31/7

Algorithms Analysis & Design

What's an algorithm ?

↳ Set of instructions to follow (Operational Definition)

≡ Time : Angle by which seconds hand moves

Actual definition ⟶ Causes revolution

- What's a computer ?
- Seperate Hardware / Software
- Programming Language
- Computer
- What's a machine ?

Truth : Algorithm [Turing Algorithm]

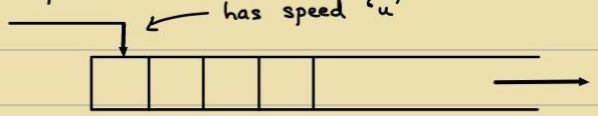
→ Axioms (Assumptions) :

1) Information travels at a finite speed [Sp. Th. of relativity]

Loophole : RAM on Earth ⇒ malloc array in outer space

↳ (Printing array[100] or array[10^{100}]

Every memory is sequential ⇒ Memory ≡ Tape



↳ Only Sequentially accessible

Read - Write Memory

W.L.G One cell at a time (OR) can stay

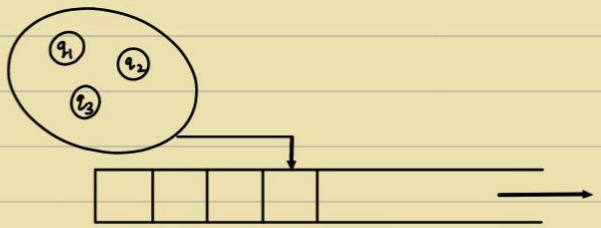
2) Finite volume in space can be used to reliably store/retrieve only finite amount of information. [Quantum Mech.]

e.g. Pendrive

Loophole : π

↳ Finite set of Tape Symbols [Σ, Γ]

3) Only finitely many instructions can be 'packed' at a time



$q_i \in Q$: Finite set of states

Turing Machine

A T.M is a 7 tuple

$< Q, \Gamma, \delta, q_{start}, q_{accept}, q_{reject} >$ $\quad \delta : \Gamma \times Q \longrightarrow \Gamma \times Q \times \{L, R\}$

7^{th} : Only occurs in Tape but not in Input

↳ Σ : Finite set of input symbols , $\Sigma \subseteq \Gamma$

$b \in \Gamma$, $b \notin \Sigma$

$< Q, \Gamma, \Sigma, \delta, q_{start}, q_{accept}, q_{reject} >$

ALGORITHM : Simulatable in some TM (Church - Turing Thesis / Hypothesis)

Computer : Algorithm

↳ Realistic machines

↳ Device which follows the axioms

VR : Anything that is real can also be virtual.

Single - structure programming Language ⟵ Turing Machine

Note : Turing Machine can be input to TM.

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

Universal Turing Machine : Universal General Purpose Software

x : Description of Circuit

M : Software

Software : Written on top of hardware

WHY :

$$F_i = F_{i-1} + F_{i-2}$$
$$F(1) = 1$$
$$F(0) = 0$$

} Recursion

Memoization } Iteration

$$F_n = \frac{1}{\sqrt{5}}(\phi)^n - \frac{1}{\sqrt{5}}(\bar{\phi})^n$$ } Formula $\quad \left[\phi = \frac{1+\sqrt{5}}{2}\right]$

↳ Eigenvalues [Hint to solve]

HOW :

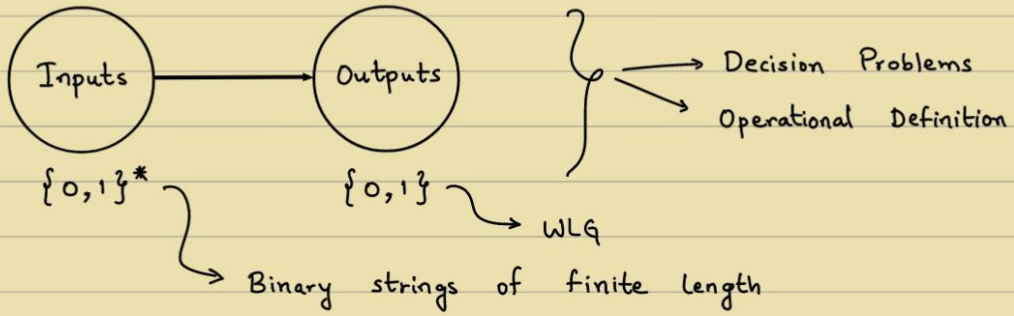
Algorithms book by DasGupta, Vaziram

→ Divide and Conquer
→ Greedy Algorithm
→ Dynammic Programming
→ Linear Programming
→ Complexity Theory
(Hardness)
→ Coping with Hard Problems
(Random, Approx, Quantum, etc)
→ Unsolvable problems

4/8

Q) WAP to ____
↳ Write A Program

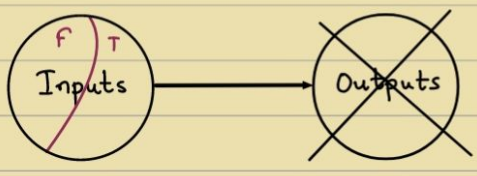
→ What are computation problems ?



Output → {0,1}
∵ If Output > 1 bit ⇒ Encode into multiple problems
i.e. if 01 → solution
0 : Solution of Problem 1 [2¹ place]
1 : Solution of Problem 2 [2⁰ place]

No output ⇒ Not a computation problem (Acc. to course)

∴ Decision Problem → 2 - Partition of the input set
↳ Language



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

Note: Decision Problem ≡ Membership queries in Languages

Q) Given a graph, is it connected or not ?

Graph ⟶ Input
↳ Encode into $\{0,1\}^*$, i.e. Adjacency Matrix / List

[OR] $G \in \{ G' \mid G' \text{ is a connected Graph} \}$?

Q) Given a natural no., is it prime ?
[OR] $n \in \{ i \mid i \text{ is prime} \}$

Note: If Answer is True / False ⟹ Always decision problem
⇓
Always computation problem
[e.g. Membership problem]

Q) WAP to print smallest prime no. ≡ WAP to print smallest even no. ?
YES in formal language
(NO in English)

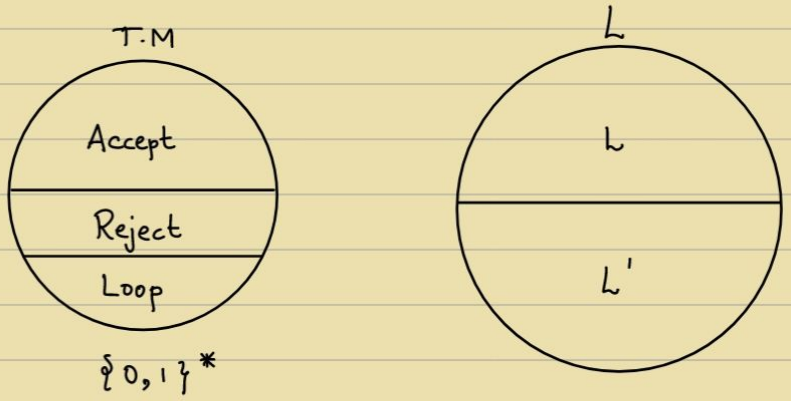
[Can you formalize it or not ?] ← Bigger Question
↳ Computation problem or not ?

- e.g. LLMs : Speech to text C program
  - ↳ Creates problems rather than solving it lol
    when we don't define the problem
  - ↳ Poses non-CSE problems as CSE problems

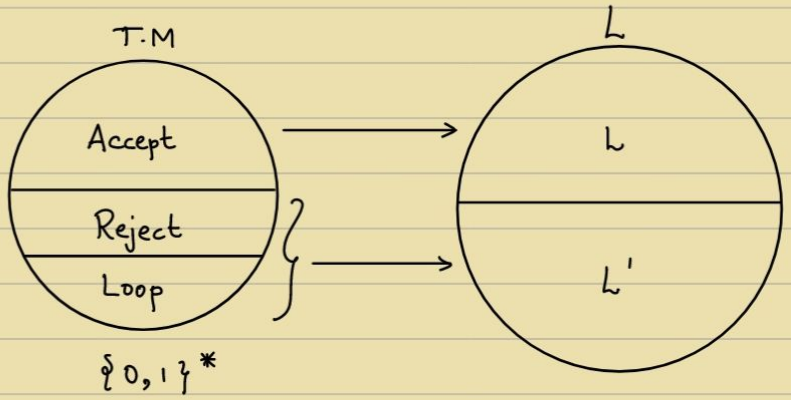
* { Given a language ⟶ We have a computation problem at hand
Given a T.M ⟶ We have an answer / solution / program at hand }
↳ What connects them ?

