CPSC 335 Project 1: empirical analysis

Fall 2015

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

# Introduction

In this project you will design, implement, and analyze two straightforward greedy algorithms for the same problem. For this problem, you will design two separate algorithms, describe the algorithms using clear pseudocode, analyze them mathematically, implement your algorithms in C/C++/Java, compile, test it, and submit BOTH the report (as a PDF file) and the programs.

# The Alternating disk problem

The problem is slightly different from the one presented in Levitin’s textbook as Exercise 14 on page 103, is as follows:

*You have a row of disks of two colors, dark and light, with n an even, positive integer. They alternate: dark, dark, light, light, dark, dark, light, light, and so on. You want to get all the light disks to the left-hand end, and all the dark disks to the right-hand end. You are ONLY allowed to interchange the positions of two neighboring disks. No other moves are allowed. You swap disks only if necessary otherwise you can skip them.   
Design an algorithm for solving this puzzle and determine the number of moves it takes.*





The *alternating disks problem* is:

**Input**: an even positive integer and a list of disks of alternating colors dark-dark-light-light,  
starting with dark

**Output**: a list of disks, the first disks are dark and the next disks are light, and an integer representing the number of swaps to move the dark ones before the light ones

There are two algorithms, presented next, that solve this problem in in You need to design them.

The first algorithm starts with the leftmost disk and proceeds to the right, doing the swaps as necessary. Once it reaches the right-hand end, it goes back to the leftmost disk and proceeds to the right, doing the swaps as necessary. It repeats times (which is the same condition as until there are no more disks necessary to move).

The second algorithm proceeds like a lawnmower: starts with the leftmost disk and proceeds to the right, doing the swaps as necessary. Once it reaches the right-hand end, it starts with the rightmost disk and proceeds to the left, doing the swaps as necessary, until it reaches the left-hand end. It repeats times (which is the same condition as until there are no more disks necessary to move).

For each algorithm, some improvement can be obtained by not going all the way to the left or to the right, since some disks at the ends are already in the correct position.

More details are given next.

# The left-to-right algorithm

It starts with the leftmost disk and proceeds to the right, doing the swaps as necessary. Once it reaches the right-hand end, it goes back to the leftmost disk and proceeds to the right, doing the swaps as necessary. It repeats times (which is the same condition as until there are no more disks necessary to move).

For the example shown on page 1, the exact list of disks changes as follows:









# The lawnmower algorithm

It starts with the leftmost disk and proceeds to the right, doing the swaps as necessary. Once it reaches the right-hand end, it starts with the disk before the rightmost disk and proceeds to the left, doing the swaps as necessary, until it reaches the disk before the left-hand end. It repeats until there are no more disks necessary to move.

For the example shown on page 1, the exact list of disks changes as follows:











# What to do

1. Design a greedy-based method for each algorithm and write clear pseudocode for your algorithm.

2. Analyze your pseudocode mathematically and prove its efficiency class using Big-Oh notation. (You need to compute the total number of steps of the algorithm.)

3. Type these notes (electronically or on paper) and submit it as a PDF report.

4. Implement your algorithm in C/C++/Java. You may use the template provided at the end of this file.

5. Compile and execute the program.

6. Create a file with the output of the program for an input value and submit it together with the program. Note, the output can be redirected to a file (for easy printing). For example, the following command line will create an output file in Linux-based operating system called a1out.txt by re-directing the output from the screen (display) to the file a1out.txt:

K:\cs202> a.out > a1out.txt

# Sample outputs Left-to-Right Algorithm:

Example #1:

K:\202> ast1a

CPSC 335-x - Programming Assignment #1

The alternating disks problem: left-to-right algorithm

Enter an even number of single color disks (light or dark)

4

Initial configuration

List of disks

d d l l d d l l

After moving darker ones to the left

List of disks

d d d d l l l l

Number of swaps is 4.

