

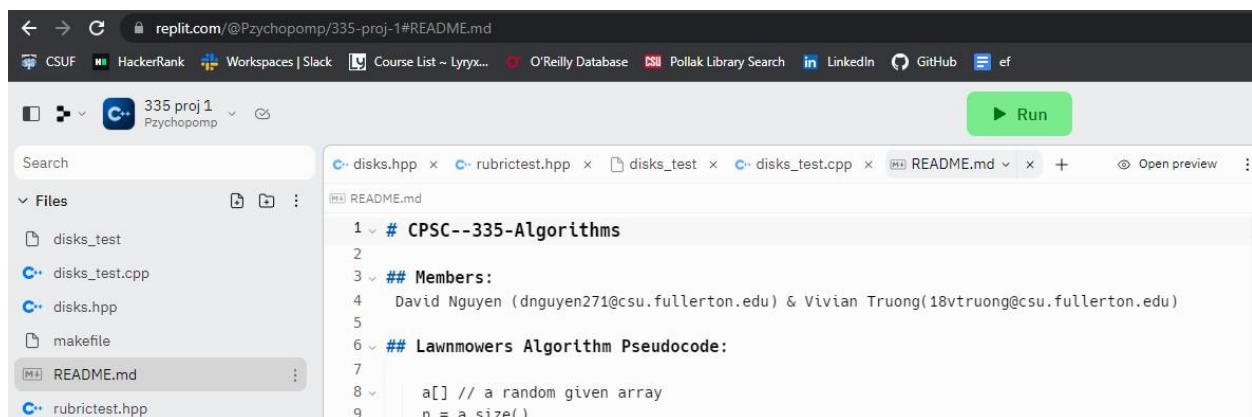
David Nguyen (dnguyen271@csu.fullerton.edu)

Vivian Truong(18vtruong@csu.fullerton.edu)

Github: <https://github.com/Pzychopomp/CPSC-335-Algorithms-Proj-1>

CPSC 335 Project 1 Submission PDF

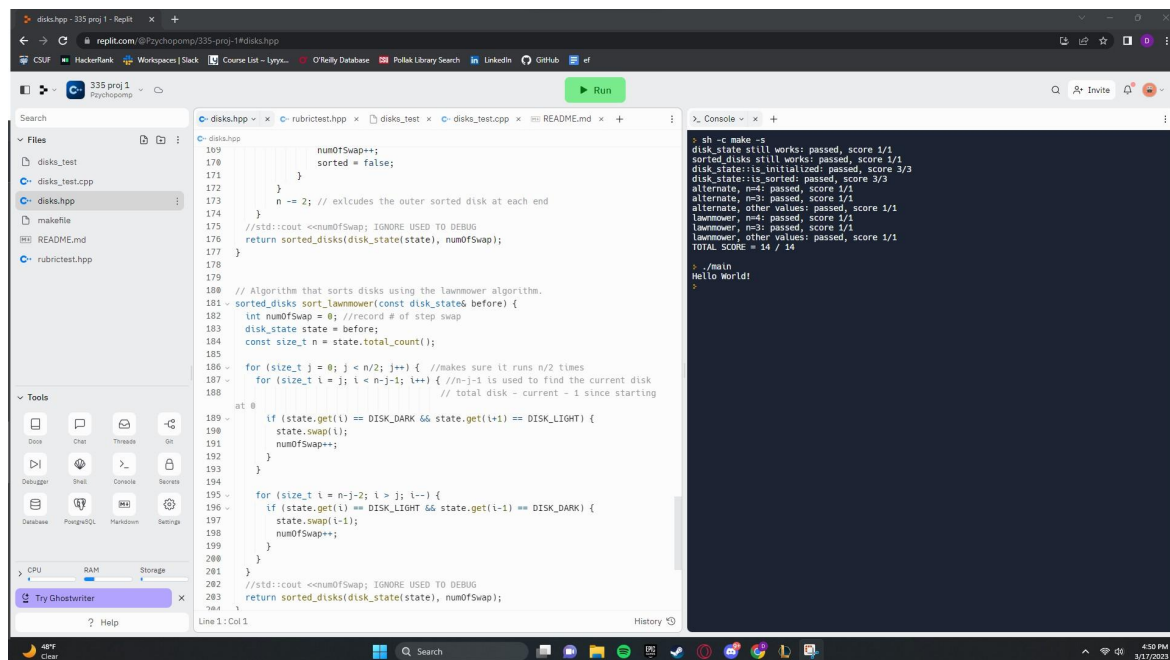
2. The following is a screenshot inside Replit, the editor used for this project:



