CPSC 335 Project 1 - The Alternating Disk Problem

Problem Statement:

You have a row of $\boldsymbol{2n}$ disks of two colors, $\boldsymbol{n}$ light and $\boldsymbol{n}$ dark. They alternate: light, dark, light, dark, and so on. You want to get all the dark disks to the left hand side and all the light disks to the right hand end. The only moves you are allowed to make are those that interchange the positions of two neighboring disks. Design an algorithm for solving this puzzle and determine the number of moves it takes.

Input: a positive integer n and a list of $2n$ disks of alternating colors light-dark, starting with light

Output: a list of $2n$ disks, the first n disks are dark, the next n disks are light, and an integer m representing the number of swaps to move the light ones after the dark ones

Pseudocode:

Step 1: Enter the input(n) that is, enter the number of the single color disks (dark or light)

Store the input in the form of array starting with light disks

The first element in the array represents ’l’, which means it is a light disk, the second element represents ‘d’, which means it is a dark disk, similarly for all the positions till it reaches the 2*n times, end of the array.

```
for (int iterator = 0; iterator < n; iterator++) {
      diskRepresentation[2 * iterator] = 'l';
      diskRepresentation[2 * iterator + 1] = 'd';
  }
```

Step 2: Write a nested for loop which iterates among the disk array

Check the element values, if there is a combination of: first element as ‘l’ and next element as ‘d’, swap them, increase the swap count

Run the loop till it reaches the end of the array.

Step 3: Write one more nested for loop and start looking for the combination of ‘d’ and ‘l’ and swap them until it reaches the end of loop. (This condition is for checking the middle elements)

Increase the swap count when there is a swap.

After this step, all the dark disks will be at first and the light disks at the last.

Step 4: Print the disk representation to the user.

Print the swap count, which is the number of moves required to get the desired output.

Big O efficiency class:

I have the following loops in my code:

```
for (int iterator = 0; iterator < n; iterator++) { -----------------→ 1
            diskRepresentation[2 * iterator] = 'l';-----------------→ n
            diskRepresentation[2 * iterator + 1] = 'd';-------------→ n
}
```

O(2n+1) => O(n)

```
for (int iterator = 0; iterator < n; iterator++) { -----------------→ 1
 for (int j = 0; j < 2 * n; j++) {                 ------------------→ 1
// all these statements are going to be executed 3n times
 if (diskRepresentation[j] == 'l' && diskRepresentation[j+1] == 'd'){
     swap(diskRepresentation[j], diskRepresentation[j+1]);
     m++;
   }
 }
}
```

O(1+1+(3n)n) => O(n^2)

```
for (int iterator = 0; iterator < n; iterator++) { -----------------→ 1
 for (int j = 0; j < 2 * n; j++) {  -----------------→ 1
// all these statements are going to be executed 3n times
   if (diskRepresentation[j] == 'l' && diskRepresentation[j + 1] == 'd') {
     swap(diskRepresentation[j], diskRepresentation[j + 1]);
     m++;
   }
 }
 for (int iterator = n - 1; iterator >= 0; iterator--) {-----------------→ 1
// all these statements are going to be executed 3n times
if (diskRepresentation[iterator] == 'd' && diskRepresentation[iterator - 1] == 'l') {
 swap(diskRepresentation[iterator], diskRepresentation[iterator - 1]);
 m++;
}
}
}
```

O(1+1+1+(3n)n+(3n)n)=>O(n^2)

Final Big Oh:

O(n+n^2+n^2) => O(n^2)

Effectively proving your algorithm

Assume, $T(n) = (2n+1) + (2+3n^2) + (3+6n^2)$

$= 9n^2 + 2n + 6$

It seems to resemble $O(n^2)$

Find c, such that : $T(n) \leq c f(n)$

$9n^2 + 2n + 6 \leq c \cdot n^2$

$c \geq \frac{9n^2 + 2n + 6}{n^2}$

since n must be ≥ 1, assume $c = 19$, $n_0 = 1$

For base case, substitute these values.

$c \geq 9 + 2 + 6$

$\Rightarrow c \geq 17 \quad \Rightarrow 19 \geq 17 \Rightarrow$ true , base case holds.

