Name: Yixi Rao
Report authors UID: u6826541
Co-authors code: u6826541, u6833622, u6832190

Report

Question 1

1. I added an 'if statement' under the second 'for statement', and the 'if statement' will distinguish whether the cell is higher than the water level or not, if so, the number of cells will add one. Besides, in the second function `area_above_water` I just changed the unit from square metres to square kilometres
2. Rspe whole above water: 21.4861 hectares
   Anu whole above water: 300.5341 hectares
   Rspe height above water: 21.4843 hectares
   Rspe buildings above water: 0 hectares
   Anu height above water: 300.5252 hectares
   Anu buildings above water: 0 hectares

Question 2

1. The highest elevation on Anu ground map: 638.7800293
   The highest elevation on Rspe ground map: 574.8742229243662

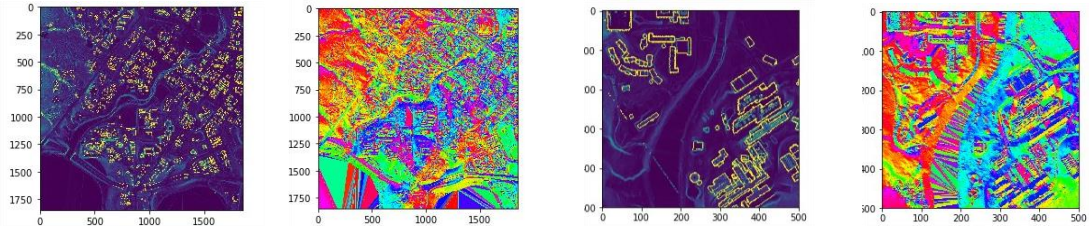
2. The highest elevation on Anu whole map: 638.7800293
   The highest elevation on Rspe whole map: 608.6250153261765

3. The tallest building of Rspe: 46.71239966334202
   The tallest building of Anu: 66.89814377000005

Question 3

1. For the boundaries of the map, I use the proportion to deal with it. for example, if you are in the right fringe (x, y), then I use the value of (x, y) and the height on its left based on the formula of "median line of a trapezoid" to calculate the height on its right. The up boundary and down boundary also be done in this way.

   The reason why I choose this method is that, in reality, the elevation of the terrain will not dramatically ascend or descend, it would descend or ascend by an appropriate height. Therefore, I assumed that in the fringe, the virtual height is can be approximated by the adjacent cell height, because Anu map also follow the nature rule.
2. Not only did I test the fringe cases, but also, I tested on the corners, which is because corners algorithm is a little more different than the fringe cases as it has to evaluate two virtual heights. Besides, I choose some value which will cause the zero-division error, however, the arctan seems to help me fix this error by returning zero.
3. (slope of anu graph) (aspect of anu graph) (slope of rspe graph) (aspect of rspe graph)



Question 4

1. Firstly, I assumed that if I in a particular cell, and I should move to the other one of 8 cells or 3 cells(corner) or 5 cells(edge), that is what I defined on the meaning of adjacent cell, the reason why I do this is that I think moving to diagonal is what a Roller-Blading player can do. Secondly, I made a definition of wining this match, which is a player who reaches the lowest height but not lower than the water level will win this match. In my opinion, a player who wins a match should arrive at the destination of a map, so in this case, the water level is the destination. Finally, I made an assumption of where to stop when arriving the water level or no other places can keep going, which is you will stop when the next height is lower than water level and you will stop at water level but you can reach to it. I defined it because I think if you reach a position where is lower than the water level, then

you will be in the lake, so it is meaningless and It does not make sense.

2. The reasons why my additional tests are appropriate are that it can test some edge cases and check all the situation where the player will stop, besides, the answers will satisfy all the assumptions.

   The first edge case is that you are in a particular cell and all the adjacent cells are higher or equal to the height where you are, so it will not move any more. After that, the edge case of player will start at the corners or the fringes will test my functionality of getting adjacent cells. Then, I deliberately make a map wherein some particular cells it has more than one minimum, so you must have to divert and choose an appropriate path which will reach the lowest height, besides, I created a heightmap where has more than one branches, so it has to make a wise choice on every branch and divert to it. At last, I wrote some test cases about the water level, I tested whether it will stop when it reaches the water level, and if there is no water level height, whether it will stop when it reaches the height which is the closed value to the water level compared to other heights.

3. The reasons why we need a more complex approach to solve this problem in a realistic fashion are we need to add an algorithm of choosing the path that will lead to the lowest elevation, actually I have already written the algorithm in my function, which uses the multiple minimum heights in a particular cell and use recursion by recalling the function again to find the partial path, then compare each path and merge the proper path with the previous path. Considering the reality, I think a player should find the shortest path and also the destination of the path is as close as possible to the water level. So, an algorithm of finding the shortest path is needed, when paths reaching the same height, therefore I should add a function of finding the shortest list (len ()) of all the possible lists to extract the appropriate path.

Question 5

1. I defined the criterion of whether two positions represent the same building is they are adjacent cell but not including the diagonal cells, just like a cross-shaped adjacent cell so they will be the same building points. This is correct because every building piece is constructed and jointed together so they must next to each other, if two cells are diagonal relation, then they are not next to each other.

