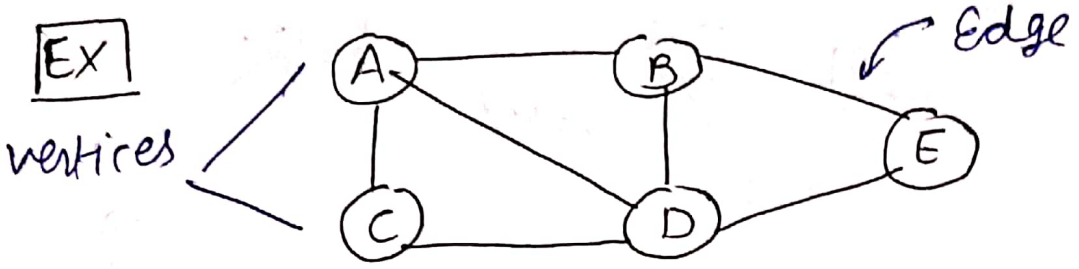
GRAPHS.

↳ Non Linear Data structure

> Graph is collection of vertices & arcs in which vertices are connected with arc

↳ Represented as $G = (V, E)$ V → vertices, E → Edges.

EX



$V = \{A, B, E, D, C\}$

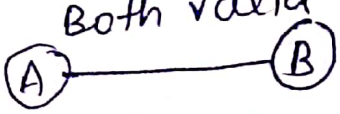
$E = \{AB, AC, AD, BD, BE, ED, CD\}$

Terminologies

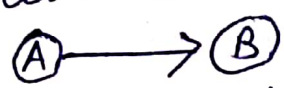
vertex - Individual element of graph
↳ also known as node.

Edge - connecting link between two vertices
↳ also known as arc.

- **undirected** (Bidirectional edge) (AB) (BA) Both valid

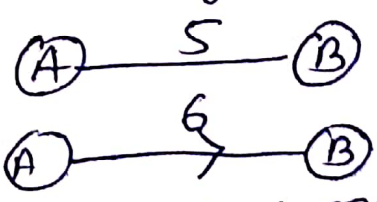


- **directed**



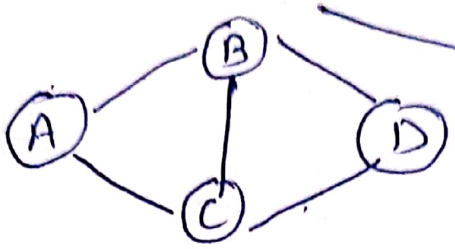
only (AB) is valid (unidirectional)

- **weighted**



have cost on it of moving

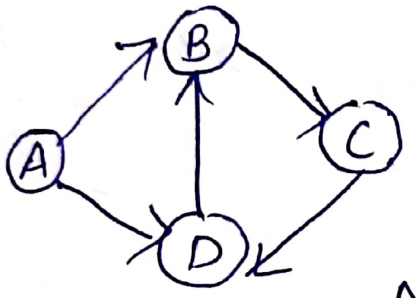
undirected graph - undirect Edges



Direction of moving is not given.

Directed graph - Directed ~~Eg~~ Edges

→ Direction of moving is given.



only move (AB, AD, BD, BC, CD)

No other Edge is Possible

Mixed - Both undirected or directed graph

End vertices or End points -

two vertices joined by Edge.

Adjacent :- ~~Edge~~ If there is Edge between two point they are called as adjacent.

Incident - If the vertex is one of the End points of that Edge

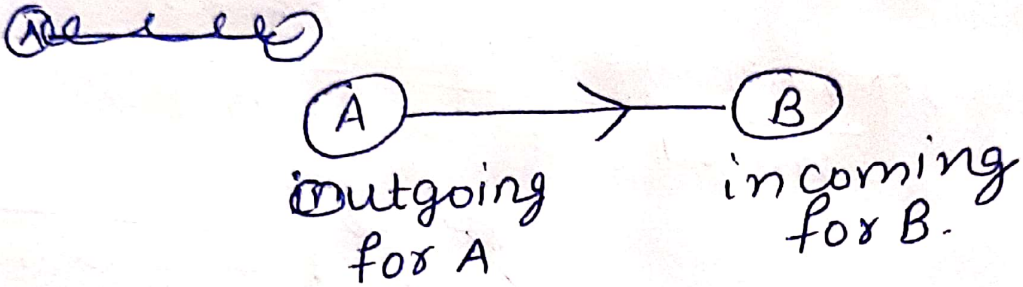
Degree - Total no. of Edges connected with a vertex.

In degree

Total no. of **In** coming Edges connected to vetex

outdegree

Total no. of Eges **outgoing** connected vertex



Self loop End point concide Each other

Path :- sequence of alternate vertices and eges start at a vertex & Ends at other.

