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

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 ligh_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; ensure k doesn't go less than 0; deincrement do
        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;  -> 1+u
  disk_state step = before;  -> 1+u

  for (size_t i = 0; i < step.light_count(); i++) {  -> n-0+1 -> n+1   A
    for (size_t j = 0; j < step.total_count() - 1; j++) {  n-1+1 -> n   B
      if (step.get(j) > step.get(j + 1) ) {  2+u
        step.swap(j); ->                      1+u      If
        numOfSwap++;  ->                      1+u
      }
    }
  }
  return sorted_disks(disk_state(step), numOfSwap);
}
```

Step-Count, Limits Theorem, and Proof by Contradiction

step-count:

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

$SCA = n+1 \cdot SCB \Rightarrow n+1(3n) = 3n^2 + 3n + 0$

$SCB = n \cdot IF \Rightarrow n(3) \Rightarrow 3n$

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

$2 + 1 = 3$

	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

$\underbrace{3n^2 + 3n + 2}_{f(n)} \leq \underbrace{n^2}_{g(n)}$

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

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

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

$3 + 3 + 2 \leq 8$

$8 \leq 8$

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

Lawnmower Algorithm

```
sorted_disks sort_lawnmower(const disk_state &before) {
  int numOfSwap = 0;                                   → 1 time unit      SC
  disk_state step = before;                            → 1 time unit
  for (size_t i = 0; i < step.light_count(); i++) {    → n/2 + 1 time unit   SCA

    // left to right
    for (size_t j = 0; j < step.total_count() - 1; j++) {   → n-1 +1 → n+1   SCB
      if (step.get(j) > step.get(j + 1)) {             → 2 time unit      SBIF
        step.swap(j);                                  → 1 time unit
        numOfSwap++;
      }                                                → 1 time unit
    }

    // right to left
    for (size_t j = step.total_count() - 1; j > 0; j--) {   → n-1+1 → n+1   SCC
      if (step.get(j) < step.get(j - 1)) {             → 2 time unit
        step.swap(j - 1);                              → 2 time unit
        numOfSwap++;                          SCC_IF   → 1 time unit
      }
    }
  }

