

CPSC 335 - Algorithm Engineering

Project 1: Implementing Algorithms

Malka Ariel Lazerson
Fall 2021

Table of Contents

Names, CSUF Email, and Intent	2
ReadMe.md Screenshot	2
Code Compilation and Execution	3
Pseudocode and Step Count	4
Time Complexity	7

Names, CSUF Email, and Intent

Name: Malka Ariel Lazerson

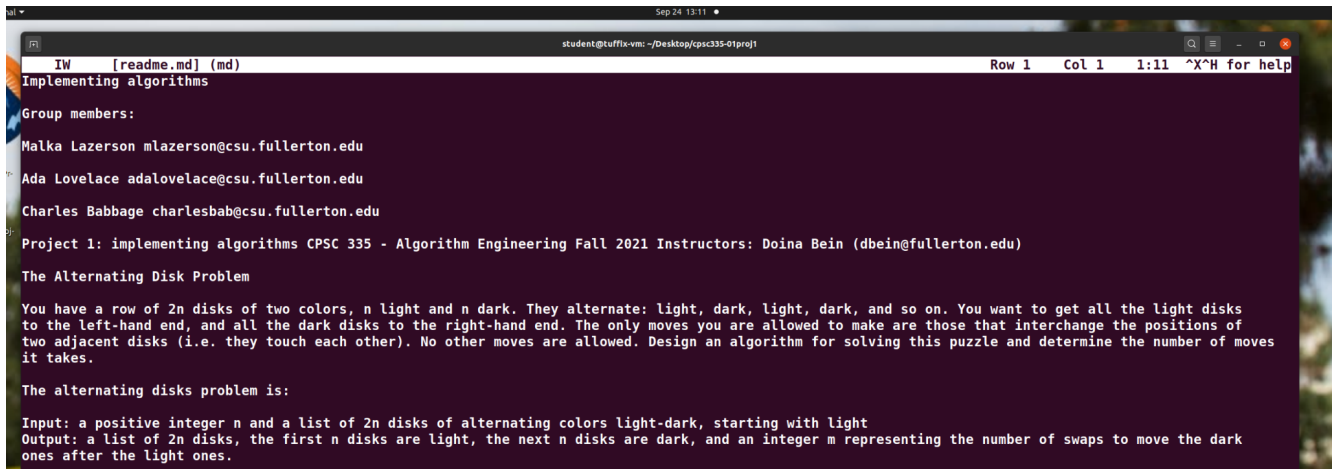
CSUF Email: mlazerson@csu.fullerton.edu

Intent: This document is intended to be one part of a submission for Project 1: Implementing Algorithms. This document contains....

1. The name of the student working on this project, the CSUF-supplied email address of said student, and an indication that the submission is for project 1.
2. A full-screen screenshot, inside Tuffix, of the readme.txt document opened by the jmacs editor containing the students name.
3. A full-screen screenshot showing your code compiling and executing.
4. Two pseudocode listings, for the two algorithms, and their step count.
5. A brief proof argument for the time complexity of your two algorithms.

