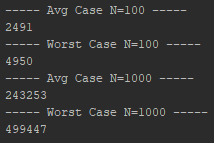
Adam Rolek

HW5



I ran this algorithm against 4 different unsorted arrays to study the time complexity.

Basic Operation: The comparison of key < array[j]

Average Case (A(n) = n2/4):

n = 100 should be roughly 2,500 basic operations

n = 1000 should be roughly 250,000 basic operations

Worst Case (W(n) = n2/2):

n = 100 should not exceed 4950 basic operations

n = 1000 should not exceed 499,500 basic operations

The worst-case time complexity would be the upper limit of this algorithm. There should not exist an input size for “n” where the algorithm will complete more basic operations than that equation would lead us to expect. It was interesting to see the algorithm function as intended. I ran this program countless times with randomly generated arrays. It never once exceeded the worst-case time complexity, this helped me reinforce the concept of worst-case time complexities and upper limits. To ensure I was reaching the worst-case scenario every iteration, I sorted the array in descending order first. The average case time complexity was also interesting to study with this implementation. The basic operations were very close to the calculated average case values. As for the best case, the array could contain a single element, or, the array could be sorted already. In this case, the basic operation would not be completed before the end of the algorithm was reached.