Project 1/Lawnmower Project Report

Group Members:

Ranny Khant Naing

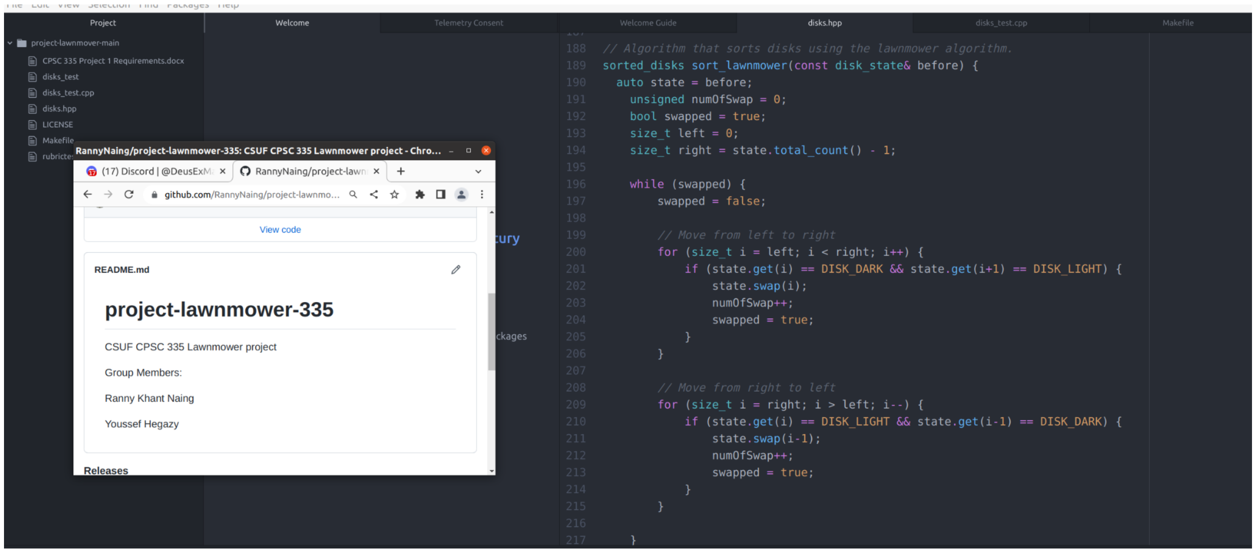
Youssef Hegazy

CSUF Emails:

khant_naing@csu.fullerton.edu

YoussefH@csu.fullerton.edu

This is a submission for Project 1.



project-lawnmower-335

CSUF CPSC 335 Lawnmower project

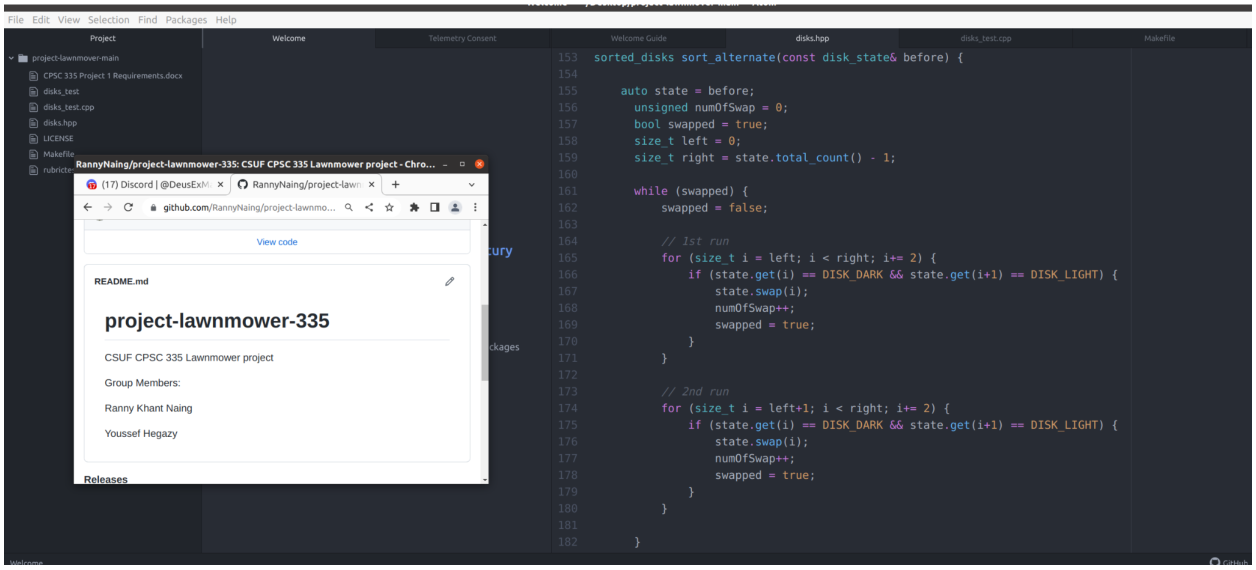
Group Members:

