Darrick Rusk - drusk1@csu.fullerton.edu

Jimy Bombela - jimybombela@csu.fullerton.edu

Project 1: Implementing Algorithms

Alternating Algorithm:

Input: a positive integer n and a list of 2n disks of alternating colors light-dark, starting with light

Output: a list of 2n disks, the first n disks are light, the next n disks are dark, and an integer m representing the number of swaps to move the dark ones after the light ones.

Pseudocode:

```
def alternating(disks):
        swapCount = 0                                                    -
        for i = 0 to 2n  do
                indexCounter = 0
                while indexCounter + 1 is less than n do
                        if n is greater than (n + 1)
                                swap(indexCounter)
                                increment swapCount
                        endif
                indexCounter++
        return sorted disks
```

Big O Efficiency:

```
        swapCount = 0                       - 1 tu                              -
        for i = 0 to 2n - 1 do              - ((2n - 1 - 0) / 1) + 1 = 2n tu
                indexCounter = 0            -1 tu
                while indexCounter + 1 is less than n do:  - 3(n  + 1) = 3n + 3
                        if n is greater than (n + 1)     -1 + max (2,0) = 3 tu
                                swap(indexCounter)     -1 tu
```

```
                    increment swapCount  -1 tu
                endif
            indexCounter++                    -1u
        return sorted disks
```

= **2n(3n+ 3) + 1 = <u>6n^2 + 6n + 1</u>**

Proving 6n^2 + 6n + 1 belongs to O(n^2) using limit theorem:

f(n) = 6n^2 + 6n + 1

g(n) = n^2

lim f(n) / g(n) // as n -> infinity

= lim (6n^2 + 6n + 1) / (n^2) // as n -> infinity

using l'hopital:

lim (12n + 6) / 2n // as n -> infinity

using l'hopital again:

lim 12 / 2 // as n-> infinity

= 6 > 0

Because L = 6 > 0, by the Limit Theorem, we can say that 6n^2 + 6n + 1 belongs to O(n^2)

Time Complexity = O(n^2)

<u>Lawnmower Algorithm:</u>

Input: a positive integer n and a list of 2n disks of alternating colors light-dark, starting with light

Output: a list of 2n disks, the first n disks are light, the next n disks are dark, and an integer m representing the number of swaps to move the dark ones after the light ones.

Pseudocode:

```
def lawnmower(disks):
        swapCount = 0
        for i = 0 to n / 2  do:
                indexCounter = 0
                while indexCounter + 1 is less than n do
                        if n is greater than (n + 1)
                                swap(indexCounter)
                                increment swapCount
                        endif
                indexCounter++
        return sorted disks
```

Big O Efficiency:

```
def lawnmower(disks):
        swapCount = 0                    -1 tu
        for i = 0 to n / 2  do:          -((n / 2) /1) +1  = (n / 2) + 1 tu
                indexCounter = 0         -1 tu
                while indexCounter + 1 is less than n do        - 3(n + 1) = 3n + 3
                        if n is greater than (n + 1)        -1 + max(2,0) = 3 tu
                                swap(indexCounter)          -1 tu
                                increment swapCount         -1 tu
                        endif
                indexCounter++                              -1 tu
        return sorted disks
```

= (**(n / 2)+1) (3n+ 3) + 1 = ((3n^2)/2) + (3n / 2) + (3n) + 4 = <u>((3n^2)/2) + (9n/2) + 4</u>**

Proving (3n^2)/2 + (9n/2) + 4 belongs to O(n^2) using limit theorem:

f(n) = (3n^2)/2 + (9n/2) + 4

g(n) = n^2

lim f(n) / g(n) // as n -> infinity

= lim ((3n^2)/2 + 9n/2 + 4)/ (n^2) // as n -> infinity

using l'hopital:

lim (6n + 9/2) / 2n // as n -> infinity

using l'hopital again:

lim 6 / 2 // as n-> infinity

= 3 > 0

Because L = 3 > 0, by the Limit Theorem, we can say that (3n^2)/2 + (9n/2) + 4

belongs to O(n^2)

Time Complexity = O(n^2)

Screenshots: