

PROJECT REPORT

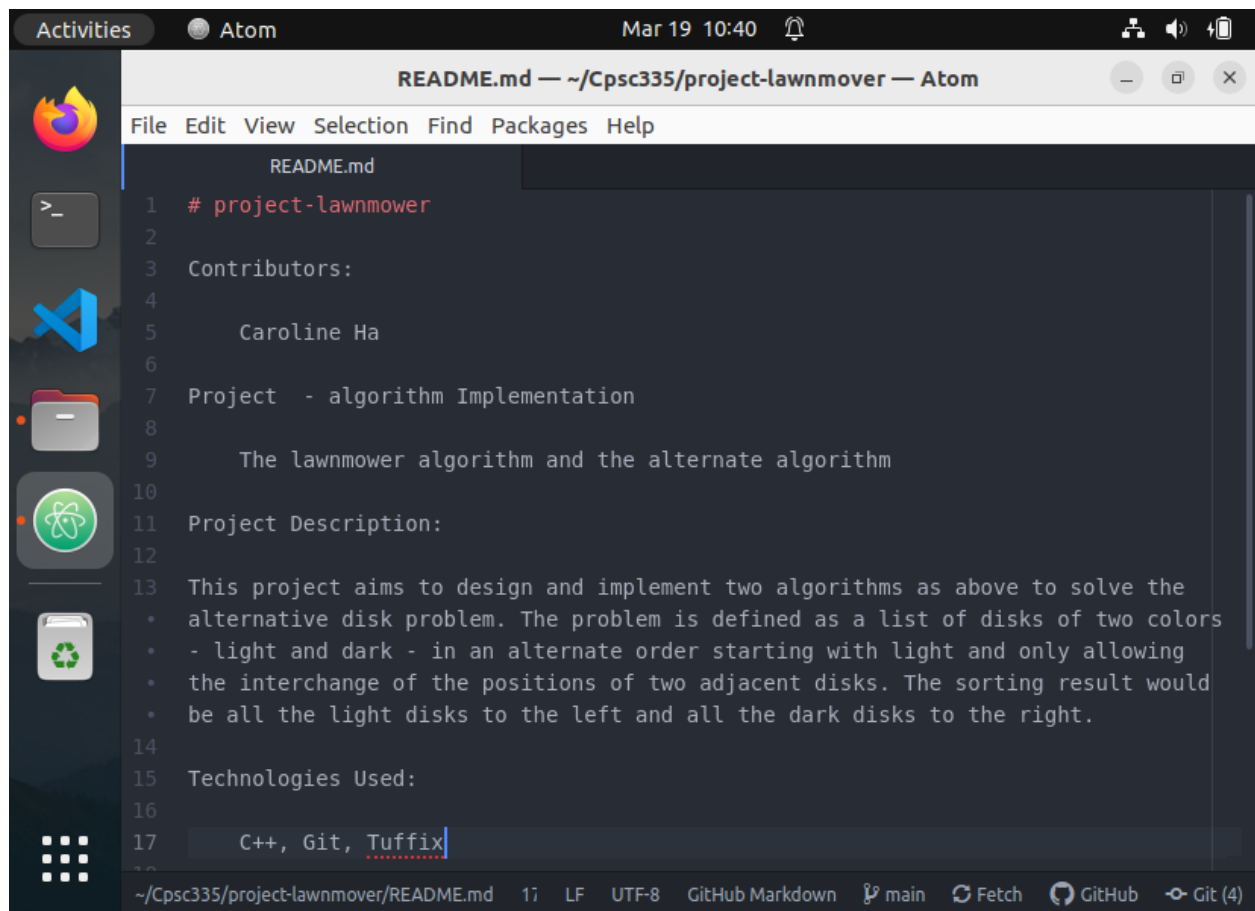
Project 1: implementing algorithms

Caroline Ha

Email: carolineh@csu.fullerton.edu

Project Description

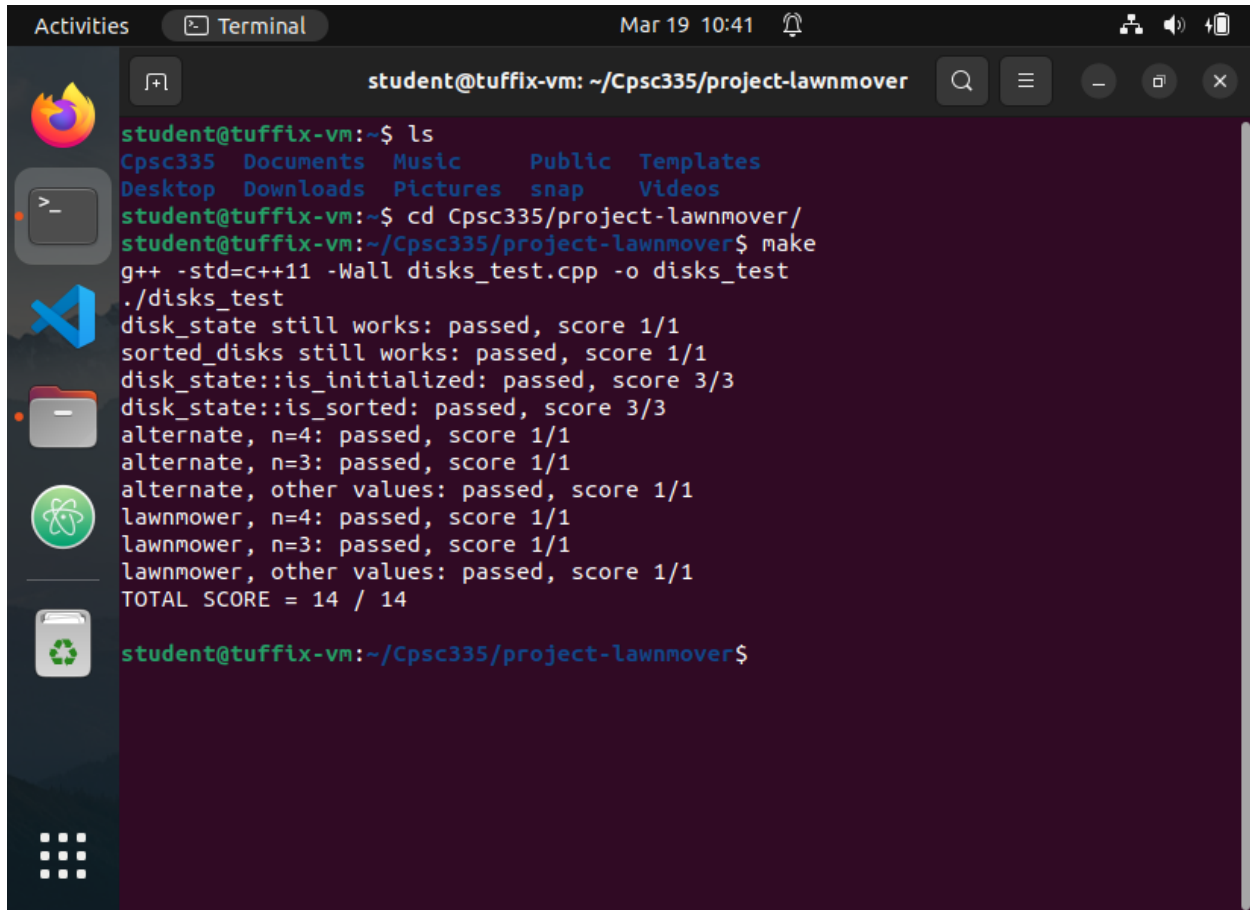
- Screenshot- README.md



```
1 # project-lawnmower
2
3 Contributors:
4
5     Caroline Ha
6
7 Project - algorithm Implementation
8
9     The lawnmower algorithm and the alternate algorithm
10
11 Project Description:
12
13 This project aims to design and implement two algorithms as above to solve the
14 alternative disk problem. The problem is defined as a list of disks of two colors
15 • - light and dark - in an alternate order starting with light and only allowing
16 • the interchange of the positions of two adjacent disks. The sorting result would
17 • be all the light disks to the left and all the dark disks to the right.
18
19 Technologies Used:
20
21     C++, Git, Tuffix
```

Project Compiling and Executing

- Screenshot- compiling and executing



The screenshot shows a terminal window titled "student@tuffix-vm: ~/Cpsc335/project-lawnmover". The terminal output is as follows:

```
student@tuffix-vm:~$ ls
Cpsc335  Documents  Music      Public  Templates
Desktop  Downloads  Pictures  snap    Videos
student@tuffix-vm:~$ cd Cpsc335/project-lawnmover/
student@tuffix-vm:~/Cpsc335/project-lawnmover$ 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
student@tuffix-vm:~/Cpsc335/project-lawnmover$
```