→ When do we say that a program solves a problem ?

e.g. $n \in$ Primes ?



Q) When do we say a T.M solves a problem ?



[Mapping]

Def^n : A Language L is said (↳ Acc. to world) to be recognized by T.M 'M' if

$\forall x \in L \rightarrow M(x)$ accepts

$\forall x \notin L \rightarrow M(x)$ does not accept

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

Def^n : A Language L is said (↳ Acc. to world) to be decided by T.M 'M' if

$\forall x \in L \rightarrow M(x)$ accepts

$\forall x \notin L \rightarrow M(x)$ rejects

" This one, ChatGPT will also not know "

— AAD Prof, 2025

{ Active voice is preferred in Scientific Computer Literature } ☹

→ Towards Optimization :

- What are resources ?
- Quantifying their use
- Best Notions , etc

Computability Theory

↳ What is a Theory ?

(Note : When something is an art , that means not everyone can do it
Science : Every man's art [No talent required]
Gifted Scientist ✗
Scientist ✓)

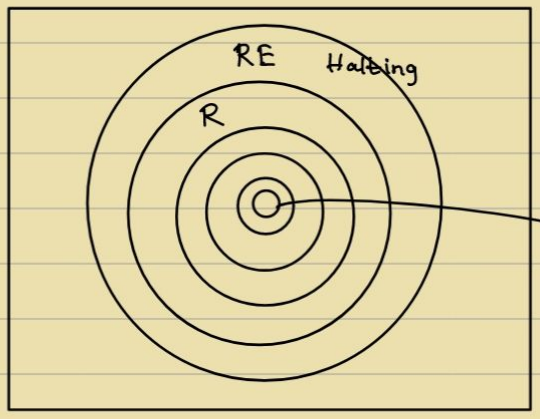
Theory gives you the answer to scientific problems.
when you can't perform scientific experiments.
Computability Theory : Tells what can be solved , what can't

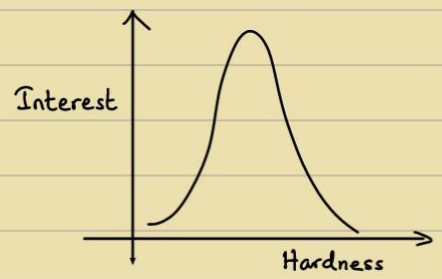
↳ Does ∃ a C program ?

Functionality Theory :
TM → works in steps , ∴ Complexity can be measured
i.e. How much memory it takes ?
How much time it takes ?
↳ default

" Whenever there is a reality , there is a virtual reality "

Theorem : There are Languages that are unrecognizable
Theorem : There are Languages that are recognisable but undecidable.





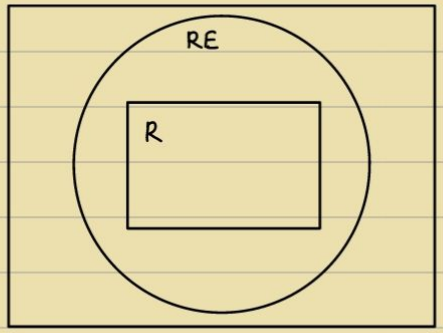
7/8

Def^n: The class of Recursively enumerable (RE) Language is

$$RE = \{L \mid \exists \text{ T.M 'M' that recognises } L\}$$

Def^n: The class of Recursive (R) Language is

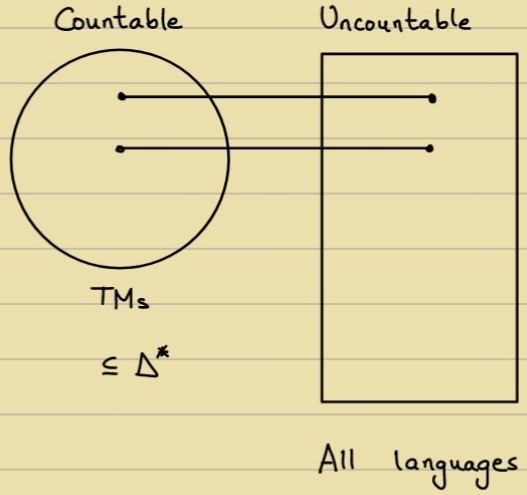
$$RE = \{L \mid \exists \text{ T.M 'M' that decides } L\}$$



Q) Are there languages unrecognisable by TMs / Computers / C-Programs ?

Q) Are there undecidable languages ?

Proof:



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

Proof: $f : \mathbb{N} \to \{0,1\}^*$

$$\varepsilon, 0, 1, 00, 01, 10, 11, \underbrace{\longleftarrow}_{8}, \underbrace{\longleftarrow}_{16}$$

$f(1) = \varepsilon$	$f(3) = 1$
$f(2) = 0$	$f(4) = 00$

Theorem: Set of all languages is uncountable.

Proof: A language $L \subseteq \{0,1\}^*$

Powerset of $\{0,1\}^* \longrightarrow 2^{\{0,1\}^*}$

Every element in powerset can be uniquely described by a binary string.

$\therefore$ Bijection

e.g. $A = \{1,2,3\}$, $|A| = 3$

$P(A) = \{\ldots, \{2,3\}, \ldots\}$

$\{2,3\} \rightarrow 011$

1 is absent

2 & 3 are present

On the contrary, Suppose $2^{\{0,1\}^*}$ is countable

$f(1) = b_{11}\, b_{12} \ldots b_{1i} \ldots$

$f(2) = b_{21}\, b_{22} \ldots b_{2i} \ldots$

$\vdots$

$f(i) = b_{i1}\, b_{i2} \ldots b_{ii} \ldots$

Method of diagonalisation:

$L = b_{11}\, b_{22}\, b_{33} \ldots b_{ii} \ldots$

Can't be $f(1)$ $\because$ Differs atleast 1 location from $f(1)$

Can't be $f(2)$ $\because$ Differs atleast 1 location from $f(2)$

Can't be $f(i)$ $\because$ Differs atleast 1 location from $f(i)$

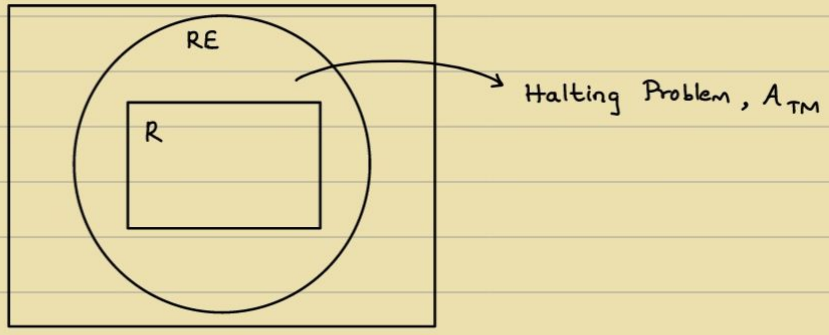
$\therefore$ Can't get bijection

$\therefore$ Can't be onto

$\therefore$ Uncountable

RE : Yes $\longrightarrow$ Yes

No $\longrightarrow$ No [OR] Unknown



- Given a TM 'M' and input 'x', does it accept x ?