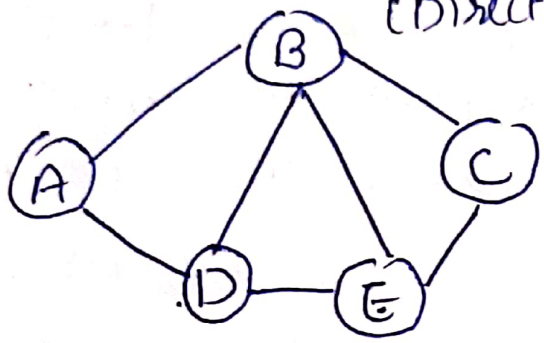
Graph Repersentation

- Adjacency matriv
- Incidence Matrix
- Adjacency list

Adjacency matrix

Matrix of order no. of vertices x no. of vertices.

Let the graph be (Directed)



	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	1
C	0	1	0	0	1
D	1	1	0	0	1
E	0	1	1	1	0

Put connected vetices 1

Not connected vertices = 0

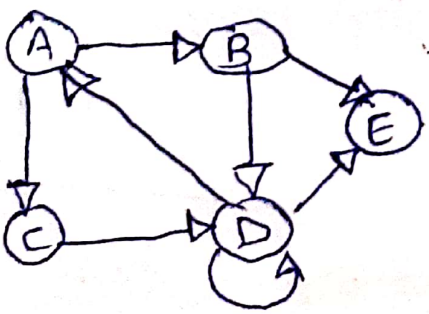
For AA, BB, CC, DD, EE - If self loop is there than 1 otherwise 0

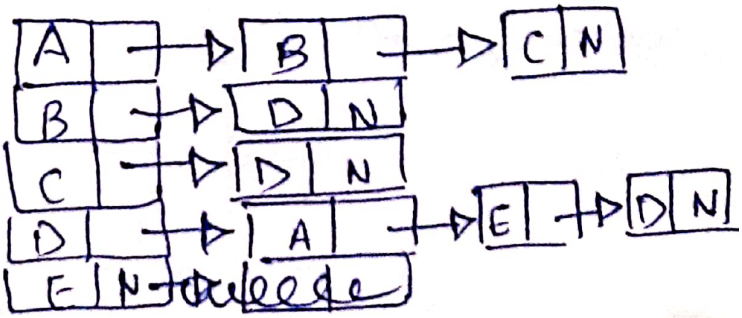
X

Incidence matrix

Adjacent list -

Every vertex of graph contains list of adjacent vertices





Graph traversal

DFS
{Depth first search}

BFS
{Breadth First Search}

DFS - {Depth first search}

Produces spanning tree as Result

Spanning Tree - Graph without loops (cycle include nhi honi chaiee)

↳ Stack Data Structure with Max size of total no. of vertices

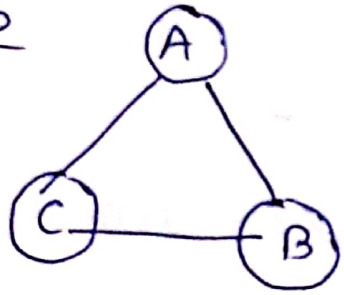
Steps
1. Define a stack of size total no. of vertices in graph.
2. Select any vertex as starting point visit the vertex & push it on the stack.
3. visit any one of non visited adjacent vertices of vertex which is at top of stack, & Push it on to the stack.
4. Repeat step 3, there is no new vertex is to be visited from vertex which is at top of stack.
5. when there is no new vertex, to visit then use Back tracking & pop one vertex from stack.

(6) Repeat 3, 4, 5 until stack becames Emnty.

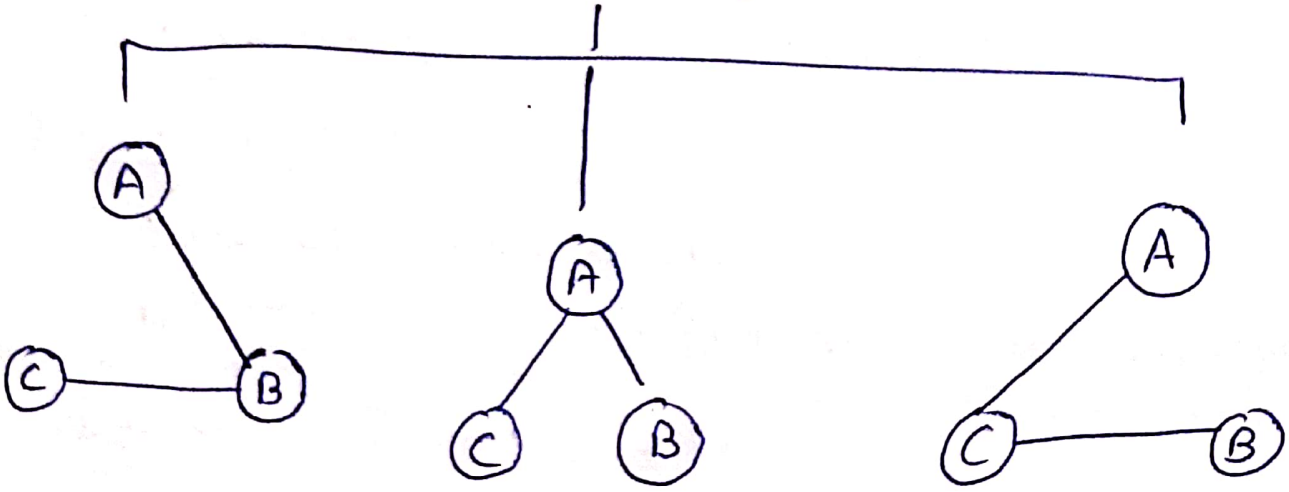
(7) when stack becames Empty produce final spanning tree by Removing unused Edges of the graph

