

Algorithms Analysis and Design

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

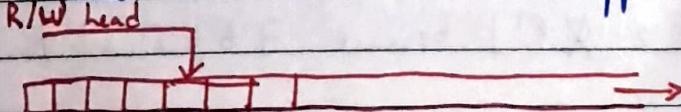
10th Hilbertian Question :- Give an algo to solve a random Diophantine Eqn. [1900]

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

Axioms Θ

1) Information travels at finite speeds. [Sp. Theory of Relativity]

→ Random Access Memory is a myth. All memory is sequential.
Hence can be considered tapes or tapes.



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]

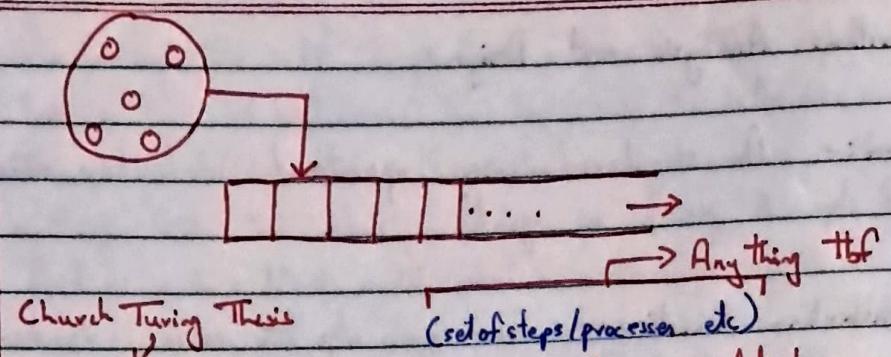
→ This implies that we can not store or precision of irrational numbers. Thus using (perfectly precise) π or the like in an algorithm is not allowed.

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.



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

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

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

(q, e) $Q \rightarrow$ Finite state set of states

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

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

$\Sigma \subset \Gamma$ [$\Sigma \neq \Gamma$] because $\exists b \in \Gamma$ s.t. $b \notin \Sigma$

Text

A Turing Machine can have another Turing Machine as its input

Quick

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

broken,

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

Judge

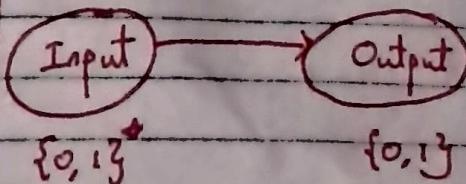
over.

The U_{TM} is the Universal Turing Machine

Lazj

Dog 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 into a T/F inputs. Since the two partitions are M & 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) Language.

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
 $G \in \{G' | G' \text{ is a connected graph}\}$ Membership Query

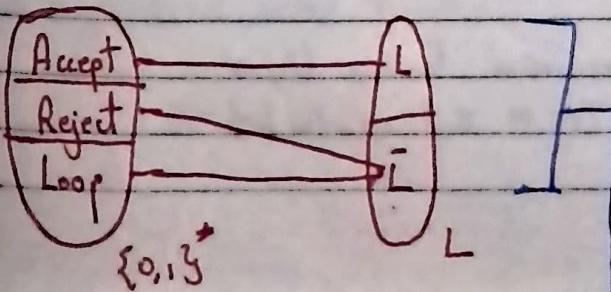
Is a given number n prime?



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

Problems are languages, solutions are Turing Machines

Turing Machines have three states, Accept, Reject and Loop
Languages have only two states, L & \bar{L}

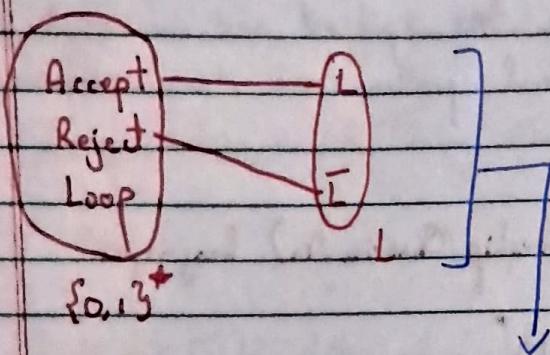


Definition: A language L is said to be recognized by a Turing Machine M if:-

$\forall x \in L ; M(x) \text{ accepts}$

else ; $M(x)$ rejects.

