TURING MACHINE

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

$(\Gamma \supseteq \Sigma)$ $[\forall \in \Gamma, \# \notin \Sigma]$

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

(L: left, R: right)

$Q = \{q_0, q_1\}$

$\Gamma = \{a, b\}$

$\delta(q_0, a) = (q_1, b, L)$

$\delta(q_0, b) = (q_0, a, R)$

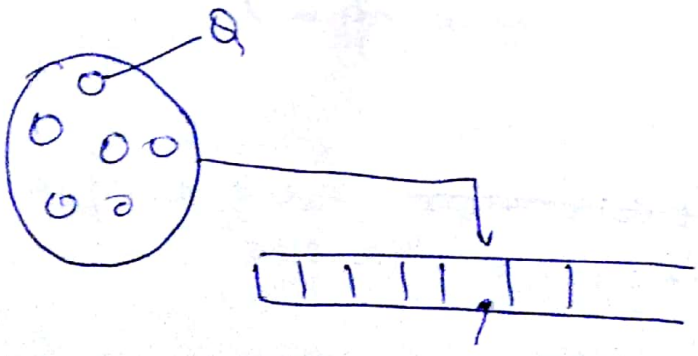
$q_{start} \in Q$ - Initial state of machine

$q_{acc} \in Q$ - Accept

$q_{rej} \in Q$ - Reject

CHURCH-TURING HYPOTHESIS

An algorithm is a Turing machine.



- Machines are not omnipotent
In a program of final length, only finite control instructions

Q) Show that there are problems for which no programs exist (Turing machines don't exist)

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

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

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

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

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.

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

C programs are a subset of A.
The strings possible from set A are

$\epsilon, 0, 1, 00, 01, 10, \ldots$

↳ null string

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

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

∴ 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

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

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

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

and so on.

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

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

where $x_1 \neq d_{11}$ and $x_1 \neq 0$ 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 $x_1 \neq d_{11}$, $x \neq x_1$, $f(1)$

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

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

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

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

Therefore x does not appear in the range and is not onto and contradicts our assumption that it is bijection 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 $f : \mathbb{N} \to P(\mathbb{N})$ 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 = 01010001 0...

Set of primes = 01101010...

Let $f(1) = b_{11} b_{12} b_{13} \ldots$ (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.

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

P.S.T H does not exist

Let D be a program such that on input M, D(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) = Yes ⇒ H(D, <D>) is an

⇓

D(D) = No

(↯ contradiction)

If D(D) = No ⇒ H(D, <D>) = Yes

⇓

D(D) = Yes

(↯ contradiction)

Therefore, D cannot exist which contradicts (A)

∴ H cannot exist and solve this 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 $(c+id)$

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

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

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

$$\downarrow \qquad\qquad \downarrow$$

$$P_1 - P_2 \qquad P_3 - P_1 - P_2$$

In integer multiplication,

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

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

where D and E are integers (eg. $1539 = 15\times10^2 + 39$)

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

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

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

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 $T(n) = aT(n/b)$, $O(n^{\log_b a})$]~~

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

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

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

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

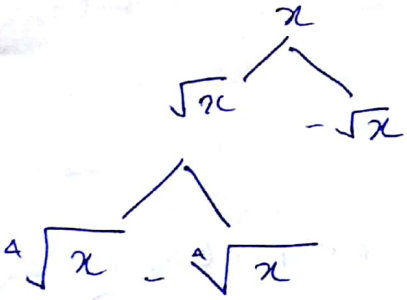
$p(x) = p_e(x^2) + x\, p_o(x^2)$

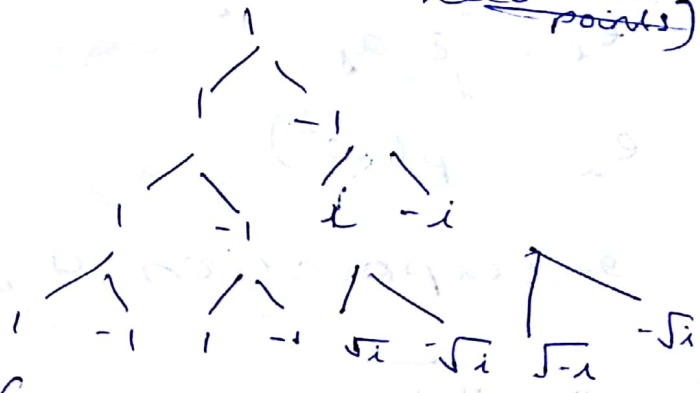
$p_e(x) = p_{ee}(x^2) + x\, p_{eo}(x^2)$

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

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

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





(n^{th} root of unity)

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

Coefficient $\longrightarrow [p(\omega_n^0), p(\omega^1), p(\omega^2), \dots$

$\dots p(\omega^n)]$

where ω is n^{th} root of unity

This is discrete fourier transform.

