Black and White

[Medium]
January 2021

Summary:

A computer artist works with black-and-white images of 32 x 32 units, for a total of 1024 pixels per image. He wishes to combine two images together to form a new image. Upon adding two images together, how many pixels will be black in the resulting image?

The two images are provided in their string representations as "quadtrees". A quadtree is a representation format used to encode images. An image can be split into four quadrants. For each quadrant: if it is not purely black or purely white, it may again be split into four sub-quadrants (and so on) until the sub-quadrants are either purely black or purely white.

Constraints:

- Test cases $T \leq 100$
- Time limit: 5s

Problem Details:

Each quadrant of an image is ordered as follows:

2	1
3	4

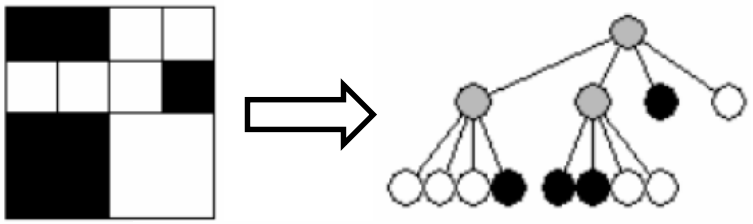
Each image is represented by a pre-order string indicating the state of the quadrant.

'p' indicates a parent node

'f' indicates a black quadrant

'e' indicates a white (empty) quadrant

As an example, see image on the right:

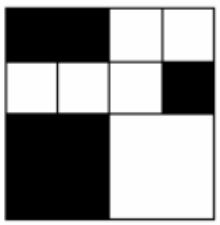


ppeeefpffeefe

p	p	e	e	e	f	p	f	f	e	e	f	e
Top parent	Q1 parent	Children of Q1 parent				Q2 parent	Children of Q2 parent				Q3 black quadrant	Q4 (empty) white quadrant

An image has a total of 1024 pixels.

Number of black pixels per given image must also be computed:



Q1 has **64** black pixels $(1024 \times \frac{1}{16})$

Q2 has **128** black pixels $(1024 \times \frac{2}{16})$

Q3 has **256** black pixels $(1024 \times \frac{4}{16})$

Q4 has **0** black pixels $(1024 \times \frac{0}{16})$

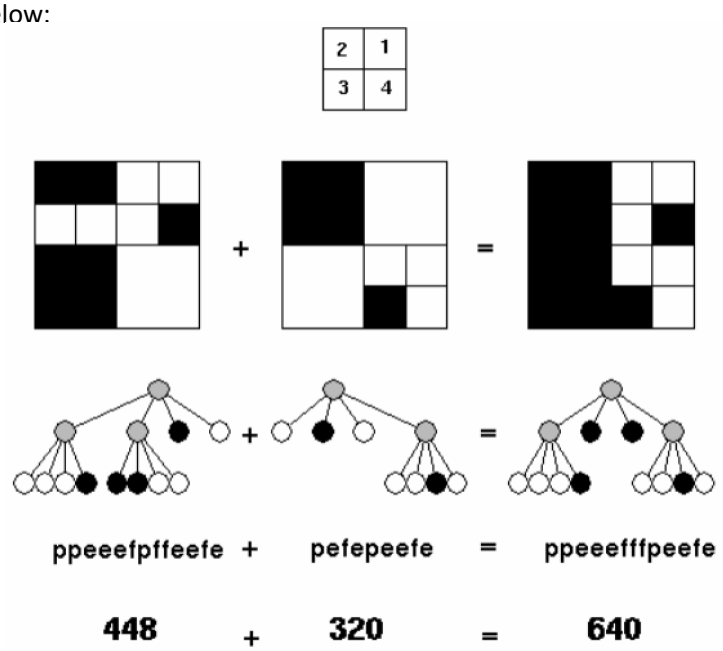
The image on the left has a total of **448** black pixels

Black and White

[Medium]
January 2021

For each test case, two image strings will be given. These two image strings must be combined: if an area in either image 1 or image 2 is black, then it must also be black in the combined image. In technical terms, we are taking the "union" of all black pixels in image 1 and image 2 to get the combined image.

After combining the two images, the program must output the number of black pixels from the resulting image as shown in the example below:



Input Specifications:

The first line contains the number of test cases T.

The input for each test case is two strings, each string on its own line.

It is guaranteed that each string represents a valid quadtree, while the depth of the tree is not more than 5 (because each pixel has only one color. t.

Output Specifications:

For each test case, output the case number with the format "Case #x: There are y black pixel(s).\n", where x is the test case number, and y is the resulting number of black pixels in the combined image.

Sample Input

```
3
ppeeefpffeefe
pefepeefe
peeef
peefe
peeef
peepefefe
```

Sample Output

```
Case #1: There are 640 black pixel(s).
Case #2: There are 512 black pixel(s).
Case #3: There are 384 black pixel(s).
```

Black and White

[Medium]
January 2021

Proposed Solution:

We can solve the problem in two ways: (1) we can simultaneously check each quadrant of both image 1 and image 2 recursively, and add to our count of black pixels whenever either the sub-quadrant from image 1 or from image 2 is black; or (2) we can directly convert each image string into its equivalent 32x32 image (with 1024 pixels), then perform pixel-wise OR operation (where black = 1 and white = 0), then finally count the number of black pixels in the combined 32x32 image.

Below is the pseudocode for the 1st option:

Let **quadrant_value** = 1024 **image1_idx** = 0 **image1_string** = *image 1 string input*
total_black_pixels = 0 **image2_idx** = 0 **image2_string** = *image 2 string input*

main_function:
Loop: 1 to 4:
1. Check base case, if max length of either image strings have been reached, if yes, output **total_black_pixels**.
2. Get character from current index from both strings
   c1 = image1_string[**image1_idx**]
   c2 = image2_string[**image2_idx**]
3. Check if either **c1** or **c2** is an F, if yes, add **quadrant_value** to **total_black_pixels**
4. Check next action:
   a. If **c1** is P and **c2** is F, all child quadrants of image 1 can be skipped
      Move **image1_idx** to next sibling quadrant index
      Move **image2_idx** to next index
      Continue loop
   b. If **c2** quadrant is P and **c1** quadrant is F, all child quadrants of image 2 can be skipped
      Move **image2_idx** to next sibling quadrant index
      Move **image1_idx** to next index
      Continue loop
   c. If **c1** is P and **c2** quadrant is P
      Move **image1_idx** to next index
      Move **image2_idx** to next index
      Recursive call of **main_function** passing **quadrant_value** ***of quadrant_value / 4***
   d. If (**c1** is E and **c2** is P) OR (**c1** is P and **c2** is E)
      i. Calculate the number pixels for the parent quadrant and add to **total_black_pixels**
      ii. Continue loop

For the second option, we recursively parse the image string and generate the corresponding 32x32 image for each image string. Every time a quadrant is subdivided into its four corresponding sub-quadrants (i.e. when the next character in the image string is a 'p'), we take compute the grid coordinates of the sub-quadrants. If the sub-quadrants are either 'f' or 'e', we color the corresponding coordinates of the sub-quadrants as black or white respectively. If it is 'p', we subdivide the sub-quadrant again.

Once we have the two 32x32 image grids, we just get the OR of the two image grids (e.g. if cell (1,1) is black for image 1 or image 2, it is set as black for the combined image). Then we just count the number of black cells in the combined image.