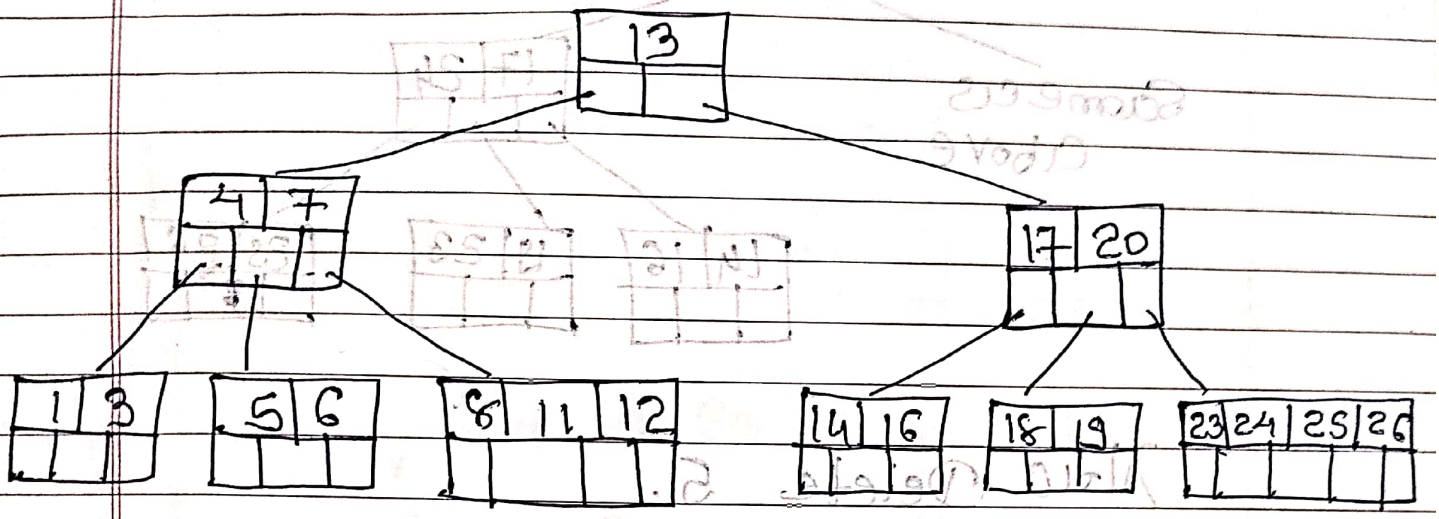
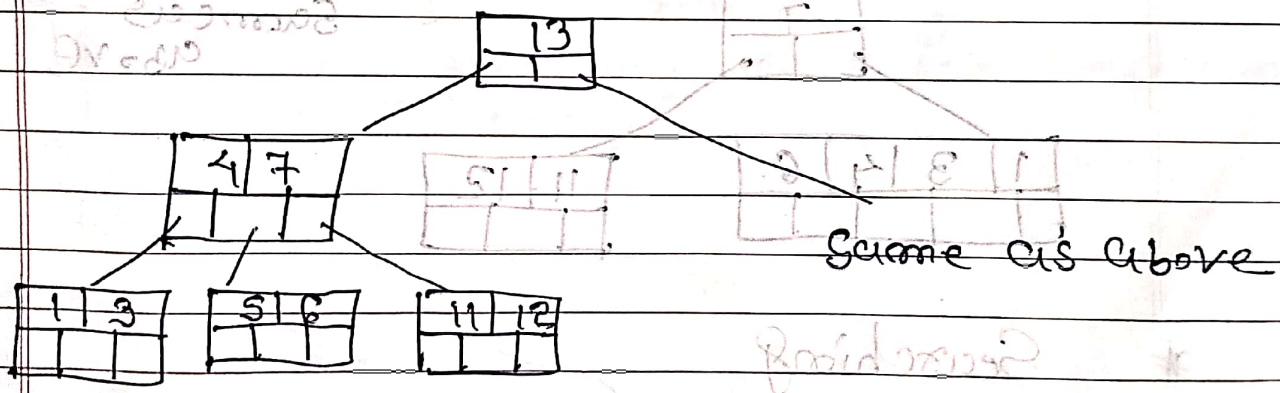
Deletion

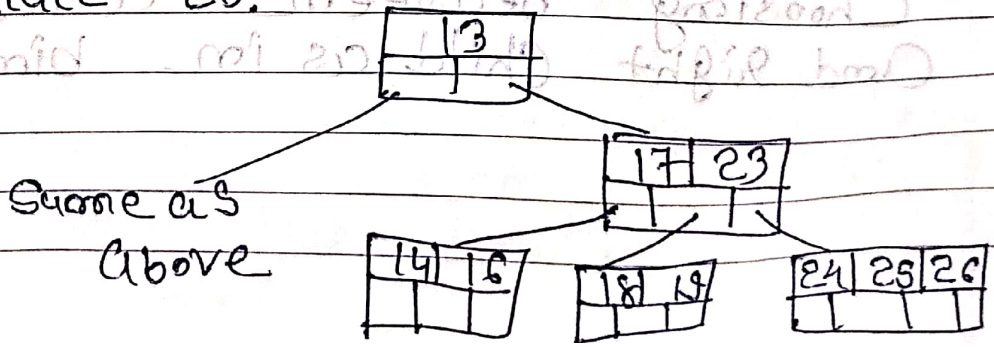
Consider a B- tree.



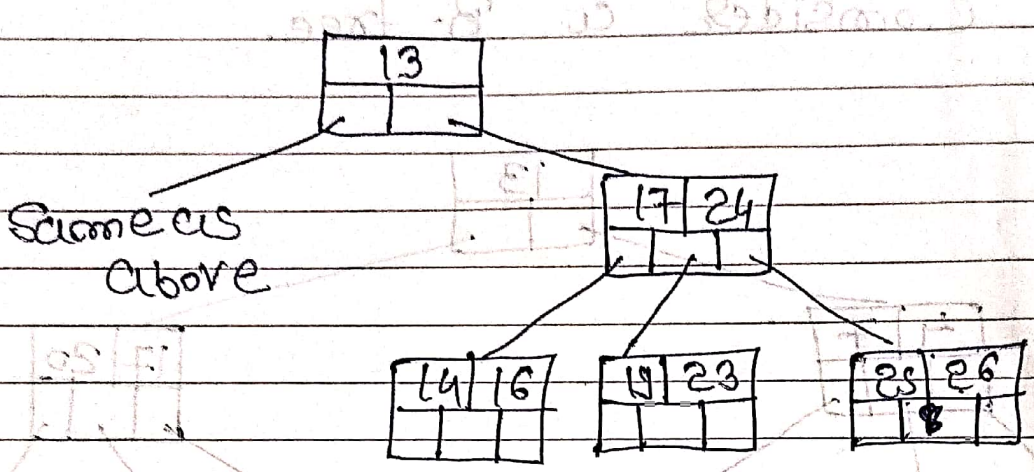
if we want to delete 8 & then it is very simple.



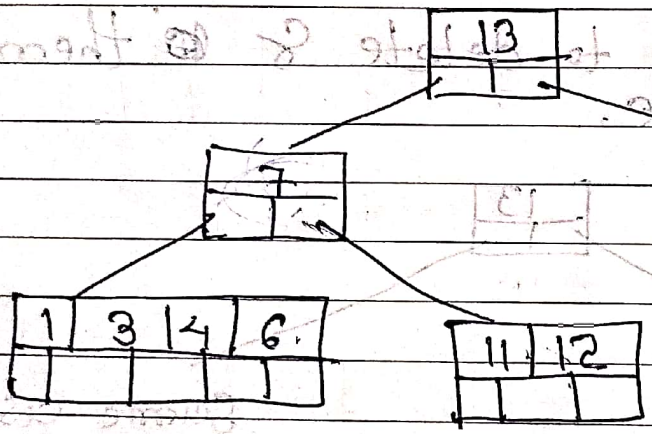
Now we will delete 20, Not in leaf node so we will find its Successor is 23 so 23 will be moved upto replace 20.



Next we will delete 18. (B-tree of order 5)



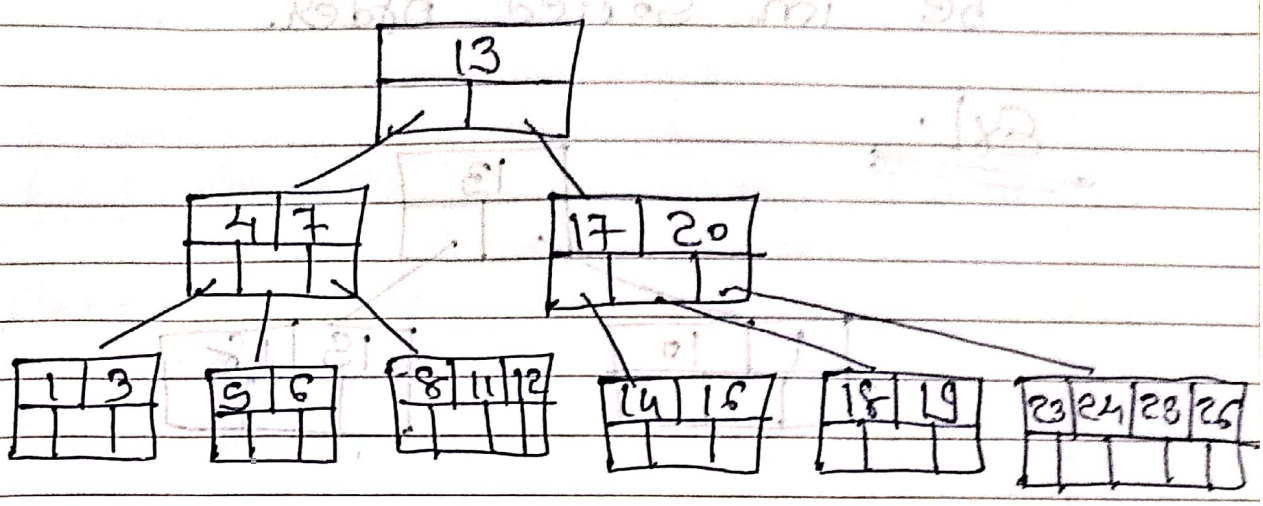
Now Delete 5.



