TURING MACHINE

A 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 tape symbol alphabet set
$(\Gamma \supseteq \Sigma)$ $[\ \sqcup \in \Gamma, \sqcup \notin \Sigma\ ]$

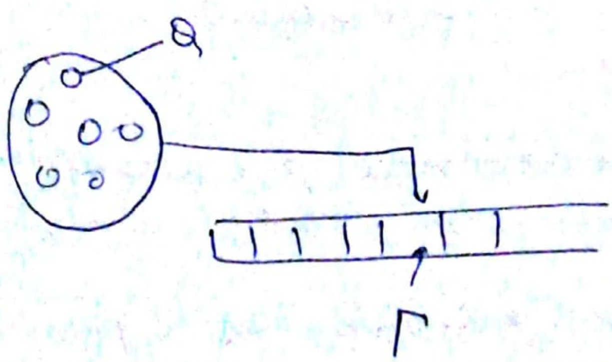
$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
L: left, R: right

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

$q_{start} \in Q$ - 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

$f: N \rightarrow A$

If f is bijective, A is countable

Theorem : Z is countable

Proof: $f: N \rightarrow Z$

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

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 program is countable.

Ans) C programs are finite length binary strings

C programs are a subset of A
The strings possible from set A are
$\epsilon, 0, 1, 00, 01, 10, \ldots$
Smallest string

$A = \{\epsilon, 0, 1, 00, 01, 10, \ldots\}$
$f(i)$ 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 $f: N \rightarrow (0,1)$ be a bijection
$\Rightarrow f(1) = 0.d_{11}\,d_{12}\,d_{13}\ldots$
$f(2) = 0.d_{21}\,d_{22}\,d_{23}\ldots$
$f(3) = 0.d_{31}\,d_{32}\,d_{33}\ldots$
and so on
T.S.T $\exists\, x \in (0,1)$ st $\forall i \in N, f(i) \neq x$

Let $x = 0.x_1\,x_2\,x_3\,x_4\ldots$
where $x_i \neq d_{ii}$ and $x_i \neq 0$ or 9 (since we don't want $x = 0.00\ldots$ or $x = 0.999\ldots$ since in that case $x = 1$ and x is between 0 and 1)

Since $x_1 \neq d_{11}$, $x \neq x_1 f(1)$

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

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

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

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

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...

set of powers of 2 : 010100010...

set of primes = 01101010...

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 $H(M, \langle M \rangle)$
- 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 $D(D) = \text{Yes} \Rightarrow H(D, \langle D \rangle)$ is a

$\Downarrow$

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

(A contradiction)

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

$\Downarrow$

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

(A contradiction)

Therefore, D cannot exist which contradicts (A)

∴ H cannot exist and solve this problem.

REVIEW

* starting week
  - 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 $P_1 = ac$, $P_2 = bd$, $P_3 = (a+b).(c+d)$

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

$\downarrow \quad P_1 - P_2$ $\qquad \downarrow \quad P_3 - P_1 - P_2$

In integer multiplication,

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

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

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

$T(n) = 4T(n/2) + O(n)$ for D.E

(solve using masters theorem)

$T(n) = O(n^2)$ [~~If T(n) = ...~~]

Hence divide and conquer has failed us.

Let $P_1 = D_1 E_1$

$P_2 = D_R E_R$

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

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

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

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

$= \underbrace{(5x^4 + 2x^2 + 7)}_{\text{all even degrees}} + \underbrace{(3x^3 - x)}_{\text{all odd degrees}}$

$5y^2 + 2y + 7$ } where $y = x^2$

$3y \ldots$

$p(x) = P_e(x^2) + x\ P_o(x^2)$

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

where P_e and q_e are even degree polynomials.

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

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

" $r(x) \xleftarrow{\text{Interpolate}} [r(1), r(2) \dots r(2n+1)]$

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

$p(x) = P_e(x^2) + x\ P_o(x^2)$

$P_e(x) = P_{ee}(x^2) + x\ P_{eo}(x^2)$

$P_a(1) = P_e(1^2) + x\ P_o(x^2)$

$p(1) = P_e(-1^2) + x\ P_o(-1^2)$

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

(nth root of unity)

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

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

where w is n^{th} root of unity

This is discrete fourier transform

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

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

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 $E = [e_0\ e_1 \ldots e_n]$

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

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

$$e_i = \sum_{j=0}^{n} a_j \omega^{ji}$$
$$e_i = p(\omega^i)$$

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

$$\omega = \sqrt[n]{1}$$
$$\omega = e^{(i2\pi)/n}$$

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

$$A_e = [a_0, a_2, \ldots\ldots a_n]$$
$$A_o = [a_1, a_3, \ldots\ldots a_{n-1}]$$

FFT ([$a_0, a_1, \ldots a_n$], ω)

if $n == 0$ return a_0

1. $[s_0\ s_1 \ldots s_{n/2}] = FFT[A_e, \omega^2]$
2. $[t_0\ t_1 \ldots t_{n/2}] = FFT[A_o, \omega^2]$
3. ~~$[e_0\ e_1\ e_2 \ldots e_n]$~~

for $j = 0$ to n

$e_j = s_j + \omega^j t_j$

return ~~output~~ $[e_0, e_1, \ldots e_n]$.

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

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

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 $[A_L]\ [A_v]\ [A_R]$

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

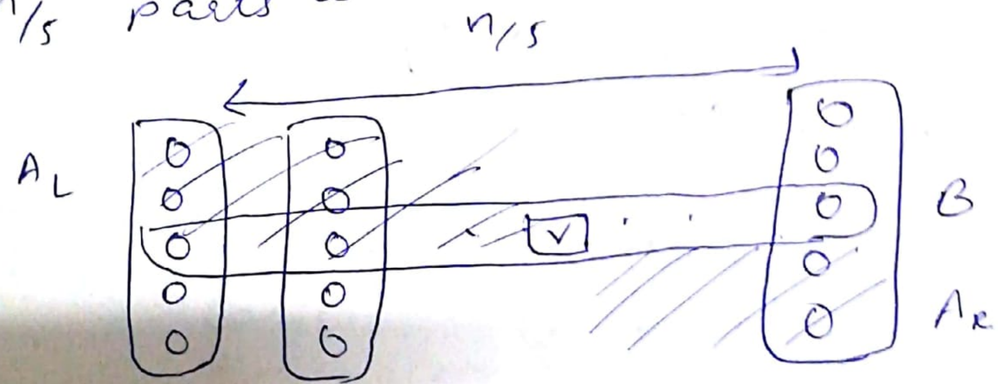
However this gives $T(n) = T(n-1) + O(n)$

in the worst case if $A_R = 0$, $A_V = 1$

$A_L = n-1$ which is $O(n^2)$

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 $O(n)$)

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

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

Therefore there exists a value for c

Acc. to Masters theorem

$$T(n) = T(\alpha n) + T(\beta n) + O(n)$$

so
$\Rightarrow T(n)' = O(n)$

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

$\cancel{T(3n/4) + O(n)}$

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

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

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 ~~a~~ random vector $\vec{r}$ ~~= A~~

of size $n \times 1$.

Do $C \cdot \vec{r} = AB\vec{r}$ 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 $\vec{r}$.