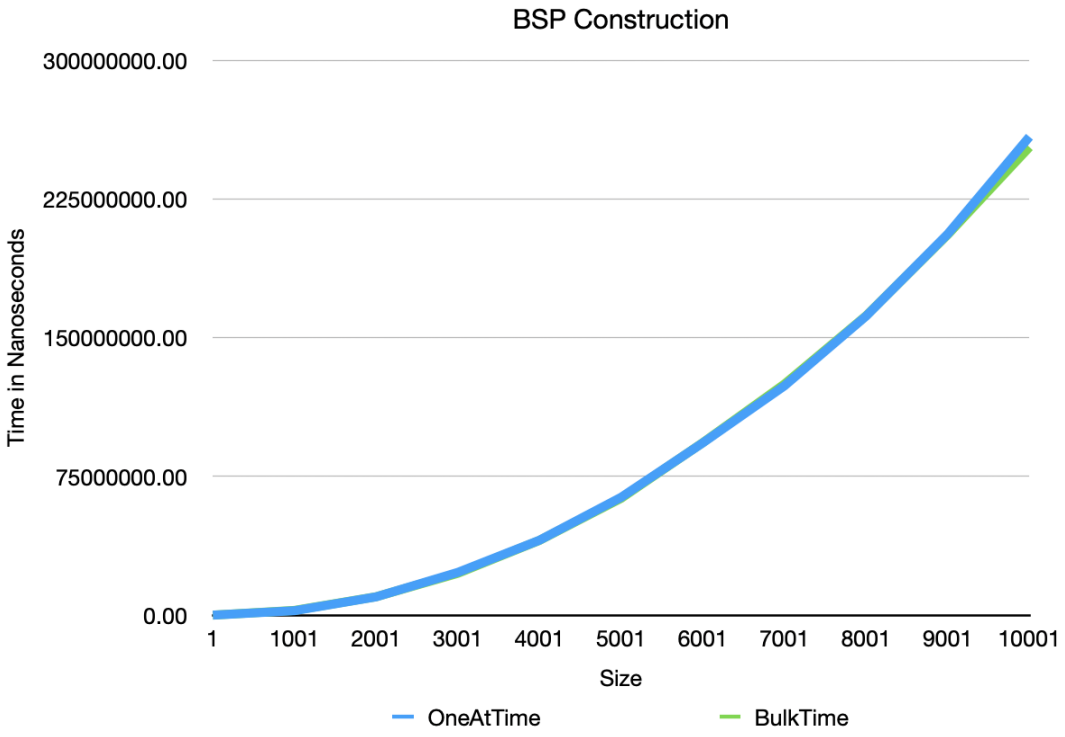
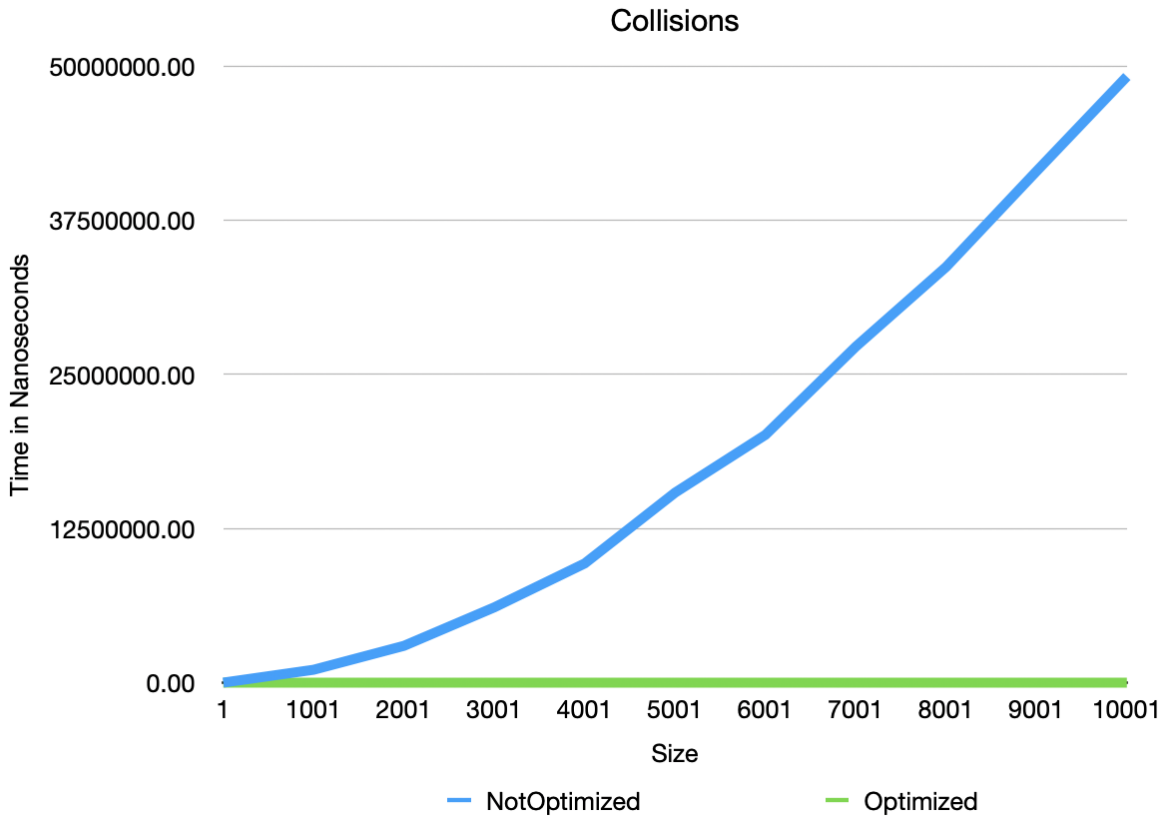
Assignment09 Analysis Document
By: Corinne Jones

Experiment 1 The following experiment compares the time required to construct a BSP tree with N elements (this correlates with "size" on the below graph's x axis) using a bulk construction and one at a time construction. This experiment was designed by first creating an outer for loop which would run the experiment X times, then divide it by the times ran, creating an average for each size. This loop began at 1, went to 100001, and incremented by 1000. Within the experiment, an ArrayList of segments was created, corresponding to the current size of the outer loop. Each segment had the coordinates "0, y, 1, y" with y+=.01 with each iteration of the loop. This essentially created horizontal segments that slightly increased on top of one another. A timer was then started and the whole ArrayList was added to a BSP tree, then the timer was stopped. The total time was calculated. A second experiment within the for loop reset the timer, started the time, and created an empty BSPTree. The ArrayList was then looped through, and each item was individually added.



Graph 2 Demonstrates the results from the experiment explained above. This plots two lines, the blue being the items from the ArrayList being added "one at a time," and the green line being the items added in bulk. As you can see, they have the same runtime. This is what I expected. Adding them one at a time calls the BSP.insert(a[i]) method. A for each loop went through and added every array item. When the items were added in bulk, it used the constructor which passed through all items in the array. This constructor creates a for each loop that goes through and adds every array item by calling the insert method. Essentially, these two different experiments called the same method, causing them to have the same runtime. The big O analysis of this is O (height).

Experiment 2 The following experiment compares the time required to check for collisions in a BSP tree with N elements (this correlates with “size” on the below graph’s x axis). This experiment was designed by first creating an outer for loop which would run the experiment X times, then divide it by the times ran, creating an average for each size. This loop began at 1, went to 100001, and incremented by 1000. Within the experiment, an ArrayList of segments was created, corresponding to the current size of the outer loop. Each segment had random coordinates, with the random being “seeded.” The ArrayList was then added into a BSP Tree. A query was created to check for collisions against. A timer was started, and the optimized collision check was called. The timer was then stopped and the average time required was calculated. The timer was reset, and the non optimized collision check was called. The timer was stopped and then this average time was calculated.



Graph 2 Demonstrates the results from the experiment explained above. This plots two lines, the blue being the runtime of the non optimized collision check and the green line being the optimized collision check. These run in the big O I expected. The non optimized has a big O of O(N) to check all segments, and the optimized is O(1). Part of the runtime of this is that the non optimized found many more collisions than optimized, since optimized did not run through every segment. Considering this, it makes sense that the not optimized grew in runtime and that the optimized stayed relatively even.