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

Yes ⟶ Accept

No ⟶ Reject [or] Run forever

$\therefore A_{TM} \in R.E$

$A_{TM} \notin R$

Theorem : A_{TM} is undecidable

Proof : Suppose A_{TM} is decidable

Let TM 'H' decide A_{TM}

$\forall (M, x) \in A_{TM}$, $H(M, x) = \text{Accept}$

$\forall (M, x) \notin A_{TM}$, $H(M, x) = \text{Reject}$ [H : Halting TM]

Consider a TM 'D'

On input $\langle M \rangle$

→ Runs $H(M, \langle M \rangle)$ $\langle M \rangle$: Typecast program 'M' as strings

→ If H accepts, REJECT

Else if H rejects, ACCEPT

Execute $D(\langle D \rangle)$

On input $\langle D \rangle$,

→ Run $H(D, \langle D \rangle)$

→ If H accepts, then REJECT

if D accepts $\langle D \rangle$, then REJECT ← Which is impossible

→ If H rejects, then ACCEPT

If D does not accept $\langle D \rangle$, then ACCEPT ← Which is illogical

$\therefore$ D does not exist $\Longrightarrow$ H does not exist

(OR) Proof by diagonalisation metho

no. of TMs ⟶ countable

∴ Can be enumerable & orderable

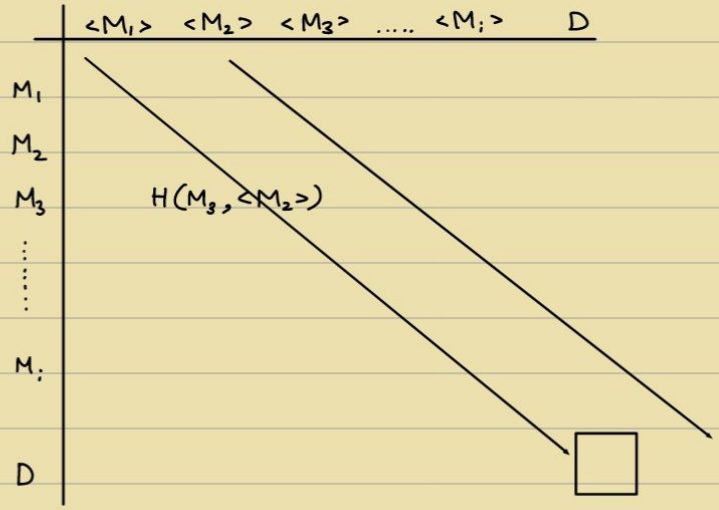


Table must have ACCEPT or REJECT.
But we get Blank
i.e. Whatever in diagonal, do opposite
∴ H cannot decide on all inputs

Theorem: If $L \in RE$, $\overline{L} \in RE$, then $L \in R$

Proof: $L \in RE$ ∃ TM 'M' s.t

$\forall x \in L \implies M$ accepts x

$\forall x \in \overline{L} \implies M$ does not accept x

$\overline{L} \in RE$ ∃ TM 'M'' s.t

$\forall x \in \overline{L} \implies M'$ accepts x

$\forall x \in L \implies M'$ does not accept x

Decider for L:

Just run M and M' in parallel

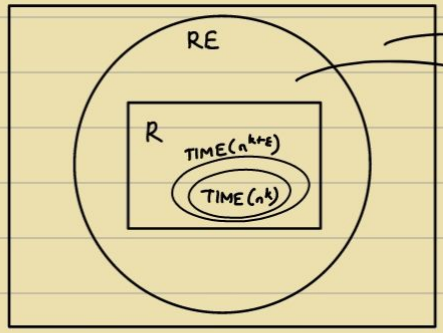
Theorem: $\overline{A_{TM}} \notin RE$

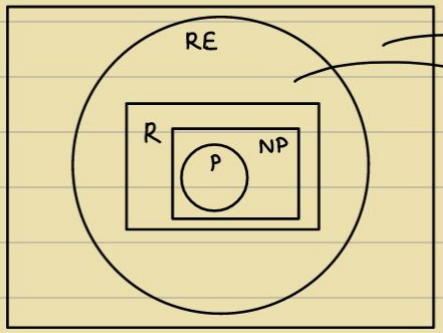
Proof: $A_{TM} \in RE$, $A_{TM} \notin R$ $\Big\}$ $\implies \therefore \overline{A_{TM}} \notin RE$

⟶ Take a class $O(n^k) \equiv TIME(n^k)$

Take another class $O(n^{k+\epsilon}) \equiv TIME(n^{k+\epsilon})$

For what value of ϵ, will they be diff.?





P : Easy Problems
↳ Solvable in Polynomial time

- **Reduction** : (Technique)

  Problem A
  Problem B
  } Given

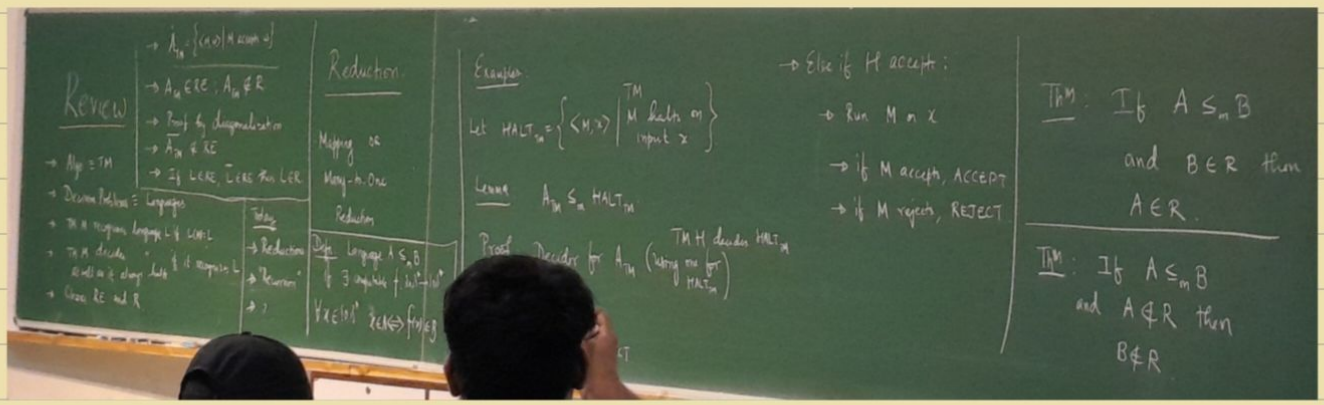
  Use Problem B as a subroutine to solve Problem A
  (Given there is a program for B.)

Def^n : A Language is mapping-reduced to L_2 if
$\exists$ computable function f, $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ S.T.

$$\forall x \in \{0,1\}^*, \quad x \in L_1 \iff f(x) \in L_2$$

Denoted as $L_1 \leq_m L_2$

11/8



→ Completeness Theory :

Theorem : $Halt_{TM}$ is RE - Complete

Suppose $\forall A \in RE$, $A_M \leq Halt_{TM}$,

Then, $Halt_{TM}$ is RE - Complete

Proof :

Let TM 'M' recognise A &

H decides $Halt_{TM}$

Decider for A :

On input x, Run $H(M, x)$ if H reads REJECT

Else, Run M on x and do linking

→ $EQUAL_{TM} = \{ \langle M_1, M_2 \rangle \mid L(M_1) = L(M_2) \}$

Theorem : $EQUAL_{TM}$ is unrecognisable

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

$M, x \in \overline{A_{TM}}$

Let E recognise $EQUAL_{TM}$

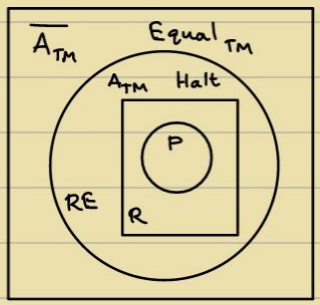
M_1 : On input w, REJECT

M_2 : On input w, if $x \notin w$, REJECT

if M accepts x, ACCEPT

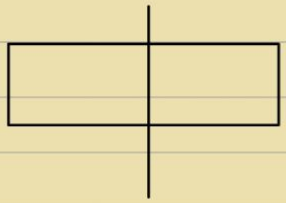
Else REJECT

if $E(M_1, M_2)$ accept $\iff$ M does not accept x



→ Divide and Conquer :

- Merge Sort



$T(1) = 1 \longrightarrow O(n)$

$T(n) = 2T(n/2) + O(n)$

$T(n) = O(n \log n)$

- Given $A = [\underbrace{\quad\quad}_{\hookrightarrow n \text{ elements}}]$

& $x \in A$

What is rank(x) in A ?