  return sorted_disks(disk_state(step), numOfSwap);
}
```

Step-Count, Limits Theorem, and Proof by Contradiction

Step count

$SC = 2 + SCA \Rightarrow 2 + 6n^2 + 2n \Rightarrow \frac{6n^2 + 2n + 2}{2} \Rightarrow \boxed{3n^2 + n + 1}$

$SCA = \frac{n}{2} + 1 (SCB)(SCC) \Rightarrow (\frac{n}{2} + 1)(3n)(4n) \Rightarrow (\frac{n}{2} + 1)(12n) \Rightarrow 6n^2 + 12n$

$SCB = n(SBIF) \Rightarrow n(3) \Rightarrow 3n$

$SBIF = 2 + max(1, 1)$
$= 2 + 1$
$= 3$

$SCC = n(SCCIF) \Rightarrow n(4) \Rightarrow 4n$

$SCCIF = 2 + max(2, 1)$
$= 2 + 2$
$= 4$

	$\frac{n}{2}$	1
$12n$	$\frac{12n^2}{2}$	$12n$
0	0	0

Proof by contradiction

$3n^2 + n + 1 \in O(n^2)$

$c = 3 + 1 + 1 = 5, \ n_0 = 1$

$3(1)^2 + 1 + 1 \leq 5(1)^2$

$3 + 1 + 1 \leq 5$

$5 \ \boxed{\leq} \ 5$ ✓

$\boxed{\text{True, by definition, } 3n^2 + n + 1 \in O(n^2)}$

Limits Theorem

$\underbrace{3n^2 + n + 1}_{f(n)} \in \underbrace{O(n^2)}_{g(n)}$

$\lim_{t \to \infty} \frac{3n^2 + n + 1}{n^2}$

$\lim_{t \to \infty} \frac{6n^1 + 1 + 0}{2n^1}$

$\lim_{t \to \infty} \frac{6n + 1}{2n}$

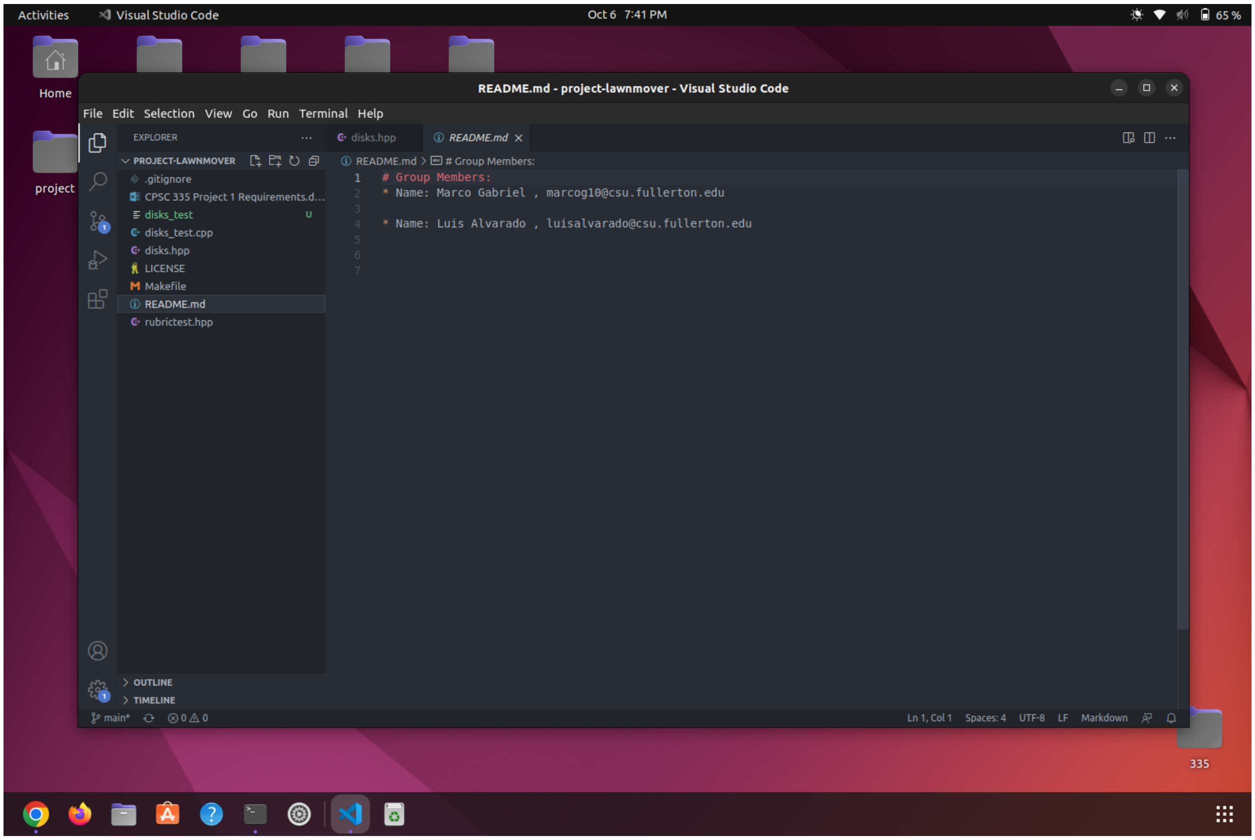
$\lim_{t \to \infty} \frac{6 + 0}{2}$

$\lim_{t \to \infty} \frac{6}{2} = 3$

$\boxed{\text{by limits theorem, we can conclude that } 3n^2 + n + 1 \in O(n^2)}$

Screenshots

README.MD



IS_SORTED FUNCTION:

```
// @Breif: Return true when this disk_state is fully sorted, with all light disks on
// the left (low indices) and all dark disks on the right (high indices).
bool is_sorted() const {
  // loop through the total_count()
  for (size_t i = 0; i < total_count(); i++) {
    if (i < total_count() / 2) {
      if (_colors[i] == DISK_DARK) {

        return false;
      }
    } else {
      if (_colors[i] == DISK_LIGHT) {
        return false;
      }
    }
  }
  return true;
}
```

SORT_ALTERNATE FUNCTION

