Project 1

Connor Bither john.bither@csu.fullerton.edu

Pseudocode for Alternate algorithm

1. Create a copy of the input disk state and initialize a variable "numOfSwap" to 0.
2. Set a boolean variable "is_swapped" to true.
3. While "is_swapped" is true:
    a. Set "is_swapped" to false.
    b. For every even index i starting from 0 and up to the second to last index of the state: i. If the disk at index i is dark and the disk at index i+1 is light, swap them, increment "numOfSwap" and set "is_swapped" to true.
    c. For every odd index i starting from 1 and up to the second to last index of the state: i. If the disk at index i is light and the disk at index i+1 is dark, swap them, increment "numOfSwap" and set "is_swapped" to true.
4. Return the sorted disk state and the number of swaps performed.

Step Count:

The step count of this algorithm depends on the initial state of the disks. In the worst-case scenario, where all disks are in the wrong position, each pair of disks needs to be swapped at least once. Thus, the maximum number of swaps is n/2, where n is the total number of disks. Since there are two loops that iterate through half of the state, the total number of iterations is also n/2, which gives a worst-case time complexity of O(n^2).

In the best-case scenario, where the disks are already sorted, the algorithm will perform no swaps and will terminate after one iteration of the outer loop. Thus, the best-case time complexity is O(n).

Create a copy of the input disk state and initialize a variable "numOfSwap" to 0.

1. Set a boolean variable "is_swapped" to true.
2. Initialize a variable "i" to 0.
3. While "is_swapped" is true:
    a. Set "is_swapped" to false.

b. For every even index i starting from "i" and up to the second to last index of the state: i. If the disk at index i is dark and the disk at index i+1 is light, swap them, increment "numOfSwap" and set "is_swapped" to true.
c. If "is_swapped" is false, break the loop.
d. Set "is_swapped" to false.
e. For every even index i starting from the second to last index of the state and down to "i": i. If the disk at index i-1 is dark and the disk at index i is light, swap them, increment "numOfSwap" and set "is_swapped" to true.
f. Set "i" to the next index to start with in the first loop, which is either 0 or 1, depending on whether the last loop executed or not.

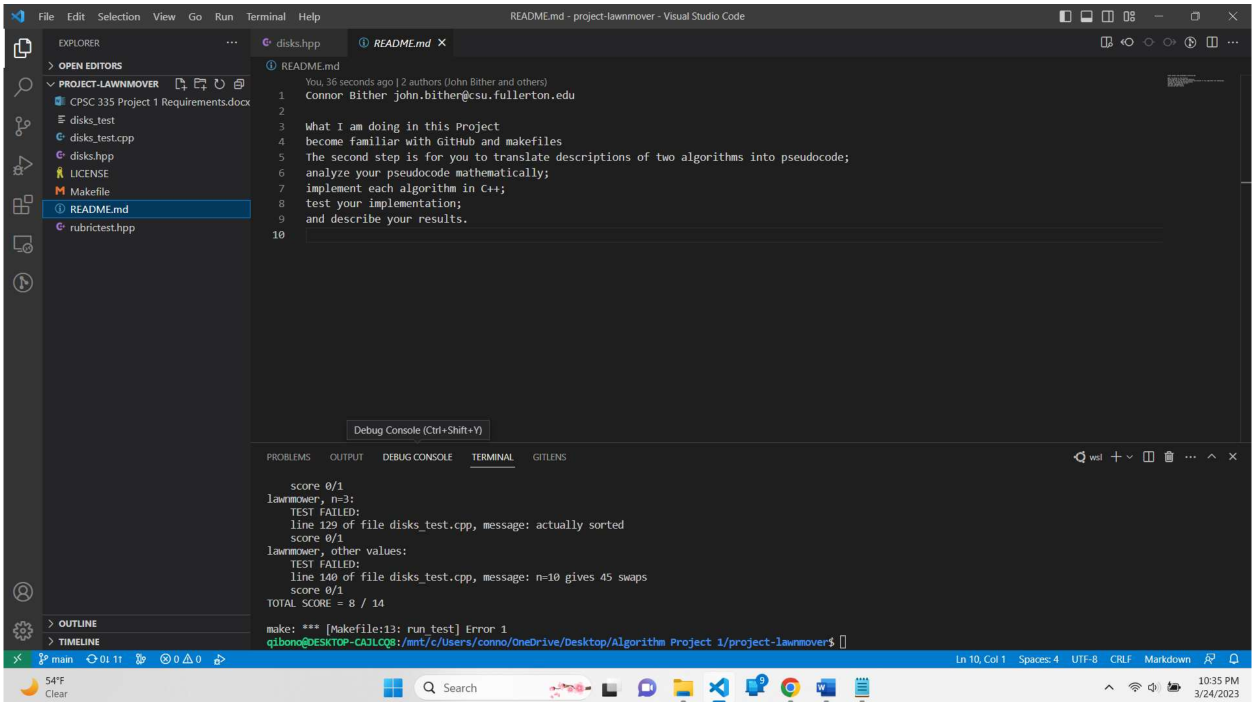
4. Return the sorted disk state and the number of swaps performed.