Proving by limits :

$T(n) = 9n^2 + 2n + 6$

$f(n) = n^2$

$$\lim_{n\to\infty} \frac{T(n)}{f(n)} = \lim_{n\to\infty} \frac{9n^2+2n+6}{n^2}$$

$$= 9 + \lim_{n\to\infty}\left(\frac{2}{n}\right) + \lim_{n\to\infty}\left(\frac{6}{n^2}\right)$$

$$= 9 + 0 + 0$$

$$= 9$$

which is non-negative & constant

$\Rightarrow 9n^2 + 2n + 6 \in O(n^2)$

$\Rightarrow T(n) \in O(f(n))$

—— .

Statement on possible Improvement

From Step 2 and 3,

We look for certain combinations like ‘l’ and ‘d’ or ‘d’ and ‘l’ for the consecutive position in the array by running a loop. Once this is found, we swap the contents which results in nested loops.

Instead of this,

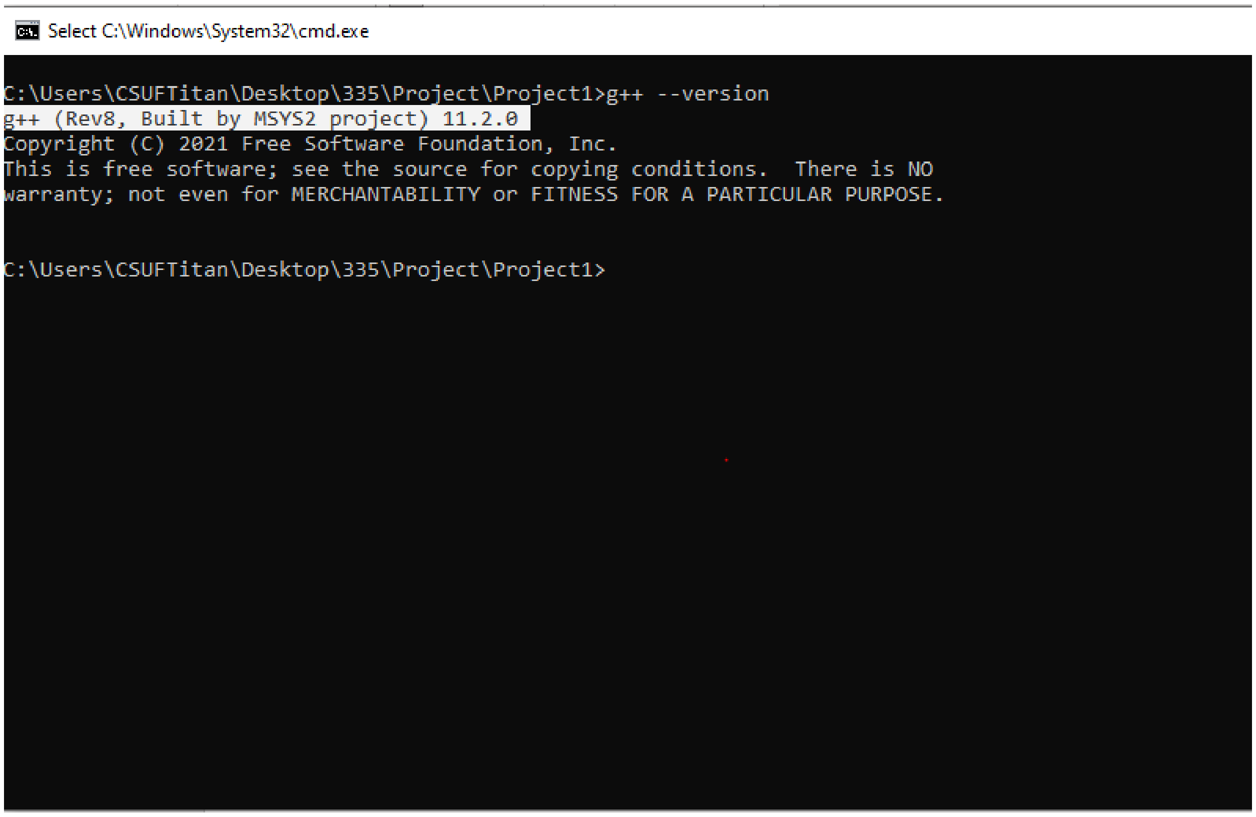
We know that we have to place all the dark disks first, so lets us take a value and check with all the elements previous to it and place it in the right position.

