A Turing machine is a 7-tuple
$< Q, \Sigma, \Gamma, \delta, q_{start}, q_{acc}, q_{rej} >$

Q - finite set of states
Σ - finite alphabet set (for input)
Γ - finite tape symbol alphabet set
$(\Gamma \supseteq \Sigma)$ $[\forall \in \Gamma, \forall \notin \Sigma]$

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$
left right

$Q = \{q_0, q_1\}$
$\Gamma = \{a, b\}$
$\delta(q_0, a) = (q_1, b, L)$
$\delta(q_0, b) = (q_0, a, R)$

$q_{start} \in Q$ - Initial state of machine
$q_{acc} \in Q$ - Accept
$q_{rej} \in Q$ - Reject

CHURCH - TURING HYPOTHESIS

An algorithm is a Turing machine.

- Machines are not omniscient
- In finite space, only finite info
- Machines are not omnipresent
- Info travels at finite speed

• Machines are not omnipotent
In a program of final length, only finite control instruction

Q) Show that there are problems for which C programs exist (Turing machines don't exist)

Number of C programs = N
" " computational problems = P
s.t. $P >> N$

$f: N \to A$
If f is bijective, A is countable

Theorem: Z is countable
Proof: $f: N \to Z$

$$f(x) = \begin{cases} 0 & \text{if } x = 1 \\ x/2 & \text{if } x \text{ is even} \\ -\left(\frac{x-1}{2}\right) & \text{if } x \text{ is odd} \end{cases}$$

It is bijective
No. of natural nos. = number of integers

$\nexists$ bijection between set of all programs (countable) & set of all problems (uncountable)
R is uncountable

Q) Write a ~~computational~~ program/problem such that no program exists

Q) Show that the number of C programs is countable

C programs are finite length binary strings
$\{0,1\}^*$

Scanned by CamScanner

C programs are a subset of A.

The strings possible from set A are

ϵ, 0, 1, 00, 01, 10, . . .

(null string)

$A = \{\epsilon, 0, 1, 00, 01, 10, \dots\}$

$f(i)$ is the i^{th} string occurring in the enumeration and is hence a bijection.

$\therefore$ A is countable.

A subset of A would be the C programs (since not all element of A might be a C program)

Q) Are there programs that can have $O(n^2)$ complexity but not $O(n^{2-\epsilon})$? [Use diagonalization technique]

DIAGONALIZATION

Theorem: No. of reals nos. between $(0,1)$ is uncountable

Proof: Suppose the contrary.

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

$\Rightarrow f(1) = 0.d_{11}\, d_{12}\, d_{13} \dots$

$f(2) = 0.d_{21}\, d_{22}\, d_{23} \dots$

$f(3) = 0.d_{31}\, d_{32}\, d_{33} \dots$

and so on.

T.S.T $\exists x \in (0,1)$ s.t. $\forall i \in N, f(i) \neq x$

Let $x = 0.x_1\, x_2\, x_3\, x_4 \dots$

where $x_1 \neq d_{11}$ and $x_1 \neq 0$ or 9 (since we don't want $x = 0.00\dots$ or $x = 0.999\dots$ since in that case $x = 1$ or 0 and x is between 0 and 1) since $x_1 \neq d_{11}$, $x \neq x_1 f(1)$

$x_2 \neq d_{22} \Rightarrow x \neq f(2)$

$x_3 \neq d_{33} \Rightarrow x \neq f(3)$

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

$\therefore x \neq f(1), f(2), f(3) \ldots f(n) \ldots$

Therefore x does not appear in the range and is not onto and contradic our assumption that it is bijecti is wrong.

This is proof by diagonalization

Q) Write a problem for there is no program

Ans) We'll assume only problems where

Input : natural no. n

Output : Boolean

and show it is uncountable

Problems can be those like :

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

Is n a power of 2? Does $n \in \{2, 4, 8, 16 \ldots\}$?

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

So we ask question whether n belongs to some particular subset. Therefore a subset of $\mathbb{N}$ is a problem.

