INTRODUCTION

- Start Well - Know the assumptions / limitations
- Understand the obstacles - The right adversary and interferance
- Flow Well - Techniques
- End Well - Be proactive

Assumptions:

① (Machines are not omniscient)
In finite amount of space, only finite amount of information can be stored

② (Machines are not omnipresent)
Information travels at a finite speed.

③ (Machines are not omnipotent)
In finite length of programme, only finite control instructions can be written.

Turing Machine

Turing Machine is a 7 tuple $\langle Q, \Sigma, \Gamma, \delta, q_{start}, q_{acc}, q_{rej} \rangle$

Q: finite set of states

Σ: finite alphabet set (for input)

Γ: finite

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

$q_{start} \in Q$: Initial State of the machine.

$q_{acc} \in Q$:

$q_{rej} \in Q$:

Church-Turing Hypothesis:

An algorithm is a Turing Machine

- There are problems for which programs do not exist. So, we prove $\underbrace{\text{\# programs}}_{\text{countable}} < \underbrace{\text{\# problems}}_{\text{uncountable}}$.

Bijection : $f : N \to A$ then A is countable

A program is a finite length binary string $\Rightarrow$ countable

$A = \{0,1\}^*$

$A = \{\epsilon, 0, 1, 00, 01, 10, 00, 000, \ldots, 111, \ldots\}$ $\Rightarrow$ countable.

$f(1)$: first c program

$f(2)$: second c program

- **Diagonalization Technique**

Theorem : $(0,1)$ is uncountable.

$\therefore$ We need to show that a bijection does not exist.

Proof : Suppose the contrary

Let $f : N \to (0,1)$ be a bijection

$$\left.\begin{array}{l} f(1) = 0.d_{11}d_{12}d_{13}\ldots \\ f(2) = 0.d_{21}d_{22}d_{23}\ldots \\ f(3) = 0.d_{31}d_{32}d_{33}\ldots \end{array}\right\} \text{will differ in atleast one location.}$$

Aim: $\exists x \in (0,1)$ such that $\forall i \in N$, $f(i) \neq x$.

So we want to prove that this is into. Thus, it cannot be a bijection.

$x = 0.x_1x_2x_3\ldots$

$x_1 \neq d_{11}$ $(\neq 0,9)$

$x_2 \neq d_{22}$ $(\neq 0,9)$

$x_3 \neq d_{33}$ "

$\forall j \in N$ $x_j \neq d_{jj}$

$\therefore \forall i$, x differs from $f(i)$ in the i^{th} position.

$\therefore f(i) \neq x$

$\therefore$ This isn't onto.

$\therefore$ f is not a bijection.

$\therefore$ Real Numbers are uncountable.

- We now prove that #problems is uncountable

Consider : Input - Natural no n Output - Boolean

Given n, is n even? $\Rightarrow$ Does $n \in \{2,4,6,8,\ldots\}$

Given n, is it a power of 2? $\Rightarrow$ Does $n \in \{2,4,8,\ldots\}$

Given n, is n a prime? $\Rightarrow$ Does $n \in \{2,3,5,7,\ldots\}$

So, we are looking at subsets of N.

Theorem : $P(N)$ is uncountable [$P(N)$ is power set]

Proof : Suppose not. Let $f: N \to P(N)$

lets consider each subset to be a binary string

E : 010101010...

2^* : 010100010...

Prime : 01101010 ...

$f(1) = b_{11} b_{12} b_{13} b_{14} \dots$

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

Using Diagonalization Technique,

$S \subseteq N$

$S = p_1 p_2 p_3 \dots \quad p_j = \overline{b_{jj}}$

$\forall j \in N, \; S \neq f(j)$

$\therefore$ Bijection does not exist

$\Rightarrow$ Contradiction

$\Rightarrow P(N)$ is uncountable

$\Rightarrow$ #problems is uncountable.

