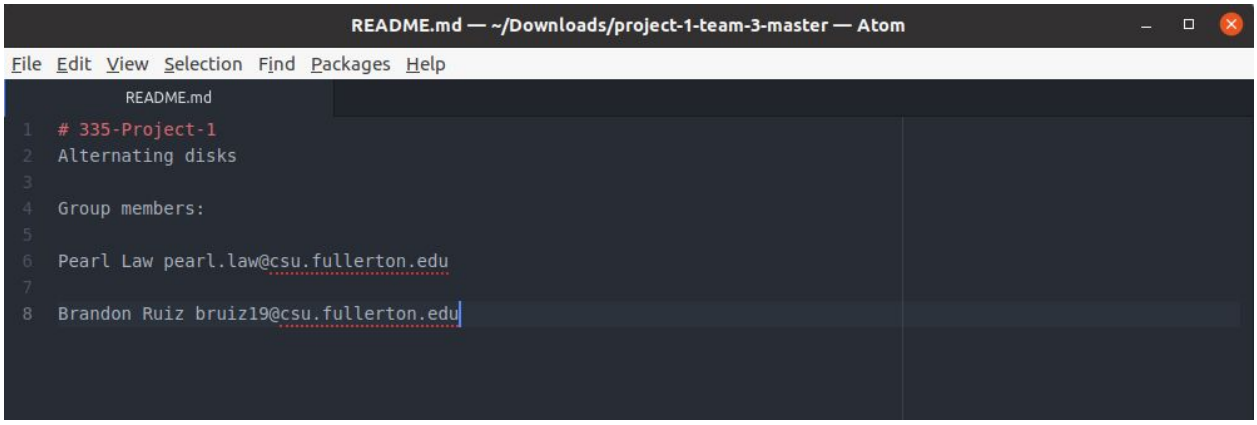
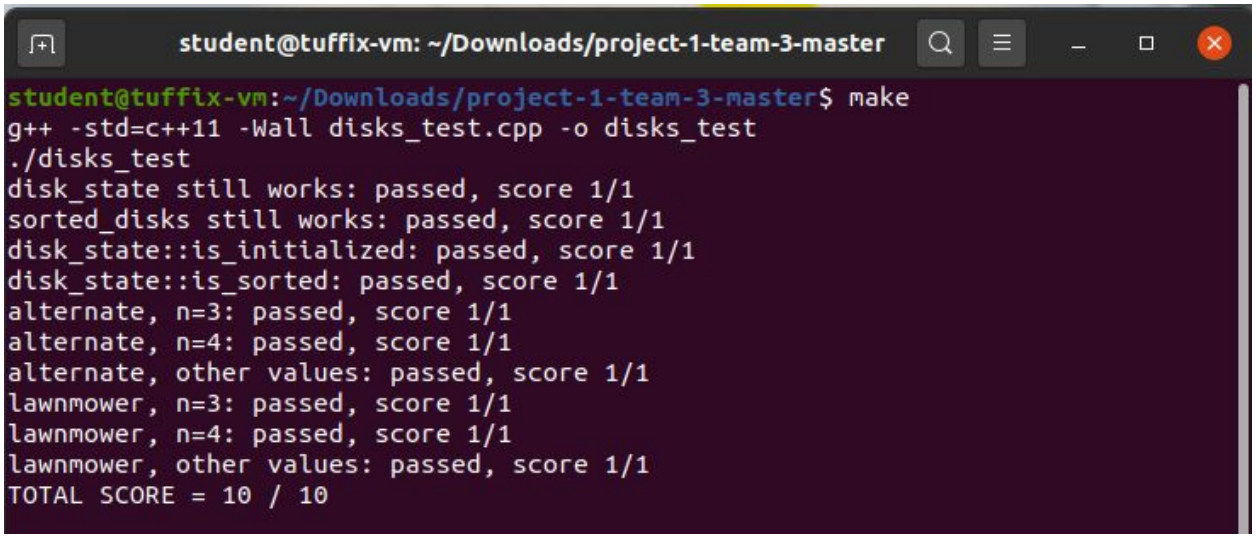
CPSC 335- Algorithm Engineering
Fall 2020
Instructor: Mike Peralta
Project 1: Team 3

Pearl Law (pearl.law@csu.fullerton.edu)
Brandon Ruiz (bruiz19@csu.fullerton.edu)

Tuffix Screenshot of README.md inside Atom editor:



Screenshot of code executing make command:



Problem name: Disk sorter

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 dark, the next n disks are light, and an integer m representing the number of swaps to move the light ones after the dark ones

Lawnmower Algorithm Pseudocode:

```
def lawnmower(integer n, list of disks L):
    swaps = 0

    if n == 0:
        return Nothing

    else:
        for i = 0 to 2n do
            for j = 0 to 2n - 1 do      // left to right movement
                if L[j] > L[j + 1]:     // swap if light disk (L) > dark disk (D)
                   then
                      temp = L[j]
                       L[j] = L[j + 1]
                       L[j + 1] = temp
                       swaps += 1

            for k = 2n - 1 to 1 do      // right to left movement
                if L[k] < L[k - 1]:     // swap if dark disk (D) < light disk (L)
                   then
                      temp = L[k]
                      L[k] = L[k - 1]
                      L[k - 1] = temp
                      swaps += 1

    return sorted disk list and number of swaps needed
```

Alternate Algorithm Pseudocode:

```
define alternate (integer n, list of disks L):
        swaps = 0

        if n = 0;
        return no value (0)

        else
        for i = 0 to n - 1 do
                for j = 0 to n - 1 do
                        if L[j + 1] < L[j]
                                then
                                        temp = L[j]
                                        L[j] = L[j + 1]
                                        L[j + 1] = tmp
                                        swap += L[j + 1]

return disk list after sorted with executed swaps
```

Mathematical Analysis:

*Abbreviations: s.c = step count

1. Lawnmower algorithm:
   a. **Evaluate if statement** (if n == 0) = 1 step
   b. **Evaluate else statement** = 1 step
      1. s.c $_{\text{1st inner for loop}}$ = # loop iterations * s.c loop block
         # loop iterations = ((2n – 1) – 0 / 1) + 1 = 2n

         s.c loop block => evaluate if statement:
         if statement = 1 step
         then block = 4 steps
         else = 0 steps (no else statement to execute)
         s.c. $_{\text{if statement}}$ = 1 + max(4, 0) = 1 + 4 = 5 steps

         s.c $_{\text{1st inner for loop}}$ = 2n * 5 = **10n**

      2. s.c $_{\text{2nd inner for loop}}$ = # loop iterations * s.c loop block
         # loop iterations = ((1 – (2n – 1))/1) + 1 = -2n + 2 + 1 = |-2n + 3| = 2n + 3
         (take absolute value because we cannot have negative iterations/steps)

         s.c loop block => evaluate if statement:
         if statement = 1 step
         then block = 4 steps
         else = 0 steps (no else statement to execute)
         s.c $_{\text{if statement}}$ = 1 + max(4, 0) = 1 + 4 = 5 steps

         s.c $_{\text{2nd inner for loop}}$ = (2n+3) * 5 = **10n + 15**

      3. s.c $_{\text{outer for loop}}$ = else overhead + $\Sigma_{\text{outer for loop}}$ * (s.c $_{\text{1st inner for loop}}$ + s.c $_{\text{2nd inner for loop}}$)

         $= 1 + \sum_{i=0}^{2n} (10n + 10n + 15)$

         $= 1 + \sum_{i=0}^{2n} (20n + 15)$

         $= 1 + \sum_{i=0}^{2n} (20n) + \sum_{i=0}^{2n} (15)$

         $= 1 + 0 + \sum_{i=1}^{2n} (20n) + (15 * (2n + 1))$

         $= 1 + 20 * \sum_{i=1}^{2n} (n) + 30n + 15$

         = 1+ 20 * (n * 2n) + 30n + 15

= 1+ 20 * ($2n^2$) + 30n + 15

s.c $_{\text{outer for loop}}$ = $40n^2 + 30n + 16$

c. **Evaluate function statements-**
initializing variable swaps = 1 step
return statement = 1 step

Entire function s.c = $4 + 40n^2 + 30n + 16 = 40n^2 + 30n + 20$

The step count for the entire algorithm, after dropping the constant, dominated term, and multiplicative constant, is n^2. This corresponds to the time complexity efficiency class $O(n^2)$.

2. Alternate algorithm:
   d. **Evaluate if statement** (if n == 0) = 1 step
   e. **Evaluate else statement** = 1 step

   01. Step count (1st for loop): number of loop executed * step count block
   number of loop executed: [((n-1)-0) / 1] + 1 = n

   step count block: evaluating the if statement
   if statement = 1 step
   step count block = 0 steps (does not execute anything)
   else = 0 steps (does not follow up on execution)
   step count (1st loop): 1 + max(0, 0) = 1 + 0 = 1 step

   Step count (1st for loop): n * 1 = **n**

   02. Step count (2nd for loop): number of loop executed * step count block
   number of loop executed: [((n-1)-0) / 1] + 1 = n

   step count block: evaluating the if statement
   if statement = 1 step
   step count block = 3 steps
   else = 0 steps (does not follow up on execution)
   step count (2nd loop): 1 + max(3, 0) = 1 + 3 = 4 steps

   Step count (2nd for loop): n * 4 = **4n**

   03. Step count (outside for loop): outside else + Σ (outside for loop) * (1st for loop + 2nd for loop)
   = 1 + $\sum_{i=0}^{n} (n + 4n)$

$= 1 + \sum_{i=0}^{n} (5n)$

$= 1 + 0 + 5 * \sum_{i=1}^{n} (n)$

$= 1 + 0 + 5 * (n * n)$

Step count (outside for loop) = $5n^2 + 1$

f. **Evaluate whole algorithm:** $5n^2 + 1 + 3 = 5n^2 + 4$
The step count for the whole algorithm after removing the values unaffected by n is n^2. This falls under the category of time complexity efficiency of the class: $O(n^2)$.