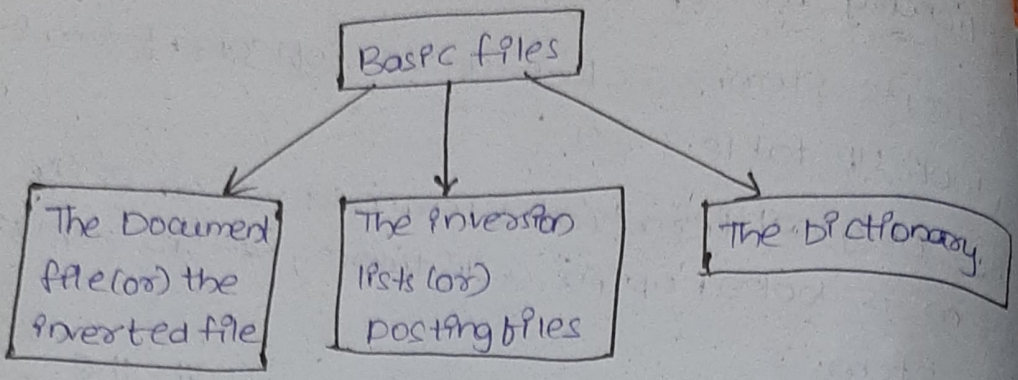
Inverted File Structures

- Inverted file structures are used because they provide optimal performance in searching large databases.
- The most common datastructure used in both database management & IRS is the inverted file structure.
- Inverted file structures composed of 3 basic files.
- It is used to minimize the secondary storage memory.



Inverted file (or) Document File:

→ The name "inverted file" comes from its underlying methodology of storing the inversion of documents: the inversion of document perspective: for each word a list of document in which the word is formed and stored i.e. (the inversion list for that word).

The Inversion List:

The inversion list sometimes called as posting files. In this each document in the system given a "unique numerical identifier." The identifier is stored in the invertion list.

The Dictionary:

- The dictionary is typically a stored list of all unique words (processing tokens in the system).
- It provide a way to locate the inversion list for a particular word is "via" the dictionary.
- The dictionary can also store other information used in query optimization such as length of invertion list.

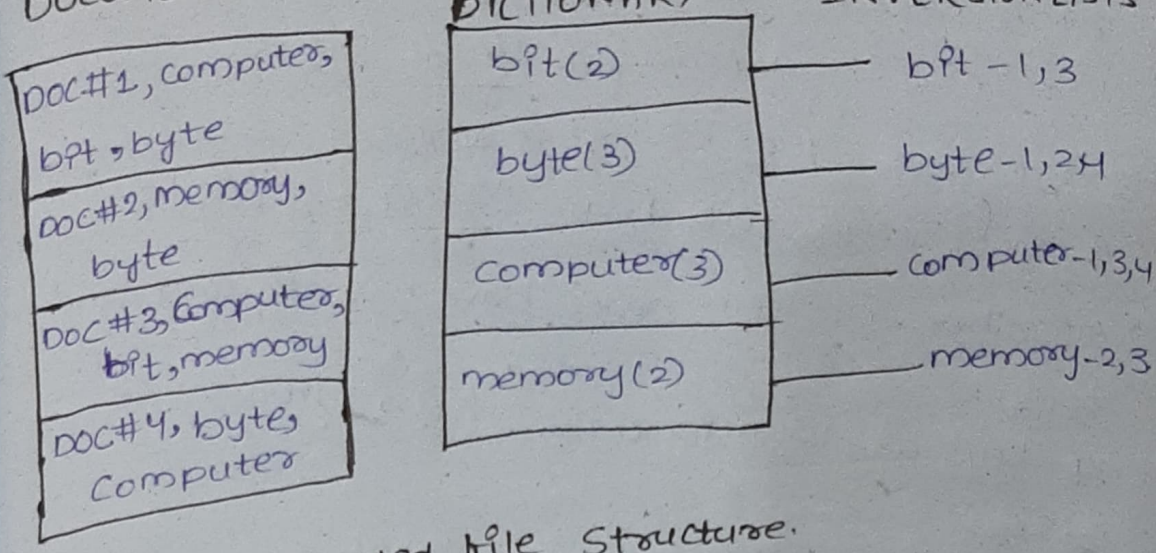


fig:- Inverted file structure.

from the fig:-

* The Documents Contains information of words that is how many words are there in particular documents.
* The Dictionary:- It contains how many no:of times the particular word is appear in the document is written inside the paranthesis.
* The Inversion list contains in which documents the particular word is appearing.

