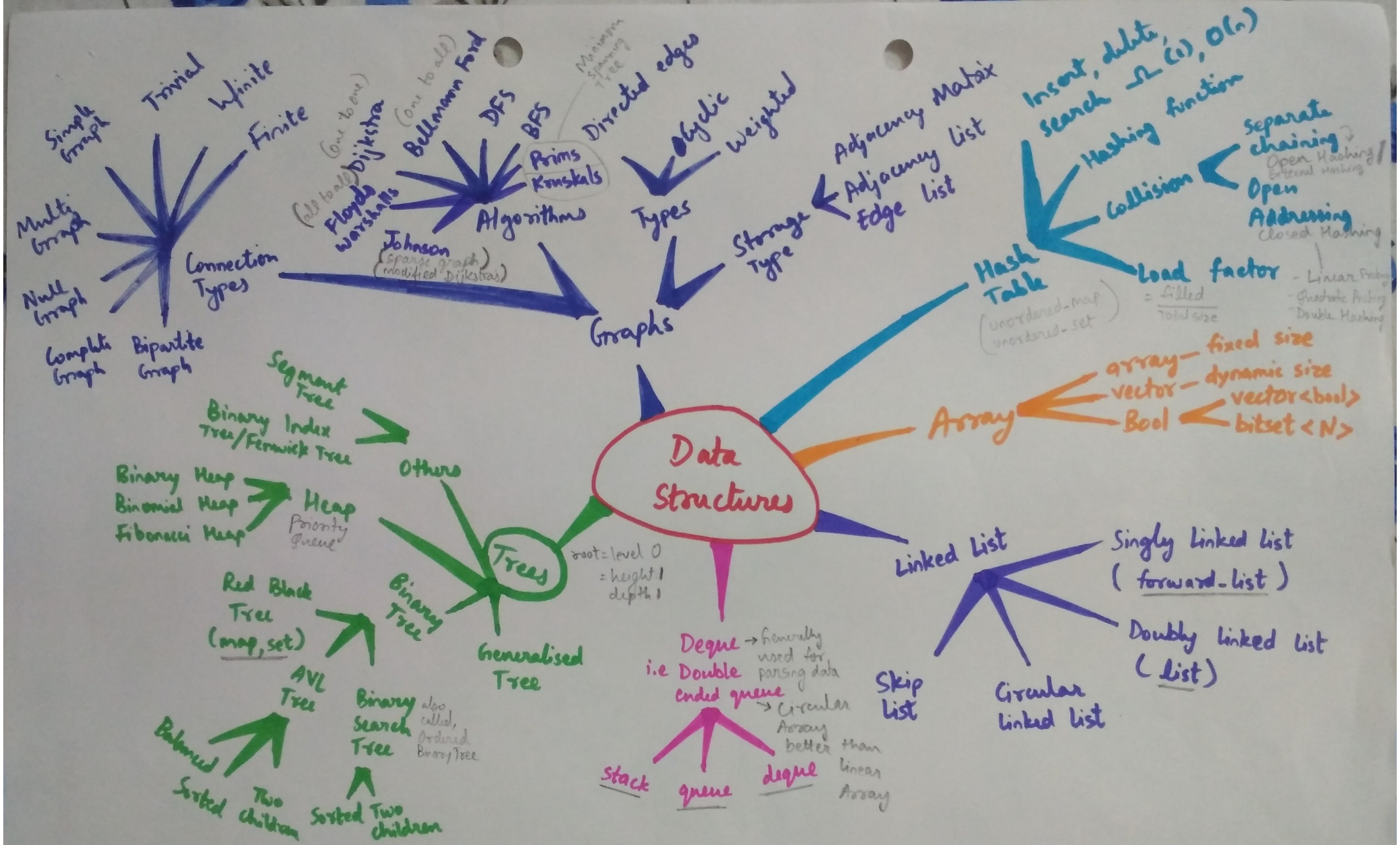
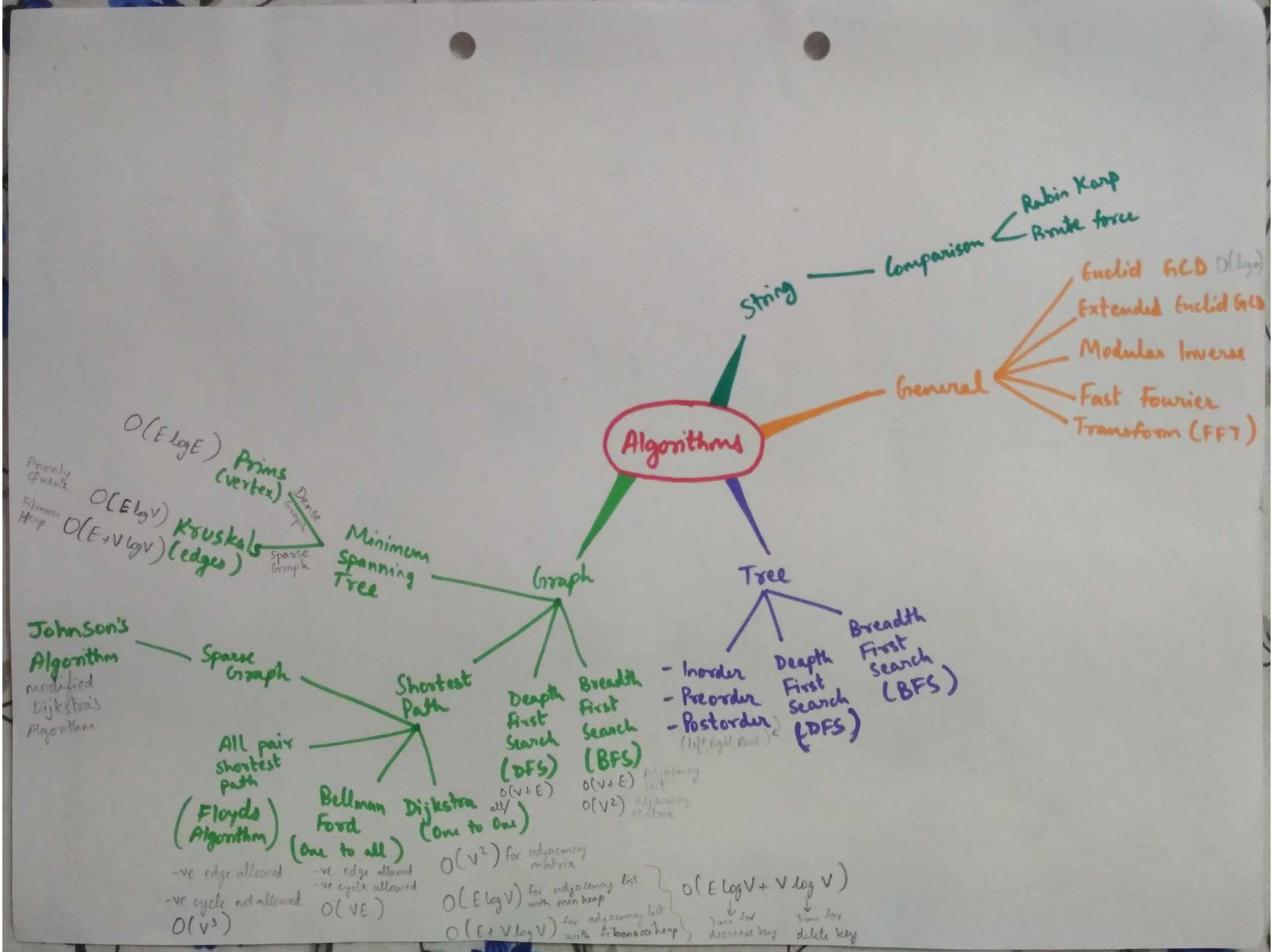

