Assignment: asg5

Name: satish Dodda

ULID: sdodda1

Secret Directory: behappy

Methods discerptions:

**divide\_lines(lines):**, which accepts a list of lines as a parameter, will recessively divide the lines into two halves when the number of lines is greater than or equal to 2, and class merger function to merge the lines

**find\_intersection\_point(line1,line2):**  method accepts two parameters line1 and line2 which both contain slope and y intersect this will find the intersection point between these lines and it will return the inserting point

**merge\_lines(left\_half, right\_half):** method will merge the left and right half and check if any parallel lines are found if found, then it will remove the line with less y-intersect, in other cases , if three lines are interesting with each other then it will remove the invisible line

**Algorithm:**

It will read the slope and y-intersect and store in a list, and sort the list by using the slope , then it will pass the list to divide\_lines method will recursively the divided the lines into two half named left and right half’s ,if number of lines greater than or equal to 2, while backtracking it will call the merge function by passing left and right halfs to merge function here data is already is in sorted order so it will combine both the list and check if there are any parallel lines in the list if it found any parallel lines in the list then it will remove all the lines with less y-intersect , and it will check for another condition whether the three lines are intersecting or not , if three lines are intersecting and the inserting point of one and two is greater than one and three that means lines number 2 is invisible remove the line,this process will continue until all recursive calls are completed at the end it will return all visible lines

**DataStructures used :**

**Lines is a list to store the list of lines**

**Lines it is a list used in merge function to merge left and right halfs**

**Final\_visible\_lines is a list to store the lines after removing invisible parallel lines**

**Visible\_lines is a list that is used to store the lines after removing invisible intersecting lines**

**Time Complexity:**

**For recursively dividing, it will take O(logn)**

**For merging it will take O(N)**

**The recurrence relation is T(N) =2T(n/2)+O(n)**

**For diving and merging it will take O(NlogN)**

**For sorting it will take O(NLogN)**

**Overall time complexity will be O(NlogN)**

**Space Complexity:**

for storing the lines in the list it will take O(n)

for storing final\_visble\_lines it will take O(n)

for storing visible\_line it will take O(n)  
**Total space required is O(N)**

**Difficulties Faced:**

**The algorithm is a little bit tricky, if more than two lines are intersecting then only we will find some invisible lines , I spent some time to figure out the condition when three lines are inserting with each other and one and two inserting point is higher then one and three then the line in the middle is invisible for finding this point , I spend a lot of time**