Harry Dinh
20hdinh@csu.fullerton.edu
CPSC 335 Project 1

Pseudocode:

Assumption - always an even number of disks (always in pairs) because number of disks = 2n. There are n disks light and n disks dark.

Sort Alternate:

```
def sort_alternate(n, disks):
    begin = 0
    end = 2n
        // begin and end will converge in the middle of the disks
        // while loop ends when begin > end
        while (begin HAS NOT PASSED end) do:
            for (i = begin to end) do:
                if <CURRENT AND NEXT DISK OUT OF ORDER> then <SWAP BOTH DISKS>
                i += 2
                // for loop ends when begin > end
            begin++
            end--
    return disks
```

Sort Lawnmower:

```
def sort_lawnmower(n, disks):
    // docx description sounds a lot like cocktail sort
    // EDIT: it is
    // TL;DR do bubble sort, but do both directions, left then right
    for (i = 0 to n / 4) do:
        // left to right
        for (j = 0 to 2n) do:
            if <CURRENT AND RIGHT DISK OUT OF ORDER> then <SWAP BOTH DISKS>
            j++
        // right to left
        for (k = 2n to 0) do:
            if <CURRENT AND LEFT DISK OUT OF ORDER> then <SWAP BOTH DISKS>
            k--
    return disks
```

Analysis:

Sort Alternate: (comments removed)

```
def sort_alternate(n, disks):
   begin = 0  1 tu
   end = 2n  1 tu
      while (begin HAS NOT PASSED end) do:  n tu
         for (i = begin to end) do:  n tu
            if <CURRENT AND NEXT DISK OUT OF ORDER> then <SWAP BOTH DISKS>  1 tu
            i += 2  1 tu
         begin++  1 tu
         end--  1 tu
   return disks
```

1 + 1 + (n * (n * (1 + 1)) * (1 + 1)) = 2 + (n * (2n) * 2) = 2 + 4n^2 tu → O(n^2)

alternatively:

n = 1: 1 step
    run 1: 1 pair (1 step)

n = 2: 2 + 1 = 3 steps
    run 1: 2 pairs (2 steps)
    run 2: 1 pair (1 step)

n = 3: 3 + 2 + 1 = 6 steps
    run 1: 3 pairs (3 steps)
    run 2: 2 pairs (2 steps)
    run 1: 1 pair (1 step)

…

n pairs:

n + (n – 1) + (n – 2) + … + 1 = $\sum_{i=1}^{n} x_i = \frac{n(n+1)}{2}$ steps

$$\frac{n(n+1)}{2} = O(n^2)$$

Sort Lawnmower: (comments removed)

```
def sort_lawnmower(n, disks):
    for (i = 0 to n / 2) do:  n / 2 tu
        for (j = 0 to 2n) do:  2n tu
            if <CURRENT AND RIGHT DISK OUT OF ORDER> then <SWAP BOTH DISKS>  1 tu
            j++  1 tu
        for (k = 2n to 0) do:  2n tu
            if <CURRENT AND LEFT DISK OUT OF ORDER> then <SWAP BOTH DISKS>  1 tu
            k--  1 tu
    return disks

n/2 * [ (2n * (1 + 1)) + (2n * (1 + 1)) ] = n/2 * (4n + 4n) = 4n^2 tu = O(n^2)
```