Data Structures
Graphs
Simple Graph
Trivial
Infinite
Finite
Multi Graph
Null Graph
Complete Graph
Bipartite Graph
Connection Types
(one to one)
Dijkstra
(all to all)
Floyds warshalls
(one to all)
Bellmann Ford
DFS
BFS
Algorithms
Johnson
(sparse graph)
(modified Dijkstras)
Minimum spanning Tree
Prims
Kruskals
Directed edges
Acyclic
Weighted
Types
Storage Type
Adjacency Matrix
Adjacency List
Edge list
Hash Table
(unordered_map)
(unordered_set)
Insert, delete, search Ω(1), O(n)
Hashing function
Collision
separate chaining
Open Hashing / External Hashing
Open Addressing
Closed Hashing
Load factor
= filled / Total size
- Linear Probing
- Quadratic Probing
- Double Hashing
Array
array – fixed size
vector – dynamic size
Bool
vector<bool>
bitset<N>
Linked List
Singly linked list
(forward_list)
Doubly linked list
(list)
Skip list
Circular linked list
Trees
root = level 0
= height 1
depth 1
Others
Segment Tree
Binary Index Tree / Fenwick Tree
Heap
Priority queue
Binary Heap
Binomial Heap
Fibonacci Heap
Binary Tree
Red Black Tree
(map, set)
AVL Tree
Balanced
Sorted
Two children
Binary Search Tree
also called, Ordered Binary Tree
Sorted
Two children
Generalised Tree
Deque → Generally used for parsing data
i.e Double ended queue
→ Circular Array better than linear Array
stack
queue
deque


Algorithms
String
Comparison
Rabin Karp
Brute force
General
Euclid GCD O(log n)
Extended Euclid GCD
Modular Inverse
Fast Fourier Transform (FFT)
Graph
Minimum spanning Tree
O(E log E) Prims (vertex)
Priority Queue
Dense Graph
O(E log V) Kruskals (edges)
Fibonacci Heap
O(E + V log V)
Sparse Graph
Shortest Path
Johnson's Algorithm
modified Dijkstra's Algorithm
Sparse Graph
All pair shortest path
(Floyds Algorithm)
-ve edge allowed
-ve cycle not allowed
O(V³)
Bellman Ford (one to all)
-ve edge allowed
-ve cycle allowed
O(VE)
Dijkstra (One to One / all)
O(V²) for adjacency matrix
O(E log V) for adjacency list with min heap
O(E + V log V) for adjacency list with fibonacci heap
Depth First Search (DFS)
O(V + E)
Breadth First Search (BFS)
O(V + E) Adjacency list
O(V²) Adjacency matrix
Tree
- Inorder
- Preorder
- Postorder
(Left, Right, Root)
Depth First Search (DFS)
Breadth First Search (BFS)
O(E log V + V log V)
Time for decrease key
Time for delete key

Programming Languages

* Lexical scoping = Static scoping (eg- C, C++)
Dynamic data structures require heap.
* Recursive programs require dynamic memory management. (not heap)
Recursive programs and iterative programs are equally powerful.
Both recursive and iterative programs are equally expressive.

* For every iterative program there is an equivalent recursive program.
& For every recursive program there is an iterative program.

For C & C++
* NEVER compare floating point numbers like Integers, because they are store in mantissa, exponent form in binary where all numbers cannot be stored accurately (eg- 1.1)
ALWAY use difference for comparison.

```
if (abs(f - 1.1) <= 0.000001)   // Works
    cout << " f is equal to 1.1 " << endl;
if ( f == 1.1)   // NOT working because computer may store f as 1.000001 or 1.099999
    cout << " will not work " << "no output by this line as the condition will return false " << endl;
```

* Global and static variables are always initialized to 0, if they are not initialized by the programmer.

* Compiler always associates the else part with closest unmatched if part.

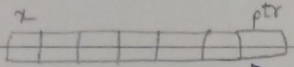
* In while & do...while loop, continue transfers the control to the condition. HOWEVER, in for loop, the control is transfered to the update expression.
Compiler gives error if continue is used in switch case.

* Using goto statement, control can only be transferred within a function.

* Which language necessarily need heap allocation in the runtime environment?
Ans. Those that allow dynamic data structures.
NOT FOR recursion, dynamic scoping, global variables

(A) * Use bottom up approach to solve recursions
Be careful when print or multiplication/addition is performed after recursive call and static or global variables are used.