Spanning tree Diagram

Let Graph be



Spanning trees



A complete undirected graph have n^{n-2} Spanning trees ↳ Max

Minimum no. of spanning tree = 1 a graph can have

★★ Applications

1. Civil network planning
2. computer Network Routing protocol
3. Cluster analysis

Minimum Spanning tree -

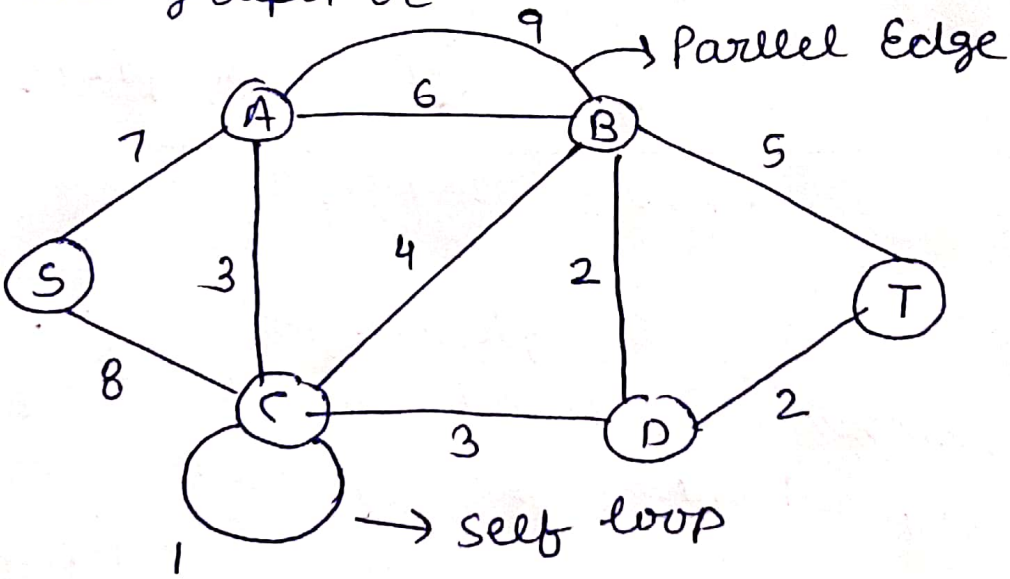
Has minimum weight than other spanning tree of same graph.

algorithms

- Kruskal's
- prim's

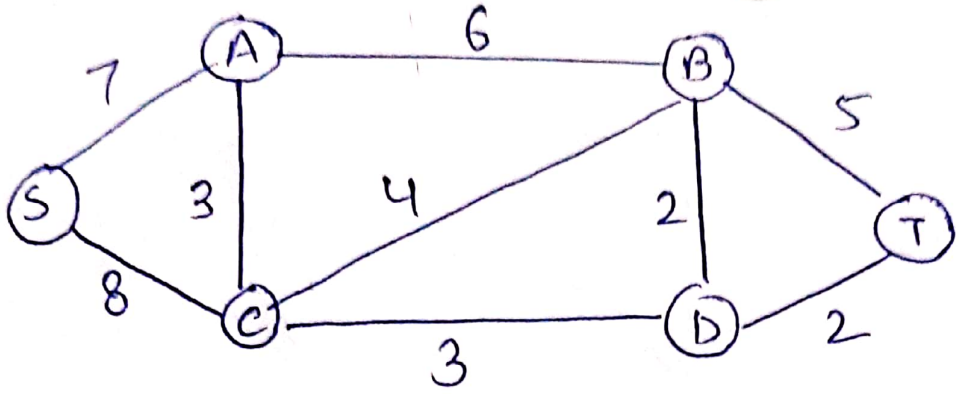
Kruskal's :- to find minimum cost spanning tree - uses Greedy approach.