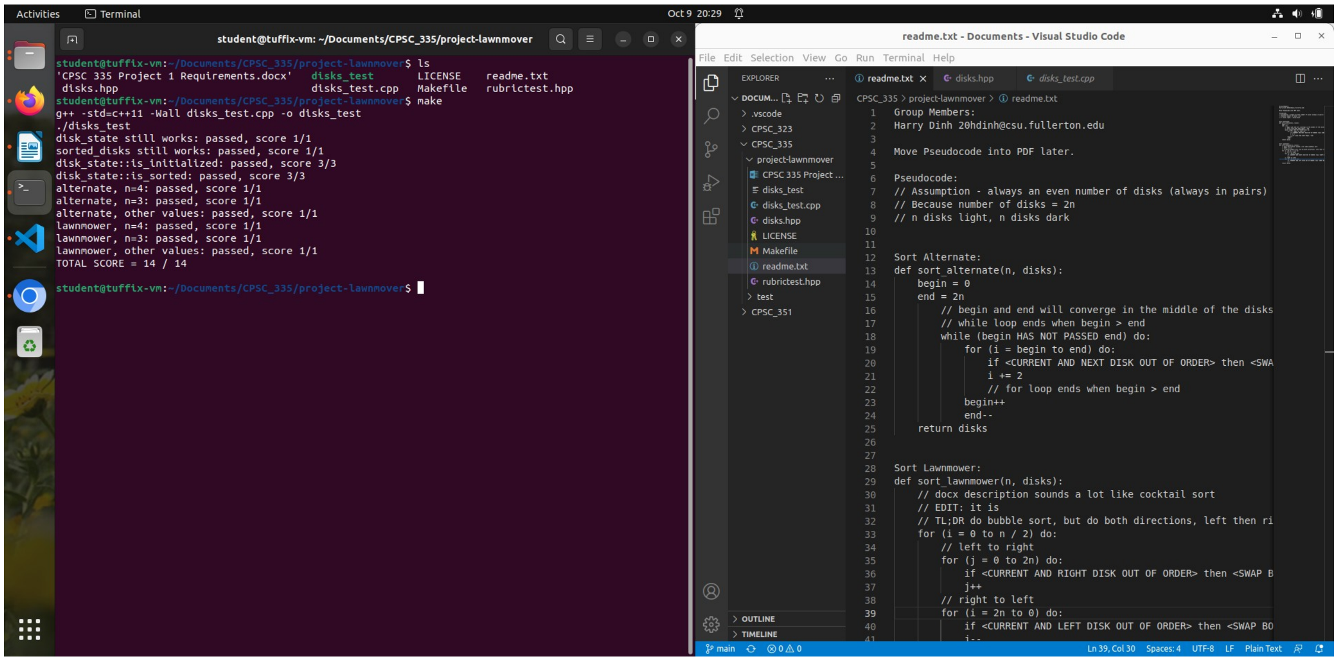
alternatively:

Lawnmower sort = n/2 runs * [bubble sort (left to right) + bubble sort(right to left)]

$n/2 \rightarrow O(n)$
Bubble Sort = $O(n)$

$O(n) * O(n) = O(n^2)$

Screenshots:

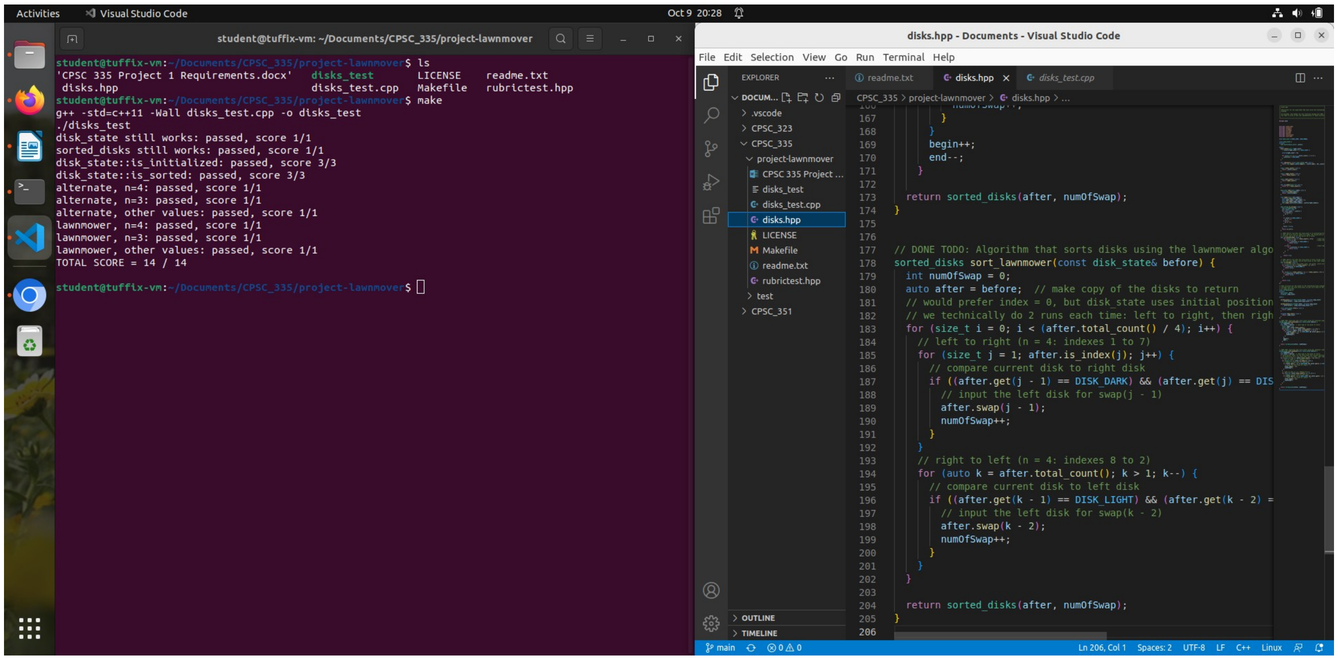


```
student@tuffix-vm:~/Documents/CPSC_335/project-lawnmover$ ls
'CPSC 335 Project 1 Requirements.docx'   disks_test      LICENSE    readme.txt
 disks.hpp                               disks_test.cpp  Makefile   rubrictest.hpp
student@tuffix-vm:~/Documents/CPSC_335/project-lawnmover$ make
g++ -std=c++11 -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:~/Documents/CPSC_335/project-lawnmover$
```

