Общее описание задач

№	Название задачи	Ограничение на объем используемой памяти	Ограничение на время выполнения программы, секунды	Количество баллов, присвоенных задаче
1.	Joc	≤ 256 Mb	≤ 0,5	100
2.	Zboruri	≤ 256 Mb	≤ 0,5	100
3.	Avioane	≤ 256 Mb	≤ 2,0	100

Примечание. При равенстве общего количество очков, приоритет будет отдан участнику, набравшему первым соответствующее количество очков.

Joc

На свой день рождения Арина получила видеоигру. Игра состоит из матрицы с *N* строками и *M* столбцами со значениями *0* и *1*. Блок определяется как подматрица размером *2*x*2*, содержащая одинаковые значения, *0* или *1*. Некоторые значения могут быть частью нескольких блоков. У Арины есть четыре кнопки: ***вверх, вниз, вправо, влево***, которые соответственно перемещают строки или столбцы циклически в соответствующем направлении кнопки. Например, операция **вверх** переместит все строки на одну позицию вверх, и первая строка становится последней. Соответственно, операция **вправо** переместит все столбцы вправо, и последний столбец станет первым. Цель игры - получить как можно больше блоков, используя эти четыре кнопки.

Задача. Разработайте программу, которая, используя четыре кнопки, упорядочивает значения в матрице так, чтобы количество образованных блоков было максимальным.

Входные данные. Первая строка стандартного ввода содержит два целых числа *N* и *M*, разделенных пробелом, количество строк и столбцов матрицы. Каждая из следующих *N* строк содержит *M* значений *0* или *1* - значения матрицы (без пробелов между значениями).

Выходные данные. Первая и единственная строка стандартного вывода будет содержать максимальное количество блоков, которое Арина может получить.

Ограничения. $2 \le N, M \le 1000$. Ограничения по времени выполнения и объему используемой памяти даны в общем описании задач, предложенных для решения. Исходный файл будет иметь название `joc.pas`, `joc.c` или `joc.cpp`.

Пример 1.

Вход

```
2 4
0110
0110
```

Выход

```
2
```

Объяснение: Одно из решений - это движение вправо, которое сдвинет последние значения вправо на одну единицу, а последний столбец станет первым столбцом. В результате мы получим 2 блока:

```
0011
0011
```

Пример 2.

Вход

```
3 2
11
11
11
```

Выход

```
2
```

Объяснение: В этом примере все начальные значения расположены таким образом, что образуют 2 блока. Строки 1 и 2 формируют один блок, строки 2 и 3 формируют второй блок. Значения в строке 2 являются частью обоих блоков.

Пример 3.

Вход	*Выход*
3 3 000 010 000	3

Объяснение: Одно из решений - нажать кнопки вправо и вниз. Это перенастроит матрицу следующим образом:

```
000
000
001
```

Мы видим, что значения с 0 оптимально распределены и формируют 3 блока

Оценка. Тесты будут сгруппированы в следующие категории:

A. Для 35% тестов: $2 \leq N, M \leq 50$.
B. Для 30% тестов: $2 \leq N, M \leq 300$.
C. Для остальных 35% тестов: $2 \leq N, M \leq 1000$.

Zboruri

В определённой стране есть *N* городов, обозначенных номерами *1, 2, ..., N* и соединённых между собой двусторонними авиарейсами. В городе есть диспетчерская, которая управляет расписанием авиарейсов. Каждый день диспетчерская вносит изменения в расписание авиарейсов следующим образом:

- Выбирается город X.
- Добавляются рейсы из города X во все остальные города, куда рейсы не осуществляются.
- Все существующие ранее рейсы из города X отменяются.

Например, если из Города 5 есть рейсы в Города 1 и 2, но нет рейсов в Города 3 и 4, то после внесённых изменений будут рейсы из Города 5 в Города 3 и 4, а рейсы в Города 1 и 2 будут отменены.

Граждане этой страны интересуются, могла ли бы диспетчерская создать полное расписание авиарейсов, то есть расписание, в котором будет прямой рейс между каждой парой различных городов.

Задача. Напишите программу, которая определяет, может ли диспетчерская создать полное расписание авиарейсов, то есть каждая пара городов была бы соединена прямыми рейсами, независимо от последовательности добавления и отмены рейсов.

Входные данные. Первая строка стандартного ввода содержит целое число *N* – количество городов. Вторая строка содержит целое число *M*, представляющее количество текущих рейсов. Каждая из следующих M строк содержит два различных числа, разделённых пробелом – метки городов, которые в настоящее время соединены.

