Algorithm Design and Analysis

what is an algorithm?

- ~~Series of steps to solve a problem~~.

Date: 11/01/2023

M	T	W	T	F	S	S

- Is a well defined (unambiguous) series of steps.
- Executes in a finite amount of time.
-

Computational Problem

A computational problem defines (in general terms) a specified relationship between the input and output for problem instance.

A problem instance consits of the input needed to compute a solution to the problem.

Big-O Notation.

~~$f(n) = O(g(n))$~~

$\exists\, c, n_0 \ni f(n) \leq c\,g(n),\ \forall n \geq n_0 \iff f(n) = O(g(n))$

Big-Ω Notation.

$\exists\, c, n_0 \ni \forall n \geq n_0\ \ cg(n) \leq f(n) \iff f(n) = \Omega(g(n))$

Big-Θ Notation.

$\exists\, c_1, c_2, n_0 \ni \forall n \geq n_0,\ c_1 g(n) \leq f(n) \leq c_2 g(n)$

$f(n) = \Theta(g(n)) \iff g(n) = \Omega(f(n))$

RC

Signature _____

Initalization
Maintianance
Termination.

Date: 17/01/2023

M	T	W	T	F	S	S

Insertion Sort

Best case: $T(n) = an+b$: elements are sorted.

Worst case: $T(n) = an^2+bn+c$: elements are reverse sorted.

Prove that for $f(n) = 4n^2+10n+50$, $f(n) = O(n^2)$.

$$4n^2+10n+50 \leq 1000n^2 \quad , \forall\ n \geq 0.229$$

$$f(n) = \Theta(g(n)) \iff f(n) = \Omega(g(n)) \wedge g(n) = O(f(n))$$

small o:

$\forall c>0, \exists n_0 \ni n_0 \geq n_0, \quad 0 \leq f(n) < cg(n) \iff f(n) = o(g(n))$

small ω:

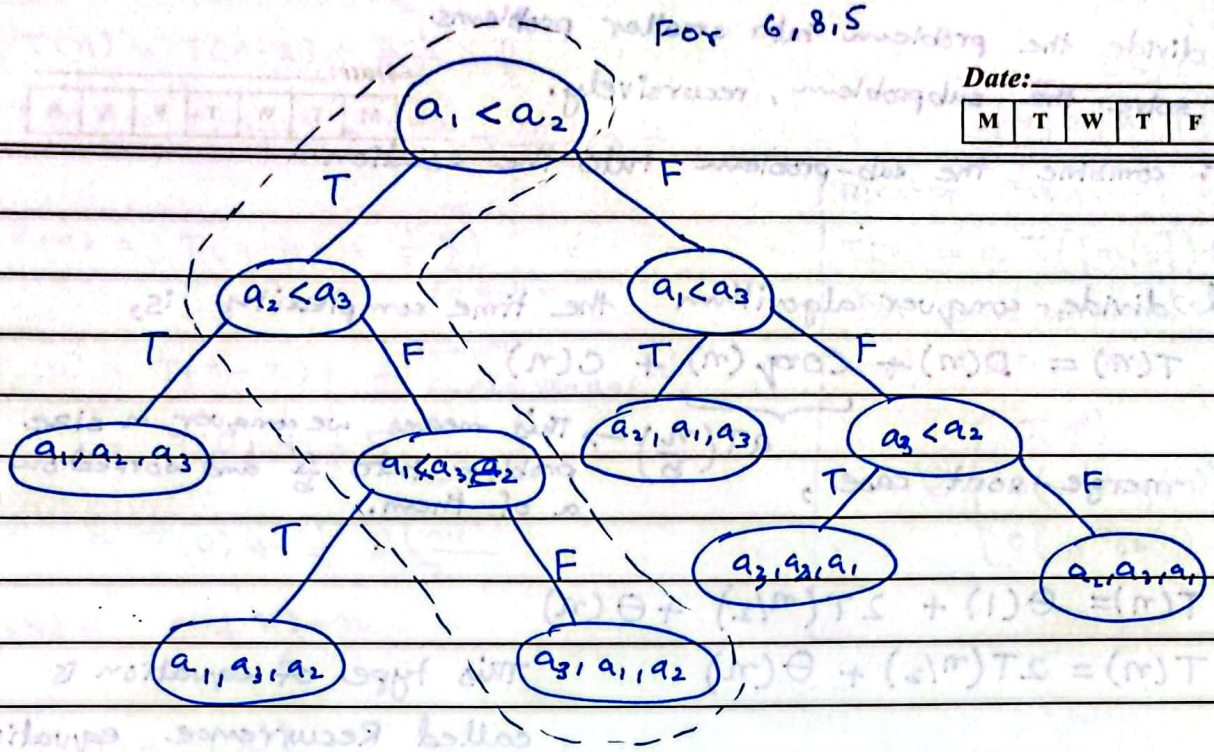
$\forall c>0, \exists n_0 \ni n \geq n_0, \quad 0 \leq cg(n) < f(n) \iff f(n) = \omega(g(n))$

If $f(n) = O(g(n)) \implies f(n) = o(g(n)) \wedge g(n) \neq \Theta(f(n))$

$$f(n) = o(g(n)) \iff \lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) = \omega(g(n)) \iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

Input; a_1, a_2, a_3



Let height of the tree be "h" 24/01/2023

$n!$ ≤ number of leaver in a tree of height $h \leq 2^h$

$\Rightarrow n! \leq 2^h$

$\Rightarrow h \geq \lg(n!)$

$\Rightarrow h \geq \lg\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)\right)$

stirling approximation

$\Rightarrow h \geq \frac{1}{2}\lg(2\pi) + \frac{1}{2}\lg(n) + n\lg(n) - n\lg(e) + \lg\left(1+\Theta\left(\frac{1}{n}\right)\right)$

$\Rightarrow h \geq \lg(n)\left(\frac{1}{2}+n\right) - n\lg(e) + \frac{1}{2}\lg(2\pi) + \lg\left(1+\Theta\left(\frac{1}{n}\right)\right)$

$\Rightarrow h \geq \Theta(n\lg(n))$

$\Rightarrow h = \Omega(n\lg(n))$

No comparission based sorting algorithms can have a worst-case senario be as best as $\Theta(n\lg(n))$

Divide and conquer Strategy.

