

Algorithm Project

CPSC 335

October 8th, 2022

By

Luis Alvarado - Luisalvarado@csu.fullerton.edu

Marco Gabriel - Marcog10@csu.fullerton.edu

Table Of Contents:

Algorithm Design	2
Alternate Algorithm	2
Pseudocode	2
Lawnmower Algorithm	3
Pseudocode	3
Mathematical Analysis	4
Alternative Algorithm	4
Step-Count, Limits Theorem, and Proof by Contradiction	4
Lawnmower Algorithm	5
Step-Count, Limits Theorem, and Proof by Contradiction	6
Screenshots	7

Algorithm Design

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

0 = Light

1 = Dark

Ex. 0 1 0 1 0 1 0 1 0 1

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.

Result Ex. 0 0 0 0 1 1 1 1

Alternate Algorithm

Pseudocode

```
sorted_disks sort_alternate(const disk_state& before) {
    initialize numOfSwap to zero
    initialize a variable from disk_state to before

    for each element in total_count() do
        for each element in total_count() - 1 do
            if get(j) is greater than get(j + 1) then do
                swap(j)
                Increment numOfSwap
            endif
        endfor
    endfor

    return sorted_disks()
}
```

Lawnmower Algorithm

Pseudocode

```
sorted_disks sort_lawnmower(const disk_state& before) {  
    initialize numOfSwap to zero  
    initialize a variable from disk_state to before  
  
    //left to right method  
    for each element in ligh_count() do  
        for each element in total_count() - 1 do  
            if get(j) is greater than get(j + 1) then  
                swap(j)  
                Increment numOfSwap  
            endif  
        endfor  
  
    //right to left method  
    For each element in total_count() - 1; decrement by 1  
        If get(k) is less than get(k-1) do  
            swap(k-1)  
            Increment numOfSwap  
        endif  
    endfor  
endfor  
    return the sorted_disks()  
}
```

Mathematical Analysis

Alternative Algorithm

```
sorted_disks sort_alternate(const disk_state &before) { // record # of step swap
    int numOfSwap = 0;
    disk_state step = before;
    for (size_t i = 0; i < step.total_count(); i++) {
        for (size_t j = 0; j < step.total_count() - 1; j++) {
            if (step.get(j) > step.get(j + 1)) {
                step.swap(j);
                numOfSwap++;
            }
        }
    }
    return sorted_disks(disk_state(step), numOfSwap);
}
```

Handwritten annotations on the code:

- SC** (Step-Count) is written next to the initialization of `numOfSwap`.
- SCA** (Step-Count A) is written next to the outer loop.
- SCB** (Step-Count B) is written next to the inner loop.
- IF** is written next to the `if` statement.

Step-Count, Limits Theorem, and Proof by Contradiction

Step-Count:

$$J.C = 1 + 1 + J.C.A \rightarrow 2 + 3n^2 + 3n + 0 \rightarrow 3n^2 + 3n + 2$$

$$J.C.A = n + 1 + S.C.B \rightarrow n + 1 + (3n) = 3n^2 + 3n + 0$$

$$S.C.B = n \quad \& \quad I.F \rightarrow n(3) \rightarrow 3n$$

$$S.C.I.F = 2 + \max(1, 1)$$

$$2 + 1 = 3$$

$$3n \quad \begin{array}{|c|c|c|} \hline n & + & j \\ \hline 3n & 3n^2 & 3n \\ \hline +0 & 0 & 0 \\ \hline \end{array} \quad 3n^2 + 3n + 0$$

Limits theorem:

$$f(n) \in O(n^2)$$

$$\lim_{n \rightarrow \infty} \frac{3n^2 + 3n + 2}{n^2}$$

$$\lim_{n \rightarrow \infty} \frac{6n + 3 + 0}{2n} \rightarrow$$

$$\lim_{n \rightarrow \infty} \frac{6 + 0 + 0}{2} \rightarrow \frac{6}{2} \rightarrow 3$$