Finally,

The inverted file structure uses some salient features i.e.

1) It increases precision & Recall.
2) It also uses browse capabilities such as Ranking & zoning.
3) It uses the NLP (Natural Language processing).
4) It is mainly used to store the concepts & Relationships.

5) Rather than using a dictionary to point the inversion list B-Trees can be used. In B-trees the inversion list may be appear at "leaf level".

6) A B-Tree of order 'm' is defined as:

* A root node with b/w "2 and 2m keys".
* All other internal nodes have b/w 'm' and '2m' keys.
* All keys are kept in order from smaller to larger.
* All leafs are at the same level.

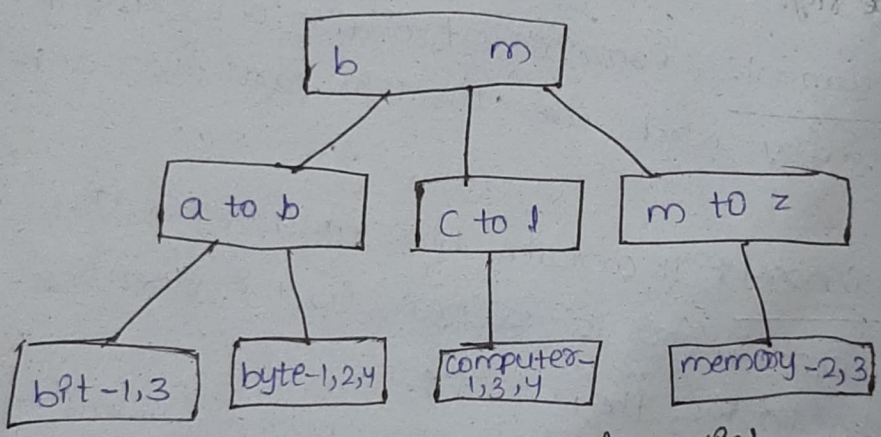
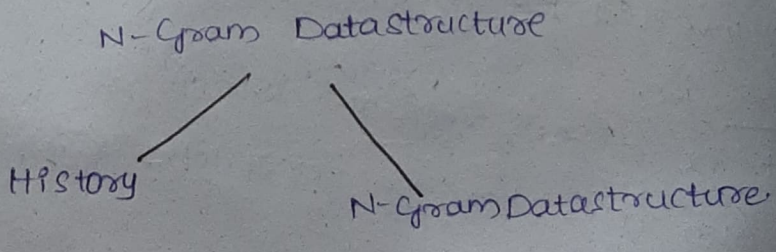


fig:- B-Tree inversion list

N-Gram Data structure:-

→ N-Gram Data Structure.



History:-

The first use of n-grams dates to world war II when it was used by cryptographers. Fletcher Pratt states that "with the backing of bigram & trigram. use of bigrams was described by Adamson as a method for conflating terms (Adamson-74). It does not follow the normal

definition of stemming because what is produced by creating n-grams are word fragments versus semantically meaningful word stems. It is this characteristic of mapping longer words into shorter n-gram fragments that seems more appropriately classified as a ds process than a stemming process.

Another major use of n-grams (in particular trigrams) is in spelling error detection and correction (Angell-83, McIlroy-82, Morris-75, Peterson-80, Thorelli-62, Wang-77, and Zamora-81). Most approaches look at the statistics on probability of occurrence of n-grams (trigrams in most approaches) in the English vocabulary to indicate any word that contains non-existent to seldom used n-grams as a potential erroneous word. Damerau specified four categories of spelling errors (Damerau-64). Using the classification scheme, Zamora showed trigram analyses provided a viable ds for identifying misspellings & transposed characters. This impacts information systems as a possible basis for identifying potential I/P errors for correction as a procedure within the normalization process. Frequency of occurrence of n-gram patterns also can be used for identifying the language of an item (Damashek-95, Cohen-95).