readme.txt - Documents - Visual Studio Code

```
Group Members:
Harry Dinh 20hdinh@csu.fullerton.edu

Move Pseudocode into PDF later.

Pseudocode:
// Assumption - always an even number of disks (always in pairs)
// Because number of disks = 2n
// n disks light, n disks dark


Sort Alternate:
def sort_alternate(n, disks):
    begin = 0
    end = 2n
        // begin and end will converge in the middle of the disks
        // while loop ends when begin > end
        while (begin HAS NOT PASSED end) do:
            for (i = begin to end) do:
                if <CURRENT AND NEXT DISK OUT OF ORDER> then <SWA
                i += 2
                // for loop ends when begin > end
            begin++
            end--
    return disks


Sort Lawnmower:
def sort_lawnmower(n, disks):
    // docx description sounds a lot like cocktail sort
    // EDIT: it is
    // TL;DR do bubble sort, but do both directions, left then ri
    for (i = 0 to n / 2) do:
        // left to right
        for (j = 0 to 2n) do:
            if <CURRENT AND RIGHT DISK OUT OF ORDER> then <SWAP B
            j++
        // right to left
        for (i = 2n to 0) do:
            if <CURRENT AND LEFT DISK OUT OF ORDER> then <SWAP BO
```

Ln 39, Col 30   Spaces: 4   UTF-8   LF   Plain Text



```
student@tuffix-vm:~/Documents/CPSC_335/project-lawnmover$ ls
'CPSC 335 Project 1 Requirements.docx'   disks_test      LICENSE    readme.txt
 disks.hpp                               disks_test.cpp  Makefile   rubrictest.hpp
student@tuffix-vm:~/Documents/CPSC_335/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:~/Documents/CPSC_335/project-lawnmover$
```

disks.hpp - Documents - Visual Studio Code

```cpp
        }
      }
      begin++;
      end--;
    }

  return sorted_disks(after, numOfSwap);
}


// DONE TODO: Algorithm that sorts disks using the lawnmower algo
sorted_disks sort_lawnmower(const disk_state& before) {
  int numOfSwap = 0;
  auto after = before;  // make copy of the disks to return
  // would prefer index = 0, but disk_state uses initial position
  // we technically do 2 runs each time: left to right, then righ
  for (size_t i = 0; i < (after.total_count() / 4); i++) {
    // left to right (n = 4: indexes 1 to 7)
    for (size_t j = 1; after.is_index(j); j++) {
      // compare current disk to right disk
      if ((after.get(j - 1) == DISK_DARK) && (after.get(j) == DIS
        // input the left disk for swap(j - 1)
        after.swap(j - 1);
        numOfSwap++;
      }
    }
    // right to left (n = 4: indexes 8 to 2)
    for (auto k = after.total_count(); k > 1; k--) {
      // compare current disk to left disk
      if ((after.get(k - 1) == DISK_LIGHT) && (after.get(k - 2) =
        // input the left disk for swap(k - 2)
        after.swap(k - 2);
        numOfSwap++;
      }
    }
  }

  return sorted_disks(after, numOfSwap);
}
```

Ln 206, Col 1   Spaces: 2   UTF-8   LF   C++   Linux