Project 1: implementing algorithms

CPSC 335 - Algorithm Engineering
Spring 2020
Instructors: Doina Bein (dbein@fullerton.edu)

Pseudo Code:

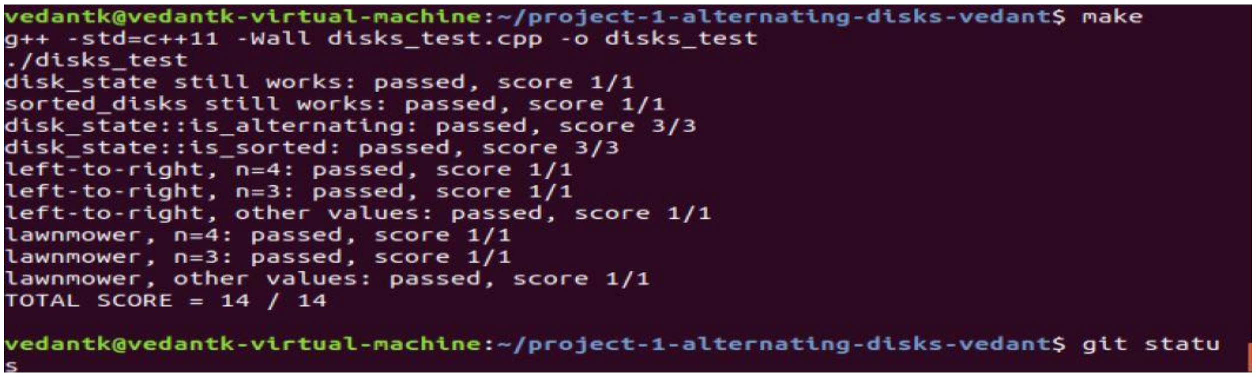
sorted_disks sort_left_to_right

```
1.Start
2.Create first loop for number of transverse in the list(count of transverse depend on number of values in list)
3. Use second loop inside the first loop for swap of value which are not in ordered.(adjuecent swapping only)
.transverse only in one direction
4.Repeat the step 2 until it complete transverse count.
5. Return the ordered list.
6. Stop
```

sort_lawnmower

```
1.Start
2.Create first loop for number of transverse in the list(count of transverse depend on number of values in list)
3. Use second loop inside the first loop for swap of value which are not in ordered.(adjuecent swapping only)
4. Use third loop inside the first loop, same as second loop but transverse in reverse direction as of second
loop.(adjuecent swapping only)
5.Repeat the step 2 until it complete transverse count.
6. Return the ordered list.
7. Stop
```

Screenshot of the result:



```
vedantk@vedantk-virtual-machine:~/project-1-alternating-disks-vedant$ 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_alternating: passed, score 3/3
disk_state::is_sorted: passed, score 3/3
left-to-right, n=4: passed, score 1/1
left-to-right, n=3: passed, score 1/1
left-to-right, 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

vedantk@vedantk-virtual-machine:~/project-1-alternating-disks-vedant$ git statu
s
```

Mathematical analysis

Get Function:

```
disk_color get(size_t index) const {
   assert(is_index(index));                                          ----O(1)
   return _colors[index];                                            ----O(1)
 }
                                              Total:                 ----O(2)
```

Swap Function:

```
void swap(size_t left_index) {
   assert(is_index(left_index));                                         ----O(1)
   auto right_index = left_index + 1;                                    ----O(2)
   assert(is_index(right_index));                                        ----O(1)
   std::swap(_colors[left_index], _colors[right_index]);                 ----O(1)
 }
                                              Total:                     ----O(5)
```

Check Alternating:

```
 bool is_alternating() const {
        for(std::vector<disk_color>::size_type i = 0; i < _colors.size(); i+=2) {   ----O(n/2)
                if(_colors[i]!=0 || _colors[i+1]!=1)                                -----O(5)
                        {
                                return false;                                       -----O(1)
                        }
                }
        return true;                                                                -----O(1)
 }
                                         Total:                                     -----5n/2 +1
```

Sorting

Sort left to right

```
sorted_disks sort_left_to_right(const disk_state& before) {
 assert(before.is_alternating());                                       ----O(5n/2 +1)
int swapCount=0;                                                        ----O(1)
disk_state disks(before);                                               ----O(1)
for(std::vector<disk_color>::size_type j = 0; j < disks.total_count()-1; j++)    ---O(n)==(12n^2)
{
for(std::vector<disk_color>::size_type i = j; i < disks.total_count()-1; i++) {  ----O(n)
        if(disks.get(i)>disks.get(i+1))                                 ----O(6)
```

```
                    {
                              disks.swap(i);                                                ----O(5)
                              swapCount++;                                                  ----O(1)

                    }
          }
}
  return sorted_disks(disks, swapCount);                                                   ----O(1)
5n/2+1+2+12n^2
}
```

Total: O(3+12n^2+2.5n))

<u>Sort_lawnmower</u>

```
sorted_disks sort_lawnmower(const disk_state& before) {
assert(before.is_alternating());                                                      ----O(5n/2 +1)
int swapCount=0;                                                                              --O(1)
disk_state disks(before);                                                                     --O(1)
for(std::vector<disk_color>::size_type j = 0; j < disks.total_count()-1; j+=2)               -O(n/2)
{
          for(std::vector<disk_color>::size_type i = j; i < disks.total_count()-1; i++)   {       -O(n)
                    if(disks.get(i)>disks.get(i+1))                                           --O(7)
                              {
                                          disks.swap(i);                                      --O(5)
                                        swapCount++;                                          --O(1)
                              }
                    }
          for(std::vector<disk_color>::size_type i = disks.total_count()-2; i>j; i--)        --O(n)
                    {
                    if(disks.get(i)>disks.get(i+1))                                           --O(7)
                              {
                                          disks.swap(i);                                      ---O(5)
                                        swapCount++;                                          ---O(1)
                              }
                    }
}
  return sorted_disks(disks, swapCount);                                                   ----O(1)
}
```

1+1+n/2(n*(7+6) + n*(7+6))+1=4+13n^2+5n/2   **Total: O(4+13n^2+5n/2)**