

Alternate and Lawnmower Project 1 PDF Report

Submission:

This report and project submission is for project 1 for Computer Science Algorithms 335.

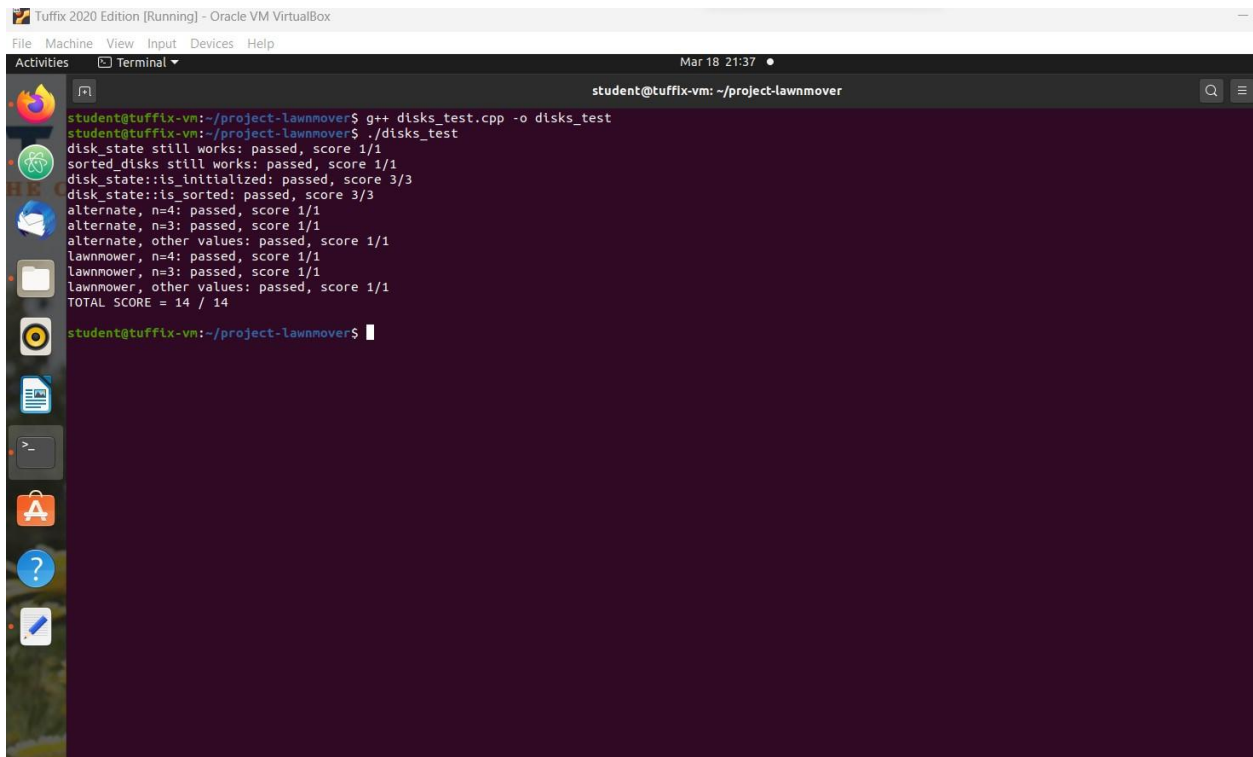
Contributors:

Ricardo Granados Macias: ricardog2002@csu.fullerton.edu

Tommy Ly: lytommy321@csu.fullerton.edu

Project Screenshots:

Program Compiling & Passing Tests:



```
Tuffix 2020 Edition [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Mar 18 21:37
student@tuffix-vm: ~/project-lawnmover
student@tuffix-vm:~/project-lawnmover$ g++ disks_test.cpp -o disks_test
student@tuffix-vm:~/project-lawnmover$ ./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:~/project-lawnmover$
```

