Technical Report

Name: Jiahe Xu

ID: U7279920

Question 1:

Using the functions defined in question 2, we obtained the minimum elevation, 464.28 meters, the maximum elevation, 788.23 meters and the average elevation, 594.69 meters for the `elevation_data_small` set. These values were validated by importing the CSV file to Microsoft Excel and using MAX(), MIN() and AVERAGE() functions to find the largest, smallest and the average values respectively. The slope of the steepest point in this data set is 2.62, while the maximum surface area is 3068350m^2.

Question 2:

Despite our familiarity with Python lists, our group chose the `numPy` array as the data structure for questions 1 through to 5 because of its superior speed in large scale data analysis. While the speed improvement offered by `numPy` is trivial when working with a small data set; when dealing with `elevation_data_small` and `elevation_data_large`, it's vital to consider how a small runtime difference in one task can accumulate and result in great difference in the total runtime. Since we're analyzing large data sets, it's worthwhile learning `numPy`.

Another reason why we chose `numPy` arrays is because the `numPy` library contains many functions useful for analysis and visualization of elevation data. If used appropriately, this could save us the time of otherwise writing unnecessary code. For instance, using `numPy` function `min()`, we defined `minimum_elevation` in one line.

Initially, we considered using Python lists; but it wasn't long before a quick Internet search showed that `NumPy` or `Panda` are far superior options for data analysis. We chose `Numpy` because both of the data sets, `data_elevation_small` and `data_elevation_large` have less than 500,000 rows. For data of this size or smaller, `NumPy` has better performance than `Panda`.

Question 3:

`Read_dataset` was tested by calling `shape` with `data_set_small` to reveal the size of `elevation_data_small`. This call evaluates to (883, 1189), meaning the array has 883 rows and 1189 columns. I then opened `data_set_small` in a CSV file viewer and confirmed that the file indeed has 883 rows and 1189 columns. Through this test, I know that `Read_dataset` correctly imports a CSV file to Python.

To test `minimum_elevation`, `maximum_elevation` and `average_elevation`. I called these functions with `data_set_small` as the argument. Next, I imported `elevation_data_small` into Excel and used Excel functions to find the maximum, minimum and average value of the cells. The numbers returned match with the results obtained using our Python program. Thus, it's reasonable to say that `minimum_elevation`, `maximum_elevation` and `average_elevation` all behave as expected.

To answer question 3, we dissected the problem into 4 parts and wrote a helper function to solve each of them. First, we identified the necessity to record the neighbouring cells of the given cell; and if the given cell lies on one or more edges of the data set, then we need a method to estimate the value for the edge cell. We eventually agreed that using the gradient leading up to an edge cell to estimate the neighbouring cells of an edge cell is the best method. With `cell_neighbours`, we could store a given cell's four neighbours in `cell_left`, `cell_right`, `cell_above` and `cell_below`. With these variables we were able to concisely define `x_gradient` and `y_gradient`. Using all the functions mentioned above, we were able to put together the `slope` function.

Using `surface_area` defined in question 4, I found that the surface area of Cotter Dam is 2714325 m^2 (2.71 km^2). Comparing this to the surface area shown on the Wikipedia page of Cotter Dam, 2.85km^2, we concluded that the numbers were close enough to indicate that the function is most likely correct. However, since the dam's water level at the time of measuring isn't provided, we couldn't make a justified conclusion about the accuracy of the program. Aside from using existing data to validate the results, we tried to calculate the surface area of Cotter Dam by approximately tracing the outline of Cotter Dam on Google Map and using its surface area option, which automatically calculates the area of an enclosed shape. This returns 2.28km^2, which is less reliable than the data found on the WikiPedia page. However, the fact that the surface area value returned by our program lies in between 2.85km^2 and 2.28km^2 shows that the function is mostly correct.

To test `expanded_surface_area`, we created a graph of Cotter Dam when it's at its maximum capacity. This graph displays a similar, but larger shape, with thicker lines branching off from the main water body and some parts of the nearby streams included, but overall preserving the original shape of the dam. This function, although not perfect, is adequate at producing an estimate of the dam's surface area if the water level were to increase.

