Detecting the largest black area in a binary image

# Problem definition

Given a binary image which contains only one and zero. Find the coordinates of the largest black area in the image.

# Solution

The problem needs to be divided into two sub-problems:

## Finding the largest area completely fit under a histogram

Given a histogram. Finding the largest rectangle completely fit under this histogram.

For instance, consider a histogram as follows

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 |  | 1 |
| 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 |

The largest area is 8 (contains cells with red text).

The problem can be solved using dynamic programming.

### Converting a binary image to histogram

The next step is to convert a binary image to a histogram so we can use above solution. The idea is to start from the bottom row of the image, and for each “one” cell, count how many “one” cells above the current cell in a same column.

Consider the binary image

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 1 | 1 |  |  |
| 1 | 1 | 1 | 1 |
|  | 1 |  | 1 |

The histogram will be

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 0 | 0 |
| 3 | 3 | 1 | 1 |
| 0 | 4 | 0 | 2 |

Then we can apply the above function in each round.

### Complexity

The complexity of the algorithm is O(m\*n), with m and n are width and height of the input image.

# Deliveries

I provied a Python (version 2.7) function called “problem1.py”

To run the program:

python problem1.py

To unit-test (requires pytest to be installed):

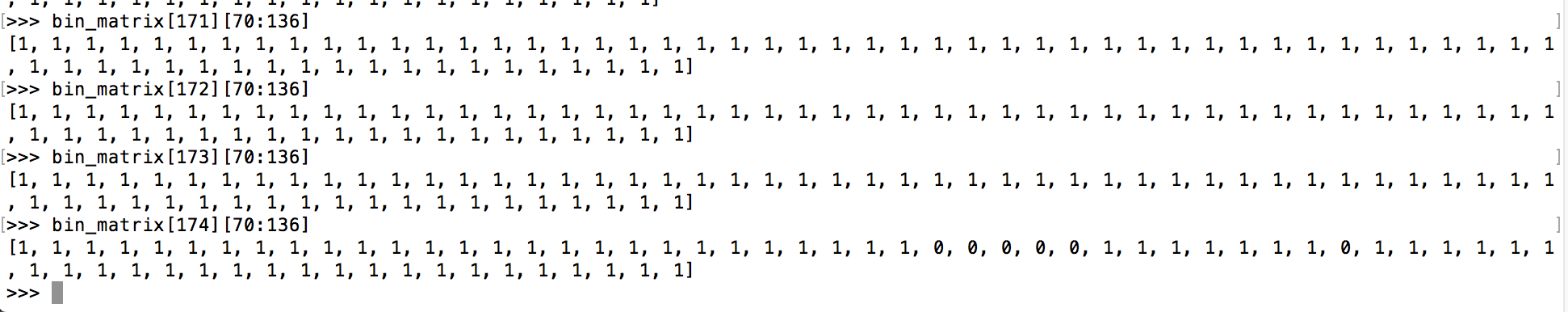
py.test problem1.py

## Notes

The original description stated that, for the file “binary\_image\_1”, the output should be [71,120,135,170].

However, I believe that the correct answer should be [71,120,135,173].

We could see that, the line from 171-173, colums 71-120 all contains “one” as the following picture.



Therefore these three lines should be included in the largest area.