Computational Complexity Theory

Aniket Pandey

2 February 2017

1 Acknowledgement

I would like to thank T. Aravind Reddy (B.Tech CSE) for mentoring me in the project and in helping me prepare this report. I would also like to thank ACA for giving me the opportunity to learn the topic in detail.

2 Introduction

Computability and Complexity are the two facets of this topic. Although both are interconnected in some ways, my focus would be in the complexity part of the theory, to understand how efficient certain algorithms are in solving a problem. Major part of my study is primarily based on classifying the problems within certain Complexity Classes. For example, one of the very famous open questions in Theoretical Computer Science, **P** vs **NP** , which tries to differentiate between the complexity classes **P** (Polynomial Time) and **NP** (Non-deterministic Polynomial Time) is whether they are same or different.

Next, I'll discuss about Turing Machines , its working and how influencial it was in modern computing. There are different properties associated with Turing Machines, different terminologies and different types as well.Then finally I'll cover Cryptography and usefulness of Complexity Theory in cryptography.

3 Notations

3.1 Big-O Notation

Big-O Notations are used in mathematics to characterize functions according to their growth rate. In Complexity Theory, efficiency of an algorithm is measured in terms of the input length n as $n \to \infty$.
Formal definition would be
If $f : N \to N$ and $g : N \to N$ are two functions, then f =O(g) if and only if $f(n) < c \cdot g(n)$ for a constant c as $n \to \infty$.

3.2 Other Notations

There are a few more notations which complement Big-O Notation. I will give a brief information about these.
For functions f & g from N to N

$$f = \Omega(g) \quad \text{if } g = O(f) \tag{1}$$
$$f = \Theta(g) \quad \text{if } f = O(g) \ \& \ g = O(f) \tag{2}$$
$$f = o(g) \quad \text{if there exists } \varepsilon \text{ such that } f(n) < \varepsilon \cdot g(n) \tag{3}$$
$$f = \omega(g) \quad \text{if } g = o(f) \tag{4}$$

4 Turing Machine

4.1 Computational Model

Computational model is a set of operations which are used to measure the resources(time and space) that are needed for a certain problem ot be solved. In layman's terms, using this model , we can determine if the given algorithm is efficient or not.
One such model is *Turing Machine* . It was invented by Alan Turing in 1936.

4.2 Turing Machine

Turing Machine is an abstract machine which is made up of infinite tape(s) divided into discreet boxes called *cells* and an imaginary head which can read the information in the boxes and can change the values of the cells. It also has a *state register* which stores the information about the state of the head.
The basic *Transition function* of a Turing Machine is defined as follows

$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^{k-1} \times \{L, S, R\}^k \tag{5}$$

The above equation describes the functioning of a simple Turing Machine consisting of k tapes where $1st$ tape is the input tape, the rest $k-1$ tapes are called *work tapes.* Last one of them is designated as output tape.
Here, the input tape takes the input, the work tapes decide what to do with the input and in which direction to move the head. Then finally, changes(if any) are done in the output tape.

4.2.1 Properties of Turing Machine

1. A