Lecture 1: INTRODUCTION TO COMPLEXITY THEORY

1 What is Complexity Theory

- One begins (naively) by asking about the "intrinsic complexity" of a problem P. For example, P_0 is integer multiplication (asked by Cobham) Another problem P_1 is graph isomorphism.

- Complexity theory must deal with different kinds of domains: integers, graphs, sets, polynomials, etc. E.g., for integer multiplication, the domain is $D = \mathbb{N}^2$, and for P_2, D =graphs. To unify all these domains, we insist:

  (A) The problem input and output are encoded as strings over Σ.

  Important: we side-step the issue of how to encode a natural domain (integers, graphs, sets, etc) into Σ^*. That is considered part of the problem specification! E.g., numbers are usually assumed to be in binary notation.

- Again, what is a problem P? There are many forms of problems. The most obvious is where P is a function. Complexity theory will focus on the special case where the function P is 0/1.

  (B) A problem P is a language $P \subseteq \Sigma$.

- Because of (A), some strings in Σ^* may not be well-formed (does not represent elements in the natural domain of P). But we insist:

  (C) All strings in Σ^* are valid inputs for our problem.

  In other words, any algorithmic solution must know how to parse the input (and reject if the input is not well

- Once we have agreed on the problem, we need to agree on the computational model. E.g., if P is the sorting problem, there are still many VERY different models (comparison tree, comparator network, tape models).

- There are many models that have been studied over the years. Here are 3 important ones: Turing Machines, Pointer Machines, RAMs.

  In classical computability theory, we have a powerful principle:

  CHURCH'S THESIS: the notion of computability is invariant across all (general) computational models.

- In complexity theory, the picture is much more complicated. To put some order into this mess, we must make two important observations that cuts across all "general" computational models:

  (1) COMPUTATIONAL MODES: Each computational model can operate in different modes: e.g., non-deterministic, probabilistic, parallel. EACH OF THESE MODES ARE SIGNIFICANT!

(2) COMPUTATIONAL RESOURCES: Within each computational model, we can identify computational resources corresponding as TIME, SPACE and REVERSAL. (Occasionally, a model may be missing some of these.)

- We now have a modified version of CHURCH'S THESIS:

  (H) A problem is tractable if it has polynomial complexity.

  (I) In each general computational model, there are natural notions of time, space and reversal complexity resource.

  (J) For each computational mode, and each computational resource, the notion of intractability is invariant.

  Many basic questions of complexity relates to the tractable classes defined in (J). For example:

  (P versus NP Question) With respect to time resource, the notion of intractability in the fundamental mode is different from the notion of intractability in the non-deterministic mode.

2 PRELIMINARIES

- Numbers: $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$.

- Complexity Function: these are partial functions from $\mathbb{R}$ to $\mathbb{R} \cup \{\infty\}$. Say $f \geq g$(ev.) if $f(x) \geq g(x)$ for $x \geq x_0$. Then $O(f)$ is the set of complexity functions g such that there is some $C = C(g) > 0$, $0 \leq g \leq Cf$.

- Languages: Σ is a finite set of symbols (alphabet). Σ^* are all finite strings over Σ. We let $|w|$ denote the length of a string. $\emptyset \in \Sigma^*$ denote the empty string.

  A language is a pair (Σ, L) where $L \subseteq \Sigma^*$. We usually simply call L the language. The complement of L is $\Sigma^* \setminus L$.

  If K is a collection of languages, co-K is the set of complements of languages in K.

- CONVENTION ALPHA: Σ_∞ is the universal set of symbols. Distinguished symbol: $\square$. Alphabets are finite subsets of Σ_∞, minus $\square$.

- CONVENTION BETA: Q_∞ is the universal set of states, disjoint from Σ_∞. Three distinguished states: q_0, q_a, q_r.