Error category	Example
Single character Insertion	Computer
Single character Deletion	Compter
Single character Substitution	Compiter
Transposition of two adjacent characters	Comptuer

Fig: Categories of spelling Errors.

N-Gram Data structure

* The first use of N-Gram is dates to World War-II by cryptographers.
* N-Gram can be viewed as a special technique for conflation (stemming) & has a unique datastructure in information systems.
* A variant of a searchable datastructure is the N-gram data structure that breaks the processing tokens in to smaller string units & use the token fragment for the search.
* N-grams are a fixed length consecutive series of "n" characters.
* The major use of the n-gram is in spelling error detection & correction.
* In this concept, the searchable data structure is transformed into overlapping n-grams, which are then used to create the searchable db.
* In N-gram datastructure it divide the text into bigrams, trigrams, pentagrams.
* Some systems allows interword symbols to the part of the n-gram. The symbol '#' is used to represent the interword symbol. The '#' is representing any one of set of symbols. for ex:- '#' may be represent –, ;, :, period, etc..
* In n-gram ds each of the n-grams created a separate processing tokens & they are searchable.
* It is possible that the same n-gram can be created multiple times from a single word for ex:- bigrams, trigrams & pentagrams for 'Sea colony'.

Sea → Se ea
Colony → co ol lo on ny } Bigrams

Sea → Sea
Colony → Col olo lon ony } Trigrams

Se ea co ol lo on ny — Bigrams (no interword symbols)

Sea col olo lon ony — Trigrams (no interword symbols)

#Se Sea ea# #co col olo lon ony ny# — Trigrams (with interword symbols #)

#Sea# #Colo Colon olony lony# — Pentagrams (with interword symbols #)

fig:- Bigrams, Trigrams, Pentagrams for "Sea Colony"

* There is no semantic meaning in a particular n-gram since it is a fragment processing token & may not represent a concept. Therefore n-grams are poor representation of concepts & their relationships.

Advantage:

The advantage of n-gram is that they place a finite limit on the no. of searchable tokens.

$$maxseq_n = (\lambda)^n$$

Where maximum no. of unique n-grams that can be generated, maxseq → calculated as a function

of 'n'.

n → which is the length of n-gram.

λ → which is the no. of processable symbols from the alphabet

Disadvantage

The disadvantage of n-gram is increased size of inversion list that stores the linkage ds.

PAT Data structure

The name PAT is the short-form for PATRICIA trees PATRICIA stands for practical Algorithm to Retrieve Information coded in Alpha numeric.

PAT DS's were described by Frakes-92, gone Gonnet-83, Knuth-73 and moorison-68 has patricia trees.

* The original concept of PAT tree ds's were described as PATRICIA trees used for searching text & images & the applications are in genetic db.

* The PAT represent the information in two ways i.e. PAT trees & PAT Arrays.

* The PAT trees & PAT Arrays are addressing the different view of continuous text. The input stream is transformed into searchable ds consisting of substrings.

Substrings:-

A substring can start at any point in the text and can uniquely indexed by its location & length.

If all strings are to the end of the i/p, only the starting location is needed i.e, the length is different from the location.

* It is possible to have a substring go beyond the length of the i/p string by adding additional null characters. These substrings are called Si string. (Semiinfinite string).

* A PAT tree is an unbalanced, binary digited tree defined by the Si strings.

for ex:- Text: Economics for warsaw is complex.

Sistring 1 Economics for warsaw is complex.

Sistring 2 conomics for warsaw is complex.

Sistring 5 onomics for warsaw is complex.

Sistring 10 for warsaw is complex.

Sistring 20 w is complex.

Sistring 30 ex.

fig:- examples of Sistrings.