Let the graph be



Step 1 - Remove parallel & self loop

{ Parallel mein vo Edge graph mein rakhni hai jiski cost kam hai }

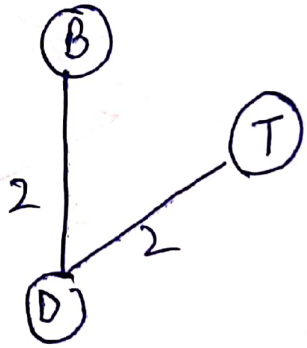


Step2 Arrange all edges in increasing order of weight/cost (ascending)

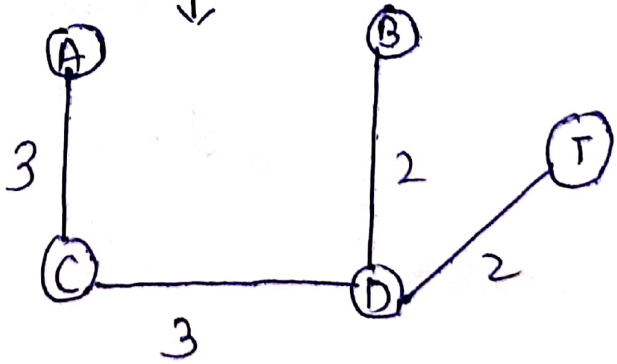
BD – 2
DT – 2
CD – 3
AC – 3
BC – 4
ABT – 5
AB – 6
AS – 7
SC – 8

Draw the grap by including less weight

1)

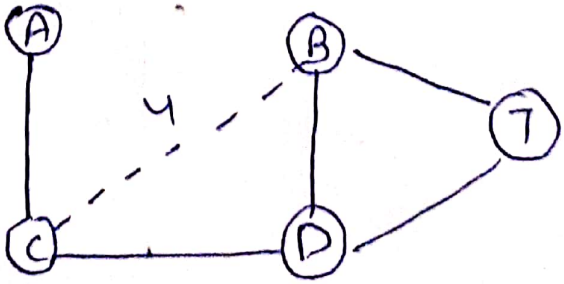


2)



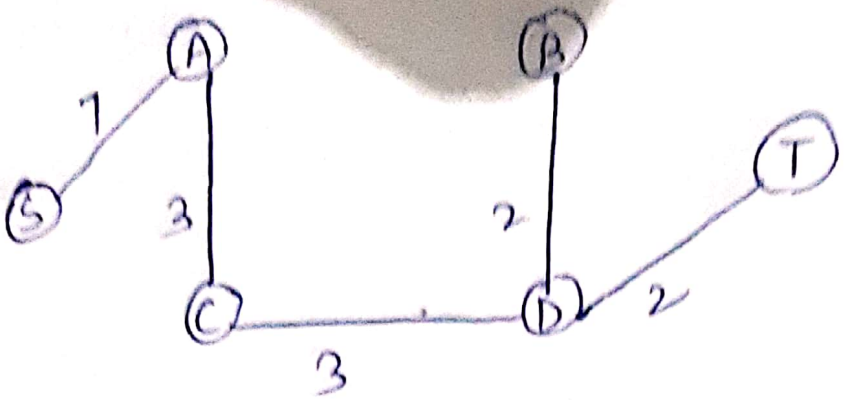
By including BC loop Bn raha hai, so usko neglect kar denge

3)



Same BT Ko, AB Ko neglect Karenge

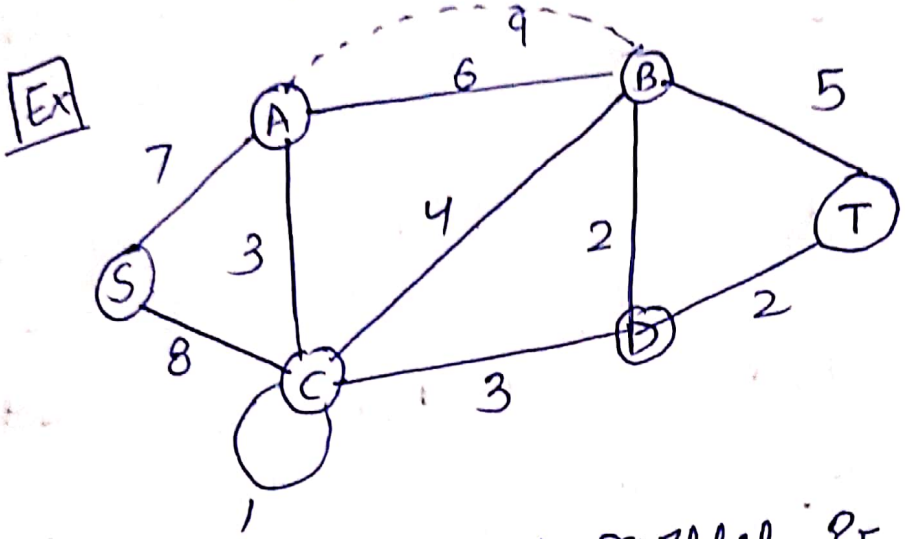
(4)