$O(n) \longrightarrow$ Easy

- Order Statistics

Given $A = (\quad)$ and index i

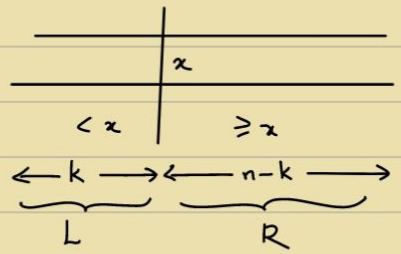
Find the i^{th} ranked element in A

$O(n \log n) \longrightarrow$ Easy

$O(n) \longrightarrow$?

$S(A, i)$

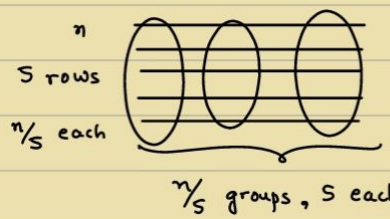
Pivot x



$$S(A, i) = \begin{cases} S(L, i), & \text{if } i \le k \\ S(R, i-k), & \text{if } i > k \end{cases}$$

$T(n) = O(n) + \max(T(k), T(n-k))$

Choose a 'good' x



$n/5$ medians

Let x be median of these $n/5$ medians

$S(n/5, n/10)$

$$T(n) = O(n) + T(n/5) + T(n/10)$$

$$T(n) = O(n)$$

$T(n) = cn$

$\Rightarrow cn = O(n) + c(n/5) + c(n/10)$

$= O(n) + nc(1/5 + 1/10)$

< 1

$= (0.1)\, cn = O(n)$

$n/3 + 2n/3 \longrightarrow$ Not < 1

$\therefore$ Won't work

Optimal group size ?

14/8

$\longrightarrow$ Fast Fourrier Transform (FFT) :

Input : Two polynomials of degree 'n-1' say A(x) and B(x)

Output : Their product C(x) , i.e. $C(x) = A(x) \cdot B(x)$

A[n] B[n] $\longrightarrow$ C[2n-1]

A[n] $\rightarrow$ Coefficient Array

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \qquad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

$$C(x) = \sum_{i=0}^{2n-2} c_i x^i$$

c_i : 'Direct Formula'

$$\sum_{j=0}^{i} a_j b_{i-j}$$

$O(n^2)$ algorithm

Can we do better ? i.e. $O(n \log n)$? FFT

Algorithm 'style'

1. Select (2n-1) or more points $x_1, x_2, x_3, \ldots\ldots, x_n$ $[O(n)]$
2. Evaluate $A(x_i)$ and $B(x_i)$ $[O(n^2)]$
3. Multiply $A(x_i) \cdot B(x_i) = C(x_i)$ $[O(n)]$
4. Interpolate $C(x_i)$'s to obtain C(x) $[O(n^3)]$

Task : ② → $O(n^2)$ to $O(n \log n)$

$$A(x) = A_e(x^2) + x\,A_o(x^2)$$

e.g. $A(x) = 5x^4 - 3x^3 + 7x^2 + 2x + 1$

$A_e(x) = 5x^2 + 7x^2 + 1$

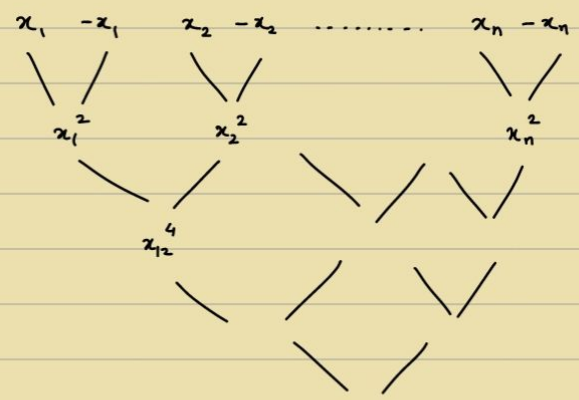
$A_o(x) = -3x + 2$

$$A(x_i) = A_e(x_i^2) + x_i \cdot A_o(x_i^2)$$
$$A(-x_i) = A_e(x_i^2) - x_i \cdot A_o(x_i^2)$$

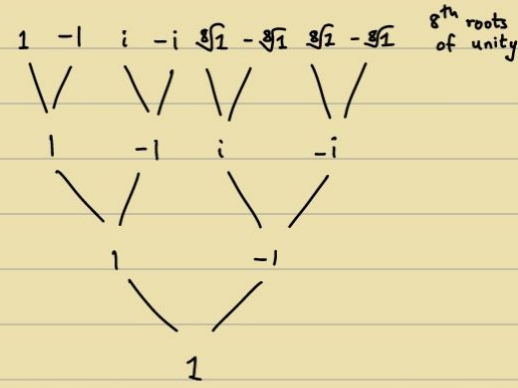
$T(n) = 2T(n/2) + O(n)$

$A_e(x_i^2) = A_{ee}(x_i^2) + x^2 \cdot A_{eo}(x_i^2)$

A



n^{th} roots of unity



$m \geq 2n-1$, $e^{i \cdot \frac{2k\pi}{n}}$

Goal : To evaluate a polynomial 'A' on the m^{th} roots of unity

$1, \omega, \omega^2, \ldots\ldots, \omega^{m-1}$

Input : A[ ]

Output : $(A[1], A[\omega], A[\omega^2], \ldots\ldots, A[\omega^{m-1}])$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & \omega & \omega^2 & \omega^4 & \cdots \\ 1 & \omega^2 & \omega^4 & \cdots & \\ 1 & \omega^4 & \vdots & \cdots & \end{bmatrix}_{m \times m} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{bmatrix} = \begin{bmatrix} A(1) \\ A(\omega) \\ A(\omega^2) \\ \vdots \\ A(\omega^{n-1}) \end{bmatrix}$$

1, i entry

Discrete Fourier Transform

[Interpolation]

FFT(A, ω):

```
if (ω == 1)
    return A(1);
else
    Ae ← [A(0), A(ω), .....]
    Ao ← [A(1), A(i), .....]

E[ ] ← FFT(Ae, ω²)
O[ ] ← FFT(Ao, ω²)

for i = 0 to m-1
    A[i] ← E[i/2] + ωⁱ · O[i/2]

for i = 0 to m/2
    A[i] ← E[i] + ωⁱ · O[i]
    A[i + m/2] ← E[i] + ωⁱ · O[i]
```

$$A(x) = A_e(x^2) + x \cdot A_o(x^2)$$
$$A(\omega^i) = E(\omega^{2i}) + \omega^i \cdot O(\omega^{2i})$$
$$A(-\omega^i) = E(\omega^{2i}) - \omega^i \cdot O(\omega^{2i})$$

$\therefore O(m \log n)$ multiplication

Algorithm 'style'

① Select $m \geq (2n-1)$ or more points, namely m^{th} roots of unity
$1, \omega, \omega^2, \omega^3, \ldots, \omega^{m-1}$ $\left(\omega = e^{i \cdot \frac{2k\pi}{m}}\right)$ (m: exact power of 2)

② Evaluate $A(x_i)$ and $B(x_i)$ $[O(n \cdot \log n)]$
FFT(A, ω) & FFT(B, ω)

③ Multiply $A(x_i) \cdot B(x_i) = C(x_i)$ $[O(n)]$

④ Interpolate $C(x_i)$'s to obtain $C(x)$ $[O(n^3)]$

$$C(x) = A(x) \cdot B(x)$$
$$C(1) = A(1) \cdot B(1)$$
$$C(\omega) = A(\omega) \cdot B(\omega)$$
$$C(\omega^2) = A(\omega^2) \cdot B(\omega^2)$$

$$\begin{bmatrix} & M & \end{bmatrix}_{m \times m} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} C(1) \\ C(\omega) \\ C(\omega^2) \\ \vdots \\ C(\omega^{m-1}) \end{bmatrix}$$

$$M_{ij} = \omega^{ij}$$
$$\omega = e^{i \cdot \frac{2\pi}{m}}$$

$M(\omega)$ = $(i,j)^{th}$ entry, ω^{ij}

$$\begin{bmatrix} c_0 \\ c_2 \\ \vdots \\ c_{m-1} \end{bmatrix} = M^{-1} \begin{bmatrix} c(1) \\ c(\omega) \\ \vdots \\ c(\omega^{m-1}) \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}_{m \times m} \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} = \begin{bmatrix} m & 0 & \cdots & 0 \\ 0 & m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m \end{bmatrix}$$