- Types of problems -
  - decidable - solved (finite steps)
  - undecidable - program
  - unrecognizable

Problem of YES

WAP to input a C program M and its input w, and decide if the answer is yes.

Theorem : Problem YES is undecidable.

Proof : Suppose some code H solves YES problem.

$$H(M, w) = \begin{cases} \text{Yes} & \text{if } M(w) = \text{yes} \\ \text{No} & \text{otherwise} \end{cases}$$

D : on input M
- Runs $H(M, \langle M \rangle)$
- If H says yes, says NO
- Else if H says no, says YES

What's D(D)?

Yes: $H(D, \langle D \rangle) = \text{No}$ $\Downarrow$ $D(D) \neq \text{Yes}$

No: $H(D, \langle D \rangle) = \text{Yes}$ $\Downarrow$ $D(D) \neq \text{Yes}$

$\therefore$ D can neither give yes nor No

D never halts.

10/8/18

DIVIDE & CONQUER

① Merge Sort

② Multiplication

- Two complex nos -

$(a+ib)(c+id) = \underbrace{(ac-bd)}_{P_1-P_2} + i\underbrace{(ad+bc)}_{P_3-P_1-P_2}$ (4 multiplications)

$P_1 = ac \qquad P_2 = bd \qquad P_3 = (a+b)(c+d)$

- Integer -

D - $d_{n-1} d_{n-2} \dots d_2 d_1 d_0$

E - $e_{n-1} e_{n-2} \dots e_2 e_1 e_0$

$O(n^2)$ multiplication

$D = D_L (B)^{n/2} + D_R$ $\qquad 1539 = (15)(10)^2 + 39$

$E = E_L (B)^{n/2} + E_R$

$D * E = (B)^n D_L E_L + (B)^{n/2}(D_L E_R + D_R E_L) + D_R E_R$

$P_1 = D_L E_L \qquad P_2 = D_R E_R \qquad P_3 = (D_L + D_R)(E_L + E_R)$

$DE = P_1 (B^n) + P_2 + (B)^{n/2}(P_3 - P_1 - P_2)$

$T(n) = 3T(n/2) + O(n)$

$O(n^{\log_2 3})$ from $O(n^{\log_2 4})$

- Two polynomials

$p(x) = \sum_{i=0}^{n-1} p_i x^i$

$q(x) = \sum_{i=0}^{n-1} q_i x^i$

$p(x) \cdot q(x) = \sum_{i=0}^{2n-1} r_i x^i \qquad r_i = \sum_{k=0}^{i} p_i q_{i-k}$

$p(x) = p_e(x)^2 + x p_o(x)^2$

$q(x) = q_e(x)^2 + x q_o(x)^2$

	×	
$p(1)$	$q(1)$	$r(1)$
$p(2)$	$q(2)$	$r(2)$
⋮		
$p(n)$	$q(n)$	$r(n)$
⋮		
$p(2n-1)$	$q(2n-1)$	$r(2n-1)$

Coeff of $p(x) \xrightarrow{\text{Eval}} [p(1)\ p(2) \dots p(2n-1)]$

" " $q(x) \xrightarrow{\text{Eval}} [q(1)\ q(2) \dots q(2n-1)]$

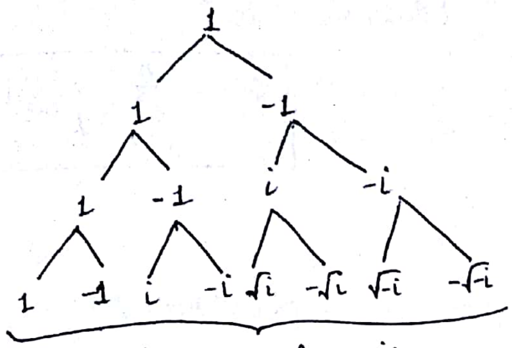
" " $r(x) \xleftarrow[\text{polate}]{\text{Inter-}} [r(1)\ r(2) \dots r(2n-1)]$