→

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

M

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

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

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

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

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

$e_i = p(\omega^i)$

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, $0x^{n+1}$

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

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

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 $[A_L]\ [A_r]\ [A_R]$

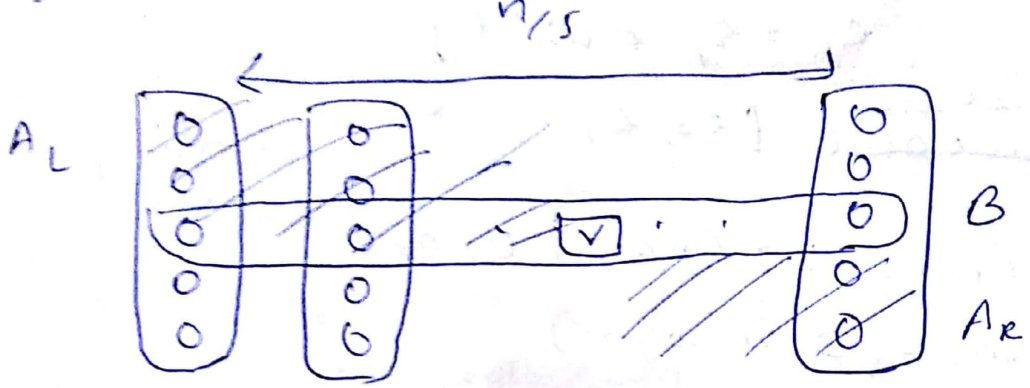
$$select(A, k) = \begin{cases} \text{(i) } select(A_L, k) & \text{if } |A_L| \geq k \\ \text{(ii) } select(A_r, k - |A_L|) & \text{if } |A_L| < k \leq |A_L| + |A_r| \\ \text{(iii) } select(A_R, k - |A_L| - |A_r|) & \text{if } k > |A_L| + \end{cases}$$

However, this gives $T(n) = T(n$

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~~ do 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:

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

$T(n) = c \cdot n$ (we claim its $O(n)$)

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

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

Therefore there exists a value ~~for~~ c.

Acc. to masters theorem

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

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]

None 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 $n \times 1$.

Do $C \cdot \vec{r} = A \cdot B \vec{r}$ 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 $\vec{r}$.

GREEDY ALGORITHM

- shortest path in a graph
- Huffman codes
- activity selection
- minimum spanning tree etc.

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 a finish time f_i. $[s_i, f_i)$

Output: max size subset of A that are mutually compatible
a_i & a_j can be scheduled together iff they do not overlap
i.e., $(f_i \leq s_j)$ OR $(f_j \leq s_i)$

Assume that the activities are in ascending order of f_i
$f_1 \leq f_2 \leq f_3 \leq \ldots \leq f_n$

Let 'i' be the no. of the last added activity.
First add a_1 to S

```
S ← {a1}
i ← 1
for j = 2 to n
{
```

```
if (s_i ≥ f_x)
{
    S ← S ∪ {a_i}
    x ← i
}
}
output S.
```

algo has greedy-choice-property. (i.e., there exists atleast one optimal solution with a_1)

Proof: suppose there's a set B without a_1

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

consider another set B'

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

[Remove a_{i_1} and add a_1]

No. of elements remain same

now $f_1 \leq f_{i_1}$ since a_1 was the first element.

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

$\Rightarrow f_1 \leq s_{i_2}$

$\therefore$ B' is valid.

since B is the max. size, B' must also be max. size subset.

Now we have to show that algorithm has optimum substructure property. (show that greedy choice property can be applied by induction)

Consider A' that is the set of all a_i such that a_i does not overlap with a_1.