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



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

$\forall x \in L ; M(x) \text{ accepts}$

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

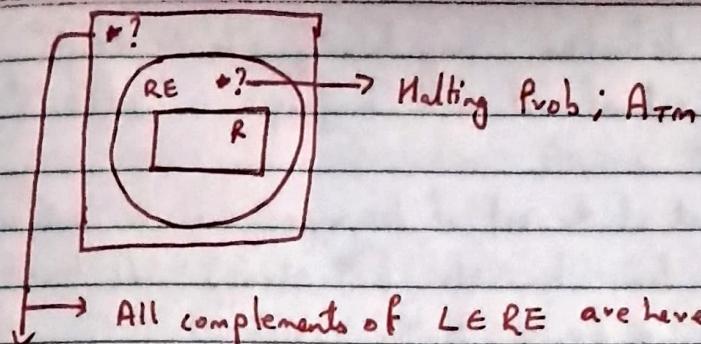
Theorem: There are languages that are recognizable but undecidable

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

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

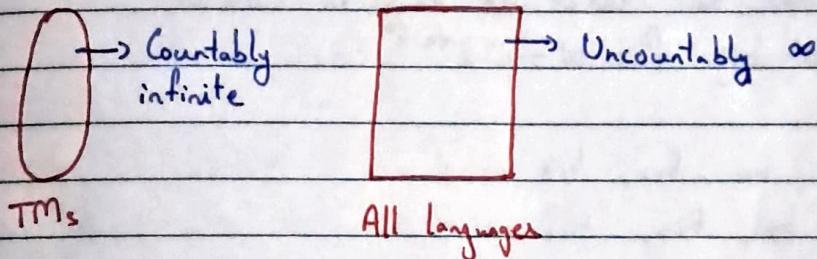
Defn: The class of recursive languages (R) is

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



How do you show that $\exists \exists$ 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 $TMs \subseteq \{0, 1\}^*$ → They can be enumerated

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 eg lens.

$\epsilon, 0, 1, 00, 01, 010, 11, 000, \dots, \underbrace{\dots}_{8}, \underbrace{\dots}_{16}, \dots$

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

Theorem:- The set of languages is uncountable.

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

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 $L \in \mathcal{P}(\{0,1\}^*)$

Assume that the set of all langs is countable.

Then \exists a bij $f : N \rightarrow \mathcal{P}(\{0,1\}^*)$

$$f(1) = b_{11}, b_{12}, b_{13}, \dots$$

$$f(2) = b_{21}, b_{22}, b_{23}, \dots$$

$$f(3) = b_{31}, b_{32}, b_{33}, \dots$$

Since it's countable, all ~~langs~~ have been mapped to these $f(i)$

Now we diagonalization to create a new language B

$$L = \overline{b_{11}}, \overline{b_{22}}, \overline{b_{33}}, \overline{b_{44}}, \dots$$

This L has no mapping and is distinct from all $f(i)$: 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 $\rightarrow \in \text{RE}$

$\langle M \rangle \rightarrow$ Program type cast 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} .
 $\rightarrow \forall \langle M, x \rangle \in L, H(\langle M, x \rangle) \text{ accepts};$
 $\forall \langle M, x \rangle \notin L, H(\langle M, x \rangle) \text{ rejects; [not loop]}$

Consider a TM D which does the following

i) On input M

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

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

B

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..

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

(iv)

	$\langle M_1, \langle M_1 \rangle \dots \langle M_i \rangle \dots \rangle$
M_1	$H(M_1, \langle M_2 \rangle)$
M_2	
:	
$\vdash M_i$	

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

- That's how diag
diagonalization works!

