VASIREDDY VENKATADRI
INSTITUTE OF TECHNOLOGY

Data Structures

UNIT-V

Syllabus

- **UNIT-5: Tries:** Digital Search Trees(Tries), Operations, Different types of Tries
- **Pattern Matching Algorithms**.

Tries

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- A trie is a tree-based data structure for storing strings in order to support fast pattern matching.
- The word ‘trie’ has been extracted from the word “re**trie**val”.
- Fredkin introduced tries in the 1960’s.
- The primary query operation that a trie can do is prefix matching.
- A *trie* (from retrieval), is a multi-way tree structure useful for storing strings.
- It has been used to store large dictionaries of English (say) words in spelling-checking programs and in natural-language "understanding" programs.

Applications of a Trie

- Spell checkers.
- Data compression
- Computational biology.
- Routing tables for IP addresses.
- Storing and querying XML documents.
- Associative arrays, associative indexing

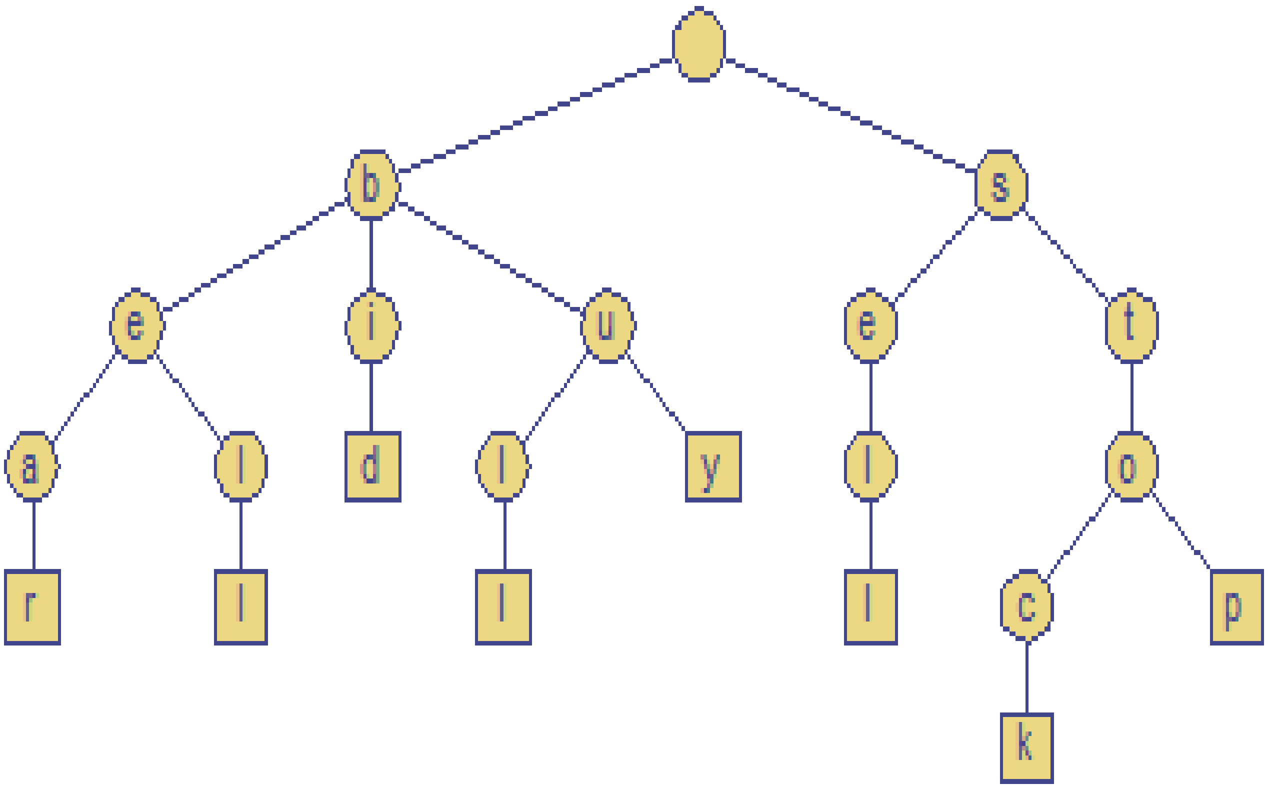
Types of Tries

VVIT
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- Standard trie
- Compressed trie
- Suffix trie
- Patricia trie

- **Standard trie:**
- Let S be a set of s strings from alphabet sigma such that no string in S is a prefix of another string. A standard trie for S is an ordered tree T with the following properties:
  - Each node of T, except the root, is labeled with a character of sigma.
  - The ordering of children of an internal node of T is determined by a canonical ordering of the alphabet sigma.
  - T has s external nodes, each associated with a string of S.