We tested `full_catchment_area` by generating plots of catchment areas of Cotter Bendora and Corin Dams. We expect the catchment to be a single, continuous area of land. However this was not the case for the catchment of Bendora and Corin Dams. As shown in the plots below, the catchment includes a random scatter of small areas of land that are isolated from each other. This result indicates that `full_catchment_area` doesn't accurately return the catchment area of a dam.

Question 4:

When dealing with edge cells in question 3, we had initially decided to use the value of the given cell as the value of the neighbouring cell/cells of an edge cell. Later we found that using the gradient leading up to the cell to get a rough estimate of the values of the neighbouring cell is a better practice because it doesn't assume that the elevation would suddenly plateau for all the edge cells. Instead, we should make the assumption that they keep rising or falling at the current rate.

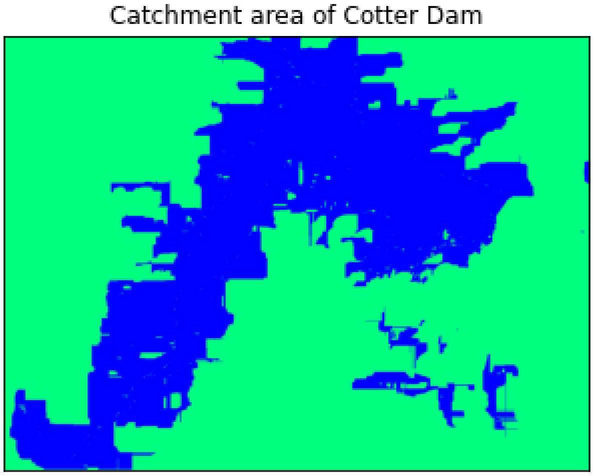
For question 4, we determined whether a given point is part of the dam based on two criterias. First, we get the elevation at the given point, from which we add and subtract 0.5 to define `elevation_upper_bound` and `elevation_lower_bound` respectively. This decision is based on the assumption that the elevation data was recorded when the dam's surface is relatively level, as should be the case most of the time - unless wind, earthquake or other causes, manmade or natural, lead to disturbance in the water. On top of this, we set `slope_tolerance` to 0.065. If the gradient of any cell exceeds this limit, it will not be counted as part of the dam. We did this because the gradient of a level water surface should be very close to 0. Initially, we had set `slope_tolerance` to 0.06, but through comparing a plot of our dam to a satellite image of Cotter Dam, we discovered that setting `slope_tolerance` to 0.065 produced a plot that has closer resemblance to the satellite image of Cotter Dam.

Using `numPy` function `logical_and`, we were able to construct a Boolean array where cells containing `True` represent points that lie on the dam, while the cells whose coordinates don't lie on the dam take Boolean value `False`. To calculate the surface area of the dam, we simply counted the occurrence of `True` - the total number of points that lie on the dam, and multiply the result by 25 - the area occupied by each individual point. This multiplication by 25 is based on the assumption that each point is 5 meters apart from the next one.

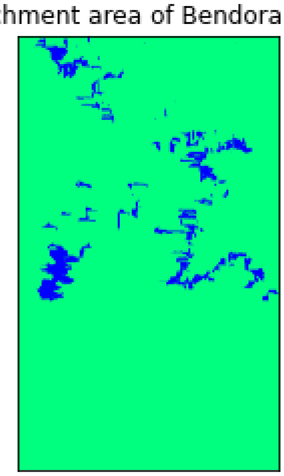
For question 5, we made the assumption that the user will always input coordinates of a point that lies on the dam, rather than any point from the data set. Using the given inputs, we constructed a Boolean array of the dam at its new water level and then called `dam_mask_to_surface_area` to return the surface area of the elevated dam. If the user inputs a point from non-dam areas such as the surrounding land, creek, or concrete structures, the program would interpret flat areas surrounding that point as the dam. Our program doesn't detect this situation and would proceed to execute, thus returning misleading results.

For question 6, we had initially decided to search and replace all -3.403e^38 because the assignment instructions suggests that this is the only outlier. Although this approach works for this assignment, we weren't satisfied with having to make the assumption that -3.403e^38 is the only outlier. Instead, we wanted a function that automatically replaces the outliers in any given array. We found we can do this using z-score, which removes all the -3.403e^38 in the array without removing any valid values.

Catchment area of Cotter Dam



Catchment area of Bendora Dam



Catchment area of Corin Dam