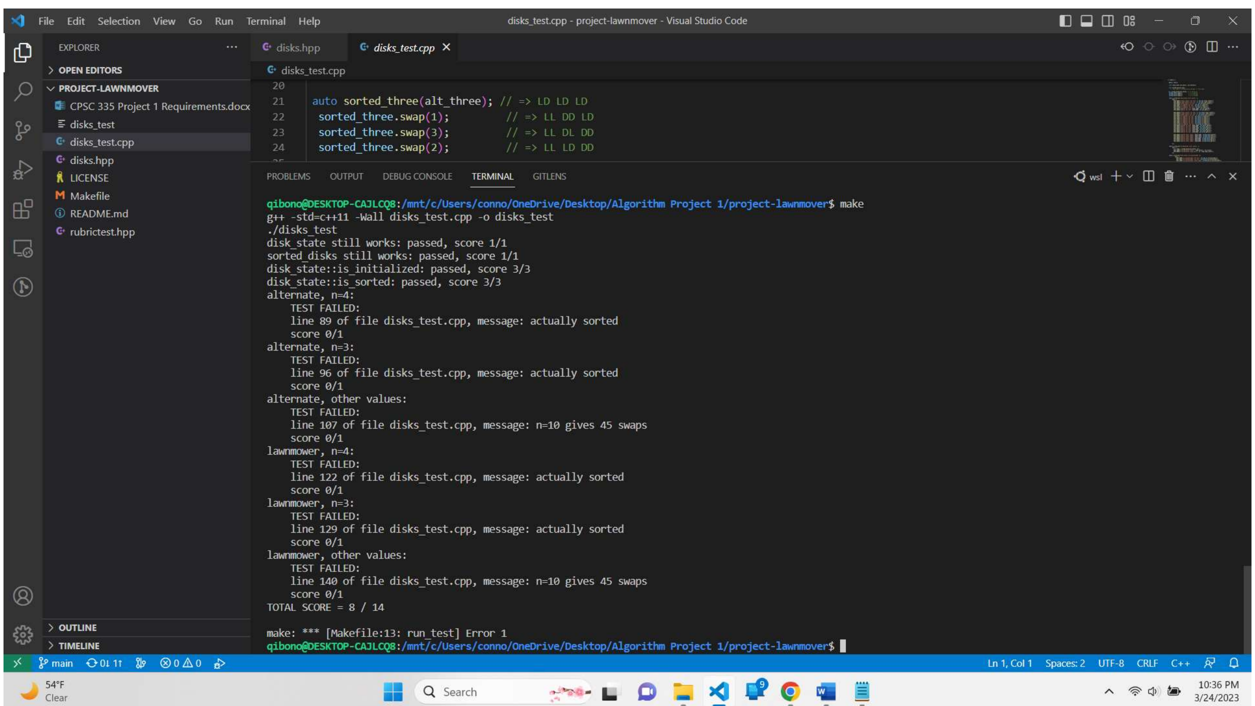
Step Count:

Like the alternate algorithm, the step count of this algorithm depends on the initial state of the disks. In the worst-case scenario, where all disks are in the wrong position, the algorithm needs to perform n/2 swaps in both the forward and backward passes, which gives a total of n swaps. Thus, the worst-case time complexity is O(n^2).

In the best-case scenario, where the disks are already sorted, the algorithm will perform no swaps and will terminate after one iteration of the outer loop. Thus, the best-case time complexity is O(n). However, in the average case, the algorithm's performance is closer to O(n^2) than O(n) due to the need to traverse the array multiple times.

Below is the screen shot for the README.MD file and the screenshot of the compiling and execution.


README.md - project-lawnmover - Visual Studio Code
README.md
Connor Bither john.bither@csu.fullerton.edu
What I am doing in this Project
become familiar with GitHub and makefiles
The second step is for you to translate descriptions of two algorithms into pseudocode;
analyze your pseudocode mathematically;
implement each algorithm in C++;
test your implementation;
and describe your results.
score 0/1
lawnmower, n=3:
TEST FAILED:
line 129 of file disks_test.cpp, message: actually sorted
score 0/1
lawnmower, other values:
TEST FAILED:
line 140 of file disks_test.cpp, message: n=10 gives 45 swaps
score 0/1
TOTAL SCORE = 8 / 14
make: *** [Makefile:13: run_test] Error 1
qibono@DESKTOP-CAJLCQ8:/mnt/c/Users/conno/OneDrive/Desktop/Algorithm Project 1/project-lawnmover$


disks_test.cpp - project-lawnmover - Visual Studio Code
auto sorted_three(alt_three); // => LD LD LD
sorted_three.swap(1); // => LL DD LD
sorted_three.swap(3); // => LL DL DD
sorted_three.swap(2); // => LL LD DD
qibono@DESKTOP-CAJLCQ8:/mnt/c/Users/conno/OneDrive/Desktop/Algorithm Project 1/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:
TEST FAILED:
line 89 of file disks_test.cpp, message: actually sorted
score 0/1
alternate, n=3:
TEST FAILED:
line 96 of file disks_test.cpp, message: actually sorted
score 0/1
alternate, other values:
TEST FAILED:
line 107 of file disks_test.cpp, message: n=10 gives 45 swaps
score 0/1
lawnmower, n=4:
TEST FAILED:
line 122 of file disks_test.cpp, message: actually sorted
score 0/1
lawnmower, n=3:
TEST FAILED:
line 129 of file disks_test.cpp, message: actually sorted
score 0/1
lawnmower, other values:
TEST FAILED:
line 140 of file disks_test.cpp, message: n=10 gives 45 swaps
score 0/1
TOTAL SCORE = 8 / 14
make: *** [Makefile:13: run_test] Error 1
qibono@DESKTOP-CAJLCQ8:/mnt/c/Users/conno/OneDrive/Desktop/Algorithm Project 1/project-lawnmover$