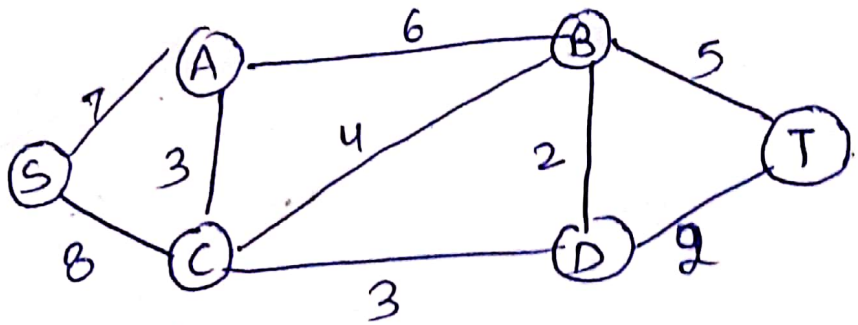
Prim's spanning tree Algo-

- minimum cost spanning tree
- uses greedy approach
- Shortest path algorithm

Ex



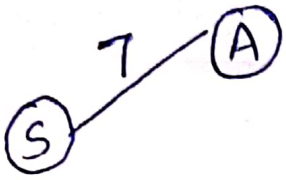
Step 1 Remove all parallel & Self loops



{In Parallel - Keep the one which has less weight}

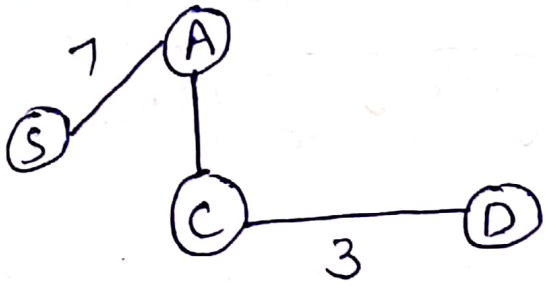
Step 2 Let take S point (start) compare all the outgoing weights & join which has less one

1)

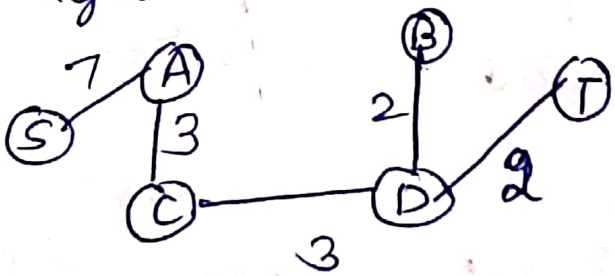


Repeat the process with all the vertices but make sure it will not make cyclic path

2)
3)



4) Since both DB, DT has same weight include both

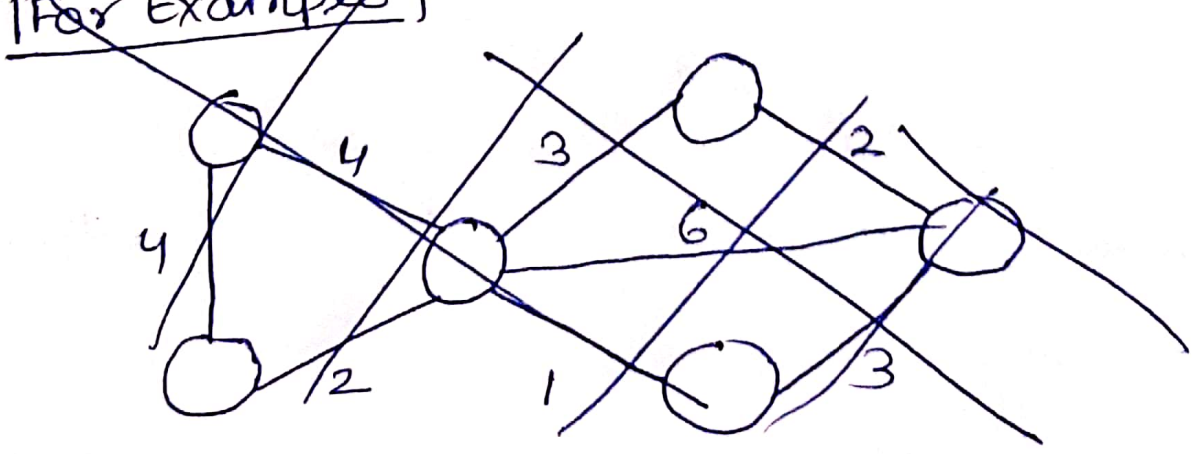


Kruskal's	Prim's
1) Select Edges based on weights	starts with vertex & grows the tree by selecting shortest Edge.
2) Based on sorted order	chooses Edges based on nearest neighbours
3) Runs faster in Sparse graphs.	Run faster in dense graphs
4) Prefer heap data structure	Prefer list data structure