In PAT tree key values are stored at the leaf nodes. for the text i/p of size 'n' there are n no. of leaf nodes & n-1 atmost "higher level nodes."

Example of the Sistrings used in generating a PAT.

INPUT: 1 0 0 1 1 0 0 0 1 1 0 1

Sistring 1 — 1 0 0 1 .....
Sistring 2 — 0 0 1 1 0 0 .... (→ 0/1)
Sistring 3 — 0 1 1 0 0 ....
Sistring 4 — 1 1 ....
Sistring 5 — 1 0 0 0 ....
Sistring 6 — 0 0 0 ....
Sistring 7 — 0 0 1 1 0 1 -
Sistring 8 — 0 1 1 0 1 -

fig:- Sistrings for input "1001100011o1"

* n=8 (sistrings)
* n-1=7 (trees)

→ In patricia tree:-

A '0' bit (zero bit) will cause a branch to left sub-tree.

A '1'-bit (one-bit) will cause a branch to right subtree

Hence the patrica trees are binary digital trees.

Note:-

ex:- To reach the external node for the query 001100
we first inspect bit 1 (if it is '0', we goto left).
then bit-2 (if it is '0', we goto left),
then bit-3 (if it is '1' we goto right)
then bit-4 (if it is '1' we goto right)
then bit-5 (if it is zero, we goto left)
once we reach our desired node we have to make one final comparison with one of sistring.

→ The input stream is used in defining the PAT tree i.e it define the path from the root node to existance of sistrings.

Binary Tree

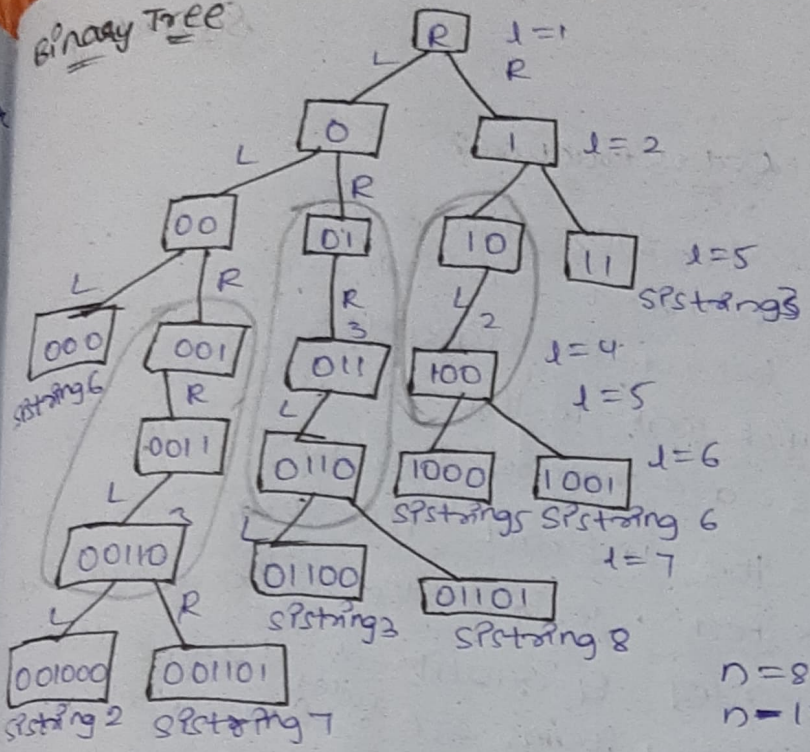
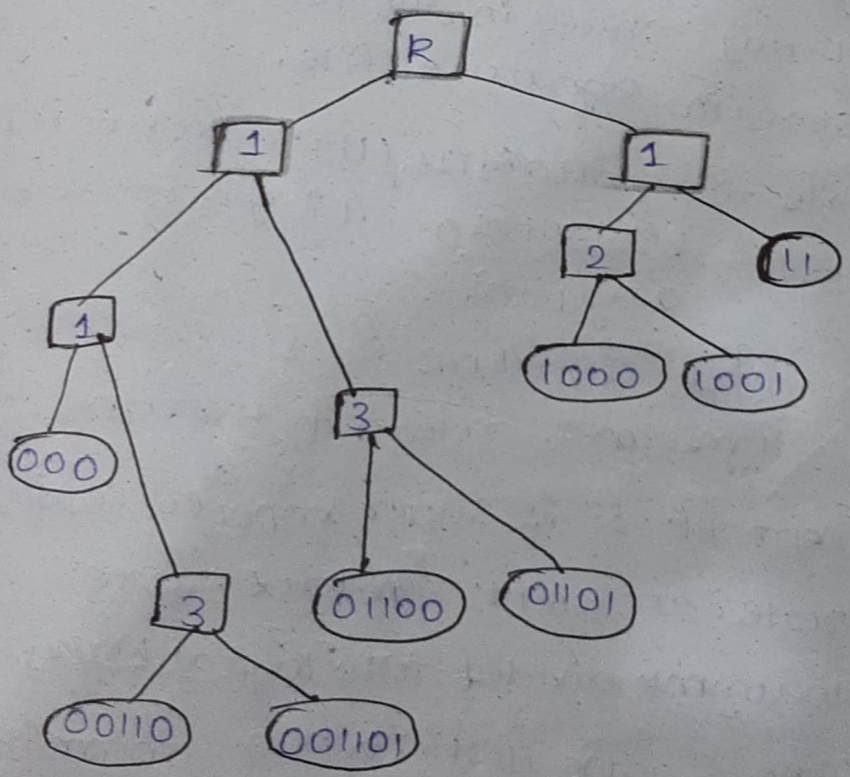


