Matt Muldowney
Homework 6 due 4/2

Written Problems

Problem 1

Coding Problem 2 (palindrome):

1. **Cost: O(n)**. There is one for-loop that loops through the half of the array (a constant order of n) in the worst case.
2. **Extra memory: O(1)**. There is only a constant amount of memory being allocated.

Coding Problem 3 (insertion sort):

1. **Cost: O(n^2)**. The outer for loop is looping through n times in the worst case. Each of these times, the inner while loop is looping through on the order of n in the worst case.
2. **Extra memory: O(1)**. There is only a constant amount of memory being allocated.

Coding Problem 4 (merge sort):

1. **Cost: O(nlog3(n))**. Each recursive call of merge_sort3() deals with a subarray a third of the size of the previous subarray, so the number of times the subarrays are merged is log3(n). Each time subarrays are merged, merge3() is called, which in the worst case loops through the subarray passed to it n times.
2. **Extra memory: O(n)**. Each time merge3() is called, extra memory is allocated for the temporary array B, which is then freed once merge3() returns. The most that is set aside is the size of the initial array (n).