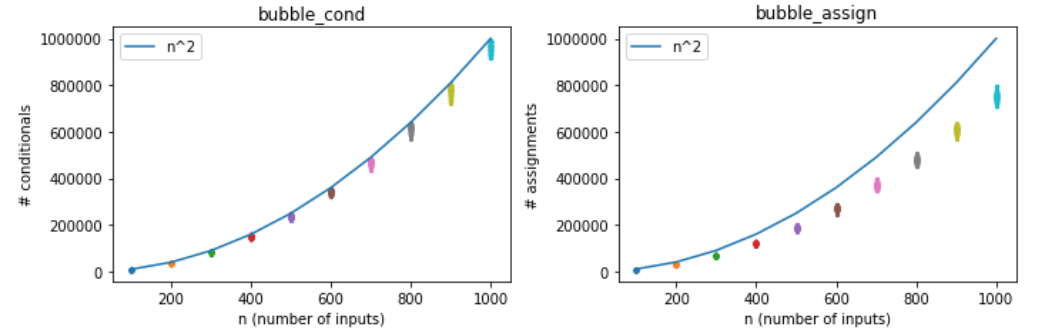
Jessica Gaines

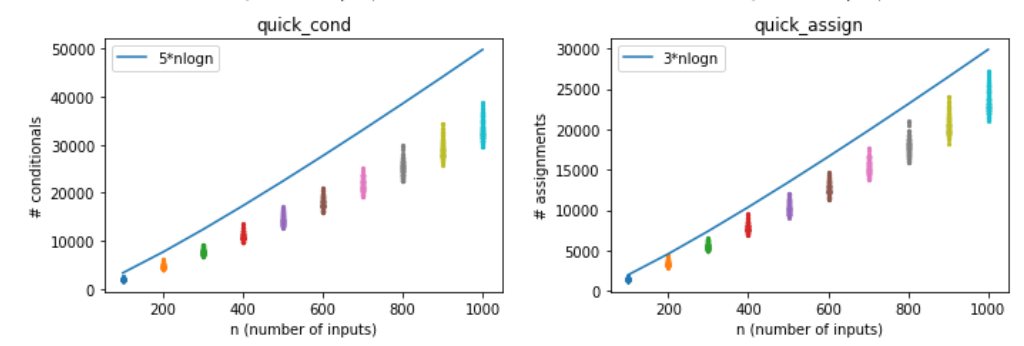
Github repository: <https://github.com/jessicagainesbmi203/example>

**Bubblesort:**



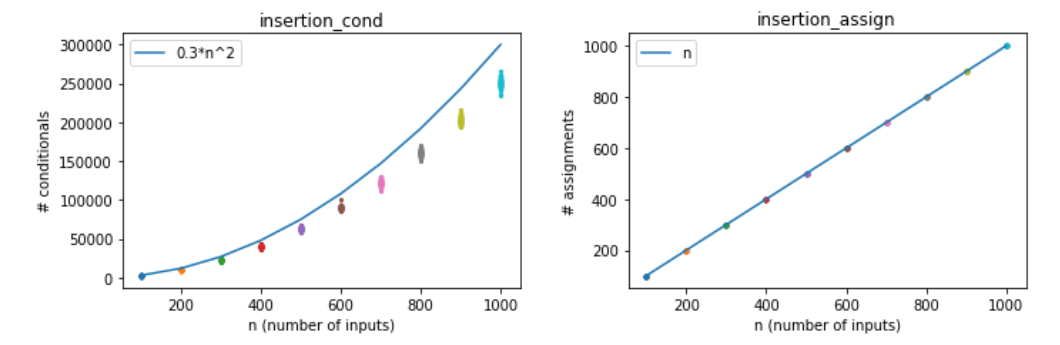
For the bubblesort algorithm, the number of conditionals that compared array elements were counted. Flag checks were not counted in the total number of conditionals because these do not compare elements of the array. The total number of assignments was incremented every time an array element was assigned to a new location, and every time a counter was assigned a new value. This included assignments to a temporary location used to swap elements. The number of conditionals used during sorting of 100 random arrays for each of 10 array sizes is plotted above on the left. The number of assignments used to sort the same arrays is shown on the right. It can be observed from the graphs that, while there is variability in the exact number of conditionals and assignments needed to sort an array, the number trends upwards with the size of the array in a pattern upper-bounded by the function y = n2, where n is the size of the array. As can be seen in the graphs, the increase in the number of conditionals and assignments with array size follows the same shape as y = n2 and does not exceed it, so the algorithm has O(n2).

**Quicksort:**



The quicksort algorithm also counted a conditional each time two array elements were compared. An assignment was counted whenever an element was assigned to a new array. An individual element assigned to a partition counted as one assignment, and merging partitions counted as n assignments because every element in each partition was assigned to the new, merged, array. The number of conditionals counted during sorting of 100 arrays of each of 10 array sizes are shown on the left, and the assignments counted while sorting those same arrays are shown on the right. As can be seen from the graph, the trend with which conditionals and assignments increase with array size is upper-bounded by a multiple of n\*log(n) where n is the array size. The number of conditionals and assignments needed to quicksort an array of size n follow the same shape and do not exceed the scaled n\*log(n) trendlines, so quicksort has O(n\*log(n)).

**Insertionsort:**



For insertion sort, conditionals were counted only when two array elements were compared. Assignments were counted only when an element was reassigned to a new array. Since each element is only placed once, the number of assignments exactly tracked the array size, which is O(n). This count did not account for the need to reassign all elements of the sorted array to a bigger array during each placement, because this happened within numpy’s insert function. If this had been accounted for, the assignment count would have had O(n2) because there would have been up to n assignments of elements to a bigger array for each of n new element added to the sorted array. The number of conditionals used to sort vectors of size n was upper-bounded by a scaled n2 function, following the same shape but not exceeding y = n2 in the graph on the left. This makes sense because in this algorithm, each element of the array can be compared to as many as all of the other elements in the array before it is placed in the correct location. Therefore, we can infer that the number of conditionals used in insertion sort is O(n2) while the number of assignments is O(n).