“Number of Islands” is a programming problem often used in Software Engineering interviews to test the interviewees’ problem-solving skills and grasp of algorithms. To be solved efficiently, a graph algorithm must be written to traverse the graph and count the number of “islands” without revisiting the same node twice or storing too much information in memory. Graphs consist of a list of lists where the integer 0 represents water and 1 represents land. For example, a graph might look like

[[0, 1, 0, 1],

[0, 1, 0, 0],

[0, 0, 0, 0],

[1, 1, 1, 0]]

Using both Rook-based contiguity and Queen-based contiguity, this graph contains 3 distinct islands.

I intend to build a tool that produces the correct number of islands for any raster (in the correct format) of island chains. The tool’s input would consist of rasters represented as a list of 3 Numpy Arrays, where each array corresponds to a raster band (typically r, g, b). The tool would also accept a python lambda function that returns a boolean representing if a cell is ‘land’ or ‘water’, based on 3 values passed in as input (i.e., the r, g, b value of a given cell). This would allow users to specify their own definitions of what band values constitute ‘land’. Next, the tool would run an algorithm on the grid, and return the number of islands in the input raster, according to the lambda function’s decision process.

I intend to implement this tool with both Depth First and Breadth First search algorithms in order to compare their executions. The runtime of these algorithms will be compared across a variety of input sizes.

In addition to producing the correct output, I intend to visualize the processes of both of these search algorithms for educative purposes. This could be done by ‘highlighting’ the cell currently being visited by the algorithm and outputting a raster for (nearly) every single step of the execution, and then collating each raster into a video that tracks the search across the input raster over time. The worst-case algorithmic complexity for both Depth First and Breadth First searches is O(n \* m), which although incredibly efficient, would still produce an enormous amount of rasters. This visualization would therefore require that some steps are not visualized (i.e., skip every few output rasters when not on land) and that only smaller rasters are used for visualization.

This solution would allow for users to easily check and verify the number of islands in a given raster. Algorithms to solve this theoretical Computer Science problem can be found online but are never applied to real-world datasets. The script itself would produce a unique analysis tool, and the video of execution would be useful for learning about and understanding how different search algorithms work. This tool could be run routinely with satellite data for a given region and produce a notification if the output is different than that of before – allowing the effects of things like rising sea levels and volcanic activity on island chains to be monitored autonomously. It should also be noted that while the task is called “Number of Islands”, the tool need not be constrained to only finding islands. The same analysis could be performed to find distinct clusters of any type of landcover. For example, the tool could be used to find the number (and locations) of ‘islands’ of Amazon Rainforest among the ‘sea’ of newly slash-and-burned grazing land.

The deliverables for this project will include a python module, a number of rasters used for testing purposes, and a video (likely on YouTube) showing the tool in use. An explanation of the tool’s functionality will also be provided.