Is Sorted Function:

```
Tuffix 2020 Edition [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

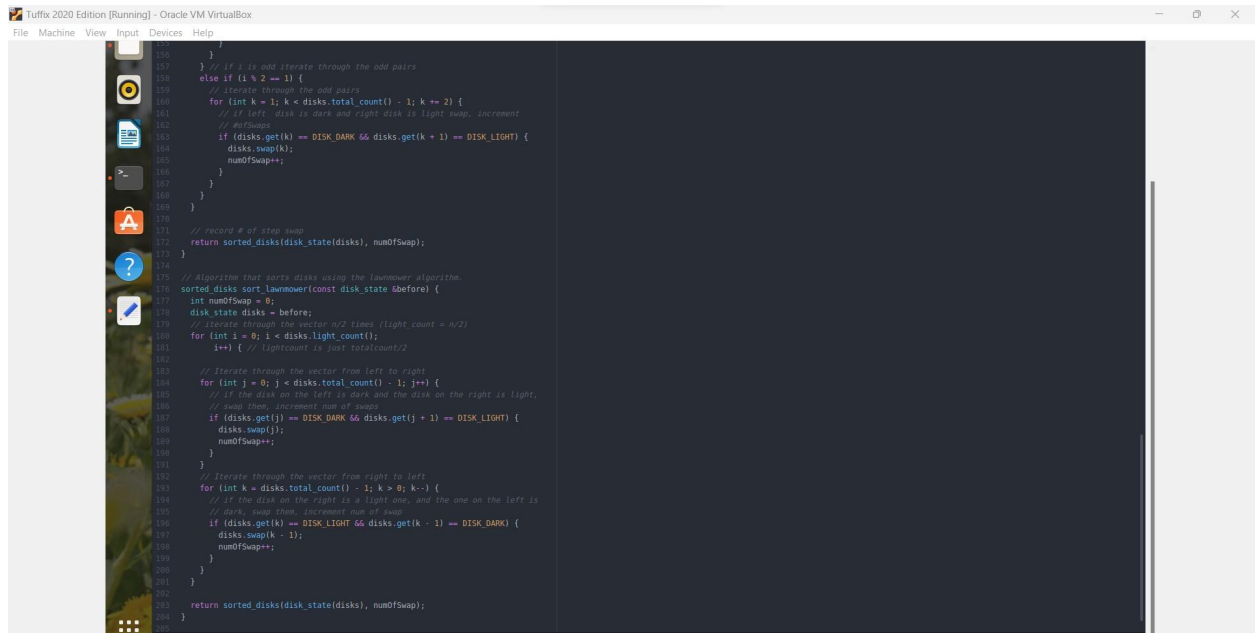
74     } else {
75         ss << "D";
76     }
77     first = false;
78 }
79 return ss.str();
80 }
81
82 // Return true when this disk state is in alternating format. That means
83 // that the first disk at index 0 is light, the second disk at index 1
84 // is dark, and so on for the entire row of disks.
85 bool is_initialized() const {
86     for (size_t i = 0; i < total_count(); i++) {
87         // check each position before function
88         if (i % 2 == 0) { // check even position -> should be light
89             if (colors[i] == DISK_DARK) {
90                 return false;
91             }
92         } else { // check odd position -> should be dark
93             if (colors[i] == DISK_LIGHT) {
94                 return false;
95             }
96         }
97     }
98     return true;
99 }
100
101 // Return true when this disk state is fully sorted, with all light disks on
102 // the left (low indices) and all dark disks on the right (high indices).
103 bool is_sorted() const {
104     // Iterate through the vector containing disk state
105     for (int i = 0; i < total_count(); i++) {
106         // Check if we have dark disk in the lower indices, return false if we do
107         if (get(i) == DISK_DARK && i < total_count() / 2) {
108             return false;
109         }
110         // Check if we have dark disk in the lower indices, return false if we
111         // do
112         else if (get(i) == DISK_LIGHT && i > total_count() / 2) {
113             return false;
114         }
115     }
116     // We didn't find any disk out of place, return true
117     return true;
118 }
119 };
120
121 // Data structure for the output of the alternating disks problem. That
122 // includes both the final disk state, as well as a count of the number
123
124 -/project/alternating/disk.hpp 1.1
```