* Searching

→ The Search Operation on B-tree is similar to a search on binary Search tree instead of choosing between a left and right child as in binary tree,

Consider the B- tree



if we want to search node 11 then

i) $11 < 13$: Search left node.
ii) $11 > 7$: Rightmost node.
iii) $11 > 8$: more in second block.
iv) Node 11 is found.

The running time for Searching is $O(\log n)$.

The 2-3 Trees

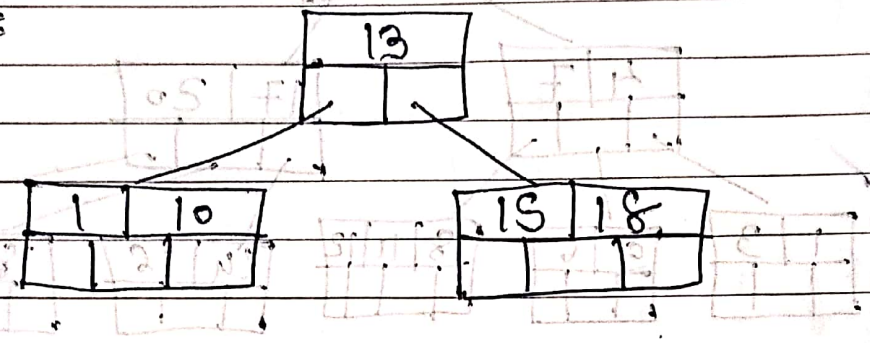
A 2-3 tree is type of B-tree in which every node has either 2 or 3 children except the leaf node.

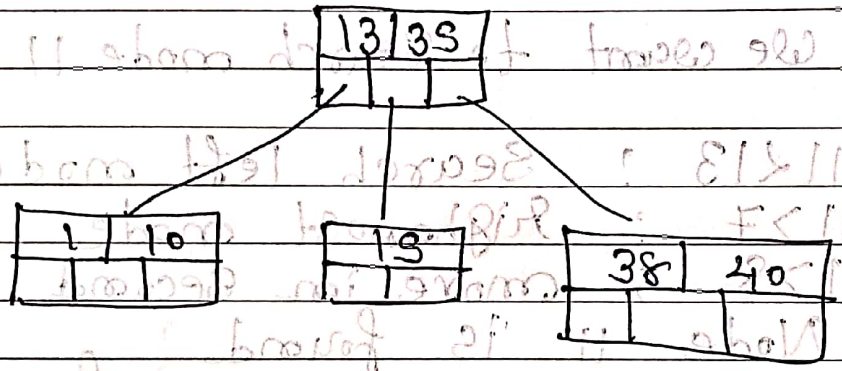
* Properties of 2-3 trees.

1) Every internal node must have either 2 or 3 children.
2) All the leaf nodes must be on the same level.

3) The data in each node must be in sorted order.

Ex1:



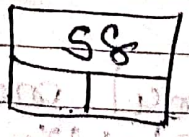


2-3 Trees

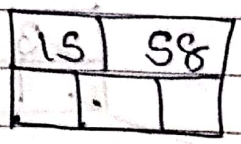
Ex1: Creation of 2-3 tree by inserting the nodes.

58, 15, 82, 36, 23, 64, 27, 35, 45

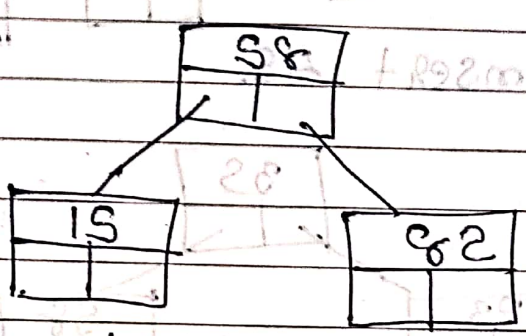
Step 1 Insert 58



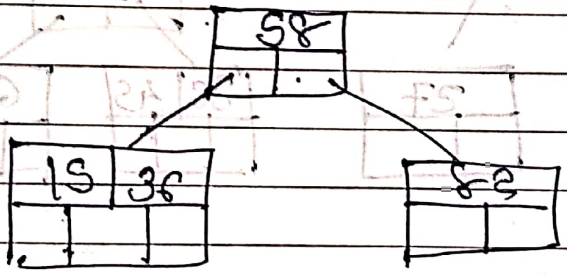
Step 2 Insert 15



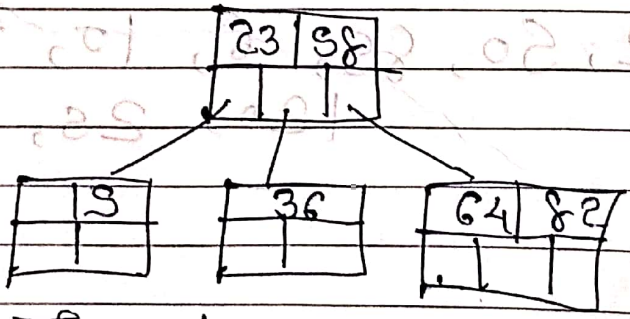
Step 3 We will arrange it as 15, 58, 62 As only 3 Children are allowed the middle element 58



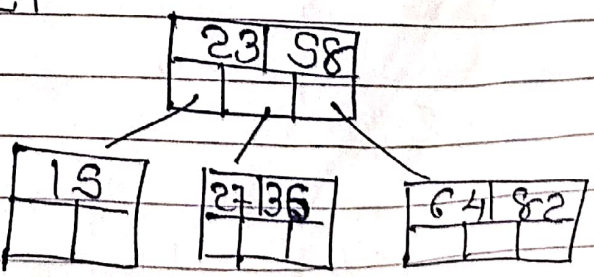
Step 4 Insert 36



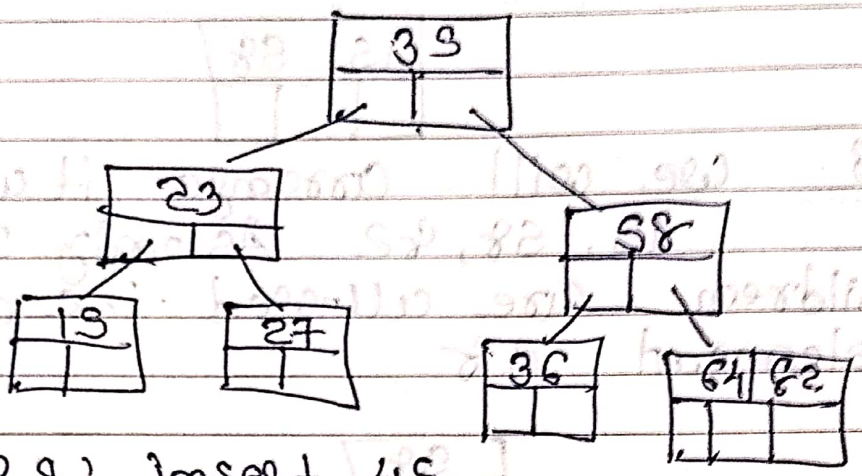
Step 5 Insert 23, 64



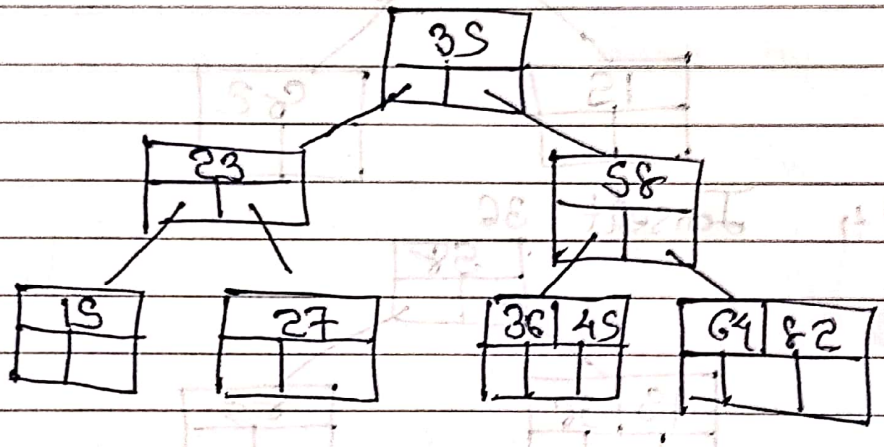
Step 7 Insert 27



Step 8: insert 35.

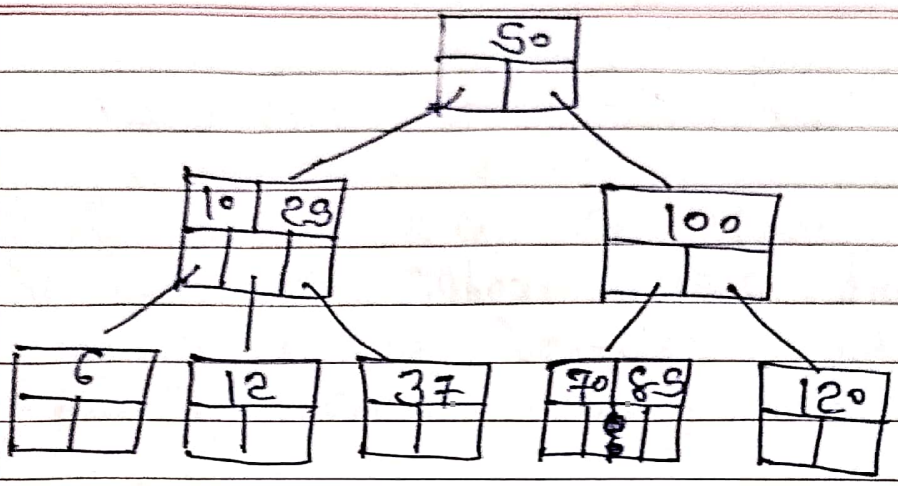


Step 9: Insert 45.