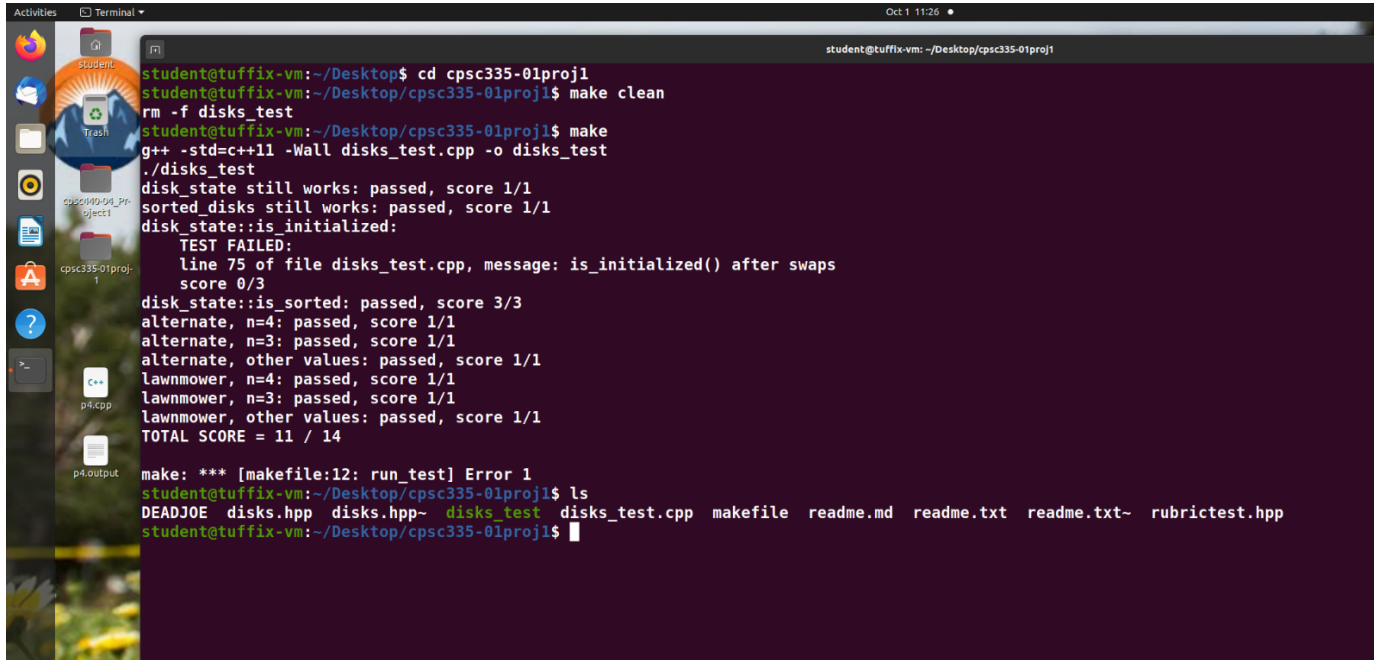
Code will be placed in the github files for the instructor to review

ReadMe.md Screenshot



```
student@tuffix-vm: ~/Desktop/cpsc335-01proj1
IW [readme.md] (md) Row 1 Col 1 1:11 ^X^H for help
Implementing algorithms
Group members:
Malka Lazerson mlazerson@csu.fullerton.edu
Ada Lovelace adalovelace@csu.fullerton.edu
Charles Babbage charlesbab@csu.fullerton.edu
Project 1: implementing algorithms CPSC 335 - Algorithm Engineering Fall 2021 Instructors: Doina Bein (dbein@fullerton.edu)
The Alternating Disk Problem
You have a row of 2n disks of two colors, n light and n dark. They alternate: light, dark, light, dark, and so on. You want to get all the light disks to the left-hand end, and all the dark disks to the right-hand end. The only moves you are allowed to make are those that interchange the positions of two adjacent disks (i.e. they touch each other). No other moves are allowed. Design an algorithm for solving this puzzle and determine the number of moves it takes.
The alternating disks problem is:
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.
```

Code Compilation and Execution



```
student@tuffix-vm: ~/Desktop$ cd cpssc335-01proj1
student@tuffix-vm: ~/Desktop/cpsc335-01proj1$ make clean
rm -f disks_test
student@tuffix-vm: ~/Desktop/cpsc335-01proj1$ make
g++ -std=c++11 -Wall disks_test.cpp -o disks_test
./disks_test
disk_state still works: passed, score 1/1
sorted_disks still works: passed, score 1/1
disk_state::is_initialized:
TEST FAILED:
line 75 of file disks_test.cpp, message: is_initialized() after swaps
score 0/3
disk_state::is_sorted: passed, score 3/3
alternate, n=4: passed, score 1/1
alternate, n=3: passed, score 1/1
alternate, other values: passed, score 1/1
lawnmower, n=4: passed, score 1/1
lawnmower, n=3: passed, score 1/1
lawnmower, other values: passed, score 1/1
TOTAL SCORE = 11 / 14
make: *** [makefile:12: run_test] Error 1
student@tuffix-vm: ~/Desktop/cpsc335-01proj1$ ls
DEADJOE disks.hpp disks.hpp~ disks_test disks_test.cpp makefile readme.md readme.txt readme.txt~ rubrictest.hpp
student@tuffix-vm: ~/Desktop/cpsc335-01proj1$
```

Pseudocode and Step Count

Lawn Mower Algorithm Pseudocode

.....Details.....

