CPSC 335 Project 1

ALTERNATE ALGORITHM PSEUDOCODE:

```
function sort_alternate(before: disk_state) -> sorted_disks
temp = before
swapCount = 0

for i = 0 to temp.light_count() - 1 do
   for j = 0 to temp.total_count() - 2 do
      if temp.get(j) > temp.get(j + 1) then
         temp.swap(j)
         swapCount = swapCount + 1
      end if
   end for
end for
return sorted_disks(disk_state(temp), swapCount)
end function
```

ALTERNATE ALGORITHM TIME COMPLEXITY:

In the given algorithm, the outer loop iterates n/2 times (n is the total count of the disks), and the inner loop iterates n-1 times. Each iteration of the inner loop performs constant time operations, so the total number of operations performed by the algorithm is (n/2) * (n-1), which simplifies to (n^2 - n) / 2. We can drop the lower term (-n) and the constant factor (1/2), and say the time complexity of the given algorithm is O(n^2). Therefore, we can conclude that the time complexity of the given algorithm is O(n^2).

ALTERNATE ALGORITHM TIME STEP COUNT:

The Alternate algorithm has a time complexity of O(n^2) because it uses nested loops to iterate over all pairs of disks and swaps them if they are out of order. The outer loop iterates n times, where n is the number of light disks in the input, and the inner loop iterates n-1 times, where n is the total number of disks in the input. Therefore, the step count for this implementation is roughly proportional to n^2.

LAWNMOWER ALGORITHM PSEUDOCODE:

```
function sort_lawnmower(before: disk_state) -> sorted_disks
temp = before
swapCount = 0
isIncrementing = true
```

```
for i = 0 to temp.light_count() - 1 do
   if i % 2 == 0 then
      isIncrementing = true
   else
      isIncrementing = false

   if isIncrementing then
      start = 0
      end = temp.total_count() - 2
      increment = 1
   else
      start = temp.total_count() - 2
      end = 0
      increment = -1

   for j = start to end step increment do
      if temp.get(j) > temp.get(j + 1) then
         temp.swap(j)
         swapCount = swapCount + 1
      end if
   end for
end for
return sorted_disks(disk_state(temp), swapCount)
end function
```

LAWNMOWER ALGORITHM TIME COMPLEXITY:

The given algorithm is an implementation of the Lawnmower sorting algorithm. The given algorithm iterates the outer loop n/2 times (n is the total count of the disks), and the inner loop also iterates n/2 times, resulting in a total of n^2/4 iterations. Since each iteration of the inner loop performs constant time operations (comparisons, swaps, and increments), the time complexity of the algorithm is O(n^2). Therefore, we can conclude that the time complexity of the given algorithm is O(n^2).

LAWNMOWER ALGORITHM TIME STEP COUNT:

The outer loop iterates n/2 times. The inner loop iterates n-1 times for each outer loop iteration, where n is the number of total disks in the input. Therefore, the total number of iterations is n^2/2.

OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login

Run Debug Stop Share Save { } Beautify   Language C++

main.cpp   rubrictest.hpp   disks.hpp

```cpp
///////////////////////////////////////////////////////////////////////////////
//GROUP MEMBER:
//WILLIAM LUA
// disks.hpp
//
// Definitions for two algorithms that each solve the alternating disks
// problem.
//
// As provided, this header has four functions marked with TODO comments.
// You need to write in your own implementation of these functions.
//
///////////////////////////////////////////////////////////////////////////////

#pragma once



#include <algorithm>
#include <cassert>
#include <cstddef>
#include <sstream>
#include <string>
#include <vector>
#include <functional>
#include <iostream>

enum disk_color { DISK_LIGHT, DISK_DARK};

class disk_state {
private:
  std::vector<disk_color> _colors;

public:
  disk_state(size_t light_count)
    : _colors(light_count * 2, DISK_LIGHT) {

    assert(light_count > 0);

    for (size_t i = 1; i < _colors.size(); i += 2) {
      _colors[i] = DISK_DARK;
    }
  }
```

input

```
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


...Program finished with exit code 0
Press ENTER to exit console.
```