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

If S' was optimum solution for A'. and should be a solution S where S is the solution = S' U {1 for A

If it were not i.e.,

$\{a_1, \underbrace{\ldots}\}$

If S' doesn't lie here, either its elements are not compatible or there are more elements (contradiction)

Huffman codes

Consider a string abbaaabbcbbbb we want to store in memory or encode through a channel in least number of bits.

We would normally encode it like this:

a: 00
b: 01
c: 10
d: 11

This would take 30 bits to st

Now I take

a : 10
b : 0
c : 110
d : 111

(make sure nothing is a prefix)
This would take 25 bits.

The frequency of occurence for the string is

S	f
a	4
b	8
c	1
d	2

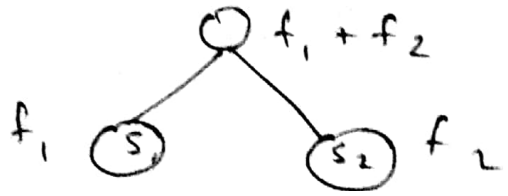
Now for any general string

S_1 - f_1
S_2 - f_2
.
.
S_n - f_n

Now store them in increasing order of frequencies $f_1 \leq f_2 \leq \ldots \leq f_n$

eg. c (1)
d (2)
a (4)
b (8)

construct a Huffman tree that is a fully binary
start with 2 lowest frequencies make them leaves and create a parent that is sum of freque

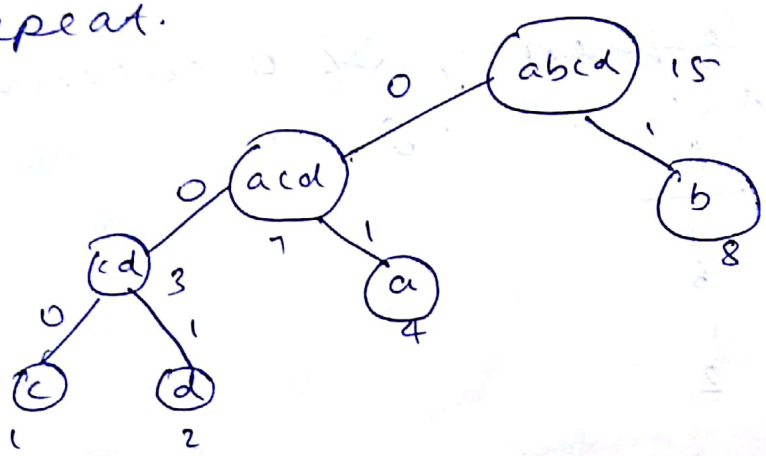


$\therefore S_{12} : f_1 + f_2$

$\vdots$

$S_n : f_n$

Again sort tree in ascending and repeat.



Now starting with leftmost branch, start labelling branches with 0, 1. Prefix property will be taken care of since it's a tree

$\therefore$ a : 01
b : 1
c : 000
d : 001

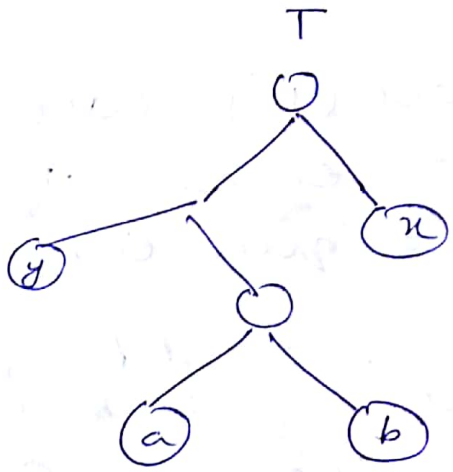
Two least frequently occuring will have maximum length

cost is the summation over all leaves of frequency multiplied by no. of bits

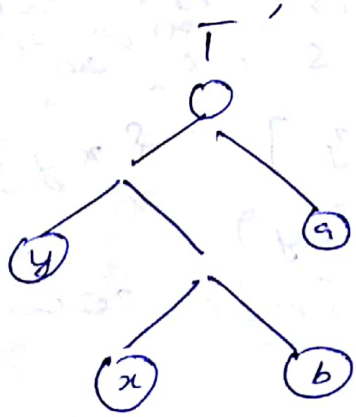
$$\text{cost} = \sum_{x \in \text{leaves}} f(x) \cdot d(x)$$

Theorem: Huffman codes have the greedy choice property i.e., if symbols x and y are the 2 least frequent ones, then $\exists$ an optimal tree with x & y as siblings at max. depth.