Algorithm Design & Mathematical analysis :Input Size 2n

1	function lawnmowerSort(arr of alternative disks):	Complexity
2	// initialize variables	1 tu
3	swapCount = 0	1 tu
4	isSorted = False	1 tu
5	traversingStartIndex = 0	3 tu
6	traversingEndIndex = length(arr of alternative disks) - 1	(n+1)/2 times
7	while !isSorted do	1 tu
8	isSorted = True	2n times
9	// iterate from left to right	1 tu
10	for i from traversingStartIndex to traversingEndIndex	1 tu
11	if arr[i] > arr[i+1] do	1 tu
12	swap(arr[i], arr[i+1])	1 tu
13	isSorted = False	1 tu
14	swapCount += 1	Step is 1
15	++i	1 tu
16	// if no swaps were made, the arr is sorted	1 tu
17	if isSorted do	1 tu
18	exit the Loop	2n times
19	// iterate from right to left	1 tu
20	for i from traversingEndIndex to traversingStartIndex	1 tu
21	if arr[i-1] > arr[i] do	1 tu
22	swap(arr[i-1], arr[i])	1 tu
23	isSorted = False	1 tu
24	swapCount += 1	1 tu
25	--i	Step is 1
26	++traversingStartIndex	1 tu
27	--traversingEndIndex	1 tu
28	return [arr of sorted disks, swapCount]	0 tu

Step Count

$$SC_{\text{while}} = 1 + 2n \cdot SC_{\text{firstfor}} + SC_{\text{if}} + 2n \cdot SC_{\text{secondfor}} + 1 + 1$$

$$= 1 + 2n \cdot (1 + (\text{Max}(3,0)) + (1 + \text{Max}(1,0)) + 2n \cdot (1 + \text{Max}(3,0)) + 1 + 1$$

$$= 16n + 5$$

$$SC = 1 + 1 + 3 + \frac{n+1}{2} \cdot SC_{\text{while}} = 5 + \frac{n+1}{2} \cdot (16n+5) = 8n^2 + 21n + \frac{2}{5}$$

Time complexity is $O(n^2)$.

$$8n^2 + 21n + \frac{2}{5} \in O(n^2)$$

Proof by limitation:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{8n^2 + 21n + \frac{2}{5}}{n^2} = \lim_{n \rightarrow \infty} 8 + \lim_{n \rightarrow \infty} \frac{21}{n} + \lim_{n \rightarrow \infty} \frac{2}{5n^2} = 8 + 0 + 0 = 8$$

$$\text{By limitation, } 8n^2 + 21n + \frac{2}{5} \in O(n^2)$$

Algorithm Design & Mathematical analysis :Input Size 2n

1- function alternateSort(arr of alternative disks):	Complexity
2 // initialize variables	
3 swapCount = 0	1 tu
4 isSorted = False	1 tu
5 traversingEndIndex = length(arr of alternative disks) - 1	3 tu
6 while isSorted == True do	n/2 + 1 times
7 // iterate odd-indexed elements	
8 for i from 1 to traversingEndIndex do	n times
9 if arr[i] > arr[i+1] do	1 tu
10 swap(arr[i], arr[i+1])	1 tu
11 isSorted = False	1 tu
12 swapCount += 1	1 tu
13 i += 2	Step is 2
14 // iterate even-indexed elements, include index 0	
15 for i from 0 to traversingEndIndex do	n + 1/2 times
16 if arr[i] > arr[i+1] do	1 tu
17 swap(arr[i], arr[i+1])	1 tu
18 isSorted = False	1 tu
19 swapCount += 1	1 tu
20 i += 2	Step is 2
21 return [arr of sorted disks, swapCount]	0 tu

Step Count

First For iteration step is 2: the iteration times are = $\frac{(2n-1)-1}{2} + 1 = n$

Second For iteration step is 2: the iteration times are $\frac{(2n-1)-0}{2} + 1 = n + \frac{1}{2}$

$$SC_{\text{while}} = n * SC_{\text{firstfor}} + (n + \frac{1}{2}) * SC_{\text{secondfor}} = n * (1 + \text{Max}(3, 0)) + (n + \frac{1}{2}) * (1 + \text{Max}(3, 0))$$

$$= 8n + 2$$

$$SC = 1 + 1 + 3 + (\frac{n}{2} + 1) * SC_{\text{while}} = 5 + (\frac{n}{2} + 1) * (8n + 2) = 4n^2 + 9n + 7$$

Time complexity is $O(n^2)$.

Proof by limitation:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{4n^2 + 9n + 7}{n^2} = \lim_{n \rightarrow \infty} 4 + \lim_{n \rightarrow \infty} \frac{9}{n} + \lim_{n \rightarrow \infty} \frac{7}{n^2} = 4 + 0 + 0 = 4$$

By limitation, $4n^2 + 9n + 7 \in O(n^2)$