Ranny Khant Naing

Youssef Hegazy

```cpp
// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state& before) {
  auto state = before;
    unsigned numOfSwap = 0;
    bool swapped = true;
    size_t left = 0;
    size_t right = state.total_count() - 1;

    while (swapped) {
        swapped = false;

        // Move from left to right
        for (size_t i = left; i < right; i++) {
            if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
                state.swap(i);
                numOfSwap++;
                swapped = true;
            }
        }

        // Move from right to left
        for (size_t i = right; i > left; i--) {
            if (state.get(i) == DISK_LIGHT && state.get(i-1) == DISK_DARK) {
                state.swap(i-1);
                numOfSwap++;
                swapped = true;
            }
        }

    }
```

Screenshot for Lawnmower algorithm



project-lawnmower-335

CSUF CPSC 335 Lawnmower project

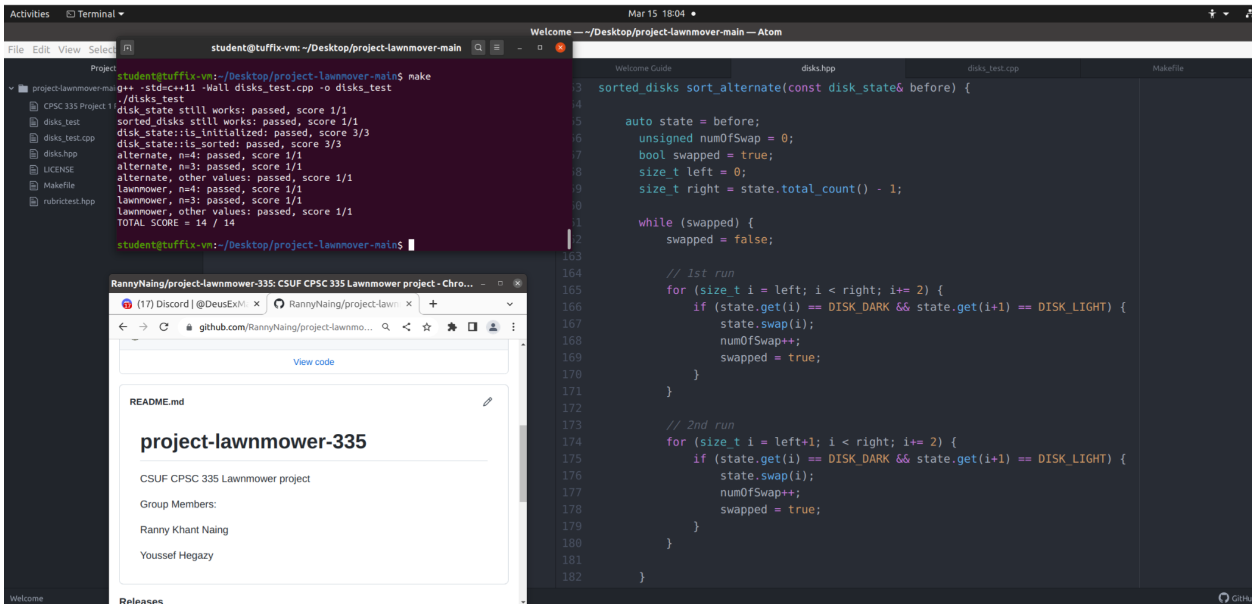
Group Members:

Ranny Khant Naing

Youssef Hegazy

```cpp
sorted_disks sort_alternate(const disk_state& before) {

    auto state = before;
      unsigned numOfSwap = 0;
      bool swapped = true;
      size_t left = 0;
      size_t right = state.total_count() - 1;

      while (swapped) {
          swapped = false;

          // 1st run
          for (size_t i = left; i < right; i+= 2) {
              if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
                  state.swap(i);
                  numOfSwap++;
                  swapped = true;
              }
          }

          // 2nd run
          for (size_t i = left+1; i < right; i+= 2) {
              if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
                  state.swap(i);
                  numOfSwap++;
                  swapped = true;
              }
          }

      }
```

Screenshot for Alternate algorithm



```
student@tuffix-vm:~/Desktop/project-lawnmover-main$ 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:~/Desktop/project-lawnmover-main$ 
```

```
sorted_disks sort_alternate(const disk_state& before) {

  auto state = before;
   unsigned numOfSwap = 0;
   bool swapped = true;
   size_t left = 0;
   size_t right = state.total_count() - 1;

   while (swapped) {
     swapped = false;

     // 1st run
     for (size_t i = left; i < right; i+= 2) {
       if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
         state.swap(i);
         numOfSwap++;
         swapped = true;
       }
     }

     // 2nd run
     for (size_t i = left+1; i < right; i+= 2) {
       if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
         state.swap(i);
         numOfSwap++;
         swapped = true;
       }
     }

   }
```

