Team South 3/7/20

We believe that binary searches are more practical in complexity terms for sorted lists than linear searches. In ways that binary searches are more effective is when “find2()” is called this will have all the elements in an array where it checks the array in the middle for the requested object than checks the “sublist” (“subarray”) in the same motion. The “find()” will be the linear search where the iterator checks all objects from the beginning 1 by 1 and stops until the first object requested is found.

We’ll be using an array of cars as our objects, using a linear search will have “find()” starting at the beginning of the array and check 1 by 1 in order for what type of object was asked for. In complexity terms this is O(n). For example, when someone inputs “CarSet1.find(‘Car5’)” the “find()” will check every object to see where “Car5” is, the linear search will be CarSet1[Car1], CarSet1[Car2], CarSet1[Car3], CarSet1[Car4], CarSet1[Car5] and “Car5” is found. “find2()” will start in the middle of the array, compares what’s greater than or less than the selected object and will jump to the sub lists to compare again. In complexity terms this is O(log n). For example, when someone inputs “CarSet1[Car10]” the binary search will be compare CarSet1[Car2] Car2 with Car10: “Too cheap”, look for better car (Range=2-20), compare CarSet1[Car9] Car9 with Car10: “Too cheap”, look for better car (Range=10-20), compare CarSet1[Car12] Car12 with Car10: “Too expensive”, look for cheaper car (Range=10-11), compare CarSet1[Car10] Car10 with Car10: “That’s my car!!”, “I’ll take it!” (Range=10-10).