Alternate Sort Algorithm:

```
Tuffix 2020 Edition [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

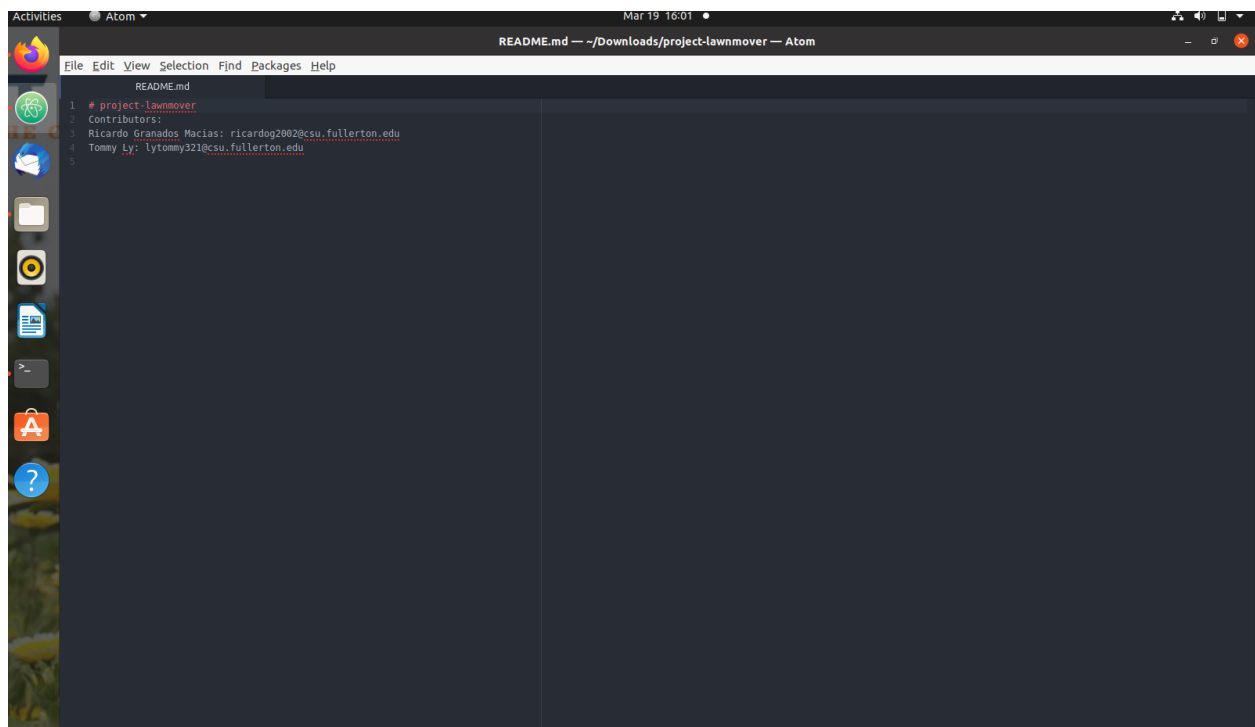
129 public:
130     sorted_disks(const disk_state &after, unsigned swap_count)
131     { _after(after), _swap_count(swap_count) {} }
132
133     sorted_disks(disk_state &after, unsigned swap_count)
134     { _after(after), _swap_count(swap_count) {} }
135
136     const disk_state &after() const { return _after; }
137
138     unsigned swap_count() const { return _swap_count; }
139 };
140
141 // Algorithm that sorts disks using the alternate algorithm.
142 sorted_disks sort_alternate(const disk_state &before) {
143     int numOfSwap = 0;
144     disk_state disks = before;
145     // Run swap for every disk in the vector
146     for (int i = 0; i < disks.total_count() - 1; i++) {
147         // If i is even, then we iterate through every even pair
148         if (i % 2 == 0) {
149             // Iterate through each even pair
150             for (int j = 0; j < disks.total_count() - 1; j += 2) {
151                 // If left disk is dark and right disk is light swap, increment #ofSwaps
152                 if (disks.get(j) == DISK_DARK && disks.get(j + 1) == DISK_LIGHT) {
153                     disks.swap(j);
154                     numOfSwap++;
155                 }
156             }
157         } // If i is odd iterate through the odd pairs
158         else if (i % 2 == 1) {
159             // Iterate through the odd pairs
160             for (int k = 1; k < disks.total_count() - 1; k += 2) {
161                 // If left disk is dark and right disk is light swap, increment
162                 // #ofSwaps
163                 if (disks.get(k) == DISK_DARK && disks.get(k + 1) == DISK_LIGHT) {
164                     disks.swap(k);
165                     numOfSwap++;
166                 }
167             }
168         }
169     }
170
171     // record # of swap swap
172     return sorted_disks(disk_state(disks), numOfSwap);
173 }
174
175 // Algorithm that sorts disks using the Towers algorithm
```