Example: standard trie for the set of strings S = {bear, bell, bid, bull, buy, sell, stock, stop}

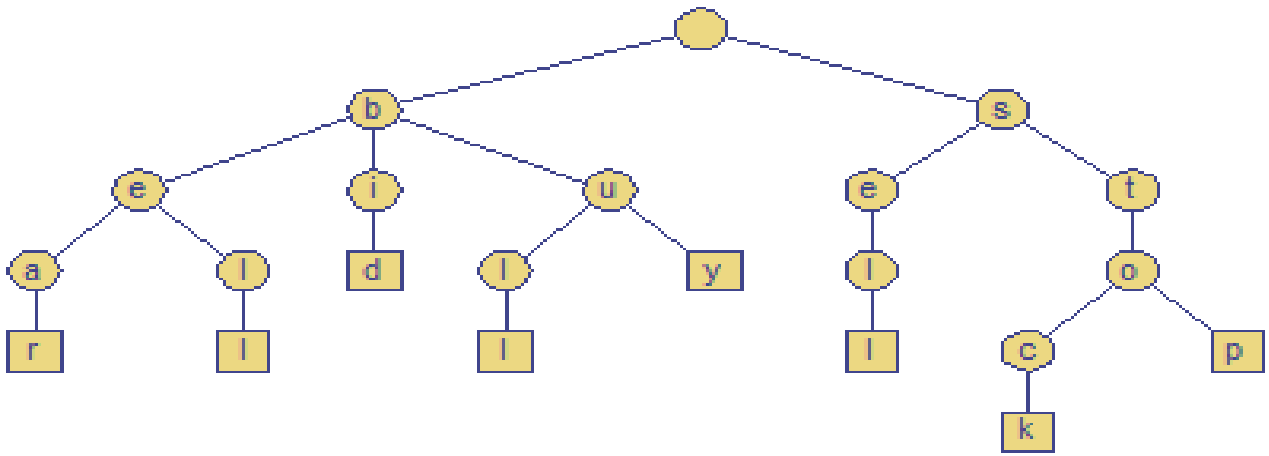


compressed Trie

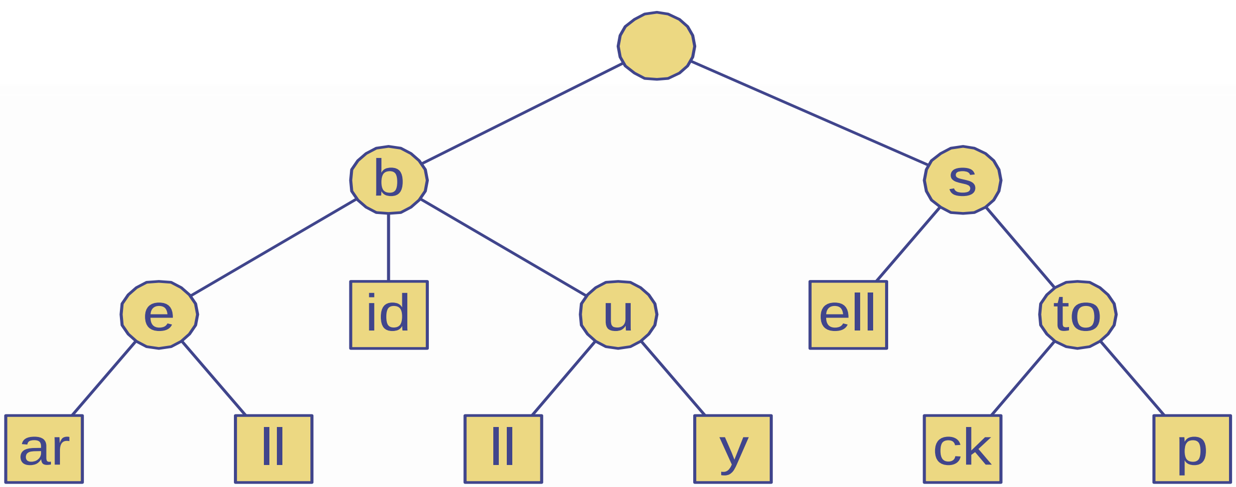
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- There is potential space inefficiency in the standard trie.
- The compressed trie overcomes this drawback.
- It was introduced by D.R.Morrison in 1968.
- A compressed trie is also similar to a standard trie but it ensures that each internal node in the trie has at least two children.

