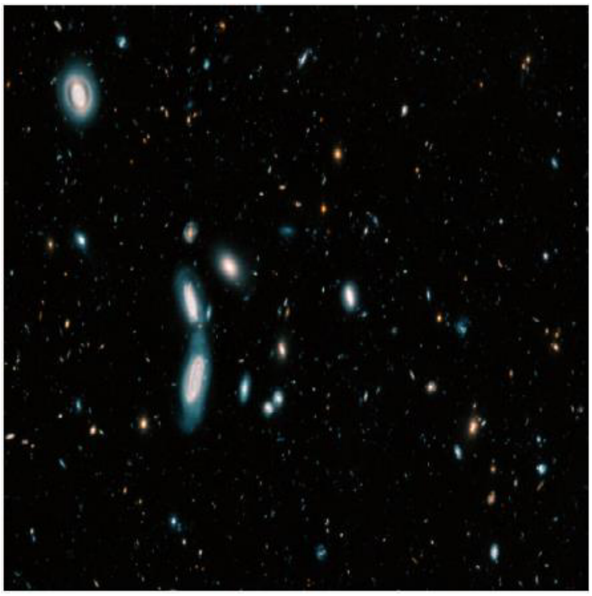
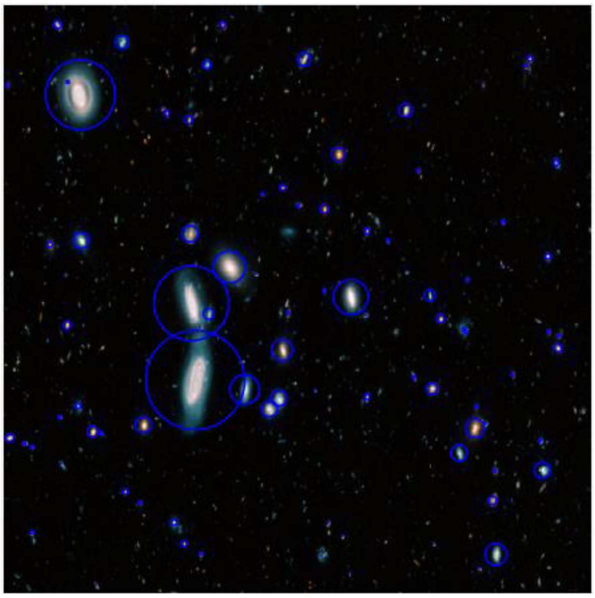
Data Structures and Algorithms 2

CA Exercise 1 – Deep Space Imagery Analyser

"Create a deep space imagery analyser system in JavaFX."

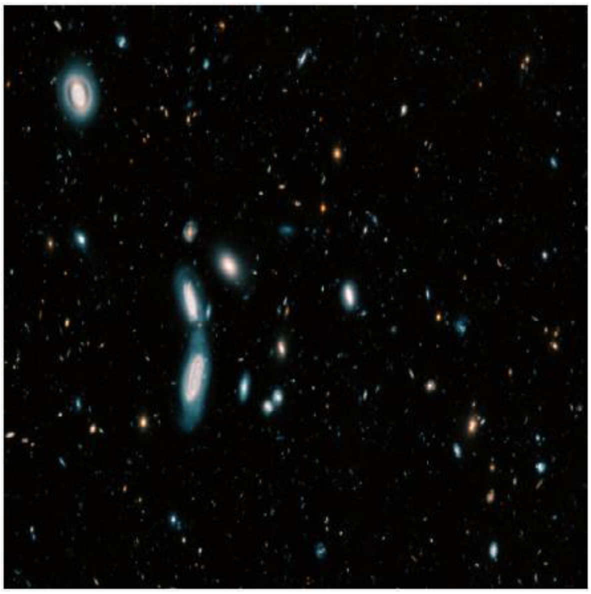
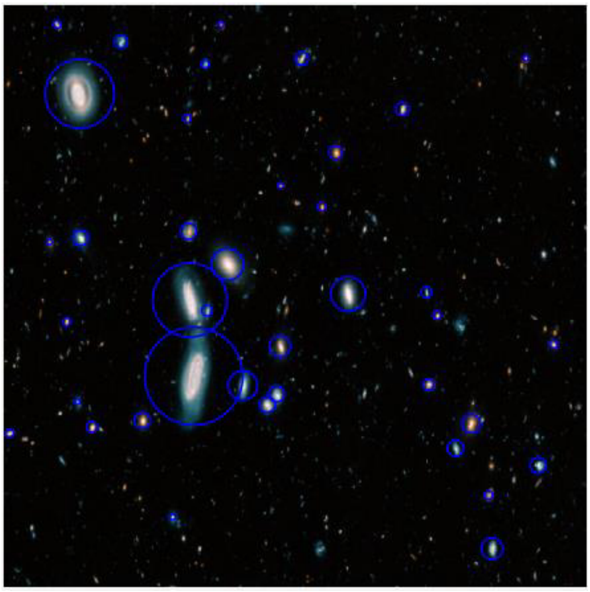
The objective of this CA exercise is to create a JavaFX application that can analyse a given image of deep space containing various celestial objects (stars, galaxies, nebulae, etc.). When given a deep space image (and some user settings), the application will automatically locate the various celestial objects on the deep space image and estimate how many of them are in the image overall. The application should also mark the location of celestial objects on the deep space image with blue circles, optionally number them sequentially from largest to smallest, and allow for gaseous analysis of individual celestial bodies.

