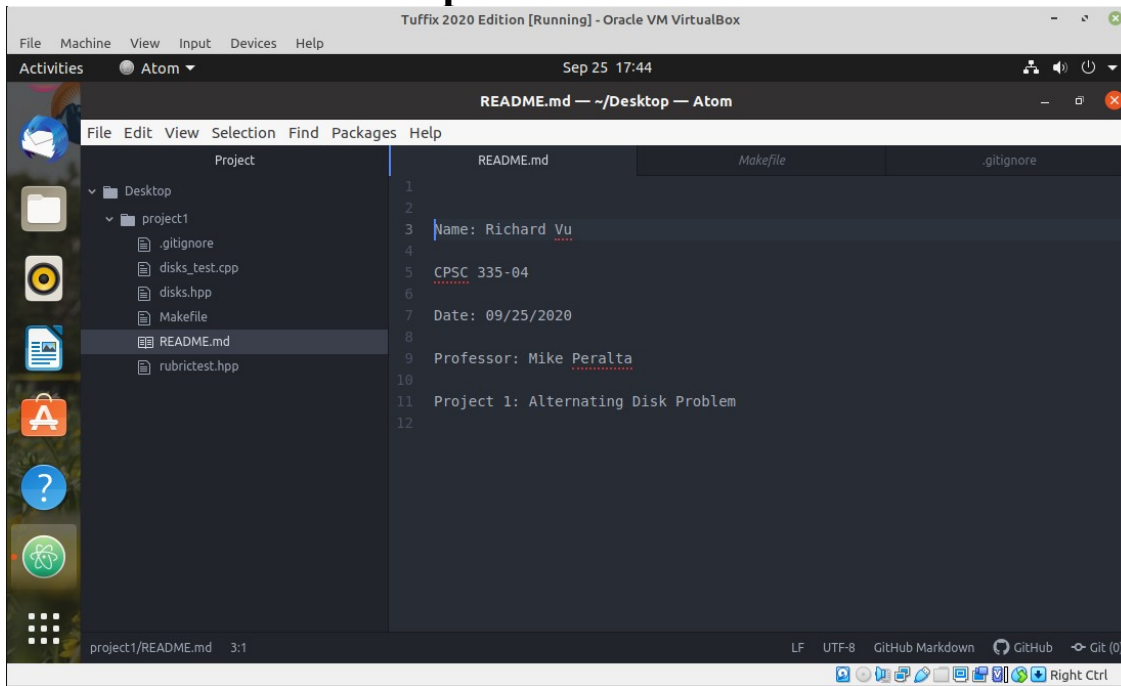


Richard Vu
CPSC 335
Project 1
09/25/2020

Programming Assignment 1

First Screenshot: Group member names in Atom in Tuffix



Second Screenshot: code executing with the command make

```
student@tuffix-vm: ~/Desktop/project1
command 'cake' from deb cakephp-scripts (2.10.11-2)
command 'fake' from deb fake (1.1.11-3)
command 'jake' from deb node-jake (0.7.9-1)
command 'rake' from deb rake (13.0.1-4)

See 'snap info <snapname>' for additional versions.

student@tuffix-vm:~/Desktop/project1$ cd ~
student@tuffix-vm:~$ cd Desktop
student@tuffix-vm:~/Desktop$ ls
project1
student@tuffix-vm:~/Desktop$ cd project1
student@tuffix-vm:~/Desktop/project1$ ls
disks.hpp disks_test.cpp Makefile README.md rubrictest.hpp
student@tuffix-vm:~/Desktop/project1$ 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:
TEST FAILED:
line 77 of file disks_test.cpp, message: is_initialized() for n=1
score 0/1
disk_state::is_sorted:
TEST FAILED:
line 87 of file disks_test.cpp, message: is_sorted() after swap
score 0/1
disks_test: disks.hpp:64: disk_color disk_state::get(size_t) const: Assertion `!
s_index(index)' failed.
make: *** [Makefile:12: run_test] Aborted (core dumped)
student@tuffix-vm:~/Desktop/project1$
```

Lawnmower Algorithm:

Lawnmower Algorithm
Input: positive integer N , A list of $2 * N$ disks of alternating colors light-dark, starting with light.
Output: A list of $2*N$ disks, where the first n disks are dark, the next n disks are light, and an integer m representing total number of <i>swaps</i> to move the light disks after the dark disks.

