Practice 1. Introduction

María Segura Bolaños

Computational Complexity
School of Engineering
University of Cádiz

maria.segurabolanos@alum.uca.es

29th October 2023

Summary

In this Computational Complexity practice, two Turing Machine (TM) programs were designed and analyzed to solve different binary number problems. The first program was developed to accept binary strings with an equal number of zeros and ones, and the second one determines whether a binary number is a multiple of three. Also, both programs were implemented in a RAM model in order to see which algorithm is faster. Finally, it is concluded with the comparison of the different programs and checking that the Cobham-Edmons Thesis is fullfiled.

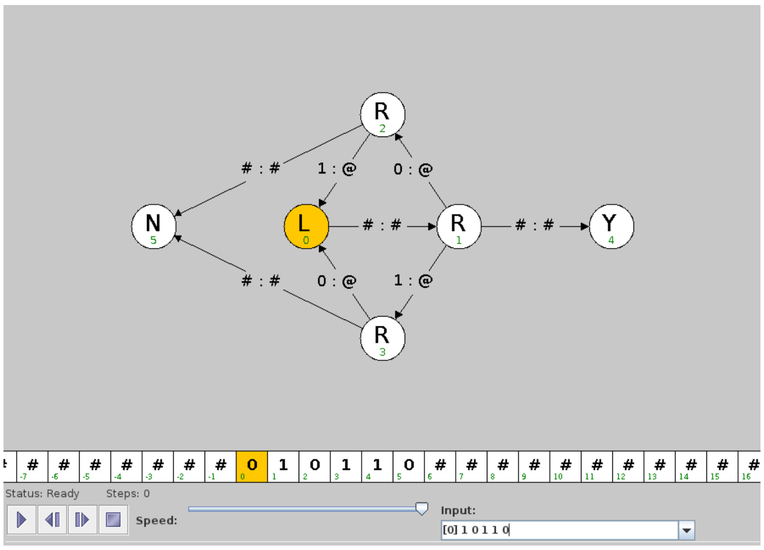
Contents

Tables

Figures

1 Binary strings with equal number of zeros and ones

First of all, I have implemented a program for the Touring Machine simulator, that accept all the binary numbers that have the same number of zeros and ones, and reject the others, by using the '@' symbol as an auxiliary character. It starts by moving the pointer to the beginning of the string, and after that, it searches the first number and replacing it with a @. If the number was one, it will keep searching a zero to replace it again with the auxiliary character and vice verse, if it was a zero, it will keep continue searching for a one. Once that I have processed a pair of different numbers, the pointer comes back to the beginning and the same procedure is repeated. The program will end by accepting the number when all the numbers have been processed in pairs, and rejecting it if it has processed a number and doesn't find its opposite.



Status: Ready Steps: 0

Input: [0] 1 0 1 1 0

Figure 1: Exercise 1 - Scheme of the program for the TM simulator

For the second section, I have run the program with ten random binary numbers, each of a different length (0, 10, 011, 1001, 10010, 010110, 0011110, 11001110, 010010101 and 0011010011) and I have gotten this data:

Nº Bits	1	2	3	4	5	6	7	8	9	10
Nº Steps	4	9	10	19	20	33	38	41	58	77

Table 1: Exercise 1 - Table of time vs input length

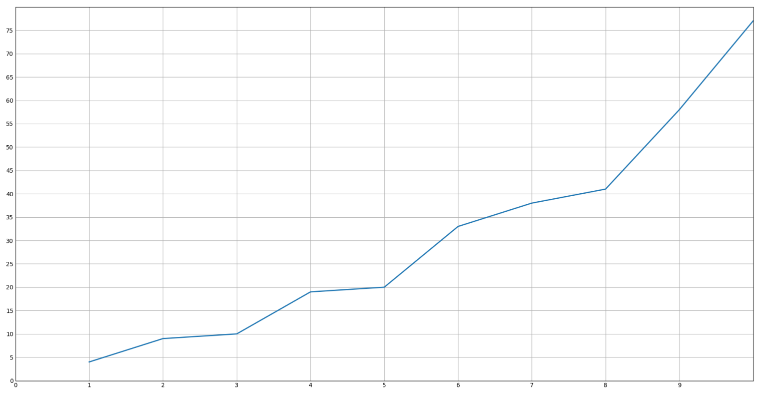


Figure 2: Exercise 1 - Graph of time vs input length

With a naked eye, I can think that the number of steps depend on the length of the input and how much it is sorted, because if for each pair of ones and zeros you have to go through many bits, the number of steps increases. So knowing this, the time complexity will probably be of order n^2.

