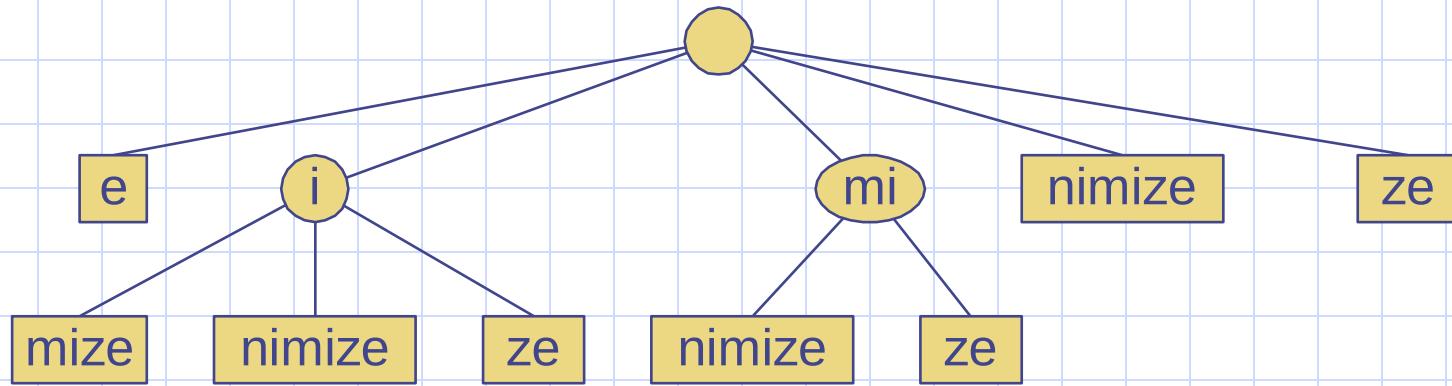


Tries



Preprocessing Strings

- ◆ Preprocessing the pattern speeds up pattern matching queries
 - After preprocessing the pattern, KMP's algorithm performs pattern matching in time proportional to the text size
- ◆ If the text is large, immutable and searched for often (e.g., works by Shakespeare), we may want to preprocess the text instead of the pattern
- ◆ A trie is a compact data structure for representing a set of strings, such as all the words in a text
 - A tries supports pattern matching queries in time proportional to the pattern size