Consider another optimum tree.



$f(a) \leq f(b)$, $f(x) \leq f(y)$
x is least frequent



Swap x and a to get T'

$$\text{Cost}(T') - \text{Cost}(T)$$
$$= f(x) \cdot d_{T'}(x) + f(a)\, d_{T'}(a) - f(x) \cdot d_T(x) - f(a)\, d_T(a)$$
$$= f(x) \cdot d_T(a) + f(a) \cdot d_T(x) - f(x) \cdot d_T(x) - f(a) \cdot d_T(a)$$

$= (f(x) - f(a)) \; (d_T(a) - d_T(x))$

[This is non negative since T is most optimized]

since $f(x) \leq f(a)$ [since x is least frequent], $f(x) - f(a)$ is negative or zero

However, $d_T(a) - d_T(x)$ must be positive.

But since cost(T') - cost(T) is non negative, either T' is also optimized or a quantity is zero.

Construct T'' by swapping b and y. and a and x. it becomes the optimum tree we are looking for.

To show optimum substructure property. Let S be an optimum tree. T (symbols for optimum)

$S' = [S \setminus \{x, y\}] \cup \{xy\}$

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

and S' is optimal solution for T'

$$\text{cost}(T) = \sum_{x \in \text{leaf}} f(x) \cdot d(x)$$

$$\text{cost}(T') = \sum f(x) \cdot d(x)$$

$$\text{Cost}(T) - \text{Cost}(T') = d(x) \cdot f(x) + d(y) \cdot f(y) - d(xy) \cdot f(xy)$$

$d(xy) = d(x) - 1$

$\therefore$ cost (T) - cost (T')

$= d(x)\,[f(x) + f(y)] - (d(x) - 1)$

$(f(x) + f(y))$

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

Therefore, it is independent of depth or where x, y is.

If T' was not optimum, you can get an optimum T'' and make that optimum by adding x and y as leaves somewhere and cost will be less than T.

Set-cover

given an instance of S and a set family $F = \{S_1, S_2, S_3, \ldots S_n\}$

$S_i \subseteq S$

Output: Indices $i_1, i_2, \ldots i_k$ such that $\bigcup S_{i_j} = S$. We want to minimise cost k.

Take $I = \phi$ (null set)

$U = S$ (universal set)

repeat
- choose a set S_j from F that covers the max. no. of elements in U
- $I \leftarrow I \cup \{j\}$
- $U \leftarrow U \setminus S_j$

until $U = \phi$

This is the greedy approach but it will not work

$S = \{1, 2, 3, 4, 5, 6, 7\}$

$F = \{ \{1, 2, 3, 4\}, \{5, 1, 2\}, \{6, 3\}, \{7, 4\} \}$ — S_1, S_2, S_3, S_4

Acc to algo.,

we choose S_1 first

Then it goes on to S_2, S_3, S_4.

S_1 need ~~to~~ not belong to optimum solution, hence does not fulfil greedy choice property.

Output of greedy choice is ~~atlea~~ max. ($\log n$) times optimum.

$$|I| \leq O(\log n |I^*|)$$

Matroid theory

M is a matroid

$M = \langle S, F \rangle$

Matroid is an ordered pair of set and family provided that

(i) $S \neq \phi$

(ii) Hereditary property

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

$\therefore$ ϕ always belongs to F

(iii) Exchange property.

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

then $\exists x \in B \setminus A$ s.t. $A \cup \{x\} \in F$

Every element of F is called independent set.

find a max. weighted independent set

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

S_i has a weight > 0;

minimum spanning tree (MST)

Input: undirected graph $G = (V, E)$
Edges have the weight

Output: min. cost spanning tree of G.

solve using matroid

Define $M_G = \langle E, F_G \rangle$

(E: set of edges)

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

$E \neq \emptyset$

Hereditary property is true because a subset of A cannot be acyclic.

To show exchange property:

Let $A \in F_G$

$B \in F_G$

$|A| < |B|$

Theorem: Any forest on n nodes with t trees has exactly $(n - t)$ edges.