Theorem : The power set of $\mathbb{N}$, $P(\mathbb{N})$ is uncountable

(we can prove this using diagonalization or by showing a bijection between this and set of real nos.)

Proof : Let $f : \mathbb{N} \rightarrow P(\mathbb{N})$ be a bijection subsets can be represented as a binary string

eg if $\{2,3\} \subseteq \{1,2,3,4\}$ can be represented as 0110

Therefore a subset of $\mathbb{N}$ can be represented as a binary string

eg set of even nos. $E = 010101\ldots$

set of powers of 2 $= 010100010\ldots$

set of primes $= 01101010\ldots$

set $f(1) = b_{11} b_{12} b_{13} \ldots$ (string representation)

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

Find a subset S of $\mathbb{N}$ $(S \subseteq \mathbb{N})$ such that S is not in range.

$S = s_1 s_2 s_3 \ldots$

$s_j = \overline{b_{jj}}$ (complement)

$\forall j \in \mathbb{N}, S \neq f(j)$

If these were in base 1 number system, an approach different to diagonalization will have to be used.

A problem is decidable if it can be solved in finite resources or steps.

Undecidable problem takes ∞ steps in the worst case. but program exist.

Unrecognizable problem is one where we can't write a program.

Q) Write a program to input a c program M and its input w &

decides if answer is yes. i.e., given a program and input and decide the answer

This problem is undecidable.

Proof: Suppose some code H solves this YES problem.

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

T.S.T H does not exist.

Let D be a program such that on input M,

- Runs $H(M, <M>)$
- If H says Yes, says No
- Else if H says No, says Yes.

D must exist because H exists (assuming). → A

What is D(D)?

If $D(D) = \text{Yes} \Rightarrow H(D, <D>)$ is an

$\Downarrow$

$D(D) = \text{No}$

(A contradiction)

If $D(D) = \text{No} \Rightarrow H(D, <D>) = \text{Yes}$

$\Downarrow$

$D(D) = \text{Yes}$

(A contradiction)

Therefore, D cannot exist which contradicts A

∴ H cannot exist and solve the problem.

REVIEW

- starting well
* - proving impossibility problems
- unifying problems

- greedy algo (matroid theory)
- Dynamic programming
- linear programming

DIVIDE AND CONQUER

say 2 complex nos. $(a+ib)$ and $(c+id)$

$(a+ib).(c+id) = (ac-bd) + i(ad+bc)$

There are 4 multiplications occuring. To multiply 3 times, use $P_1 = ac$, $P_2 = bd$, $P_3 = (a+b)(c+d)$

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

In integer multiplication,

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

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

where D and E are integers

(eg. $1539 = 15 \times 10^2 + 39$)

$D = (B)^{n/2} D_L + D_R$

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

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

Where B is the base

The recurrence relation is for D.E

$T(n) = 4T(n/2) + O(n)$

(masters theorem)

$T(n) = O(n^2)$ ~~t~~

Hence divide and conquer has failed us.

Let $P_1 = D_L E_L$

$P_2 = D_R E_R$

$P_3 = (D_L + D_R)(E_L + E_R) = D_L E_L + D_L E_R + D_R E_L + D_R E_R$

$\therefore D \cdot E = (B)^n P_1 + P_2 + B^{(n/2)}[P_3 - P_1 - P_2]$

Here $T(n) = 3T(n/2) + O(n)$ [Karatsuba algorithm

$T(n) = O(n^{\log_2 3})$

We can still do better using Fast Fourier Transform.