* `int * a = malloc (sizeof (int));` // Works, NO error
`int * b = (int *) malloc (sizeof 1);` // Works, NO error



* `int a[6] = {1,2,3,4,5,6};` // sizeof (x) = 6 * 4 (↓ sizeof(int))
`int *ptr = (int *) (&x + 1);` // type of &x is int (*)[6]
`printf ("%d, %d", *(x+1), *(ptr-1));`

// Output = 2, 6

* Sizeof require brackets for datatypes (i.e int, char, float, ...)
HOWEVER brackets are not necessary for literals & variables (1, 2.1, a, b)

* Pointer type - can perform comparison (<, >, <=, ...) and (+, -, ++, --, ...)

- Void pointer — (+, -, ++, -- not allowed in this)
- Wild Pointer / Bad Pointer — i.e uninitialized
- Dangling Pointer — i.e pointed to a dead or invalid location WHICH WAS valid in past
- Function pointer — datatype (*name) (parameter types) = &functionName;

Data Structures

* A labeled rooted binary tree can NOT be uniquely constructed given its postorder and preorder traversals.
HOWEVER, inorder & postorder can
inorder & preorder can

* In tree -
- Internal node - atleast one children (HOWEVER root node is always internal node even if it is the only node in the tree)
- Leaf node - no children

* Maximum number of Binary trees (Binary Search Tree) that can be formed with n unlabeled node = $\frac{^{2n}C_n}{n+1}$ = Binary Search Tree with n different nodes
(Catalan Number)

If n labeled nodes then Binary Trees = $\frac{^{2n}C_n}{n+1} \cdot n!$

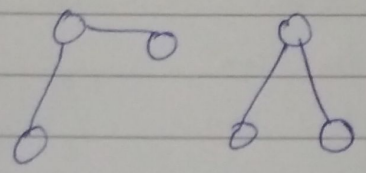
* Time complexity to construct binary search tree from (Inorder, Postorder) or (Inorder, Preorder) traversal is $\Omega(n)$, $\theta(n)$, $O(n)$

* Number of ways to insert 1, 2, ... n into Binary Search Tree such that height of tree is $(n-1)$ = 2^{n-1}
(Note - the height of a tree with a single node is 0)
Explanation - only the leftmost or the rightmost node can be root at each step.
Eg - 1, 2, 3, 4, 5, 6 - First, either 1 or 6 can be root
- If 1 is root, then either 2 or 6 can be root...

Date:
Page No:

* If DFS of graph G with n vertices, k edges are marked as tree edges. Then number of connected components = n-k

eg-



n = 6

k = 4

components = 6 - 4 = 2

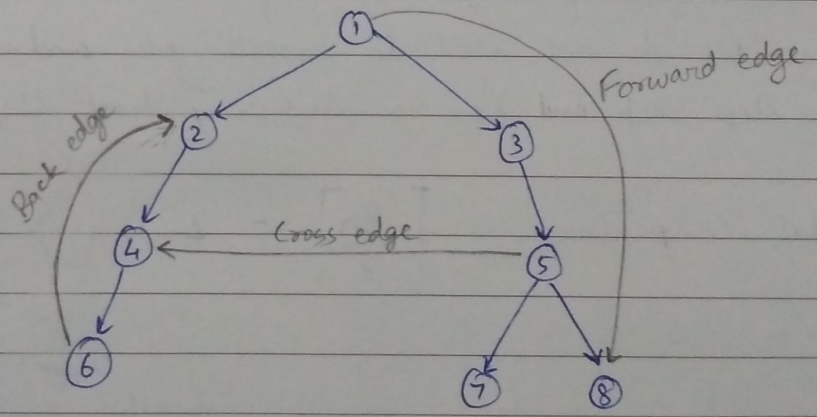
* In Graphs

Back edge – an edge that is from a node to itself (self-loop) or one of its ancestors in the tree produced by DFS.
Number of back edges ≠ Number of cycles
Edge 6 to 2

Forward edge – edge (u, v) such that "v" is descendant and the edge is not part of the DFS tree
edge 1 to 8

Cross edge – an edge which connects two nodes such that they do not have any ancestor or descendant relationship between them.
edge 5 to 4

Tree edge – all edges which are present in tree obtained after applying DFS on the graph.
(All edges marked with pen are tree edges)



* Number of inversion in an array of size $n = \frac{n(n-1)}{2}$

↓
Number of swaps required to convert array to sorted

* Minimum number of comparisons to find minimum and maximum of "n" numbers (Refer - geeksforgeeks)

1. Take first two numbers and compare them and ~~assign~~ assign them to min, max

```
if (arr[0] < arr[1]) {
    min = arr[0]
    max = arr[1]
} else {
    min = arr[1]
    max = arr[0]
}
```

2. Starting from index 2 (in a zero 0 indexed array)
   a. Pick 2 elements (a, b), compare them. (say a > b)
   b. Update min by comparing (min, b)
   c. Update max by comparing (max, a)

Therefore, we need 3 comparisons for each 2 elements, so total number of required comparisons will be $3\left(\frac{n}{2}\right) - 2$, because we do not need to update min and max like step 2.b and 2.c in the very first step.

Total comparisons = $\left\lceil \frac{3n}{2} \right\rceil - 2$ (works for even & odd values of n)

* Let $G = (V, E)$ be any connected undirected edge weighted graph.
[The weights of the edges of E are +ve & distinct.
⇒ Minimum spanning tree of G is always unique.
G has a unique Minimum spanning tree, if, for every cut of G, there is a unique minimum weight edge crossing the cut.

* Time complexity to compute transitive closure of a binary relation on a set of n elements $= O(n^3)$

* If Input array is sorted, then time complexity is
  - Bubble sort $= O(n)$
  - Insertion sort $= O(n)$
  - Merge sort $= O(n \log n)$
  - Quick sort $= O(n^2)$

* Number of elements that can be sorted in $\theta(\log n)$ using heap sort $= \theta\left(\frac{\log n}{\log \log n}\right)$

* Solve recursion using back substitution, option substitution

* Bellman - Ford shortest path algorithm
  1. Finds whether any -ve weighted cycle is reachable from the source
  2. NOT always can it find -ve weighted cycle

