Algorithms Analysis and Design!

Algorithm:- No singular answer, operational definitions exist, like "Set of steps to reach a goal".

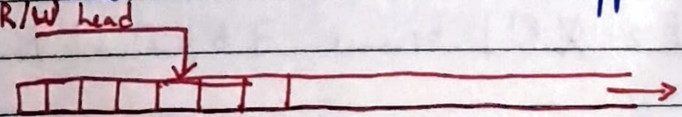
10th Hilbertian Question:- Give an algo to solve random Diophantine Eqn. [1900]

The definition of an algorithm has answered multiple questions like "What is a machine?", "What is a computer?", etc.

Axioms:

1) Information travels at finite speeds. [Spl. Theory of Relativity]
   → Random Access Memory is a myth. All memory is sequential, hence can be considered ~~tappcos~~ as tapes.

R/W head



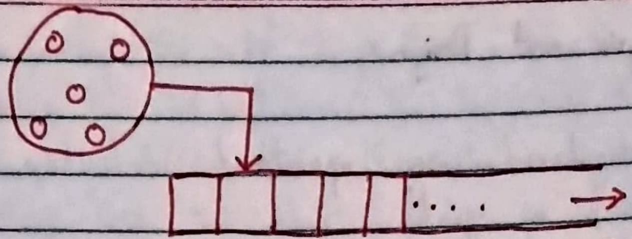
Speed of the R/W head is 1 cell/per unit time.

2) Finite volume in space can be used to store or retrieve only finite amounts of information reliably [Quantum Mechanics]
   → This implies that we can not store ∞ precision of irrational numbers. Thus using (perfectly precise) pi or the like in an algorithm is not allowed.
   Thus each cell can only store a finite amount of data [Based on A1 & A2]

Data in any cell $\in \Gamma$: Finite set of tape alphabet

3) Only finitely many instructions can be packed a priori. [ISA is finite]
   → Computers are finite; hence.

Algorithms book by Das Gupta



→ Anything tbf

(set of steps / processes etc)

Church Turing Thesis

Algorithm :- Any algorithm that can be simulated using a Turing Machine

Turing Machine :- A Turing Machine is a 7-tuple

$\langle Q, \Gamma, \delta, q_{start}, q_{accept}, q_{reject}, \Sigma \rangle$

$(q_i \in)$ $Q \rightarrow$ Finite ~~state~~ set of states

$\Gamma \rightarrow$ Finite set of state alphabets (that can fill cells)

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

$\Sigma \subset \Gamma$ $[\Sigma \not\subseteq \Gamma]$ because $\exists\, b \in \Gamma$ s.t. $b \notin \Sigma$

Test Quick brown Fox Jumps over The Lazy Dog

A Turing Machine can have another Turing Machine as its input

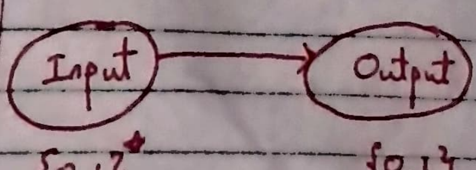
$U_{TM}(\langle M_{TM} \rangle, x) = M_{TM}(x)$

Where U_{TM} & M_{TM} are Turing Machines, x is the input to M_{TM} and $\langle\rangle$ is the string typecast.

U_{TM} is the Universal Turing Machine

What is a computational problem?

Decision Problems



Problems having multiple bits can be considered as multiple problems having single bit outputs

All ~~problems~~ computational problems are thus ~~considered~~ defined as problems that ~~are~~ have a set of inputs which can be ~~bi-partitioned~~ bi-partitioned into a T/F inputs. Since the two partitions are ME & E, just being given the input set and one of its subsets is sufficient. All computational problems are hence decision problems.

Decision Problems = (Membership Queries in) Languages.

A Language $L \subseteq \{0,1\}^*$

Given a graph G, is it connected or not? Decision problem

(or)

Does G belong to the set of all connected graphs Membership Query

$G \in \{G' \mid G' \text{ is a connected graph}\}$

Is a given number n prime?

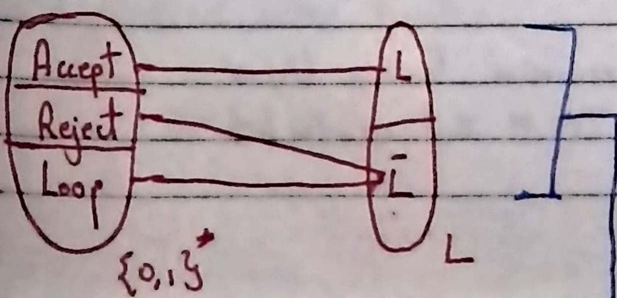
↓

$n \in \{i \mid i \text{ is prime}\}$

Problems are languages, solutions are Turing Machines

Turing Machines have three states, Accept, Reject and Loop

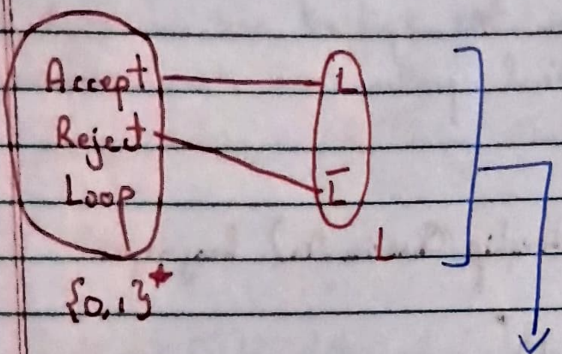
Languages have only two states, L & $\bar{L}$



Definition:- A language L is said to be recognized by a Turing Machine M if:-

$\forall x \in L$; $M(x)$ accepts.
else ; $M(x)$ rejects.