Dijkstra's Algorithm

- Shortest path
- differs from Minimum Spanning tree. (It might not include all vertices)
- Greedy approach (End Result → Best solution for problem)

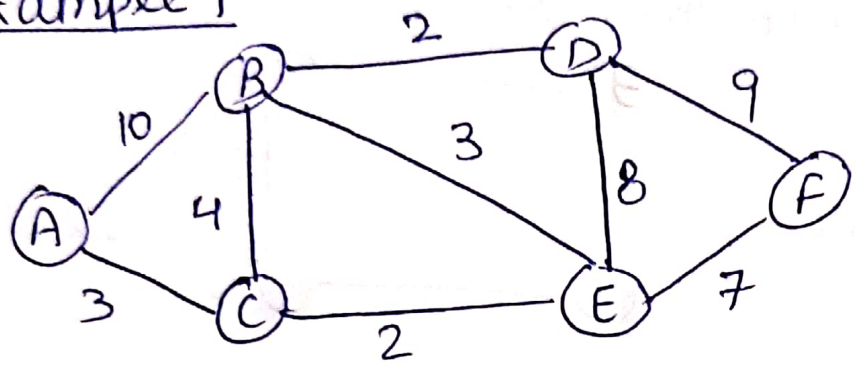
For Example



Applications

- To find the shortest path
- In social networking applications
- In telephone network
- To find the locations in map

Example



Let Start point be A.

(10, A) B

(∞, A) D

A

F (∞, A

C

(3, A)

E

(∞, A)

Join AC as it is shortest from A

(∞, AC)

(10, A)

(7, A) B

D

F (∞, AC)

A

3

C

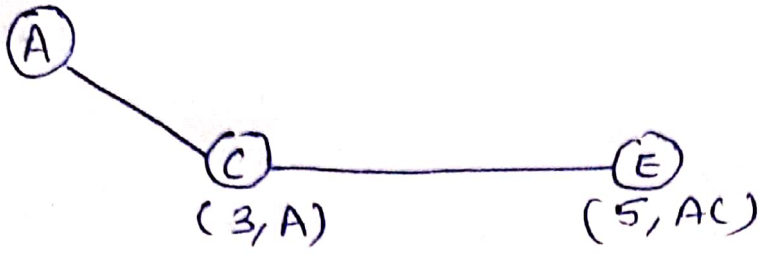
(3, A)

E

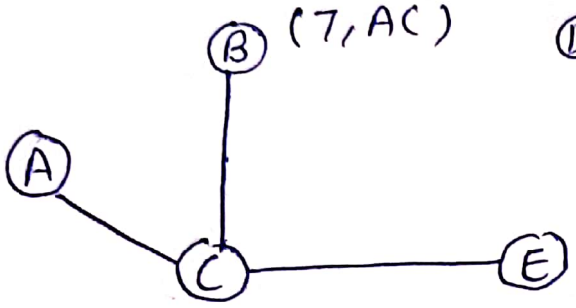
(5, AC)

join CE as it cover chort

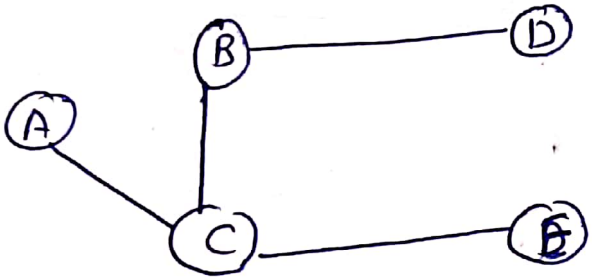
B (7, AC) D (13, ACE) F (12, ACE)



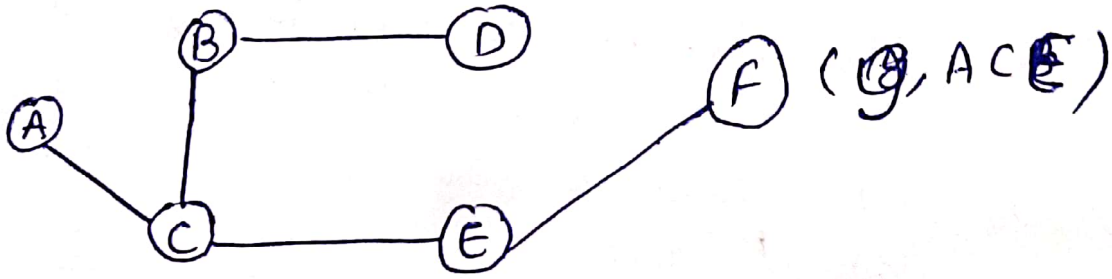
shortest → B(7, AC) (9, ACB) (13, ACE)



Join BD (13, ACE) (9, ACB) (18, ACBD) (9, ACE)



So, Join EF (shortest



Hashing

↳ algorithm use hoti hai convert karne ke lie input ko fixed size output mein.