- Divide: divide the problem into smaller problems.
- conquer: solve the subproblem, recursively.
- combine: combine the sub-problem into the solution.

Date: M T W T F S S

In usual divide-conquer algorithm the time complexity is,

$$T(n) = D(n) + \underbrace{C\,aT\left(\frac{n}{b}\right)}_{\text{conq}(n)} + C(n)$$

→ This means we conquer n size problem into $\frac{n}{b}$ and worked on a of them.

In the merge-sort case,

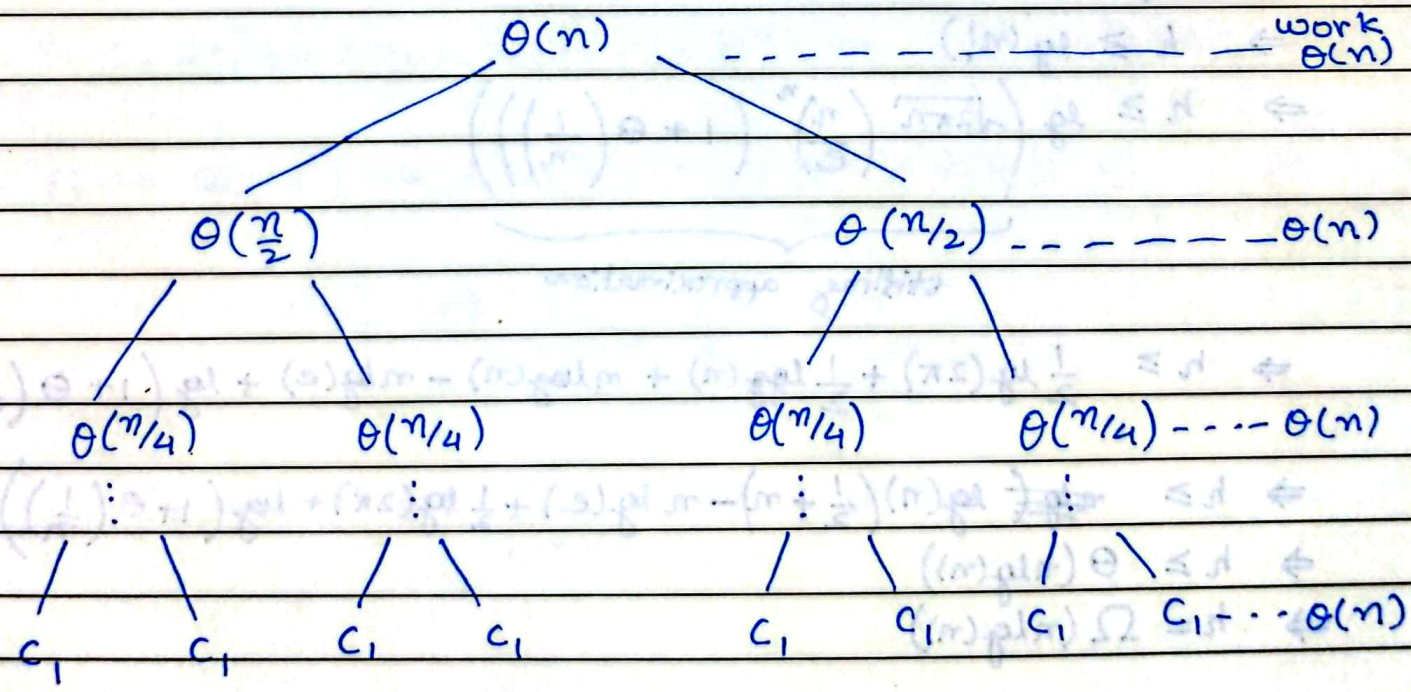
$$T(n) = \Theta(1) + 2T(n/2) + \Theta(n)$$

$$\Rightarrow T(n) = 2T(n/2) + \Theta(n)$$

This type of equation is called Recurrence equation, recurrence relation, recurrence or functional equation.

Our base case is $T(0) = T(1) = c$

or in other words, $T(n) = c$ for $n \le n_0$.



If n is some power of 2, then there would be n leaves.

In the end the total time would be

$$T(n) = c_2 n \lg(n) + c_1 n$$

$T(n) = T(n-1) + \frac{n}{2}$

$T(n) = T(n-2) + \frac{n-1}{2} + \frac{n}{2}$

$T(n) = T(n-3) + \frac{n-2}{2} + \frac{n-1}{2} + \frac{n}{2}$

Date: 26/01/2023

M	T	W	T	F	S	S

$$T(n) = T(n-k) + \frac{1}{2}\sum_{i=n-k+1}^{n} i$$

$$T(n) = T(n-n) + \frac{1}{2}\sum_{i=n-n+1}^{n} i$$

$$= T(0) + \frac{1}{2}\sum_{i=1}^{n} i$$

$$= T(0) + \frac{1}{2}\cdot\frac{n(n+1)}{2}$$

$$T(n) = c + \frac{n^2+n}{4}$$

$\Rightarrow T(n) = \Theta(n^2)$

Master Theorem: If $T(n) = aT(\lceil n/b \rceil) + O(n^d)$ for some constants $a > 0$, $b > 1$ and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log(n)) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

31/01/2023

For two matrices of $n \times n$ dimension, let say A and B we can show their product as,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$\forall$ i, j, $A(A_{ij}) = \frac{n}{2} \times \frac{n}{2}$ $\wedge$ $(B_{ij}) = \frac{n}{2} \times \frac{n}{2}$

$$C = AB = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & AB_{11\,12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

The recursive equation for this is,

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(1) \quad ; \quad T(1) = c_0$$

Solving the recurrence relation with unrolling.

