку. Коли жоден підрядок у поточному рядку не міститься, автомат повертає цей рядок як результат. Список замін автомата називається *програмою*. Кількість правил замін називається *розміром програми*. Кількість зроблених замін на конкретних вхідних даних називається *кількістю операцій*.

Розглянемо приклад, коли на вхід автомат приймає рядок abcbca, а його програма розміру 2 має такий вигляд: 1) bac → c і 2) bc → cba. Порядок дій автомата такий:

1. Поточним рядком є abcbca.
2. Оскільки abcbca не містить підрядка bac з першого правила, автомат переходить до другого правила.
3. Оскільки abcbca містить підрядок bc з другого правила, автомат заміняє перше входження цього підрядка на cba і повертається на початок.
4. Поточним рядком є acbabca.
5. Оскільки acbabca не містить підрядка bac з першого правила, автомат переходить до другого правила.
6. Оскільки acbabca містить підрядок bc з другого правила, автомат заміняє перше (і єдине) входження цього підрядка на cba і повертається на початок.
7. Поточним рядком є acbacbaa.
8. Оскільки acbacbaa містить підрядок bac з першого правила, автомат заміняє перше (і єдине) входження цього підрядка на c і повертається на початок.
9. Поточним рядком є accbaa.
10. Оскільки accbaa не містить підрядка bac з першого правила, автомат переходить до другого правила.
11. Оскільки accbaa не містить підрядка bc з другого правила, автомат намагається перейти до наступного правила.
12. Оскільки правила закінчилися, автомат повертає рядок accbaa як результат.

Таким чином, з рядка abcbca автомат утворив accbaa. Кількість операцій у даному випадку дорівнює 3: замінили bc на cba, потім ще раз bc на cba, далі bac на c.

Завдання

Напишіть набір програм для рядкового автомата, що вирішують різноманітні задачі перетворення рядків (див. нижче). Для кожної підзадачі ви здаєте лише текст програми рядкового автомата. За кожну підзадачу ви отримуєте або повний бал за відповідну підзадачу, якщо програма пройшла всі її тести, або нуль, якщо хоча б один тест даної підзадачі не пройдено.

Формат програми

У кожному рядку файла міститься відповідне правило заміни: спочатку підрядок, який треба знайти, потім один пробіл, потім рядок, на який потрібно замінити підрядок. Якщо рядок заміни порожній, пробіл необов’язковий (інакше кажучи, у відповідному правилі можна вказати лише рядок, який потрібно видалити, і після нього пробіл не ставити). Між будь-якими двома сусідніми правилами для читабельності за бажанням можна вставити один порожній рядок. Після останнього правила можна як ставити, так і не ставити перенесення рядка.

Обмеження

У кожному правилі довжина як підрядка пошуку, так і рядка заміни не повинна перевищувати 10 символів (але в сумі вони можуть бути довшими); при цьому рядок заміни може бути порожнім, а підрядок пошуку — ні. Автомат завжди може оперувати такими символами: цифри, малі та великі латинські літери (причому малі та великі літери в розумінні автомата — різні символи), а також символ «плюс» (+). Ви можете користуватися всіма цими символами як у підрядках пошуку, так і в рядках заміни в усіх підзадачах. Розмір однієї програми не може перевищувати 100. Для жодних вхідних даних кількість операцій не повинна перевищувати 1000. Означення розміру програми та кількості операцій див. вище.

Перевірка за допомогою емулятора

В електронному варіанті умов наведено посилання для завантаження програми, що емулює рядковий автомат та надає допоміжну інформацію про перебіг виконання програми. Щоб використати емулятор, розташуйте його в окремому каталозі, створіть у цьому каталозі файл program.txt з текстом програми, а також файл input.txt, у який введіть вхідний рядок для вашої програми (можна додати або не додавати перенесення рядка в кінці файла), та запустіть програму-емулятор. Вона створить такі файли:

!output.txt: вихідний рядок автомата (якщо програма чи вхідні дані некоректні або кількість операцій перевищує обмеження, файл не буде створено / буде видалено).

!debug.txt: у кожному рядку цього файла вказано стан рядка після відповідної кількості операцій (якщо програма чи вхідні дані некоректні, цей файл не буде створено / буде видалено; якщо кількість операцій перевищує обмеження, файл міститиме стан рядка тільки до моменту, коли було перевищено обмеження).

!info.txt: у першому рядку цього файла вказано, чи є коректними програма та вхідні дані (а якщо вони некоректні, то чому); якщо програма та вхідні дані коректні, у другому рядку вказано кількість операцій виконаної програми; у третьому рядку вказано розмір програми; у четвертому рядку вказано довжину найдовшого рядка пошуку та довжину найдовшого рядка заміни.

Додатково емулятору можна передавати до чотирьох параметрів командного рядка: обмеження на кількість операцій, на розмір програми, на довжину рядка пошуку та на довжину рядка заміни. Наприклад, якщо запустити емулятор командою auto.exe 2000 100 15, обмеженням на кількість операцій буде встановлено 2000 (замість 1000); обмеження на розмір програми буде стандартним (100); обмеження на довжину рядка пошуку — 15; невказане в параметрах командного рядка обмеження на довжину рядка заміни також залишиться стандартним (10).

Підзадача 1 (5 балів)

На вході — рядок з цифр довжини від до включно. На виході — рядок із тих самих цифр у порядку від найменших до найбільших. Приклад: 9101 → 0119.

Підзадача 2 (15 балів)

На вході — рядок з символів I (велика латинська літера «i»), . На виході — десятковий запис числа . Приклад: IIIIIIIIIIII → 12. Відомо, що число не містить у своєму десятковому записі нулів.

Підзадача 3 (10 балів)

На вході — рядок з символів I (велика латинська літера «i»), . На виході — десятковий запис числа . Приклад: IIIIIIIIII → 10.

Підзадача 4 (20 балів)

На вході — десятковий запис числа , . На виході — рядок з символів I (велика латинська літера «i»). Приклад: 10 → IIIIIIIIII.

Підзадача 5 (20 балів)

На вході — рядок з малих літер a, b та c довжини від до включно. На виході — рядок з тих же літер, але великих і записаних у зворотному порядку. Приклад: abac → CABA.

Підзадача 6 (15 балів)

На вході — десятковий запис двох натуральних чисел, що не перевищують ; числа розділено символом +. На виході — десятковий запис суми відповідних чисел. Приклад: 4+19 → 23.

Підзадача 7 (15 балів)

На вході — десятковий запис двох натуральних чисел, що не перевищують ; числа розділено символом +. На виході — десятковий запис суми відповідних чисел. Приклад: 204+119 → 323.