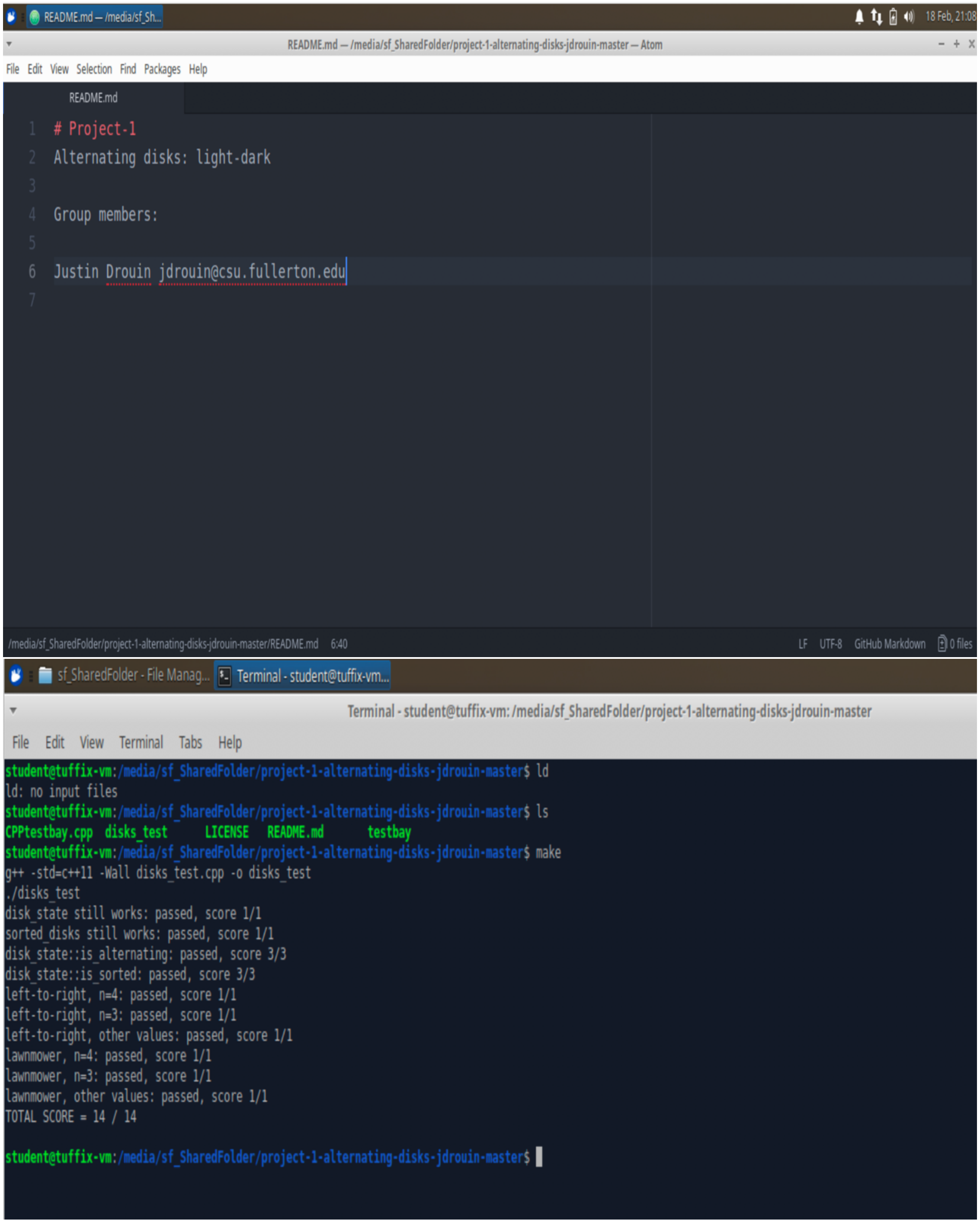
Project 1

Justin Drouin
jdrouin@csu.fullerton.edu



The Alternating Disk Problem

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 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.

With the assumption of DARK DISKS = LIGHT DISKS and the first disk is a LIGHT DISK, the last disk will always be a DARK DISK, thus the outer loop only needs to loop while I < the total of DARK_COUNT for the left-to-right algorithm. For the Lawnmower algorithm, assuming the same, the outer loop count only needs to loop while I < the total number of DARK_COUNT/ 2. If DARK_COUNT != LIGHT_COUNT, then the DARK_COUNT would be substituted with total DISK_COUNT

The left-to-right algorithm

Void lefttoright():

```
For i = 0;  i < DARK_COUNT; i++{
   For  j = 0; j < DISK_COUNT-1; j++{
     If [j] == DISK_DARK {
       If [j+1] == DISK_LIGHT {
         swap([j] & [j+1]);
       }
     }
   }
 }
```

The lawnmower algorithm

Void lawnmower():

```
For i = 0;  i < DARK_COUNT/2 ;  i++ {
     For j = 0;  j < DISK_COUNT -1 ;  j++ {
       If [j] == DISK_DARK {
         If [j+1] == DISK_LIGHT {
           swap([j] & [j+1])
         }
       }
     }
     For j = DISK_COUNT-1; j > 0;  j-- {
       If [j] == DISK_LIGHT {
         If [j-1] == DISK_DARK {
           swap([j-1] & [j])
         }
       }
     }
}
```

The left-to-right algorithm

Void lefttoright():

```
4  For i = 0;  i < DARK_COUNT; i++{           n-1+1 = n
      For  j = 0; j < DISK_COUNT-1; j++{       n-1-1+1 = n-1
        3 If [j] == DISK_DARK {          index access 1tu + comparison 1tu = 2tu
         2   If [j+1] == DISK_LIGHT { index access & j+1 (j=j+1) 3tu + comparison = 4 tu
          1    swap([j] & [j+1]); std::swap() 3tu + index access 1tu + index access & j+1 (j=j+1)3tu = 7tu
             }
          }
      }
  }
```

1. 7 tu
2. 4 + max(7,0) = 11 tu
3. 2 + max(11,0) = 13 tu
4. 13(n-1)(n) = (13n – 13) (n) = 13n^2 – 13n (indicating O(n^2) efficiency)

Proving Efficiency Class With Limits: Assuming O(n^2) Efficiency

Lim 13n^2 – 13n / n^2 = 13 – (13/n)

Lim 13 = 13

Lim 13/n = 0 = 13 – 0 = 13

13 is non-negative and constant with respect to n^2. <u>Therefore 13n^2 – 12n ∈ O(n^2)</u>

The lawnmower algorithm

Void lawnmower():

```
9  For i = 0;  i < DARK_COUNT/2 ;  i++ { 13n + 13 + 13n – 13 = 26n(n/2) = 26n^2/2 = 13n^2
   8   For j = 0;  j < DISK_COUNT -1 ;  j++ { 13(n-1) = 13n - 13
     7   If get(j) == DISK_DARK { index access 1tu + comparison 1tu = 2tu
       6   If get(j+1) == DISK_LIGHT { index access & j+1 (j=j+1) 3tu + comparison = 4 tu
         5 swap(get(j) & get(j+1)) std::swap() 3tu + index access 1tu + index access & j+1 (j=j+1)3tu = 7tu
         }
       }
     }
   4   For j = DISK_COUNT-1; j > 0;  j-- { 13(n+1) = 13n + 13
     3   If get(j) == DISK_LIGHT { index access 1tu + comparison 1tu = 2tu
       2   If get(j-1) == DISK_DARK { index access & j-1 (j=j-1) 3tu + comparison = 4 tu
         1   swap(get(j-1) & get(j)) std::swap() 3tu + index access 1tu + index access & j-1 (j=j-1)3tu = 7tu
         }
       }
     }
   }
```

1. 7 tu
2. 4 + max(7,0) = 11 tu
3. 2 + max(11,0) = 13 tu
4. 13(n+1) = 13n + 13
5. 7 tu
6. 4 + max(7,0) = 11 tu
7. 2 + max(11,0) = 13 tu
8. 13(n-1) = 13n – 13
9. 13n + 13 + 13n – 13 = 26n(n/2) = 26n^2/2 = 13n^2

Proving Efficiency Class With Limits: Assuming O(n^2) Efficiency

Lim 13n^2 / n^2 = 13

13 is non-negative and constant with respect to n^2. <u>Therefore 13n^2 ∈ O(n^2)</u>

for i = 0 ; i < n ; i ++

for j = 0 ; j < n - 1 ; j ++

if get(j) == D

if get (j+1) == W

7 — swap(get(j) + get(j+1))

4 + max (7,0) = 11 + U

2 + max (11,0) = 13 + U

n - 1 - 1 + 1 = n - 1

n - 1 + 1 = n

13 (n-1)(n)

(13n - 13) (n)

$13n^2 - 13n$

$\lim \frac{13n^2 - 13n}{n^2} = 13 + \frac{13}{n}$

$\lim \frac{13n}{n} = 13 \quad - \lim \frac{9}{n^2} = 0$

13

---

for j = 0 ; j < n - 1 ;

if get(j) ==

$\frac{n}{2} - 1 + 1 = n/2$

$13n^2 \in n^2$

for i = 0 ; i < n/2 ; i ++

for j = 0 ; j < n - 1 ; j ++

13(n-1)

13n - 13

if get(j) == B

if get (j+1) == W

11

7 Swap [(j)] and [j+1]

$\frac{13}{11}$

$\lim \frac{13n^2}{n^2}$

for j = n - 1 ; j > 0 ; j --

n - 1 + 1 + 1

n + 1

if get (j) == W

13(n+1)

13n + 13

if get(j-1) = B

11

7 - Swap [(j-1) and (j)]

4 + max (7,0) = 11

2 + max (11,0) = 13

(13n + 13)(13n - 13) ($\frac{n}{2}$) $\quad 26n$

((13n - 13)(13n + 13)

$26n \left(\frac{n}{2}\right)$

$\frac{n}{2} (26n)$

$\frac{26n^2}{2}$

$13n^2$