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);
```

SC

SCA

SCB

If

Step-Count, Limits Theorem, and Proof by Contradiction

step-count:

S.C = 1 + 1 + SCA → $2+3n^2+3n+0 \Rightarrow 3n^2+3n+2$

SCA = n+1 & SCB → n+1(3n) = $3n^2+3n+0$

SCB = n & IF → n(3) → 3n

SCIF = 2 + max(1, 1)

2 + 1 = 3

	n	+1
3n	$3n^2$	3n
+0	0	0

$3n^2+3n+0$

limits theorem:

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

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

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

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

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

Proof by contradiction

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

f(n) 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++) {                     //(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
}
```

SC
SCA
SCB
SCIF
SCD
SCIF

Step-Count, Limits Theorem, and Proof by Contradiction

step count:

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

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

$SCB = n-1+1 \Rightarrow n(SCIf) \Rightarrow 3n$

$SCIf = 2 + \max(1,1) \Rightarrow 3$

$SCD = n(SCIf) \Rightarrow 4n$

$SCIf = 2 + \max(1,1)$

$= 2 + 2$

$= 4$

$\Rightarrow (\frac{n+1}{2})(\frac{12n}{1})$

$\Rightarrow \frac{12n^2+12}{2}$

$\Rightarrow 6n^2 - 6$

limits theorem:

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

$\lim_{n\to\infty} \frac{3n^2+3n+2}{(n^2)}$

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

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

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

Proof by contradiction

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

$f(n)$ $g(n)$

$c = 3+3+2 \Rightarrow 8$, $n_0 = 1$

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

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

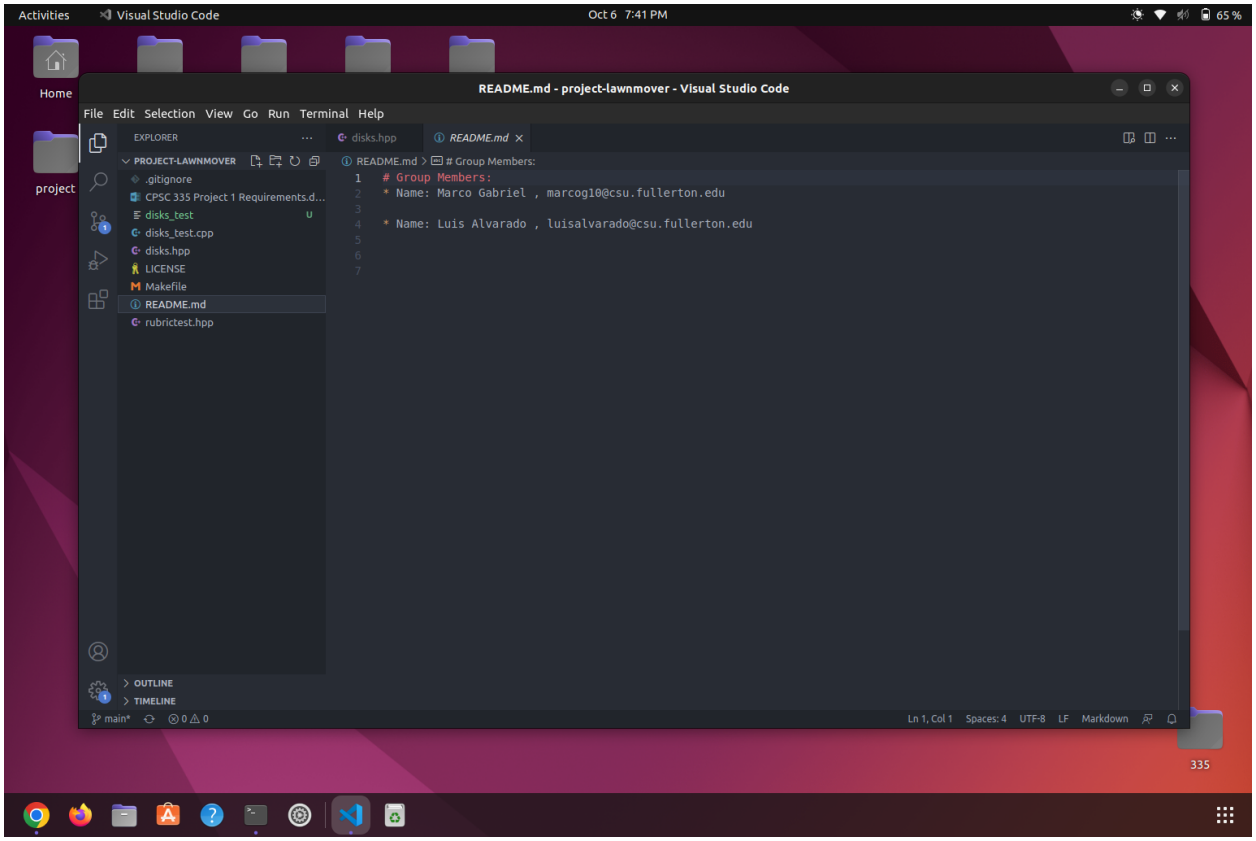
$3+3+2 \leq 8$

$8 \leq 8$

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

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
110           return false;
111         }
112       } else {
113         if (_colors[i] == DISK_LIGHT) {
114           return false;
115         }
116       }
117     }
118     return true;
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