$O(\text{Evaluation} + \text{Interpolation}) = O(n^2)$

$p(x) = p_e(x^2) + x p_o(x^2)$

$p_e(x) = p_{ee}(x^2) + x p_{eo}(x^2)$

Evaluate on 1 and -1.



n^{th} root of unity

$[a_n \; a_{n-1} \; a_{n-2} \ldots a_0] \longrightarrow [p(\omega^0) \; p(\omega^1) p(\omega^2) \ldots p(\omega^n)]$

ω is n^{th} root of unity

Discrete Fourier Transform

$$\begin{array}{c} \\ 0 \\ \vdots \\ \vdots \\ n \end{array}\underbrace{\begin{array}{c} 0 \; \cdots \; n \\ \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^n \\ 1 & \omega^2 & \omega^4 & \omega^{2n} \\ & & & \end{bmatrix}\end{array}}_{M_{ij} = \omega^{ij}} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(\omega) \\ \vdots \\ p(\omega^n) \end{bmatrix}$$

$M^{-1} = M \times \lambda$

Inverse of Fast Fourier Transform = Fast Fourier Transform

Product of 2 numbers in $O(n)$?

$n =$ product of primes

$a = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$

$b = p_1^{\beta_1} p_2^{\beta_2} \ldots p_k^{\beta_k}$

$a \cdot b = p_1^{\alpha_1+\beta_1} \cdot p_2^{\alpha_2+\beta_2} \ldots p_k^{\alpha_k+\beta_k}$

③ Fast Fourier Transform

Input: coeff array $A = [a_0 a_1 a_2 \ldots a_n]$

Output: evaluated array $E = [e_0 e_1 e_2 \ldots e_n]$

$$\boxed{e_i = \sum_{j=0}^{n} a_j \omega^j}$$

$p(x) \cdot q(x)$

$\curvearrowright n\log n$

$p(x) = p(\omega^0), p(\omega^1), \ldots, p(\omega^n)$

$q(x) = q(\omega^0), q(\omega^1), \ldots, q(\omega^n)$

$p(x)q(x) - p(\omega^0)q(\omega^0), p(\omega^1)q(\omega^1), \ldots, p(\omega^n)q(\omega^n)$

FFT$([a_0, a_1, \ldots, a_n], \omega)$

if $n == 0$, return a.

else{ ① $[s_0, s_1, \ldots, s_{n/2}]$ = FFT $[A_e, \omega^2]$

② $[t_0, t_1, \ldots, t_{n/2}]$ = FFT $[A_o, \omega^2]$

③ $[e_0, e_1, \ldots, e_n]$ =

for $j = 0$ to $n \rightarrow e_j = s_j + \omega^j t_j$

}

output: $e_0\ e_1 \ldots e_n$

$A_e = [a_0\ a_2 \ldots a_n]$

$A_o = [a_1\ a_3 \ldots a_{n-1}]$

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

$$\boxed{T(n) = O(n\log n)}$$

④ Median / Selection of k^{th} ranked element

Select (A, k)