ω^{ij} $\qquad$ ω^{-ij}

$M^{-1} = \frac{1}{m} M(\omega^{-1})$ $\qquad$ $\frac{1}{m}$ FFT$(m\omega^{-1})$

18/8, 21/8, 25/8

Absent - Not well

28/8

→ Greedy Strategy :

- Minimum Spanning Tree – Krushkal, Prims, etc
- Activity Selection
- Knapsack Problem
- Huffman coding
- Djikstra's shortest path algorithm

→ Matroid Theory :

Def^n: A matroid is a pair $\langle S, I \rangle$, where

(a) S is a finite non-empty set

(b) I is a family of subsets of S. (Set of independent sets)

① Heredetiary Property: If $A \subseteq S$ belongs to I (if $A \in I$),

Then, $\forall B \subseteq A$, $B \in I$

② Exchange Property : If $A \in I$ and $B \in I$, and $|B| > |A|$,

Then, $\exists x \in B/A$ s.t. $A \cup \{x\} \in I$

Q) Given a matroid $\langle S, I \rangle$ and given non-negative integer weight to element of S. Find the maximum weight element of I. (A subset of S)

$$w(A) = \sum_{x \in A} w(x)$$

$x \in S$

$w(x)$ is the weight of x

- Universal Greedy Algo for weighted Matroids :
  (1) Sort the elements of S in non-increasing order of weights.
  (2) $A \leftarrow \phi$
  (3) In that order, let x be the root element
  if $A \cup \{x\} \in I$, then $A \leftarrow A \cup \{x\}$

- Proof of correctness / optimality :

  Greedy-choice Property : The element $x \in S$ with the maximum weight belongs to some optimum solution.

  Proof :

  Let $A \in I$ be an optimum solution

  if $x \in A$, done

  Suppose $x \notin A$, $\{x\} \in I$

  Let $A = \{a_1, a_2, \ldots\ldots, a_k\}$

  $\exists B = (A \cup \{x\}) - \{a_i\}$, $B \in I$

  $w(B) = w(x) + w(A) - w(a_i) \geq w(A)$

- Optimal sub-structure property :

  M contracts to M'

  $\langle S, I \rangle$ $\langle S', I' \rangle$

  $A = A' \cup \{x\}$

  $S' = S - \{x\}$

  I' = Sets of I with x erased

  $= \{B \mid B \cup \{x\} \in I\}$

  → All sets in I that originally contained x

- **Graph Matroid:**

Given a edge-weighted graph $G = (V, E)$

let $S_G = E$

$I_G = \{ E' \subseteq E \mid E' \text{ does not contain a cycle} \}$

$M_G = \langle S_G, I_G \rangle$ is a matroid.

I_G is hereditary.

I_G has exchange property

$|E_1| > |E_2|$

→ Forest with $|V| - |E_2|$ Trees

→ Forest with exactly $|V| - |E_1|$ Trees

E_1 has an edge (u, v) s.t. u and v are in different trees in E_2.

$E_2 \in I$

$E_2 \cup \{(u,v)\} \in I_G$

$\therefore$ Weighted Matroid

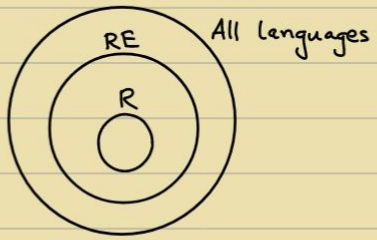
Quiz - 1

→ Knapsack Problem ✓
Greedy ✓
Activity Selection ✓
Krushkal's Algorithm ✓
Prim's Algorithm ✓
Cut Property ✓
Minimum Spanning Tree ✓

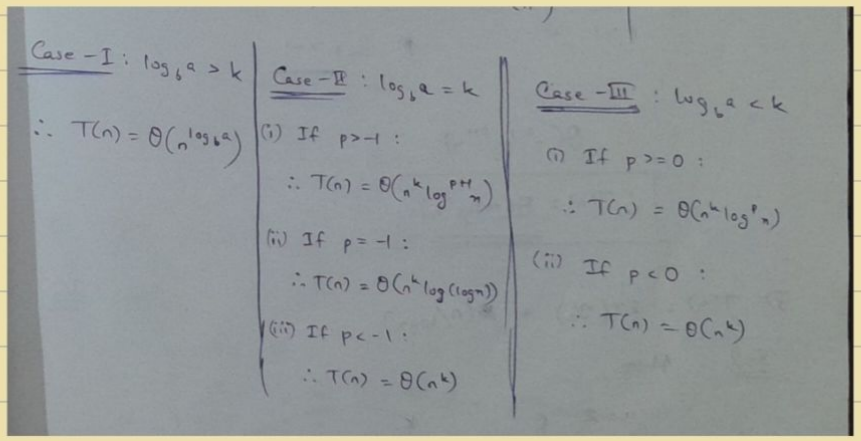
Strassen's Technique ✓ $O(n^{2.87})$
Binary Search ✓
Sorting - Counting Sort, Radix Sort
Merge Sort ✓
Karatsuba's Algorithm
Djikstra's Algorithm ✓
Master's Theorem ✓
Fast Fourier Transform https://www.youtube.com/watch?v=h7apO7q16V0

$Equal_{TM}$
$A_{TM} = \{ \langle M, x \rangle , M \text{ accepts } x \}$
$Halting_{TM}$
Completeness Theory
Reduction

→ Recursive : Decidable ← Turing Machine ⇒ Accept if in L, Reject otherwise
Recursive Enumerable : Recognisable ← TM ⇒ Accept if in L, Halt/Loop otherwise
↳ Partially / Semi - Decidable



https://www.youtube.com/playlist?list=PLUbapHgKkROmeWBKu1Dajo0REOB-MVBlU" ← Lec 32,33



$T(n) = a\,T(n/b) + f(n)$

Conditions :
$a >= 1$,
$b > 1$,
$k >= 0$,
$p \in R$

[where $f(n) = \Theta(n^k \log^p n)$]

Aim: Find a language that is not R.E.

$L_d = \{ x \in \{0,1\}^* \mid$ the TM whose code is x does not accept $x \}$

↳ binary strings

we will prove that L_d is not R.E.

Every B.S represents a TM.

Proof:- Assume L_d is R.E. ⇒ There exists a TM that accepts L_d.

⇒ Let us say that particular TM is i^{th} TM.

⇒ Let us say the encoding of that TM is x.

↳ B.S

↳ TM that accepts L_d

Case 1:- $x \in L_d$

⇒ The TM whose code is x does not accept x

⇒ i^{th} TM does not accept x

⇒ we said that $x \in L_d$ & i^{th} TM accepts L_d. ⇒ i^{th} TM accepts x

Case 2: $x \notin L_d$

⇒ TM whose code is x accepts x

⇒ i^{th} TM accepts x

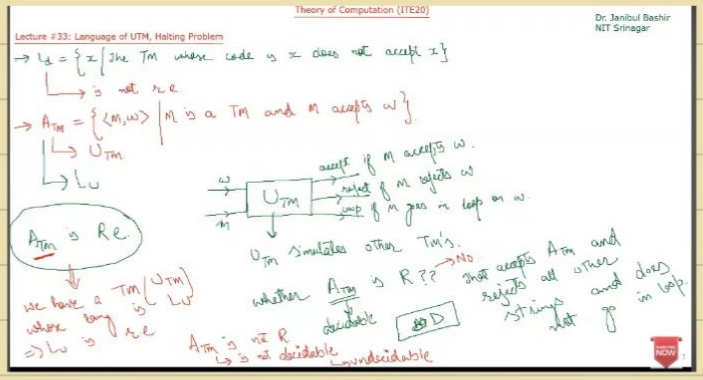
⇒ we said that $x \notin L_d$ & i^{th} TM accepts L_d.

