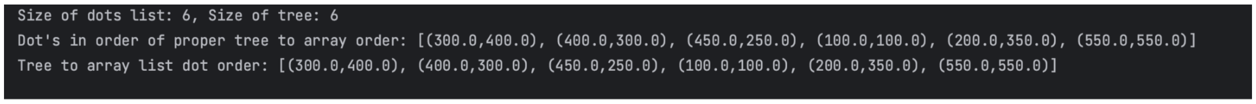
To make the pointQuadtree insert method we added a helper method to identify which quadrant we'd be adding the new point in since we knew we'd be using that value quite a bit. We chose to expand each quadrant by one row or column to account for new points being added on one or both of the axes. In the base case, where the point is inserted, we create new variables for the upper left and bottom right corner boundaries of the new region that the child creates and add the child to the variable corresponding to its quadrant.

For the findInCircle we chose to use a recursive helper and accumulator. The helper method finds all points that intersect with the circle and puts them in the main method's accumulator list.

Tree only tests:

As described in the instructions, we tested the PointQuadtree independently of any GUI to ensure that points were validly being inserted according to the algorithm. Below is the output of tests that show that the tree's methods are working as expected.

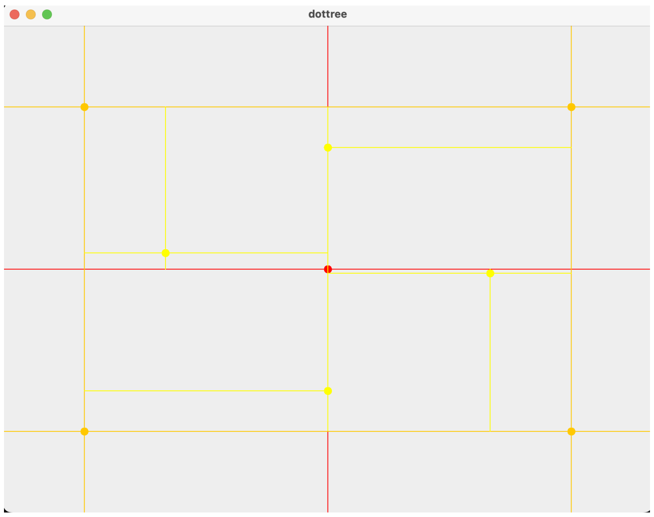


```
Size of dots list: 6, Size of tree: 6
Dot's in order of proper tree to array order: [(300.0,400.0), (400.0,300.0), (450.0,250.0), (100.0,100.0), (200.0,350.0), (550.0,550.0)]
Tree to array list dot order: [(300.0,400.0), (400.0,300.0), (450.0,250.0), (100.0,100.0), (200.0,350.0), (550.0,550.0)]
```

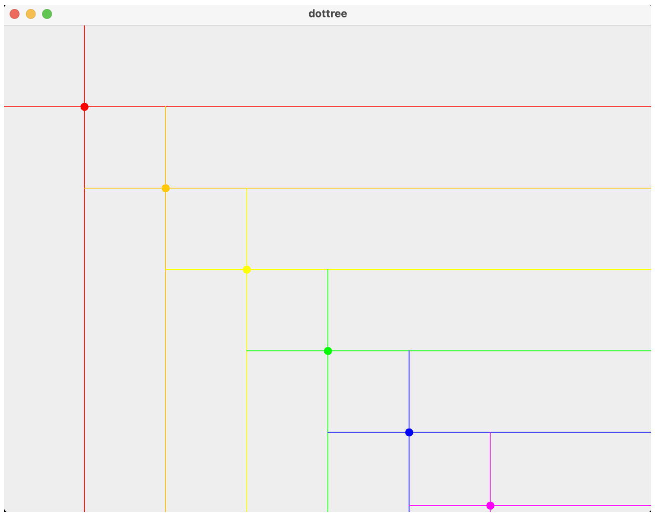
Quadtree test2 and test3 method write-up:

To create the new tests, we came up with points and then used the circle-finding visual mode and our understanding of the finding algorithm to come up with the necessary parameters for the testFind method. For test2, we focussed on testing points close to other points/regions. For test3, we used only concentric rectangles to make sure the algorithm worked in that context.

Test2

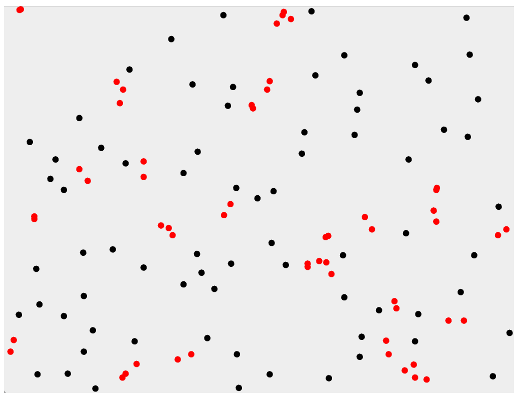


Test 3

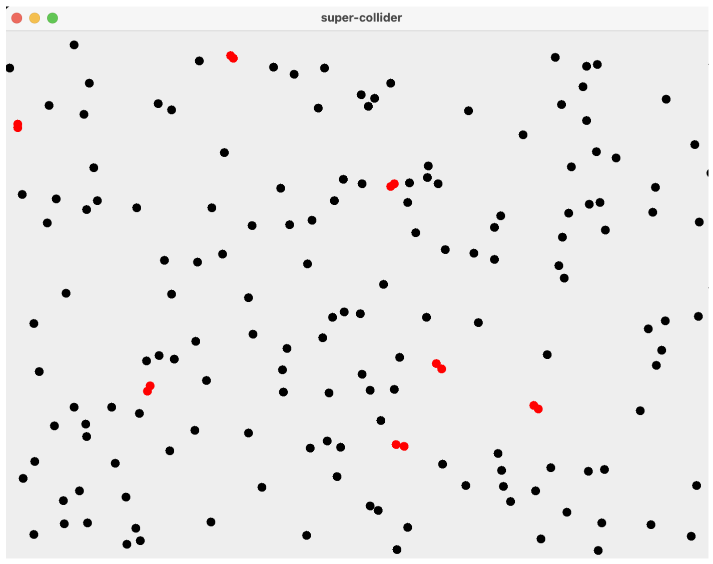


Collision detection tests:

Testing with a higher collision radius to see that multi-point collisions are happening and are not creating any unexpected visual collisions/artifacts



Test with many dots that require dots be touching or overlapping to collide shows that everything from the findDotsInCircle algorithm to the CollisionGUI class is properly working



In visual testing of the “destroy” mode, we launched pixels closely to one another and slowed them down to ensure that pixels would be removed after a collision