* Find median $= O(n)$

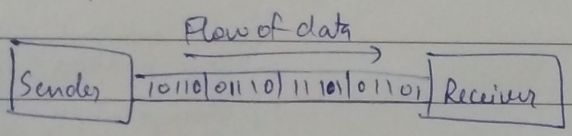
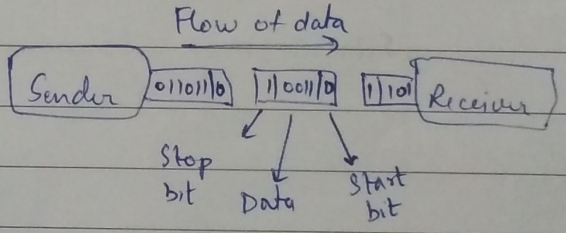
* Quick sort, where pivot always divides array in $\frac{n}{\alpha}$ and $(1-\frac{1}{\alpha})n$
  i.e $T(n) = T\left(\frac{n}{\alpha}\right) + T\left(\frac{\alpha-1}{\alpha} n\right) + O(n)$

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

* Time complexity to sort (merge sort) array of length n of string of length n $= O(n^2 \log n)$

* Let G be a weighted connected undirected graph with distinct +ve edge weights. **IF** every edge weight is increased by the same value, **then**

1. Minimum spanning tree of G does **NOT** change. - **TRUE**
2. Shortest path between any pair of vertices does **NOT** change - **FALSE**

*	Synchronous Transmission	Asynchronous Transmission
1	Data is sent in the form of blocks or frames	• Data is sent in the form of bytes or character
2	Fast	• Slow
3	Costly	• Economical
4	Time interval of transmission is constant	• Time interval of transmission is random
5	NO gap between data	• Gap present between data
6	Extra bits **not** considered part of data. Removed from the group of data bits at receiver end.	• Extra bits are considered part of data, hence not removed at receiver's site
7		• 

* Socket - create a new end point allocated table space for it within transport entity

B Bind - created sockets are assigned address

L Listen - (to Passive Socket) It allocate space to queue incoming calls for the case that several clients try to connect at same time.

A Accept - To block, waiting for an incoming connection, server executes accept primitive.

Book Pg 77, 78, 76

* In TCP, the accept() function does not assign the client to a new ephemeral port, instead the same port is used. The tuple (local address, local port, remote address, remote port) uniquely identifies a TCP connection. Hence one server can handle any number of clients on the same source IP and port.

PDU - Protocol Data Unit

APDU Application layer / Desktop layer
HTTP
FTP
SMTP - send mail
POP3 - download mail from server
IMAP - server mail management
MIME - send data other than ASCII

- Network Virtual Terminal
- Mail Services
- Directory Services
- File Transfer (FTAM) access & management
- Browser

(Presentation layer)

PPDU Presentation layer / Translation layer
→ IMAP
SSL
TLS
Telnet

(eg ASCII to EBCDIC)

- Encryption, Encoding / Translation
- Compression,
- Check data format
- Deal with UI
- Encapsulate components & enhance interface.

SPDU Session Layer

- Session establishment, maintenance and termination
- Synchronization - add check points / synchronization points to identify errors to avoid data loss and premature cutting of end of message
- Dialog controller - allow systems to start communication in either half duplex or full duplex.

Data is called Segments

TPDU Transport layer
TCP,
UDP

Adds source & destination Port number (Destination Port Required) → Also called Service Point Address

It is part of OS

Heart of OSI model

Functions:
- End to End delivery of complete message
- Flow and Error control
- Segmentation and Reassembly
- Service Point Addressing (Port number) - ensures data is delivered to the correct process.

Services:
- Connection Oriented Service (3 phases) - reliable and secure. Acknowledgement is sent
- Connectionless Service (1 phase) - faster, no acknowledgement

Acknowledgement Number
Sequence Number = 32 bits → One unique number for each byte
Window size = 16 bits → $2^{16} \times 2^{14}$

Segment is referred as Packets/ ~~Datagram~~

IP Address

Implemented by ~~networking devices~~ such as routers

Packet is referred as **Frame**

MAC Address destination ~~MAC address from IP~~ obtained using **ARP** (Address Resolution Protocol) segment onto the wire

Handled by the **NIC** (~~Network Interface Card~~) and device drivers of host machine

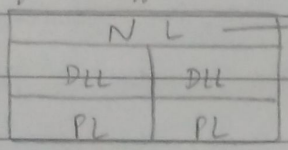
Bits

→ Data should be multiple of 64 bits / 8 bytes

Network layer - router, multilayer switch

Total Length (Bytes) = 16 bit
~~Identification~~
~~Sequence Number~~ = 16 bit
Fragment offset = 13 bits (measure in 64 bit units)

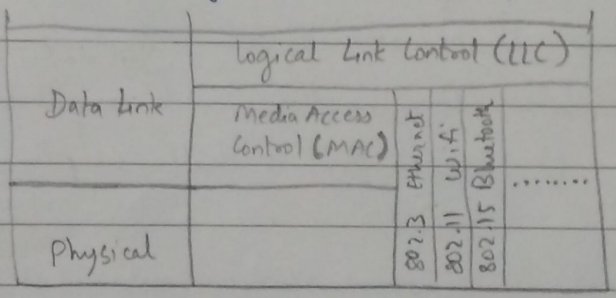
Router layers



Network layer

Protocols - NAT, IP, ICMP, ARP, RIP, OSPF, BGP

Data link layer - switch, bridge



Protocols - Stop & Wait, Sliding Window, Point to Point

Physical layer - hub, repeater, modem, cables

Host to Host
- Routing
- Congestion control
- Allow interconnection of heterogenous networks
- Concerned with controlling the operation of the subnet
- Packet need not be in the order in which they were ~~transmitted~~.

Link to Link
Node to Node / Hop to Hop
- Framing
- Physical Addressing
- Error control - detects and retransmits damaged or lost frames
- Flow control
- Access control - when a single communication channel is shared by multiple devices, MAC sub-layer helps to determine which device has control over the channel at a given time

- Bit synchronization
- Bit rate control
- Physical topologies - eg. star, mesh
- Transmission mode - simplex, half duplex, full duplex

Software layers → Host of OSI → Hardware layers

Application
Presentation
Session
Transport
Network
Data Link
Physical

Trivial File transfer protocol ← TFTP uses UDP
FTP uses TCP

* FTP - port 20 for data
port 21 for control connection

* Nagle algorithm - used to automatically concatenate a number of small buffer messages. It increases the efficiency of a network application system by reducing the number of packets that must be sent. It requires that the sender sends the first segment even if it's a small one, then it wait until an ack is received or a Maximum Segment size (MSS) is accumulated.
This algorithm can introduce delay as it sends only one data segment per round trip. This algorithm is turned off for those applications that require data to be immediately send.
Clark's solution closes the window until another segment of MSS can be received or the buffer is half empty.