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

Proof:- $L \in RE \Rightarrow \forall x \in L, \exists TM m \text{ s.t } M \text{ accepts } x$

i) $L \in RE \Rightarrow \exists \text{ a TM } m \text{ s.t }$

$\forall x \in L; M \text{ accepts } x$

$\forall x \in \bar{L}; M \text{ rejects } x \text{ (loops on } x)$

ii) $\bar{L} \in RE \Rightarrow \exists \text{ a TM } m' \text{ s.t }$

$\forall x \in \bar{L}; \bar{M} \text{ accepts } x$

$\forall x \in L; \bar{M} \text{ rejects } x \text{ (loops on } x)$

Create a decider that by running M and M' in parallel.

\bullet $\bullet M$ accepts $x \in L$ and M' accepts $x \in \bar{L}$.

Theorem:- $\overline{A_m \notin RE} \quad \overline{A_m \notin R}$

Proof:- $A_m \in RE ; A_m \notin R$

By contradiction and previous theorem, $\overline{A_m \notin RE}$.

$\text{Time}(n^k) \rightarrow$ Problems solved in $O(n^k)$, where $k \in \text{Rational}$

$\text{Time}(n^{k+\epsilon}) \rightarrow$ Superset of $\text{Time}(n^k)$

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

$L_1 \leq_m L_2$

Defn: Language $A \leq_m B$

$$f(\langle m, x \rangle) = \{ h(m, x) \wedge [m(x)] \}$$

apsara

Date: _____

Example:-

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

Lemma:- $A_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}$

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 $\nexists A \in R_E$, $A \leq_m \text{HALT}_{\text{TM}}$.

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



$$\text{EQUAL}_{\text{TM}} = \{ \langle m_1, m_2 \rangle \mid L(m_1) = L(m_2) \}$$

Theorem:- EQUAL_{TM} is unrecognizable

Proof:- $\overline{\text{A}_{\text{TM}}}$ is unrecognizable ($\overline{\text{A}_{\text{TM}}} \notin \text{RE}$)

Let E recognize EQUAL_{TM}

w

M_1 : On input ~~w~~, Reject

M_2 : On input w, if $x \in w$ Reject

If M accepts x , ACCEPT.

Else Reject

If $E(m_1, m_2)$ accepts $\Rightarrow M$ does not accept x .

$(M, x) \in \text{A}_{\text{TM}}$

Divide and Conquer.

Mergesort, Binary Search, Median of Medians.

Fourier Transforms:-

Input:- Two polynomials $A(x)$ & $B(x)$ of degree $n-1$

Output:- Their product $C(x) = A(x) \cdot B(x)$

$A[n]$ → coefficients of A

$B[n]$ → coefficients of B

$C[2n]$ → coefficients of C

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

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

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i$$

$$c_i = \sum_{j=0}^i a_j b_{i-j} \rightarrow \text{But this is an } O(n^2) \text{ algorithm}$$

But Fast Fourier Transforms can do it in $O(n \log n)$

Algorithm :-

- (a) i) Evaluate Select $2n-1$ or more points x_1, \dots, x_m [$m \geq 2n-1$]
ii) Evaluate $A(x_i)$ and $B(x_i)$ $\rightarrow O(n^2)$ [n pts with $n-1$ substitutions per pt]
iii) Multiply $A(x_i) \times B(x_i) \Rightarrow C(x_i) \rightarrow O(n)$
iv) Interpolate $C(x_i)$, to obtain $C(x) \rightarrow O(n^2)$

Let $A(x)$ be the polynomial

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

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

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

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

For any $A(x_i) = A_e(x_i^2) + x_i A_o(x_i^2)$
 $A(-x_i) = A_e(x_i^2) - x_i A_o(x_i^2)$