Eg, say we have this representation: d l l d

Whenever we see d, we know that it should be placed at the starting positions, so now the position of 2nd d is 3, so lets check all the previous elements till 2 and if we see any ‘l’ replace it with ‘d’ and continue the loop till the previous value of the array is ‘d’.

This way, it reduces the iterations and also the loops thus have high efficiency.

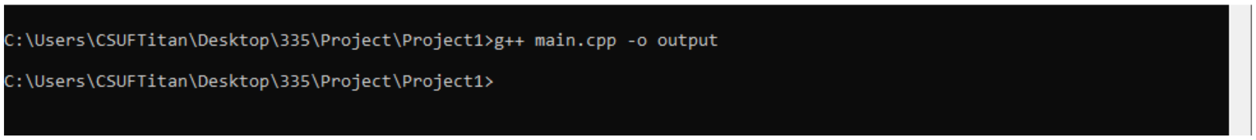
Compile Code:



Select C:\Windows\System32\cmd.exe

```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>g++ --version
g++ (Rev8, Built by MSYS2 project) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


C:\Users\CSUFTitan\Desktop\335\Project\Project1>
```

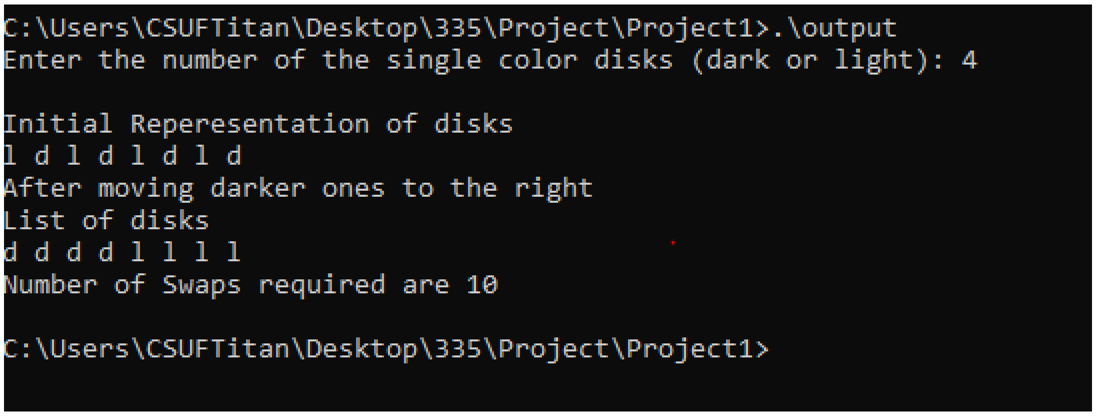


```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>g++ main.cpp -o output

C:\Users\CSUFTitan\Desktop\335\Project\Project1>
```

Compiled Successfully!

Execution:



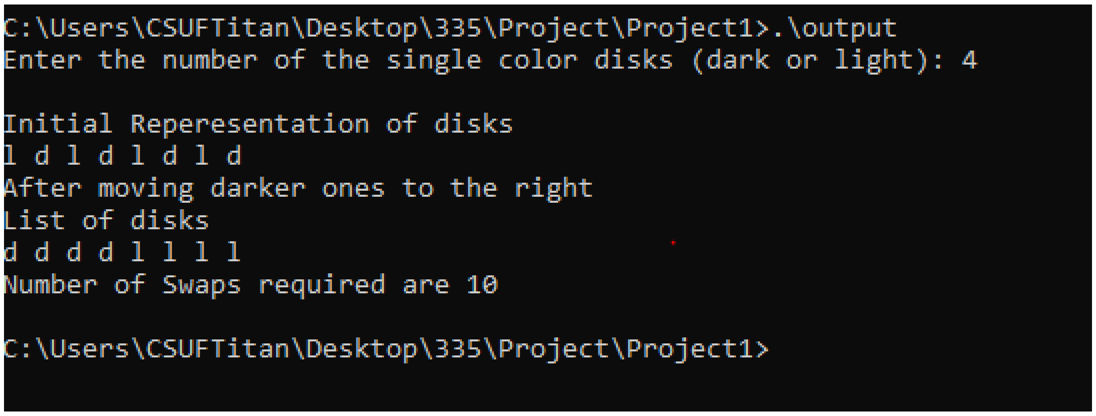
```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>.\output
Enter the number of the single color disks (dark or light): 4

Initial Reperesentation of disks
l d l d l d l d
After moving darker ones to the right
List of disks
d d d d l l l l
Number of Swaps required are 10

C:\Users\CSUFTitan\Desktop\335\Project\Project1>
```