The screenshot shows the Replit editor interface. The top bar displays the URL `replit.com/@Pzychopomp/335-proj-1#README.md` and various navigation links. The left sidebar shows the file explorer with the following files: `disks_test`, `disks_test.cpp`, `disks.hpp`, `makefile`, `README.md` (selected), and `rubrictest.hpp`. The main editor area displays the `README.md` file with the following content:

```
1 # CPSC--335-Algorithms
2
3 ## Members:
4 David Nguyen (dnguyen271@csu.fullerton.edu) & Vivian Truong(18vtruong@csu.fullerton.edu)
5
6 ## Lawnmowers Algorithm Pseudocode:
7
8 a[] // a random given array
9 n = a.size()
```

3. Screenshot of execution:



The screenshot shows the Replit editor interface with the `disks.hpp` file selected in the left sidebar. The main editor area displays the `disks.hpp` file with the following content:

```
169 numOfSwap++;
170 sorted = false;
171 }
172 }
173 n -= 2; // excludes the outer sorted disk at each end
174 }
175 //std::cout << numOfSwap; IGNORE USED TO DEBUG
176 return sorted_disks(disk_state(state), numOfSwap);
177 }
178
179 // Algorithm that sorts disks using the lawnmower algorithm.
180 sorted_disks sort_lawnmower(const disk_state& before) {
181 int numOfSwap = 0; //record # of step swap
182 disk_state state = before;
183 const size_t n = state.total_count();
184
185 for (size_t j = 0; j < n/2; j++) { //makes sure it runs n/2 times
186 for (size_t i = j; i < n-j-1; i++) { //n-j-1 is used to find the current disk
187 // total disk - current - 1 since starting
188
189 if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
190 state.swap(i);
191 numOfSwap++;
192 }
193 }
194
195 for (size_t i = n-j-2; i > j; i--) {
196 if (state.get(i) == DISK_LIGHT && state.get(i-1) == DISK_DARK) {
197 state.swap(i-1);
198 numOfSwap++;
199 }
200 }
201 }
202 //std::cout << numOfSwap; IGNORE USED TO DEBUG
203 return sorted_disks(disk_state(state), numOfSwap);
204 }
```

The right sidebar shows the console output, which displays the results of the execution:

```
> sh -c make -s
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, m=2: passed, score 1/2
alternate, other values: passed, score 1/1
lawnmower, n=4: passed, score 1/1
lawnmower, m=2: passed, score 1/1
lawnmower, other values: passed, score 1/1
TOTAL SCORE = 14 / 14

./main
Hello World!
```

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 = $(n/2) * (n * n - 1) * (n * n - 1) \rightarrow$

$(n/2) * 2n * (n-1) \rightarrow$

$(n/2) * (2n^2 - 2n) \rightarrow$

$(n/2) * n * (2n-2) \rightarrow$

$n * (2n-2) \rightarrow 2n^2 - 2n$

As $n \rightarrow \text{Infinity}$, $2n^2 - 2n$ 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 = $n-1 + \text{FirstLoop} + \text{SecondLoop}$

Step Count = $(n-1) * 3(n-1) + (n-3) * 3(n-3) \rightarrow 9n^2 - 36n + 27$

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