Christopher Mobley

CPSC 335

CWID: 891-10-3897

Project 1 Problem 1: Left-to-Right

**Pseudocode:**

int n; // Size of user input

int disk\_n; // Size of disk which is user input \* 2

// Loop to push dark (0) before light (1)

endOfList = size of disk (used to optimize by tracking right-most swap)

for i = 0 to n // Number of traverses across array

for k = 0 to disk\_n // Traverse array

if the two side-by-side values are [1|0]

then swap

Update endOfList to k (Optimize)

m++ // Count number of swaps

After traversing array, set disk\_n (size of disk) to endOfList (the last-most swap)

**Big-Oh Notation:**

Proof. Setting endOfList to size of disk takes 1 step. The first for loop executes n times. The second for loop executes 2n times. When the Boolean if statement returns TRUE, there are a few steps taken: Swapping of two values is 1 step, Updating end of list is 1 step, and increasing counter is 1 step. Finally, setting disk\_n to endOfList is 1 step. When Boolean if statement is FALSE, it continues traversing array. From what I understood in the book the if/else statement counts as two steps? So all together, the most inner loop will be at most 5 steps.

T(n) = 1 + Summation[n= 0 to n]( Summation[n=0 to 2n] (5)) + 1)

= 1 + n((5n) + 1)

= 1 + 5n^2 + n

T(n) = O(5n^2 + n + 1) = O(5n^2) = **O(n^2)**

Problem 2: Lawnmower

**Pseudocode:**

int n; // User input

int disk\_n; // Size of disk which is user input \* 2

// Loop to push dark (0) before light (1)

rightLimiter = disk\_n; (used to optimize by tracking right-most swap)

for i = 0 to n/2 // Number of traverses across array

for k = 0 to disk\_n // Traverse array left-to-right

if the two side-by-side values are [1|0]

then perform a swap

then update rightLimiter to k (for optimization)

then m++ // Count number of swaps

After traversing array, update disk\_n (size of disk) to rightLimiter (location of right most swap)

for k = n to 0

if the two side-by-side values are [1|0]

then perform a swap

then m++ // Increase swap count

**Big-Oh Notation:**

Proof. Setting rightLimiter to size of disk (disk\_n) take 1 step. The outside-most for loop takes n/2 times to execute. The first for loop within the outer-most for loop take n steps [(n/2)\*2)]. When the Boolean if statement returns TRUE, there are a few steps taken: Swapping of two values is 1 step, updating rightLimiter to k takes 1 step, and m++ the counter takes 1 step. When the Boolean if statement is FALSE, it continues traversing the array. Once the first for loop is done, it takes 1 step to update disk\_n to the rightLimiter. The second for loop within the outer-most for loop takes n steps (from rightLimiter counting down to 0). When the Boolean if statement returns TRUE, the two side-by-side values swap performing 1 step and increments counter for number of swaps which is 1 more step.

T(n) = 1 + Summation [n= 0 to n]( [Summation[n= 0 to n/2](5) + 1 \* Summation[n=0 to n](4) ] )

= 1 + n(5n + 1 \* 4n)

= 1 + 20n^2 + n

T(n) = O(20n^2 + n + 1 ) = O(20n^2) = **O(n^2)**