Fig:-
PAT binary tree for
I/p "100110001101"

n=8
n-1=7

Fig:-
PAT tree skipping bits for 100110001101



Signature file structure:-

Signature file is a technique applied for the "Document Retrieval."

The main idea behind signature files is to create a quick link to the documents which match the queries passed by the user.

This is done by creating a signature for each document.

A signature is created as an "abstraction" of a document.

A signature is a compressed version of a database.

⇒ All signatures that represent the documents are kept in a file called "SIGNATURE FILES".

⇒ The goal of a signature file is to provide a fast test to eliminate the "majority of items" that are not related to a query.

⇒ Signature files are based on the idea of "the inexact filter":

i.e. They provide a quick test, which discards many of the non-qualifying items.

- The "qualifying items" definitely "pass the test".
- "some additional items" (i.e. false hits (or) false drops) may also pass it accidentally.

Working:-

The signature file approach works as follows:

The "documents" are stored sequentially in the "text file", then "signatures" are stored in the signature file. (i.e signature are hash coded bit pattern).

→ When a query arrives, the signature file is scanned & many non-qualifying documents are discarded. The rest are either checked (or) they are return to the users as they are

Basic Concepts:-

→ A signature file typically use "Superimposed coding" (Moders-1949) to create the signature of a document

→ The document divided into "logical blocks" i.e indicated with 'D'.

→ The coding is based upon words in the item.

→ The 'words' are mapped into a "word signature"

word signature:

The word signature is a fixed length code 'F' with a fixed number 'F' of bits set to "1". i.e.m.

(or)

The word signature is a fixed length code (or) fixed size pattern of size 'F' with m-bits set to "1" and the rest of bits set to '0'.

here "F&m" are designed parameters.

→ The word signatures are "OR" ed together to form the "block signature".

The block signatures are concatenated to form the document signature.

for example (1): text : Free Text

	word	signature			
D=2 (logical blocks)	Free	001	000	110	010
F=12 (fixed length code)	Text	000	010	101	001
m=4					
	block signature:-	001	010	111	011

ex:-2:-

TEXT:- Computer science graduate students study

(assume block size is five words)

WORD	Signature			
Computer	0001	0110	0000	0110
Science	1001	0000	1110	0000
graduate	1000	0101	0100	0010
Students	0000	0111	1000	0100
Study	0000	0110	0110	0100
Block signature	1001	0111	1110	0110

fig:- superimposed coding.

Here

D=5, i.e it contain 5 words.