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

But ~~$A_e(x_i^2) = A_{ee}(x_i^4) + x_i^2 A_{eo}(x_i^4)$~~
 ~~$A_e(-x_i^2)$~~

$\rightarrow +/-$ space

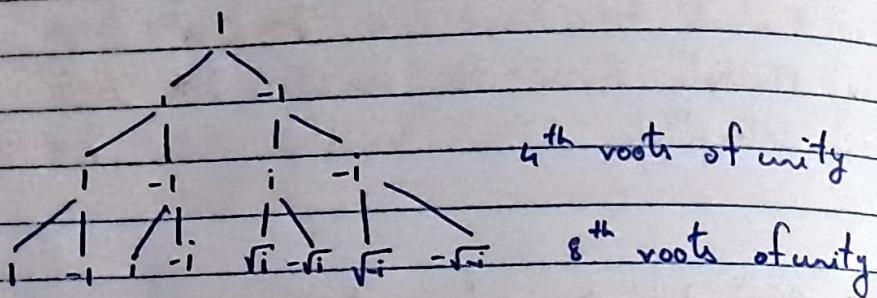
$$A: \{x_1, -x_1, x_2, -x_2, \dots, x_d, -x_d\}$$

x_1^2 x_2^2 \dots x_d^2

\rightarrow Not $+/-$ space \Rightarrow Can't reverse further.

Hence, we pick complex no.s to evaluate.

Let begin from the bottom.



m^{th} roots of unity

$$e^{i \frac{2\pi}{m} j}$$

m is an exact power of 2 $\geq 2^{n-1}$

$$j \rightarrow [1, m]$$

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

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

Input: $A[]$

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

$$A = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} \quad A(x) = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{d+1} & x_{d+1}^2 & \dots & x_{d+1}^d \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}$$

To evaluate our output, we multiply A with

$$\left[\begin{array}{cccccc} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^d \\ \vdots & & & & \\ 1 & \omega^{m-1} & \omega^{2m-2} & \dots & \omega^{(n-1)} \end{array} \right] \left[\begin{array}{c} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{array} \right] = \left[\begin{array}{c} A(1) \\ A(\omega) \\ \vdots \\ A(\omega^{n-1}) \end{array} \right]$$

$\hookrightarrow i, j^{\text{th}}$ entry $\rightarrow \omega^{ij} + i, j \in [0, m-1]$

\hookrightarrow Discrete Fourier Transform.

FFT FFT(A, ω):

if $\omega = 1$:

return $A(1)$

else:

$A_e = [A[0], A[2], \dots]$

$A_o = [A[1], A[3], \dots]$

~~FFT(A)~~

$E = FFT(A_e, \omega^2)$

$O = FFT(A_o, \omega^2)$

for i in range ($m-1$):

~~$A[i] = E[i/2] + \omega^2 O[i/2]$~~

~~# E and O are half the size of A, hence we~~

~~# must read at half the index~~

for i in range $((m-1)/2)$:

$A[i] = E[i] + \omega^{i/2} O[i]$

$A[m/2+i] = E[i] - \omega^{i/2} O[i]$

~~# Check penultimate page for the formula~~

Now, the second step in our algorithm (check ^{antipenultimate} page) runs in $O(n \log n)$

In step 1, b. we pick $m \geq 2n-1$ where m is a power of 2, which are just the m^{th} roots of unity.

