Таня започна да учи програмиране и нейната поредна задача е да напише програма, която да сортира въведена пермутация на числата от *1* до *N*. “Тривиално!“ тя си помисли. Може да използва някой прост метод например метод на мехурчето или, ако скоростта е от значение, някой по бърз.

Оказва се обаче, че задачата далеч не е толкова тривиална, а именно защото всяка размяна на два елемента си има цена. Целта на задачата всъщност не е просто редицата да бъде сортирана, а това да се извърши за минимална обща цена.

Цената на дадена размяна зависи от две неща – позициите, между които се извършва размяна (всяка двойка позиции има някаква конкретна цена), но също и самите стойности, които биват разменяни (отново всяка двойка стойности има някаква конкретна цена). Т.е. Цената на размяната е сумата на тези две. Цената на размяна на два елемента не зависи от реда, в който те са дадени – . Също така е винаги вярно.

В някои случаи цените са произволни, но обикновено те не са. Всъщност разглеждайки една от двете таблици с цени откриваме 5 различни типа:

* 1. *произволно число между 1 и 4N*
  2. *(закръглено до най-близкото цяло число)*

Тези типове характеризират и цените по позиции, и цените по стойности. Когато става въпрос за цени по позиции, *i* и *j* означават позициите (номерирани от *1* до *N*), а когато става въпрос за цени по стойности, *i* и *j* означават самите числа на тези позиции.

Помогнете на Таня, като напишете програма, която да сортира дадената пермутация чрез поредица от размени на двойки елементи за възможно най-ниска цена.

**Вход**

От първия ред на файла sorting.in се въвежда едно цяло положително число *N*. От втория ред се въвежда пермутацията, която Таня трябва да сортира. От следващия ред се въвежда типът на цените по позиции (цяло число от 0 до 4). От следващите *N* реда се въвеждат по *N* числа на ред – цените по позиции. От следващите *N+1* реда се въвеждат еквиваленти данни, но за цените по стойности.

**Изход**

На първия ред на изходния файл sorting.out трябва да се изведе едно цяло неотрицателно число *K* – броят размени, които програмата ви използва за сортиране на пермутацията. На следващите *K* реда да се изведат по две числа – позициите на двата елемента, които да се разменят на съответната стъпка.

**Оценяване**

Ще получите 0 точки, ако изходът ви е невалиден. Невалиден е всеки изход, който отговаря на поне едно от следните условия:

* *K* е отрицателно или надвишава *1 000 000*.
* Броят изведени размени е по-малък от *K*.
* Позициите в размените надвишават *N* или са по-малки от *1*.

Ако изходът Ви е валиден, ще получите процента от точките, предвидени за съответния тест. Дефинираме yourScore като резултатът, който е получилa Вашата програма, а minScore като най-малкият резултат, който е получила някоя от програмите на участниците.

**Ограничения**

**Ограничение по време: 3 сек**

**Ограничение по памет: 256 MB**

**Подзадачи**

Типът на цените по позиции ще бъде 1, 2, 3 или 4, а типът на цените по стойности ще бъде 0, 1, 2, 3 или 4. Всяка от тези 20 (4\*5) комбинации ще се среща равен брой пъти, т.е. всяка носи по 5% от общия брой точки. *N* винаги ще бъде равно на 400.

**Примерен тест**

|  |  |
| --- | --- |
| **Вход (sorting.in)** | **Изход (sorting.out)** |
| 5  1 5 3 2 4  1  0 17 3 12 16  17 0 3 6 1  3 3 0 11 8  12 6 11 0 3  16 1 8 3 0  1  0 10 5 6 17  10 0 16 4 17  5 16 0 4 20  6 4 4 0 14  17 17 20 14 0 | 2  4 5  2 5 |

**Обяснение на тестовия пример**

В началото редицата е 1 5 3 2 4 и се разменят четвъртият и петият елемент. Техните стойности са 2 и 4 съответно. От първата таблица виждаме, че цената по позиции е 3. От втората, че тази по стойности е 4. След тази размяна редицата е 1 5 3 4 2 и се разменят вторият и петият, чиито стойности са 5 и 2 съответно. Цената по позиции е 1, а тази по стойности е 17. Общата цена е 25.

Възможно е да сортираме редицата и с размените 2 4 и след това 4 5, но тогава цената е по-висока – 40.