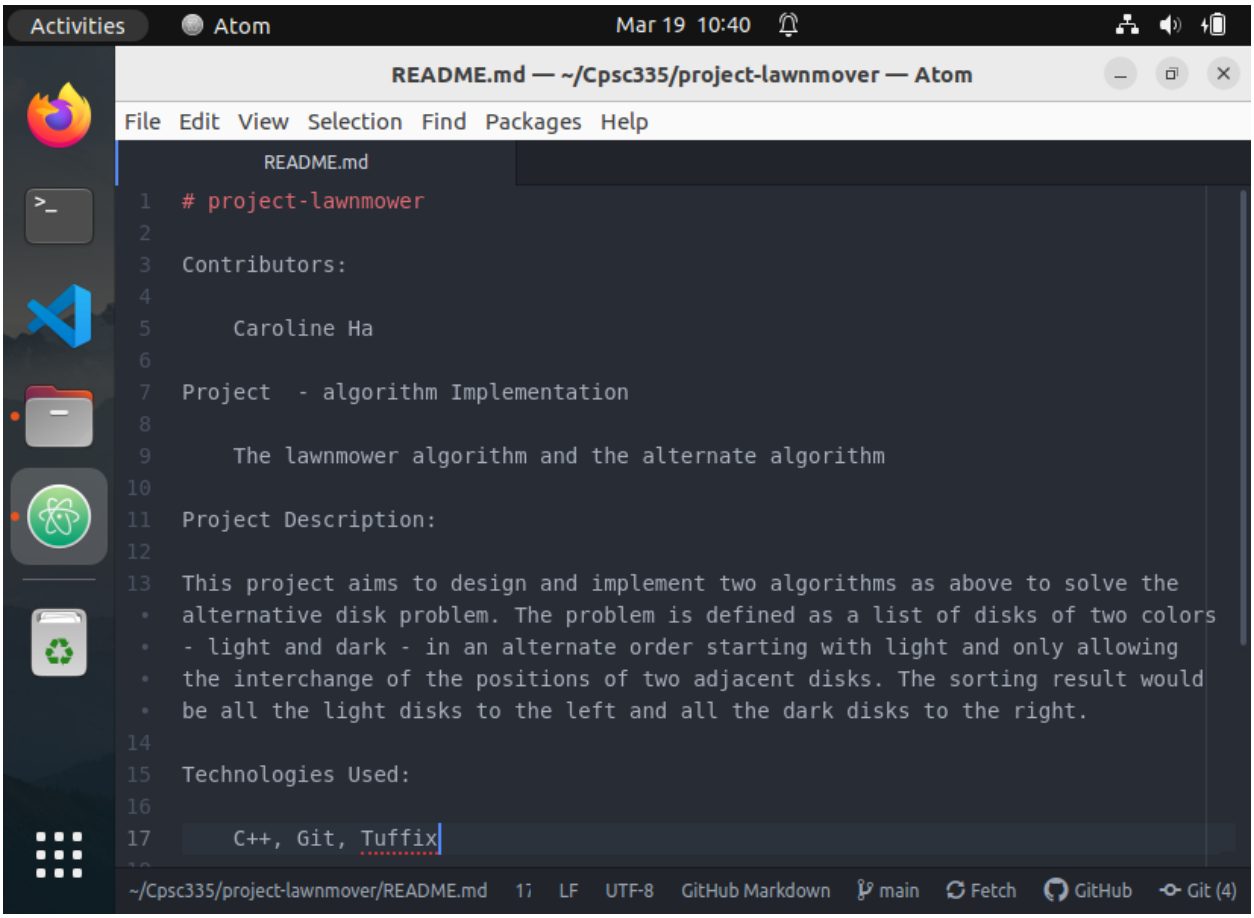
PROJECT REPORT

Project 1: implementing algorithms	Caroline Ha Email: carolineh@csu.fullerton.edu