Example 1: Given the following Hubble deep space image (left below) as input, the output might be as follows (right below; 95 celestial objects estimated/located in total).

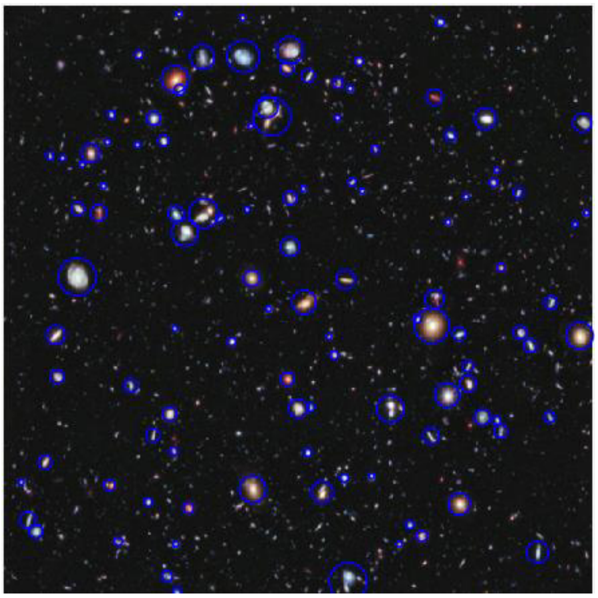
You can see that the main bright celestial objects have been located and blue circles superimposed on them. Objects deemed too dim were not located, but this is simply a user setting (luminance threshold or similar) and could be adjusted to locate more (or fewer) objects at the user's discretion. Celestial object size was ignored in this example, but it should also be possible to have the user specify min/max size settings to help better identify and locate the celestial objects being sought.

Example 2: Same deep space image (and luminance threshold etc.) as Example 1 but with a minimum celestial object size of 5 pixels (39 celestial objects estimated/located in total):

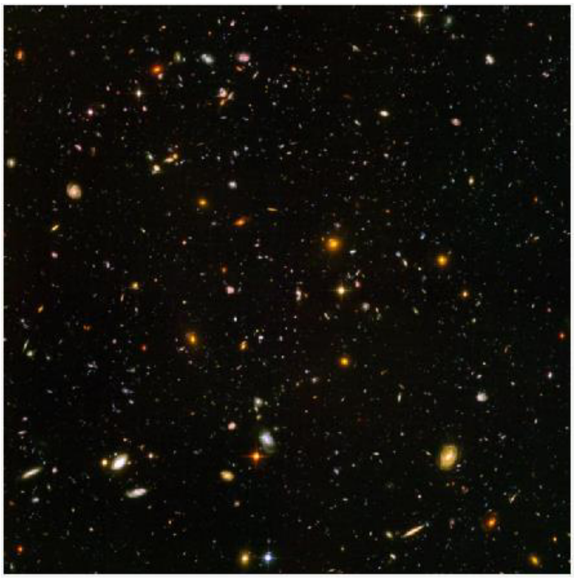



You can see that a number of smaller celestial bodies (56 in this example) are no longer circled blue.