↳ use hota hai, create karne ke lie unique identifier piece of data ko lie matlab ke data mein present sabhi input ko lie different index value data hai.

Hash Key - hash value / hash code
↳ fixed size numerical or alphanumerical Representation jo hashing algo se generate hoti hai

Hashing process mein involve hota hai, apply karna mathematical function ko input data pe, jo ek unique hash key bnata hai.

working - three steps

Input → Hash function → output

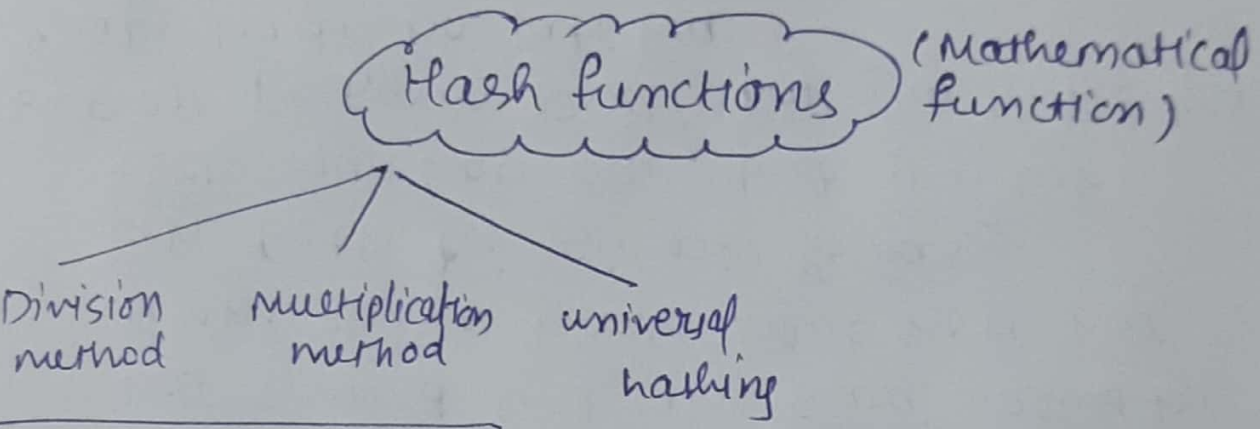
Input: jis input ko hash karna hai, usko hashing algorithm mein dalenge

Hash function: input data leke uspe ~~kaala function~~ ka mathematical function laga ke generate karenge fixed size hash

value, isko aise design kia jata hai ki

har input value different hash value produce kare, small change kare input mein or large change kare output mein.

Output - hash value return hoti hai,
- jo index ki lgh use hoti hai data ko retreive karne ke lie.



Division method - divide karna key ko table size mein or Remainder lena hash value

For Exp table size = 10 key = 23

hass value = 23 % 10 = 3

Multiplication method - involve multiply krna key value ko ek constant se or uske product ka fractional part lena as hash value.

key = 23, constant = 0.618.

hash value = 2.

(floor (10*(0.61823) - floor (0.61823))) = floor (2.239)

= 2.

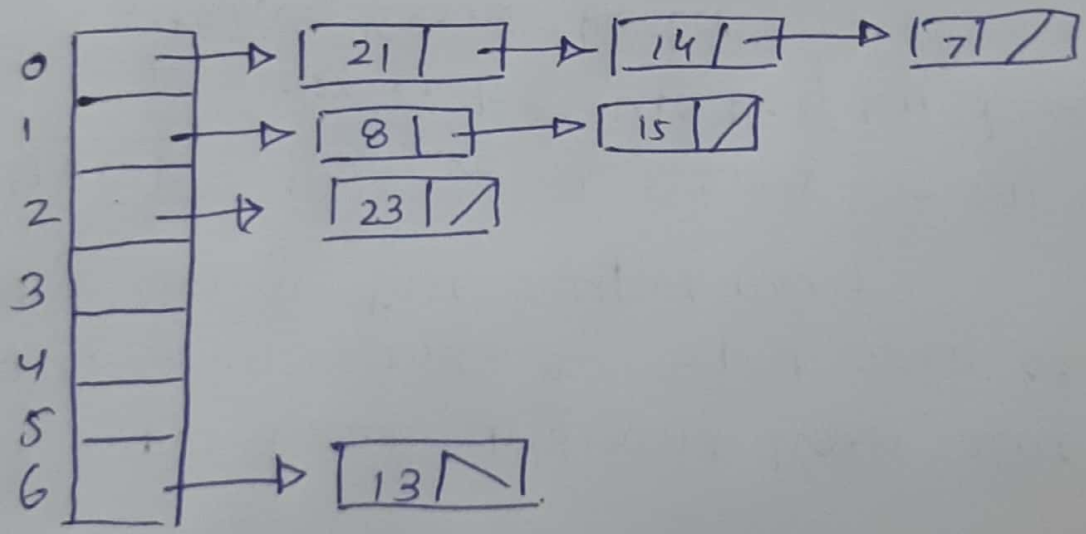
universal hashing - Random hash function kisi bhi family se, bs yeh dhyan rakhna hai hash function Biased na ho.