F=16, i.e The signature size is 'F'. F=16-bits

m=5 bits per word (i.e. how many 1's are present in each word).

ex:- 3:-

TEXT:- Data Base management system (assume block size is 4).

WORD	Signature				
Data	0000	0000	0000	0010	0000
Base	0000	0001	0000	0000	0000
management	0000	1000	0000	0000	0000
System	0000	0000	0000	0000	1000
Block signature	0000	1001	0000	0010	1000

D → Distinct non-common words

D = 4 (i.e 4 words Data, Base, management, system).

F → Signature size (How many bits present in signature)

F = 20 (i.e 20-bits)

m → (i.e How many bits set to '1')

m = 1

→ An important concept in signature files is the "False drop probability" "Fd".

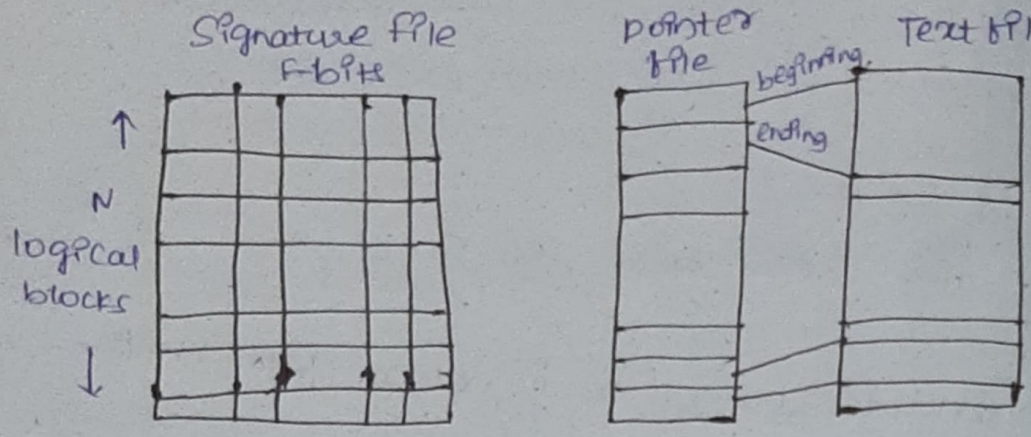
i.e. False drop probability gives the probability that the signature test will fail (or). The probability that a block signature seems to qualify but the block does not actually qualify. It is mathematically expressed as:

Fd = prob {signature qualifies / block does not}

* Symbols and definitions.

Symbol	Definition
F	signature size in bits.
m	no. of bits per word. (i.e. How many bits set to '1').
D	no. of distinct non-common words per document.
Fd	false drop probability.

Structure of Sequential Signature file(SSF)



* The signature files have been used in the following environments:

1. PC-based, medium size db.
2. WORMs (write once, read-many) optical disks.
3. parallel machines
4. distributed text db.

* Hypertext and XML Datastructures:

→ The advent (arrival) of the Internet and its exponential growth has introduced new mechanism for representing information.

→ This structure is called Hyper text and it is different from the traditional information storage data structure.

→ The Hyper text is stored in Hypertext Markup language (HTML) and extensible Markup language (XML).

Hypertext:

The Hypertext is based on the concept of "Expanding Information".

i.e, Blocks of information (text or graphics) can be linked to other blocks of information.

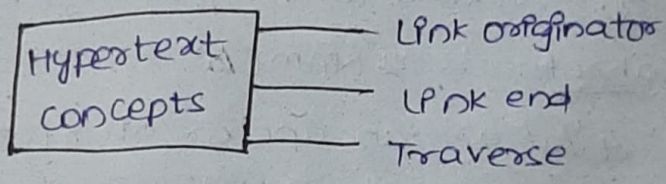
once two blocks of information have been linked together they provide an instant "gateway" to the other.
→ The Hypertext document is a NON-LINEAR document with many links.

Father of Hypertext:- Vannevar Bush - "1983"

He ~~wrote~~ wrote an article in 1945 on Hypertext.