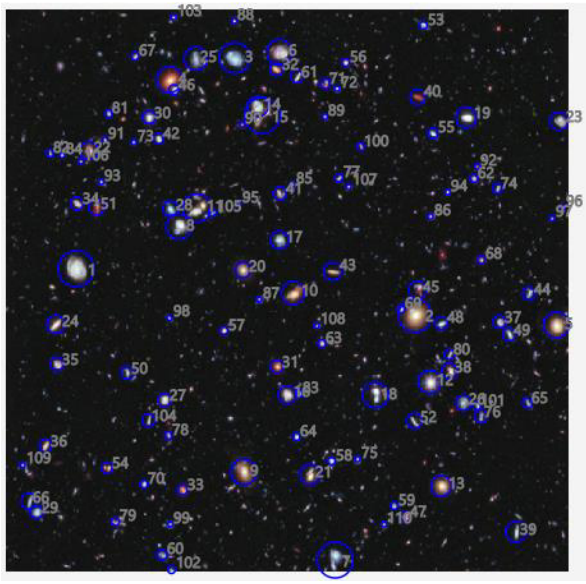
<u>Example 3</u>: Another deep space image with same settings (luminance threshold, minimum size, etc.) as Example 2 (110 celestial objects estimated/located in total):

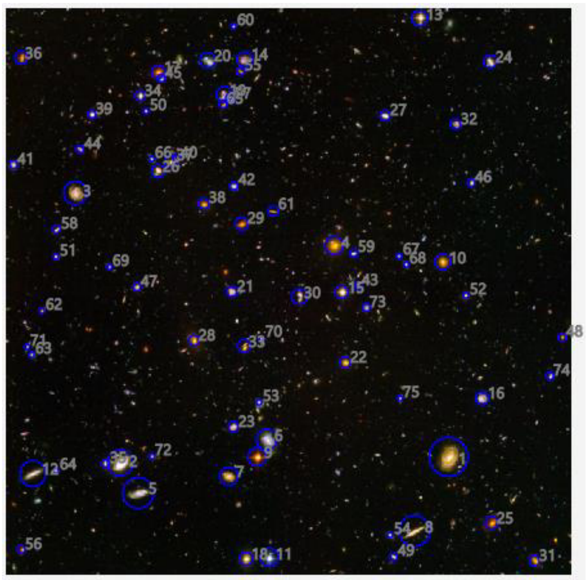



<u>Example 4</u>: Another deep space image with same settings (luminance threshold, minimum size, etc.) as Example 2 (75 celestial objects estimated/located in total):



<u>Example 5</u>: Same input images and settings as Examples 3 and 4 but with sequential numbering of celestial bodies (from largest to smallest) enabled (output images only shown):