Construct 2-3 tree for following 7 Marks

12, 50, 85, 6, 10, 37, 100, 120, 25, 70

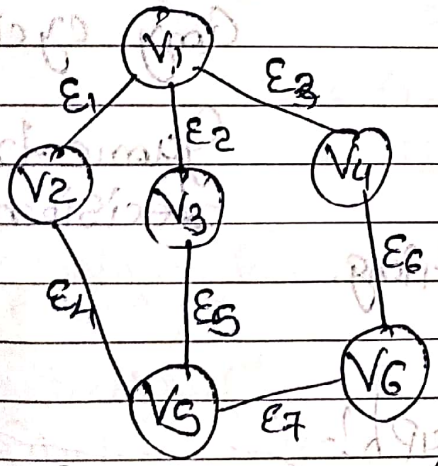


Non-linear Graph

Concept of Graph

→ A Graph is a collection of two sets V and E. Where V is a finite non-empty Set of Vertices and E is a finite non-empty Set of edges.

(1) Vertices are nothing but the nodes in the graph.
(2) Two adjacent Vertices are joined by Edges.
(3) Any Graph is denoted by $G_T = \{V, E\}$.



$G_T = \{\{V_1, V_2, V_3, V_4, V_5, V_6\}, \{E_1, E_2, E_3, E_4, E_5, E_6, E_7\}\}$

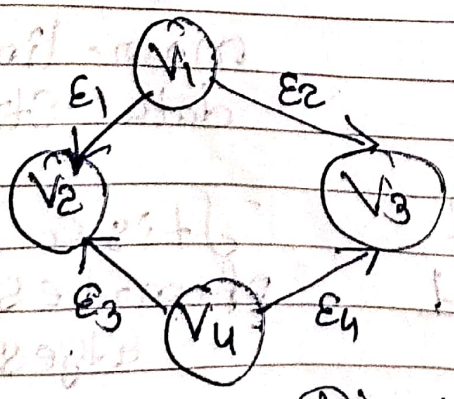
Compare Graph & Tree

Graph	Tree
(1) non-linear data Structure	non-linear data Structure
(2) it is a collection of Vertices / nodes and edges.	It is a collection of nodes and edges.

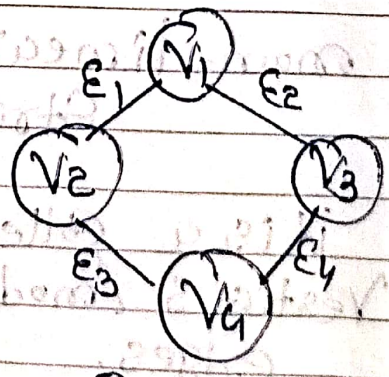
(3)	Each node can have any number of edges.	General tree consist of the node having any no. of child. binary tree every node can have at the most two child node.
(4)	no similar node called root in graph.	There is a similar node called root in trees.
(5)	Cycle can be formed	There will not be any cycle.
(6)	Allication for finding shortest Path in Networking.	game tree, decision tree,

Types of Graph:

(1) Directed Graph
(2) Undirected "



Directed



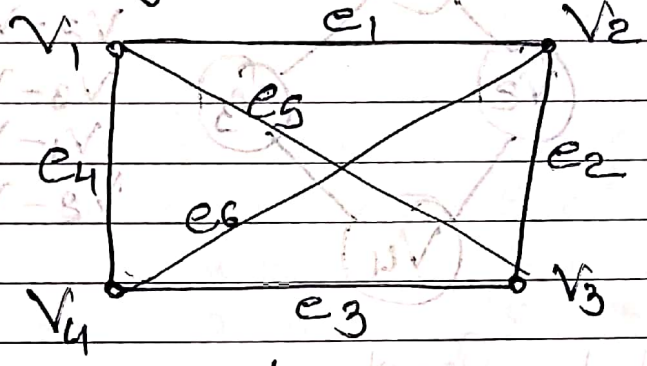
Undirected.

GTU/ Maths

Basic Terminology

(1) Complete graph

→ if an undirected graph of n vertices consists of $\frac{n(n-1)}{2}$ number of edges then that graph is called a Complete graph.

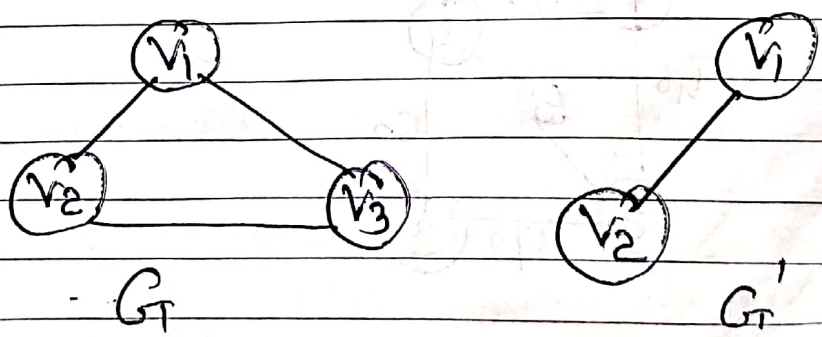


$n = 4$

$\therefore e = \frac{n(n-1)}{2} = \frac{4(3)}{2} = 6$

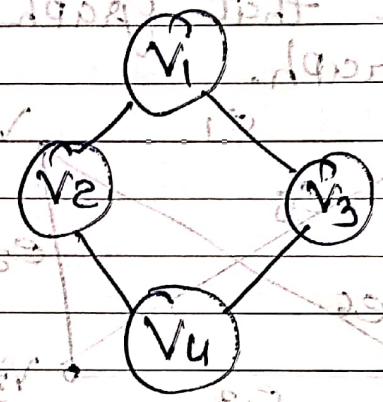
(2) Subgraph

A subgraph G_T' of graph G_T is a graph such that the set of vertices and set of edges of G' are Proper subset of the set of edges of G_T.



(3) Connected graph

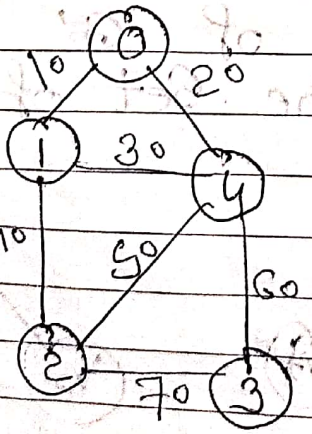
→ An undirected graph is said to be connected if for every pair of distinct vertices V_i and V_j in $V(G_T)$ there is an edge V_i to V_j in G_T



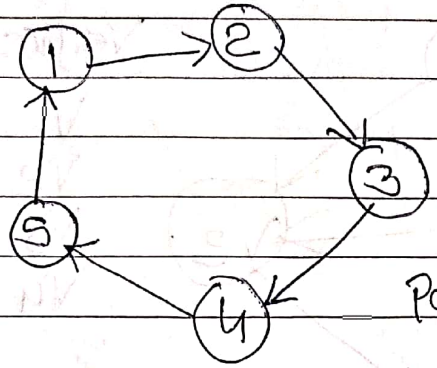
V_1-V_2
V_2-V_1
V_3-V_1
V_1-V_3
$V_2-V_1-V_3$ like this

(4) Weighted graph

→ A weighted graph is a graph which consists of weights along its edges.

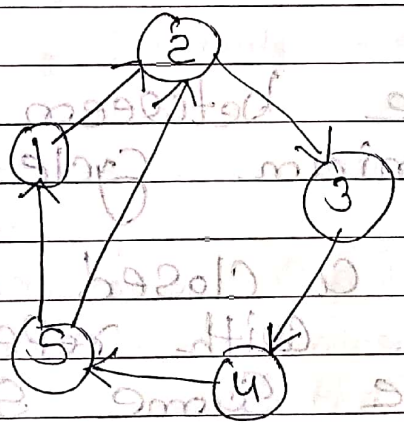


(5) Path Sequence of Vertices and there exists an edge from one Vertex to the next Vertex.



Path of G is
1-2-3-4-5

(6) Cycle A closed walk through the graph with repeated Vertices, mostly having the same Starting and ending Vertex is called a Cycle.

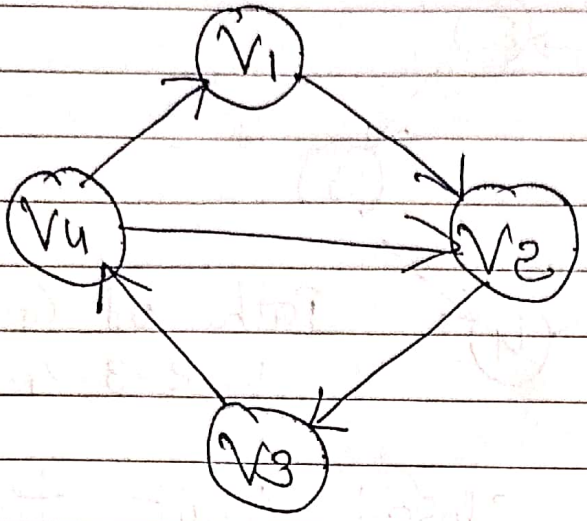


cycle 2-3-4-5-2
1-2-3-4-5-1

(7) In-degree and out-degree

→ The Indegree of Vertex is the number of edges that incident to that Vertex

Out- degree is total member of edges that are going away from the vertex.



Vertices	In-degree	Out-d
V1	1	1
V2	2	1
V3	1	1
V4	1	2

(8) Self loop- is an edge that connects the same vertex, to itself.