Lawnmower Algorithm:



```
156     }
157 } // If i is odd iterate through the odd pairs
158 else if (i % 2 == 1) {
159     // Iterate through the odd pairs
160     for (int k = 1; k < disks.total_count() - 1; k += 2) {
161         // If left disk is dark and right disk is light swap, increment
162         // numSwaps
163         if (disks.get(k) == DISK_DARK && disks.get(k + 1) == DISK_LIGHT) {
164             disks.swap(k);
165             numOfSwaps++;
166         }
167     }
168 }
169 // Record # of step swap
170 return sorted_disks(disk_state(disks), numOfSwaps);
171 }
172
173 // Algorithm that sorts disks using the lawnmower algorithm.
174 sorted_disks sort_lawnmower(const disk_state &before) {
175     int numOfSwaps = 0;
176     disk_state disks = before;
177     // Iterate through the vector n/2 times (light count = n/2)
178     for (int i = 0; i < disks.light_count(); i++) { // lightcount is just totalcount/2
179         // Iterate through the vector from left to right
180         for (int j = 0; j < disks.total_count() - 1; j++) {
181             // If the disk on the left is dark and the disk on the right is light,
182             // swap them, increment num of swaps
183             if (disks.get(j) == DISK_DARK && disks.get(j + 1) == DISK_LIGHT) {
184                 disks.swap(j);
185                 numOfSwaps++;
186             }
187         }
188         // Iterate through the vector from right to left
189         for (int k = disks.total_count() - 1; k > 0; k--) {
190             // If the disk on the right is a light one, and the one on the left is
191             // dark, swap them, increment num of swap
192             if (disks.get(k) == DISK_LIGHT && disks.get(k - 1) == DISK_DARK) {
193                 disks.swap(k - 1);
194                 numOfSwaps++;
195             }
196         }
197     }
198     return sorted_disks(disk_state(disks), numOfSwaps);
199 }
200 }
```

ReadME FILE:



```
1 # project-lawnmover
2 Contributors:
3 Ricardo Granados Macias: ricardog2002@csu.fullerton.edu
4 Tommy Ly: lytommy321@csu.fullerton.edu
5
```

Video of the Program Compiling:

Video of program compiling and passing the make tests:

<https://drive.google.com/file/d/1bchFSZuqp1I7RF05jVEuptOeJTr3KxuT/view?usp=sharing>

Pseudocode:

Alternate Algorithm Pseudocode:

```
I/P: Unsorted vector of disks, can't be empty, size n
O/P: Sorted vector of disks, can't be empty, size n
/* def alternate_sort(unsorted) :
sorted = unsorted      // 1TU
numSwap = 0            // 1TU
for i = 0 to n - 1 do: // (n - 1 + 1) TU
    if ( i % 2 == 0) then // ( 2 TU)
        for j = 0 to n - 1 step 2 do: // ( (n - 1) / 2 ) + 1 TU
            if (left == dark && right == light) then // 3 TU
                swap(left, right) // 1 TU
                numSwap++ // 1 TU
            endif
        end innerfor
    end if
    if ( i % 2 == 1) do: // 2 TU
        for k = 1 to n - 1 step 2 do: // ((n - 1 - 1) / 2) + 1 TU
            if (left == dark && right == light) then // 3 TU
                swap(left, right) // 1 TU
                numSwap++ // 1 TU
            endif
        end innerfor
    end if
end outerfor
return (sorted, numSwap) // 0 TU
```