$$T(n) = 8T\left(\frac{n}{2}\right) + c_1$$

Date: M T W T F S S

$$T(n) = 8\left(8T\left(\frac{n}{4}\right) + c_1\right) + c_1$$

$$= 8^2 T\left(\frac{n}{2^2}\right) + 8c_1 + c_1$$

$$= 8^2\left(8T\left(\frac{n}{2^3}\right) + c_1\right) + 8c_1 + c_1$$

$$= 8^3 T\left(\frac{n}{2^3}\right) + 8^2 c_1 + 8c_1 + c_1$$

$$\vdots$$

$$= 8^k T\left(\frac{n}{2^k}\right) + \left(8^{k-1} + 8^{k-2} + \cdots + 8 + 1\right) c_1$$

$$T(n) = 8^k T\left(\frac{n}{2^k}\right) + \frac{8^k - 1}{7} c_1$$

For $n = 2^k \Rightarrow k = \log_2(n)$

$$T(n) = (2^k)^3 T\left(\frac{n}{2^k}\right) + \frac{(2^k)^3 - 1}{7} c_1$$

$$T(n) = n^3 T(1) + \frac{n^3 - 1}{7} c_1$$

$$T(n) = n^3 c_0 + \frac{n^3 - 1}{7} c_1$$

$$T(n) = \left(c_0 + \frac{c_1}{7}\right) n^3 - \frac{c_1}{7}$$

$$T(n) = c_1' n^3 + c_0'$$

$$\Rightarrow T(n) = \Theta(n^3)$$

Prove of Master Theorem:-

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$

Date: 02/02/2023

Total Work	MN with of W	T	Size	of	WI. numb
cn^d	nodes $n^0 = 1$		n		0
$c\left(\frac{n}{b}\right)^d$ $c\left(\frac{n}{b}\right)^d$ ... $c\left(\frac{n}{b}\right)^d$	$ac\left(\frac{n}{b}\right)^d$	$\frac{n}{b}a$	$\frac{n}{b}$		1
$c\left(\frac{n}{b^2}\right)^d$ .... $c\left(\frac{n}{b^2}\right)^d$	$a^2c\left(\frac{n}{b^2}\right)^d$	a^2	$\frac{n}{b^2}$		2
⋮	⋮	⋮	⋮		⋮
	$a^k c\left(\frac{n}{b^k}\right)^d$	a^k	$\frac{n}{b^k}$		k

we assume two thing, $T(1) = c_0$ is the base-case, and n is a perfect power of b. with this assumption.

$$\frac{n}{b^k} = 1 \Rightarrow k = \log_b(n)$$

The total work done is,

$$T(n) = \sum_{k=0}^{\log_b(n)} a^k c\left(\frac{n}{b^k}\right)^d$$

$$= cn^d \sum_{k=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^k$$

$$= cn^d \frac{\left(a/b^d\right)^{\log_b(n)+1} - 1}{a/b^d - 1}$$

$$= cn^d \frac{n^{\log_b(a)-d}\left(a/b^d\right) - 1}{a/b^d - 1}$$

$$T(n) = c\frac{n^{\log_b(a)}a - b^d n^d}{a - b^d}$$

Page No. _____ Signature _____

For a ~~perf~~ n-bit integer, where n is a power of 2. we can write it as,

Date: 07/02/2023

M	T	W	T	F	S	S

$$x = 2^{n/2} x_L + x_R$$

$$y = 2^{n/2} y_L + y_R$$

$$x \cdot y = (2^{n/2} x_L + x_R)(2^{n/2} y_L + y_R)$$

$$= 2^n x_L y_L + 2^{n/2} x_L y_R + 2^{n/2} y_L x_R + x_R y_R$$

$$T(n) = 4T(\tfrac{n}{2}) + \Theta(n)$$

$$\Rightarrow T(n) = \Theta(n^2)$$

$$x \cdot y = 2^n x_L y_L + (x_L y_R + x_R y_L) 2^{n/2} + x_R y_R$$

Gauss noticed that,

$$(a+bi)(c+di) = (ac - bd) + (ad+bc)i$$

$$ad+bc = (a+b)(c+d) - ac - bd$$

With this the expression becomes,

$$(a+bi)(c+di) = (ac - bd) + \big((a+b)(c+d) - ac - bd\big)i$$

Here we have 3 unique products.

with this the ~~fast~~ time complexity becomes,

$$T(n) = 3T(\tfrac{n}{2}) + \Theta(n) \Rightarrow T(n) = \Theta(n^{\lg(3)})$$

The n-bit multiplication will be,

$$x \cdot y = 2^n x_L y_L + 2^{\frac{n}{2}} \Big((x_L + x_R)(y_L + y_R) - x_L x_R - y_L y_R\Big) + x_R y_R$$

This way of multiplication is called Karatsuba multiplication Algorithm.

strassen came up with similar approach for matrix multiplication.

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

Date: M T W T F S S

For this approach we got,

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) \Rightarrow T(n) = \Theta(n^3)$$

But strassen was clever enough to find out,

$$P_1 = A(F-H) \qquad P_5 = (A+D)(E+H)$$
$$P_2 = (A+B)H \qquad P_6 = (B-D)(G+H)$$
$$P_3 = (C+D)E \qquad P_7 = (A-C)(E+F)$$
$$P_4 = D(G-E)$$

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

This will take,

$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2) \Rightarrow T(n) = \Theta(n^{\lg(7)})$$

Linear Homogenous Recurrence Relation 9/02/2023

The general form of Linear homogenous recurrence relation of degree k.