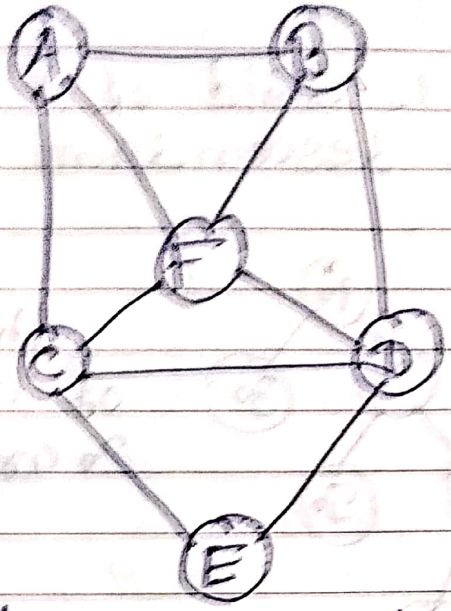
* Difference between Cycle and Hamiltonian Cycle

→ Cycle is a closed walk through the graph with repeated vertices having the same starting and ending vertex called a cycle.

→ Hamiltonian Cycle is also a cycle having a closed walk through the graph but the vertices in the cycle are not repeated at all.

→ Cycle having all vertices with same starting and

ending vertex in which every vertex appears only once.



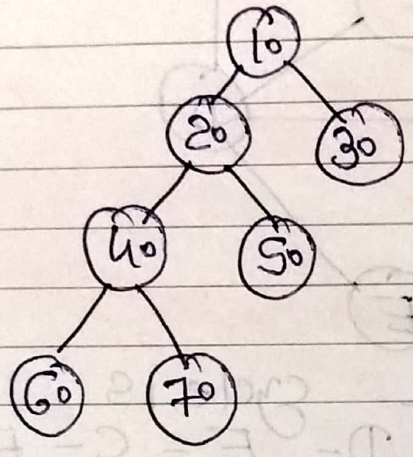
Hamiltonian cycle is
A-B-D-E-C-F-A.

* Define following terms. (GTU 7 marks)

(1) Graph-
(2) Tree -
(3) Multigraph - is a graph in which there can be more than one edge to join the pair of vertices.
(4) Weighted graph -
(5) Elementary Path - A Path is a sequence of vertices and there exist an edge from one vertex to the next vertex.
Elementary Path is a Path of a graph in which no vertex occure more than once.
(6) Complete binary tree -

(7) Descendant node :-
The node X is descendant of Y if

i) X is a child of Y or
ii) X is a descendant of a child of Y



The descendant of node 20 are, 40, 50, 60 & 70

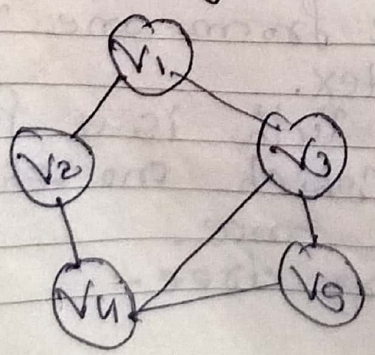
* Representation of Graphs :-

(1) Adjacency matrix

→ In this, matrix or 2 dimentional array is used to represent the graph.

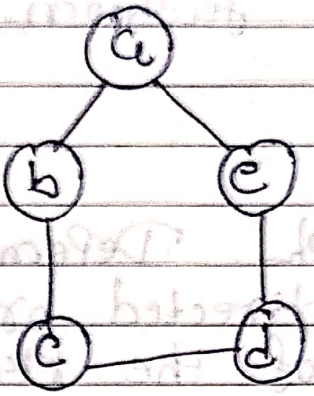
(2) Adjacency list : In this a linked list is used to represent the graph.

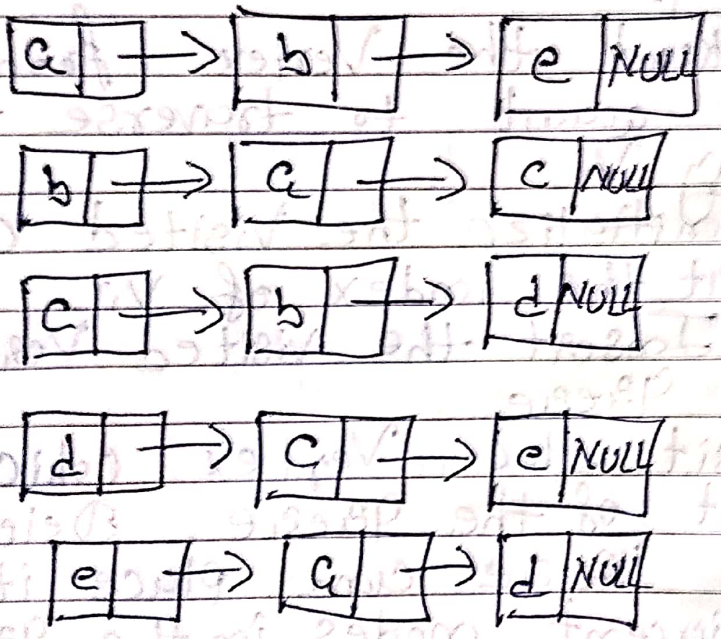
(1)



	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	0
3	1	0	0	1	1
4	0	1	1	0	1
5	0	0	1	1	0

(2)





* Display of Graph

(CO1) (U) 7 Marks.

There are two traversal technique for graph

(1) Breath first Search
(2) Depth first Search

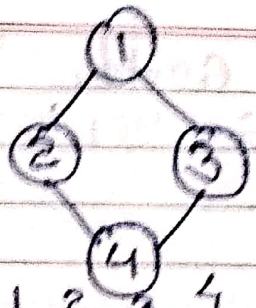
Breadth First Search (BFS) Traversal.

→ here we visit all the nodes on each level.
→ we follow the Path in breadthwise fashion.

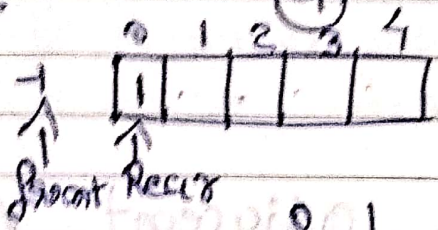
Algorithm

(1) Create a graph. Depending on the type of graph. directed or undirected set the value of the flag as either 0 or 1.
(2) Read the Vertex from which you want to traverse the graph say Vi
(3) Initialize the visited array to 1 at the index of Vi
(4) Insert the visited Vertex Vi in the queue.
(5) Visit the Vertex which is at the front of the queue. Delete it from the queue and place its adjacent nodes in the queue.
(6) Repeat the steps, till the queue is not empty.
(7) Stop.

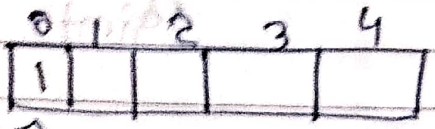
Ex 1

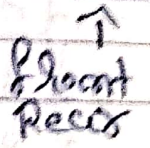


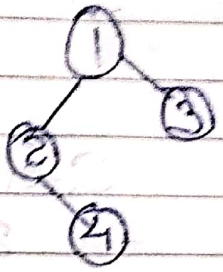
Step 1:



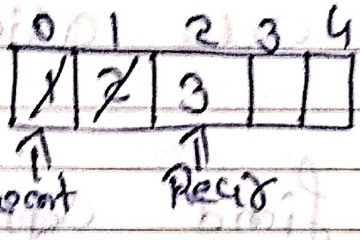
Step 2:



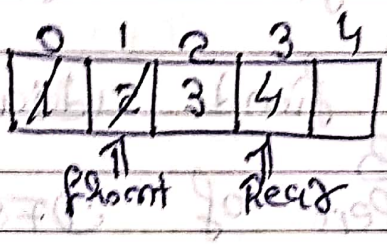




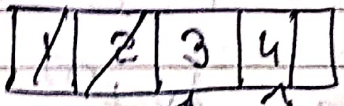
Step 3



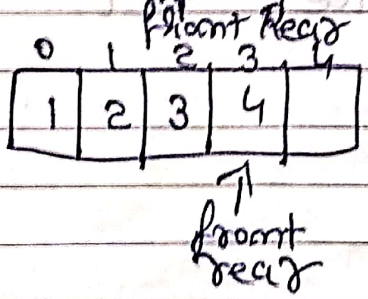
Step 4



Step 5



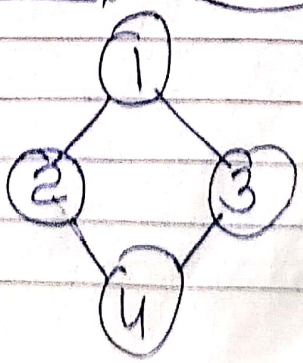
Step 6



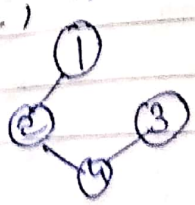
Output will be - BFS

1 2 3 4

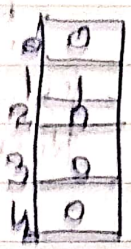
Depth First Search (DFS) Traversal



1, 2, 4, 3



Step 1: Start with Vertex 1, Print it So '1' gets Printed.



Step 2: call adjacent Vertex of '1' 2.

Step 3: find adjacent to '2' is 4.