∴ Hence the language L of M $[L(M)] = \{x \mid M(x) \text{ accepts}\}$



Definition:- A language L is said to be ~~recognize~~ decided by TM M is

$\forall x \in L$; $M(x)$ accepts
else ; $M(x)$ rejects

A TM that decides L also recognizes L, but vice versa does not hold.

Theorem:- There are languages that are unrecognizable

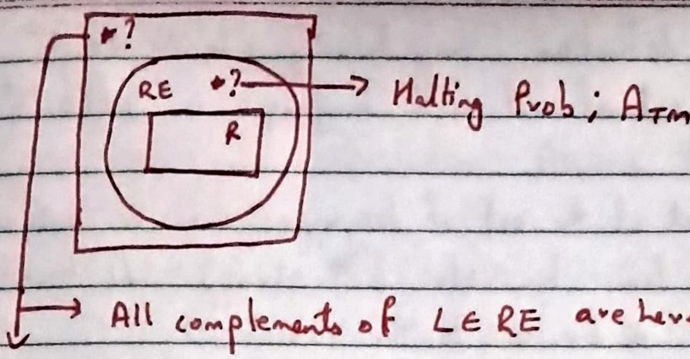
Theorem:- There are languages that are recognizable but undecidable

Defn:- The class of recursively ~~enumerated~~ enumerable (RE) languages is

$$RE = \{L \mid \exists \text{ a TM } M \text{ that recognizes } L\}$$

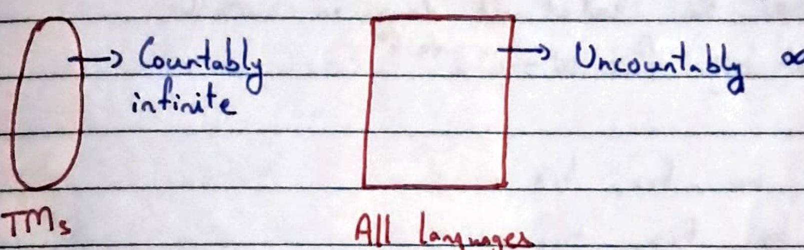
Defn:- The class of recursive languages (R) is

$$R = \{L \mid \exists \text{ a TM } M \text{ that decides } L\}$$



How do you show that $\exists$ languages here?

Take the set of all TMs and the set of all problems



- Any TM can be stored in a machine with finite bit strings, hence the no. of TMs $\leq$ no. of finite len bin strings
Thus TMs $\subseteq \{0,1\}^* \longrightarrow$ They can be enumerated
How?

Theorem:- $\{0,1\}^*$ is countable

Proof:- $f: N \rightarrow \{0,1\}^*$ must be a bijection
Consider the following enumeration of $\{0,1\}^*$
Shorter first, lexicographic ordering for eq lens.

$\epsilon, 0, 1, 00, 01, 10, 11, \underbrace{000, \ldots}_{8}, \underbrace{0000 \ldots}_{16}, \ldots$

This is a bijective map cause every string has a fixed index.
[Find a closed order formula]

Theorem:- The set of languages is uncountable.

apsara

Date: __.__.__

Proof:- By definition, any language $L \subseteq \{0,1\}^*$.
So the set of all such languages is $P(\{0,1\}^*)$

Each subset of the set of languages can be uniquely identified by an ∞ len bin str [str[i] = 0/1 based on whether the i^{th} element $\in$ or $\notin$ to the subset]

Any $L \equiv$ inf len bin str.

Assume that the set of all langs is countable.
Then $\exists$ a bij $f : N \rightarrow 2^{\{0,1\}^*}$

$f(1) = b_{11}, b_{12}, b_{13} \dots$
$f(2) = b_{21}, b_{22}, b_{23}, \dots$
$f(3) = b_{31}, b_{32}, b_{33}, \dots$

Since its countable, all N have been mapped to these $f(i)$
Now we diagonalization to create a new language
$L = \overline{b_{11}}, \overline{b_{22}}, \overline{b_{33}}, \overline{b_{44}}, \dots$
This L has no mapping and is distinct from all $f(i)$ in at least 1 pos. Hence its uncountably ∞.

The class RE are those programs which give yes, no or loop, but never returns a wrong answer.

Given a TM M and input x, will M accept x?

$A_{TM} = \{ \langle M, x \rangle \mid M \text{ accepts } x \}$

This problem is recognizable, not decidable $\rightarrow \in$ RE

$\langle M \rangle \rightarrow$ Program Typecast to a str to be used as input!