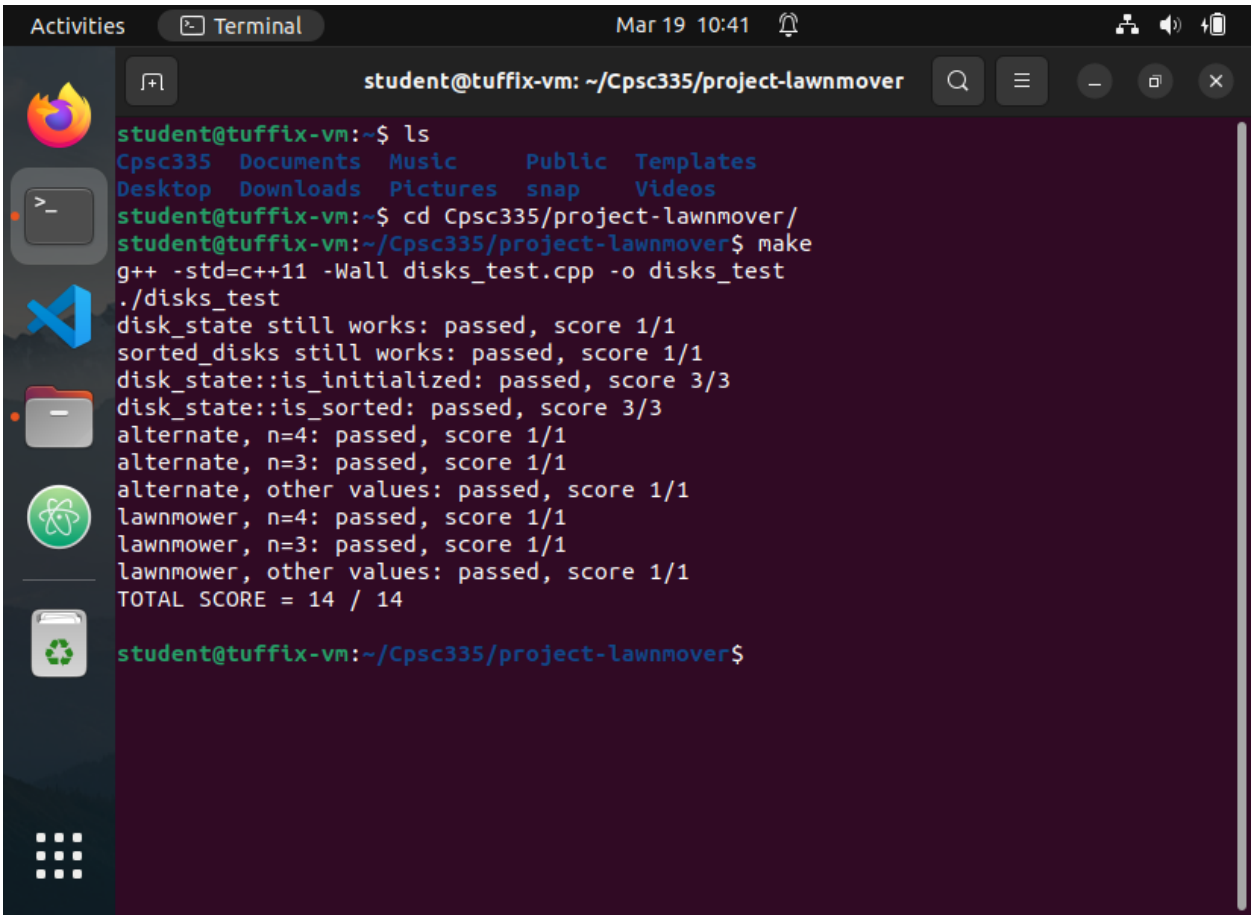
Project Description

- Screenshot- README.md



Project Compiling and Executing

- Screenshot- compiling and executing



Algorithm Design & Mathematical analysis :Input Size 2n

```
function lawnmowerSort(arr of alternative disks):
    // initialize variables
    swapCount = 0
    isSorted = False
    traversingStartIndex = 0
    traversingEndIndex = length(arr of alternative disks) - 1
    while !isSorted do
        isSorted = True
        // iterate from left to right
        for i from traversingStartIndex to traversingEndIndex
            if arr[i] > arr[i+1] do
                swap(arr[i], arr[i+1])
                isSorted = False
                swapCount += 1
            ++i
        // if no swaps were made, the arr is sorted
        if isSorted do
            exit the Loop
        // iterate from right to left
        for i from traversingEndIndex to traversingStartIndex
            if arr[i-1] > arr[i] do
                swap(arr[i-1], arr[i])
                isSorted = False
                swapCount += 1
            --i
        ++traversingStartIndex
        --traversingEndIndex
    return (arr of sorted disks, swapCount)
```

Complexity
1 tu
1 tu
1 tu
3 tu
(n+1)/2 times
1 tu
2n times
1 tu
1 tu
1 tu
1 tu
Step is 1
1 tu
1 tu
2n times
1 tu
1 tu
1 tu
1 tu
Step is 1
1 tu
1 tu
0 tu

Step Count

$SC_{while} = 1 + 2n*SC_{firstfor} + SC_{if} + 2n*SC_{secondfor} + 1 + 1$

$= 1 + 2n*(1+ (Max(3,0)) + (1 + Max(1,0)) + 2n*(1+Max(3,0)) + 1 + 1$
$= 16n + 5$

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

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

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

Proof by limitation:

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

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

Algorithm Design & Mathematical analysis :Input Size 2n

```
function alternateSort(arr of alternative disks):
    // initialize variables
    swapCount = 0
    isSorted = False
    traversingEndIndex = length(arr of alternative disks) - 1
    while isSorted == True do
        // iterate odd-indexed elements
        for i from 1 to traversingEndIndex do
            if arr[i] > arr[i+1] do
                swap(arr[i], arr[i+1])
                isSorted = False
                swapCount += 1
            i += 2
        // iterate even-indexed elements, include index 0
        for i from 0 to traversingEndIndex do
            if arr[i] > arr[i+1] do
                swap(arr[i], arr[i+1])
                isSorted = False
                swapCount += 1
            i += 2
    return (arr of sorted disks, swapCount)
```

Complexity
1 tu
1 tu
3 tu
n/2 + 1 times
n times
1 tu
1 tu
1 tu
1 tu
Step is 2
n + 1/2 times
1 tu
1 tu
1 tu
1 tu
Step is 2
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 $\frac{(2n-1)-0}{2} + 1 = n + \frac{1}{2}$

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

$= 8n + 2$

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

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

Proof by limitation:

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

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