$|A| < |B|$

$|A|$: $(n - |A|)$ trees

$|B|$: $(n - |B|)$ trees

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

There must exist at least one tree T_B in $(n - |B|)$ that cuts multiple trees ie $|B|$ since there are more trees in $|A|$

Let $(u, v) \in B$, $(u, v) \notin A$

$A \cup \{(u, v)\} \in F_G$

adding (u, v) to A does not make it acyclic.

$M_a = \langle E, F_a \rangle$ is a matroid

This solves max. problem. To get solution of min; for each edge $w_i' = W - w_i$ (subtract individual weights from a large no.)

Optimum answer $A \in F$

A is maximal

$\nexists x$ s.t. $A \cup \{x\} \in F$.

all maximal independent sets are of the same size.

$M = \langle S, I \rangle$

w_i = weight of $S_i \in S$, $w_i > 0$

sort elements of S in non increasing order of weights

$A = \phi$

Loop/ Choose the next S_i (in that order)

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

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

Output A

A will be maximal independent set in matroid.

Must show that algorithm has greedy choice property.

$\exists A$ st. ~~x~~ $\in A$

Let B be optimum solution,

$x \notin B$.

$\phi \in F$

$B \in F$ since B is optimum soln.

Using exchange property.

~~$\exists x$~~ $\{x\} \in F$ ~~(due to heriditary property)~~

If $|B|$ is 1, then $\{x\}$ is also optimal.

If $|B| > 1$, by exchange property, add elements to $\{x\}$ from B.

$\{x, y_1, y_2, y_3, \ldots y_{|B|-1}\} \in F$

until they become nearly same size. We know $w(x) \geq w(y)$ since we chose max. in the beginning.

Therefore $\{x, y, y_2, \ldots\}$ must be optimal.

To show optimum substructure property:

Given $x \in A$

To find rest of optimal solns.

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

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

consider optimum soln. A.

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

set-cover (ctd.)

Let I^* be the min. sized set cover and I be the greedy output.

In iteration i, if n_i elements are not yet covered, it will choose a set S_i.

$\frac{n_i}{|I^*|}$ elements are covered by atleast some set S_j by pigeonhole principle.

S_i covers at least $\frac{n_i}{|I^*|}$

Before starting $n_0 = n$

$n_1 \leq n_0 - \frac{n_0}{|I^*|} = n\left(1 - \frac{1}{|I^*|}\right)$

$n_2 \leq n_1 - \frac{n_1}{|I^*|} \leq n\left(1 - \frac{1}{|I^*|}\right)^2$

$n_3 \leq n_2 - \frac{n_2}{|I^*|} \leq n\left(1 - \frac{1}{|I^*|}\right)^2$

.
.
.

$$n_t \leq n_t \left(1 - \frac{1}{|I^*|}\right)^t$$

If $x < 1$,

$$(1-x) \leq e^{-x}$$

$$\therefore n_t \leq n\left(1 - \frac{1}{|I^*|}\right)^t \leq n e^{-t/|I^*|}$$

Greedy will stop when

$$\frac{t}{|I^*|} \leq \log n$$

$$\frac{|I|}{|I^*|} \leq \log n \quad (\text{since } t = |I|)$$

DYNAMIC PROGRAMMING

Given a DAG with edge weights, find shortest path from u to v in the topological sorting.

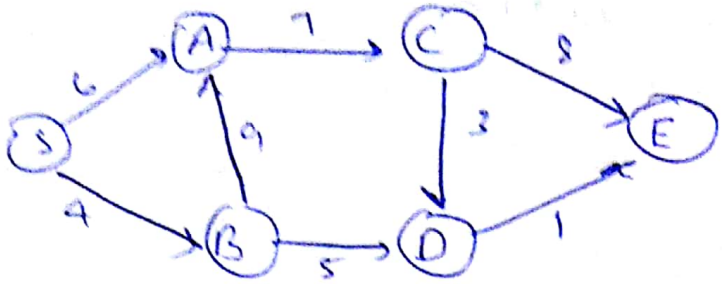
$$L(s, E) = \min_{(u,E) \in \text{Edgeset}} \{ L(s,u) + w(u,E) \}$$

i.e., for all neighbours of ending node E, the ~~for~~ shortest path till the neighbours + weight connecting to E and take minimum of them.