$A = [\quad ? \quad]$

$[A_L][A_v][A_R]$

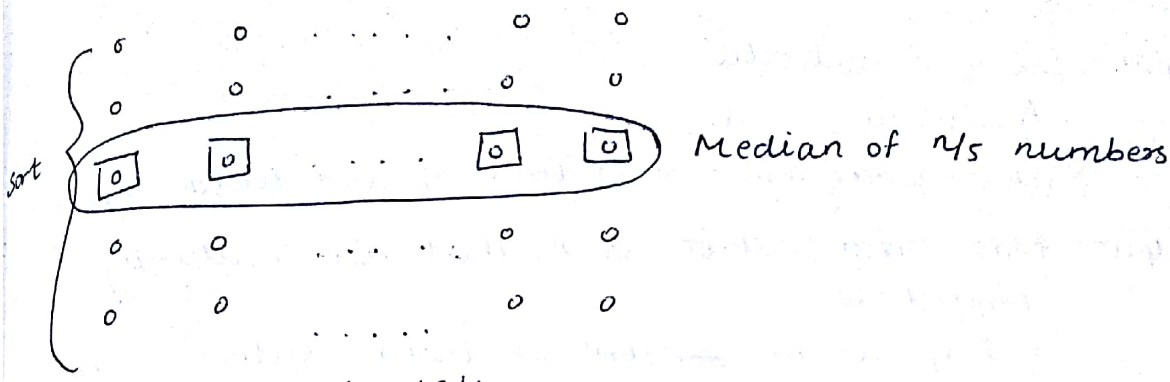
$$\text{Select}(A,k) = \begin{cases} \text{select}(A_L, k) & \text{if } |A_L| \geq k \\ \text{select}(A_v, k - |A_L|) & \text{if } |A_L| < k \leq |A_L| + |A_v| \\ \text{select}(A_R, k - |A_L| - |A_v| & \text{if } k > |A_L| + |A_v| \end{cases}$$

Worst case : $T(n) = T(n-1) + O(n)$

$$\boxed{T(n) = O(n^2)}$$

Median of Medians

$n/5$ partitions



$v \leftarrow \text{Select}\left(B, \frac{|B|}{2}\right)$

Run-time : $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$

Suppose $T(n) = c\, O(n)$

Let some ϵ

such that, $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + \epsilon . N$

Then for some c,

$$c \cdot T(n) \leq c\left(T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + \epsilon N\right)$$

$$cn = \frac{cn}{5} + c\left(\frac{7n}{10}\right) + \epsilon N$$

$$c = 10\epsilon_0$$

$v \leftarrow$ a random element of A

If $\frac{n}{4} \leq \text{rank}(v) \leq \frac{3n}{4}$ then $T(n) = T\left(\frac{3n}{4}\right) + O(n)$

Fingerprinting Algo

$C = AB$

$C\bar{r} = \underbrace{A\underbrace{B\bar{r}}}$ $\qquad \bar{r} = n \times 1$

17/8/18

GREEDY

Activity Selection Problem

Input: Set of n activities

$A = \{a_1, a_2, \ldots, a_n\}$

Each activity has a start time s_i and finish time f_i

Output: Max sized subset of A that are mutually compatible

a_i & a_j can be ~~selected~~ scheduled together iff they do not overlap i.e $f_{i} \leq s_i$

Ordered an ascending order of f_i

$f_1 < f_2 < f_3 \ldots < f_n$

```
S ← {a1}
l ← 1
for i = 2 to n
{
    if (si ≥ fl)
    {
        S ← S U {ai}
        l ← i
    }
}
output S
```

Theorem - Algo has 'Greedy choice Property'

Proof : Suppose not

$B = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$

$B' = \{B \setminus \{a_{i_1}\}\} \cup \{a_1\}$

$f_1 \leq f_{i_1}$

$f_{i_1} \leq s_{i_2}$

$f_1 \leq s_{i_2}$

Theorem Algo has 'Optimum Substructure Property'

$A' = \{a_i \mid a_i$ doesn't overlap with $a_1\}$

$\downarrow$

$S = S^c - \{a_1\}$ if $S = \{a_1 \boxed{}\}$

(maximum sized activities)

- Greedy can always be made into iterative code (It has failed recursion)
- If recursion works ⇒ dynamic programming

Hoffman Codes

A string - abbaaabbcbbbdd

a - 00		a = 10
b - 01	30 bits	b = 0
c - 10		c = 110
d - 11		d = 111

. Can it be stored in less than 30 bits in any type of encoding?

. Encoding should be prefix free.