Step 4: find adjacent to '4' is 3

So output of DFS is

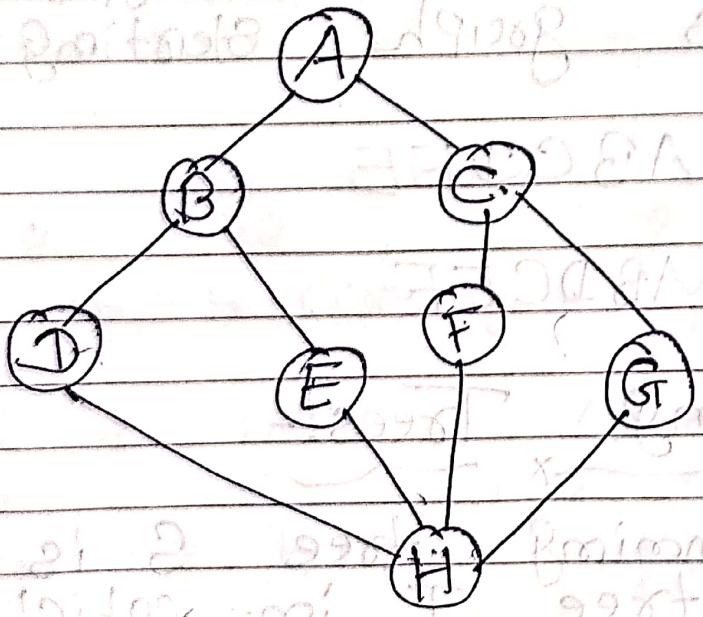
1 2 4 3

* Difference bet^n DFS & BFS.

Compare the efficiencies of BFS and DFS.

	BFS	DFS
Adjacency matrix	$O(V^2)$	$O(V^2)$
Adjacency List	$O(V+E)$	$O(V+E)$

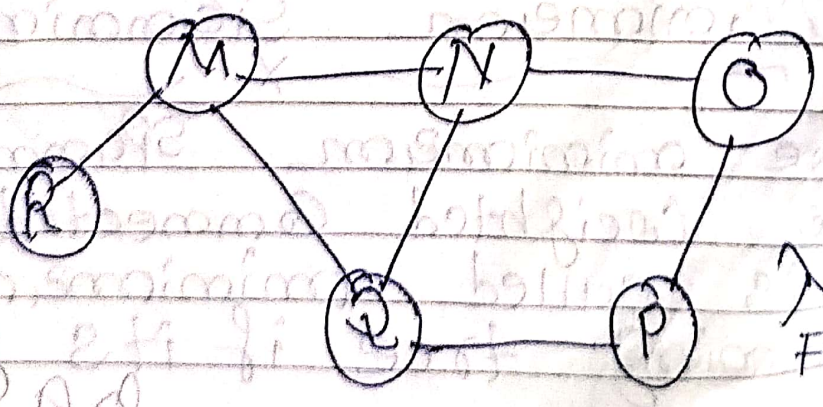
Ex 1:- Give the DFS & BFS Spanning tree for the graph given below.



DFS: The DFS Sequence is
ABDHECFG

BFS: BFS Sequence is
ABCDEFGH

Ex 1



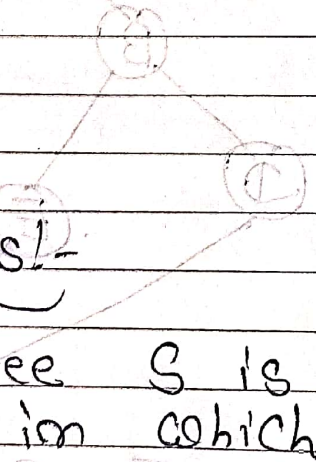
Starting Node is M
Find BFS.

GTU / 6 Marks

* Application of Graph

(1) For LAN, WAN, internetworking to find shortest Path.
(2) For telephone cabling graph theory is used.
(3) For job scheduling algorithm.

Q. Consider the graph given below find DFS and BFS traversal of this graph starting at A.



BFS - ABCDFE

DFS - ABDCEF

GTU / 6 Marks

* Spanning Trees:-

A spanning tree S is a subset of a tree T in which all the vertices of tree T are present but it may not contain all the edges.

GTU

* Minimum Spanning Tree:-

→ The minimum spanning tree of a weighted connected graph G is called minimum spanning tree if its weight is

minimum.

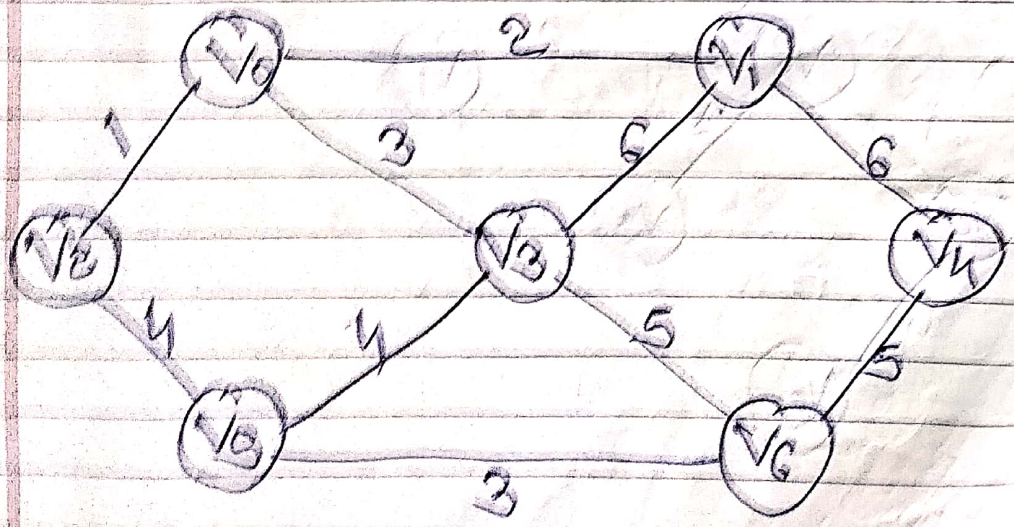
Prim's Algorithm

Step 1. Select the pair with minimum weight.

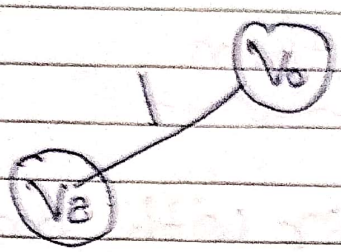
Step 2. Select the adjacent vertex and select the minimum weighted edge using this adjacent vertex. The selected adjacent vertex should not form the circuit.

Step 3 Repeat step 1 and 2 until all the vertices are getting covered.

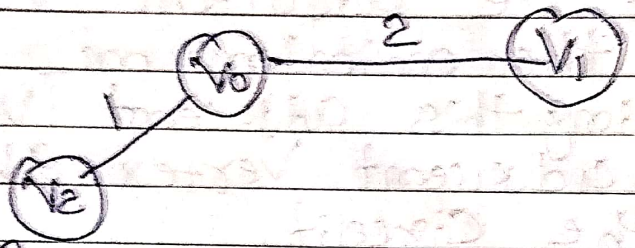
Q1. Write Prim's algorithm to find the minimum cost spanning tree of following.