Выходные данные. На одной строке будет выведен ответ «DA» или «NU», соответственно, может ли диспетчерская создать полное расписание авиарейсов или нет.

Ограничения. $2 \leq N \leq 1\,000,\ 0 \leq M < N * (N-1)/2$. Ограничения по времени выполнения и объему используемой памяти даны в общем описании задач, предложенных для решения. Исходный файл будет иметь имя `zboruri.pas`, `zboruri.c` или `zboruri.cpp`.

Пример 1.

Вход

```
2
0
```

Выход

```
DA
```

Объяснение: В данном примере 2 города и ни одного текущего рейса. На следующий день будет прямой рейс между этими двумя городами.

Пример 2.

Вход

```
3
2
1 2
2 3
```

Выход

```
NU
```

Объяснение: В данном примере 3 города и 2 текущих рейса. Можно заметить, что в четные дни будут рейсы 1-2 и 2-3, а в нечетные дни будет рейс 1-3, так что никогда не будет всех рейсов в один и тот же день.

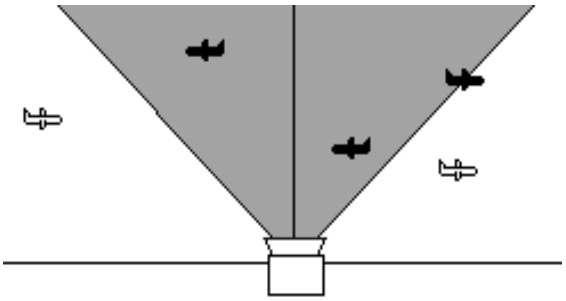
Оценка. Тесты будут сгруппированы в следующие категории:

- Для 15% тестов: $N \leq 15$.
- Для других 23% тестов: $N \leq 25$.
- Для других 53% тестов: $N \leq 500$.
- Для остальных тестов (9%) нет дополнительных ограничений.

Avioane

Алекс любит самолеты. Однажды он решил сделать фото с как можно большим количеством самолетов.

Все самолеты летают в одной плоскости с одинаковой скоростью одна единица в секунду. Всего есть N самолетов, характеризующихся тремя целыми числами, h_i - высота полета самолета относительно земли, x_i - позиция, где находится самолет в момент времени $t = 0$, и d_i, которое равно 0, если самолет летит влево, и 1, если вправо. Гарантируется, что каждый самолет пройдет через точку $(0, h_i)$ в момент времени $t > 0$.



Алекс выберет, куда поставить камеру $(x_0, 0)$, направленную прямо вверх. Угол обзора камеры составляет 45 градусов от вертикальной оси. Все самолеты, находящиеся в зоне, описываемой углом, попадут в кадр.

После выбора позиции Алекс может сделать снимок в любой момент времени.

Задача. Напишите программу, которая определяет максимальное количество самолетов, которые могут попасть в фотографию.

Входные данные. Первая строка стандартного ввода содержит целое число N - количество самолетов. Следующие N строк содержат по три целых числа, разделенных пробелом h_i, x_i и d_i - высоту, позицию и направление самолета i.

Выходные данные. Первая и единственная строка стандартного вывода будет содержать максимальное количество самолетов, которые могут попасть в фотографию Алекса.

Ограничения. $1 \le N \le 3*10^5$; $1 \le h_i, x_i \le 10^9$; $d_i = 0$ или 1. Ограничения, касающиеся времени выполнения и объема используемой памяти, даны в общем описании задач, предложенных для решения. Исходный файл будет иметь название `avioane.pas`, `avioane.c` или `avioane.cpp`.

Пример 1.

Вход

```
3
3 -4 1
4 -12 1
2 3 0
```

Выход

```
2
```

Объяснение: Алекс может разместить камеру на координатах (0, 0) и сделать снимок в момент времени t = 1. Тогда в кадре окажутся самолеты 1 и 3. Нет никакого способа попасть в кадр всем трем самолетам.

Пример 2.

Вход	*Выход*
5 2 -2 1 1 1 0 3 8 0 1 4 0 4 -8 1	4

Оценка: Тесты будут сгруппированы в следующие категории:

- Для 5% тестов, $N \leq 10$
- Для 21% тестов, $N \leq 200$
- Для 33% тестов, $N \leq 2000$
- Для 41% тестов, без дополнительных ограничений