Nagle's algorithm - used when sender creates silly window syndrome
Clark's solution - used when receiver creates silly window syndrome.

* Minimum interfaces for router = 2

* IEEE 802.3 - defining physical layer and DLL media access control of wired ethernet. Contention Protocol
IEEE 802.5 - the token passing scheme is used in place of CSMA/CD on a ring topology LAN. Contentionless Protocol

* ARP (Address Resolution Protocol) - find MAC address if IP address is known.
- It broadcasts the ARP request if destination is same subnet
- It takes help of default gateway if destination in different subnet.

COA

* Cache

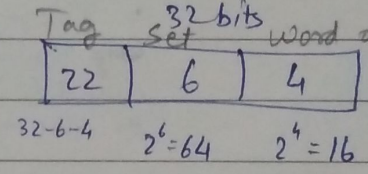
Conflict miss possible
1. Direct mapping
2. Set associative

Conflict miss not possible with LRU replacement
3. Fully associative

If any question asks total size of tags in the cache directory or size of cache tag directory, always use total number of **blocks** for calculating the results **NOT** just **sets**

When calculating size of metadata (tags) for cache or cache tag directory, use (tag bits + valid bit + modified bit + replacement bits)

Q1 * A 17 way set associative cache has 16 bytes blocks and 32 bits byte addressable memory. The cache size is 17408. The total bits required for both tag and word offset for any CPU reference is ____



16 bytes/block $\Rightarrow \log_2 16 = 4$ bits

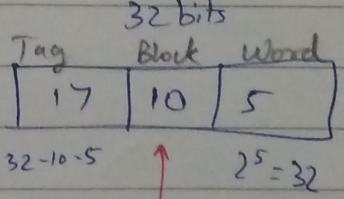
$$\text{Cache blocks} = \frac{17408\ B}{16\ B} = 1088 \text{ blocks}$$

$$\text{Cache sets} = \frac{1088}{17} = 64 \text{ sets} \Longrightarrow \log_2 64 = 6 \text{ bits}$$

Ans = 22 + 4 = 26

Q2 Consider a Direct mapped cache of size 32KB with block size 32 bytes. CPU generates 32 bit address. The number of bits needed for cache indexing and number of tag bits are-

Cache size = 32KB = 2^{15} B



Block size = 32B = 2^5 B

Cache lines (Blocks) = $\frac{32KB}{32B} = 2^{10}$

$\Rightarrow$ 10 bits required for indexing

* Minimum number of page colour bits = Index bits (Cache) + Offset bits − Page Index Bits ($\log_2$(Page Size))

Minimum page colours = $2^{\text{Minimum number of page colour bits}}$

* 3 Types of misses
1. Compulsory miss – occurs when block is brought first time in cache.
2. Conflict miss – misses that would not occure if cache were fully associative with LRU replacement
3. Capacity miss – (occur only in fully associative ← maybe cache)
   The program is actively using more data than the cache can hold

(Other link – gateoverflow.in/118745)

* Consider a 5 stage pipeline. Let P be the probability of branch instruction. Value of P such that speedup is atleast 4. Assume each stage takes 1 cycle and branch is predicted on 4th stage of the pipeline.

$$\text{Speed up} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

(Pipeline depth = 5; Branch frequency = P; Branch penalty = 3 → one less than cycles for branch instruction)

As branch is predicted in 4th stage of pipeline,

Branch penalty = 3

$$\frac{5}{1 + P \times 3} \geq 4$$

$$\therefore \quad 5 \geq 4 + 12P$$

$$\therefore \quad P \leq 1/12$$

* Consider a 5 stage pipeline. Assume no cycle time overhead of pipelining. When the application is executed on 5 stage pipeline, then how many ~~instruct~~ instruction incure 3 pipeline ~~cycle~~ stall cycles if the speedup achieved w.r.t non-pipeline is 3.

$$\text{Speed up} = \frac{\text{Pipeline depth}}{1 + \text{Number of stalls per instruction}}$$

$$3 = \frac{5}{1 + 3x}$$

$$\therefore \quad 3 + 9x = 5$$

$$\therefore \quad x = 2/9$$

i.e 22.22 %.

OS

Inverted page table (pg 89 for theory)

eg- A computer system of 32 bit logical address support upto 64 MB of physical memory. If page size is 8 KB then what is the size of inverted page table?

A. 24 KB
B. 25 KB
C. 26 KB
D. 23 KB

Solution - Number of pages = $\frac{2^{32}}{2^{13}} = 2^{19}$

∴ Bits required to identify each page = 19 bits (3 Bytes)

Number of frames = $\frac{2^{26}}{2^{13}} = 2^{13}$

Deadlock - PAD
- Prevention: Mutual Exclusion, Hold & wait, Preemption, Circular wait
- Avoidance: Bankers, DAG, RAG
- Detection: RAG

Inverted page table size = Number of frames * number of bits required to represent each page

$= 2^{13} * 3$ bytes

$= 24$ KB

∴ Option A is correct

NOTE - number of entries in Inverted page table = Number of frames in Physical Address Space (PAS) = $\frac{\text{Physical address space size}}{\text{Page size}}$

In General number of entries in Page Table = $\frac{\text{virtual address space size}}{\text{page size}}$

* SJF gives least waiting time, turn around time & max throughput (Non preemptive)

SJF preemptive SRTF - least waiting time, turn around time & max throughput

* With LRU page replacement policy, when page size is halved, page fault can be ~~double~~ but NOT more