I'm lost \rightarrow uuu [We're fucked]
 $\#$ Interpolation

$$[M] \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} c(1) \\ c(w) \\ \vdots \\ c(w^{m-1}) \end{bmatrix}$$



$$m_{ij} = w^{ij}$$

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{bmatrix} = M^{-1} \begin{bmatrix} c(1) \\ c(w) \\ \vdots \\ c(w^{m-1}) \end{bmatrix}$$

$\Rightarrow M^{-1} = \frac{1}{m} M(w^{-1}) \cdot [Yum Yum]$

or
 $[Yum]^*$

$$[w^{ij}]_{m \times m} \times [w^{-i}]_{m \times m}$$

$$= \begin{bmatrix} m & * & & \\ m & m & & \\ & & m & \\ & & & \ddots \end{bmatrix}$$

This (somehow! some fucking how!) makes the last step $O(n \log n)$

Karatsuba's Algorithm:-

Large integer multiplication [Two n-bit +ve integers]

Normal multiplication takes $O(n^2)$

Fastest way to multiply large integers is still FFT by converting the no. into a polynomial $f(x)$ where $f(2) = \text{int.}$

Sorting Detour:-

Assuming all numbers are bounded.

Countsort :- Store $\underset{\text{count}(x_i)}{\downarrow}$ in an array at loc x_i

Radix sort:- Apply countsort on every position, units, tens, etc. until the largest place.

End detour :-

Multiplying 2 complex numbers $(a_0+ib)(x+iy)$

$$\Rightarrow ax - by + i(bx + ay)$$

Looks like it needs 4 real products, but one can just use 3 :-

i) $p_1 = (x_1 + y_1)(x_2 + y_2)$

ii) $p_2 = x_1 \cdot x_2$

iii) $p_3 = y_1 \cdot y_2$

Similarly, for n bit numbers a & b

$$a = a_L \cdot 2^{n/2} + a_R$$

$$b = b_L \cdot 2^{n/2} + b_R$$

$$ab = a_L b_L 2^n + 2^{n/2} (a_L b_R + a_R b_L) + a_R b_R$$

We can use this to establish a recurrence, but it would be $O(n^2)$. Instead we apply Gauss' Logic, reducing it to 3 multiplications:-

$$P_1 = (a_1 + a_2)(b_1 + b_2)$$

$$P_2 = a_1 b_1$$

$$P_3 = a_2 b_2$$

$$\text{and } ab = P_2 2^n + 2^{n/2} [P_1 - P_2 - P_3] + P_3$$

Recurrence on this results in

$$\begin{aligned} T(n) &= 3 T(n/2) + O(n) \\ &= O(n \log n) \end{aligned}$$

$$\begin{aligned} T(n) &= 3 T(n/2) + O(n) \\ &= 3 [3 T(n/4) + O(n/2)] + O(n) \\ &= 3^2 T(n/2^2) + 3^{n/2} + n \\ &= \dots \\ &= 3^i T(n/2^i) + 3^i n/2^i + n \\ &= 3^i T(n/2^i) + n \sum_{j=0}^{i-1} 3^j / 2^j \end{aligned}$$

$$\text{Let } i = \log_2 n$$

$$\begin{aligned} T(n) &= 3^{\log_2 n} + n \sum_{j=0}^{\log_2 n - 1} 3^j \\ &\approx O(n \log_2^3 n) \end{aligned}$$

Strassen's Algorithm:-

$$\begin{bmatrix} A \\ B \end{bmatrix}_{n \times n} \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}_{n \times n} \quad \begin{bmatrix} E & F \\ G & H \end{bmatrix}_{n \times n}$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}_{n \times n}$$

$$T(n) = 8 T\left(\frac{n}{2}\right) + O(n^2) \rightarrow \text{Addition}$$

$$= 8 \left[8 T\left(\frac{n}{4}\right) + O\left(\left(\frac{n}{2}\right)^2\right) \right] + O(n^2)$$

=

$$\approx O(n^3)$$

BAD! :<

Strassen's ~~technique~~ technique :- He does it in 7 multi

$$\begin{aligned} T(n) &= 7(T\left(\frac{n}{2}\right)) + O(n^2) \\ &= \dots \\ &= O(n^{\log_2 7}) \end{aligned}$$

The product would then be

$$\left[\begin{array}{c|c} P_5 + P_6 + P_4 - P_2 & P_1 + P_2 \\ \hline P_3 + P_4 & P_5 - P_7 + P_1 - P_3 \end{array} \right]$$

$$P_1 = A(F-H)$$

$$P_2 = (A+B)H$$

$$P_3 = (C+D)E$$

$$P_4 = D(G-E)$$

$$P_5 = (A+D)(E+H)$$

$$P_6 = (B+D)(G+H)$$

$$P_7 = (A-C)(E+F)$$

Fibonacci?