Date ________

Theorem:- A_{TM} is undecidable.

Proof:- We prove by contradiction. Assume that A_{TM} was decidable

$\rightarrow \exists$ a TM H that decides A_{TM}.

$\rightarrow \forall \langle M, x \rangle \in L$, $H(\langle M, x \rangle)$ accepts;

$\forall \langle M, x \rangle \notin L$, $H(\langle M, x \rangle)$ rejects; [Not loop]

Consider a TM D which does the following

i) On input M

ⓐ Run $\cancel{H(\langle M \rangle, m)}$ $H(M, \langle m \rangle)$

ⓑ Return $\neg$ H's result (accept to reject and vice versa)

Execute $D(\langle D \rangle)$

On input $\langle D \rangle$, Run $H(D, \langle D \rangle)$

If H accepts, then rejects $\rightarrow (D, \langle D \rangle) \in A_{TM}$

This implies D accepts $\langle D \rangle$ then D rejects $\therefore$

Contradiction

If H rejects, D accepts $\rightarrow (D, \langle D \rangle) \notin A_{TM}$

This implies D rejects $\langle D \rangle$ then D accepts.

Contradiction

(or)

	$\langle M_1 \rangle, \langle M_2 \rangle \cdots \langle M_i \rangle$
M_1	$H(M_1, \langle M_2 \rangle)$
M_2	
$\vdots$	
M_i	

Use diagonalization create a new D, but $H(D, \langle D \rangle)$ is a contradiction.

- That's how ~~digo~~ diagonalization works!

Theorem:- If $L \in RE$, and $\bar{L} \in RE$, then $L \in R$

Proof:- ~~$L \in RE \Rightarrow \forall x \in L, \exists$ TM M s.t M accepts x~~
~~$\forall$~~

i) $L \in RE \Rightarrow \exists$ a TM M s.t
$\forall x \in L$; M accepts x
$\forall x \in \bar{L}$; M rejects/loops on x

ii) $\bar{L} \in RE \Rightarrow \exists$ a TM $\underline{M}'$ s.t
$\forall x \in \bar{L}$; $\bar{M}$ accepts x
$\forall x \in L$; $\bar{M}$ rejects/loops on x

Create a decider ~~that~~ by running M and M' in parallel.
M accepts $x \in L$ and M' accepts $x \in \bar{L}$.

Theorem:- ~~$\overline{A_{TM}} \notin RE$~~ $\overline{A_{TM}} \notin RE$

Proof:- $A_{TM} \in RE$; $A_{TM} \notin R$
By contradiction and prev theorem, $\overline{A_{TM}} \notin RE$.

$Time(n^k) \rightarrow$ Problems solved in $O(n^k)$, where $k \in$ Rational
$Time(n^{k+\epsilon}) \rightarrow$ Superset of $Time(n^k)$

Reduction:- A language L_1 is mapping reduced to L_2 if $\exists$ a computable fn $f : \{0,1\}^* \rightarrow \{0,1\}^*$ s.t. $x \in L_1 \Leftrightarrow f(x) \in L_2$
$\forall x \in \{0,1\}^*$

$L_1 \leq_m L_2$

~~**Defn:** Language $A \leq_m B$~~

$f(\langle M, x \rangle) = \{ H(M,x) \wedge [M(x)] \}$

Examples:-

Let $HALT_{TM} = \{ \langle M, x \rangle \mid M \text{ halts on input } x \}$

Lemma:- $A_{TM} \leq_m HALT_{TM}$

Proof:- Decider for A_{TM} (using the one for $HALT_{TM}$)

Let H decide $HALT_{TM}$

On input $\langle M, x \rangle$:-

→ Run $H(M,x)$

→ If H rejects, REJECT

→ If H accepts, ACCEPT or REJECT based off $M(x)$

Theorem:- If $A \leq_m B$ and $B \in R \rightarrow A \in R$

Theorem:- If $A \leq_m B$ and $A \notin R \rightarrow B \notin R$

Completeness Theory:-

Suppose $\forall A \in RE$, $A \leq_m HALT_{TM}$.

Then $HALT_{TM}$ is RE-complete

Theorem:- $HALT_{TM}$ is RE-complete

Anything in RE

Proof:- Let TM M recognize A & H decides $HALT_{TM}$

Construct a decider for A.

On input x :

Run $H(M,x)$. If

$EQUAL_{TM} = \{<M_1, M_2> \mid L(M_1) = L(M_2)\}$

Theorem:- $EQUAL_{TM}$ is unrecognizable

Proof:- $\overline{A_{TM}}$ is unrecognizable ($\overline{A_{TM}} \notin RE$)

Let E recognize $EQUAL_{TM}$

M_1 : On input w, Reject

M_2 : On input ω, if $x \neq \omega$ Reject

If M accepts x, ACCEPT.

Else Reject

If $E(M_1, M_2)$ accepts $\rightleftharpoons$ M does not accept x.

$(M, x) \in A_{TM}$

Divide and Conquer.

Mergesort, Binary Search, Median of Medians.