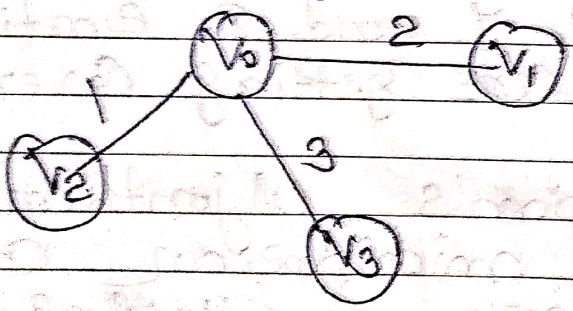
Path length = 1



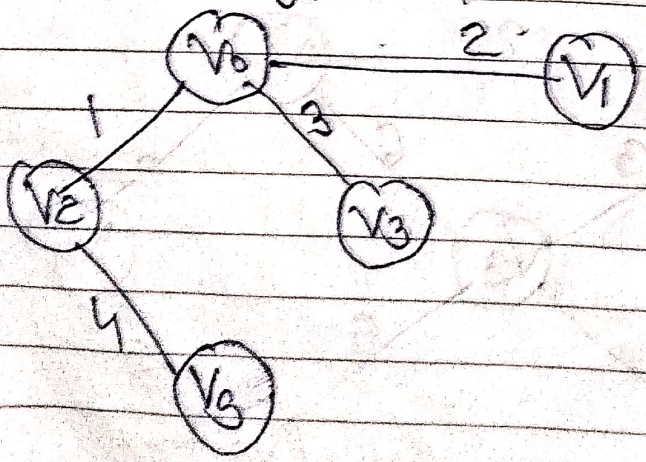
Path length = 3



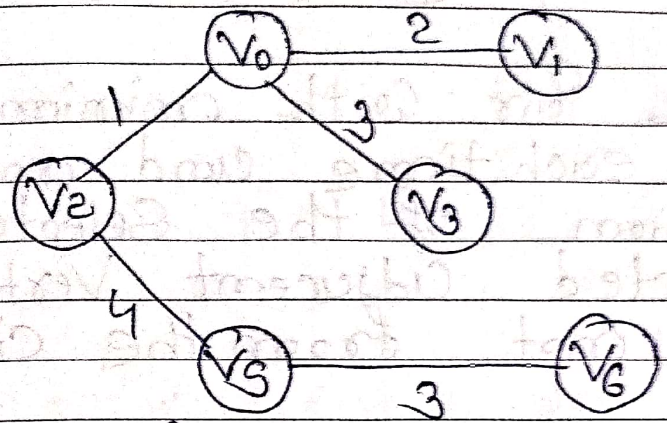
Path length = 6



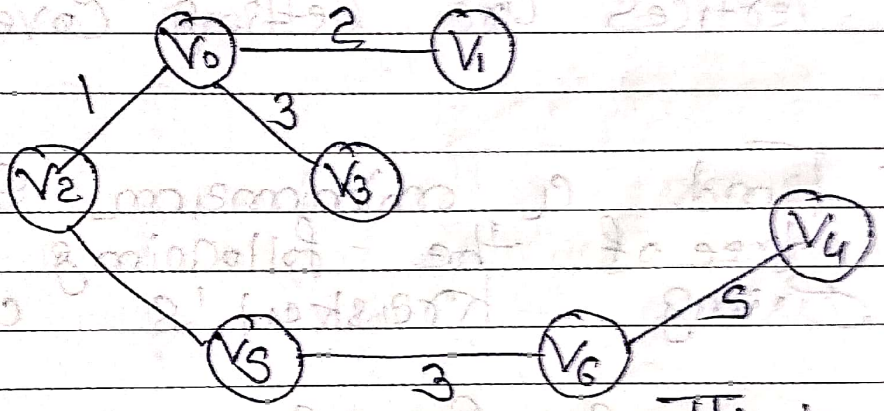
Path length = 10



Path length = 13

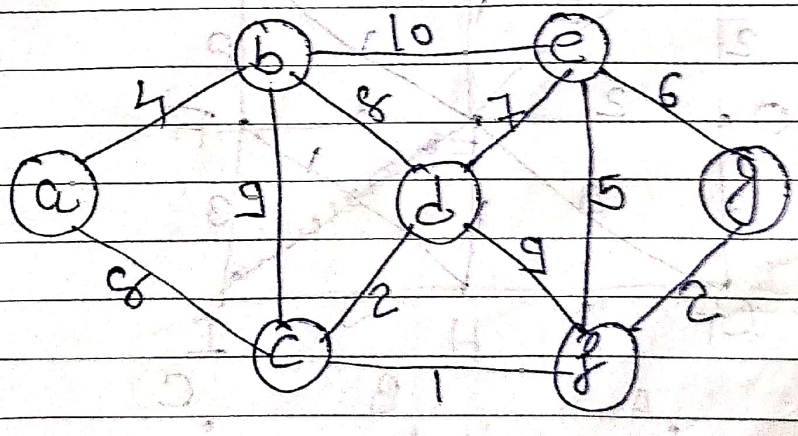


Path length = 18

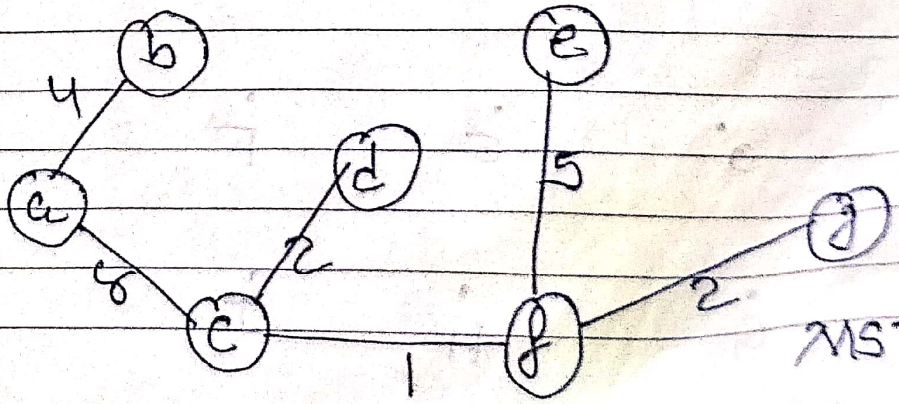


This is MST.

Ex 1)
GTU
7 Marks



Sol^n



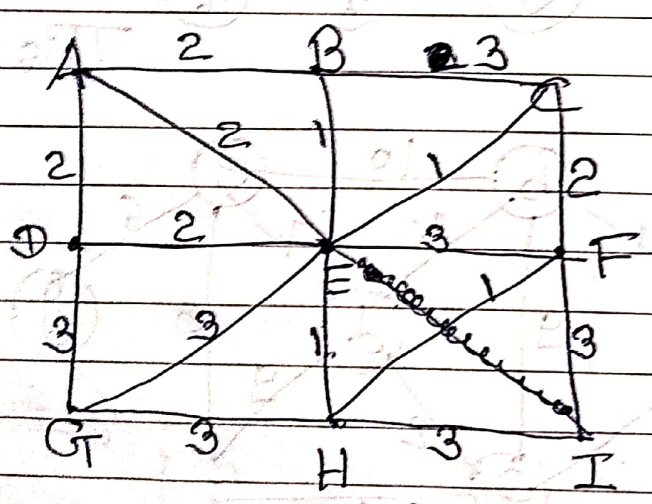
MST.

Kraskal's Algorithm

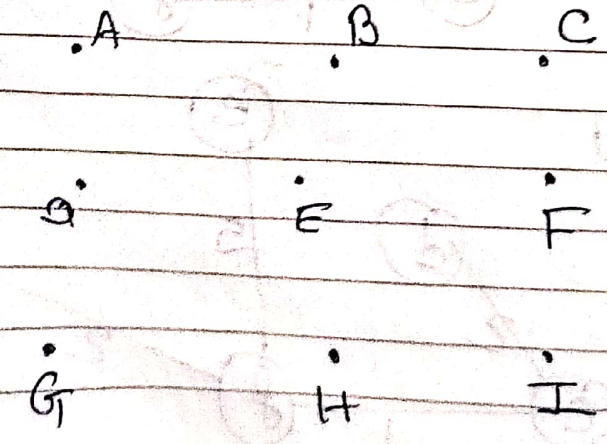
Step 1: Select the pair with minimum weight each time and make the union of the selected edge. The selected adjacent vertex should not form the circuit.

Step 2: Repeat Step 1 until all the vertices are getting covered.

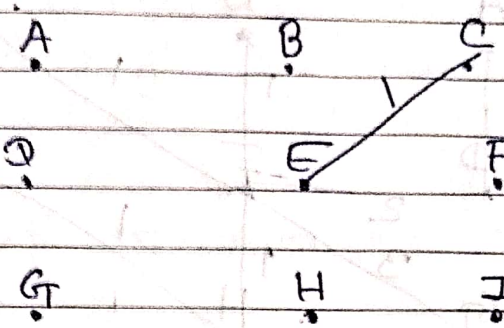
Ex 1: Find a minimum spanning tree of the following graph using Kraskal's algorithm.



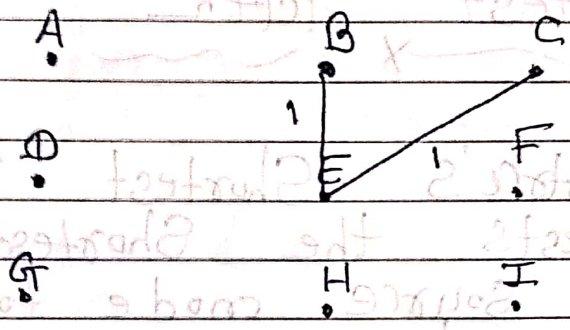
Step 1



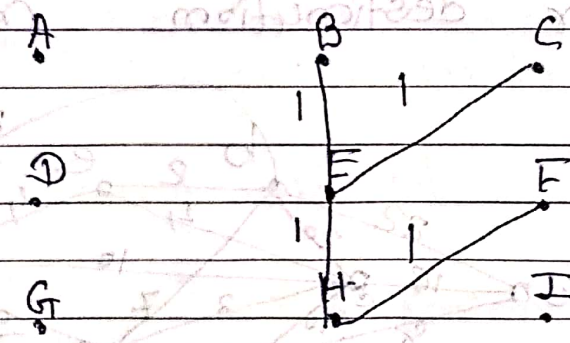
Step 2: We will select edge with minimum distance.



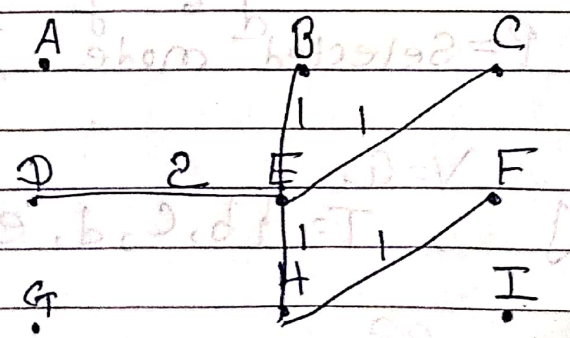
Step 3: then select next minimum distance



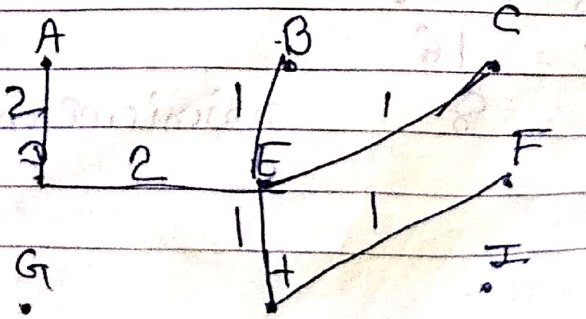
Step 4:



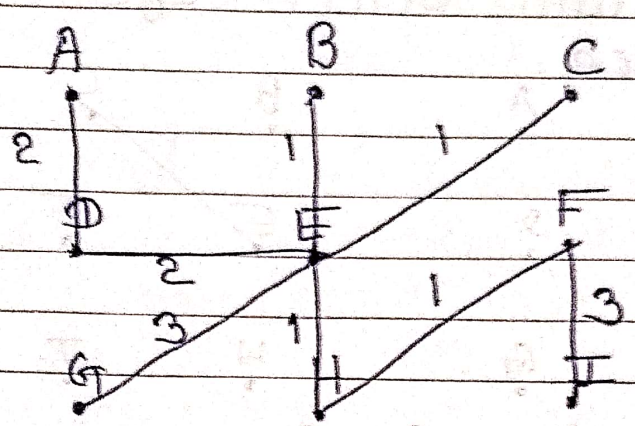
Step 5:



Step 6



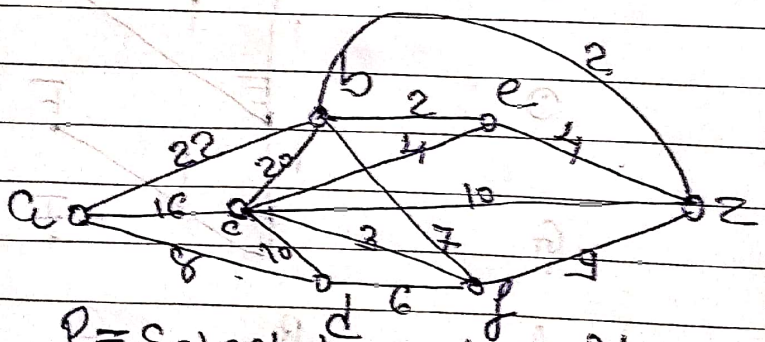
Step 7.