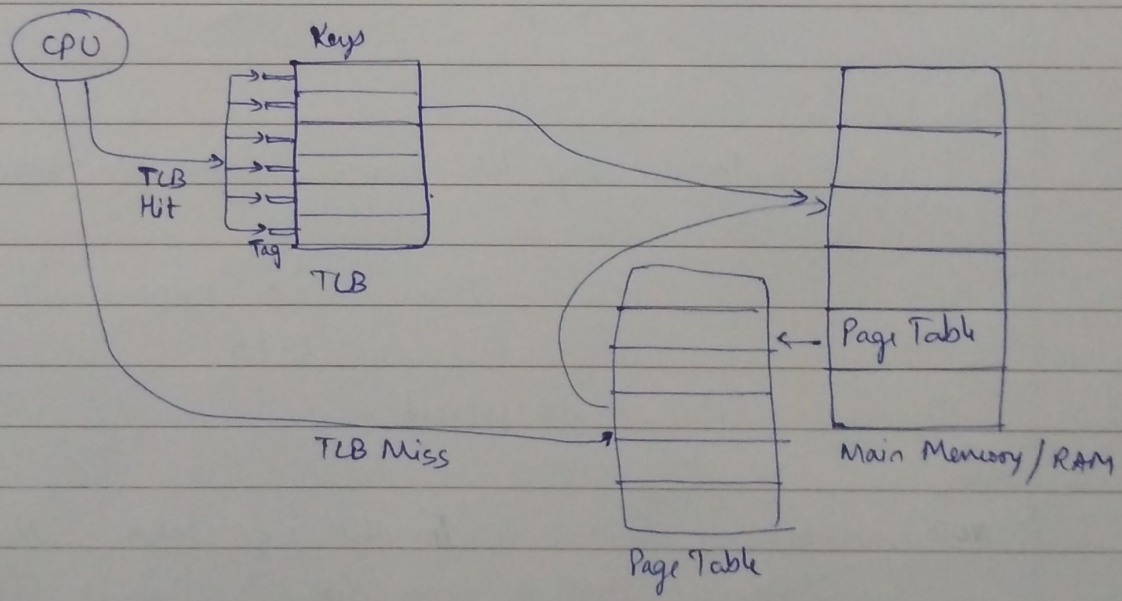
Consider virtual memory with page size of 2KB, virtual address is 32 bits and physical address is 32 bit. For each page, it also stores additional 3 bits for protection. What is the size of page table (MB) if single level page table is used —

$$\text{Number of pages} = \frac{2^{32}}{2^{11}} = 2^{21} \quad \leftarrow 2KB$$

Page table entry size = 21 + 3 bits = 3 Bytes

Page table size = $2^{21} \times 3$ = 6 MB

TLB - Translation lookaside Buffer



Effective Access Time (EAT) = $p(t+m) + (1-p)(t + km + m)$

- TLB hit rate
- TLB access time
- memory access time
- Level of paging (eg - k=1 for single level paging)

* Physically accessed cache

CPU ↔ TLB ↔ Cache ↔ RAM

Virtually accessed cache

CPU ↔ Cache ↔ TLB ↔ RAM

DBMS

* Read - write conflict - unrepeatable reads
Write - read conflict - reading uncommitted data / dirty read
Write - write conflict - overwriting uncommitted data / blind writes

Strict 2 Phase locking prevents all the above conflicts

* Read write conflict

T_1	T_2
R(A)	
	R(A)
	W(A)
	Commit
R(A)	
W(A)	
commit	

Write read conflict

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
R(B)	
W(B)	
Commit	

Write Write Conflict

T_1	T_2
W(A)	
	W(B)
W(B)	
Commit	
	W(A)
	Commit

* First record of data block is called Anchor Record / Block Anchor

TOC

* Finite Automata (FA)

- FA with output
  - Moore — Every state has output
  - Mealy — Every transition has output
- FA without output
  - DFA
  - NFA
  - ε-NFA

* Regular language (RL)

- Generator
  - Regular Grammar (V_T, P, S)
    - Right Linear Grammar
    - Left Linear Grammar
- Representer
  - Regular Expression (RE)
- Acceptor
  - Finite Automata (FA)
    - With Output $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$
      - Moore — state
      - Mealy — transition
    - Without Output $(Q, \Sigma, \delta, q_0, F)$
      - DFA
      - NFA
      - ε-NFA

* Decidable language - Deterministic TM exists that accepts every string of L and rejects every string L'
  - Every Regular language, CFL, CSL, Recursive Language

A+

Semi Decidable - Recursive Enumerable language

Undecidable language - i.e not accepted by any Deterministic TM that halts on every input.
- Non Recursive Enumerable language, halting problem of TM, equality of CFGs, ambiguity of CFGs, Language of CFG is RL? language of CFG is Σ^* or not?
- TM is Empty, finite, infinite, RL, CFL, CSL, Recursive, is Σ^*? $L = L^R$, L contains atleast n string?

PTO

* Other (i.e Non REL)
Recursive Enumerable (REL) Type 0 $(Q, \Sigma, \Gamma, B, \delta, q_0, F)$ 7
Complementation — Recursive (REC)
— Context Sensitive (CSL) Type 1
Context Free (CFL) — Type 2 $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ 7
— Deterministic Context Free (DCFL)
— Regular (RL) Type 3 $(Q, \Sigma, \delta, q_0, F)$ 5

* Countable set - set of all TMs or REL
set of all LBAs or CSLs
set of all PDAs or CFLs
set of all FAs or Regular language
Σ^*

Un-Countable set - 2^{Σ^*}
- set of Non-REL

* DFA = NFA = ϵ-NFA Regular Language

ϵ-NFA + Stack = PDA = FA + Stack Context Free Language

NPDA > DPDA
LL(k) $\subset$ DCFL

Deterministic TM = Non Deterministic TM
= Multi track TM
= Multi tape TM
= Multi head TM
= Multi dimensional TM
= Offline TM
= Universal TM
= FA + 2 Stacks
= FA + 2 counters (Read Only)

Computable / Enumerable / Acceptable
Turing Decidable Language = Recursive Language
Turing Recognizable Language = Recursive Enumerable Language

1. Membership
2. Emptiness
3. Finiteness / Infiniteness
4. Equivalence, Intersection, Completeness, Subset
5. Ambiguity
6. Regularity
7. Disjointedness
8. Everything....