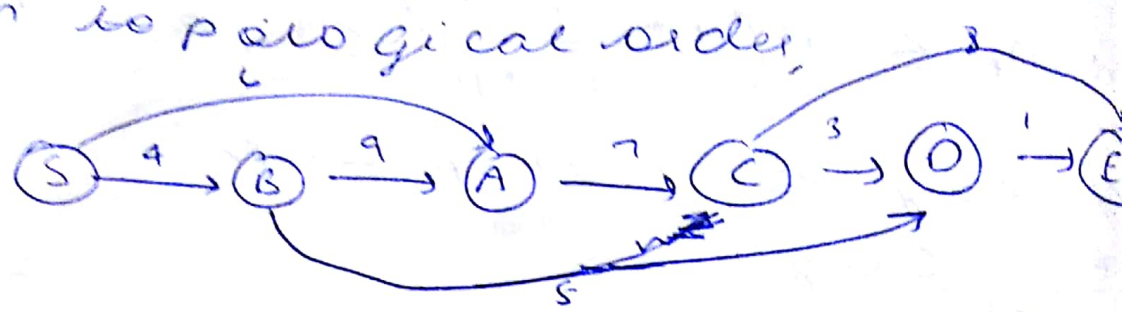
Loop
In the topological sorted order,

$$L(u) = \min_{(v,u) \in E} \{ L(v) + w(v,u) \}$$

$L(S) = 0$



In topological order,



$L(B) = 4$

$L(A) = \min \begin{cases} L(S) + 6 \\ L(B) + 9 \end{cases} = 6$

$L(C) = \min \begin{cases} L(A) + 7 \\ \cancel{L(B)} \end{cases} = 13$

$L(D) = \min \begin{cases} L(C) + 3 \\ L(B) + 5 \end{cases} = 9$

$L(E) = \min \begin{cases} L(C) + 8 \\ L(D) + 1 \end{cases} = 10$

Longest increasing subsequence

Input: sequence of nos. $a_1, a_2, a_3, \ldots, a_n$

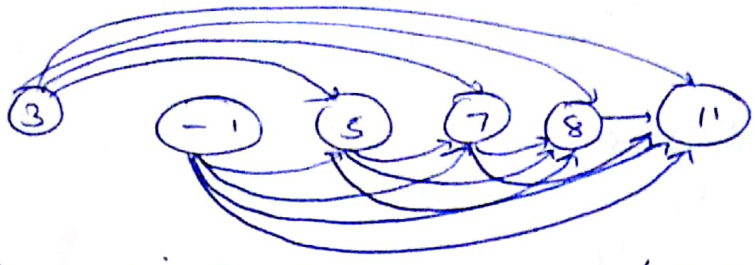
Output: $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_k}$

$i_1 < i_2 < \ldots < i_k$

$\forall\, j > t, \; a_j > a_{i_t}$

Create a DAG.

eg. 3 -1 5 7 8 11



LIS will be the longest ~~subsequence~~ path in the DAG.

If suppose we want to find LIS ending at a defined a_i

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

If we don't specify an a_i, LIS will be max. of all $L(i)$'s

Edit distance

~~Green~~

given string operation edit, delete, overload

eg. EARTH → (2) → HEART

Edit distance is min. ~~edit~~ no. of operations required to convert.

Align the 2 strings

_EARTH
HEART_

$X = x_1 x_2 x_3 \ldots x_n$

$Y = y_1 y_2 \ldots y_m$

$E(i,j)$ = Edit distance between $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$

$E(i,0) = i$

$E(0,j) = j$

alignment could end as

(i) $\overline{\quad}$ y_j OR (ii) x_i $\overline{\quad}$ OR (iii) x_i y_j

~~If (i) $E(i,j) =$~~

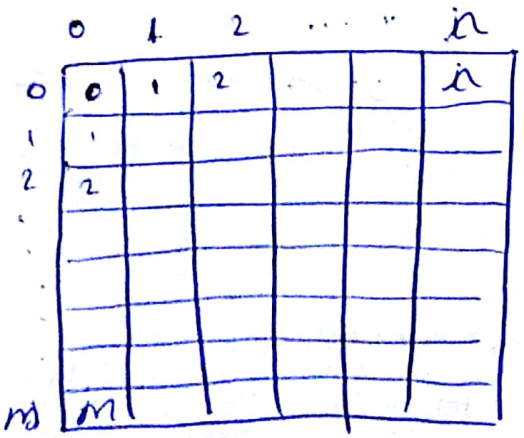
(i) ↖ (ii) ↑ (iii)

$$E(i,j) = \begin{cases} 1 + E(i, j-1) & \text{, in case (i)} \\ 1 + E(i-1, j) & \text{, in case (ii)} \\ E(i-1, j-1) + diff(x_i, y_i) & \text{in case (iii)} \end{cases}$$