Lawnmower Pseudo-code:

def sort_lawnmower(disks):

```
    if len(disks) == 0:
        return 0
    endif
    swaps = 0
    for k = 0 to 2*n do
        for j = 0 to 2*n - 1 do
            if disks[j] greater than disks[j + 1] do
                swap(disks[j], disks[j + 1])
                swaps += 1
            endif
        endfor
        for m = 2 * n - 1 to 1 do
            if disks[i] is less than disks[i - 1] do
                swaps(disk[i], disks[i - 1])
                swaps += 1
            endif
        endfor
    endfor
    return (disks, swaps)
```

Big Oh $O(n)$ mathematical analysis for the Lawnmower Algorithm

```
if len(disks) == 0: +1
    return 0 + 1
swaps = 0 → +1
for k = 0 to 2*n do    +1 →  $[(2n - 0) / 1] + 1 = 2n + 1$ 

    for j = 0 to 2*n - 1 do +1 →  $[(2n - 1 - 0) / 1] + 1 = 2n$ 
        if disks[j] greater than disks[j + 1] do +1
            swap(disks[j], disks[j + 1])          +1
                                                     $\max(1, 4) = 4$ 
            swaps += 1                             +1
        endif
    endfor

for m = 2 * n - 1 to 1 do +1 →  $\text{abs}([1 - (2n - 1) / 1] + 1) = 2n + 3$ 

    if disks[i] is less than disks[i - 1] do +1
        swaps(disk[i], disks[i - 1])          +1
        swaps += 1                             +1
                                                     $\max(1, 4) = 4$ 
    endif
endfor
endfor
return (disks, swaps) → + 1
```

Mathematical Analysis and efficiency class for Lawnmower Algorithm.

$$\sum_{i=0}^{2n} 1(2n+1) \sum_{j=0}^{2n-1} 4(2n) + \sum_{k=2n-1}^1 4|-2n+3|$$

$$(2n+1)[8n+8n+12]$$

$$32n^2+40n+12+4 \quad \leftarrow \begin{array}{l} +2 \text{ initial if} \\ +1 \text{ swaps} = 0 \\ +1 \text{ return} \\ \text{statement} \end{array}$$

$$32n^2+40n+16$$

$$32n^2+40n+16 \in O(n^2) (\text{trivial case})$$

$$32n^2 = O(n^2) (\text{dominated term})$$

$$O(n^2) (\text{constant factor})$$

Alternating Algorithm:

Alternating Algorithm
Input: positive integer N , A list of $2 * N$ disks of alternating colors light-dark, starting with light.
Output: A list of $2*N$ disks, where the first n disks are dark, the next n disks are light, and an integer m representing total number of <i>swaps</i> to move the light disks after the dark disks.

Alternating Pseudo-code:

def sort_alterate(disks):

 if len(disks) == 0:

 return 0

 endif

 swaps = 0

 for j = 0 to $2*n$ do + 1 $\rightarrow [(2n - 0) / 1] + 1 = 2n + 1$

 for k = 0 to $2*n - 1$ do

 if disks[i] is greater than disk[j+1] do

 swap(disks[i], disks[j+1])

 swaps += 1

 endif

 endfor

 return (disks, swaps)

Mathematical Analysis of the Alternating Algorithm:

```

if len(disks) == 0:    +1
    return 0          + 1
endif
swaps = 0              + 1

for j = 0 to 2*n do    + 1 → [(2n - 0) / 1] + 1 = 2n + 1
    for k = 0 to 2*n - 1 do + 1 → [(2n - 1 - 0) / 1] + 1 = 2n
        if disks[i] is greater than disk[j+1] do + 1
            swap(disks[i], disks[j+1])          +1
            swaps += 1                           +1
                                                max(1, 3) = 3 + 1 = 4
        endif
    endfor
endfor

return (disks, swaps) + 1

```

$$\begin{aligned}
 & \sum_{0}^{2n+1} 1 \sum_{0}^{2n} 4 \\
 & 1(2n+1)(4*2n) + 4 \\
 & (16n^2 + 8n) + 4 \\
 & 16n^2 + 8n + 4
 \end{aligned}$$

$$4n^2 + 10n + 8 \in O(n^2) \text{ (trivial)}$$

$$4n^2 = O(n^2) \text{ (dominated term)}$$

$$n^2 = O(n^2) \text{ (dropped constant)}$$