by limits theorem, we can conclude that $3n^2 + 3n + 2 \in O(n^2)$

Proof by contradiction

$$3n^2 + 3n + 2 \leq n^2$$

$$\underbrace{\quad}_{f(n)} \quad \underbrace{\quad}_{g(n)}$$

$$C = 3 + 3 + 2 \rightarrow 8, n_0 = 1$$

$$3n^2 + 3n + 2 \leq 8n^2$$

$$3(1) + 3(1) + 2 \leq 8(1)$$

$$3 + 3 + 2 \leq 8$$

$$8 \leq 8$$

True, by definition, $3n^2 + 3n + 2 \in O(n^2)$

Lawnmower Algorithm

```

sorted_disks sort_lawnmower(const disk_state &before) {
    //Step_Count
    int numOfSwap = 0; // 1 time unit
    disk_state step = before; // 1 time unit

    for(int i = 0; i < step.light_count(); i++) {
        // left to right - compares every two adjacent disks and swaps if necessary (i%2 == 0) - meaning it is even
        for(size_t j = 0; j < step.total_count() - 1; j++) {
            if (step.get(j) > step.get(j + 1)) {
                step.swap(j);
                numOfSwap++;
            }
        }
        // else it is odd
        //right to left - compares every two adjacent disks and swaps if necessary
        for (size_t k = step.total_count() - 1; k > 0; k--) {
            if (step.get(k) < step.get(k - 1)) {
                step.swap(k - 1);
                numOfSwap++;
            }
        }
    }

    return sorted_disks(step, numOfSwap);
}

```

Handwritten annotations on the code:

- SC** (Step Count) is written next to the initial variable declarations.
- SCA** (Step Count for the first pass) is written next to the first `for` loop.
- SCB** (Step Count for the first pass inner loop) is written next to the first `for` loop's inner loop.
- SCIF** (Step Count for the first pass inner loop if) is written next to the `if` statement in the first pass.
- SCD** (Step Count for the second pass) is written next to the second `for` loop.
- SCIF** (Step Count for the second pass inner loop if) is written next to the `if` statement in the second pass.

Step-Count, Limits Theorem, and Proof by Contradiction

Step count:

$$SC = 2 + SCA \rightarrow \frac{6n^2 + 6n + 2}{2} \rightarrow 3n^2 + 3n + 2$$

$$SCA = \frac{n+1}{2} (SCB) (SCD) \rightarrow \frac{n+1}{2} (3n) (4n) \rightarrow 12n$$

$$SCB = n - 1 + 1 \rightarrow n(SCIF) \rightarrow 3n \rightarrow \left(\frac{n+1}{2}\right) \left(\frac{12n}{1}\right)$$

$$SCIF = 2 + \max(1, 1) \rightarrow 3 \rightarrow \frac{12n^2 + 12}{2}$$

$$SCD = n(SCIF) \rightarrow 4n \rightarrow 6n^2 - 6$$

$$SCIF = 2 + \max(2, 1) \rightarrow 3$$

$$= 2 + 2$$

$$= 4$$

Limits theorem:

$$f(n) \in O(n^2)$$

$$\lim_{n \rightarrow \infty} \frac{3n^2 + 3n + 2}{n^2}$$

$$\lim_{n \rightarrow \infty} \frac{6n + 3 + 0}{2n}$$

$$\lim_{n \rightarrow \infty} \frac{6 + 0 + 0}{2} \rightarrow 3 \rightarrow 0$$

by limits theorem, we can conclude that $3n^2 + 3n + 2 \in O(n^2)$

Proof by contradiction

$$3n^2 + 3n + 2 \leq n^2$$

$$f(n) \quad g(n)$$

$$0 = 3 + 3 + 2 \rightarrow 8, n_0 = 1$$

$$3n^2 + 3n + 2 \leq 0n^2$$

$$3(1) + 3(1) + 2 \leq 0(1)$$

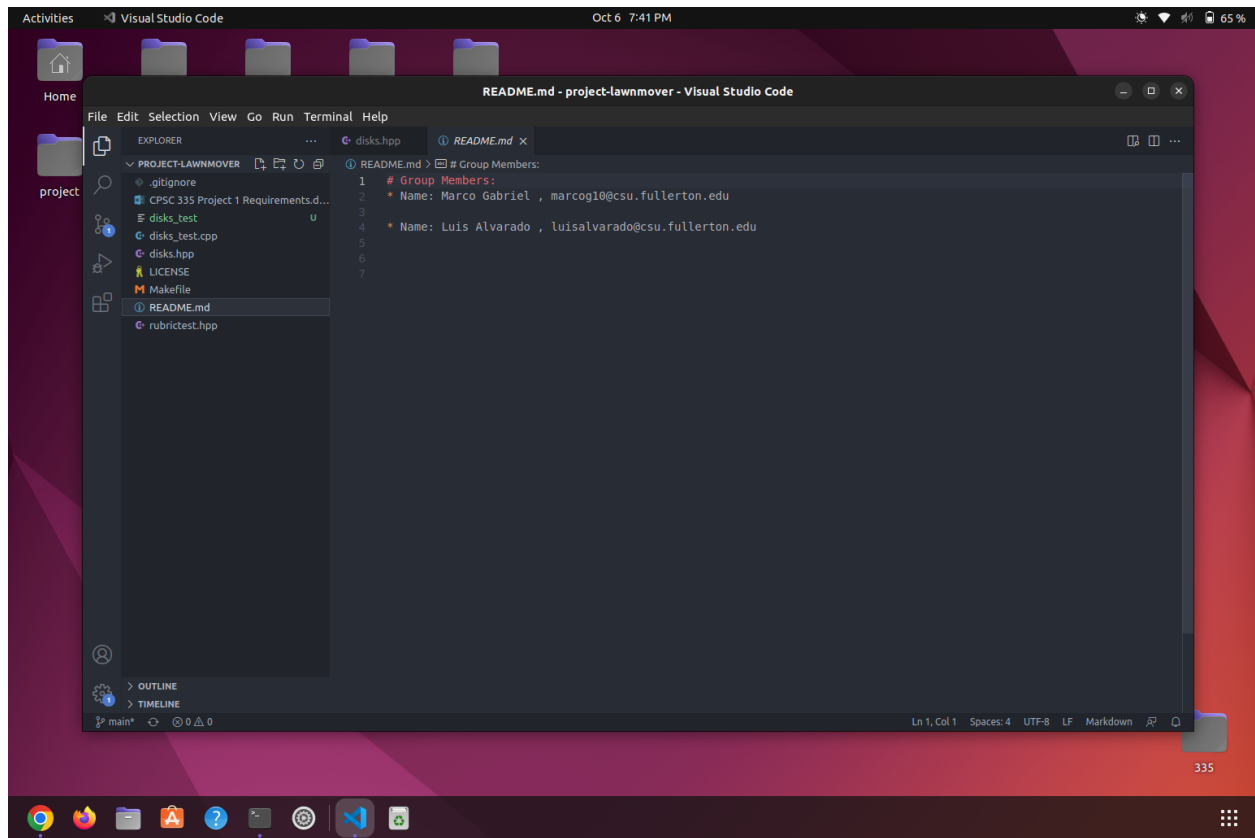
$$3 + 3 + 2 \leq 0$$

$$8 \leq 0$$

True, by definition, $3n^2 + 3n + 2 \in O(n^2)$

Screenshots

README.MD



IS_SORTED FUNCTION:

```
102 // @Breif: Return true when this disk_state is fully sorted, with all light disks on
103 // the left (low indices) and all dark disks on the right (high indices).
104 bool is_sorted() const {
105     // loop through the total_count()
106     for (size_t i = 0; i < total_count(); i++) {
107         if (i < total_count() / 2) {
108             if (_colors[i] == DISK_DARK) {
109                 return false;
110             }
111         } else {
112             if (_colors[i] == DISK_LIGHT) {
113                 return false;
114             }
115         }
116     }
117     return true;
118 }
119 }
```