Type 3 Regular language - all are decidable

Type 2 CFL - 1, 2, 3

Type 1 CFG, REC - 1

Type 0 REL - nothing

A+ * Reducibility

$A \leq_m B$ ⇒ A cannot be harder than B

1. B is decidable ⇒ A is decidable
2. B is recognizable ⇒ A is recognizable
3. A is undecidable ⇒ B is undecidable
4. A is unrecognizable (↳ Recursive Enumerable) ⇒ B is unrecognizable

B is recursive enumerable ⇒ A is recursive enumerable

A is **NOT** recursive enumerable ⇒ B is **NOT** recursive enumerable

A+ * UICKRC

		Union	Intersection	Complementation	Kleene closure	Reversal	Concatenation
Type 3	Regular	✓	✓	✓	✓	✓	✓
Type 2	Deterministic CFL	x	x	✓	x	x	x
	CFL (Context Free)	✓	x	x ⇒ CSL	✓	✓	✓
Type 1	CSL (Context Sensitive)	✓	✓	✓	✓	✓	✓
Unrestricted Type 0 Grammar	Recursive (REC)	✓	✓	✓	✓	✓	✓
	Recursive Enumerable (REL)	✓	✓	x (⇒ NOT recursive enumerable)	x	✓	✓

Inverse Relation → R → relation
R^{-1} → inverse of a relation
$R^{-1} = \{(b,a) \mid (a,b) \in R\}$

DM

Date:	
Page No:	1

Diagonal Relation → $\Delta_A = \{(a,a) \mid a \in A\}$

Ch 4 - Set Theory

Complementary Relation → $\bar{R} = \{(a,b) \mid (a,b) \notin R\} = (A \times B) - R$

Reflexive Relation → $(x,x) \in R \ \forall x \in A$
(any superset of Δ_A $(2^{n(n-1)})$)

Irreflexive Relation (Anti) $-\Delta_A$
$(x,x) \notin R \ \forall x \in A$ $(2^{n(n-1)})$

A
Set - unordered collection of distinct objects

R
Subset of A×B ← Relation

Equivalence Relation
Reflexive, Symmetric, Transitive

Partial Order Relation
Reflexive
Anti-symmetric
Transitive

Symmetric Relation $(2^{\frac{n(n+1)}{2}})$
if $(x,y) \in R$ then $(y,x) \in R \ \forall x,y \in A$

Anti-Symmetric Relation $(2^n \cdot 3^{\frac{n(n-1)}{2}})$
if $(x \, R \, y)$ and $(y \, R \, x)$ then $x = y \ \forall x,y \in A$

Asymmetric Relation $(3^{\frac{n(n-1)}{2}})$
if $(x,y) \in R$ then $(y,x) \notin R \ \forall x,y \in A$

Transitive Relation
if $(x \, R \, y)$ & $(y \, R \, z)$ then $(x \, R \, z) \ \forall x,y,z \in A$

R^* Transitive closure
$R^{\#}$ Reflexive closure
R^+ Symmetric closure

Poset
Partially Ordered Set
[A; R] Set + Partial Order Relation

→ Totally Ordered set / linearly ordered set / chain
- every pair of elements of A are comparable
- Poset [A; R]
- Is always a lattice, distributive lattice

Meet Semi lattice
- Poset + GLB (0)

Join Semi Lattice
- Poset + LUB (I)

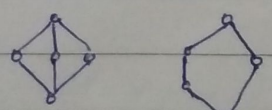
Lattice
[L, ∧, ∨]
- Poset
- Each pair of elements has a LUB & GLB
i.e Meet Semi lattice + Join Semi lattice

Distributive lattice
(L, ∨, ∧)
- Following laws hold

$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

- Sublattice not isomorphic to.



- Unique complement of element

Complemented lattice
- Bounded and every element in L has a complement

↳ x is complement of a if $a \vee x = I$ & $a \wedge x = 0$

Boolean Algebra
- lattice is distributive & complemented

Binary Operation (Closed Operation)

- $(a * b) \in A \quad \forall a, b \in A$

Algebraic Structure

- non empty set A, binary operation *
- * is a closed operation
- $(A, *)$

↓

Semi Group

- Algebraic Structure + Associativity

$$(a * b) * c = a * (b * c)$$

↓

Monoid

- Semi Group + Identity Element(e)

$$a * e = e * a = a$$

↓

Group

- Monoid + every element has inverse

$$a * b = b * a = e$$

a^{-1}

↓

Abelian Group

- Group + commutative

$$a * b = b * a$$

AI In C
SM G A

I
In
C

Identity element unique
Inverse of any element is unique
Inverse of identity $e = e$
$a * b = a * c \Rightarrow b = c$
$a * c = b * c \Rightarrow a = b$
$(a * b)^{-1} = b^{-1} * a^{-1}$

DM

Ch 5 - Probability and Statistics

Binomial Distribution / Bernoulli's Distribution

p = probability that event will occure
q = " " not occure = 1 - p

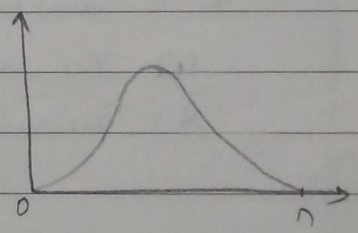
Probability that even will happen exactly x times in n trials

$$f(x) = P(X = x) = {}^{n}C_x \, p^x q^{n-x}$$

X - denotes number of success in n trials
x - range [0, n]

E(x)
$E((x-\mu)^2) = E(x^2) - [E(x)]^2$

Mean = $\mu = np$
Variance = $\sigma^2 = npq$
Standard deviation = $\sigma = \sqrt{npq}$



Poisson Distribution

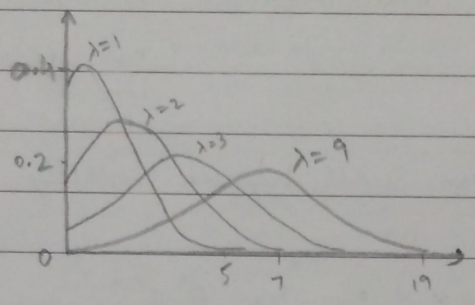
λ - positive constant, parameter of distribution
x - discrete random variable, range $[0, \infty)$

$$f(x) = P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Mean = $\mu = \lambda$
Variance = $\sigma^2 = \lambda$
Standard deviation = $\sigma = \sqrt{\lambda}$

