Cal State Fullerton

College of Engineering and Computer Science

Project 1:

Implementing Algorithm for Sorting Disks

Student: Holly Thuy Do

Class: CPSC 335-11

CWID: 884916222

Date: 02/17/2024

Introduction

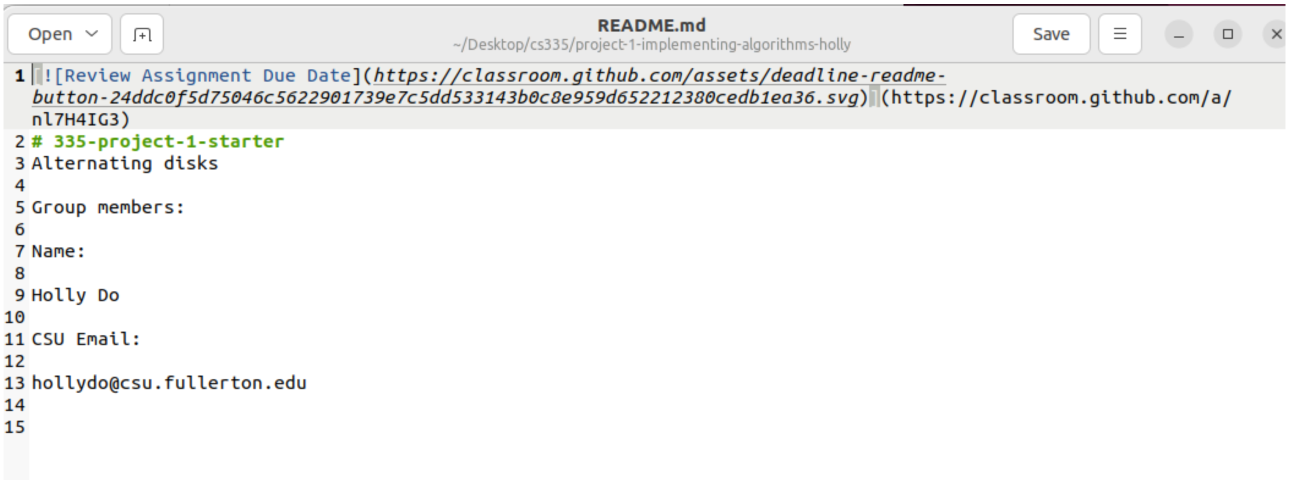
In this first assignment for CS335 Algorithm class, we will implement two ways of sorting an alternating disks vector.

The definition of the problem is:

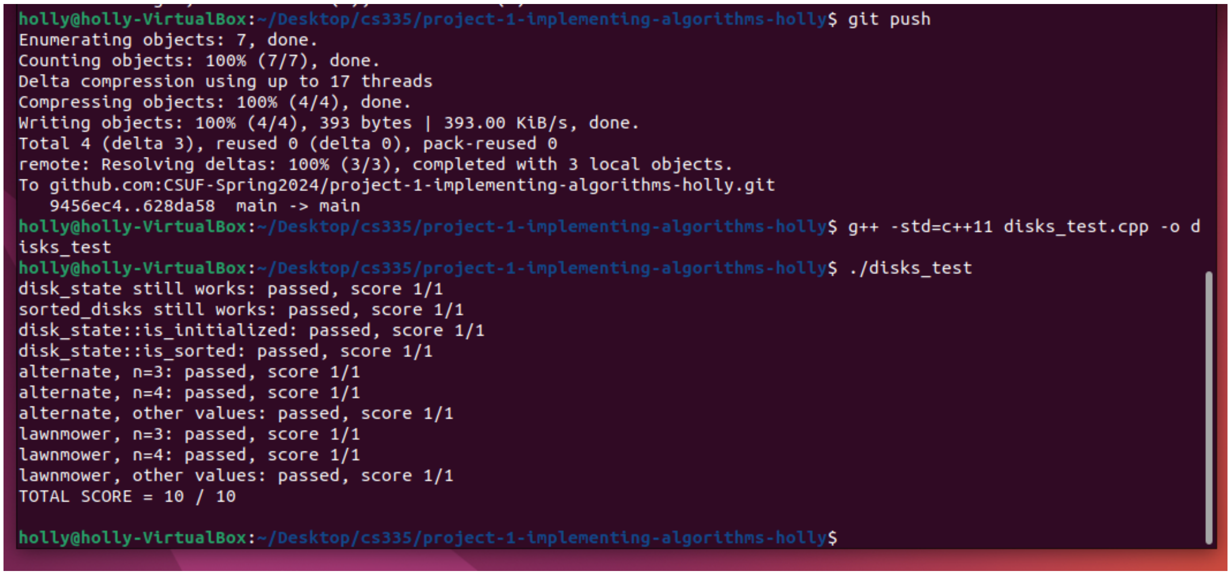
Input: a list of alternating light and dark disks size of 2n, with n is a positive integer.

Output: sorted vector with dark disks on the left and light disks on the right.