- Standard Trie

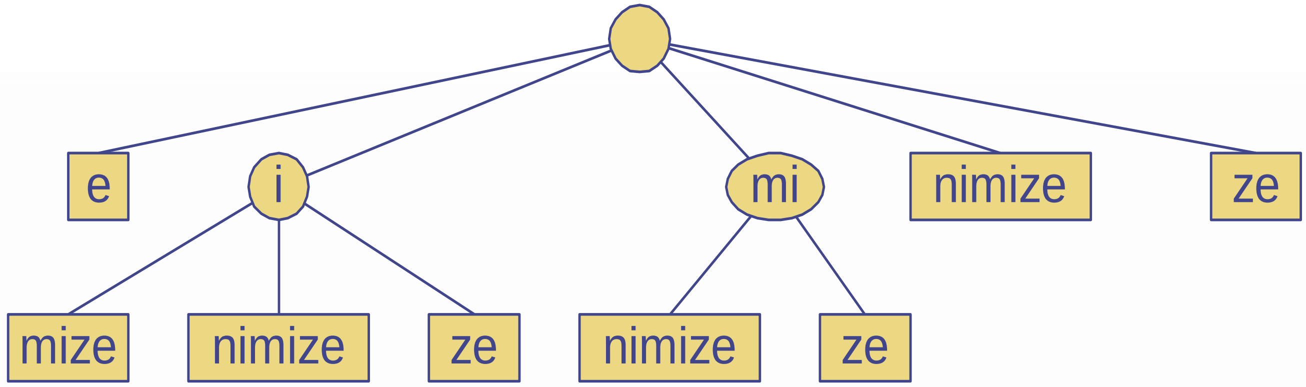


- Compressed Trie



Suffix Trie

- **Suffix trie:** One of the primary applications for tries is for the case when the strings in the collection S are all the suffixes of a string X. Such a trie is called the suffix trie (or suffix tree or position tree).
- For example: {e,imize,inimize,ize,minimize,mize,nimize,ze}

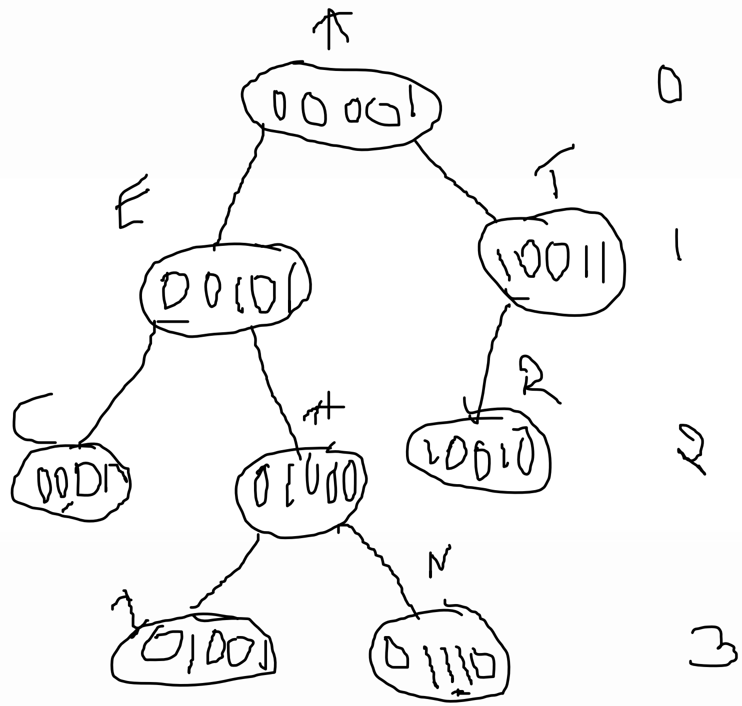


Digital Search Trees

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- A Digital Search tree is a binary tree in which each node contains one element
- Every element is attached as a node using the binary representation
- The bits are read from left to right
- All the keys in the left subtree of a node at level i have bit 0 at ith position similarly the right subtree of a node at level i have bit 1 at ith position.