README.md

project-lawnmower-335

CSUF CPSC 335 Lawnmower project

Group Members:

Ranny Khant Naing

Youssef Hegazy

Screenshot for the compiling

Pseudo code for Lawnmower algorithm

Def sort_lawnmower(before):	Step Counts
state = before;	1
numOfSwap = 0;	1
swapped = true;	1
left = 0;	1
right = total_count()-1;	2
while (swapped):	n/2

```
            swapped = false;                                        1
            for(i = left; i < right; ++i):                          n
                if(state[i] == dark && state[i+1} == light):        3
                    Swap[i];                                        1
                    numOfSwap++;                                    1
                    swapped = true;                                 1
                end if
            end for
            for(i = right; i > left; --i):                          n
                if(state[i] == light && state[i+1} == dark):        3
                    swap[i];                                        1
                    numOfSwap++;                                    1
                    swapped = true;                                 1
                end if
            end for
        end while
    return disk_state(state,numOfSwap);
```

Total Step Count = 1+1+1+1+ 2+ n/2*(1+n*(3+max(3, 0))+n*(3+max(3 0)))

= 6+n/2*(1+6n+6n)

= 6+n/2*(1+12n)

=6n^2 + 6 + n/2

The Lawnmower algorithm has an efficiency of O(n^2).

Time Complexity for Lawnmower algorithm

6n^2 + 6 + n/2 ∈ O(n^2)

By Def

6n^2 + 6 + n/2 ≤ c.(n^2)

choose ,c = 12, n0 = 2

6 * 2^2 + 6 + 2/2 ≤ 12 * 2^2

31 ≤ 48

∴ 6n^2 + 6 + n/2 ∈ O(n^2)

Implementations:

```
/ Algorithm that sorts disks using the lawnmower algorithm.

sorted_disks sort_lawnmower(const disk_state& before) {

  auto state = before;

    unsigned numOfSwap = 0;

    bool swapped = true;

    size_t left = 0;

    size_t right = state.total_count() - 1;
```

```
while (swapped) {
    swapped = false;

    // Move from left to right
    for (size_t i = left; i < right; i++) {
        if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {
            state.swap(i);
            numOfSwap++;
            swapped = true;
        }
    }

    // Move from right to left
    for (size_t i = right; i > left; i--) {
        if (state.get(i) == DISK_LIGHT && state.get(i-1) == DISK_DARK) {
            state.swap(i-1);
            numOfSwap++;
            swapped = true;
        }
    }

}
```

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

—---

Alternate Algorithm

Pseudo code for Alternate algorithm

```
Def sort_lawnmower(before):                          Step Counts
          state = before;                                  1
          numOfSwap = 0;                                   1
          swapped = true;                                  1
          left = 0;                                        1
          right = total_count()-1;                         2
```

```
    while (swapped):                                                n+1
        swapped = false;                                            1
        for(i = left; i < right; i += 2):                           n/2
            if(state[i] == dark && state[i+1} == light):            3
                Swap[i];                                            1
                numOfSwap++;                                        1
                swapped = true;                                     1
            end if
        end for
        for(i = left +1 ; i < right; i += 2):                       n/2
            if(state[i] == dark && state[i+1} == light):            3
                Swap[i];                                            1
                numOfSwap++;                                        1
                swapped = true;                                     1
            end if
        end for
    end while
return disk_state(state,numOfSwap);
```

Total Step Count = 1+1+1+1+ 2+ (n+1)*(1+n/2*(3+max(3, 0))+n/2*(3+max(3 0)))

= 6+(n+1)*(1+3n+3n)

= 6+(n+1)*(1+6n)

=6+ n + 6n^2 + 1 + 6n

=6n^2 + 7n + 7

The alternate algorithm has an efficiency of O(n^2).

Time Complexity for Alternate algorithm

6n^2 + 7n + 7 ∈ O(n^2)

By Def

6n^2 + 7n + 7 ≤ c.(n^2)

choose ,c = 20, n0 = 2

6 * 2^2 + 7 * 2 + 7 ≤ 20 * 2^2

45 ≤ 80

∴ 6n^2 + 7n + 7 ∈ O(n^2)

Implementations:

```
sorted_disks sort_alternate(const disk_state& before) {


  auto state = before;

   unsigned numOfSwap = 0;

   bool swapped = true;
```

```
size_t left = 0;

size_t right = state.total_count() - 1;


while (swapped) {

    swapped = false;


    // 1st run

    for (size_t i = left; i < right; i+= 2) {

        if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {

            state.swap(i);

            numOfSwap++;

            swapped = true;

        }

    }


    // 2nd run

    for (size_t i = left+1; i < right; i+= 2) {

        if (state.get(i) == DISK_DARK && state.get(i+1) == DISK_LIGHT) {

            state.swap(i);

            numOfSwap++;

            swapped = true;

        }
        }


    }


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