- Check indices to sort the disks
- Runs $2n$ times
- Compare $d[0]$ and $d[1]$, then $d[1]$ and $d[2]$, etc.
- Because we got left, right left, right, we compare $d[0]$ and $d[1]$, then $d[1]$ and $d[2]$, $d[2]$ and $d[3]$...and then $d[3]$ and $d[2]$, then $d[2]$ and $d[1]$, and lastly $d[1]$ and $d[0]$
- D = dark, L = light

//.....Pseudocode: Lawn Mower Algorithm.....

//record number of swap for return

NumSwaps = 0

//main for loop

for $i=0$ to $(n+1)/2$ do:

 //if starting on left side

 for $j=i+1$ to $2n-2$ do:

 //if and only if left = D and right= L

 if $\text{disk}[j]$ is D and $\text{disk}[j+1]$ is L

 swap $\text{disk}[j]$ and $\text{disk}[j+1]$

 NumSwaps++;

 //if starting on right side

 for $j=2n-1$ down to 1 do:

 //if and only if left = D and right= L

 if $\text{disk}[j-1]$ is D and $\text{disk}[j]$ is L

 swap $\text{disk}[j-1]$ and $\text{disk}[j]$

 NumSwaps++;

return NumSwaps;

.....STEP COUNT.....

```

//record number of swap for return
NumSwaps = 0 // 1 time

//main for loop
for i=0 to (n+1)/2 do: // (n+1)/2 times

    //if starting on left side
    for j=i+1 to 2n-2 do: // n/2 times

        //if and only if left = D and right= L
        if disk[j]== D and disk[j+1]== L // 3 times

            swap disk[j] and disk[j+1] // 1 time
            NumSwaps++; // 1 time

    //if starting on right side
    for j=2n-1 down to 1 do: // n/2 times

        //if and only if left = D and right= L
        if disk[j-1]== D and disk[j]== L // 3 times

            swap disk[j-1] and disk[j] // 1 time
            NumSwaps++; // 1 time

return NumSwaps; //1 time

```

.....STEP COUNT CALCULATIONS.....

$$\begin{aligned}
 &1 + ((n+1)/2) * (((n/2) * 3 + 1 + 1) + ((n/2) * 3 + 1 + 1)) + 1 = \\
 &1 + ((n+1)/2) * (((n/2) * 5) + ((n/2) * 5)) + 1 = \\
 &((n+1)/2) * (((n/2) * 5) + ((n/2) * 5)) + 2 = \\
 &((n+1)/2) * ((5n/2) + (5n/2)) + 2 = \\
 &((n+1)/2) * (10n/2) + 2 =
 \end{aligned}$$

Simplify

$$\begin{aligned}
 &(10n^2 + 10n) / (2) + 2 = \\
 &5n^2 + 5n + 2
 \end{aligned}$$

ANSWER: $5n^2 + 5n + 2$

Alternate Algorithm Pseudocode

.....Details.....

- check PAIRS, not indices, to sort the disks
- algorithm has $n+1$ runs
- compare $d[0]$ and $d[1]$, then $d[2]$ and $d[3]$, etc.
- D = dark, L = light

.....Pseudocode: Alternate Algorithm.....

```
//record number of swap for return
```

```
NumSwaps = 0
```

```
//To move by 2 spaces/1 pair each time....
```

```
// i=i+2 leads to outer for loop having i = 0,2,4,8...
```

```
//alternating algorithm has n+1 runs
```

```
for i=0 to n do:
```

```
    //to get 1 space for right most of the pair...
```

```
    //j=i/2 = 1,2,3,4...
```

```
    for j=i%2 to 2n-2 step 2 do:
```

```
        if disks[j] is D and disks[j+1] is L
```

```
            swap disks[j] and disks[j+1]
```

```
            NumSwaps++;
```

```
return NumSwaps;
```