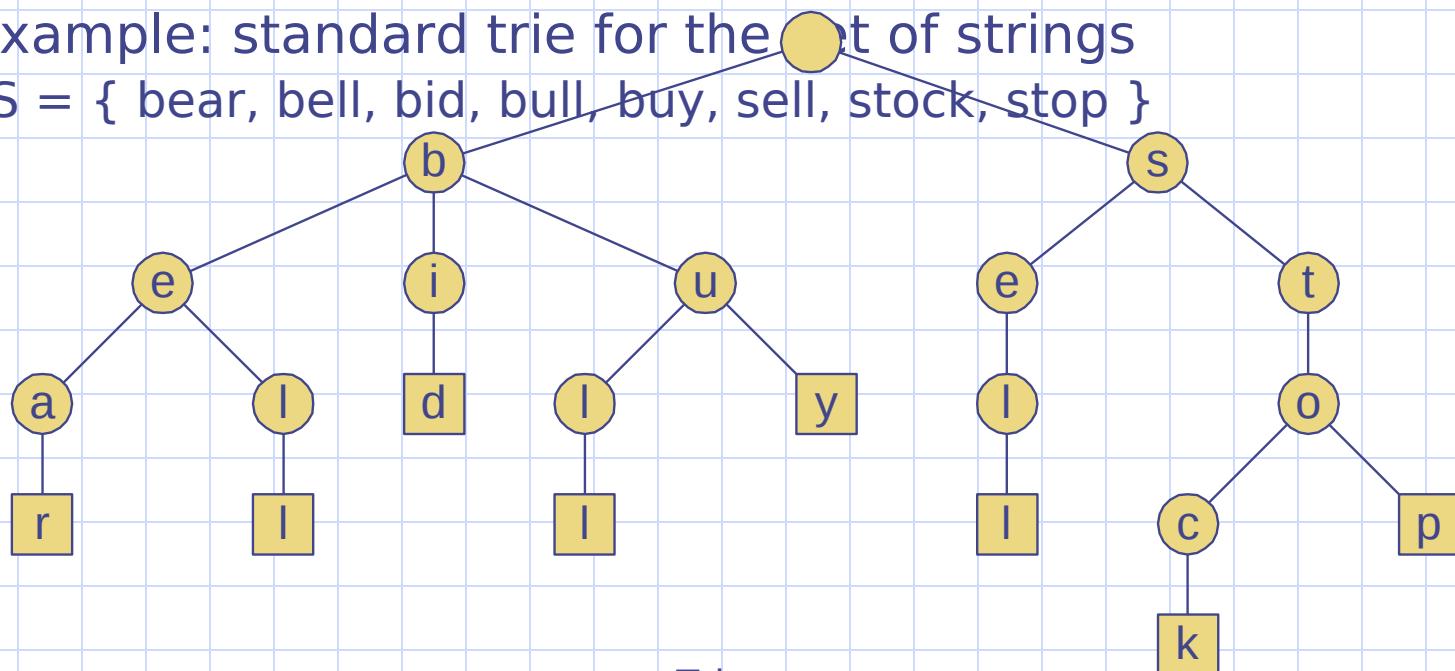
Standard Tries

◆ The standard trie for a set of strings S is an ordered tree such that:

- Each node but the root is labeled with a character
- The children of a node are alphabetically ordered
- The paths from the external nodes to the root yield the strings of S

◆ Example: standard trie for the set of strings

$S = \{ \text{bear}, \text{bell}, \text{bid}, \text{bull}, \text{buy}, \text{sell}, \text{stock}, \text{stop} \}$