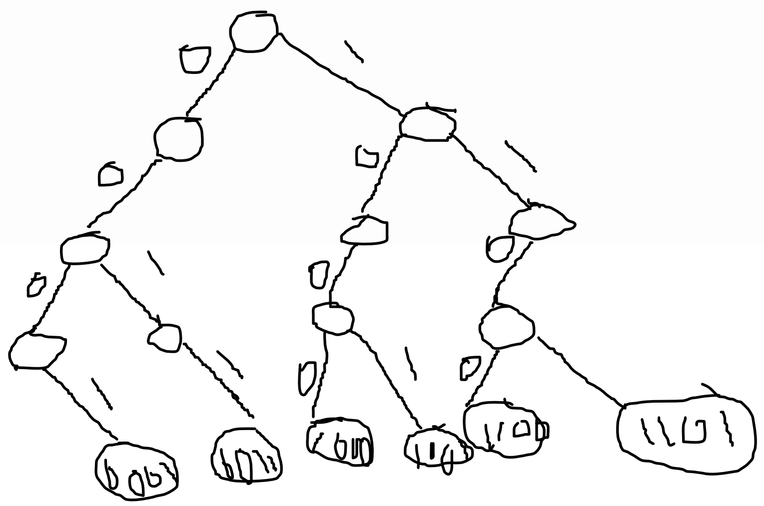
A	T	E	R	C	H	I	N
00001	10011	00101	10010	00011	01000	01001	01110



Binary Trie

- A Binary trie is a binary tree that has two kinds of nodes
  - Branch nodes
  - Element nodes
- Branch nodes has two fields left child and right child. It has no data member.
- The element node has single data member.

- 0001,0011,1000,1001,1100,1101



Compressed Binary Trie

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- The binary trie may contain branch nodes whose degree is one.
- For creating compressed binary trie, eliminate nodes with degree one.

Patricia Trie

- The Patricia stands for Practical Algorithm to Retrieve Information Coded in Alphanumeric.
- Building a Patricia trie is quite simple.
- In Patricia trie every node will have bit index.
- This number is written at every node.
- Based on this number the trie will be searched