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

Assume $a_n = r^n$

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$$
$$\Leftrightarrow r^n - c_1 r^{n-1} - c_2 r^{n-2} - \cdots - c_k r^{n-k} = 0$$
$$\Leftrightarrow r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

Page No. _____ Signature _____

By fundamental theorem of algebra there are k ~~dt~~ roots of it, let say, $r_1, r_2, \ldots, r_k$. Assuming that the characteristic roots are distinct,

$$a_n = \sum_{i=1}^{k} \alpha_i r_i^n$$

Date: M T W T F S S

Example,

$$a_n = a_{n-1} + 2a_{n-2}, \quad a_0 = 2, a_1 = 7$$

For characteristic equation,

$$r^n = r^{n-1} + 2r^{n-2}$$

$$\Rightarrow r^2 - r - 2 = 0$$

$$\Rightarrow r = 2 \vee r = -1$$

$$\Rightarrow a_n = \alpha_1 2^n + \alpha_2 (-1)^n$$

$$a_0 = 2 \Rightarrow 2 = \alpha_1 + \alpha_2$$

$$a_1 = 7 \Rightarrow 7 = 2\alpha_1 - \alpha_2$$

$$\Rightarrow \alpha_1 = 3 \wedge \alpha_2 = -1$$

~~To prove the~~ $a_n = 3 \cdot 2^n + (-1)^{n+1}$

To prove the result,

$$a_n = a_{n-1} + 2a_{n-2}$$

$$\Rightarrow a_n = 3 \cdot 2^{n-1} + (-1)^n + 2\left(3 \cdot 2^{n-2} + (-1)^{n+1}\right)$$

$$a_n = 3 \cdot 2^{n-1} - (-1)^{n+1} + 3 \cdot 2^{n-1} + 2(-1)^{n+1}$$

$$a_n = 6 \cdot 2^{n-1} + (-1)^{n+1}$$

$$a_n = 3 \cdot 2^n + (-1)^{n+1}$$

$a_n = 6a_{n-1} - 9a_{n-2}, a_0 = 1, a_1 = 6$

The charatristic equation is,

$r^n = 6r^{n-1} - 9r^{n-2}$

Date: _____

M	T	W	T	F	S	S

$\Rightarrow r^2 - 6r + 9 = 0$

$\Rightarrow r = \frac{6 \pm \sqrt{6^2 - 4(1)(+9)}}{2(1)}$

$\Rightarrow r = \frac{6 \pm 6}{2}$

$\Rightarrow r = 3, 3$

In this case, $a_n = \alpha_1 3^n + \alpha_2 n 3^n$

$a_0 = 1 \Rightarrow 1 = \alpha_1$

$a_1 = 6 \Rightarrow 6 = 3 + \alpha_2 3 \Rightarrow \alpha_2 = 1$

$\Rightarrow a_n = (n+1)3^n$

No algorithm can have a runtime expressed as a linear combination of homogeneous recurrence, there would be some constant involve.

That means,

$T(n) = T(n-1), T(0) = c$

can not be a runtime of any algorithm. The least we would have is,

$T(n) = T(n-1) + c_1, T(0) = c$

The algorithms in real life are of form,

$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n)$

This is called linear non-homogenous recurrence relation of degree k.

Page No. _____ RC Signature _____

Graph is a pair of set of vertices and edges. 14/02/2023

$G = (V, E)$

where, $V := \{v_1, v_2, \cdots, v_n\}$

$E := \{(v_i, v_j) \mid 1 \le i, j \le n\}$

Date: M T W T F S S

DFS : Depth first search

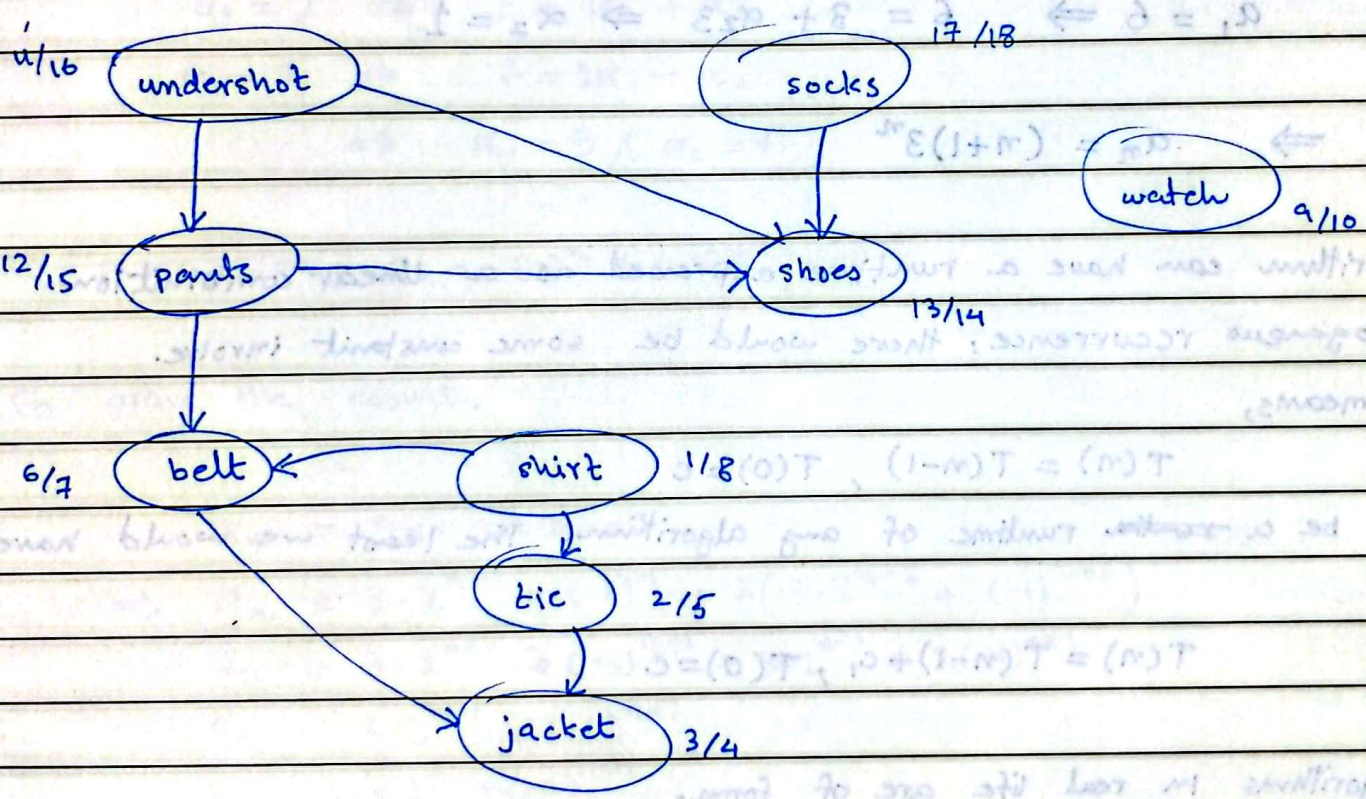
DFT : Depth first traversal

BFS : Breath first search

BFT : Breath first traversal.