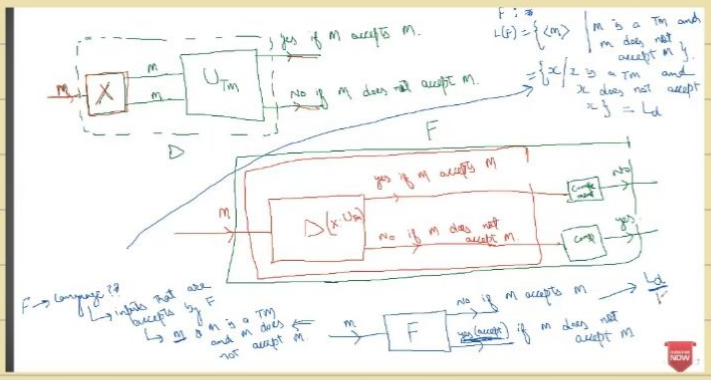
⇒ i^{th} TM does not accept x

⇒ L_d is not R.E.

contradiction

contradiction





Halt is R.E

Proof:- we will prove it by designing a TM where language is Halt.

$Halt = \{ \langle M, w \rangle \mid M$ halts on $w \}$

Accept if M accepts w.

reject if M rejects w.

Loop if M is in loop

accept if M accepts w and M rejects w

loop if M halts on w

loop if M goes in loop on w

$L(D) = \{ \langle M, w \rangle \mid M$ halts on $w \}$ = Halt → so we have a TM D, s.t

L(D) = Halt

⇒ Halt is R.E language.

Halt is not recursive ? decidable.

$Halt = \{ \langle M, w \rangle \mid M$ halts on $w \}$

Proof:- let us assume halt is R.

⇒ ∃ a TM (decider) that accepts Halt and rejects all other inputs (never goes in loop).

⇒ let us name that decider as D.

L(D) = Halt

yes if M halts on w.

if M does not halt on w.

if M halts on M

if M does not halt on M

Let us give F as input to F

F will halt on F — (1)

F will go in loop on F — (2)

(1) F halts on F

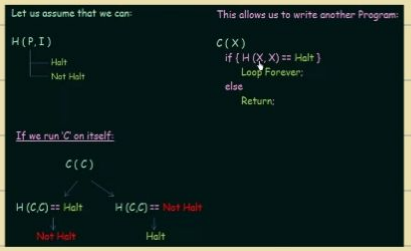
loop

contradiction

(2) here is also contradiction

⇒ our assumption is wrong

⇒ Halt is not R, decidable



Strassen's Matrix Multiplication

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

A, B, C, +T, −S, +Q, −R

$B_{11}\ A_{11}\ A_{22}\ B_{22}$

$P = (A_{11}+A_{22})(B_{11}+B_{22})$

$Q = B_{11}(A_{21}+A_{22})$

$R = A_{11}(B_{12}-B_{22})$

$S = A_{22}(B_{21}-B_{11})$

$T = B_{22}(A_{11}+A_{12})$

$U = (A_{21}-A_{11})(B_{11}+B_{12})$

$V = (B_{21}+B_{22})(A_{12}-A_{22})$

$C_{11} = P+S-T+V$

$C_{12} = R+T$ (RAT)

$C_{21} = Q+S$

$C_{22} = P+R-Q+U$

No. of Multiplic = 7

" Addit = 18

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 7T(n/2) + n^2 & \text{if } n > 2 \end{cases}$$

$a = 7$, $b = 2$, $f(n) = n^2$, $\log_b a = \log_2 7 = 2.81$

$O(n^{2.81})$

Karatsuba's Multiplication Algorithm

$X = 7768$ $a = 77$ $b = 68$

$y = 9276$ $c = 92$ $d = 76$

Step1- Compute $a*c = 77*92 = 7084$ → 70840000

Step2. Compute $b*d = 68*76 = 5168$ → 5168

Step3. Compute $(a+b)*(c+d)$

$(77+68)*(92+76)$

$(145)*(168) = 24360$ → 1210800

Step4, Compute Step3 − Step2 − Step1

$24360 - 5168 - 7084 = 12108$

$7768 * 9276 = 72055968$

```
def FFT(P):
    # P - [p_0, p_1, ..., p_{n-1}] coeff representation
    n = len(P) # n is a power of 2
    if n == 1:
        return P
    ω = e^{2πi/n}
    P_e, P_o = [p_0, p_2, ..., p_{n-2}], [p_1, p_3, ..., p_{n-1}]
    y_e, y_o = FFT(P_e), FFT(P_o)
    y = [0] * n
    for j in range(n/2):
        y[j] = y_e[j] + ω^j y_o[j]
        y[j + n/2] = y_e[j] - ω^j y_o[j]
    return y
```

FFT $P(x) : [p_0, p_1, \ldots, p_{n-1}]$, $\omega = e^{\frac{2\pi i}{n}} : [\omega^0, \omega^1, \ldots, \omega^{n-1}]$

$n = 1 \Rightarrow P(1)$

FFT $P_e(x^2) : [p_0, p_2, \ldots, p_{n-2}]$, $[\omega^0, \omega^2, \ldots, \omega^{n-2}]$

$y_e = [P_e(\omega^0), P_e(\omega^2), \ldots, P_e(\omega^{n-2})]$

FFT $P_o(x^2) : [p_1, p_3, \ldots, p_{n-1}]$, $[\omega^0, \omega^2, \ldots, \omega^{n-2}]$

$y_o = [P_o(\omega^0), P_o(\omega^2), \ldots, P_o(\omega^{n-2})]$

$P(\omega^j) = y_e[j] + \omega^j y_o[j])$

$P(\omega^{j+n/2}) = y_e[j] - \omega^j y_o[j])$

$j \in \{0, 1, \ldots (n/2 - 1)\}$

$y = [P(\omega^0), P(\omega^1), \ldots, P(\omega^{n-1})]$

IFFT(<values>) $\Leftrightarrow$ FFT(<values>) with $\omega = \frac{1}{n} e^{\frac{-2\pi i}{n}}$

```
def FFT(P):
    # P - [p_0, p_1, ..., p_{n-1}] coeff rep
    n = len(P) # n is a power of 2
    if n == 1:
        return P
    ω = e^{2πi/n}
    P_e, P_o = P[::2], P[1::2]
    y_e, y_o = FFT(P_e), FFT(P_o)
    y = [0] * n
    for j in range(n/2):
        y[j] = y_e[j] + ω^j y_o[j]
        y[j + n/2] = y_e[j] - ω^j y_o[j]
    return y
```

```
def IFFT(P):
    # P - [P(ω^0), P(ω^1), ..., P(ω^{n-1})] value rep
    n = len(P) # n is a power of 2
    if n == 1:
        return P
    ω = (1/n) * e^{-2πi/n}
    P_e, P_o = P[::2], P[1::2]
    y_e, y_o = IFFT(P_e), IFFT(P_o)
    y = [0] * n
    for j in range(n/2):
        y[j] = y_e[j] + ω^j y_o[j]
        y[j + n/2] = y_e[j] - ω^j y_o[j]
    return y
```

8/9

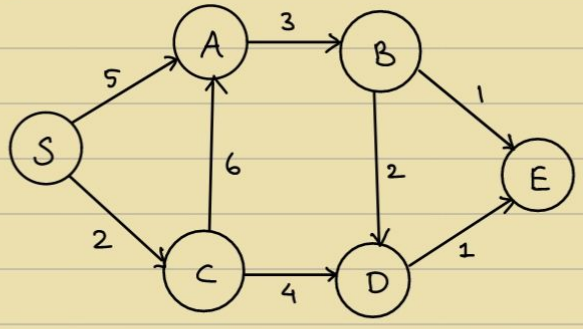
→ Dynamic Programming :

- A bag of subproblems S.T. it forms a DAG of 'smallest' to the 'largest' in Topological Sorted order.
- Problem : Shortest Path in DAGs

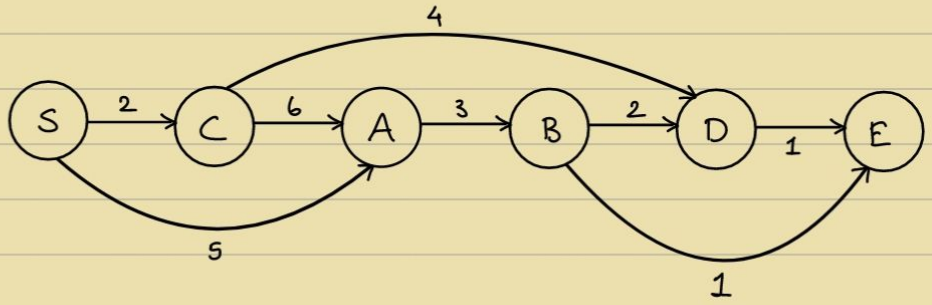
  Input : Weighted Directed Acyclic Graph
  Source S and Destination D

  Output : Shortest path from S and D.

