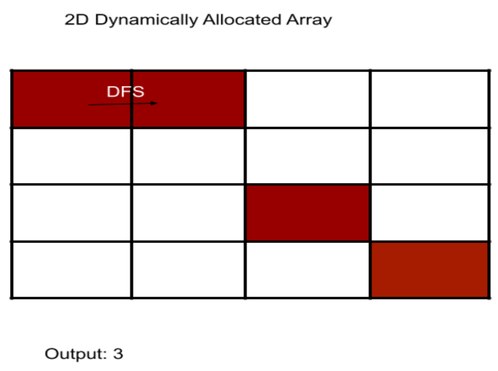
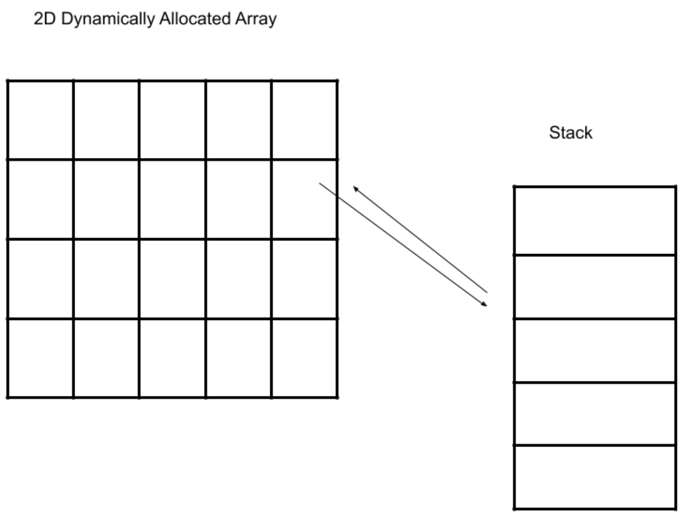
Recursive Method

The recursive solution of the project deals with 3 main functions that help the solution work recursively and a class called Object. The Object function has data fields of “value” which is type char and checked which is of type boolean. The value data field stores the value of the char that is read in from the input file. The checked data field holds a boolean value which stands whether that object has been checked by the DFS (Depth First Search) function . It is initially set to false since no object has been checked by the DFS function when the 2D array is first made. The first function, CountObjectsRecursive is basically a simple function that traverses the 2D array item by item. The input parameters are: A double pointer of Object (), totalRows (total number of rows which is read in from the text file) and totalCols (total number of columns which is read in from the text file). The double pointer is used to dynamically allocate a 2D array which is done with a for loop in main. The function finds the object count and the area of each object in the text file. It contains a double for loop which traverses through each item and when a square that is not 0 or not checked is found, a depth first search is conducted on that item in order to find the neighboring items and find the area of the object. The DFS function implements a depth first search algorithm which helps find the neighboring squares. The function takes in the double pointer to object, the current row and column, total rows and columns in the text file, and the area of the object that the algorithm is conducting the DFS on. It creates two arrays which helpS the recursive call in DFS to find the neighboring squares. The third function is check() which takes in the parameters: double pointer of Object, current row and column, and total rows and columns. It checks whether the row and column being searched are not out of range, the value is not 0, and the square has not been searched yet.



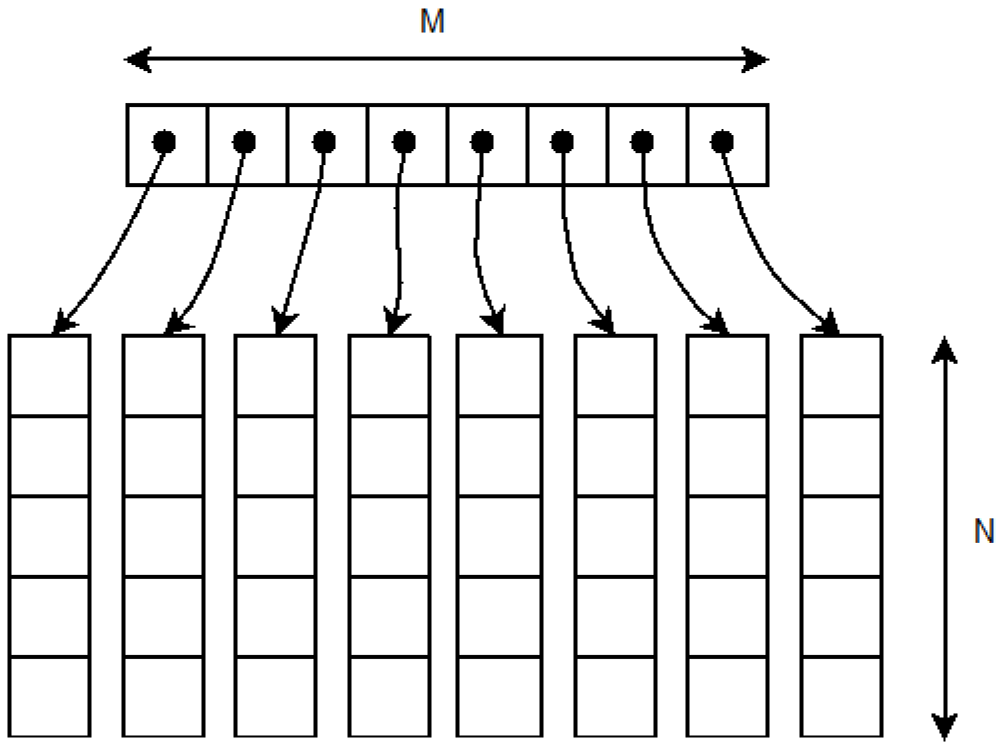
Iterative Method

The iterative solution of the project deals with 2 main functions that help the solution work recursively, a class called Object, and a Stack file for implementing stacks. The first function, CountObjectsIterative is basically a simple function that traverses the 2D array item by item. The input parameters are: A double pointer of Object (), totalRows (total number of rows which is read in from the text file) and totalCols (total number of columns which is read in from the text file). The double pointer is used to dynamically allocate a 2D array which is done with a for loop in main. The function finds the object count and the area of each object in the text file. It contains a double for loop which traverses through each item and when a square that is not 0 or not checked is found, an iterative depth first search is conducted on that item in order to find the neighboring items and find the area of the object. The DFS function is not called since it uses recursion. The function creates two arrays which helps the function iteratively find the neighboring squares. The stacks are used to store the locations of the neighboring squares that are being searched. At the end, the stacks are emptied when the area of the object is calculated.



Comparison

The time complexity and efficiency of both of the solutions are the same O(Rows x Cols). The advantage of the iterative solution is that it won't run into a stack overflow but if the stack gets too big, it might have memory errors. Personally, I would choose the recursive version because it is easier to understand and simpler to understand compared to the iterative solution since the DFS function makes it easy to search the neighboring squares.



Dynamically allocated 2D array