The graph obtained from DFT is called Forest.

DAGs : Directed Acyclic Graphs 21/02/2023



undershot, socks, watch, shirt, pants, tie, belt, shoes, jacket

Topological sort of a DAG: If there is an edge (u, v) in DAG G, then u appears before v in the ordering.

Date: _____

M	T	W	T	F	S	S

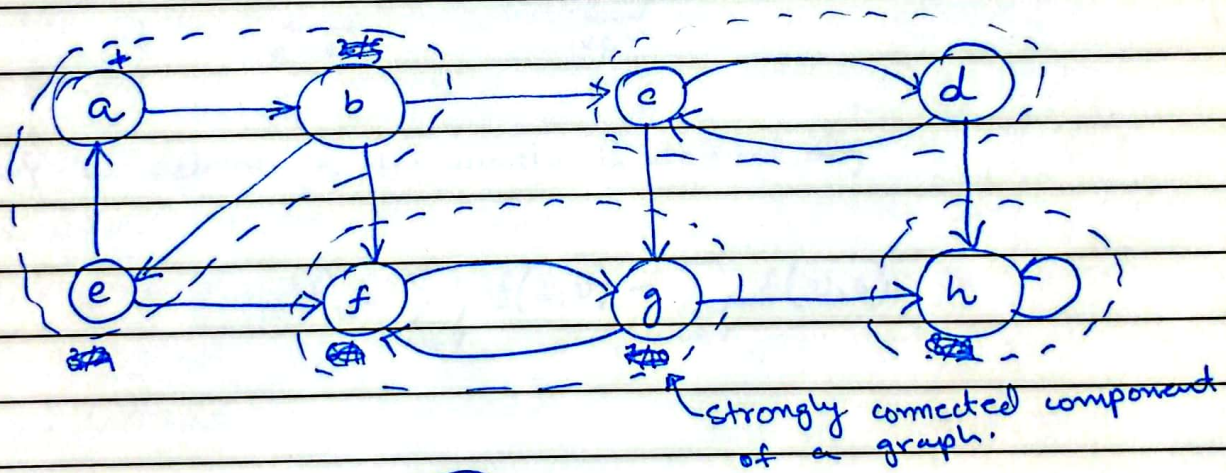
we can do this by writing all the nodes with no inward edge, then remove them from the graph and apply the same process untill the graph is empty.

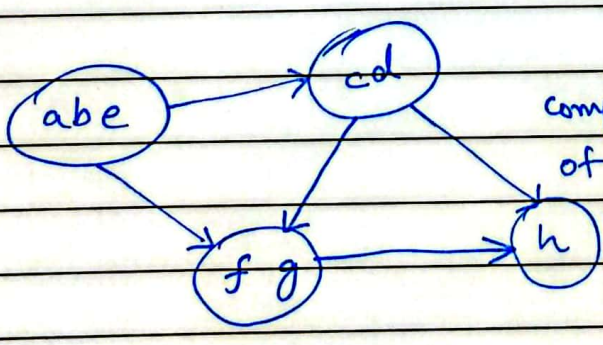
Books algo:

Apply DFS on G to compute finish time of the vertices. As soon as you finish and find the finish time of the vertex add it to the linked list and return it in the end.

23/02/2023

Given a directed graph $G = (V, E)$, a strongly connected component (SCC), $C \subseteq V$, is a maximal set such that any two vertices in C are mutually reachable.





component graph of the upper graph.

Signature _____

Given, $U \subseteq V$, let ~~$d(U) = \min_{u \in U} \{u.d\}$~~

$$d(U) = \min_{u \in U} (u.d) \quad \text{(discovery time)}$$

Date:

M	T	W	T	F	S	S

$$f(U) = \max_{u \in U} (u.f) \quad \text{(finish time)}$$

Lemma F20.14.

Let C and C' be distinct strongly connected component in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C'$ and $v \in C$. Then

RC

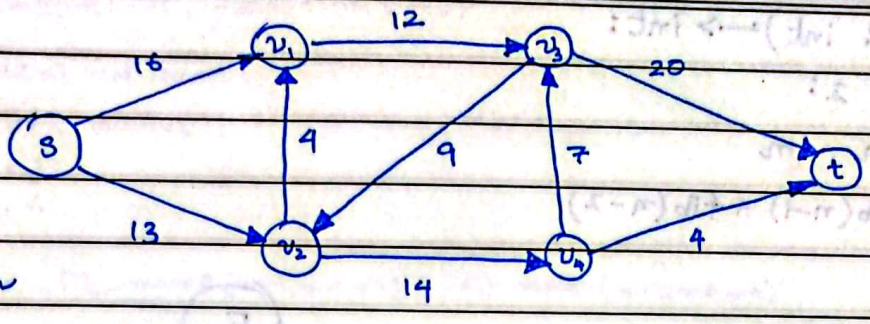
Signature

Flow network is a directed graph $G = (V, E)$ with a capacity $c: V \times V \to \mathbb{R}^+ \cup \{0\}$.

There are two special vertices that are called sourse s and sink t.

Date: 28/02/2022

M	T	W	T	F	S	S



Any other vertex beside s and t is on a ~~path~~ any path from s to t.

It can have multiple sources and sinks.

It also has no-anti paralled edge.

$(u,v) \in E \Rightarrow (v,u) \notin E$

flow $f: V \times V \to \mathbb{R}$ in a flow network satisfies.

Capacity constraint: $0 \leq f(u,v) \leq c(u,v)$ Flow is always in bounds of capacity.

Flow conservation: Incoming flow is equals to outgoing flow.