⇒ Difference betⁿ Prims' & Kruskal's Shortest Path

⇒ Dijkstra's Shortest Path algorithm suggests the Shortest Path from some source node to the some other destination node.

Ex:



P = Selected node N = remaining node

Step 1:

V = a

P = {a} , T = {b, c, d, e, f, z}

a → b = 22

a → c = 16

a → d = 8 minimum distance

Step 2:

P = {a, d}

Node d → f = 6 ← minimum dis.
      d → c = 10
      d → b = 20

f will be selected

Step 3

P = {a, d, f}

Node f → z = 9
      f → b = 7
      f → c = 3 ← minimum dis

Step 4

P = {a, d, f, c}

c → z = 10
c → b = 20
c → e = 4 ← minimum dis

Step 5

P = {a, d, f, c, e}

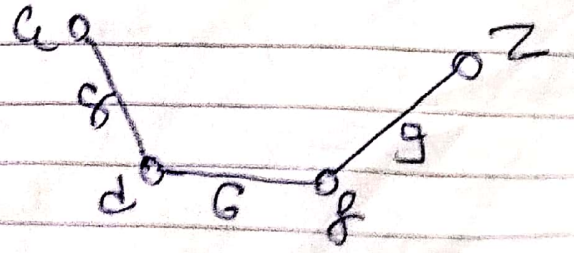
e → b = 2 ← minimum dis
e → z = 4

Step 6

P = {a, d, f, c, e, b}

The target node is z

Shortest path is



File Structure

Concepts of

⇒ **Field :-** Field is a Piece of data which is associated with the entire record. It is refered as Column.

⇒ **File :-** A file is a Collection of records.

⇒ **Record :-** Record is nothing but the Source of information. it is stored in rows in the file structure.

Roll No	Name	Marks
1	John	66
2	Mathew	70
3	Steve	60

8 records. it is written in file. with associated field such as Roll No, Name and Marks.

Operations on File

(1) Declaration of file Pointer :

Syntax: FILE *filePointer;

Ex: FILE *fp;

- FILE is a structure which is defined in "stdio.h"
- Each file we open will have its own FILE structure. The structure contains information like usage of file, its size, memory location and so on.
- fp is a Pointer Variable which contains the address of the structure FILE.

2) * Creating a file / Opening a file:

Syntax: filePointer = fopen("filename", mode)
- mode - read - "r"
- write - "w"
- append - "a"

* ⇒ Using "r" open the file in read mode.
⇒ if file does not exist create a new file
⇒ You cannot write to any file in read mode.

* ⇒ Using "w" open file in write mode.

* ⇒ "a" opens file in append mode. In append mode allows you to append new contents at the end of the file.

fp = fopen("student.dat", "

(3) Closing the file

fclose (filePointer);

fclose(fp);

(4) Setting the Pointer to start of file:

rewind (filePointer);

(5) Writing a character to file:

fputc(character, filePointer);

"mode")

Char ch = 'a';

To write character to the file,

fputc (ch, fp);

(6) Reading a character from file:

fgetc (fp);

Ex:- Char ch;

ch = fgetc(fp);

(7) Write string to file

fputs (String, filePointer);

Char s[] = "abcd";

fputs(s, fp);

w");

(8) Reading String from file:

fgetc(Stringaddress, length, filePointer);

Char S[8];
fgets(S, 79, fp);

We will read the 78, (79-1) char of string from file into S.

(9) Writing of characters, strings, integers, float to file:

fprintf(filePointer, "format string", list of variables);

(10) Reading of variables from file:

fscanf(filepointer, "format String", list of addresses of variables);

Eg:-
fscanf(fp, "%d %s", &rno, &name);
Printf("Read data is: %d %s", rno, name);

(11) Writing Contents to file through structure:

fwrite(Pointer to Struct, size of Struct, no of data item, filePointer)

seek: specify from where to take the data.
Approach → system call, seek, thus repositions the file pointer to a specific place in the file.

(12) Reading Contents of file through Structure

fread (Pointer to Struct, Size of struct, no. of data items, filePointer);

Ex fread (&s, sizeof(s), 1, fp);

(13) Moving the Pointer from one record to another.

fseek (filePointer, offset, Whence);

where
offset is difference in bytes bet^n offset & Whence Position.

1) SEEK_SET Move the Pointer to beginning of the file.
2) SEEK_CUR to its current Position in file.
3) SEEK_END Move to end of file.

Ex: fseek(fp, -sizeof(s), SEEK_CUR);

⇒ Get Attributes, set Attributes, Rename, Locks

Types of file

(1) Sequential (3) Random
(2) Indexed

(1) Sequential Files

⇒ Ex. A Cassette where the Song are Stored Sequentially. Whenever we want

to play any song from it, we read it sequentially.

→ So Reading data to file or writing data from file it takes a lot of time as the data is not sorted. The data is stored on FCFS basis.

→ but here suppose we search an element which is stored at last position in file require to search entire file: time required very large.

* Advantages

→ It is very simple to implement.

* Disadvantages

→ Time required to retrive any record is more.

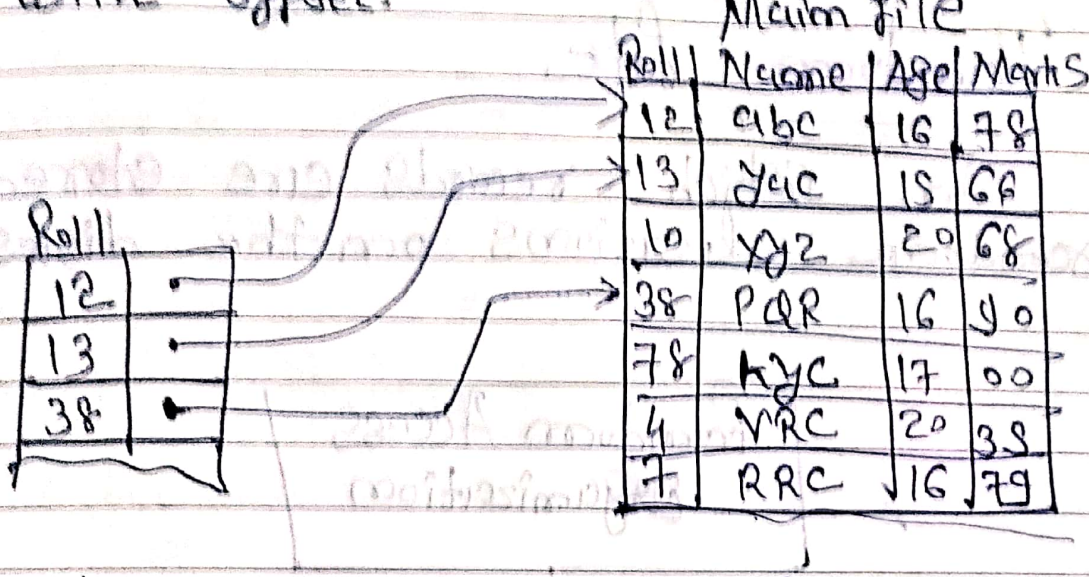
→ Efficiency is less.

(2) Indexed Sequential File

→ In an index sequential format, we maintain two files - 1) Normal sequential file
2) An sorted index file.

→ Store data in sequential file and in the index file we have the id array Primary key along

with offset.



- Whenever we are referring to any record, first index file will be Search.
- From that Search the offset is retrived and the required record can be seek from Sequential file.

* Advantages

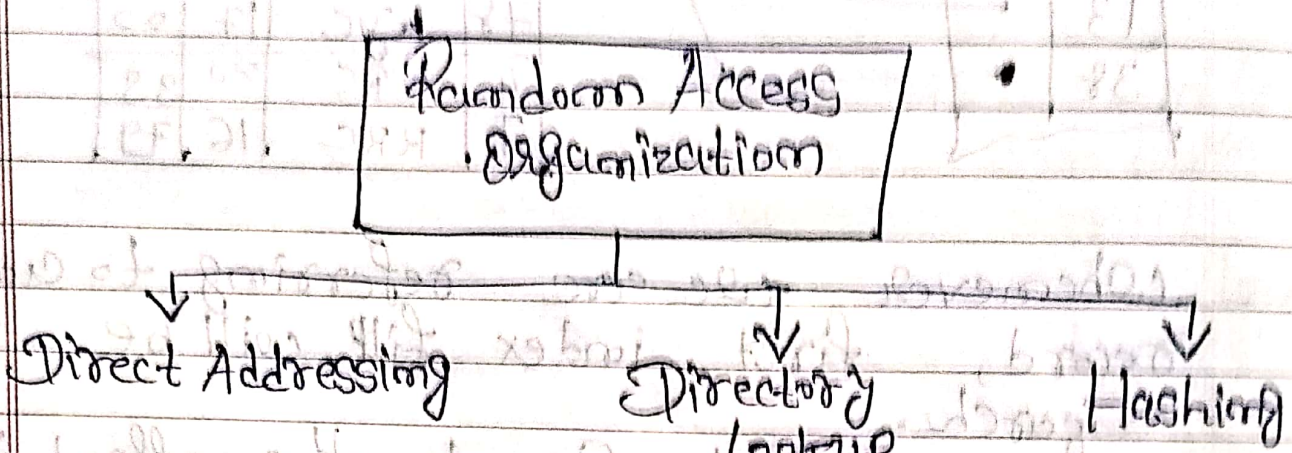
(1) More efficient than Sequential file.
(2) Time required is relatively less than Sequential file.