Example #2:

K:\202> ast1a

CPSC 335-x - Programming Assignment #1

The alternating disks problem: left-to-right algorithm

Enter an even number of single color disks (light or dark)

6

Initial configuration

List of disks

d d l l d d l l d d l l

After moving darker ones to the left

List of disks

d d d d d d l l l l l l

Number of swaps is 12

# Sample outputs for Lawnmower Algorithm:

Example #1:

K:\202> ast1b

CPSC 335-x - Programming Assignment #1

The alternating disks problem: lawnmower algorithm

Enter the number of single color disks (light or dark)

4

Initial configuration

List of disks

d d l l d d l l

After moving darker ones to the left

List of disks

d d d d l l l l

Number of swaps is 4

Example #2:

K:\202> ast1b

CPSC 335-x - Programming Assignment #1

The alternating disks problem: lawnmower algorithm

Enter the number of single color disks (light or dark)

6

Initial configuration

List of disks

d d l l d d l l d d l l

After moving darker ones to the left

List of disks

d d d d d d l l l l l l

Number of swaps is 12

# Template for a C/C++ program doing left-to-right algorithm:

// Assignment 1: Alternating disks problem, left-to-right algorithm

// XX YY ( YOU NEED TO COMPLETE YOUR NAME )

// Given 2n alternating disks (light, dark) the program reads the number of single color disks

// (light or dark), arranges the disks in the correct order and outputs the number of swaps

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

// OUTPUT: a list of 2n disks, the first n disks are light, the next n disks are dark,

// and an integer m representing the number of moves to move the dark ones after the light ones

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

void print\_disks(int n, char \*disks)

// YOU NEED TO IMPLEMENT THIS FUNCTION

// function to print the list of disks, given the number of single color disks and the actual list

// n represents the number of single color disks

// disks represents the list of disks (index 0 being the first disk) where

// 0 = a light color disks

// 1 = a dark color disks

int main() {

int n, m, k, i;

char \*disks;

// display the header

cout << endl << "CPSC 335-x - Programming Assignment #1" << endl;

cout << "The alternating disks problem: left-tor-right algorithm" << endl;

cout << "Enter the number of single color disks (light or dark)" << endl;

// read the number of disks

cin >> n;

// allocate space for the disks

disks = new char[n];

// set the initial configurations for the disks to alternate

for( i=0; i < n; i++) {

disks[2\*i] = 1;

disks[2\*i+1] = 0;

}

// print the initial configuration of the list of disks

cout << "Initial configuration" << endl;

print\_disks(n,disks);

// loop to push light one before darks ones

for (k=0; k < n ; k++) {

// YOU NEED TO COMPLETE THIS PART OF CODE

}

// after shuffling them

cout << "After moving darker ones to the left" << endl;

print\_disks(n, disks);

// print the total number of moves

cout << "Number of swaps is " << m << endl;

// de-allocate the dynamic memory space

delete [] disks;

return EXIT\_SUCCESS;

}

# Template for a C/C++ program doing lawnmower algorithm:

// Assignment 1: Alternating disks problem, left-to-right algorithm

// XX YY ( YOU NEED TO COMPLETE YOUR NAME )

// Given 2n alternating disks (light, dark) the program reads the number of single color disks

// (light or dark), arranges the disks in the correct order and outputs the number of swaps

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

// OUTPUT: a list of 2n disks, the first n disks are light, the next n disks are dark,

// and an integer m representing the number of moves to move the dark ones after the light ones

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

void print\_disks(int n, char \*disks)

// YOU NEED TO IMPLEMENT THIS FUNCTION

// function to print the list of disks, given the number of single color disks and the actual list

// n represents the number of single color disks

// disks represents the list of disks (index 0 being the first disk) where

// 0 = a light color disks

// 1 = a dark color disks

int main() {

int n, m, k, i;

char \*disks;

// display the header

cout << endl << "CPSC 335-x - Programming Assignment #1" << endl;

cout << "The alternating disks problem: lawnmower algorithm" << endl;

cout << "Enter the number of single color disks (light or dark)" << endl;

// read the number of disks

cin >> n;

// allocate space for the disks

disks = new char[n];

// set the initial configurations for the disks to alternate

for( i=0; i < n; i++) {

disks[2\*i] = 1;

disks[2\*i+1] = 0;

}

// print the initial configuration of the list of disks

cout << "Initial configuration" << endl;

print\_disks(n,disks);

// loop to push light one before darks ones

for (k=0; k < n ; k++) {

// YOU NEED TO COMPLETE THIS PART OF CODE

// DEVELOP ONE FOR LOOP FOR GOING LEFT TO RIGHT

// DEVELOP ONE FOR LOOP FOR GOING RIGHT TO LEFT

k++;

}

// after shuffling them

cout << "After moving darker ones to the left" << endl;

print\_disks(n, disks);

// print the total number of moves

cout << "Number of swaps is " << m << endl;

// de-allocate the dynamic memory space

delete [] disks;

return EXIT\_SUCCESS;

}