For $v \in V - \{s,t\}$

$$\sum_{u \in V} f(u,v) = \sum_{w \in V} f(v,w)$$

Flow of a network is $|f|$ which is defined as,

$$|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$$

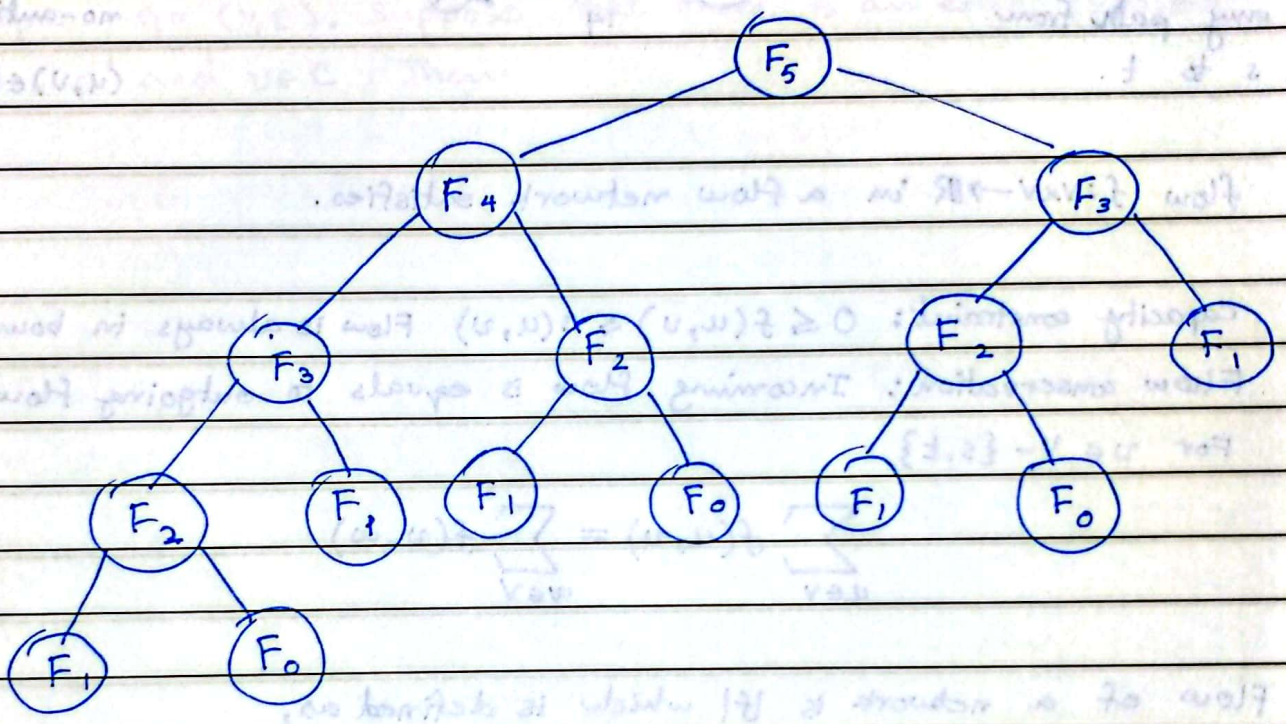
Dynammic Programming.

calculate the fibonacci series.

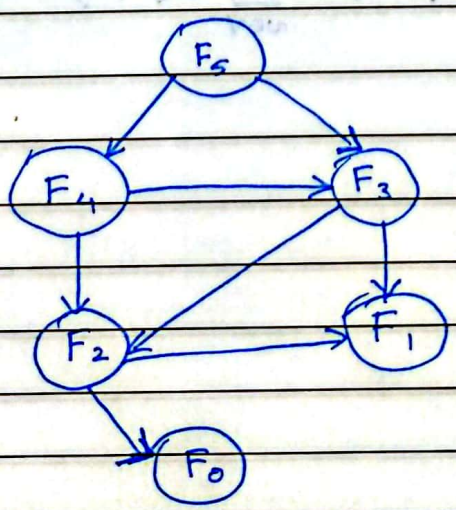
Date: 07/03/2023

M	T	W	T	F	S	S

```
def fib(n: int) -> int:
    if n < 2:
        return n
    return fib(n-1) + fib(n-2)
```



There are many redundencies in this approach, because many values would have ~~been~~ ~~cost~~ to recalculate. A simple approach could be,



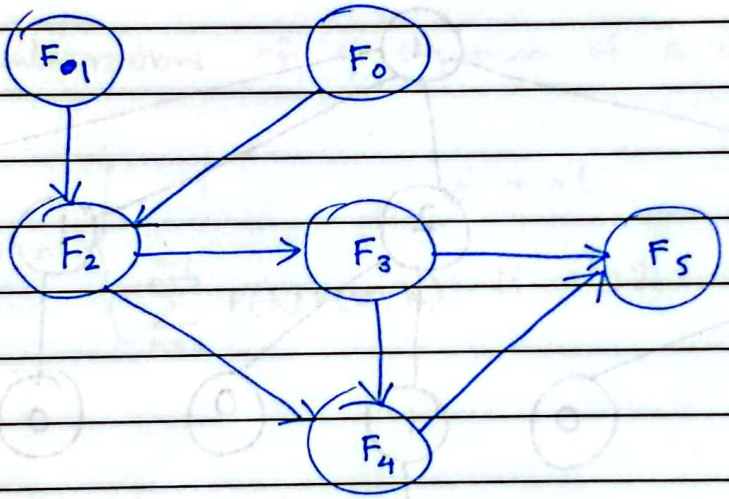
This is an interesting graph because it is a DAG. It provides a Top-Down approach. The relevant code for it is.

```
def fib_top_down(n: int) -> int:
    memo = [None for _ in range(n)]
    return _fib_memo(n, memo)

def _fib_memo(n: int, memo: List[int]) -> int:
    if memo[n] not None:
        return memo[n]
    elif n < 2:
        fib = n
    else:
        fib = _fib_memo(n-1, memo) + _fib_memo(n-2, memo)
    memo[n] = fib
    return fib
```

Date: M T W T F S S

Bottom-Up Approach for similar graph would be,



09/03/2023

Rod-cutting problem.

length i	1	2	3	...	n
price p_i	1	5	8	...	p_n

r_i : optimal price with or without cuts for a rod of length, i.

$$r_n = \max\{p_n, \max_{1 \le j \le n-1}(r_j + r_{n-j})\}$$ Optimal substructure.

