

# Project 1: implementing algorithms

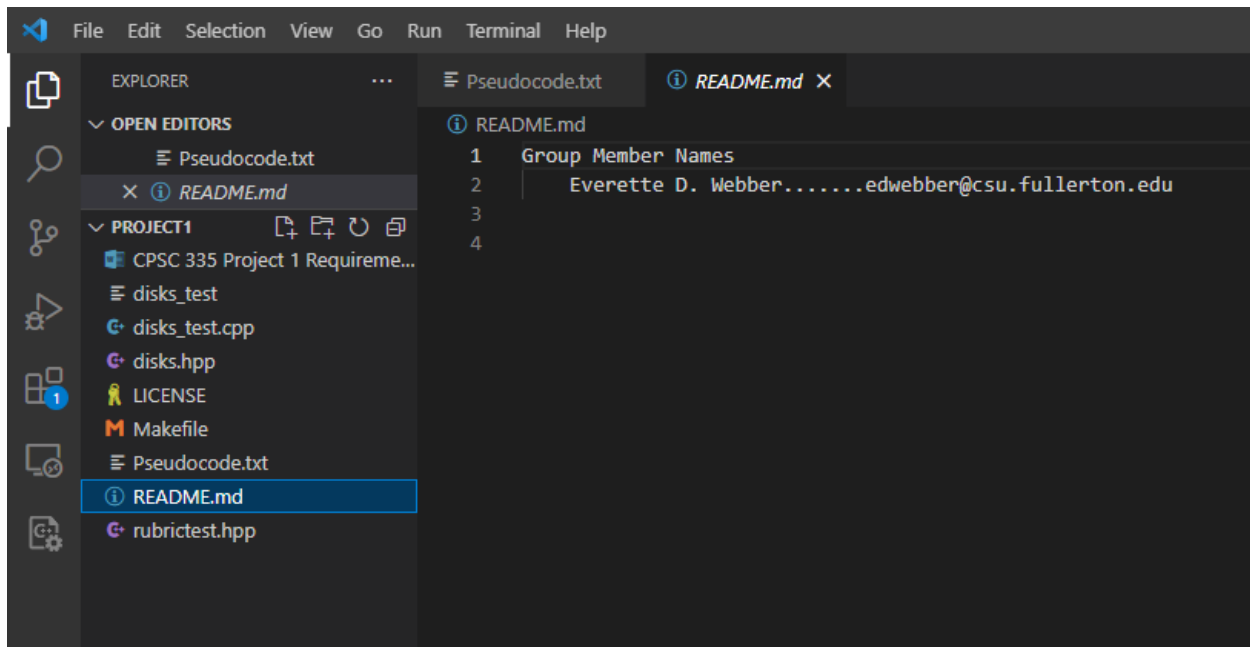
CPSC 335 - Algorithm Engineering

Fall 2022

Instructors: Himani Tawade([htawade@fullerton.edu](mailto:htawade@fullerton.edu))

Group Members: Everette D. Webber([edwebber@csu.fullerton.edu](mailto:edwebber@csu.fullerton.edu))

Screenshot of README.md



Screenshot of Code Compiling and Executing in the terminal

```
digx7@LAPTOP-80GMBDP7: ~/Algorithms/Projects/Project1$ ll
total 320
drwxrwxr-x 2 digx7 digx7 4096 Mar 17 15:46 ./
drwxrwxr-x 3 digx7 digx7 4096 Mar 16 22:55 ../
-rw-r--r-- 1 root root 66645 Feb 22 09:02 'CPSC 335 Project 1 Report.docx'
-rw-r--r-- 1 root root 66645 Feb 22 09:02 'CPSC 335 Project 1 Requirements.docx'
-rw-r--r-- 1 root root 1097 Feb 22 09:02 LICENSE
-rw-r--r-- 1 root root 363 Feb 22 09:02 Makefile
-rw-r--r-- 1 root root 2447 Mar 17 15:33 Pseudocode.txt
-rw-r--r-- 1 root root 79 Mar 16 23:47 README.md
-rw-r--r-- 1 root root 5601 Mar 16 23:45 disks.hpp
-rwxrwxr-x 1 digx7 digx7 132760 Mar 17 15:46 disks_test*
-rw-r--r-- 1 root root 6914 Feb 22 09:02 disks_test.cpp
-rw-r--r-- 1 root root 5734 Feb 22 09:02 rubricstest.hpp
-rw-r--r-- 1 root root 162 Mar 17 15:40 '~$SC 335 Project 1 Report.docx'
digx7@LAPTOP-80GMBDP7:~/Algorithms/Projects/Project1$ 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: passed, score 3/3
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 = 14 / 14
digx7@LAPTOP-80GMBDP7:~/Algorithms/Projects/Project1$
```

