Data Structures

Assignment 2: (Due on Oct 15 at 6 pm)

Twinkle twinkle little star ...NOT!

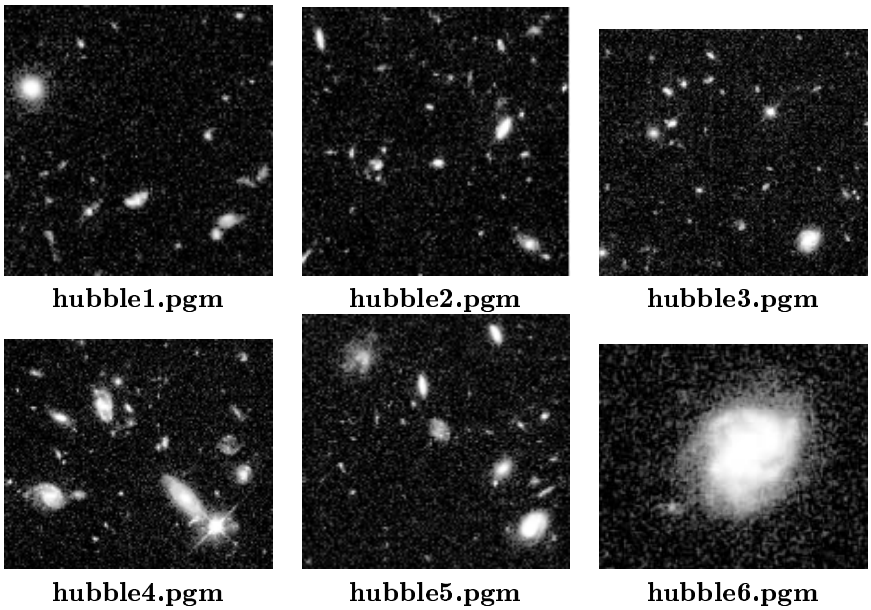

Figure 1: Images of galaxies taken by the Hubble space telescope.

Goal: The goal of this assignment is to use stack and queues to automatically count the number of galaxies in the thresholded Hubble images. The output images from your first assignment will be the input images for this assignment. You have to basically count the number of connected white regions.

For example, consider the 16 by 16 image shown in Fig 2. There are two connected black components. You will mark these pixels in a new blank image with the component number. Thus for the image Fig 2, your output will be another image with the pixels in the top-left connected component marked with a value of 1 and the pixels in the bottom-right marked with a value of 2.

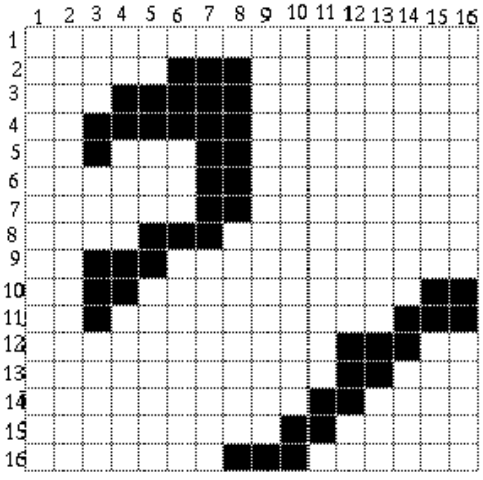

Figure 2: An example image with two connected components. The image size is 16 by 16.

Algorithm: You will build the galaxy labeling algorithm around a seed region growing algorithm. Given a starting point (the seed) the region growing algorithm finds all pixels that have the same gray level value as the seed pixel and connected to it. The idea is to iteratively invoke this seed region growing algorithm for all the galaxies. The region growing algorithm can be described as follows.

Each pixel in an image has 8 neighbors. For example, the pixel (4, 6) has the pixels at locations {(4, 5), (4, 7), (5, 5), (5, 6), (5, 7), (3, 5), (3, 6), (3, 7) } as neighbors. Not all the neighbors have the same pixel value. We start by considering the neighbors that have the same value as the starting pixels. There are two ways to proceed: depth first or breadth first. In depth first search, you recursively follow one neighbor of a pixel. In breadth first search you process all the neighbors of one pixel before proceeding to the next one. The pseudo code of the algorithms are as follows:

```
Breadth_First_Search (Input_Image, Output_Image, Start_x, Start_y, Value)
  Q: Queue of (x, y) coordinates of a pixel
  px, py: integers;
  Initialize Q to null.
  Set all pixels of the Output_Image to 255 (white).
  Enqueue(Q, (Start_x, Start_y));
  while (Q is not null) {
    (px, py) = Dequeue(Q);
    Output_Image(px, py) = Value;
    for each neighbor, (nx, ny), of (px, py) {
       if (Input_Image(nx, ny) == Input_Image(px,  py)) and
          Output_Image(nx, ny) == 255 {
         Output_Image(nx, ny) == -120;
         Enqueue(Q, nx, ny);
        }}}
end Breadth_First_Search;

Depth_First_Search (Input_Image, Output_Image, Start_x, Start_y, Value)
  S: Stack of (x, y) coordinates of a pixel
  px, py: integers;
  Initialize S to null.
  Set all pixels of the Output_Image to 255 (white).
  Push(S, (Start_x, Start_y));
  while (S is not null) {
    (px, py) = Pop(S);
    Output_Image(px, py) = Value;
    for each neighbor, (nx, ny), of (px, py) {
       if (Input_Image(nx, ny) ==  Input_Image(px,  py)) and
          Output_Image(nx, ny) == 255 {
         Output_Image(nx, ny) == -120;
         Push(S, nx, ny);
       }}}
end Depth_First_Search;
```

The above algorithms will mark with "`Value`" all pixels with the same gray level and connected to the pixel at (`Start_x`, `Start_y`). You have to iteratively invoke the above subroutines to process all the galaxies in the image.

What to code? You have to implement the above two search algorithms as a part of the image ADT package. The application should use the generic queue and stack packages from the text book and the image package you developed in the first assignment. The application should have an menu based interface that allows the user to select the search algorithm, to specify the image, to save the marked-up image, and to output the number of galaxies, with their sizes.

What to submit? For details about what and how to submit, see first assignment. Make sure you include outputs that show the states of the Output_Image after 5, 20, 50, 100, and 300 iterations of the depth first search and the breadth first search for big galaxy in (`hubble6.pgm`).