$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0$$

$$F_1 = 1$$

Recursion $\rightarrow 2^n \rightarrow$ bad :<

we have $O(n)$ additions.

Iteration $\rightarrow n \rightarrow$ okay

\hookrightarrow but for larger n , add becomes $O(n)$

$\hookrightarrow \underline{\underline{n^2}}$ for large n

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

⋮

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

\hookrightarrow log₂ multiplications

Greedy Algorithms

Used in optimization problems

Ex :- Shortest path - single source, Huffman encoding.

Optimum Substructure Property:- Subparts of a problem must also be using the most efficient solution.

If $S \rightarrow V$ is the shortest path, then $S \rightarrow T \Rightarrow S \rightarrow V + V \rightarrow T$,

In such cases we usually ~~reverse~~ given the first step to be taken.

Greedy-Choice Property :- Picking the right first step.

We need both Optimum Substructure and Greedy Choice to have an efficient Greedy Algo

If only the Optimum Substructure holds, we try out all possibilities, but this is Slow, unless we have :-

Subproblems

Overlapping Substructure Property: If we are able to create a DAG with the subproblems. We topsort first and solve subproblems in toporder

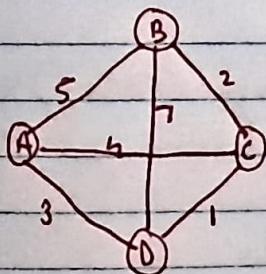
Greedy is a spl case of DP where the DAG is a path.

Minimum Spanning Trees:

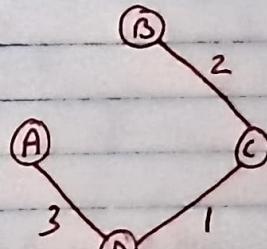
Input:- Edge Weighted Simple Undirected Path.

Subgraph of a tree on all its vertices

Cost of tree = \sum edge wts.



$$\text{Cost} := 21$$



$$\text{Cost} = 6 \rightarrow \text{MST}$$

Output:- Least cost spanning Tree T of G .

The Cut Property :-

Let $G = (V, E)$.

Suppose $X \subseteq E$ belongs to some MST of G .

Then let V be divided into S and V/S such that
 \nexists no edge spans from $S \rightarrow V/S$ or vice versa.

Then the least cost edge e b/w S & V/S is s.t
 $X \cup \{e\}$ is part of some MST of G .

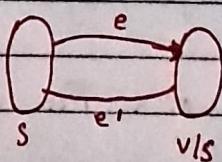
Algo:- [Kruskal]

Sort edges by wt.

Add edges in that order until unless cycle is formed
 in MST.

Proof:- Suppose $X \cup \{e\}$ is part of T if X is a part of T .

$$\begin{aligned} e &\notin E(T) \\ e' \in T \rightarrow e' &\in \text{cut} \\ \text{wt}(e) &\leq \text{wt}(e') \end{aligned}$$



$(T \cup \{e\}) \setminus \{e'\}$ has to be connected cause
 $T \cup \{e\}$ formed a cycle.

↪ had n edges, so the graph $(T \cup \{e\}) \setminus \{e'\}$
 has $n-1$ edges and is a tree \rightarrow is. tree T'

$$\begin{aligned} \text{cost}(T') &= \text{cost}(T) + \text{wt}(e) - \text{wt}(e') \\ &\leq \text{cost}(T) \end{aligned}$$

Since T is an MST, $\text{cost}(T') = \text{cost}(T)$
 Hence T' is an MST