* Disadvantage

(1) Maintaining two files relatively increases the overhead.

* Random File

⇒ In which records are stored at random locations on the disks.



(1) Direct Addressing

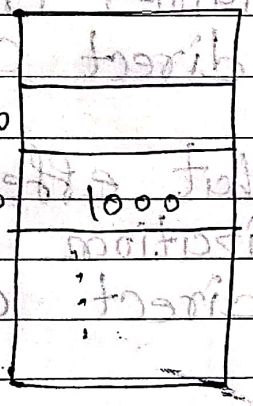
⇒ in direct addressing two types of records are handled:
- Fixed length record
- Variable " "

⇒ For storing the fixed length records the disk space is divided into nodes. These nodes are large enough to hold individual record.

⇒ if we consider that the records are stored on external storage devices then deletion and searching of record require one disk access.

if we want to update a record item it require two disk access, one for reading the record and one for writing the updated data back to the disk.

*) For Storing Variable length records on disk, the address of each individual record is stored in the file at specific index.
Variable length record make storage management more complex.

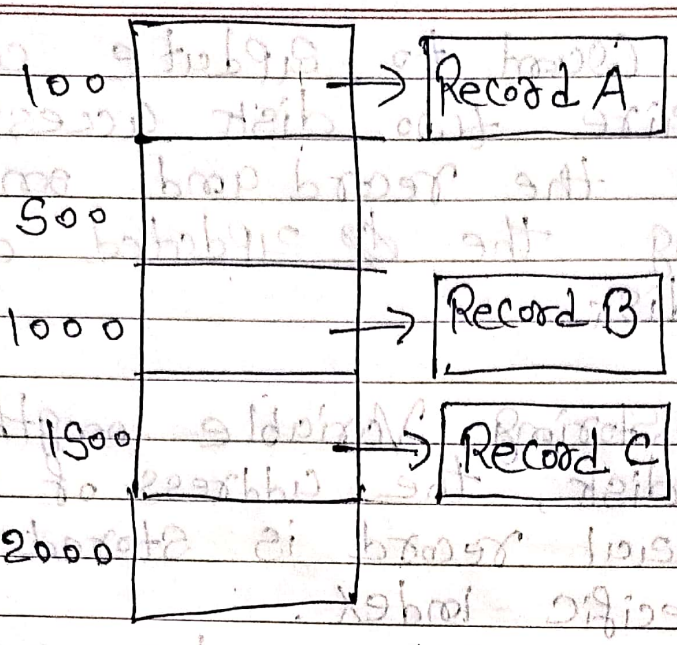


Storing fixed length record.

(2) Directory Lookup

→ In this the index for the pointers to the records is maintained.

→ For retrieving the desired record first of all the index for the record address is searched and then using this record address the actual record is accessed.



→ The drawback of this method is that it requires more disk access than direct address method.

→ Advantage is that effective disk space utilization in it as compared to direct addressing method.

(3) Hashing

→ here hash key is obtained using some suitable hash function and record is placed in hash table with the help of hash key.

→ The record can be quickly searched with the help of hash fun being used.

→ For creation of hash table the available file space is divided into backets and slots.

Multi-key File Organization

→ For understanding this consider the Roll_NO record.

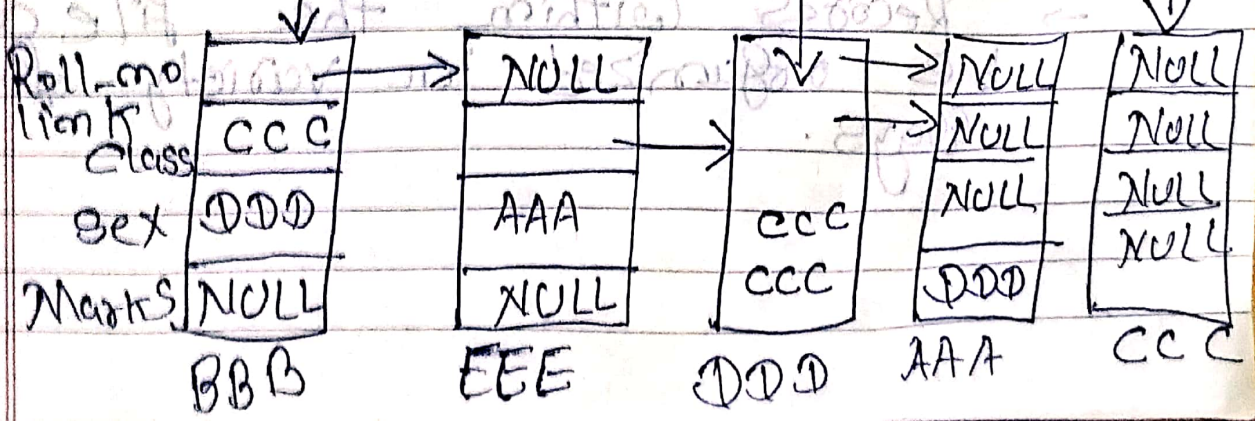
→ In Roll_no record structure, there are 3 field - value, length, and pointer to the first record.

→ The value field indicate the upper bound value for Roll_no if roll number of student is 487 it lies between 0 to 500. Upper bound is 500. if Roll-no is 689 then it associated with 700.

→ length 2 means there are 2 record in between 0 to 500.

Value	500	700	900
length	2	2	1
Pointer to first record			

(Multi-list structure)



Advantages

(1) it Provides Satisfactory Solution to Simple & range queries.

" Select * from dept_table where Salary > 10000 "

(2) Quick access to every individual record is possible.

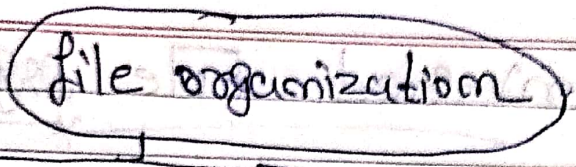
Disadvantage

(1) Some amount of memory gets Consumed in maintaining the link or address field.

Access Methods

→ The method by which records can be retrieved or updated from the file is called access method.

→ Records within the file can be organized in variety of ways.



Sequential organization

Random access or direct access

Linked organization

Inverted organization

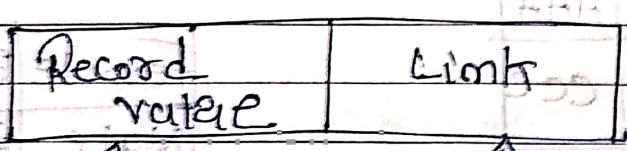
Cellular partitions.

(1) Linked Organization

→ The logical sequence of the record is different than the physical sequence.

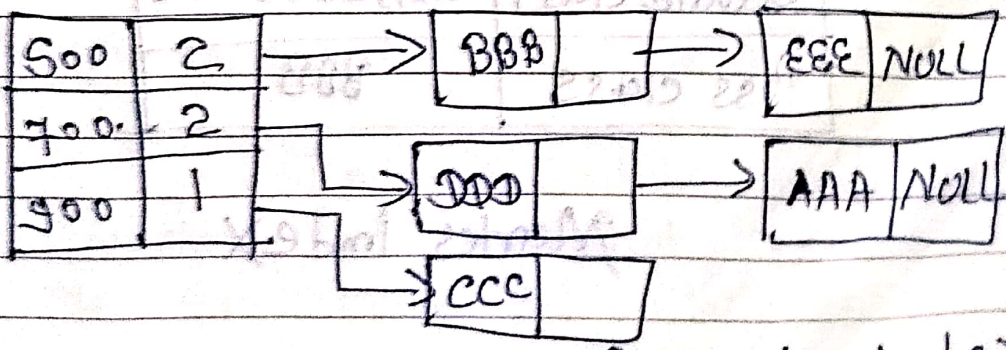
→ In linked organization we can access next logical record by following the link-value pair.
it denotes each individual record.

→ The structure of record is



Value of Current record

Using this address can be locate the next record.



Roll_No index

(8) Inverted File organization

→ it is similar to multi-lists

→ In Multi-list records with the same key value are linked together along with link information being kept in individual record.

→ In inverted file the link information is kept in index itself.

437	BBB
488	EEE
689	DDD
695	AAA
778	CCC

Roll_No index

Fifth	BBB, CCC
Tenth	EEE, DDD, AAA

Class index

Female	BBB, DDD, CCC
Male	EEE, AAA

Sex index

First class	EEE
Second class	AAA, DDD, CCC
Pass class	BBB

Marks index

→ The above index structure is dense index structure Dense Indexing

→ The dense index is a kind of indexing in which record appears for every search key value in the file.

* Advantage

— Space saving files.

* Disadvantages

— Insertion & deletion of records is complex.
— Index maintainance is complicated

(3) Cellular Partitions

→ For reducing the searching time during file operations, the storage media may be divided into cells.

i) Entire disk pack can be a cell
ii) A cylinder can be a cell

→ A list of records can occupy either entire disk pack or it may lie on particular cylinder.

→ if all the record lie on the same cylinder then without moving the read/write head the record can be accessed.

→ if the cell is nothing but entire disk pack then the disk is Partitioned into different Partitions. are called Cellular Partitions.

* Advantages

(1) Various read operations can be Performed Parallely in order to reduce the search time.

(2) Faster execution of any query.

* Disadvantage

1) if multiple records lie in the same cell then reading a single cell becomes a time consuming Process.

Page
Date

22/8/2022
A3 batch (1-2 slot)
(60, 50, 53)

29/8/2022
A3 batch (1-2 slot)
(57, 50, 53, 43[illegible], 60)

* if most of the elements of the matrix have 0 value, then it is called Sparse matrix.

* Why to use Sparse matrix ?
(1) Storage (2) Computing time

(1) Array Representation

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Rightarrow$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
value	3	4	5	7	2	6

(2) Linklist Representation