Slick

A maritime accident has caused oil to spill onto the seas of Felipistonia, which is a major natural disaster. The Felipistonia's government wants to clean up this mess before more damage occurs. To do this, they first have to know how serious was the accident and the amount of oil that has been spilled into the sea. The only instrument the Felipistonia's government has to get information of the magnitude of this disaster, is the use of satellite images. With these images they can estimate how much money they have to spend to clean this mess. For this, the number of slicks in the seas and the size of each slick must be know. A slick is a patch of oil floating on water. Unfortunately, the Felipistonia's people are not very bright, so they have hired you to help them process the image.

An example of an image obtained by the satellites is shown in Figure 1(a). This image can be transformed to 0's and 1's as seen in Figure 1(b). Given this binary matrix, your job is to count the number of slicks in the ocean and their corresponding size. Two adjacent pixels in the image are considered to be in the same slick if they are in the same row or the same column.

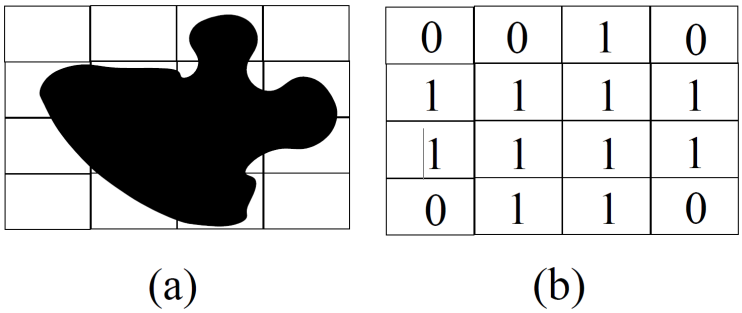

Figure 1: (a) A satellite image of the spilled oil. (b) The representation of the image in a binary matrix

Input

The input contains several test cases, each one corresponding to a different satellite image. The first line of each case contains two integers that indicate the number of rows (N) and columns (M) in the image (1 <= N, M <= 250). Then N lines follows with M integers each, containing the information of the image.

The end of input is indicated by a test case with N = M = 0. This case should not be processed.

Output

For each image, output the number of slicks in the sea. Additionally, output the size of each slick in ascending order and the number of slicks of that size.

Example

Input:
10 10
1 1 1 1 1 1 1 1 1 1
1 1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 1 1 1

1 1 0 0 1 0 0 1 1 1
1 0 1 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0

Output:
7
1 2
2 1
6 1
10 2
20 1