SORT_ALTERNATE FUNCTION

```
sorted_disks sort_alternate(const disk_state &before) { // record # of step swap
    //Step Count
    int numOfSwap = 0; // 1 times
    disk_state step = before; // 1 times

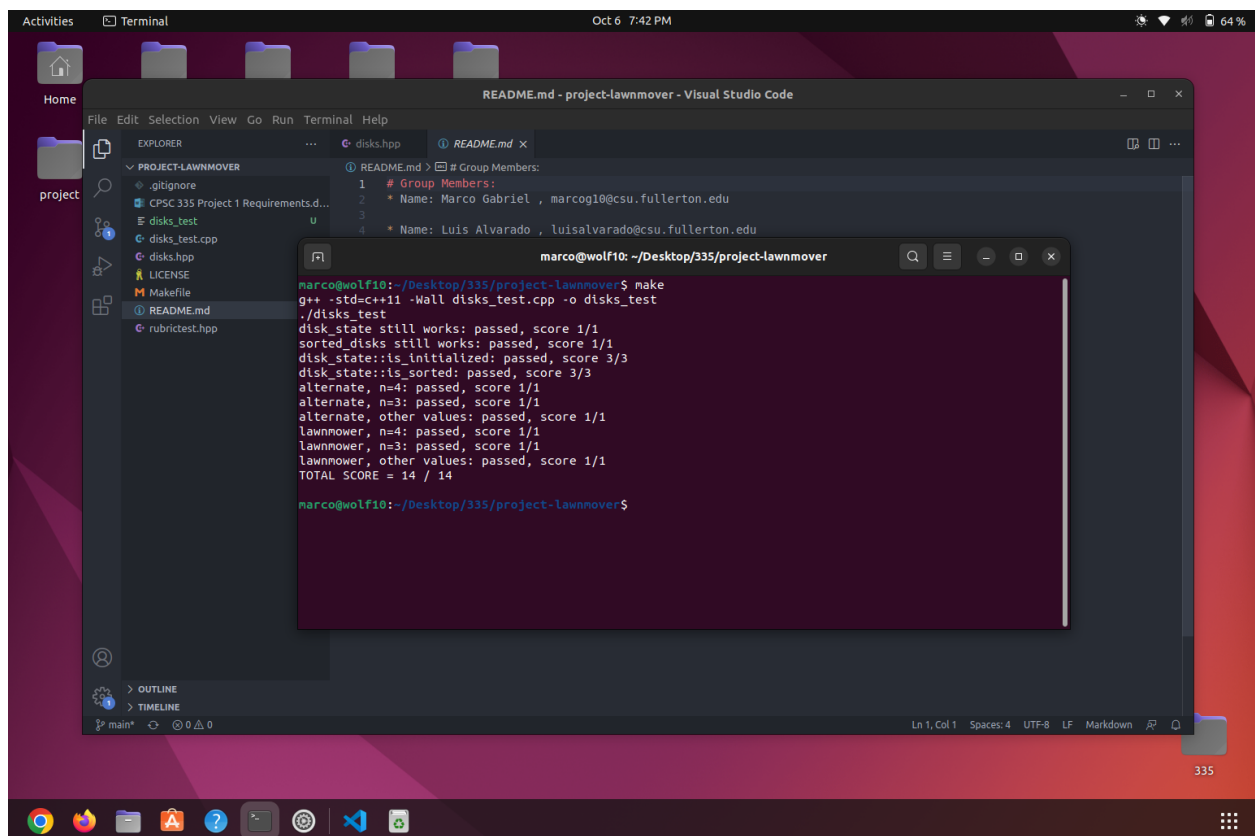
    for (size_t i = 0; i < step.total_count(); i++) { // n + 1 times
        for (size_t j = 0; j < step.total_count() - 1; j++) { // (n-1+1) -> n times
            if (step.get(j) > step.get(j + 1)) { // 2 times
                step.swap(j); // 1 times
                numOfSwap++; // 1 times
            }
        }
    }

    return sorted_disks(disk_state(step), numOfSwap); // 0 times
}
```


SORT_LAWNMOWER FUNCTION

```
sorted_disks sort_lawnmower(const disk_state &before) {  
    //Step Count  
    int numOfSwap = 0; // 1 time unit  
    disk_state step = before; // 1 time unit  
  
    for(size_t i = 0; i < step.light_count(); i++) { // (n+1/2) times  
        // left to right - compares every two adjacent disks and swaps if necessary (i%2 == 0)- meaning it is even  
        for(size_t j = 0; j < step.total_count() - 1; j++) { // n - 1 + 1 -> n times unit  
            if (step.get(j) > step.get(j + 1)) { // 2 time unit  
                step.swap(j); // 1 time unit  
                numOfSwap++; // 1 time unit  
            }  
        }  
    } // else it is odd  
    //right to left - compares every two adjacent disks and swaps if necessary  
    for (size_t k = step.total_count() - 1; k > 0; k--) { // n - 1 + 1 -> n time unit  
        if (step.get(k) < step.get(k - 1)) { // 2 times unit  
            step.swap(k - 1); // 2 times unit  
            numOfSwap++; // 1 times unit  
        }  
    }  
  
    return sorted_disks(disk_state(step), numOfSwap); // 0 times  
}
```

TESTS



The screenshot shows a Linux desktop with a terminal window open. The terminal displays the output of a 'make' command, which runs a series of tests for the 'project-lawnmover' project. The tests include 'disk_state still works', 'sorted_disks still works', 'disk_state::is_initialized', 'disk_state::is_sorted', 'alternate, n=4', 'alternate, n=3', 'lawnmower, n=4', and 'lawnmower, n=3'. Each test is followed by a 'passed' status and a score. The total score is 14 / 14.

```
marco@wolf10: ~/Desktop/335/project-lawnmover  
marco@wolf10:~/Desktop/335/project-lawnmover$ 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  
marco@wolf10:~/Desktop/335/project-lawnmover$
```