XXIX Всеукраїнська олімпіада з інформатики

Другий тур

1. Робот

В країні Олімпії сьогодні свято!

Єдине поштове відділення на вулиці Олімпійській переповнене подарунками, які необхідно якомога швидше доставити їхнім адресатам. Всього цього дня надійшло *N* подарунків, *i*-й з яких необхідно доставити в будинок з номером *hi*. Декілька різних подарунків можуть бути адресовані в один будинок.

Поштове відділення розташоване на початку вулиці Олімпійської. Праворуч від пошти вздовж прямої дороги знаходяться будинки, нумерація яких починається з 1; *i*-й будинок розташований на відстані*i*кілометрів від пошти. Для доставки предметів в експериментальному режимі використовується єдиний спеціальний поштовий робот, який одночасно вміщує не більше ніж *K*предметів. Завантаження та відвантаження одного подарунка триває *одну* хвилину, два подарунки не можуть завантажуватися чи відвантажуватися одночасно. Один кілометр робот долає теж за *одну* хвилину.

**Завдання**

Напишіть програму robot, яка за інформацією про кожен з подарунків та характеристикою робота визначатиме найменший час у хвилинах, за який робот зможе доставити всі подарунки їхнім адресатам та повернутися у поштове відділення. Спочатку всі подарунки та робот знаходяться у відділенні.

**Вхідні дані**

Перший рядок вхідного файла robot.dat містить два цілих числа *N* і *K* (1 ≤ *N* ≤ 105, 1 ≤ *K* ≤ 103) — кількість подарунків, що надійшли до поштового відділення, та найбільшу кількість предметів, що може одночасно вмістити робот. Другий рядок містить *N* цілих чисел, записаних через пропуск: *i*-те число рівне *hi* (1 ≤ *hi* ≤ 106) — номер будинку, в який необхідно доставити *i*-й подарунок. Гарантується, що будинки із вказаними номерами справді розташовані на вулиці Олімпійській.

**Вихідні дані**

Єдиний рядок вихідного файла robot.sol повинен містити одне ціле число — найменший час у хвилинах, за який робот зможе доставити подарунки їхнім адресатам та повернутися у поштове відділення.

**Оцінювання**

Набір тестів складається з 4 блоків, для кожного з яких виконуються такі умови:

1. 25 % балів: 1 ≤ *N* ≤ 10, 1 ≤ *K* ≤ 5, 1 ≤ *hi* ≤ 102.
2. 25 % балів: 1 ≤ *N* ≤ 103, 1 ≤ *K* ≤ 102, 1 ≤ *hi* ≤ 104.
3. 25 % балів: 1 ≤ *N* ≤ 105, 1 ≤ *K* ≤ 103, 1 ≤ *hi* ≤ 104.
4. 25 % балів: 1 ≤ *N* ≤ 105, 1 ≤ *K* ≤ 103, 1 ≤ *hi* ≤ 106.

**Приклад вхідних та вихідних даних**

|  |  |
| --- | --- |
| robot.dat | robot.sol |
| 7 3  3 9 3 2 1 1 3 | 40 |

**Пояснення.** Пронумеруємо подарунки від 1 до 7. Тоді один з оптимальних планів доставки має такий вигляд:

* 2 хвилини: завантаження 5 і 6 подарунків (обидва потрібно доставити у будинок 1),
* 1 хвилина: переміщення робота від відділення до будинку 1,
* 2 хвилини: відвантаження 5 і 6 подарунків,
* 1 хвилина: переміщення робота від будинку 1 до відділення,
* 2 хвилини: завантаження 2 і 7 подарунків (2 подарунок треба доставити в будинок 9, 7 подарунок — в будинок 3),
* 9 хвилин: переміщення робота від відділення до будинку 9,
* 1 хвилина: відвантаження 2 подарунку,
* 6 хвилин: переміщення робота від будинку 9 до будинку 3,
* 1 хвилина: відвантаження 7 подарунку,
* 3 хвилини: переміщення робота від будинку 3 до відділення,
* 3 хвилини: завантаження 1, 3 та 4 подарунків (1 і 3 подарунки треба доставити до будинку 3, 4 подарунок — до будинку 2),
* 2 хвилини: переміщення робота до будинку 2,
* 1 хвилина: відвантаження 4 подарунку,
* 1 хвилина: переміщення робота від будинку 2 до будинку 3,
* 2 хвилини: відвантаження 1 і 3 подарунків,
* 3 хвилини: переміщення робота від будинку 3 до відділення.