Screenshots



```
![Review Assignment Due Date](https://classroom.github.com/assets/deadline-readme-button-24ddc0f5d75046c5622901739e7c5dd533143b0c8e959d652212380cedb1ea36.svg)](https://classroom.github.com/a/nl7H4IG3)
# 335-project-1-starter
Alternating disks

Group members:

Name:

Holly Do

CSU Email:

hollydo@csu.fullerton.edu


```



```
holly@holly-VirtualBox:~/Desktop/cs335/project-1-implementing-algorithms-holly$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 17 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 393 bytes | 393.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:CSUF-Spring2024/project-1-implementing-algorithms-holly.git
   9456ec4..628da58  main -> main
holly@holly-VirtualBox:~/Desktop/cs335/project-1-implementing-algorithms-holly$ g++ -std=c++11 disks_test.cpp -o d
isks_test
holly@holly-VirtualBox:~/Desktop/cs335/project-1-implementing-algorithms-holly$ ./disks_test
disk_state still works: passed, score 1/1
sorted_disks still works: passed, score 1/1
disk_state::is_initialized: passed, score 1/1
disk_state::is_sorted: passed, score 1/1
alternate, n=3: passed, score 1/1
alternate, n=4: passed, score 1/1
alternate, other values: passed, score 1/1
lawnmower, n=3: passed, score 1/1
lawnmower, n=4: passed, score 1/1
lawnmower, other values: passed, score 1/1
TOTAL SCORE = 10 / 10

holly@holly-VirtualBox:~/Desktop/cs335/project-1-implementing-algorithms-holly$
```

Pseudo-Code

1/

```
sort_lawnmower (disk_state& before)


disk_state after = before; // 1
count = 0; //1
for j=0 to (count()/2) -1 do // (count()/2) -1 +1
    for i=j to (count()-1)-j-1 jump 2 steps forward do //          ((count()-1)-j-1 –j) / 2 +1
        if (after[i]== DISK_LIGHT && after[i+1] == DISK_DARK) // 1+1+1 = 3
        {
            swap; //1
            count++; //1
        }
        else do nothing //0
        endif
    for i = count()-1-j to 1+j-1 jump 2 steps backward do // (1+j-1 - (count()-1-j)) /2 +1
        if (after[i]== DISK_LIGHT && after[i+1] == DISK_DARK) //3
        {
```

```
swap; //1
count++; //1
}
else do nothing
endif
endfor
endfor

 return after;
```

2/

```
sort_alternate (disk_state& before)

disk_state after = before; //1
count = 0; //1
for j=0 to (count() -2) do // (count() -2) +1
for i=j to count()-2) jump 2 steps forward do // ((count() -2) - j ) / 2 +1
if (after[i]== DISK_LIGHT && after[i+1] == DISK_DARK) //3
{
swap; //1
count++; //1
}
else do nothing
endif
endfor
endfor
return after
```

Big O Analysis

The input disks arrangement starts with a white disk and then a black disk and so on until the end of the vector. The length of this array/vector is always even because we started with an integer n number of white disks then add n number of black disks. For the algorithms to complete the entire length of the vector, we need to make sure to set the iteration numbers correctly. For example, the sort_alternate() function needs to be repeated n/2 times, which means the last pair will be at indexes (n/2)-1 and n/2. Similarly, the sort_lawnmower() swapping decision needs to be iterated until the last pair of indexes (n/2)-1 and n/2. Below is the Big O analysis of the runtime for both functions.

sort_lawnmower()

1/ if $s.c = 3 + \max(2,0) = 5$

inner loop $\neq 1$: $sc = 5\left(\frac{c-1-j-1-j}{2} + 1\right) = \left(\frac{c}{2} - j\right) \cdot 5$

second inner for loop: $sc = 5\left(\frac{(1+j-1)-(c-1)-j}{-2} + 1\right) = \left(\frac{c}{2} - j + 1\right) 5$.

Total $s.c = 1 + 1 + \sum_{j=0}^{\frac{c}{2}-1} 5\left(\frac{c}{2} - j\right) + \sum_{j=0}^{\frac{c}{2}-1} 5\left(\frac{c}{2} - j + 1\right)$

$= 2 + \cancel{3} \sum_{j=0}^{\frac{c}{2}-1} 5\left(\frac{c}{2} - j + \frac{c}{2} - j + 1\right) = 2 + \sum_{j=0}^{\frac{c}{2}-1} 5(c - 2j + 1)$

$= 2 + 5\left(\frac{c}{2} \mp 1 + 1\right) - 10 \frac{\left(\frac{c}{2} - 1\right)\left(\frac{c}{2}\right)}{2} + 5c\left(\frac{c}{2} - 1 + 1\right)$

$= 2 + 5\frac{c}{2} - 5\left(\frac{c^2}{4} - \frac{c}{2}\right) + 5c\left(\frac{c}{2}\right)$

$= 2 + \frac{5c}{2} - \frac{5}{4}c^2 + \frac{5}{2}c + \frac{5c^2}{2} = 2 + 5c + \frac{5}{4}c^2$

This belongs to big O of n^2 because largest term is n^2 (or c^2)

sort_alternate()

2) if s.c $= 3 + \max(2,0) = 5$

inner for loop s.c $= 5 \cdot \left[\left(\frac{c-2}{2} - j\right) + 1\right] = 5\left(\frac{c-j}{2}\right)$

Total s.c $= 1 + 1 + \sum_{j=0}^{c-2} \frac{5}{2} c - \sum_{j=0}^{c-2} \frac{5}{2} j$

$= 2 + \frac{5}{2} c . (c - 2 + 1) - \frac{5}{2} \frac{(c-2)(c-1)}{2} \cancel{= 2 + \left(\frac{5c}{2}\right.}$

$= 2 + \frac{5}{2} c (c-1) - \frac{5}{4} (c^2 - c - 2c + 2)$

$= 2 + \frac{5}{2} c^2 - \frac{5}{2} - \frac{5}{4} (c^2 - 3c + 2)$

$= 2 + \frac{5}{2} c^2 - \frac{5}{2} - \frac{5}{4} c^2 + \frac{15}{4} c - \frac{10}{4}$

$= \frac{5}{4} c^2 + \frac{15}{4} c - 5 + 2 = \frac{5}{4} c^2 + \frac{15}{4} c - 3$

This belongs to big O of n^2 (or c^2)