To check my hypothesis with a formal analysis, I have executed the program with the worst cases for each input length (all zeros and ones grouped together and in odd numbers, the first group have an extra bit) and I have gotten the next data:

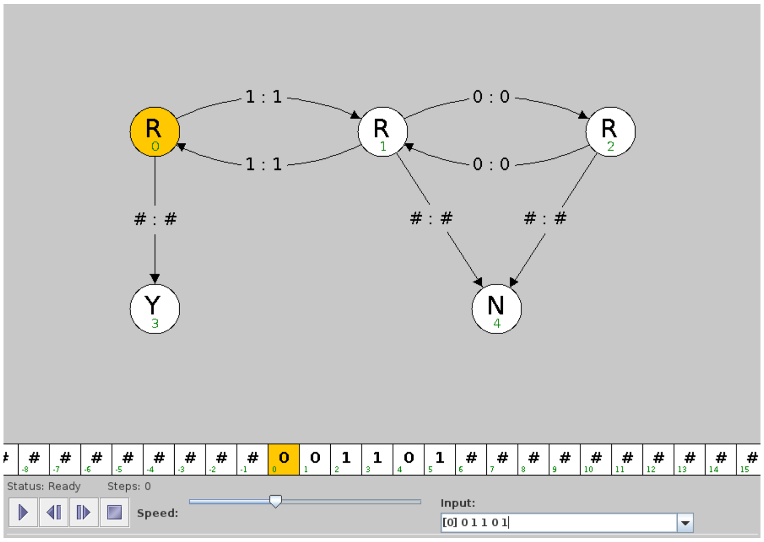
Nº Bits	1	2	3	4	5	6	7	8	9	10
Nº Steps	4	9	12	21	26	39	46	63	72	93

Table 2: Exercise 1 - Table of time vs input length for the worst cases

We can affirm that the time complexity in the worst cases is of order n^2 due to, in the worst case, we go through half of the binary string looking for the first digit and the other half looking for the second digit (order n) and we have to repeat it for all pairs of numbers.

2 Binary numbers which are a multiple of three

Before implementing this program, I looked up how to determinate if a binary number is a multiple of 3. This is when the difference between the sum of the digits in the odd positions and even positions is multiple of 3 [1], so I have designed a program for the TM simulator with the following scheme, in which the first state represents the remainder 0 of the division modulus 3, and the second and the third states represent the remainder 1 and 2, respectively, of the said division.



Status: Ready Steps: 0

Input: [0] 0 1 1 0 1

Figure 3: Exercise 2 - Scheme of the program for the TM simulator

As I have done in the first exercise, I have run this program in the TM simulator with ten random binary strings, each one of different length:

Nº Bits	1	2	3	4	5	6	7	8	9	10
Nº Steps	2	3	4	5	6	7	8	9	10	11

Table 3: Exercise 2 - Table of time vs input length

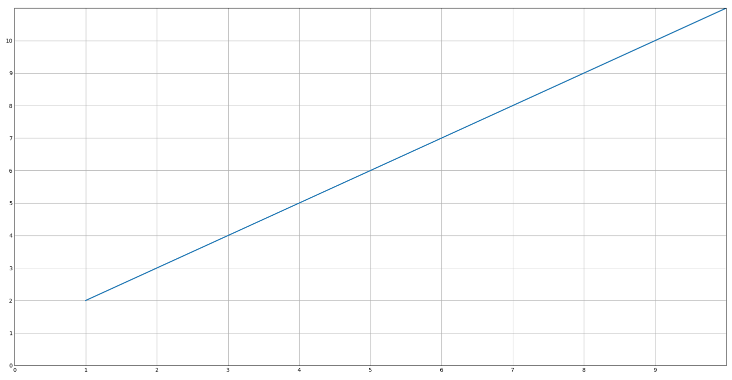


Figure 4: Exercise 2 - Graph of time vs input length

I observe that the time complexity only depends on the input length and it is of linear order.

In a formal analysis, we can say that the program always goes through all the bits and one more bit that determines the end of the string. So we can affirm that t(n) = n +1 $\in$ O(n) in all the cases.

3 Cobham-Edmonds Thesis

Simulating the first exercise in the RAM model, I obtain a time complexity of order n because we process the string digit by digit, and for each one it makes some instructions depending on whether it is 0 or 1.

The second exercise also has a time complexity of order n because of the same reason: we are going through all the string digit by digit doing some instructions depending on whether it is 0 or 1.

A RAM can't be exponentially faster than its optimal TM, but it can be polynomially faster. It is due to the Cobham-Edomnds Thesis, that affirm that the time complexity functions of a problem in two general and reasonable models of computation are related by a polynomial function [2]. As we can see if we compare both models in the exercises done in this practice, the program that accept the binary numbers if they have the same number of zeros and ones, it have a better order in the RAM model and the relation of both functions is n, and the program that accept the binary numbers if they are multiple of 3, both have the same order.

Bibliography

[1] GeeksforGeeks. *Check if binary string multiple of 3 using DFA*. 2022. URL: https://www.geeksforgeeks.org/check-binary-string-multiple-3-using-dfa/.

[2] Francisco Palomo Lozano. *Computational resources. Complexity measures and the asymptotic hierarchy*. Universidad de Cádiz.