1. Метро

Мер столиці Котоляндії врешті погодився побудувати в місті мережу метро. Тепер його очікує надзвичайно важливе завдання — зекономити якомога більше грошей на побудові цього метро.

Мережа метро буде складатись із певної кількості станцій, сполучених двосторонніми тунелями. Між будь-якою парою станцій повинен бути шлях по тунелях (можливо, через інші станції), при цьому кількість побудованих тунелів має бути мінімальною можливою. Зауважте, що за таких умов існує єдиний шлях між будь-якою парою станцій.

Пасажири повинні будуть заплатити певну кількість грошей для того, щоб увійти в метро на певній станції. При цьому для різних станцій ця сума може бути різною. А саме: для деякої станції *X* вартість входу буде рівна (в гривнях) максимальній кількості тунелів, які пасажир має проїхати для того, щоб доїхати від станції *X* до будь-якої іншої станції *Y*.

Крім того, мер міста дуже прискіпливий до чисел. Відомо, що мер любить числа *A*1, ..., *AN*, а також ненавидить числа *B*1, ..., *BM*. Мер не погодиться на побудову метро, якщо серед вартостей входу на станції будуть не всі його улюблені числа або буде хоча б одне, яке він ненавидить.

**Завдання**

Напишіть програму metro, яка за заданими числовими вподобаннями мера знаходитиме мінімальну кількість станцій, яку може містити метро, або визначатиме, що мережу із заданими вимогами побудувати неможливо.

**Вхідні дані**

Перший рядок вхідного файла metro.dat містить два цілих числа *N* та *M* — кількість чисел, які мер любить та ненавидить відповідно. Другий рядок містить список із *N* цілих чисел, розділених пропусками, — список чисел, які мер любить. Третій рядок містить в аналогічному форматі список із *M* чисел, які мер ненавидить.

**Вихідні дані**

У єдиному рядку вихідного файла metro.sol виведіть мінімальну кількість станцій, яку може містити метро. Якщо неможливо побудувати мережу зі заданими вимогами, виведіть одне число -1.

**Оцінювання**

Набір тестів складається з 2 блоків, для яких виконуються такі умови:

1. 50 % балів: 1 ≤ *N*, *M*, *Ai*, *Bi* ≤ 2000.
2. 50 % балів: 1 ≤ *N*, *M*, *Ai*, *Bi* ≤ 100000.

**Приклади вхідних та вихідних даних**

|  |  |
| --- | --- |
| metro.dat | metro.sol |
| 2 3  4 7  13 1 3 | 8 |
| 2 1  4 7  6 | -1 |

1. Лаврушка і розбиття масиву

Лаврушка — сумлінний учень, який мріє стати програмістом. На останньому уроку інформатики його улюблена вчителька запропонувала йому розв’язати таке завдання. Нехай *a*1, ..., *aN* — певна послідовність натуральних чисел. Лаврушці потрібно розділити послідовність чисел 1, 2, 3, ..., *N* на дві послідовності *b*1, ..., *bM* і *c*1, ..., *cK* таким чином, щоб:

* Кожне натуральне число 1 ≤ *r* ≤ *N* було елементом рівно однієї з послідовностей *b* або *c* (тому   
  *M* + *K* = *N*).
* Для кожної пари індексів 1 ≤ *i*, *j* ≤ *M*, *i* ≠ *j*, числа *abi* і *abj* були взаємно простими.
* Для кожної пари індексів 1 ≤ *i*, *j* ≤ *K*, *i* ≠ *j*, числа *aci* і *acj* були взаємно простими.