Example :



Topological sort :



$$P(S, D) = \min \begin{cases} 2 + P(S, B) \\ 4 + P(S, C) \end{cases}$$

Termination condition : $P(S, S) = 0$

0	2	5	8	6	X
S→S	S→C	S→A	S→B	S→D	S→E

$$P(S, i) = \min_{(u,i) \in E} \left(P(S, u) + D(u, i) \right)$$

(in-neighbours)

$S \rightarrow A$: $\min(5, 8)$

$S \rightarrow D$: $\min(8, 10)$

- Longest path in DAG :

$$P(s, i) = \max_{(u,i) \in E} \left(P(s,u) + D(u,i) \right)$$

Topological sorting : $O(V+E)$

0	2	8	11	13	14
S→S	S→C	S→A	S→B	S→D	S→E

If efficient algorithm for finding longest path for general graphs, then P = NP

- Longest Increasing Subsequence :

Input : Array of numbers

Output : LIS

Every substring is a subsequence (→ may not be continuous), but not vice-versa.

e.g. $a_1, a_2, a_3, \ldots a_n$

$a_{i_1} < a_{i_2} < a_{i_3} < \ldots < a_{i_n}$ } → Sequence

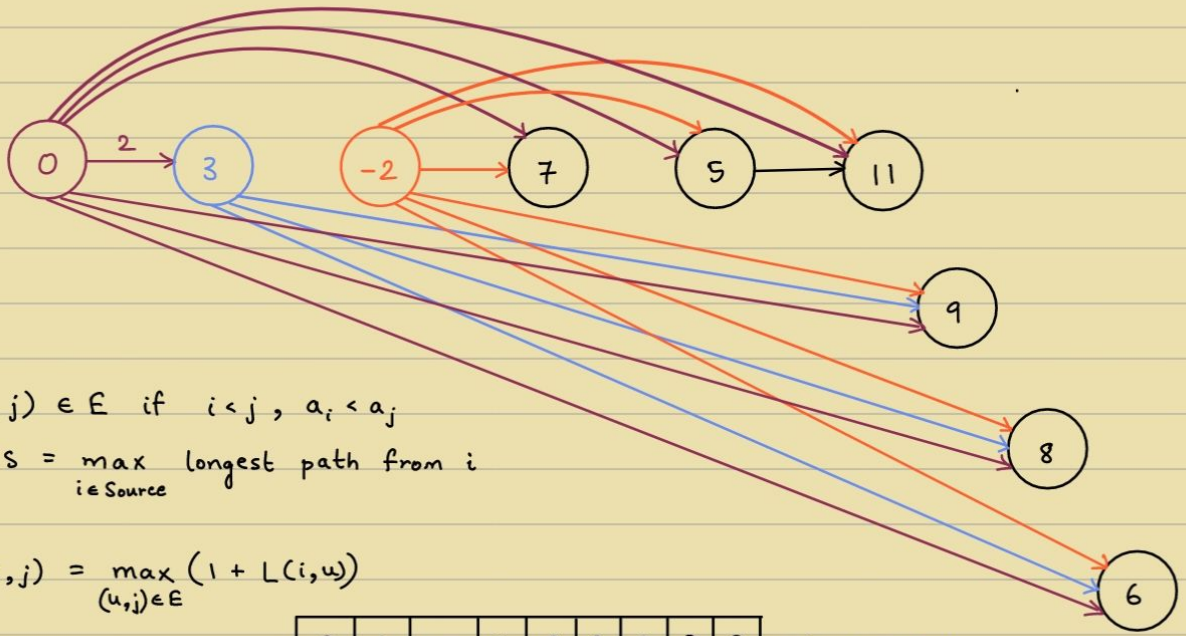
where $i_1 < i_2 < i_3 < \ldots < i_n$

e.g. 0, 3, -2, 7, 5, 11, 9, 8, 6

Sol : 0, 3, 7, 11 ⟶ length : 4

Viterbi's Algo, CYK Algo : Dynamic Programming

↓ Signal Processing ↓ ToC



$(i, j) \in E$ if $i < j$, $a_i < a_j$

$LIS = \max_{i \in \text{Source}}$ longest path from i

$$L(i,j) = \max_{(u,j) \in E} (1 + L(i,u))$$

0	1	-∞	2	2	3	3	3	3
0	3	-2	7	5	11	9	8	6

⇒ 0, 3, 7, 11

- Edit Distance : (Levenshtein distance)

$O(|E| \cdot k)$

$O(n^2) \longrightarrow O(n \log n)$?

↳ Worst case scenario

Operations : Insert , Delete , Overwrite

e.g. EARTH ⟶ HEART

① Insert H

HEARTH

② Delete H

HEART

e.g. SUNNY ⟶ SNOWY

① Overwrite U ⟶ N

② Overwrite N ⟶ O

③ Overwrite N ⟶ W

WAP that input 2 strings and finds the edit distance b/w them.

11/9

Review :

Dynamic Programming

e.g. ① Shortest / Longest Path in DAG

② Longest Increasing Subsequence

- Common Traits of D.P :
  - Bag / DAG of Subproblems
  - Subproblems are usually some part of the Input.
  - Recursive relation

- Edit Distance :

$x = x_1 x_2 x_3 \ldots\ldots x_n$

$y = y_1 y_2 y_3 \ldots\ldots\ldots y_m$

Add as many blanks to x and y S.T. they are aligned.

Cost of Allignment : No. of mismatches

EARTH ⟹ _EARTH

HEART HEART_

Subproblems :

(prefix) 1st i characters of x

(prefix) 1st j characters of y

$E(i,j)$ $\longrightarrow$ Goal : $E(n,m)$

$x_1\ x_2 \ldots\ x_i$

$y_1\ y_2 \ldots\ y_j$

Three cases :

(a) x_i, $-$ $\Rightarrow E(i,j) = E(i-1,j) + 1$

(b) $-$, y_j $\Rightarrow E(i,j) = E(i,j-1) + 1$

(c) x_i, y_j $\Rightarrow E(i,j) = \begin{cases} E(i-1,j-1) & x_i = y_j \\ E(i-1,j-1) + 1 & x_i \neq y_j \end{cases}$

$$\therefore E(i,j) = \min \begin{pmatrix} 1 + E(i-1,j) \\ 1 + E(i,j-1) \\ diff(x_i, y_j) + E(i-1,j-1) \end{pmatrix}$$

$$diff(x_i, y_j) = \begin{cases} 0, & x_i = y_j \\ 1, & x_i \neq y_j \end{cases}$$

Termination condition :

$E(0,j) = j$

$E(i,0) = i$

		x_1	x_2	x_3	...	x_n
	0	1	2	3	...	n
y_1	1					
y_2	2					
y_3	3					
⋮	⋮					
y_m	m					E(n,m)

fill it row-wise

to not get stuck anywhere.

e.g.