. Get the best encoding

Let $s_1, s_2, \ldots, s_n$ be the unique characters

$s_1 \rightarrow f_1$ $\qquad$ $f[a] = 4$

$s_2 \rightarrow f_2$ $\qquad$ $f[b] = 8$

$\vdots$ $\qquad$ $f[c] = 1$

$s_n \rightarrow f_n$ $\qquad$ $f[d] = 2$

- Sort the frequencies in ascending order

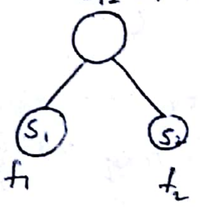
c d a b

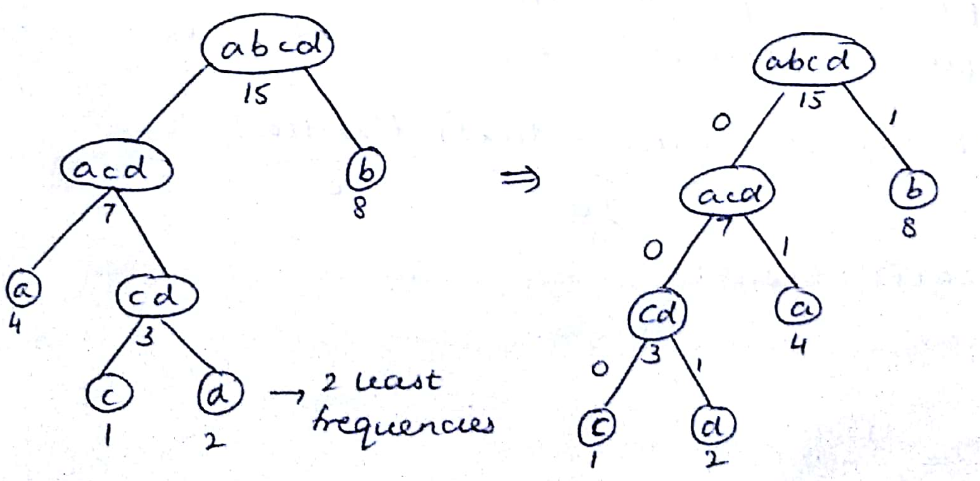
such that $f_1 \leq f_2 \leq \ldots \leq f_n$

- Build Hoffman Tree

$s_1 \rightarrow f_1$ and $s_2 \rightarrow f_2$

$S_{12} \rightarrow f_1 + f_2$





Hoffman Tree

Encode left as 1, right as 0. Then the shortest path from root to each leaf is the required code.

$c \to 000$

$b \to 1$

$a \to 01$

$d \to 001$

suffix freeness in a tree is implicit

Proof: $\text{cost} = \sum_{x:\text{leaf}} \underbrace{f(x)}_{\text{freq}} \underbrace{d(x)}_{\text{depth of } x}$

We have to minimise this cost

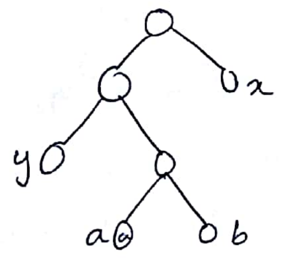
The 2 least frequently occuring leaves are of same length and differ in LSB.

Theorem: Hoffman codes have the 'Greedy Choice Property' i.e among all possible trees $\exists$ an optimal tree with x and y as max depth siblings where x and y are of least frequency.

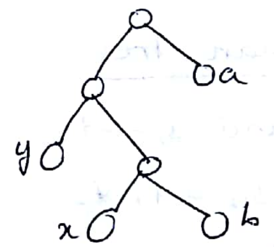
$f(x) \leq f(y)$ (let)

$f(a) \leq f(b)$

T



T' (swap a & x)



$\text{cost}(T') = \text{depth}_2(x) + \text{freq}(x) + \text{depth}_2(a) + \text{freq}(a) + \ldots$

$\text{cost}(T) = \text{depth}_1(a) + \text{freq}(a) + \text{depth}_1(x) + \text{freq}(x) + \ldots$