Collision Resolution

main challange hota hai hashing mein collision ko handle karna, jo hota hai two or more input values ko same hash value milne se.

techniques used to Resolve collision

↳ **chaining** - hash table contain karti hai linked list un sari values ki jinki same hash value hoti hai, yeh both hi simple & Easy technique hoti hai, but yeh bohot bekar perform karti hai jab linke list kaafi badi ho jati hai.



↳ open addressing: jab collision hota hai, jo algorithm hoti hai search karti hai Empty slot ke lie hash table mein by probing successive slots, jab tk empty slot nhi mil jata, this technique is efficient than Chaining.

↳ Double hashing - variation hota hai open addressing ka jo use karta hai second hash function, determine karne ke lie next slote, It Reduce clustering & improve process.

Hash table -

Hashing Hash table use karta hai key values ko store karne ke lie, hash table uske baad use karta hai hash function genrate karne ke lie index. isse one can perform insert, update & search operations.

↳ defined as Bucket of data" jaha Data Store hota hai array format mein. or inki apni index value hoti hai.

File organization -

↳ Ensure karta hai Record avilable ho processing ke lie.

↳ used hota hai determine karne ke lie Efficient file organization har Base relation ke lie

↳ Types

Sequential access -

- Storing & sorting contigous block main hota hai memory ke, jisko tape or disk mein store kia jata hai.
- har record sequential order mein hota hai, like arrange hota ascending ya descending order mein.
- Start hota hai beginning se file ki or record (new) add hota hai at the End.
- Sequential file mein yeh possible nhi hai, record ko middle mein add karna bina rewrite kare file ko.

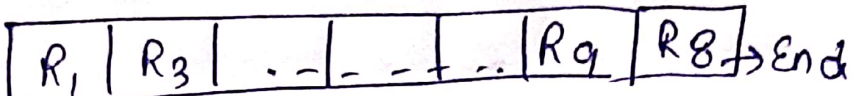
advantages

- Simple to program
- Easy to Design.
- best use of Storage space.

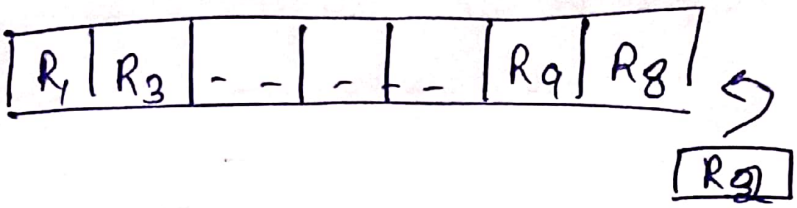
Disadvantages

time consuming process
high Data redudancy.
Random searching not possible

Diagram stating



Insertion



Direct access → Random access or Relative file organization

→ Record direct access storage device mein store hota hai Ex - hard disk,
→ Record ka sequence mein hona zarrori nhi hota.
→ useful for immideate access to large amount of information.
→ also known as hashing

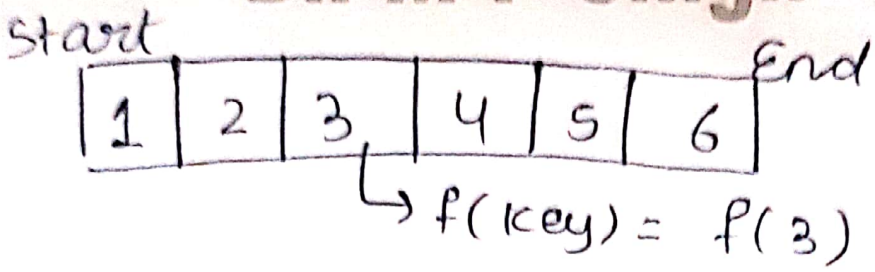
advantages

- update files Quickly
- Better control over Record allocation.
- access immediately

DISadvantages

- Expensive.
- less storage space.
- no back up facility.

Dr. M P Singh



Indexed Sequential access file organization

- combines both sequential & direct access file organization.
- Index karta hai sequential or record karta hai data ko jisse vo directly access ho ske, keys ki help se.
- Multiple keys hoti hai, alphanumeric hoti hai jisme record ko odered way me store karta hai.
- Sequentialey bhi access kar skte hai or Randomly bhi Index use karke.

Advantages

- very fast if Index table is properly organized.
- Record ko middle of file mein bhi insert kar skte hai
- Degree Reduce karta hai Sequential search ki

Disadvantages

- Require more storage space
- Expensive but Require Special Software
- less Efficient
- Requires unique keys