		H	E	A	R	T
	0	1	2	3	4	5
E	1	1	1 → 2 → 3 → 4			
A	2	2	2	1	2	3
R	3	3	3	2	1	2
T	4	4	4	3	2	1
H	5	4	5	4	3	2

e.g. E X P O N E N T I A L

P O L Y N O M I A L

		E	X	P	O	N	E	N	T	I	A	L
	0	1	2	3	4	5	6	7	8	9	10	11
P	1	1	1	2	3	4	5	6	7	8	9	10
O	2	2	2	3	2	3	4	5	6	7	8	9
L	3	3	3	3	3	3	4	5	6	7	8	9
Y	4	4	4	4	4	4	4	5	6	7	8	9
N	5	5	5	5	5	4	5	4	5	6	7	8
O	6	6	6	6	5	5	5	5	5	6	7	8
M	7	7	7	7	6	6	6	6	**6**	6	7	8
I	8	8	8	8	7	7	7	7	7	6	7	8
A	9	9	9	9	9	8	8	8	8	7	8	9
L	10	10	10	10	9	9	9	9	9	8	7	**6**

Note : IAL is not required ∵ both strings end with 'IAL'

Answer : 6

Working :

$dp[m+1][n+1]$

where $dp[i][j]$: min. edits to convert first i chars of EXPONENTIAL to first j chars of POLYNOMIAL

$dp[0][j] = j$

$dp[i][0] = i$

for $i>0$, $j>0$:

if $s_1[i-1] == s_2[j-1]$:

$dp[i][j] = dp[i-1][j-1]$

else :

$dp[i][j] = 1 + \min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1])$

↓ Delete ↓ Insert ↓ Replace

Result : $dp[1][10] = 6$

15/9

→ Review :

Dynamic Programming

Ex 1 : Shortest / Longest Path in DAG

Ex 2 : Longest Increasing Subsequence

Ex 3 : Edit distance

Ex. 4 : Chain Matrix Multiplication

→ General Technique :

- Creating subproblems using either prefix or some part of input.
- Relate the subproblems from easier to less easy in topological sorted order.

Input : Chain of Matrices

$$M^{(1)}_{P_0 \times P_1},\ M^{(2)}_{P_1 \times P_2},\ M^{(3)}_{P_2 \times P_3},\ \ldots\ldots,\ M^{(n)}_{P_{n-1} \times P_n}$$

Output : $M_{P_0 \times P_n} = \prod_{i=1}^{n} M_{P_{i-1} \times P_i}$

Cost : no. of elementary multiplications

Note : Notation :

$$M_{p,q} \times M_{q \times r} \longrightarrow pqr \text{ multiplications}$$

Example : $M^{(1)}_{100 \times 10} \times M^{(2)}_{10 \times 1000} \times M^{(3)}_{1000 \times 1} \times M^{(4)}_{1 \times 100}$

Total multiplications : $10^6 + 10^5 + 10^7 = 11100000$

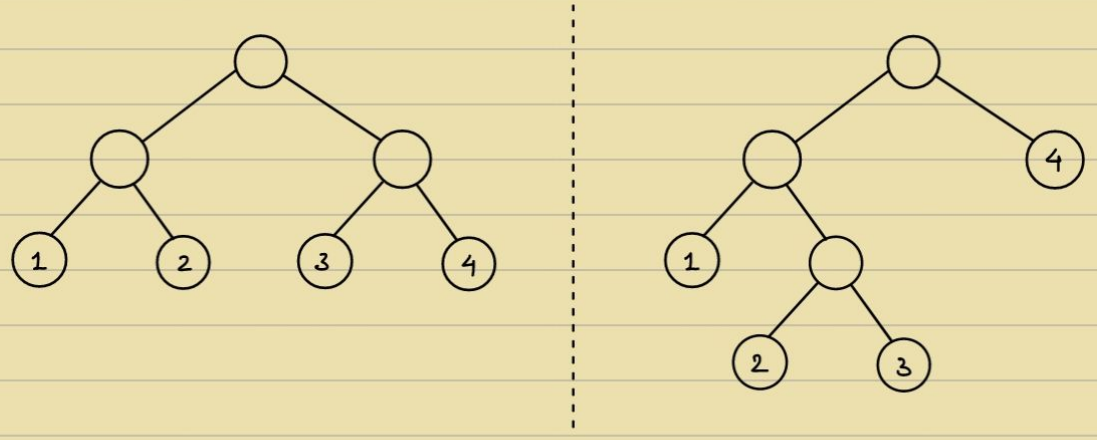
$$\left((M^{(1)} \times M^{(2)}) \times (M^{(3)} \times M^{(4)})\right)$$

But, $\left[M^{(1)} \times \left(M^{(2)} \times M^{(3)}\right)\right] \times M^{(4)}$

$10^3 + 10^4 + 10^4 = 21000$

Q) How many different ways are there to paranthesize a chain of n matrices ?

$\frac{1}{n+1}\binom{2n}{n}$ [Catalan numbers]



$C(i, j)$

Goal : To find $C(1, n)$

$C(i, i+1) = P_{i-1} P_i P_{i+1}$ ← Easy ones $\left[M^{(i)}_{P_{i-1} \times P_i} \right]$

Now, Only P_i's are given

$C(i,j) = C(1, k) + C(k+1, j) + P_i P_j P_k$

(if last step was given)

Suppose $\left(M^{(i)} \times M^{(k)}\right) \times \left(M^{(k+1)} \times M^{(j)}\right)$

Termination condition : $C(i, i) = 0$

$$C(i,j) = \min_{i \le k < j} \left\{ C(1, k) + C(k+1, j) + P_{i-1} P_j P_k \right\}$$

	1	2	3	4
1	0	10^6	11000 → 21000	← Goal
2		0	10^4	12000
3			0	10^5
4				0

Not necessary

$$C(1,3) = \min \begin{cases} C(1,2) + C(3,3) + P_0 P_2 P_3 \\ C(1,1) + C(2,3) + P_0 P_1 P_3 \end{cases}$$

$$C(1,3) = \min \begin{cases} C(1,3) + C(3,4) + P_1 P_3 P_4 \\ C(2,2) + C(2,4) + P_1 P_2 P_4 \end{cases} = \min \begin{cases} 11000 + 1000 \\ 10^5 + \dots \end{cases}$$

$$C(1,3) = \min \begin{cases} C(1,1) + C(2,4) + P_0 P_1 P_4 \\ C(1,2) + C(3,4) + P_0 P_2 P_4 \\ C(1,3) + C(4,4) + P_0 P_3 P_4 \end{cases}$$

18/9

→ Review :

D·P :

① Paths in DAG

② LIS

③ Edit distance

④ Chain Matrix Multiplication

→ Ex. Shortest Reliable Paths :

Input : Graph (Edge weighted) , s (start) , t (end) , integer k

Output : The shortest path from s to t in G using $\leq k$ edges

- Subproblems :

$\forall x \in V$, $\forall i < k$

Let dist(u, i) be the length be the length of the shortest path from s to u , using i hops

- Main problem :

$\min_{i \leq k} dist(t, i)$

Recurrence Relation :

$$dist(u, i) = \min_{(v,u) \in E} \left(dist(v, i-1) + w(v, u) \right)$$

Easiest Subproblem :

$dist(s, 0) = 0$ (Termination condition)

$dist(s, >0) = \infty$ (OR) $dist(u \neq s, 0) = \infty$

→ Ex. All Pair shortest Path : $O(n^3)$

Input : G

Output : $\forall u, v$, Length of shortest path from u to v.

- Subproblems :

Let dist (u, v, k) be the length of the length of the shortest path from u to v , using minimalistic nodes among 1, 2, ... , k

- Main Problem :

$k = |V|$

Recurrance relation :

$$dist(u, v, k) = \min \left(dist(u, v, k-1) , dist(u, k, k-1) + dist(k, v, k-1) \right)$$

$$dist(u, v, 0) = \begin{cases} w(u, v) , & \text{if } (u,v) \in E \\ \infty , & \text{otherwise} \end{cases}$$

Normal $\longrightarrow O(|E| \cdot \log(|V|))$

With fibonacii heap $\longrightarrow O(|E| + |V| \log(|V|))$

→ Review for midsem :

① Dynamic Programming
- Shortest / longest Path in DAG
- Longest Increasing Subsequence
- Edit Distance
- All Pair shortest Path
- Chain Matrix Multiplication
- Shortest Reliable Path
- Knapsack
- Independent set in Tree
- Polygon Triangulation
- Other questions in Prob. set

② Greedy Algorithm
- Huffman codes
- Djikstra's Algo
- Knapsack (0-1 & Fractional)
- Activity Selection
- Minimum / Maximum Spanning Tree (Prims Algo, Krushkal's Algo)
- Matroid Theorem
- Other questions in Prob. set

③ Divide and Conquer
- Merge Sort
- Integer Multiplication
- Matrix Multiplication
- Polynomial Multiplication
- Other questions in Prob. set

④ Computability Theory
- Church - Turing Thesis
- Model of Computation (TM)
- Diagonalization
- (a) Recognizability
  (b) Decidability
- Mapping Reduction