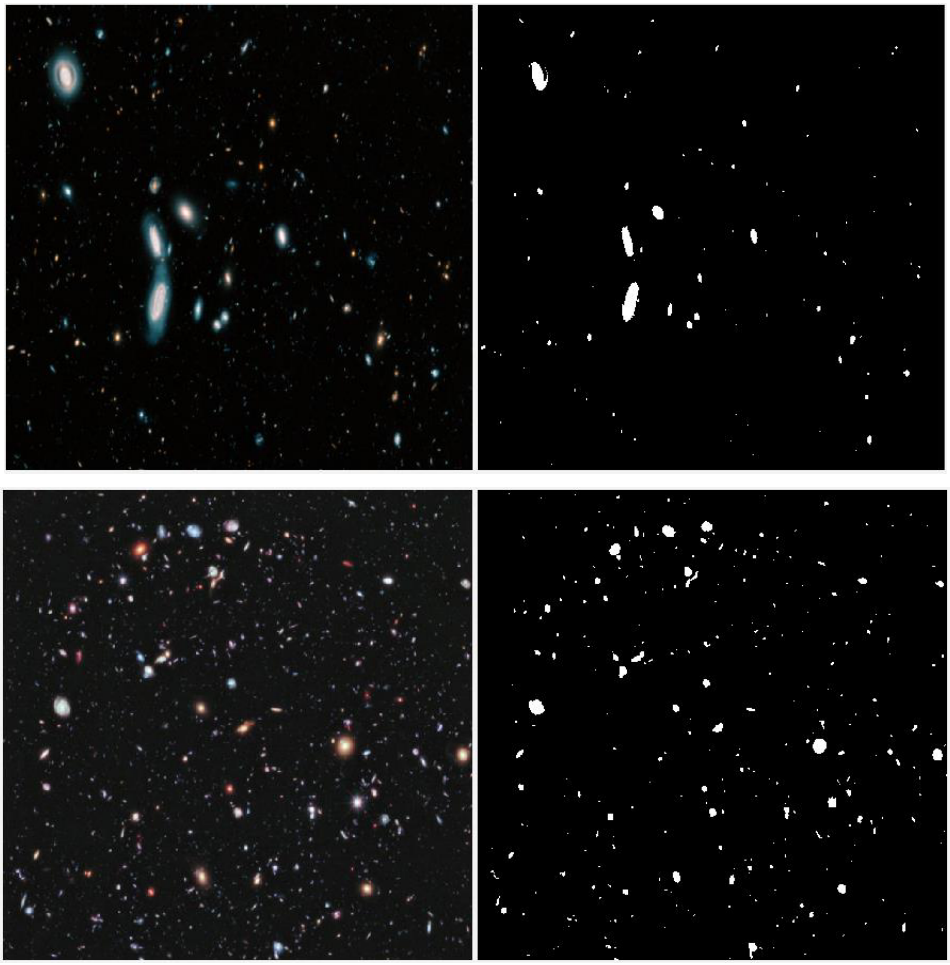




The deep space imagery analyser should be able to work <u>reasonably well</u> for these and similar scenarios too. It is not expected to work perfectly. For instance, a single celestial object may (because of colour variations, etc.) be detected as multiple celestial objects (thus making the total estimate too high). The colourisation, brightness, contrast, graininess, etc. of the deep space image will also likely impact how well the system works for it. Some form of image noise reduction/management could be used to improve this by stripping out outliers/things that are unlikely to be celestial objects (as they are too small or too big, say) and by allowing the user to set thresholds (for luminance etc.) for celestial object detection. Using heuristics and strategies such as these can make the analyser work better than in the simple examples shown here.

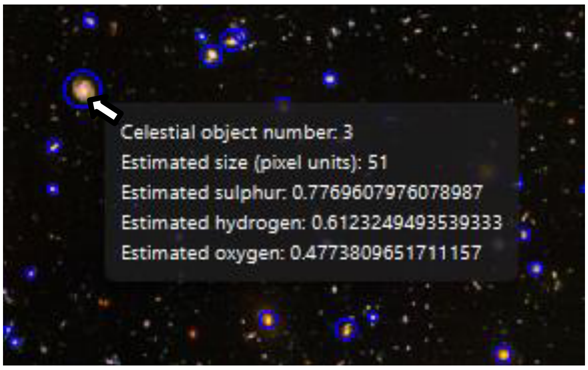
Implementation Notes

- The key aspect of this CA exercise is to use a union-find algorithm to locate the celestial objects.
  - Note: you will be dealing with a lot of data (disjoint sets/elements) so remember to use appropriate techniques from last semester to efficiently process large data sets.
- The input deep space image should be converted to black-and-white in the first instance pixel-by-pixel using suitable luminance/hue/saturation/brightness/RGB calculations.
  - Ideally, pixels belonging to any celestial object should be white, and all other pixels should be black.
  - Allow the user to specify parameters/options to help the user achieve a good black-and-white conversion (e.g. to set luminance thresholds, etc.).
  - Note that it is crucial to achieve a good/clean black-and-white conversion for the union-find to work well for a given deep space image.
  - Users should be able to view the black-and-white version of the deep space image (e.g. in a separate window, tab, or pane).
  - Black-and-white conversion examples:



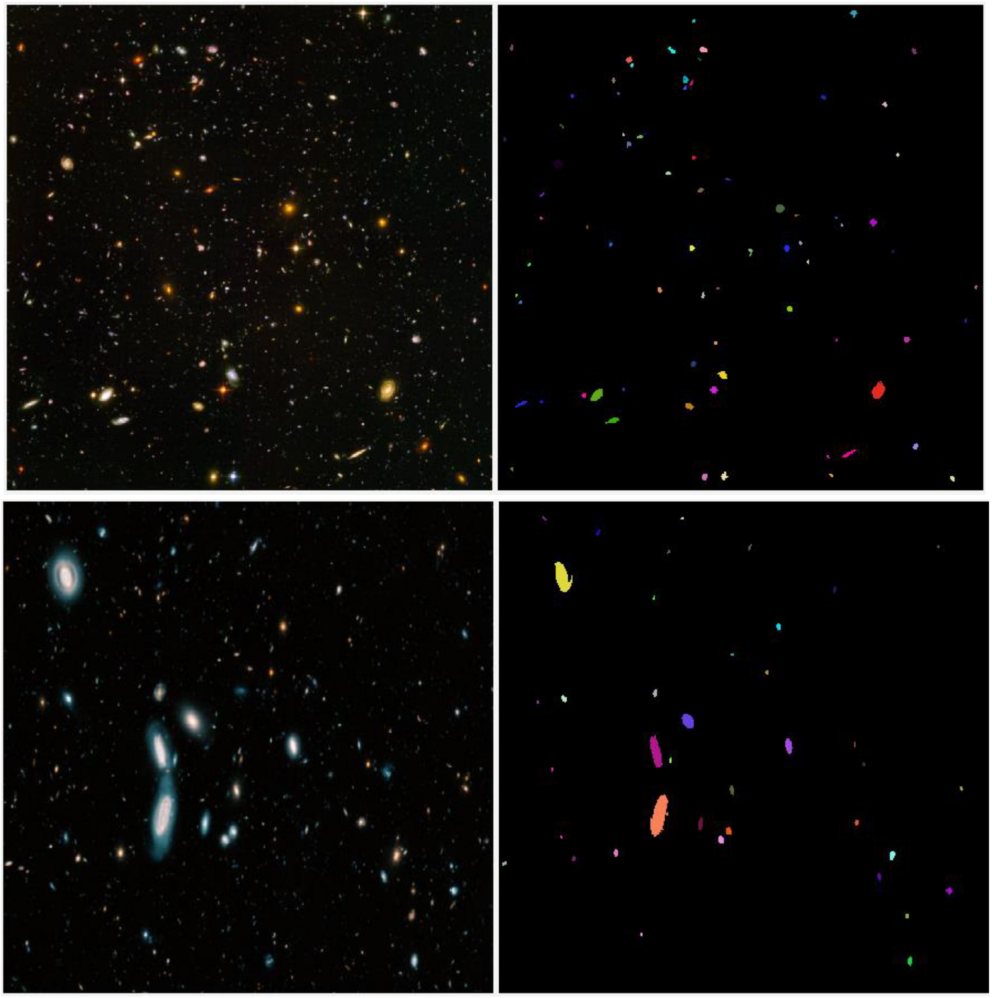
- Each pixel in the black-and-white image can initially be considered a disjoint set, with this information potentially represented in an array. Union-find can then be applied to union adjacent white pixels (up, down, left, and right) to identify celestial objects. Black pixels can be disregarded.

- You may decide to rescale the image from its original size to something smaller for the recognition process (e.g. 512x512, or similar). A smaller image will require less pixel information to be stored, will be quicker to process, and small "noise" may be (helpfully) lost during the scaling down too.
    - Tip: work with a small test (or fake) image first for development/debugging purposes before using a larger real image.
- Once the union-find is complete, the individual disjoint sets can be processed to <u>identify the set pixel boundaries</u>. The size of the boundary can potentially suggest whether a disjoint set is likely to be a valid celestial object or not (this can help with noise management). Blue coloured circles based on the boundaries can subsequently be superimposed on the original deep space image to identify celestial objects.
- You should be able to <u>analyse/examine/query individual celestial objects</u> in some appropriate fashion (e.g. hovering the mouse over them, or providing information on all celestial objects in a list, table, tree view or similar). This analysis will provide:
    - The number of pixel units within a given celestial object (i.e. the exact <u>size of the disjoint set</u> representing the celestial object).
    - A <u>gaseous analysis</u> of the celestial object by reporting the sulphur, hydrogen, and oxygen concentrations on a 0-to-1.0 scale.
        - Note that deep space images are typically artificially coloured wherein sulphur is represented by red, hydrogen by green, and oxygen by blue. So report the average pixel RGB values in a celestial object to represent sulphur, hydrogen, and oxygen respectively.
    - For instance, touching celestial objects with the mouse in the example below:



- Every celestial object (of all types) should be <u>sequentially numbered in sorted order</u> (starting from 1) whereby the largest celestial object (of any type; based on pixel unit/disjoint set size) is 1, the next largest is 2, and so on. These sequential numbers should be visible/viewable (in some appropriate way) alongside the celestial object sizes and gaseous analysis. See the above image for example (Celestial object number 3 => 3rd largest celestial object in the image based on its number of pixels).
- A menu option should also be provided to allow for the <u>size-ordered sequential numbers to be shown onscreen</u> in or alongside the blue circles/boundaries (see Example 5).
- It should also be possible to <u>see the pixels of individual disjoint sets in the black-and-white image</u>. This should be possible by both:
    - Colouring all white pixels of a single chosen celestial object in the black-and-white image to an appropriate colour (e.g. the average hue/colour of all object pixels).

- Randomly colouring all of the various disjoint sets in the black-and-white image.



- Some level of image noise reduction and outlier management should be incorporated into the analyser to allow users to strip out disjoint sets that are heuristically unlikely to be celestial objects.
    - The application could allow for minimum and maximum celestial object sizes to be user specified (e.g. in a settings dialog or similar).

    - Consider calculations based on the interquartile range (IQR), for instance, to identify outliers.
    - As already noted, some user-adjustable settings (e.g. luminance threshold levels) could also be provided to achieve a good and clean black-and-white image conversion before the union-find operation.
- Overall, an <u>appropriate interactive JavaFX graphical user interface</u> should be provided to allow users to:
    - Select a deep space image file to use.
    - View the original deep space image.
    - Perform and view the black-and-white image conversion (using the user specified settings for noise management, luminance, etc.).
    - Perform the celestial object recognition/identification using union-find, and mark celestial objects on the original deep space image (or a copy of it) with blue circles.
    - See estimates of the number of celestial objects in the image.
    - See the pixel unit/disjoint set sizes and perform simple gaseous analysis.
    - Ordered sequential numbering of celestial objects, including onscreen labelling.
    - Visualise/colour disjoint set pixels in the black-and-white image (individually as an average colour for a given set, or random colours for all sets; support both options).
    - Adjust settings to manage noise and outliers and to aid the black-and-white conversion.
    - Cleanly navigate and exit the application and have a good user experience overall.
- This is an <u>individual CA exercise</u> and is worth 35% of your overall module mark. You must submit it on Moodle and demonstrate it (in the lab or via Zoom) for it to be assessed and included in your overall marks. The demo/interview is mandatory and can only be done with your assigned lab lecturer (either Peter or Joe).

Indicative Marking Scheme

- Deep space image file selection and display = 5%
- Black-and-white image conversion and display = 10%
- Union-find implementation = 10%
- Onscreen identification of all selected celestial objects (using blue circles) = 10%
- Ordered sequential celestial objects numbering (onscreen labelling) = 10%
- Estimating/counting of celestial objects in overall image = 5%
- Reporting the size of individual disjoint sets (in pixel units) = 5%
- Gaseous analysis for sulphur, hydrogen and oxygen = 5%
- Colouring disjoint sets in black-and-white image (individual average / all random) = 10%
- Image noise reduction and outlier management = 10%
- JavaFX GUI = 5%
- JUnit Testing = 5%
- JMH Benchmarking of key methods = 5%
- General (overall completeness, structure, commenting, logic, etc.) = 5%

Note that it is not expected that all students will attempt all aspects of this assignment. Use the above marking scheme to guide your efforts as this is what you will be marked against.