Lecture 1

What are Algorithms?

Goal: Study algorithms

- what is a problem?
- Decision problems → optimize
- Deterministic terms
- computer
- can we overcome?

Recent Developments

1) Information theory: Ω is a finite field

→ Decision Problem if not exactly possible, and measuring is only approximately possible (Turing of the computer theory)

2) Finite state automata: Turing machines can be seen as exactly (DFA/NFA) machines with infinite tape

$\Rightarrow$ Finite set of tape alphabet

3) Body Activity: Turing transducer (one to another)

→ parallel

→ Turing to the beginning

$Q \times \Sigma \to$ Finite set of states

A Turing machine: states has three alternative + one additional input to each of

$\Sigma \times \Sigma \times \{L, R, S\}$

$\delta: Q \times \Gamma \to Q \times \{L, R\}$

$\Sigma \subseteq \Gamma$

$\Sigma \cap \Gamma = \emptyset$

Church-Turing Thesis: Algorithm $\equiv$ simulation in turing machine

A turing machine can be modeled into another turing machine (if both consist of this)

$\text{Plug}(\langle M_1, M_2 \rangle)$ → converted turing machine

$\to M(M_2)$

Systematic Approaching by Dai Tsuna

Approximation algorithms

Lecture 2

Π Problem

→ with a program

What are computational problems?

→ has to input and output set

Decision Problems

- a yes/no/no
- has exactly one
- Finite

- Any answer can be needed to be the output or "no" (yes/no, number, product of two "0")
- output may vary
- output may vary
- if there is output or "0" length output than it is not computational

Decision problem must be a function of input and being "0" or "1"

Basically they are called Languages i.e. $\{0,1\}^*$

Decision Problems Σ ($\{0,1\}^*$) Language

Π: Given a graph G, Δ is connected

$\{\langle G \rangle \mid G \text{ is connected graph}\}$

Π: Given a G, Δ is prime?

$\{\langle n \rangle \mid n \text{ is prime}\}$

If the language is known, then the problem is known

Algorithmic Program	Problem
TM	Language

Why? Fact: A language L is said to be recognized if there is a TM M such that

- $x \in L \Rightarrow M(x)$ Accepts
- $x \notin L \Rightarrow M(x)$ does not accept

$L(M) = \{x \mid M(x) \text{ Accepts}\}$

Why? Fact: A language L is decided by TM M if

- $x \in L \Rightarrow M(x)$ accepts
- $x \notin L \Rightarrow M(x)$ rejects

Complexity Theory

- Decidability
- Recognition
- Recursive
- Recursively enumerable

Complexity Theory

- The Δ is tractable
- Efficiency

Fundamentals Theory

Theory Problems

→ Are there the hierarchy mechanism?

Now, there are languages

Next thing: Languages have very complexity and undecidable

Lecture 3

Topic: The UNIX of Recursively enumerable (RE) Language

RE = $\{x \mid x \text{ is a recursively enumerable}\}$

Topic: The Class of recursive language (R) or $R = \{x \mid x \text{ is a recursive}\}$

Recursive

Recursively enumerable

Proof: Γ: All $\to \{0,1\}^*$

We will show it through by length in ascending order and all strings of length length lexicographical

$\{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

At that is finite length, the string can be enumerated. Hence, Having a Bijection from natural numbers to strings is countably

Now: The set of all languages is uncountable!

Proof: $2^{\{0,1\}^*}$ is countable

Hence, we do all languages $\Rightarrow 2^{\{0,1\}^*}$ is countable

I can be countably infinite

So the countably, Suppose $2^{\{0,1\}^*}$ is countable:

$f(0) = L_0, L_1, L_2 \ldots$

$f(1) = L_0, L_1, L_2, \ldots$

$f(2) = L_0, L_1, L_2, \ldots$

$f(3) = L_0, L_1$

$\Rightarrow$ Define a new language L:

$L = \{x \mid x \in L_x\}$

$\Rightarrow$ L is not equal to any language L_x in f because $x \in L_x \Leftrightarrow x \notin L_x$

Contradiction

Suppose $L = L_x$

$x \in L_x \Leftrightarrow x \notin L_x$

Hypothesis

Let $\text{HALT}_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ halts on } w\}$

Proof: Suppose HALT_{TM} is decidable by H

$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ halts on } w \\ \text{reject} & \text{otherwise} \end{cases}$

Now, we can define a new machine D as follows:

$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } H(\langle M, \langle M \rangle \rangle) = \text{reject} \\ \text{reject} & \text{if } H(\langle M, \langle M \rangle \rangle) = \text{accept} \end{cases}$

Now, D is a TM, Σ FSA, Then LBA

Proof: LBA

If halts, w accepts $\Rightarrow$

If $\Rightarrow$ Σ M does not accept $\Rightarrow$

Σ accept

$\Rightarrow$ Σ M accept $\Rightarrow$

$\Rightarrow$ Σ M does not accept

Question for Σ:

Does M halt on M in particular, for if M accepts Σ, then M does not accept

If accepts Σ, then M does accept

Now: $H(\Sigma) \neq \Sigma$

Proof: Suppose

$\Rightarrow$ Σ M $\Rightarrow$ Σ M accepts $\Rightarrow$

$\Rightarrow$ Σ M accepts $\Rightarrow$

$\Rightarrow$ Σ M accepts