Consider 2 polynomials $p(x)$ and $q(x)$

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

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

} Product $p(x)\,q(x) = \sum_{i=0}^{2n} r_i x^i$

multiplying 2 polynomials is like multiplying 2 integers with x as the base

Naive approach, $r_i = \sum_{k=0}^{i} p_k q_{i-k} x^k$

This is $O(n^2)$

Consider a polynomial

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

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

all even degrees — all odd degrees

$5y^2 + 2y + 7$ }

$3y - 1$ } where $y = x^2$

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

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

where p_e and q_e are even degree polynomials.

coefficient of $p(x) \xrightarrow{\text{evaluate}} [p(1), p(2) \ldots p(2n+1)]$

" $q(x) \xrightarrow{\text{evaluate}} [q(1), q(2) \ldots q(2n+1)]$

" $r(x) \xleftarrow{\text{interpolate}} [r(1), r(2) \ldots q(2n+1)]$

interpolation ~~takes~~ and ~~O(n) but~~ evaluation takes $O(n^2)$. Pointwise multiplication takes $O(n)$

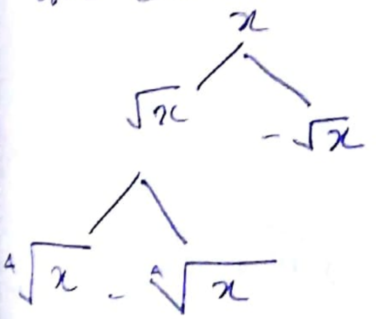
$p(x) = p_e(x^2) + x\, p_o(x^2)$

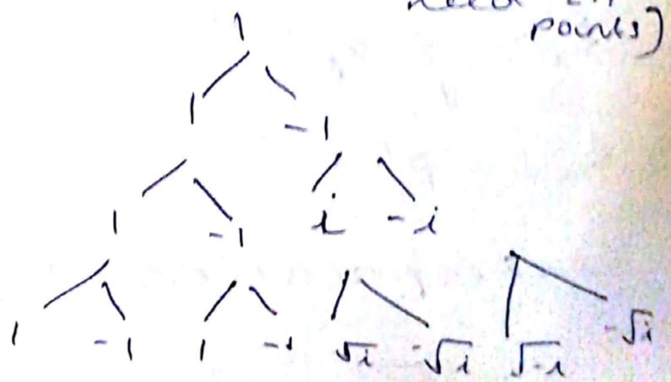
$p_e(x) = p_{ee}(x^2) + x\, p_{eo}(x^2)$

$p_e(1) = p_e(1^2) + x\, p_o(x^2)$

$p(1) = p_e(-1^2) + x\, p_o(-1^2)$

(we calculate $P(x)$ & $P(-x)$ because we need $2n$ points)





(n^{th} root of unity)

$[a_n\, a_{n-1}\, a_{n-2}, \ldots a_0]$
coefficient

$\longrightarrow [p(w_n^0), p(w^1), p(w^2), \ldots \ldots p(w^n)]$

where w is n^{th} root of unity

This is discrete fourier transform

$\longrightarrow$

$$\underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 \cdots & \omega^n \\ 1 & \omega^2 & \omega^4 \cdots & \omega^{2n} \end{bmatrix}}_{M} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} p(1) \\ p(\omega) \\ p(\omega^2) \\ \vdots \\ p(\omega^n) \end{bmatrix}$$

$$M_{ij} = \omega^{ij} \implies M^{-1} = \frac{1}{n} M(\omega^{-1})$$

The fastest speed is $O(n \log n \log \log n)$

Every no. can be represented as product of primes.

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

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

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

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

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

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

$$e_i = p(\omega^i)$$

Interpolation is just $FFT([a_0, a_1 \ldots a_n], \omega^{-1})$

$$\omega = \sqrt[n]{1}$$

$$\omega = e^{(i2\pi)/n}$$

Take n to be an exact power of 2. If it is not, just pad it with zeroes, $0x^{n+1}$

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

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

FFT ([$a_0, a_1 \ldots a_n$], ω)
if n == 0 return a_0

1. [s_0 s_1 . . $s_{n/2}$] = FFT [A_e, ω^2]
2. [t_0 t_1 . . . $t_{n/2}$] = FFT [A_o, ω^2]
3. [~~e_0 e_1 e_2 ... e_n~~]
   for j = 0 to n
   $e_j = s_j + \omega^j t_j$
   return ~~output~~ [e_0, e_1, . . . e_n]

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

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

How would you do integer multiplication using FFT ?