$r_n = \max\{p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_1 + r_{n-1}\}$

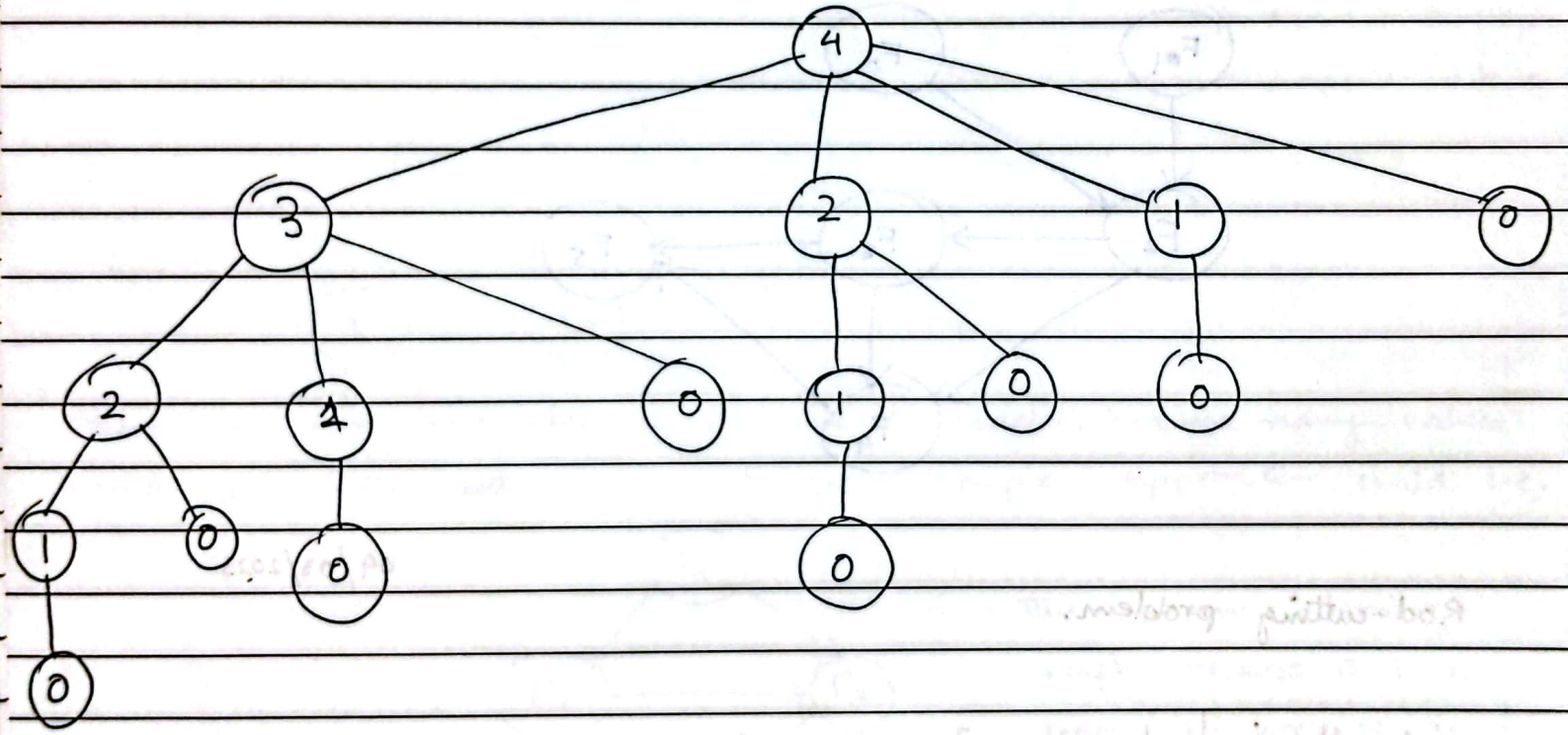
This can also be optimized as,

$$r_n = \max_{0<j\leq n} (p_j + r_{n-j})$$

Date: _____

M	T	W	T	F	S	S

```
Cut-Rod ( p , n)
    if n==0:
        return 0
    q = - ∞
    for i = 1 to n
        q = max { q, p[i] + Cut-Rod(p, n-i)}
    return q
```



$$T(n) = 1 + \sum_{i=0}^{n-1} T(i) = \Theta(2^n)$$

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Date: M T W T F S S

Matrix-Chain Multiplication

14/03/2023

Input: A sequence of matrices $A: \langle A_1, A_2, \cdots, A_n \rangle$.

Output: A parenthesization of the product of the matrices in A that result in the least number of scalar multiplication

Number of parenthesizations of a sequence of n matrices;

$$P(n) := \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)\, P(n-k) & \text{if } n \geq 2 \end{cases}$$

Let $A_{i:j} = A_i A_{i+1} A_{i+2} \cdots A_j$

Let $\dim(A_i) = p_{i-1} \times p_i$

Let $p := \langle p_{i-1}, p_{i+1}, \cdots, p_j \rangle$

$m[i,j]$: lowest cost way to compute $A_{i:j}$

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}\, p_k\, p_j, \quad m[i,i] = 0.$$

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \} \end{cases}$$

Date: Mon Tue Wed Thu Fri Sat Sun

16/03/2023

Longest Increasing / common Subsequence.

Sequence : $\langle a_1, a_2, a_3, \ldots, a_n \rangle$

Sub-sequence : $\langle a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_k} \rangle$

$0 \le k \le n$, $1 \le i_1 < i_2 < i_3 < \cdots < i_k \le n$

Let $X = \langle x_1, x_2, \cdots, x_m \rangle$ and $Y = \langle y_1, y_2, \cdots, y_n \rangle$ be a sequence and Let $Z = \langle z_1, z_2, \cdots, z_k \rangle$ be any LCS of X and Y.

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and z_{k-1} is an LCS of x_{m-1} and y_{n-1}.
2. If $x_m \neq y_n$ and $z_k \neq x_m$, then Z is an LCS of x_{m-1} and y.
3. If $x_m \neq y_n$ and $z_k \neq y_n$, then Z is an LCS of x and y_{n-1}.