Hypertext Concepts:-

Definition:- The Hypertext datastructure is used extensively in the internet environment and it requires an electronic media storage for the item.



Link originator:-
It is the starting point to the link.
It is surrounded by the "symbols" to indicate that it is a hypertext link.
The link-originator is an anchor.

Link end:-
The otherside to the link originator is Link end, where reader taken link is traversed Link end is also an anchor.

Traverse:-
The act of travelling from a link originator to its associated link end.

→ Example for hypertext is (HTML).

HTML:-

→ The Hypertext Markup language (HTML) defines the internal structure for information exchange across the www on Internet.

⇒ The HTML document is composed of the text of item along with "HTML tags" that describe how to display the document.

⇒ Tags are formatting (or) structural keywords expressed b/w < and > symbols.

ex:- <title>, <body> ....
<head> ....

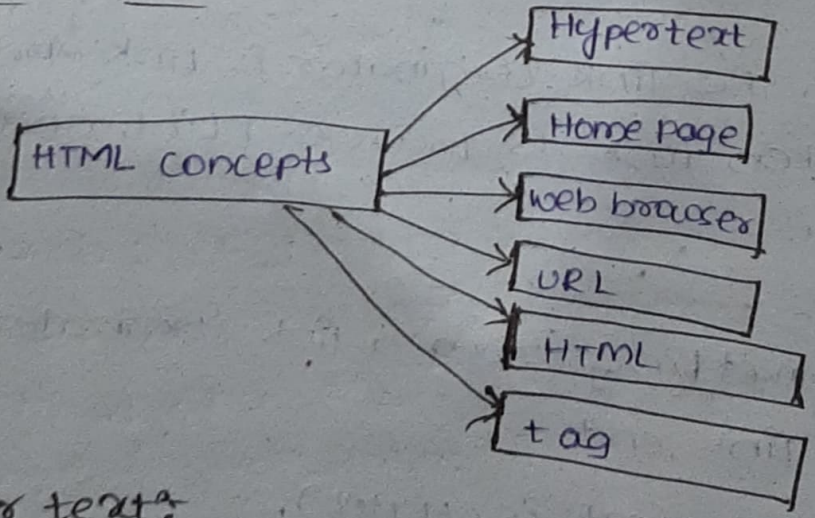
→ The HTML tag associated with hyperlinks is indicated with <a href = ... # NAME

where 'a' & '/a' are an anchor at start and end tags. denoting "text" that user can active.

"href" = Hypertext reference, that contains a filename.

"# Name" → defines "destination point".

HTML Concepts:-



Hyper text:

→ Text with the links to other texts.

Homepage:
→ The Root of Hypertext structure.

Web browser:
A tool to retrieve a document using "URL" from the web server.

URL: Uniform Resource Locator
The address of webpage is called URL.

HTML: Hypertext Markup Language.
→ HTML is the language to design the webpage.

Tags: HTML is made with the "tags" and they are enclosed in "angular brackets".

XML:
→ The extensible Markup language is starting to become a standard datastructure on the WEB.
→ Its first recomandation (1.0) was issued on feb 10, 1998.
The Hyper links for XML are being defined in the "XLink" (XML linking language) and "Xpoint" (XML pointer language) specifications.
→ XML objective is extending HTML with semantic information.
→ The logical datastructure with in XML is defined by a DataType Description (DTD).
→ XML was designed to store and transport data.
→ XML was designed to be both human-and-machine-readable.

Simple ex: <company> HP </company>
<City> Hyd </City>
<state> TS </state>
<product> HP </product>

* Hidden Markov Models:-

The Hidden Markov Model (HMM) is a relatively simple way to model "sequential data". The HMM implies that the Markov model underlying the data is hidden (or) unknown to you.

Q:- Why Hidden Markov Model?

A:- The reason it is called a Hidden Markov Model is because we are constructing an interface model based on the assumptions of a Markov process. The Markov process assumption is simply that the "future is independent of the past given the present".

Andrei Andreyvich Markov (1856-1922)

Andrei Andreyvich Markov was a Russian mathematician.