So far, divide and conquer:

1) merge sort
2) integer multiplication $O(n^{\log_2 3})$
3) FFT
4) selection of k^{th} rank element

Selecting the k^{th} ranked element:

Consider an array A in 3 parts [A_L] [A_v] [A_R]

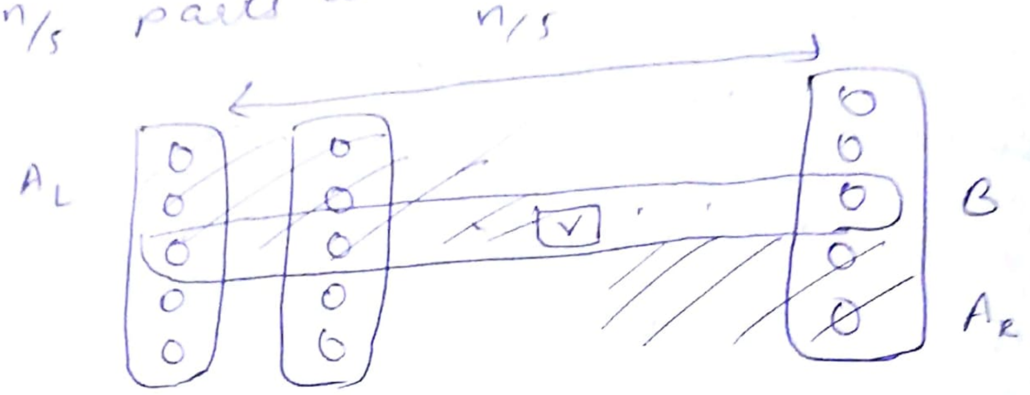
select (A, k) =
(i) select (A_L, k) if $|A_L| \geq k$
(ii) select (A_v, k - $|A_L|$) if $|A_L| < k \leq |A_L| + |A_v|$
(iii) select (A_R, k - $|A_L|$ - $|A_v|$) if $k > |A_L| + |A_v|$

$T(n) = T(n-1) + O$

However the queries

in the worst case $O(n^2)$
$A_L = n-1$ which is

so we use median of medians
First divide the array into
$n/5$ parts with 5 elements each



sort each of these $n/5$ blocks.

Now we ~~find~~ sort B and use
select $(B, |B|/2)$

$$|A_L| \geq \frac{3n}{10} \Rightarrow |A_R| \leq \frac{7n}{10}$$

$$|A_R| \geq \frac{3n}{10} \Rightarrow |A_L| \leq \frac{7n}{10}$$

$$T(n) = T(n/5) + T(7n/10) + O(n)$$

Substitution method:

$$T(n) \leq T(n/5) + T(7n/10) + \epsilon n$$

$T(n) = c.n$ (we claim its $O(n)$)

$$\Rightarrow c.n \leq c.\frac{n}{5} + c.\frac{7n}{10} + \epsilon n$$

$$\text{if } c = \frac{c}{5} + \frac{7c}{10} + \epsilon \Rightarrow c = 10\epsilon$$

Therefore there exists a value for c

Acc. to Masters theorem,

if $\alpha + \beta < 1$
$\Rightarrow T(n) = O(n)$

In quick sort, if v is a random element of A, then expected time is $O(n)$

Proof:

if $\frac{n}{4} < rank(v) \leq 3n/4$

[probability is 1/2 here]

Now at worst case, $|A_L| = \frac{3n}{4}$

~~$T(3n/4) + O(n)$~~

$T(n) = T(3n/4) + O(n)$
$\Rightarrow T(n) = O(n)$

No. of expected times = E

$E = 1 + \frac{1}{2} E$

$\Rightarrow E = 2$

$\therefore$ it is actually $T(n) = T(3n/4) + O(2n)$

Consider 2 matrices A and B
Let $C = AB$ where C is $n \times n$ matrice
Take random vector $\vec{r}$
of size $n \times 1$.
So $C \cdot \vec{r} = AB\vec{r}$ which takes $O(n^2)$. However if that comes out to be true, we can't completely say $C = AB$. So keep taking random values of $\vec{r}$.