*/

Alternate Algorithm Step Count Calculaiton:

$$SC = 2 + n(2 + 5n/2 + 5/2 + 2 + 5n/2)$$

$$SC = 2 + n(5n + 13/2)$$

$$SC = 2 + (5n^2) + 13n/2$$

Lawnmower Algorithm Pseudocode:

```
This is the lawnmower pseudoCode in a c++ coding format
//I/P unsorted list of colored disks that is not empty, size n
// O/P sorted list of colored disks, size n, not empty
/*
def lawn_mover_sort(unsorted) :
numSwaps = 0; // 1 TU
sorted_disks = unsorted // 1 TU
for i = 0 to n/2 do:    // (n/2 - 0) + 1 TU
    for i = 0 to n - 1 do: // (n - 1 - 0) + 1 TU
        if (left == dark && right == light) then: // 3TU
            swap(left,right)    // 1 TU
            sorted_disks.numSwaps++; // 1 TU
        end if
    end innerfor1
    for i = n - 1 to 0 do:    // ((0 - (n - 1)) / -1 ) + 1
        if (right == light && left == dark) then: // 3TU
            sorted_disks.swap(left, right) // 1 TU
            numSwaps++;    // 1 TU
        Endif
    end innerfor2
End outerfor
return (sorted_disks, numSwaps) // 0 TU

*/
```

Lawnmover Algorithm Step Count Calculation:

$$\text{Step Count} = 2 + (n/2 + 1) * (5n + 5n)$$

$$\text{Step Count} = 2 + (n/2 + 1) * (10n)$$

$$\text{Step Count} = 10n^2 / 2 + 10n + 2$$

$$\text{Step Count} = 5n^2 + 10n + 2$$

Time Complexity Proofs:

Alternate Algorithm Analysis / Proof:

$$\text{Assume } f(n) = 5n^2 + 13n/2 + 2$$

$$\text{Assume } g(n) = O(n^2)$$

We will prove that $f(n)$ is within $g(n)$ with a proof using limits

Proof by limit:

$$\lim_{n \rightarrow \infty} (5n^2 + 13n/2 + 2) / n^2$$

Divide by the greatest common denominator

$$\lim_{n \rightarrow \infty} (5n^2 / n^2 + 13n / 2 n^2 + 2 / n^2) / n^2 / n^2$$

Simplify to get

$$\lim_{n \rightarrow \infty} (5 + 13/2n + 2/n^2)$$

Substitute infinity for n to get

$$\lim_{n \rightarrow \infty} 5 + 13/\infty + 2/\infty$$

Simplify

$$\text{Limit } 5 + 0 + 0 = 5$$

$$n \rightarrow \infty$$

Therefore, because we get a real number 5, we can conclude by the limit theorem that $f(n)$ is within $O(n^2)$. Therefore, our algorithm is within $O(n^2)$.

Lawnmower Algorithm Analysis /Proof:

$$\text{Assume } f(n) = 5n^2 + 10n + 2$$

$$\text{Assume } g(n) = O(n^2)$$

We will prove that $f(n)$ is within $g(n)$ with a proof using limits

Proof by limit:

$$\text{Limit } (5n^2 + 10n + 2) / n^2$$

$$n \rightarrow \infty$$

Divide by the greatest common denominator

$$\text{Limit } (5n^2 / n^2 + 10n / n^2 + 2 / n^2) / n^2 / n^2$$

$$n \rightarrow \infty$$

Simplify to get

$$\text{Limit } (5 + 10/n + 2/n^2)$$

$$n \rightarrow \infty$$

Substitute infinity for n to get

$$\text{Limit } 5 + 10/\infty + 2/\infty$$

$$n \rightarrow \infty$$

Simplify

$$\text{Limit } 5 + 0 + 0 = 5$$

$$n \rightarrow \infty$$

Therefore, because we get a real number 5, we can conclude by the limit theorem that $f(n)$ is within $O(n^2)$. Therefore, our lawnmower algorithm is within $O(n^2)$.