Test Case 1:

Input is 4



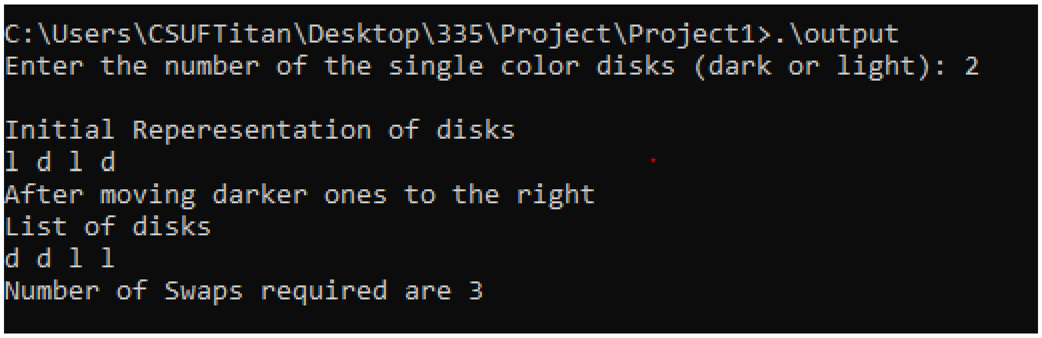
```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>.\output
Enter the number of the single color disks (dark or light): 4

Initial Reperesentation of disks
l d l d l d l d
After moving darker ones to the right
List of disks
d d d d l l l l
Number of Swaps required are 10

C:\Users\CSUFTitan\Desktop\335\Project\Project1>
```

Test Case 2:

Input is 2

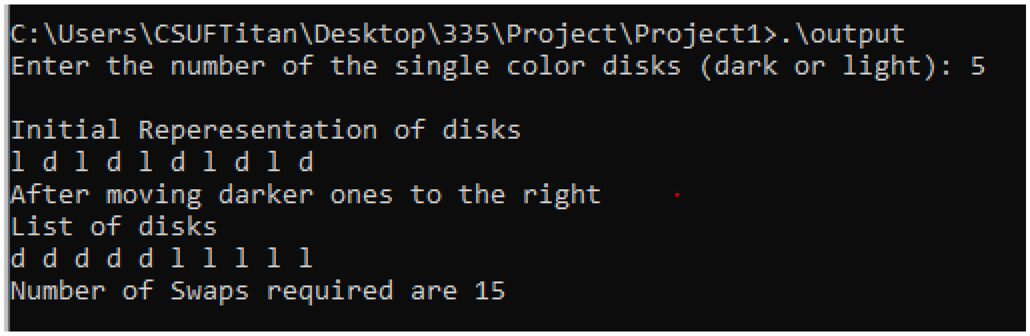


```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>.\output
Enter the number of the single color disks (dark or light): 2

Initial Reperesentation of disks
l d l d
After moving darker ones to the right
List of disks
d d l l
Number of Swaps required are 3
```

Test Case 3:

Input is 5



```
C:\Users\CSUFTitan\Desktop\335\Project\Project1>.\output
Enter the number of the single color disks (dark or light): 5

Initial Reperesentation of disks
l d l d l d l d l d
After moving darker ones to the right
List of disks
d d d d d l l l l l
Number of Swaps required are 15
```