## Pseudocode

### Lawnmower algorithm

// Inputs

Given  $n$

Given DiskList[ $2n$ ]

// Performance

Step Count =  $(26n^2 + 14n - 12)$

Given that  $n^2$  is the largest variable than this algorithm has  $O(n^2)$

// Algorithm

While the numberOfSwitches > 0 do:

    // resets the number of switches for each run  
    numberOfSwitches = 0

    // moves from left to right

    for  $i = 0, i$  to  $2n-1$  do:

        if DiskList[ $i$ ] == D && DiskList[ $i + 1$ ] == L do:

            DiskList.swap( $i$ )

            numberOfSwitches += 1

    // moves from right to left

    for  $i = (2n-1), i$  to 0 do:

        if DiskList[ $i$ ] == L && DiskList[ $i - 1$ ] == D do:

            DiskList.swap( $i-1$ )

            numberOfSwitches += 1

### Alternate algorithm

// Inputs

Given  $n$

Given DiskList[ $n$ ]

// Performance

Step Count =  $(12n^2 + 22n - 4)$

Given that  $n^2$  is the largest variable than this algorithm has  $O(n^2)$

// Algorithm

While the numberOfSwitches > 0 do:

    // resets the number of switches for each run

    numberOfSwitches = 0

    // does the first run starting at the left most position

    for i = 0, i to 2n at i+2 do:

        if DiskList[i] == D && DiskList[i + 1] == L do:

            DiskList.swap(i)

            numberOfSwitches += 1

    // does the second run starting at the second left most position

    for i = 1, i to (2n-1) at i+2 do:

        if DiskList[i] == D && DiskList[i + 1] == L do:

            DiskList.swap(i)

            numberOfSwitches += 1

## Proof For Time Complexity's

### Lawnmower Algorithm

Step Count = countInsideWhileLoop \* numberOfWhileLoops

numberOfWhileLoops = n + 1

countInsideWhileLoop = 1 + firstForLoop + secondForLoop

firstForLoop = countInsideFirstForLoop \* numberOfFirstForLoops

countInsideFirstForLoop = 6

numberOfFirstForLoops = 2n - 1

firstForLoop = 6(2n-1)

secondForLoop = countInsideSecondForLoop \* numberOfSecondForLoops

countInsideSecondForLoop = 7

numberOfSecondForLoops = (2n-1)

secondForLoop = 7(2n-1)

countInsideWhileLoop = 1 + 6(2n-1) + 7(2n-1) => (26n-12)

Step Count = (26n - 12)(n + 1) => (26n<sup>2</sup> + 14n - 12)

As  $n$  approaches infinity  $(26n^2 + 14n - 12)$  will approach  $(26n^2)$  meaning the Lawnmower Algorithm as a time complexity of  $O(n^2)$

### **Alternate Algorithm**

Step Count = countInsideWhileLoop \* numberOfWhileLoops

numberOfWhileLoops =  $n + 2$

countInsideWhileLoop = 1 + firstForLoop + secondForLoop

firstForLoop = countInsideFirstForLoop \* numberOfFirstForLoops

countInsideFirstForLoop = 6

numberOfFirstForLoops =  $2n/2$

firstForLoop =  $6(2n/2)$

secondForLoop = countInsideSecondForLoop \* numberOfSecondForLoops

countInsideSecondForLoop = 6

numberOfSecondForLoops =  $(2n-1)/2$

secondForLoop =  $6((2n-1)/2)$

countInsideWhileLoop =  $1 + 6(2n/2) + 6((2n-1)/2) \Rightarrow (12n-2)$

Step Count =  $(12n-2)(n+2) \Rightarrow (12n^2 + 22n - 4)$

As  $n$  approaches infinity  $(12n^2 + 22n - 4)$  will approach  $(12n^2)$  meaning the Lawnmower Algorithm as a time complexity of  $O(n^2)$