2931. Maximum Spending After Buying Items

Description

You are given a **0-indexed** `m * n` integer matrix `values`, representing the values of `m * n` different items in `m` different shops. Each shop has `n` items where the j^{th} item in the i^{th} shop has a value of `values[i][j]`. Additionally, the items in the i^{th} shop are sorted in non-increasing order of value. That is, `values[i][j] >= values[i][j + 1]` for all `0 <= j < n - 1`.

On each day, you would like to buy a single item from one of the shops. Specifically, On the d^{th} day you can:

- Pick any shop `i`.
- Buy the rightmost available item `j` for the price of `values[i][j] * d`. That is, find the greatest index `j` such that item `j` was never bought before, and buy it for the price of `values[i][j] * d`.

Note that all items are pairwise different. For example, if you have bought item `0` from shop `1`, you can still buy item `0` from any other shop.

Return *the **maximum amount of money that can be spent** on buying all* `m * n` *products*.

Example 1:

```
Input: values = [[8,5,2],[6,4,1],[9,7,3]]
Output: 285
Explanation: On the first day, we buy product 2 from shop 1 for a price of values[1][2] * 1 = 1.
On the second day, we buy product 2 from shop 0 for a price of values[0][2] * 2 = 4.
On the third day, we buy product 2 from shop 2 for a price of values[2][2] * 3 = 9.
On the fourth day, we buy product 1 from shop 1 for a price of values[1][1] * 4 = 16.
On the fifth day, we buy product 1 from shop 0 for a price of values[0][1] * 5 = 25.
On the sixth day, we buy product 0 from shop 1 for a price of values[1][0] * 6 = 36.
On the seventh day, we buy product 1 from shop 2 for a price of values[2][1] * 7 = 49.
On the eighth day, we buy product 0 from shop 0 for a price of values[0][0] * 8 = 64.
On the ninth day, we buy product 0 from shop 2 for a price of values[2][0] * 9 = 81.
Hence, our total spending is equal to 285.
It can be shown that 285 is the maximum amount of money that can be spent buying all m * n products.
```

Example 2:

```
Input: values = [[10,8,6,4,2],[9,7,5,3,2]]
Output: 386
Explanation: On the first day, we buy product 4 from shop 0 for a price of values[0][4] * 1 = 2.
On the second day, we buy product 4 from shop 1 for a price of values[1][4] * 2 = 4.
On the third day, we buy product 3 from shop 1 for a price of values[1][3] * 3 = 9.
On the fourth day, we buy product 3 from shop 0 for a price of values[0][3] * 4 = 16.
On the fifth day, we buy product 2 from shop 1 for a price of values[1][2] * 5 = 25.
On the sixth day, we buy product 2 from shop 0 for a price of values[0][2] * 6 = 36.
On the seventh day, we buy product 1 from shop 1 for a price of values[1][1] * 7 = 49.
On the eighth day, we buy product 1 from shop 0 for a price of values[0][1] * 8 = 64
On the ninth day, we buy product 0 from shop 1 for a price of values[1][0] * 9 = 81.
On the tenth day, we buy product 0 from shop 0 for a price of values[0][0] * 10 = 100.
Hence, our total spending is equal to 386.
It can be shown that 386 is the maximum amount of money that can be spent buying all m * n products.
```

Constraints:

- `1 <= m == values.length <= 10`
- `1 <= n == values[i].length <=` 10^4
- `1 <= values[i][j] <=` 10^6
- `values[i]` are sorted in non-increasing order.