He is best known for his work on stochastic processes. A primary subject of his research later became known as Markov chains and Markov processes.

→ Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a markov process.

→ The Hidden Markov model (HMM) have been applied for the last 20 years to solving problems in

"* Speech recognization" and to a lesser extent in the areas locating named entities.

* optical character recognization and
* topic identification.

⇒ More recently HMMs have been applied more generally to information retrieval search with good result.

the easiest way to understand HMM is by an example.

Lets take the example of three state Markov Model of the "stock market".

state 1 (S1) : Market decreased
state 2 (S2) : Market did not change
state 3 (S3) : Market increased in value.

⇒ The Movement b/w "states" can be defined by a "state transition Matrix" with "state transitions". (i.e. this assumes you can go from any state to any other state):

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.3 & 0.4 \\ 0.1 & 0.6 & 0.3 \\ 0.6 & 0.7 & 0.5 \end{bmatrix}$$

Given that the Market fall on oneday (state 1). the market suggests the probability of the market not changing the next day is "0.1"

This then, allows questions such as
The probability that the Market will increase for the next 4 days then fall.
This would be equivalent to the sequence of

$$SEQ_1 = \{S_3, S_3, S_3, S_3, S_1\}$$

→ In order to simplify our model, Let assume that instead of the current state being dependent upon all the previous states, Lets assume it is only dependent upon the last state (i.e. discrete

first order Markov chain).
then it would be calculate by the formula:

$$P(SEQ) = P[S_3, S_3, S_3, S_3, S_1]$$

$$= P[S_3] * P[S_3/S_3] * P[S_3/S_3] * P[S_3/S_3] * P[S_1/S_3].$$

$$= S_3(init) * a_{3,3} * a_{3,3} * a_{3,3} * a_{3,3}$$

$$= 1.0 * (0.5) * (0.5) * (0.5) * (0.4)$$

$$= 0.05 \text{ (or) } 0.5.$$

is the ~~post~~ probability

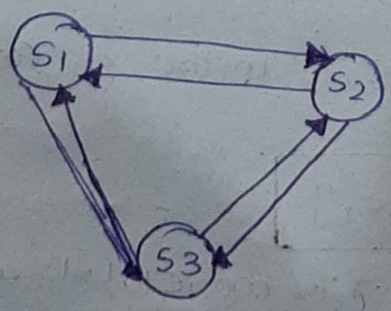
In the equation we also assume the probability of the initial state of S_3 is $S_3(init) = 1$

The following graph depicts the model.

→ The directed lines indicate the state transition probabilities $a_{i,j}$.

→ There is also an implicit loop from every state back to itself.

In the example every state corresponded to an observable event. (i.e. change in the Market).



A more formal definition of a discrete Hidden Markov Model is as consisting the following.

1. $S = \{S_0, S_1, \ldots S_{n-1}\}$ as a "finite set of states". where 'S_0' always denotes the initial state.
Typically these states are "inter connected" such that any state can be reached from any other state.

2. $V = \{V_0 \ldots V_{m-1}\}$ is a "finite set of output symbols" i.e. physical o/p from the system being Modeled.

3. $A = S \times S$ is a transition probability Matrix.
Where "a_{ij}" represents the probability of transitioning from state 'i' to state 'j' such that
for all $i = 0, \ldots, n-1$ $\sum_{j=0}^{n-1} a_{ij} = 1$ for all $i = 0, \ldots, n-1$.
→ Every value in the matrix is positive value b/w 0 and 1
→ For the case, where every state can be reached from every other state every value in the matrix will be "non-zero".

4. $B = S \times V$ is an o/p probability matrix where element $b_{j,k}$ is a function determining the probability and $\sum_{k=0}^{m-1} b_{jk} = 1$ for all $j = 0, \ldots n-1$.

5. The initial state distribution.

⇒ The complete specification of a HMM requires specification of the states, the o/p symbols and three probability measures for the state transition-s, output probability functions and the initial states. The distributions are frequently called A, B and Π, and the following notation is used to define the model: $\boxed{\lambda = (A, B, \Pi)}$