Patricia Trie

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- Index 43210

A : 00001

S : 10011

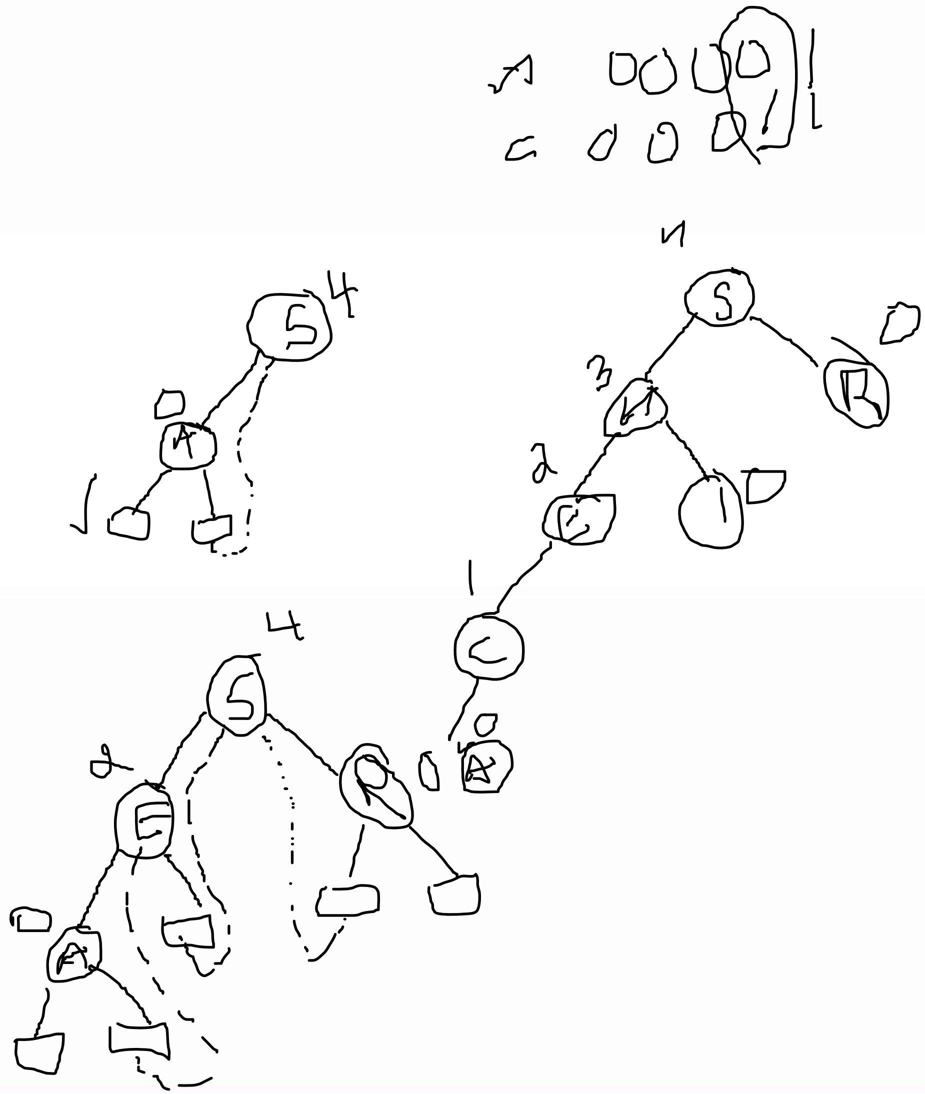
E : 00101

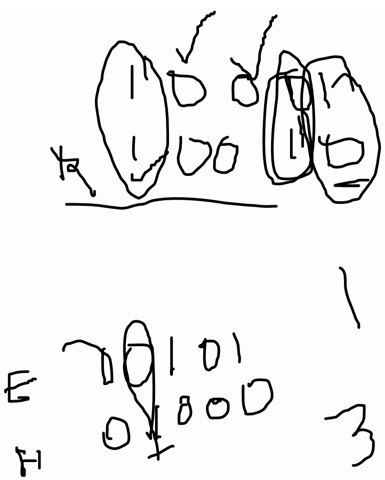
R : 10010

C : 00011

H : 01000

I : 01001





Pattern Matching Algorithms

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

- **Pattern matching** is the process of checking whether a specific sequence of characters/tokens/data exists among the given data.
- String matching is virtually very essential for computer users.
- While editing the text, the user may want to section the paragraphs, searches for a pattern, replace a pattern.
- This searching not only applied for text patterns but also to molecular biology, where people extract the required patterns from a sequence of DNA.
- There are several pattern matching algorithms available. The following are the essential techniques:
  - Brute Force or Straight forward algorithm
  - Knuth-Moris-Pratt algorithm
  - Boyer Moore Algorithm

Brute Force algorithm

- It is a simple approach
- The comparison starts at the first characters of text T and pattern P.
- If they match, comparison starts at second character and the process would continue till all the characters in the pattern matches in the text or to the end of the text.
- Example: Text: abbabbabb
- Pattern: bab

$(n-m) \times m$

$O(nm)$

abbaddabb

bab

$i = \cancel{4}\ \cancel{5}\ 6$ $\quad i = 7$

$j = \cancel{0}\ \cancel{1}\ 0$ $\quad j = 0$

$i \le$

$\boxed{i \le n - m}$

Boyer Moore Algorithm

- This algorithm searches for a pattern in quite a different way compare to brute force algorithms.
- The main idea of the BM algorithm is to improve the running time of the Brute-Force algorithm by adding two potentially time saving heuristics:
- **Looking-Glass heuristic:** when testing a possible placement of P against T, begin the comparisons from the end of P and move backward to the front of P.
- **Character-Jump heuristic:** During the testing of a possible placement of P against T, a mismatch of text character T[i] = c with the corresponding patter character P[j] is handled as follows:
- If c is not contained in the pattern P, shift the pattern beyond the position of mismatch in T otherwise move to the last occurrence of the c in the pattern.

i = m-1
j = m-1

X	Y	X	Z	X	X	Y	X	T	Z	X	Y	X	Z	X	Y	X	X	Y	Y

X	Y	X	Z	X	Y

lasts 0 1 . – – – . – – X(9) (90) (92) . . 127

-1 | -1 | -1 | -1 | -1 | -1 | 4 | 5 | 3 | -1 | -1

X Y Z

last(P.charAt(i)) = i

Knuth Morris Pratt Algorithm

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

15 Ans

- In Brute Force and Boyer Moore Algorithms, when the mismatch occurs it simple through away the information and restart the comparison for another set of characters from the text.
- Thus again and again with the next incremental position of text the characters from pattern are matched.
- This ultimately reduces the efficiency of pattern matching algorithms.

Knuth Morris Pratt Algorithm

- The main aim of KMP is to avoid the repeated comparisons of characters.
- The basic idea behind this algorithm is to build a prefix array which is also called ¶ array. π or lps
- This prefix array is built using the prefix and suffix information of the pattern.
- Overlapping prefix and suffix is used in KMP.

Knuth Morris Pratt Algorithm

- Consider the pattern abadab
- Initially put 0 in 0th location of prefix array

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5
a	b	a	d	a	b

prefix:ϵ,

suffix:ϵ,

Knuth Morris Pratt Algorithm

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

ababada

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6
a	b	a	b	a	d	a

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6
a	b	a	b	a	d	a
0	0	1	2	3	0	1

prefix:ϵ,

suffix:ϵ,

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
b	a	d	b	a	b	a	b	a	b	a	d	a	a	b

a	b	a	b	a	d	a
0	1	2	3	4	5	6