CPSC 335 Project 1: empirical analysis

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

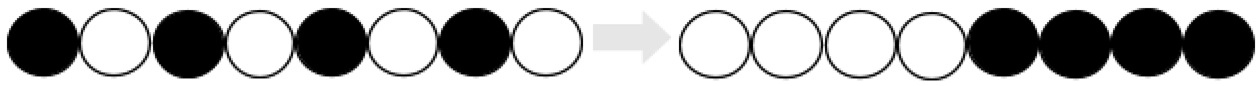
Introduction

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 submit BOTH the report (as a PDF file) and the programs.

The Alternating disk problem

The problem, presented in Levitin's textbook as Exercise 14 on page 103, is as follows:

You have a row of $2n$ disks of two colors, n dark and n light. They alternate: dark, light, dark, light, and so on. You want to get all the dark disks to the right-hand end, and all the light disks to the left-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.



The *alternating disks problem* is:
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 swaps to move the dark ones after the light ones

There are two algorithms, presented below, that solve this problem in in $O(n^2)$. You need to design them.

The first algorithm starts with the leftmost disk and proceeds to the right until it reaches the rightmost disk: compares every two adjacent disks and swaps them only if necessary. Now we have one lighter disk at the left-hand end and the darker disk at the right-hand end. 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 n times.

The second algorithm proceeds like a lawnmower: starts with the leftmost disk and proceeds to the right until it reaches the rightmost disk: compares every two adjacent disks and swaps them only if necessary. Now we have one lighter disk at the left-hand end and the darker disk at the right-hand end. Once it reaches the right-hand end, it starts with the rightmost disk, compares every two adjacent disks and proceeds to the left until it reaches the leftmost disk, doing the swaps only if necessary. The lawnmower movement is repeated $\lceil n/2 \rceil$ times.

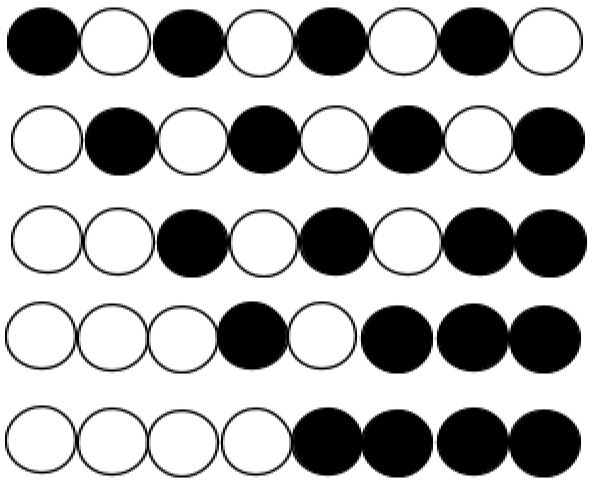
For each algorithm, some improvement can be obtained by not going all the way to the left or to the right, since some disks at the ends are already in the correct position.

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. Now we have one lighter disk at the left-hand end and the darker disk at the right-hand end. 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 until there are no more disks to move.

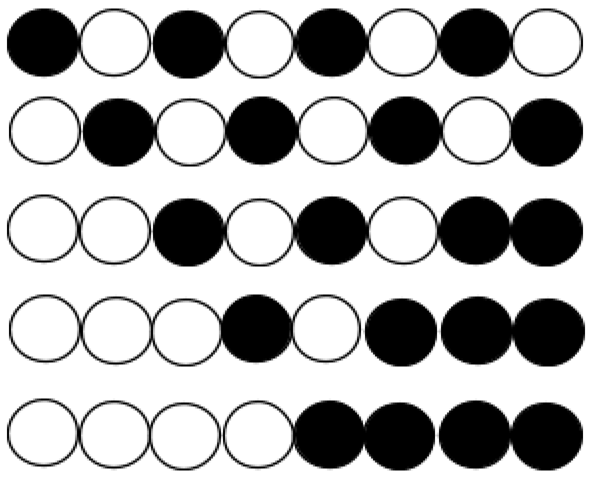
For the example shown, the exact list of disks changes as follows at the end of each run (left-to-right):



The lawnmower algorithm

It starts with the leftmost disk and proceeds to the right, doing the swaps as necessary. Now we have one lighter disk at the left-hand end and the darker disk at the right-hand end. 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 to move.

For the example shown, the exact list of disks changes as follows at the end of each run (left-to-right or right-to-left):



What you need 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:\cpsc335> a.out > a1out.txt
```

Sample outputs Left-to-Right Algorithms:

Example #1:

```
K:\cpsc335> ast1a
CPSC 335-x – Programming Assignment #1:
The alternating disks problem: left-to-right algorithm
Enter the number of single color disks (light or dark)
4
Initial configuration
List of disks
 d l d l d l d l
After moving darker ones to the right
List of disks
 l l l l d d d d
Number of swaps is 10
```

Example #2:

```
K:\cpsc335> ast1a
CPSC 335-x - Programming Assignment #1
The alternating disks problem:
Enter the number of single color disks (light or dark)
3
Initial configuration
List of disks
 d l d l d l
After moving darker ones to the right
List of disks
 l l l d d d
Number of swaps is 6
```

Sample outputs for Lawnmower Algorithm:

Example #1:

```
K:\cpsc335> 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 l d l d l d l
After moving darker ones to the right
List of disks
 l l l l d d d d
Number of swaps is 10
```

Example #2:

```
K:\cpsc335> ast1b
CPSC 335-x - Programming Assignment #1
The alternating disks problem: lawnmower algorithm
Enter the number of single color disks (light or dark)
3
Initial configuration
List of disks
 d l d l d l
After moving darker ones to the right
List of disks
 l l l d d d
Number of swaps is 6
```

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 (dark, light) 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-to-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[2*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);
// PART OF CODE MISSING
  // loop to push light one before the darks ones
  for (k=0;  k < n ; k++) {
  // YOU NEED TO COMPLETE THIS PART OF CODE FOR GOING LEFT TO RIGHT
  }
  // after shuffling them
  cout << "After moving darker ones to the right" << 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, lawnmower algorithm
// XX YY ( YOU NEED TO COMPLETE YOUR NAME )
// Given 2n alternating disks (dark, light)  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[2*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);
// PART OF CODE MISSING
  // loop to push light one before darks ones
  for (k=0;  k < n/2+1 ; k++) {
   // DEVELOP ONE FOR LOOP FOR GOING LEFT TO RIGHT
   // DEVELOP ANOTHER FOR LOOP FOR GOING RIGHT TO LEFT
  }
  // after shuffling them
  cout << "After moving darker ones to the right" << 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;
}
```