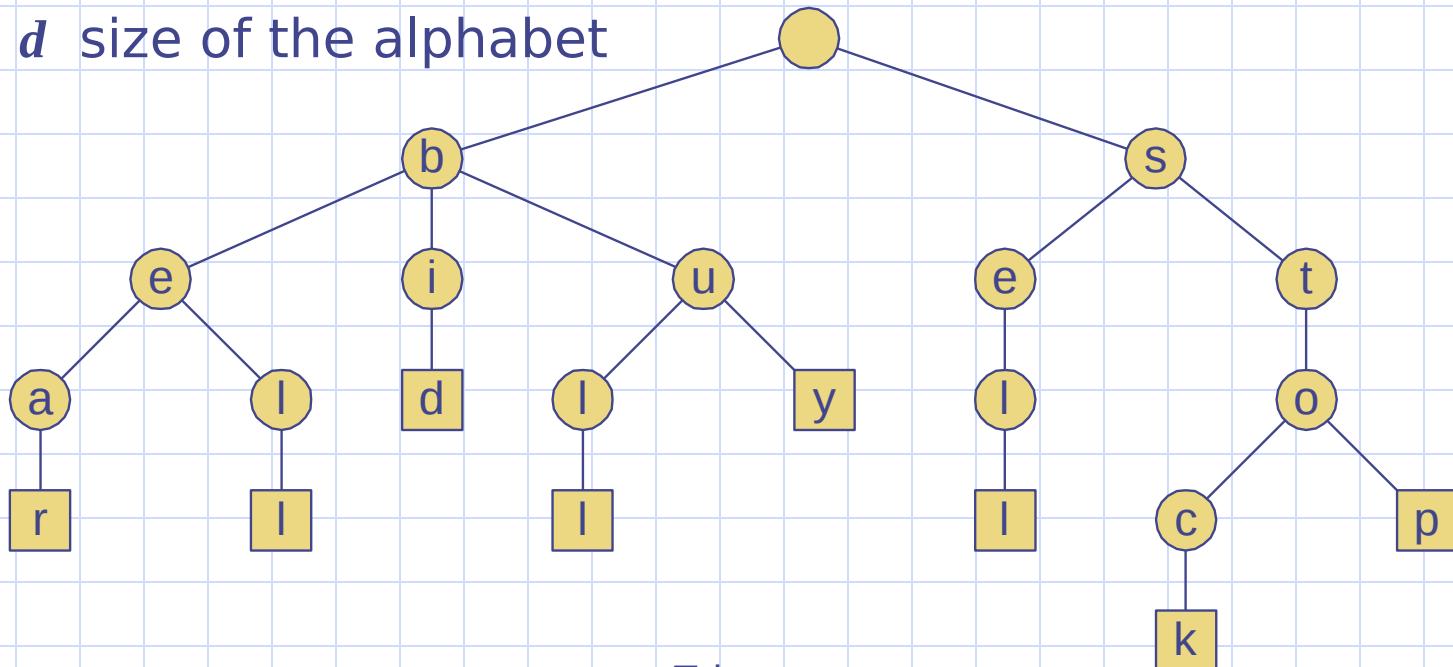
Analysis of Standard Tries

- ◆ A standard trie uses $O(n)$ space and supports searches, insertions and deletions in time $O(dm)$, where:

n total size of the strings in S

m size of the string parameter of the operation

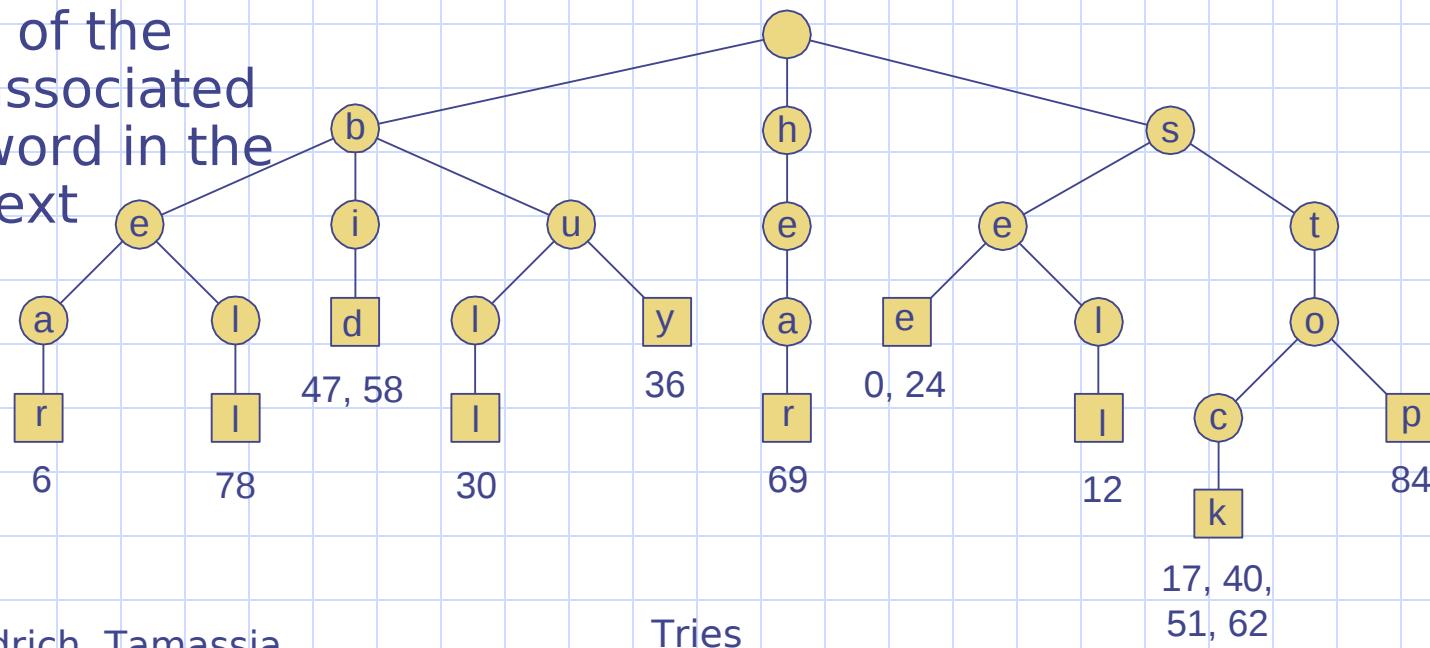
d size of the alphabet



Word Matching with a Trie

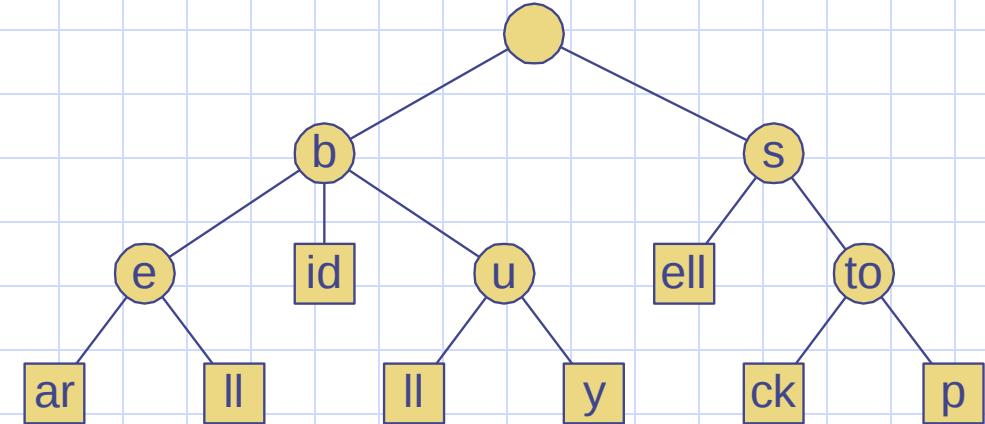
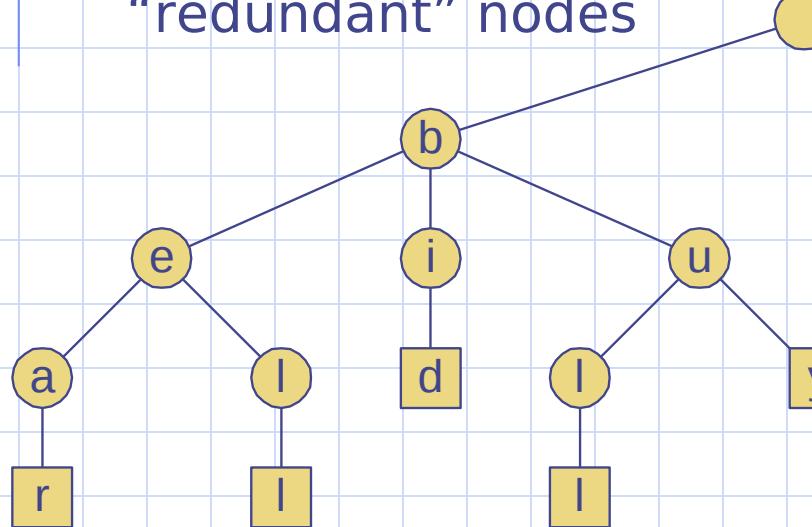
- ◆ We insert the words of the text into a trie
- ◆ Each leaf stores the occurrences of the associated word in the text

see	a	bear?	sell	i	stock!
0	1	2	3	4	5
see	a	bu	ll?	buy	stock!
24	25	26	27	28	29
b	i	d	st	ock!	b
47	48	49	50	51	52
id	s	t	o	c	k!
53	54	55	56	57	58
he	a	r	t	h	e!
69	70	71	72	73	74
ar	t	h	e	b	e!
75	76	77	78	79	80
he	l	l?	s	t	o
81	82	83	84	85	86
l	e	p	!	!	!
87	88				