$\text{depth}_2(x) = \text{depth}_1(a) = d_T(a)$

$\text{depth}_2(a) = \text{depth}_1(x) = d_T(x)$

$\text{cost}(T') - \text{cost}(T) = \underbrace{(d_T(a) - d_T(x))}_{\geq 0} \underbrace{(f(x) - f(a))}_{\leq 0}$

$\text{cost}(T') \leq \text{cost}(T)$

Optimum Substructure

Remove last 2, their parent is (xy)

x, y removed $\rightarrow (xy)$

$$s' = [s \setminus \{f(x,y)\}] \cup \{xy\}$$

$$f(xy) = f(x) + f(y)$$

$$cost(T) = \sum_{x:leaf} f(x) d_x$$

$$cost(T') = \sum_{x: leaf} f(x) d_x$$

$$cost(T) - cost(T') = d_x f(x) + d_y \cdot f(y) - d(xy) f(xy)$$

$$= d_x [f(x) + f(y)] - (d_x - 1) \underbrace{f(xy)}_{f(x)+f(y)}$$

$$cost(T) - cost(T') = f(x) + f(y)$$

T was not optimum $\rightarrow$ T' was not optimum $\rightarrow T''$ not optimum

$\boxed{\text{Approximate Algorithm}}$

Min Set-Cover Problem

Set S

Family $F = \{S_1, S_2, \ldots, S_n\}$

$S_i \subseteq S$

Output: Indices $i_1, i_2, \ldots, i_k$

such that $\bigcup_j S_j = S$

Greedy Algo:

$I = \phi$

$U = S$

Repeat

- Choose a set S_j from F that ~~comes~~ covers the maximum number of elements in U
- $I \leftarrow I \cup \{i_j\}$
  $U \leftarrow U \setminus S_j$

However, Greedy doesn't give the right answer in this case.

But it gives an answer very close to the actual answer.

MATROID THEORY :

$M = \langle S, F \rangle$

a) $S \neq \phi$

b) Hereditary Property:

if $A \in F$, $\forall B \subseteq A \Rightarrow B \in F$

c) Exchange Property:

if $A \in F$, $B \in F$

$\exists x \in B \setminus A$ such that

$A \cup \{x\} \in F$

Every element of F is called independent set.

Finding a <u>max weighted</u> independent set:

$S = \{S_1, S_2, \ldots, S_n\}$

S_i has a weight $w_i > 0$

<u>Minimum Spanning Tree</u>

Input: Undirected graph $G = (V, E)$. Edges have weights

Output: MST of G.

$M_G = \langle E, F_G \rangle$

$F_G = \{A \subseteq E \mid A \text{ is acyclic}\}$

a) $E \neq \phi$

b) Hereditary Property holds.

c) $A \in F_G$ $B \in F_G$ Graph(acyclic)-forest

$|A| < |B|$

$(n - |A|) > (A - |B|)$

<u>Theorem</u>: Any forest with n nodes and t trees has exactly $(n-t)$ edges

$\boxed{M_G = \langle E, F_G \rangle \text{ is a matroid}}$

use greedy algorithm

$w_i' = W - w_i$

$F_G = \{A \subseteq E \mid A \text{ is acyclic}\}$

$M = \langle S, F \rangle$

w_i = weight of $s_i \in S$, $w_i > 0$ (A is maximal)

$\nexists\, x.$ such that $A \cup \{x\} \in F$

Sort the elements of S in non-decreasing order of weights.

$A = \phi$

Choose the next s_i (in that order)

if $A \cup \{s_i\} \in F$

$A \leftarrow A \cup \{s_i\}$

output A

$\exists$ A such that $x \in A$

Let B be an optimal solution, $x \in B$ ($B \in F$)

$S = \{ y \mid y \cup \{x\} \in F \}$

$F' = \{ A \in F \mid A \cap \{x\} \neq 0 \}$