Взаємно простими називаються числа, найбільший спільний дільник яких дорівнює одиниці. Назвемо відповідний поділ послідовності 1, 2, 3, ..., *N* *розбиттям* послідовності *ai*.

Розбиття послідовності може бути неоднозначне. Тому вчителька попросила Лаврушу знайти таке розбиття, що максимізує кількість елементів у послідовності *bi*. Якщо існує кілька розбиттів, що максимізують кількість елементів у послідовності *bi*, то потрібно знайти серед них таке, щоб послідовність *bi* була лексикографічно найменшою.

Будемо казати, що послідовність *q*1, *q*2, ..., *qW* є лексикографічно меншою за послідовність *p*1, *p*2, ..., *pW*, якщо існує таке *i*, що *qi* < *pi*, а для довільного *j* < *i* виконується *pj* = *qj*.

**Завдання**

Напишіть програму split, яка для заданої послідовності чисел *a* знайде оптимальне розбиття послідовності чисел 1, 2, 3, …, *N* на дві послідовності *b*1, ..., *bM* і *c*1, ..., *cK*.

**Вхідні дані**

У першому рядку вхідного файла split.dat записано число *Z* (1 ≤ *Z* ≤ 3) — кількість тестових вхідних даних (учителька кілька разів пропонувала Лаврушці розв’язати це завдання для різних вхідних даних).

Далі подаються *Z* тестових даних у такому форматі.

У першому рядку даних міститься число *N* (1 ≤ *N* ≤ 100000) — кількість елементів у послідовності *a*. У наступному рядку записано *N* чисел, розділених одиничними пропусками, — елементи послідовності *a* (1 ≤ *ai* ≤ 2000000).

**Вихідні дані**

Для кожних тестових даних виведіть у вихідний файл split.sol в одному рядку число *M* — кількість елементів у послідовності *b*, а у другому — *M* натуральних чисел — у порядку зростання номери елементів послідовності *a*, що увійшли до послідовності *b*.

Якщо так трапиться, що вчителька помилилась і не існує жодного розбиття послідовності *a*, то виведіть в одному рядку -1.

**Оцінювання**

Набір тестів складається з 4 блоків, для кожного з яких виконуються такі умови:

1. 24 % балів: *N* ≤ 15, 1 ≤ *ai* ≤ 2000000.
2. 24 % балів: *N* ≤ 1000, 1 ≤ *ai* ≤ 2000000.
3. 30 % балів: *N* ≤ 20000, 1 ≤ *ai* ≤ 2000000.
4. 22 % балів: *N* ≤ 100000, 1 ≤ *ai* ≤ 2000000.

**Приклад вхідних та вихідних даних**

|  |  |
| --- | --- |
| split.dat | split.sol |
| 2  5  1 2 3 4 5  5  2 3 4 5 6 | 4  1 2 3 5  -1 |

**Пояснення.** У першому наборі тестових даних не можна отримати розбиття, де в послідовності *b* містилися б усі елементи 1, 2, ..., *N*, тому що числа 2 і 4 не взаємно прості. А в наступному наборі тестових даних взагалі не існує жодного розбиття послідовності *a*.