$E(x^2) = \lambda^2 + \lambda$

- When n is large & p is small, then Binomial distribution is very closely approximated by poisson distribution. We can choose $\lambda = np$
- Poisson distribution is a limiting case of binomial distribution when, $n \to \infty$ and $p \to 0$



Normal Distribution / Gaussian Distribution

- It is another limiting form of binomial distribution, where
  (1) The number of trials "n" is indefinitely large
  (2) Neither "p" nor "q" is very small

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$-\infty < x < \infty$

$-\infty < \mu < \infty$

$\sigma > 0$

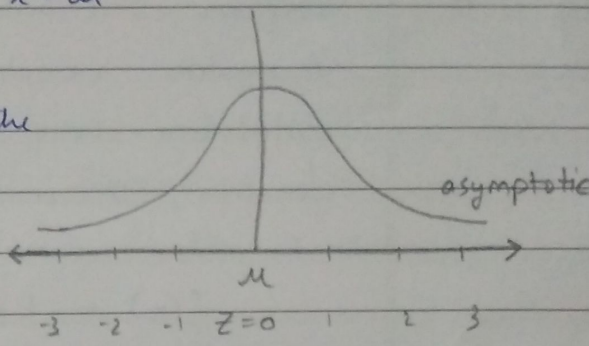
Mean $= \mu$

Standard deviation $= \sigma$

$$\frac{1}{\sigma\sqrt{2\pi}\, e^{\frac{(x-\mu)^2}{2\sigma^2}}}$$

- Bell shaped & symmetrical about $x = \mu$
- Mean, median, mode coincide
- Maximum probability occures at the point $x = \mu$,

$$P(x)_{max} = \frac{1}{\sigma\sqrt{2\pi}}$$



- Asymptotic to x-axis
- Area properties
  (1) $P(\mu - \sigma < X < \mu + \sigma) = 0.6826$
  (2) $P(\mu - 2\sigma < X < \mu + 2\sigma) = 0.9544$
  (3) $P(\mu - 3\sigma < X < \mu + 3\sigma) = 0.9973$
  (4) Area under normal curve is 1
  (5) $P(X > \mu) = P(X < \mu) = 0.5$

- Standard normal distribution

If we let $Z = \frac{X - \mu}{\sigma}$, then mean of $Z = 0$
variance $= 1$
Standard deviation $= 1$

Probability density function for Z is

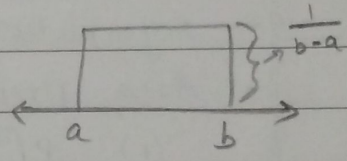
$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

$$P(a \leq X \leq b) = \int_a^b f(x)\,dx = \int_{z_1}^{z_2} f(z)\,dz$$

Uniform Distribution / Rectangular Distribution

A random variable X is said to have continuous uniform distribution over an interval (a, b) if probability density function is constant ($= k$ say) over the entire range of X.

$$f(x) = k = \frac{1}{b-a}, \quad a < x < b$$
$$= 0, \quad \text{otherwise}$$



$$\int_a^b f(x)\,dx = 1 \quad \Rightarrow \quad k = \frac{1}{b-a}$$

$$\text{Mean} = \mu = \frac{b+a}{2} \qquad E(X) \quad \int_a^b x f(x)\,dx$$

$$E(X^2) = \frac{b^2 + ba + a^2}{3} \qquad \int_a^b x^2 f(x)\,dx$$

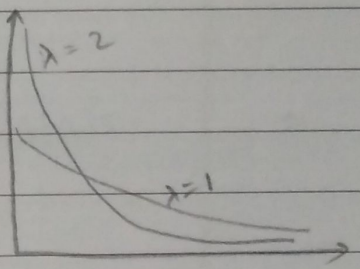
$$\text{Variance} = \underset{\sigma^2}{E((x-\mu)^2)} = \frac{(b-a)^2}{12} \qquad E(x^2) - [E(x)]^2$$

$$\text{Standard deviation} = \sigma = \sqrt{\frac{(b-a)^2}{12}}$$

Exponential Distribution

Continuous random variable x assuming non negative values is said to have an exponential distribution with parameter $\theta > 0$, if its probability density function is given by

$$f(x) = \theta e^{-\theta x} \quad x \geq 0$$
$$= 0 \quad \text{otherwise}$$



$$\text{Mean} = \mu = E(x) = \frac{1}{\theta}$$

$$\text{Variance} = \sigma^2 = E((x-\mu)^2) = \frac{1}{\theta^2}$$
$$= E(x^2) - E(x)^2$$
$$E(x^2) = \frac{2}{\theta^2}$$

$$\text{Standard deviation} = \sigma = \sqrt{\frac{1}{\theta^2}} = \frac{1}{\theta}$$

Mean (μ) ($\bar{x}$)

$P(X = x_i) = f(x_i)$

$$\mu = \bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} \quad \text{OR} \quad \sum_{i=1}^{n} x_i f(x_i)$$

Median - middle value of samples
- it is value of x for which $P(X \leq x) = P(X \geq x) = \frac{1}{2}$

Mode - sample point with high frequency
- has the greatest probability of occurring

Mean - Mode = 3 (mean - median)

Standard deviation $= \sigma = \sqrt{\frac{\Sigma (x_i - \mu)^2}{N}}$

\# $P(A/E) = \frac{P(A \cap E)}{P(E)}$

$P(A) = \sum_{i=1}^{n} P(B_i) P(A/B_i)$ $B_1, B_2, \ldots B_n$ be a set of exhaustive and mutually exclusive events

$P(B_i/A) = \frac{P(B_i) P(A/B_i)}{\sum_{i=1}^{n} P(B_i) P(A/B_i)} = \frac{P(B_i) P(A/B_i)}{P(A)}$

\# $\sum p(x) = 1$ $\int_{-\infty}^{\infty} f(x) dx = 1$

Expectation of X
Mean (μ)

$E(x) = \sum_{i=1}^{n} x_i P(x_i)$ $E(x) = \int_{-\infty}^{\infty} x f(x) dx$

Variance $(x) = E((x-\mu)^2) = E(x^2) - E(x)^2$
σ^2

Standard deviation $= \sigma = \sqrt{\text{variance}(x)}$