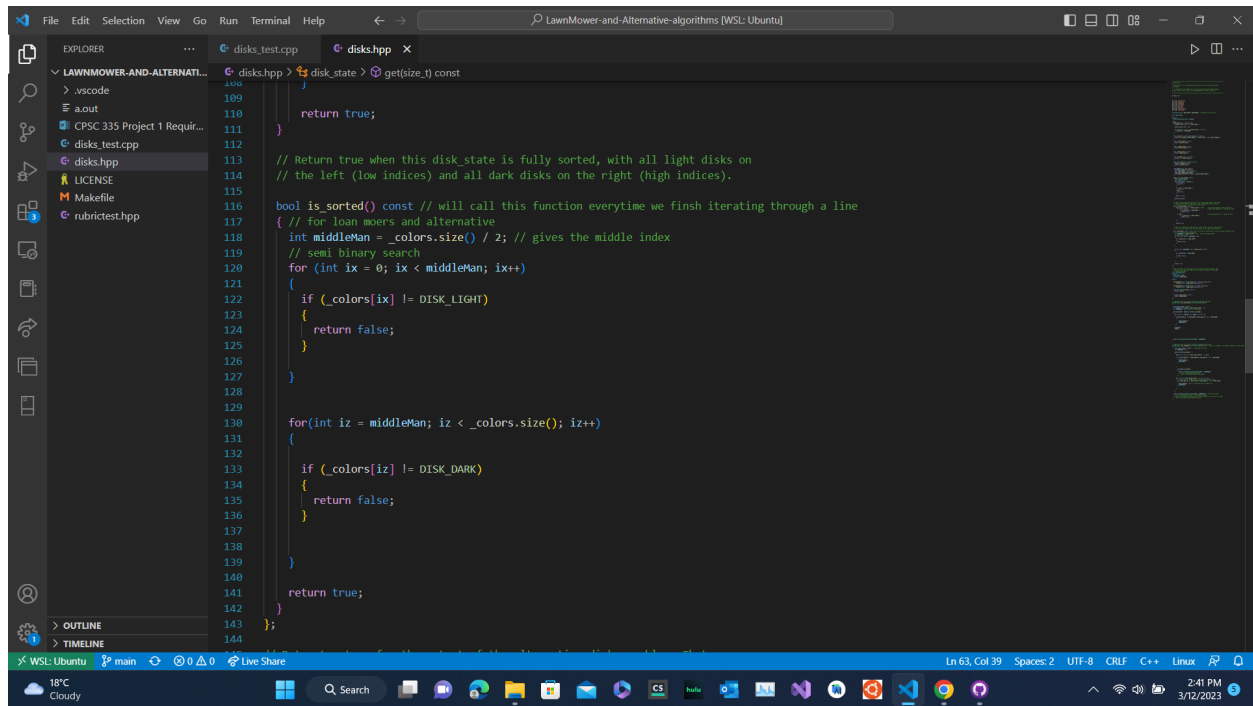


Ryan Haddadi: mochi456@csu.fullerton.edu

Naeem Khayat Albirkdar: arnaeem@csu.fullerton.edu

Code proof using Linux environment:

Is_sorted function:



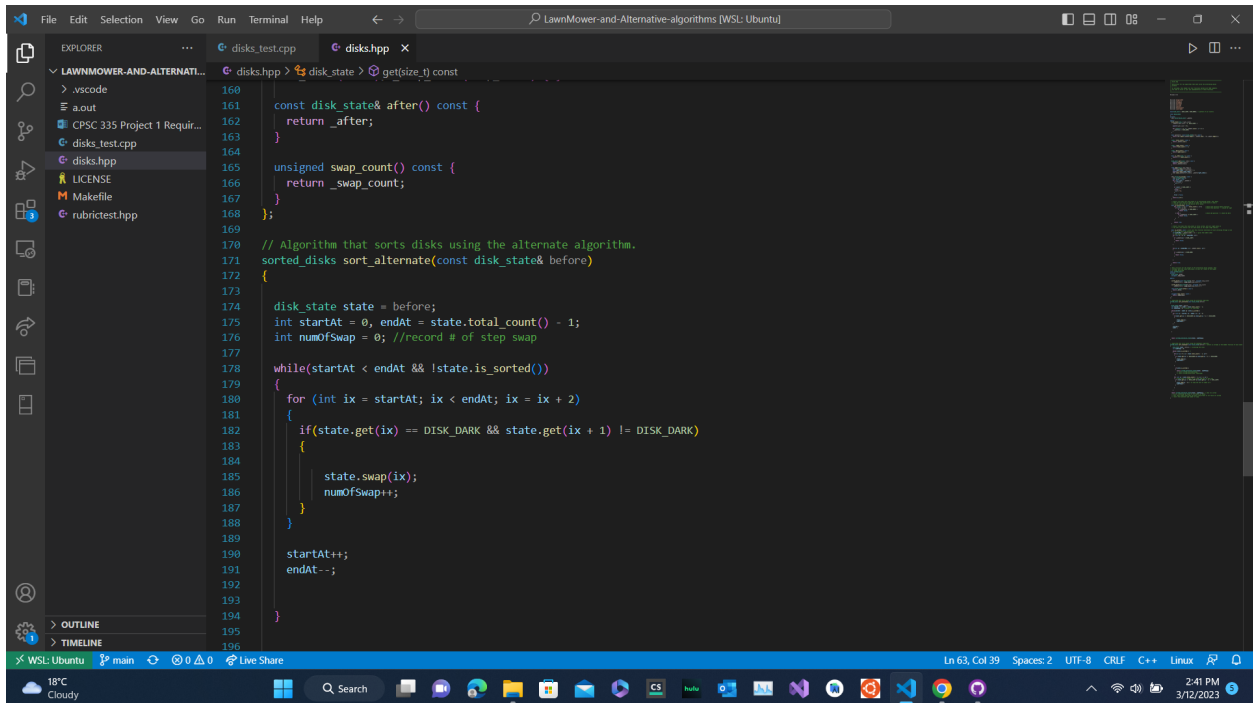
The screenshot shows a Visual Studio Code editor window with the file explorer on the left and the code editor in the center. The file explorer shows a project named 'LAWNMOWER-AND-ALTERNAT...' with files like 'disks_test.cpp', 'disks.hpp', 'LICENSE', 'Makefile', and 'rubric_test.hpp'. The code editor displays the implementation of the 'is_sorted' function in 'disks.hpp'. The function is a const member function that takes a 'disk_state' and returns a boolean. It uses a binary search approach to check if the disk state is sorted. The function is defined as follows:

```
bool is_sorted() const // will call this function everytime we finish iterating through a line
{
    // for loan moers and alternative
    int middleMan = _colors.size() / 2; // gives the middle index
    // see! binary search
    for (int ix = 0; ix < middleMan; ix++)
    {
        if (_colors[ix] != DISK_LIGHT)
        {
            return false;
        }
    }

    for (int iz = middleMan; iz < _colors.size(); iz++)
    {
        if (_colors[iz] != DISK_DARK)
        {
            return false;
        }
    }

    return true;
}
```

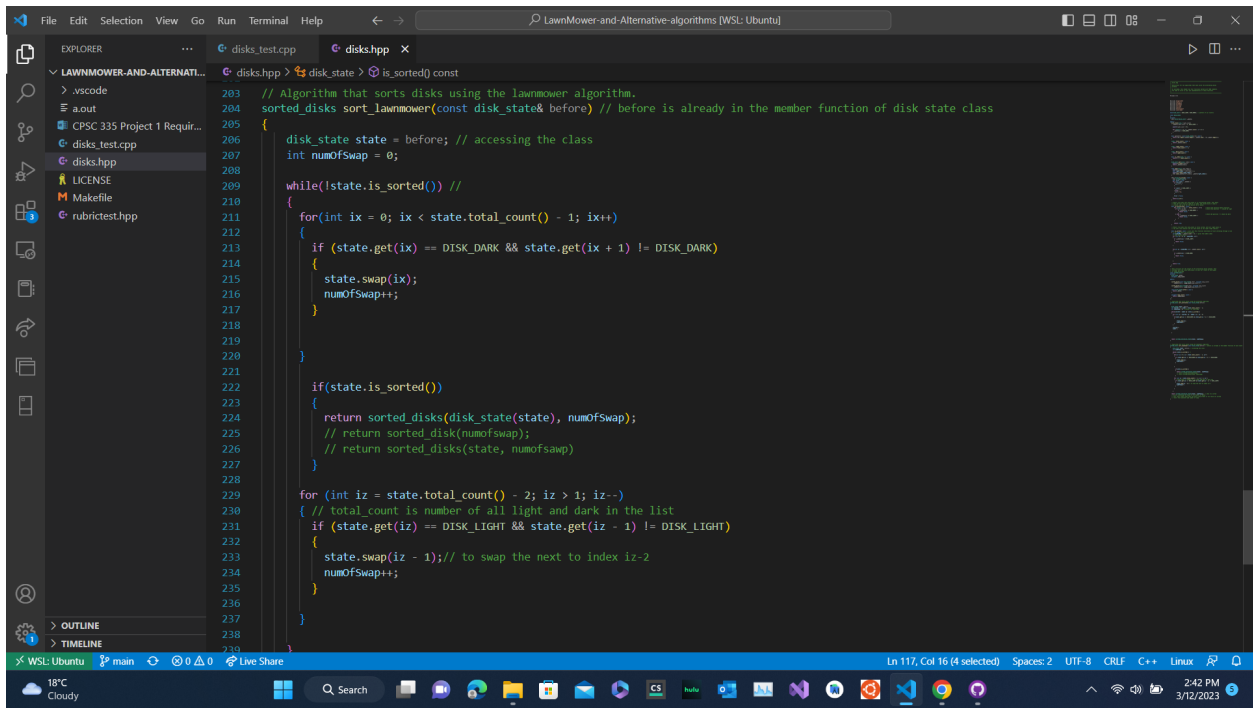
Sort_alternate:



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the implementation of the `Sort_alternate` algorithm in `disks.hpp`. The code includes a `const disk_state& after()` function, an `unsigned swap_count()` function, and a `sorted_disks sort_alternate(const disk_state& before)` function. The `sort_alternate` function uses a while loop to iterate over the disk state, swapping adjacent disks that are not in the correct order (DISK_DARK followed by DISK_LIGHT) and counting the number of swaps.

```
160 const disk_state& after() const {
161     return after;
162 }
163
164 unsigned swap_count() const {
165     return swap_count;
166 }
167
168 };
169
170 // Algorithm that sorts disks using the alternate algorithm.
171 sorted_disks sort_alternate(const disk_state& before)
172 {
173
174     disk_state state = before;
175     int startAt = 0, endAt = state.total_count() - 1;
176     int numOfSwap = 0; //record # of step swap
177
178     while(startAt < endAt && !state.is_sorted())
179     {
180         for (int ix = startAt; ix < endAt; ix = ix + 2)
181         {
182             if(state.get(ix) == DISK_DARK && state.get(ix + 1) != DISK_DARK)
183             {
184                 state.swap(ix);
185                 numOfSwap++;
186             }
187         }
188
189         startAt++;
190         endAt--;
191     }
192
193     return sorted_disks(state, numOfSwap);
194 }
195
196 }
```

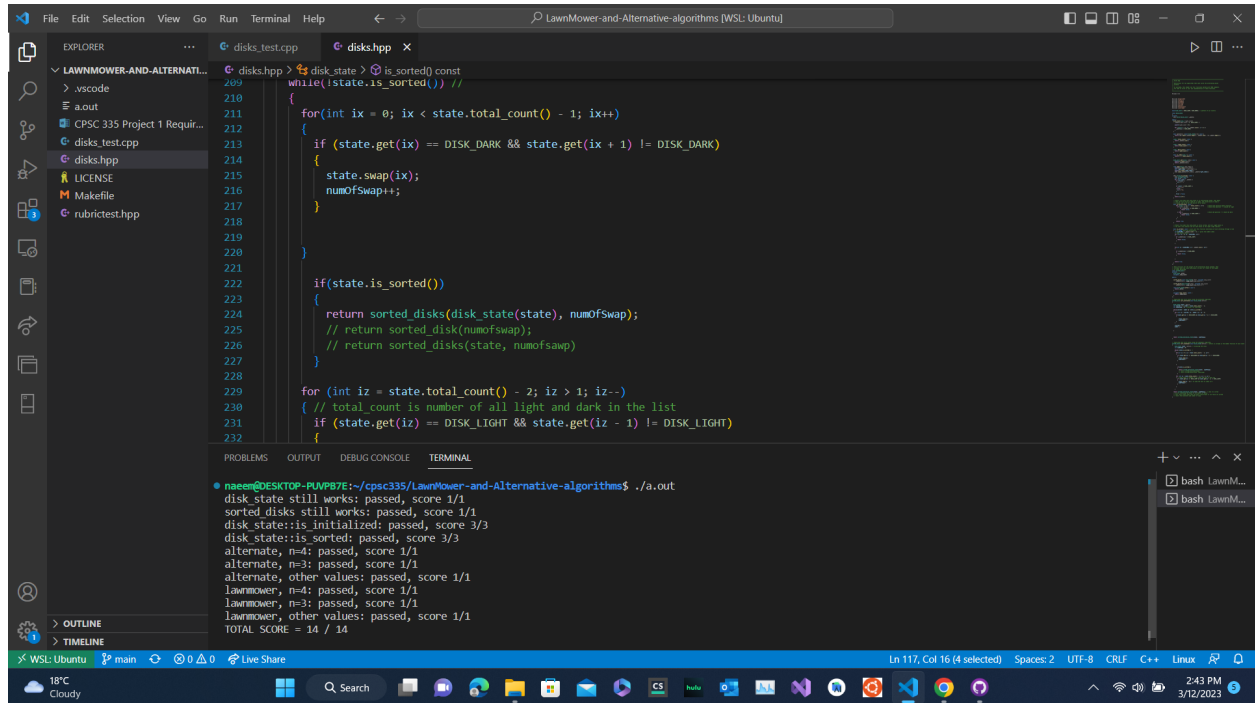
sort_lawnmoer:



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the implementation of the `sort_lawnmoer` algorithm in `disks.hpp`. The code includes a `sorted_disks sort_lawnmoer(const disk_state& before)` function. The `sort_lawnmoer` function uses a while loop to iterate over the disk state, swapping adjacent disks that are not in the correct order (DISK_DARK followed by DISK_LIGHT) and counting the number of swaps. It also includes a for loop to swap the next to index `iz-2`.

```
203 // Algorithm that sorts disks using the lawnmoer algorithm.
204 sorted_disks sort_lawnmoer(const disk_state& before) // before is already in the member function of disk state class
205 {
206     disk_state state = before; // accessing the class
207     int numOfSwap = 0;
208
209     while(!state.is_sorted()) //
210     {
211         for(int ix = 0; ix < state.total_count() - 1; ix++)
212         {
213             if (state.get(ix) == DISK_DARK && state.get(ix + 1) != DISK_DARK)
214             {
215                 state.swap(ix);
216                 numOfSwap++;
217             }
218         }
219
220         if(state.is_sorted())
221         {
222             return sorted_disks(disk_state(state), numOfSwap);
223             // return sorted_disk(numofswap);
224             // return sorted_disks(state, numofswap)
225         }
226
227         for (int iz = state.total_count() - 2; iz > 1; iz--)
228         {
229             // total_count is number of all light and dark in the list
230             if (state.get(iz) == DISK_LIGHT && state.get(iz - 1) != DISK_LIGHT)
231             {
232                 state.swap(iz - 1); // to swap the next to index iz-2
233                 numOfSwap++;
234             }
235         }
236     }
237
238     return sorted_disks(state, numOfSwap);
239 }
240 }
```

output:



The screenshot shows a Visual Studio Code editor window with a C++ project named "LawnMower-and-Alternative-algorithms". The file explorer on the left shows the project structure, including files like `disks_test.cpp`, `disks.hpp`, `Makefile`, and `rubric_test.hpp`. The main editor displays the `disks.hpp` file, which contains a C++ implementation of a sorting algorithm. The code includes a `disk_state` struct, a `sorted_disks` function, and a `main` function that tests the algorithm. The terminal at the bottom shows the output of the program, which includes the results of various tests and a final score of 14 / 14.

```
209 while(!state.is_sorted()) const
210 {
211     for(int ix = 0; ix < state.total_count() - 1; ix++)
212     {
213         if (state.get(ix) == DISK_DARK && state.get(ix + 1) != DISK_DARK)
214         {
215             state.swap(ix);
216             numofswap++;
217         }
218     }
219 }
220
221 if(state.is_sorted())
222 {
223     return sorted_disks(disk_state(state), numofswap);
224     // return sorted_disk(numofswap);
225     // return sorted_disks(state, numofswap)
226 }
227
228 for (int iz = state.total_count() - 2; iz > 1; iz--)
229 {
230     // total_count is number of all light and dark in the list
231     if (state.get(iz) == DISK_LIGHT && state.get(iz - 1) != DISK_LIGHT)
232     {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
name@DESKTOP-PUMP87E:~/cp335/LawnMower-and-Alternative-algorithms$ ./a.out
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
```

sort_alternate: pseudocode, time unit, time complexity, and limits

project 1 submission

#SortAlternate : pseudocode

set start-at to first disk
set end-at to last-disk-1

while start-at less than end-at
for every other disk in the list from start-at to end-at
if disk isn't identical to adjacent disk (and its DARK)
call swap function
end if

end for
increment start-at
decrement end-at

end while

Step count:

$$SC = \sum_{i=0}^n \sum_{j=0}^n 6 + \max(3, 0)$$

inner loop:

$$\sum_{j=0}^n 9 \quad \text{constant series}$$
$$\sum_{j=1}^n 9n$$

outer loop

$$\sum_{i=0}^n 9n \Rightarrow \sum_{i=1}^n 9n \quad \text{constant series}$$

$$SC = 9n^2 \Rightarrow \text{time complexity is } O(n^2)$$

- test the efficiency to see if the $9n^2 \in O(n^2)$:

• Using limit theorem

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \quad \frac{9n^2}{n^2}$$

$$\Rightarrow \lim_{n \rightarrow \infty} \frac{9n^2}{n^2}$$

$$= 9 \Rightarrow f(n) \gg 0 \text{ therefore } f(n) \in g(n)$$

sort_lawnmoer: pseudocode, time unit, time complexity, and limits

Sort-lawnmoer: pseudo code

while not isSorted

for each disk from left to right

if disk is black and adjacent[+1] disk isn't black

call swap function

end if

end for

if isSorted

return

for each disk from right to left

if disk is white and adjacent[-1] disk isn't white

call swap function

end if

end for

end while

return

Step count: WL: while loop, WLB: while loop block

$$SC = 2 + SC_{WL} (n * (SC_{WLB}))$$

$$SC_{WLB} = ((n-1) - 0 + 1) * (6 + \max(2, 0))$$

$$= n(8) \Rightarrow SC_{WLB} = 8n$$

∵ WLB1 is the first for loop

$$SC_{WLB2} = \left(1 - \frac{(n-2)}{-1} + 1\right) \times (6 + \max(3, 0))$$

$$= 2 + (1 - (-n+2) + 1) \times (6+3)$$

$$= (2 + 1 + n - 2 + 1) \times 9$$

$$= \cancel{11}n(2+n) \times 9$$

$$SC_{WLB2} = 18 + 9n$$

WLB2 is the second for loop block

↑ entire lawn mow function

$$SC = n + 2 + n \times (18 + 9n + 8n)$$

$$= n + 2 + n \times (17n + 18)$$

$$SC = n + 2 + 17n^2 + 18n$$

time complexity is highest power because the other terms are lower power which won't affect the n^2 so time complexity is $O(n^2)$

- test efficiency to see that $17n^2 + 19n + 2 \in O(n^2)$

- Using limit theorem:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{17n^2 + 19n + 2}{n^2}$$

$$= \lim_{n \rightarrow \infty} \left(\frac{17n^2}{n^2} + \frac{19n}{n^2} + \frac{2}{n^2} \right)$$

$$\lim_{n \rightarrow \infty} \left(17 + \frac{19}{n} + \frac{2}{n^2} \right)$$

is_sort: pseudocode, time unit, time complexity

$$= 17 + \frac{19}{\infty} + \frac{2}{\infty}$$

$$= 17 + 0 + 0$$

$$= 17$$

because $f(n) \stackrel{=17}{\geq} 0$ such that $f(n) \in g(n)$

is_Sorted: pseudocode

~~for disk is~~
for first half of disks
if disk is not light
fail

for second half of disks
if disk is not dark
fail

return success

- Step count:

$$SC = 2 + (SC_{FL1}) + (SC_{FL2}) : FL1, FL2 \text{ are first and second for loop in is_Sorted}$$

$$SC_{FL1} = (\# \text{iterations}) * SC_{FLB1}^{\text{for loop block}}$$

$$\# \text{iterations} = \frac{n}{2} - 0 + 1$$

$$= \frac{n}{2} + 1$$

$$SC_{FL1} = \left(\frac{n}{2} + 1\right) * SC_{FLB1}$$

$$SC_{FLB1} = 1 + \text{Max}(0, 0)$$

$$SC_{FLB1} = 1$$

$$SC_{FL1} = \left(\frac{n}{2} + 1\right) \times 1$$

$$\textcircled{1} \quad SC_{FL1} = \frac{n}{2} + 1$$

$$SC_{FL2} = (\text{\#iterations}) \times (SC_{FLB2})$$

$$\text{\#iterations} = n - \frac{n}{2} + 1$$

$$= \frac{n}{2} + 1$$

$$SC_{FLB2} = 1 + \text{Max}(0, 0)$$

$$= 1$$

$$SC_{FL2} = \left(\frac{n}{2} + 1\right) \times 1$$

$$\textcircled{2} \quad SC_{FL2} = \frac{n}{2} + 1$$

Now, the SC of the entire is-sorted function:

$$SC = 2 + \left(\frac{n}{2} + 1\right) + \left(\frac{n}{2} + 1\right)$$

$$= 4 + \frac{n}{2} + \frac{n}{2}$$

$$SC = 4 + n$$

time complexity is $O(n)$