.....STEP COUNT.....

```
//record number of swap for return
```

```
NumSwaps = 0 //1 time
```

```
//To move by 2 spaces/1 pair each time....
```

```
// i=i+2 leads to outer for loop having i = 0,2,4,8...even half of n
```

```
for i=0 to n do: //outer loop runs n+1 times
```

```
//to get 1 space for right most of the pair...
//j= i mod 2 = 1,3,5,7...odd half of n
for j=i%2 to 2n-2 step 2 do: //inner loop runs n/2 times
```

```
    if disks[j] is D and disks[j+1] is L // 3 times
```

```
        swap disks[j] and disks[j+1] //1 time
```

```
        NumSwaps++; // 1 time
```

```
return NumSwaps; // 1 time
```

.....STEP COUNT CALCULATIONS.....

$$1 + ((n+1) * ((n/2) * (3 + 1 + 1))) + 1 =$$

$$(n+1) * ((n/2) * 5) + 2 =$$

$$(n+1) * (5n/2) + 2 =$$

$$5n^2 + 5n/2 + 2$$

$$(5n^2 + 5n) / (2) + 2$$

Clean up by clearing denominator

$$(2) * (5n^2 + 5n) / (2) + 2 =$$

$$10n^2 + 10n + 2$$

$$\text{ANSWER: } 10n^2 + 10n + 2$$

Time Complexity

Lawn Mower Algorithm Time Complexity

$5n^2 + 5n + 2$ looks to be $O(n^2)$, but proof is needed.

“ $O(f(n)) = \{g(n) \mid \text{there exists some constants } c > 0 \text{ and } t \geq 0 \text{ such that } g(n) \leq c \cdot f(n) \text{ whenever } n \geq t\}$.”

BASE CASE: “n” is defined when $n = 0$, as proven below.....

$$5n^2 + 5n + 2 \leq c * f(n)$$

$$c > 5n + 5 + 2/n$$

So we will use $t = 1$

$$\begin{aligned} t(n) &= 5(1)^2 + 5(1) + 2 = \\ 5 + 5 + 2 &= 12 \end{aligned}$$

$$c * f(n) = 12n = 12(1) = 12$$

So $12 \leq 12$, proving there exists a constant $c > 0$ and $t \geq 0$ such that $g(n) \leq c * f(n)$ if $n \geq t$

INDUCTIVE STEP: If $n > t$ and $T(n) \leq c \cdot f(n)$, then $T(n + 1) \leq c \cdot f(n + 1)$

Let $T(n) \leq c * f(n)$

$$5n^2 + 5n + 2 \leq 12n$$

$$\begin{aligned} &\quad +2 \quad \quad + 12 \\ 5n^2 + 5n + 4 &\leq 12n + 12 \end{aligned}$$

$$5(n^2 + n) + 4 \leq 12(n + 1)$$

If $n = 1$

$$5(1 + 1) + 4 = 14$$

$$12(1 + 1) = 12(2) = 24$$

$$14 \leq 24$$

Thus, $T(n) \leq c * f(n)$ and $T(n + 1) \leq c * f(n + 1)$

ANSWER: Therefore, the mathematical proof by induction shows this algorithm's Big O Time Complexity is $O(n^2)$

Alternate Algorithm Time Complexity

$10n^2 + 10n + 2$ looks to be $O(n^2)$, but proof is required....

“If $F(x)$ is a real-valued function, then the limit of F as x approaches infinity is L ,

$\lim_{x \text{ approaches infinity}} F(x) = L$

means that for any $\varepsilon > 0$, there exists k such that $|F(x) - L| < \varepsilon$ whenever $x > k$.”

$T(n) = 10n^2 + 10n + 2 = O(n^2)$ using the limit definition....

$\lim_{n \text{ approaches infinity}} T(n)/f(n)$

$$10n^2 + 10n + 2 / n^2 =$$

$$\lim 10n^2 / n^2 + \lim 10n/n^2 + \lim 2/n^2 =$$

$$10 + 10/n^2 + 2/n^2 =$$

$$10 + 0 = 10$$

10 is not negative and constant

Thus, $10n^2 + 10n + 2$ is $O(n^2)$

ANSWER: Therefore, the mathematical proof by limits shows this algorithm's Big O Time Complexity is $O(n^2)$