4. Камінчиковий автомат

Камінчиковий автомат «Мардж-2016» працює на квадратній таблиці, розбитій на комірки, у кожній з яких у довільний момент часу лежить певна (можливо, нульова) кількість камінців. На вхід автомат приймає два натуральних числа *A* та *B* у вигляді кількості камінців, розташованих у початковий момент часу в лівій верхній та у правій верхній комірках таблиці відповідно. У решті комірок у початковий момент часу камінців немає. Кожна комірка має прикріплену до неї функцію: ця функція приймає на вхід кількість камінців, розташованих на даний момент у комірці, а повертає кількості камінців, які на наступній ітерації необхідно перекласти з цієї комірки у сусідні з нею верхню, праву, нижню та ліву клітинки таблиці. Таким чином, на кожній ітерації камінчиковий автомат просто перекладає відповідні кількості камінців з усіх комірок у сусідні відповідно до інструкцій, які повернули прикріплені до комірок функції. Усі перекладання здійснюються водночас. Якщо під час чергової ітерації жодна комірка не переклала жодного камінця у жодну сусідню комірку, після цієї ітерації виконання «програми» завершується. На виході автомат повертає два числа: кількість камінців у лівій нижній та правій нижній комірках. Кількості камінців, які на момент завершення програми містили всі інші комірки, можуть бути довільними та ігноруються.

Завдання

Дана задача має 5 підзадач:

1. У першій підзадачі на виході потрібно отримати (0, *A* + *B*), тобто у лівій нижній комірці камінців бути не має, а в правій нижній кількість камінців має дорівнювати сумі чисел на вході. Ця підзадача має вартість **10 балів**.
2. У другій підзадачі на виході потрібно отримати (0, |*A* – *B*|), тобто у правій нижній комірці кількість камінців має дорівнювати модулю різниці чисел на вході (а в лівій нижній комірці камінців бути не має). Ця підзадача має вартість **20 балів**.
3. У третій підзадачі на виході необхідно отримати (0, min{*A*, *B*}), де min{*A*, *B*} позначає менше з двох чисел *A* та *B* (якщо *A* = *B*, то min{*A*, *B*} = *A* = *B*). Ця підзадача має вартість **15 балів**.
4. У четвертій підзадачі на виході необхідно отримати (0, max{*A*, *B*}), де max{*A*, *B*} позначає більше з двох чисел *A* та *B* (якщо *A* = *B*, то max{*A*, *B*} = *A* = *B*). Ця підзадача має вартість **15 балів**.
5. У п’ятій підзадачі на виході потрібно отримати (1, 0) за умови *A* > *B*; (0, 1) за умови *A* < *B*; (0, 0) за умови *A* = *B*. Інакше кажучи, навпроти того місця, де на вході розташовувалося строго більше з двох чисел, на виході має розташовуватися один камінець. Ця підзадача має вартість **40 балів**.

Напишіть функцію/процедуру automaton, що за кількістю камінців у комірці таблиці та її координатами визначить, скільки камінців треба перекласти у сусідні з нею клітинки відповідно до вимог підзадачі. Інакше кажучи, вам необхідно реалізувати алгоритм роботи камінчикового автомата для кожної з п’яти підзадач. Усі підзадачі складаються з окремих блоків тестів; кожен блок має вартість 5 балів і може містити один або кілька тестів. При цьому, щоб заробити відповідні 5 балів, ваш алгоритм має пройти всі тести з блоку.

Деталі реалізації

Ви маєте надіслати файл, що містить реалізацію функції/процедури automaton та, за потреби, інший код, необхідний для коректної роботи цієї функції, але не містить самого тіла програми (тобто функції main у C++ чи блоку begin/end. у Pascal’і). При тестуванні ваш файл буде доповнено спеціальним тілом програми, написаним журі. Тіло відповідним чином викликатиме написану вами функцію та перевірятиме коректність алгоритму, який вона втілює. Реалізована вами функція повинна мати такий вигляд (сигнатури функції для кожної з доступних мов програмування див. в електронній версії умов):

automaton(subproblem, size, row, column, pebbles, top, right, bottom, left)

Параметри функції

1. subproblem — номер підзадачі (див. вище).Номер підзадачі не змінюється під час запусків функції на одній і тій самій програмі.
2. size — розмір квадратної таблиці (кількість клітинок в одній горизонталі/вертикалі). Розмір не змінюється під час запусків функції на одній і тій самій програмі.
3. row — рядок, у якому розташована комірка; нумерація від 1 до size зверху вниз: 1 — верхній рядок, size — нижній рядок.
4. column — стовпець, у якому розташована комірка; нумерація від 1 до size зліва направо: 1 — лівий стовпець, size — правий стовпець.
5. pebbles — кількість камінців у комірці, додатне ціле число (якщо ця кількість нульова, жодних перекладань здійснити не можна, і виклику функції не відбувається). Ви можете змінити значення цієї змінної усередині функції, але така зміна не матиме жодного ефекту: кількість камінців після перекладань автоматично стане рівною різниці між початковою кількістю камінців у клітинці та кількістю перекладених у сусідні комірки камінців (див. далі).
6. top — початкове значення цього параметра (аргумент, який передає тіло програми), дорівнює 0. Якщо у даної клітинки є сусідня зверху і туди необхідно перекласти камінці з даної, це значення слід переписати, зробивши рівним відповідній кількості камінців.
7. right — початкове значення цього параметра дорівнює 0. Якщо у даної клітинки є сусідня справа і туди необхідно перекласти камінці з даної, це значення слід переписати, зробивши рівним відповідній кількості камінців.
8. bottom — початкове значення цього параметра дорівнює 0. Якщо у даної клітинки є сусідня знизу і туди необхідно перекласти камінці з даної, це значення слід переписати, зробивши рівним відповідній кількості камінців.
9. left — початкове значення цього параметра дорівнює 0. Якщо у даної клітинки є сусідня зліва і туди необхідно перекласти камінці з даної, це значення слід переписати, зробивши рівним відповідній кількості камінців.

Кінцеві значення параметрів top, right, bottom, left повинні бути невід’ємними та в сумі не мають перевищувати значення pebbles.

На кожному кроці функція повинна визначати перекладання, виходячи **виключно з даних, переданих їй на цьому кроці**. На результат роботи функції жодним чином не повинна впливати попередня взаємодія з тілом програми. Крім того, функція повинна бути детермінованою, тобто для сталих вхідних даних завжди повертати одні й ті самі значення.

Обмеження

* Розмір таблиці може бути довільним числом від 5 до 10 включно.
* Числа на вході можуть мати довільні значення від 1 до 100 включно.
* Кількість ітерацій, після яких автомат зупиниться, не має перевищувати 10 000.

Власноручне тестування

В електронному варіанті умов наведено приклад основного тіла програми automaton\_tester.\*, що запускає функцію automaton на вхідних даних, заданих у створеному вами вхідному файлі, і виводить кінцевий стан таблиці (або стан таблиці після 10 000 ітерацій, якщо автомат досі не зупинився). Щоб використати цю програму, розташуйте її в одному каталозі з файлом automaton.\*, який містить вашу реалізацію функції, та створіть у цьому ж каталозі файл automaton.dat зі структурою, описаною в наступному абзаці. Зауважте, що основне тіло програми, яке буде використано для оцінювання вашої функції, відрізнятиметься від наданого вам у automaton\_tester.\* прикладу. Зокрема, під час перевірки функцію може бути викликано для довільної клітинки таблиці та довільної кількості камінців від 1 до 200.

automaton\_tester: вхідні дані

Єдиний рядок містить чотири натуральних числа: відповідно номер підзадачі, розмір таблиці, число *A* та число *B*.

automaton\_tester: вихідні дані

Перший рядок вихідного файла міститиме пару чисел на виході автомата або словесну діагностику помилки в разі, якщо на якомусь кроці функція порушила технічні вимоги чи обмеження на кількість ітерацій. Якщо технічних порушень не було, наступні рядки міститимуть повний кінцевий стан таблиці.

automaton\_tester: приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| automaton.dat | automaton.sol |
| 1 5 3 4 | 0 7  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 7 |