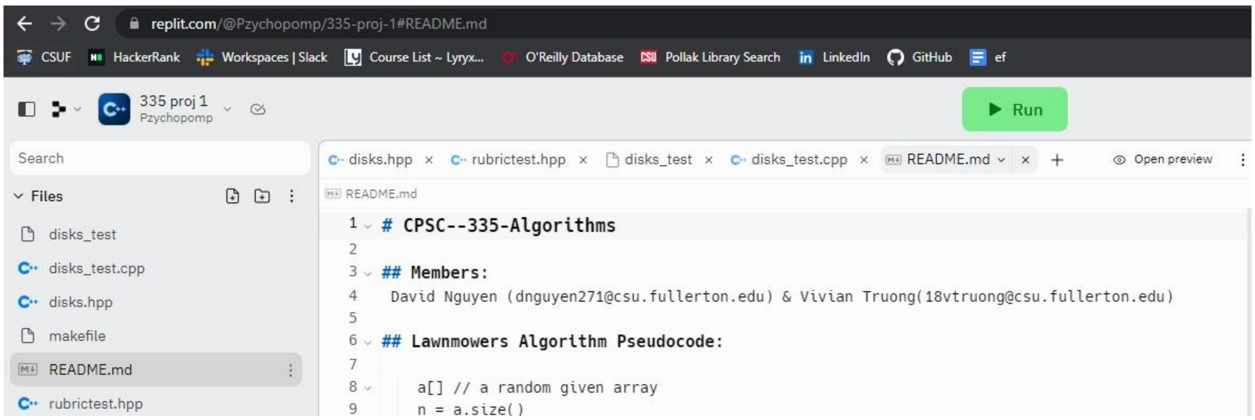
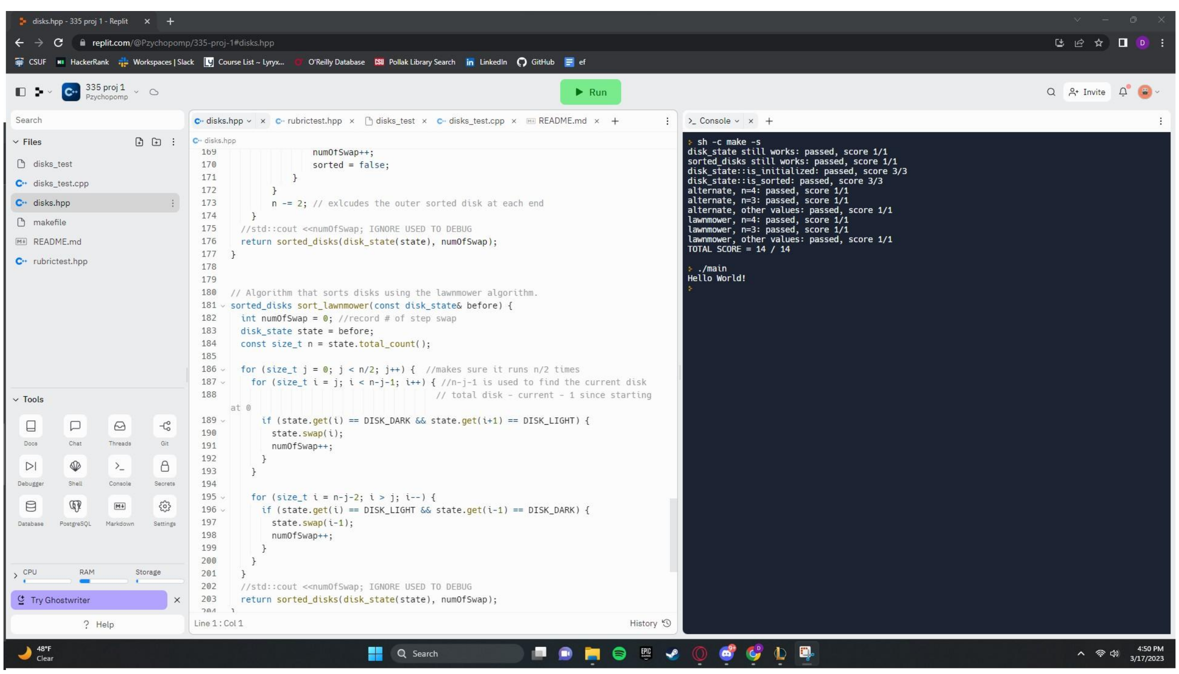
David Nguyen (dnguyen271@csu.fullerton.edu)
Vivian Truong(18vtruong@csu.fullerton.edu)
Github: https://github.com/Pzychopomp/CPSC-335-Algorithms-Proj-1

CPSC 335 Project 1 Submission PDF

2. The following is a screenshot inside Replit, the editor used for this project:



3. Screenshot of execution:



4. Step count and efficiency

Lawnmowers Algorithm:

Pseudocode:

```
a[] // a random given array
n = a.size()

for j = 0 to n/2 do:          // make sure it runs n/2 times    n/2+1 times
  for i = 1 to n-1 do:        // move from left to right          n-1 times
    if (a[i] == black && a[i+1] != black):            // check for swappable elements          3tu
      swap;

  for j = n-1 down to 1 do:  // move from right to left
    if(a[j] == white && a[j-1] != white): // check for swappable elements   3tu
      swap;
```

Step Count: 3(n^2-n)/2 + 3n-3.

Time complexity: O(n^2)

Alternate Algorithm Pseudocode:

Pseudocode:

```
a[] // a random given array
n = a.size()
bool sorted
while(!sorted) do:          n-1 times
  sorted = true
  for i = 1 to n-1 do:  // move from left to right      n-1-1+1 = n-1 times
    if (a[i] == black && a[i+1] != black): // check for swappable elements  3tu
      swap;
    sorted = false;

  for i = 2 to n-2 do: // check the second left to second right disc  n-2-2+1 = n-3 times
     if (a[i] == black && a[i+1] != black): // check for swappable elements 3tu
      swap;
     sorted = false;
```

Step Count: 2n^2 - 2n

Time complexity: O(n^2)

5. Time Complexity

Lawnmowers Algorithm:

Step Count = OuterFLoop * InnerLoop1 * InnerLoop2
OuterFLoop = n/2

InnerLoop1 = OuterFLoop * IL1runs
IL1runs = n-1
InnerLoop1 = n*n-1

InnerLoop2 = OuterFLoop * IL2runs
IL2runs = n-1
InnerLoop2 = n*n-1

Step Count = (n/2) * (n*n-1) * (n*n-1) →
(n/2) * 2n * (n-1) →
(n/2)*(2n^2-2n) →
(n/2) * n * (2n-2) →
n*(2n-2) → **2n^2-2n**

As n → Infinity, 2n^2-2n will approach 2n^2, meaning this algorithm performs at roughly O(n^2) as a time complexity

Alternate Algorithm:

Step Count = countInWhile * #ofWLoop
#ofWLoop = n-1
countInWhile = 1 + FirstLoop + SecondLoop

FirstLoop = countInLoop1 * #ofFLoop1
countInLoop1 = 3
#ofFLoop1 = n-1
FirstLoop = 3(n-1)

SecondLoop = countInLoop2 * #ofFLoop2
countInLoop2 = 3
#ofFLoop2 = n-3
SecondLoop = 3(n-3)

countInWhile = n-1 + FirstLoop + SecondLoop
Step Count = (n-1)*3(n-1) + (n-3)*3(n-3) → 9n^2-36n+27

As $n \to$ Infinity, $9n^2-36n+27$ will approach $9n^2$, meaning the Alternate algorithm has a time complexity of $O(n^2)$