21/03/2023

$c[m,n]$: Length of the LCS.

$$c[m,n] = \begin{cases} 0 & m = 0 \ \forall n = 0 \\ c[m-1, n-1] + 1 & x_m = y_n \\ \max \{ c[m,n-1], c[m-1,n] \} & \text{otherwise.} \end{cases}$$

Knap sack.

$$K(w) = \max_{i:\, w_i \le w} \{ K(w - w_i) + v_i \} \quad \text{(with repetition)}$$

Date: M T W T F S S

$$K(w, j) = \max \{ K(w - w_j, j-1) + v_j, K(w, j-1) \} \quad \text{(without repetition)}.$$

Greedy Algorithm.

04/04/2023

Activity scheduling Algorithm.

Activity, a_i has start time s_i, and finish time f_i such that the activity runs in $[s_i, f_i)$

S_{ij}: set of activities that start after f_i and finish before s_j

A_{ij}: largest compatible set of activities the begin after f_i and finishes before s_j.

$c[i,j]$: cardanality of $A_{i,j}$.

$$c[i,j] = \begin{cases} 0 & S_{ij} = \emptyset \\ \max_k \{ c[i,k] + 1 + c[k,j], \; a_k \in S_{ij} \} & S_{ij} \neq \emptyset \end{cases}$$

```
Recursive-Activity-Selector(s, f, n, k):
    m = k+1
    while m ≤ n and s[m] < f[k]
        m = m+1
    if m ≤ n
        return {a_m} ∪ Recursive-Activity-selector(s, f, m, n)
    return ∅
```

1. Charecterize the optimal solution.
2. Recursively define the value of the optimal solution.
3. Introduce a greedy choice

Date: 11/04/2023.

M	T	W	T	F	S	S

4. Prove that greedy choice yields a globally optimal solution.
5. Recursively compute the optimal solution.
6. Iteratively compute the optimal solution.

```
Greedy - Activity - Selector (s, f, n)
    A = {a_1}
    k = 1
    for m = 2 to n
        if s[m] ≥ f[k]
            A = A ∪ {a_m}
            k = m
    return A
```

String Algorithms. 13/04/2023

For a text T[1:n] and pattern P[1:m] ∋ $m \leq n$.

text T

a	b	c	a	b	a	a	b	c	a	b	a	c

Pattern P (s = 3)

a	b	a	a

s = 3 is a valid shift.

A valid shift is defined as,

$T[s+1 : s+m] = P$

or $T[s+j] = P[j], \forall\ 1 \leq j \leq m$

string matching algorithm.

Input: Text $T[1:n]$ and Pattern $P[1:m]$.

output: all valid shifts.

Date: M T W T F S S

A string x is a prefix of w, for x, w over Σ, iff $w = xy$ for some $y \in \Sigma^*$. It is denoted as $x \sqsubset w$.

For suffix, $w = yx$ for some $y \in \Sigma^*$. It is denoted as $x \sqsupset w$.

For a valid shift s,

$(T[s+1:s+m] \sqsubset P) \wedge (T[s+1:s+m] \sqsupset P)$ (My definition)

$P \sqsupset T[:s+m]$ (what book says)

```
Naive-String-Matcher(T, P, n, m)
    for s=0 to n-m
        if T[s+1:s+m] == P[1:m]
            print ("the string match!")
```

String Matching: Rabin-Karp Algorithm. 18/04/2023

p: Numeric value of $P[1:m]$

t_s: numeric value of $T[s+1:s+m]$

$$t_{s+1} = (t_s - T[s+1]10^{m-1})10 + T[s+m]$$

~~This is 2for ~~

Page No. RC Signature

The algorithm,

- Initalize (p, t_0) - $\Theta(m)$
- for all valide shifts, s: $n-m+1$ iterations.
  - compare p with t_s
  - update t_s for the nex iteration.

Date: M T W T F S S

for alphabets of size d. i.e. $|\Sigma| = d$.

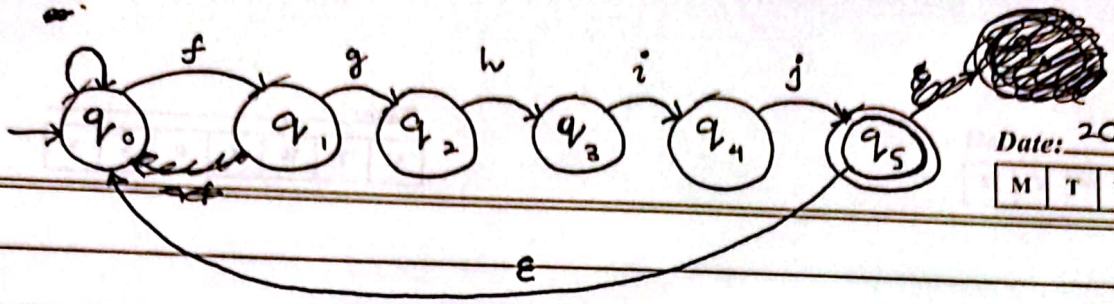
$$h = d^{m-1} \bmod q$$

$$t_{s+1} = \left((t_s - T[s+1] \cdot h)d + T[s+m]\right) \bmod q$$

generally q is a prime number. But differeut numbers could has same mods therefore if there is a hit do the naive string compresion.

```
RABIN-KARP-MATCHER(T, P, n, m, d, q)
    h = d^(m-1) mod q
    p = 0
    t_0 = 0
    for i = 1 to m
        p = (dp + P[i]) mod q
        t_0 = (dt_0 + T[i]) mod q
    for s=0 to n-m
      if p==t_s
        if P[1:m] == T[s+1:s+m]
            print("A valid shift")
      if s<n-m
        t_{s+1} = (d(t_s - T[s+1]h) + T[s+m+1]) mod q
```

The runtime complexity is $O(nm - m^2 + m)$



T: abcdefghijklmnopqr
rstuvwxyz
P: fghijk.

Date: 20/04/2023

M	T	W	T	F	S	S

If T[i] causes M to be in $q \in A$, then $i-m$ is a valid shift.

Text — Finite Automaton — state — Accepting state.

T: ababababacaba

P: abababaca

Σ: {a, b, c}