2. First, I have to copy the heightmap, because I will do some operations on the list, because I do not want to change the value of the heightmap, subsequently, I traverse all the cells in the map, if it is not zero, it will be erased first then it will give you the four adjacent cells to see whether they are not zero or not, if so, the adjacent cell will be erased as well and keep performing the recursion. If it is all zeros, it will terminate and return the set of all the building coordinates and go back to the point next to the first cell of the map, and follow the algorithm and rule above until it encounters the last cell of the map.

3. The time complexity is O(4^n), the total complexity is (n^2+n^2*4^n), because I copy a list of lists first by using for loop and it will do n operations on n lists, so it is O(n^2) and a double 'for loop' was used, which has a function recursive call, and one time of each the function recursive call will cause 4 other new functions, so the time complexity is O(4^n)

4. My tests contain a lot of edge cases and contain all the cases that will happen, and they all will eliminate the contingency. First, I wrote some tests to test my function `near_four` to check whether it can return the proper adjacent four cells, the tests will include points in four fringes and four corners to see whether it works correctly or not in special cases. Second, I wrote some tests about the `find_buildings` such as a map with just one cell's buildings is contained, which tests on how it deals with a single building situation, and some tests about two or more building complex and the building complex are very close to each other but do not next to each other, which will check whether the building cells is appended into a proper building set. Except, I wrote the edge case of a map without any buildings on the map.

5. The largest building at the ANU: 24884 square metres and it is the "Copland building" .

Question 6

1. On the `line_of_sight` function, I made some assumptions, which is the (x1, y1) and (x2, y2) must on the heightmap and no matter the position where they are. In terms of the heights between two positions, I assumed that there exists a diagonal between two positions but the diagonal is not a straight line, which means I made some approximations on it. The approximations are that I am not going to take the average value if the diagonal actually crosses two cells, instead, it will move to the adjacent cell where is proper and keep going.

   I think this is correct because, in a grid chart, the diagonal shape of any rectangle should be irregular when you want to contain some cells for drawing the diagonal.

   On the `is_hiaf_visible` function, I assumed that the height of second position (x2, y2) is higher than the first point (x1, y1) and these two heights will not be a negative value, besides, I assumed that the heights between the two positions are well-proportioned distributed on the diagonal. The reason why I suppose it is correct is that the height on the diagonal should be one

after another and the map is also a grid chart, so the interval between each height on the diagonal should be equivalent.

2. It is hard to solve a problem in just one time so I divided the problems into three parts, first I should create a new map by adding two given heights together. Second, I wrote a lot of 'if statements' to judge and distinguish different situation of the two positions, and the situation includes the slope of the line of two points is negative or positive (if so they will just move to the adjacent diagonal cell) and the line is a vertical line or horizontal line(if so they will move upward or downward to one cell), case of the two points are overlapped or they are adjacent cell, which will just return an empty list. After judging the position, I wrote a function `line_of_sight` and the algorithm is that it will keep returning all the heights between diagonal until the two positions reach some special positions, which is the two positions are in the top left and bottom right of a 2x2 square or 3x3 square and two points are adjacent, all of the positions will have unique solutions as I mention above.

   For the `is_hiaf_visible` function, I used a 'for statement' in the algorithm, and I used the principle of similar triangle to judge whether the heights between the two positions, in the 'for statement', whether the ratio of the height between diagonal and the target height do not exceed the proportion of the corresponding similar triangle or not, if so, so it will visible, else, invisible.

3. The time complexity for `line_of_sight` is O(n^2) because the create map function was contained in it, which will traverse a list of lists and for each member which is a list will do some operations on each of its n members, so it is n^2, for the other parts of the function, that will have a loop, which has to traverse n members. The time complexity of `is_hiaf_visible` is O(n^2), the total complexity is (2n^2+2n+7), it contains the function of 'create map' and 'line of sight', so they dominate the time complexity, which is (n^2) and (n^2+n+1) respectively, so it will still be O(n^2)

4. For the 6a test, my tests contain all the edge cases that may happen, I chose some tests which has an irregular diagonal such as a rectangle case and some regular diagonal such as square cases, all of it can test whether the result corresponds to what I assumed about the diagonal above. Besides, I reverse the two positions, which will test whether my algorithm works in the same way as the normal one. In addition, I wrote some tests about the vertical line and horizontal line to see if they can just return the cells in the line except the start point and end point. At last overlapped points and adjacent points are included in this test, to check whether they can just follow the assumption, returning the empty list.

   For the 6b test, my tests contain some cases that the heights between two positions are proportionable and can form a similar triangle (not block the sight), which will test whether it follows the mathematics of similar triangle. And for the function cases of return value are False, I just slightly add the heights on last example by 0.1, to see what happen. Up to now, the tests above are all tested on a regular diagonal, so I wrote a test of irregular diagonal, besides special cases of the vertical line and horizontal line are also contained in these tests. At last, to eliminate the contingency, the case of all the heights, lower than the height of point (x1, y1), between two positions are included in the test, which is supposed to return True.

5. I think it corresponds to reality, because I used the concept of visual angle or viewing angle, because in reality, if you want to see some particular building, you must guarantee that the obstruction's heights between the target height do not exceed the proportion of the distance from your position to obstruction and the distance from your position to the target building. So, my approach in Question 6 just like this, the first height of the point is like the human body height and the second position height is similar to the target building height, all of the algorithms correspond to the visual angle.