↓

if x_i is same as y_j, add 0, else add 1.

$$diff(x_i, y_i) = \begin{cases} 0, & \text{if } x_i \\ 1, & \text{otherwise} \end{cases}$$



eg. X = EARTH

Y = HEART

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

MID I

9 questions

3 questions
- class (notes)
[Q & A both discussed already]

3 question
- Only the Q's are already discussed [not the answers]

1/2 question
- make your own question on divide and conquer / greedy and answer it.

1 question
- Objective question

~~8) can you use greedy to solve activity problem?~~

8) approx set cover

elements have weights

$S = \{s_1, \ldots, s_n\}$ ~~; $w_i = wt(s_i)$~~

$F = \{c_1, c_2, \ldots, c_m\}$; $w_i = wt(c_i)$

$c_i \subseteq S$

choose those c_i's that covers S but also have min. wt.

show $(\log n)$ approx is also possible using greedy.

(show the ratio is n^{th} harmonic no)

$$\sum_{i=1}^{\infty} \frac{1}{n} \leq \int^{\infty} \frac{1}{x} \cdot dx$$

$$\therefore \sum_{i=1}^{\infty} \frac{i}{n} \leq \lg n$$

9) make change for Rs N

you have denominations of Rs. 1, Rs 5, Rs 25

make change with fewest no. of notes

write a program that will give change for N with proof this is the fewest possible no. of notes (greedy)

→

show that it works if all denominations are exact powers of c

If there are arbitrary denominations, $d_1, d_2, \ldots d_k$, show that greedy does not work.

[show that greedy works for superincreasing sequence i.e., the i^{th} no. is greater than sum of $(i-1)^{th}$]

Q) Fractional knapsack

Consider a knapsack that can carry max. wt. W.
There are k objects O_i each of which has a cost c_i

O_1 ~~c_1/kg~~ weight w_1, cost c_1
O_2 ~~c_2/kg~~ " w_2, " c_2
⋮ ⋮ ⋮
O_k c_k/kg w_k, " c_k

maximize cost of load
give an algorithm using greedy
If suppose each is not fraction but objects have weight $w_1, w_2, \ldots, w_k$, greedy won't work. (0/1 knapsack)

Q) Convex hull.

You have n points in a plane

Convex hull is the polygon formed when you stretch a rubberband

give the divide & conquer algorithm for finding the convex hull.

Q) given a sorted array that is cyclically shifted by k elements. give an algorithm to find out if all elements are equal in $O(\log n)$. k is unknown.

Q) calculate $a^b \bmod p$ for some prime p.

Do it in $O(\log(p))$ time.

[hint: $\forall$ primes p, the following holds:

$\forall a$ not a multiple of p.

$a^{p-1} \bmod p = 1$ (if $\gcd(a,p)=1$)]

Fermat's little theorem

Q) You are given 2 n-sized array x and y which are already sorted. Give $O(\log n)$ algorithm to find median of $X \cup Y$ (2n elements)

Proof of Fermat's little theorem

If $gcd(a, p) = 1$, then

$$1 \equiv a^{p-1} \bmod p \iff a^p \bmod p \equiv a$$

$a \% p, 2a \% p, \ldots (p-1)a \% p$

None of these are zero (since it lies between 1 and p-1)

Let us say

$$ia \% p = ja \% p, \quad i \neq j$$

$$\Rightarrow (ia - ja) \% p = 0$$

$$\Rightarrow (ia - ja) = qp$$

$$\Rightarrow (i-j)a = qp$$

$$\Rightarrow (i-j) \bmod p = 0$$ (as a, p are co prime)

This is a contradiction

as $i - j < p \Rightarrow i - j = 0 \Rightarrow i = j$ (as $i - j$ cannot be greater than p)

Each are distinct $\Rightarrow a \% p \ldots (p-1)a \% p$ is a ~~combination~~ congruent of $1, 2, \ldots p-1$

Multiplying throughout,

$$a^{p-1}(p-1)! \equiv (p-1)!$$

$$\Rightarrow a^{p-1} = 1$$