## Задача Food

Кольо обича да си похапва. Ако трябва да сме честни – обича да си купува похапване и да не го използва пълноценно. Всеки път поръчва повече храна отколкото корема му може да поеме, което разбира се не е оптимално за неговите финанси (не че му пука особено, де).

Всеки тип храна има следните показетели – **име**, **тегло W** **грама** и **вкусотия C**. На Кольо корема може да побере максимум **M** грама храна общо. Целта е да се намери от **N** на брой храни, тази част **K** от тях, които имат максималната обща вкусотия, но така че да се съберат в корема на Кольо и общото им тегло да не надвишава **M**.

Оптимизирайте начина на хранене на Кольо, така че да похапва най-вкусните храни от възможните.

### Входни данни

Входните данни ще се четат от конзолата.

На първия ред ще се съдържа числото **M** – максималните грама храна, които Кольо може да изяде.

На вторият ред ще се съдържа числото **N** – броя храни, между които Кольо може да избира.

На следващите **N** реда ще се съдържат типовете храни – всяка една с име, тегло **W** и вкусотия **C**, разделени с интервал (“ “).

Входните данни ще бъдат винаги валидни и в описания формат. Не е нужно да бъдат проверявани изрично.

### Изходни данни

Изходните данни ще се отпечатват на конзолата.

На първия ред трябва да отпечатите **общата максимална вкусотия** на всичките **K** храни.

На следващите **K** реда трябва да отпечатите **имената на избраните** **K** продукти, в обратен ред на реда, в който сте ги получили от входните данни.

### Ограничения

* **N** ще бъде между **1** и **20**, включително.
* **M**, **W** и **C** ще бъдaт валидни 32-битови числа, по-големи от **0**.
* Имената на храните ще бъдат между **1** и **10** символа.
* Позволено време: **0.1 секунда**. Позволена памет: **32 MB**.

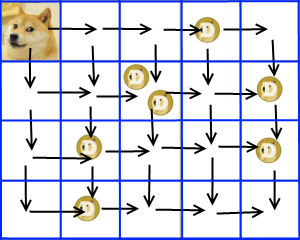
### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 10  6  pizza 3 2  burger 8 12  salad 4 5  candy 1 4  chicken 2 3  swine 8 13 | 17  swine  candy |

## Задача 2 – Doggie

Much coin, how money, such currency, so crypto. Wow.

Doge е много популярно куче. То е толкова популярно, че съществува виртуална валута на негово име. Виртуалната валута се казва DogeCoin. Doge обича своите DogeCoin-ове и иска да събере възможно най-много. Помогнете на милото клето животно!

Doge и DogeCoin-ове са поставени върху грид съдържащ **NxM клетки** (**N реда**, номерирани от 0 до N-1 и **M колони**, номерирани от 0 до M-1). Doge винаги е на позиция **[0; 0]**. Doge може да се дживи само в две посоки – **надясно и надолу**.

Върху грида има K на брой монети. Две или повече монети могат да се намират в една клетка. Също така може да има монети, където Doge стартира (0, 0) и той автоматично ги събира.

Намерете **най-голямата възможна бройка монети**, които Doge може да събере, движейки се само надолу и надясно.

### Входни данни

Входните данни ще се прочитат от конзолата.

На първия ред ще се съдържат числата **N** и **M**, разделени с интервал.

На втория ред ще бъде числото **K** – броя монети върху грида.

На следващите **K** реда ще се съдържат координатите **X** и **Y** на всяка една монета, разделени с интервал. **X** е номера на реда, а **Y** – номера на колоната, където монетата се намира.

Входните данни винаги ще бъдат валидни и в описания формат. Не е нужно да бъдат проверявани изрично.

### Изходни данни

Изходните данни ще се печатат на конзолата.

На единствения ред на изхода, отпечатайте най-голямата възможна бройка монети, които Doge може да събере, движейки се само надолу и надясно.

### Ограничения

* **N** и **M** ще бъдат между **1** и **2000**, включително.
* **K** ще бъде между **0** и **100000**, включително.
* Координатите на монетите ще бъдат винаги в рамките на грида.
* Позволено време: **0.2 секунди**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** |  | **Вход** | **Изход** |  | **Вход** | **Изход** |
| 4 5  7  1 4  0 3  1 2  2 1  3 1  1 2  2 4 | 4  // See the  // picture  // above | 10 10  11  0 0  1 1  2 2  3 3  4 4  5 5  6 6  7 7  8 8  8 9  9 9 | 11 | 4 4  11  1 1  2 1  1 2  2 1  3 3  0 3  3 0  3 1  3 3  1 1  1 0 | 8 |

* Позволена памет: **64 MB**.

## Задача – Време за битове

В една паралелна Вселена компютрите работят малко по-странно от нашите тук. Отново имаме нули и единици навсякъде, но процесорното време е различно при всяка операция с битове:

* За **промяна** от 0 на 1 или от 1 на 0 – **1** милисекунда
* За **добавяне на 1** – **1.2** милисекунди
* За **добавяне на 0** – **1.1** милисекунди
* За **изтриване на 1** – **0.8** милисекунди
* За **изтриване на 0** – **0.9** милисекунди

Вашата задача е да намерите минималното нужно време на такъв тип процесор да преобразуване една комбинация от нули и единици в друга.

### Входни данни

Входните данни ще се четат от конзолата.

На първия ред от входа ще прочетете комбинацията от нули и единици, която трябва да преобразувате.

На втория ред от входа ще прочетете комбинацията от нули и единици, до които процесора трябва да достигне като краен резултат от преобразуването.

### Изходни данни

Изходните данни ще се отпечатват на конзолата.

На единствения ред трябва да отпечатите **минималното нужно време** на процесора да преобразува нулите и единиците.

### Ограничения

* Комбинациите от нули и единици ще съдържат между **1** и **200** символа, включително.
* Комбинациите ще съдържат само нули и единици.
* Позволено време: **0.1 секунда**. Позволена памет: **16 MB**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 0  1 | 1 |
| 01  1 | 0.9 |
| 0  10 | 1.2 |
| 0110  1101001 | 4.4 |