```
// @Breif:starts with the leftmost disk and proceeds to the right until it reaches the rightmost disk:
// compares every two adjacent disks and swaps them only if necessary.
// It does not iterate through each index, but iterates over each pair (i.e., it moves 2 steps at a time).
// We consider a run to be a check of adjacent disks from left-to-right.
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.light_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);
}
```

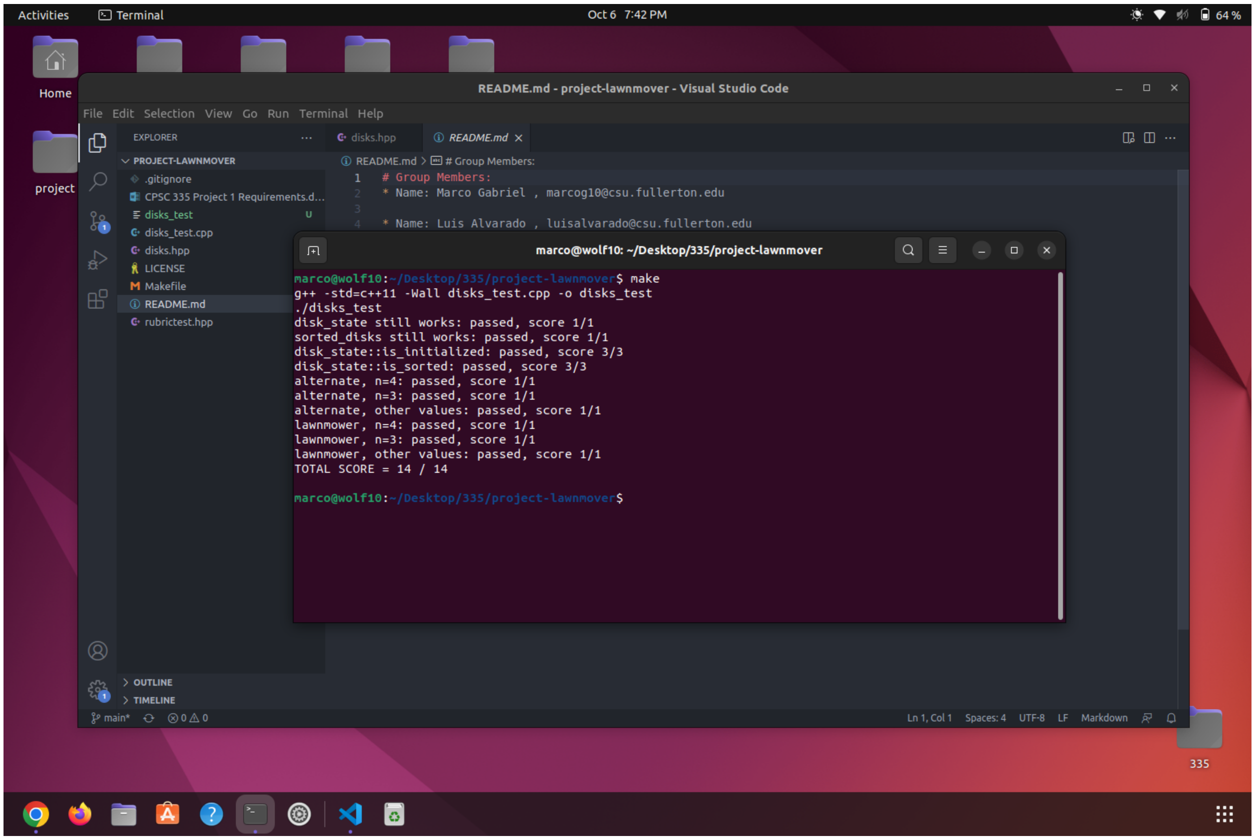
SORT_LAWNMOWER FUNCTION

```
//@Breif: compares every two adjacent disks and swaps them only if necessary.
//Now we have two light disks at the left-hand end and two dark disks at the
//right-hand end. Once it reaches the right-hand end, it starts with the rightmost disk,
//compares every two adjacent disks and proceeds to the left until it reaches the leftmost disk,
// doing the swaps only if necessary. The lawnmower movement is repeated [n/2] times.
sorted_disks sort_lawnmower(const disk_state &before) {
  int numOfSwap = 0;
  disk_state step = before;

    for(size_t i = 0; i < step.light_count(); i++) {
    // left to right - compares every two adjacent disks and swaps if necessary
    for (size_t j = 0; j < step.total_count() - 1; j++) {
      if (step.get(j) > step.get(j + 1)) {
        step.swap(j);
        numOfSwap++;
      }
      //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(disk_state(step), numOfSwap);
}

```

TESTS



README.md - project-lawnmower - Visual Studio Code
marco@wolf10: ~/Desktop/335/project-lawnmower
marco@wolf10:~/Desktop/335/project-lawnmower$ 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-lawnmower$
Group Members:
* Name: Marco Gabriel , marcog10@csu.fullerton.edu
* Name: Luis Alvarado , luisalvarado@csu.fullerton.edu