Compressed Tries

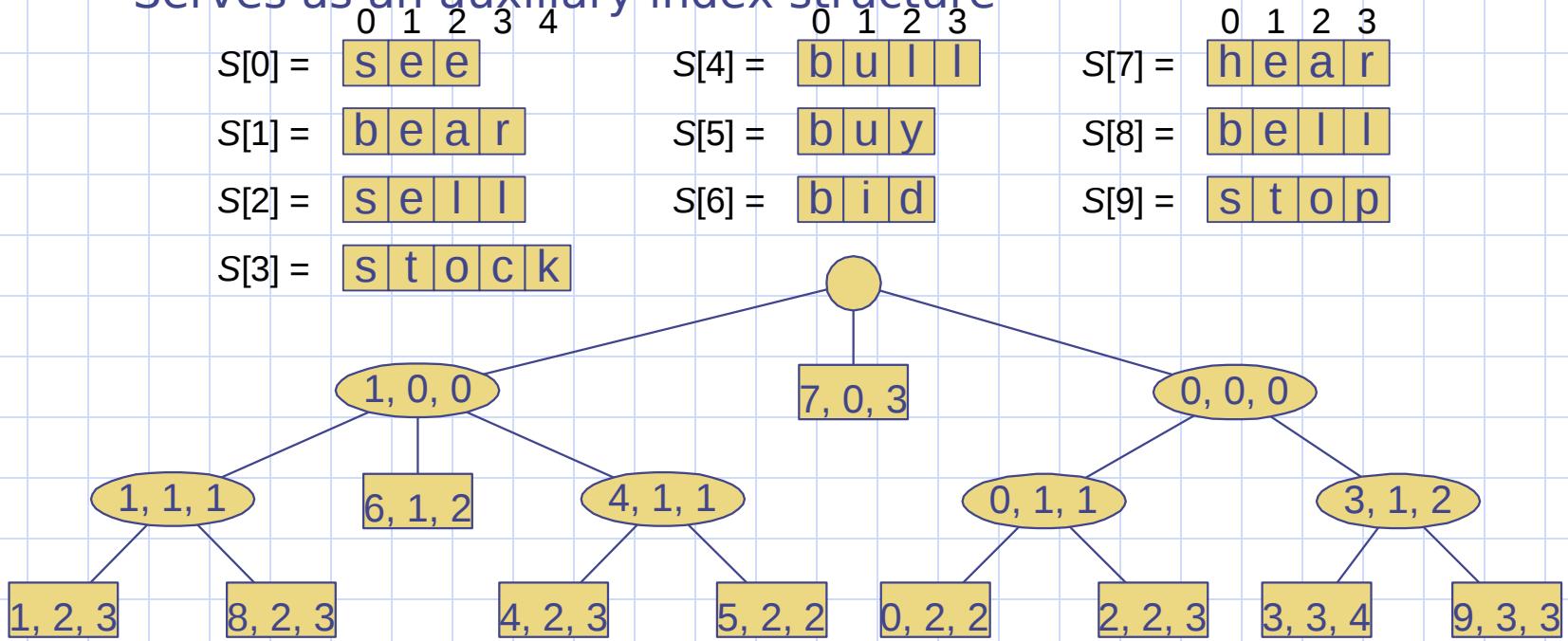
- ◆ A compressed trie has internal nodes of degree at least two
- ◆ It is obtained from standard trie by compressing chains of “redundant” nodes



Compact Representation

- ◆ Compact representation of a compressed trie for an array of strings:

- Stores at the nodes ranges of indices instead of substrings
- Uses $O(s)$ space, where s is the number of strings in the array
- Serves as an auxiliary index structure

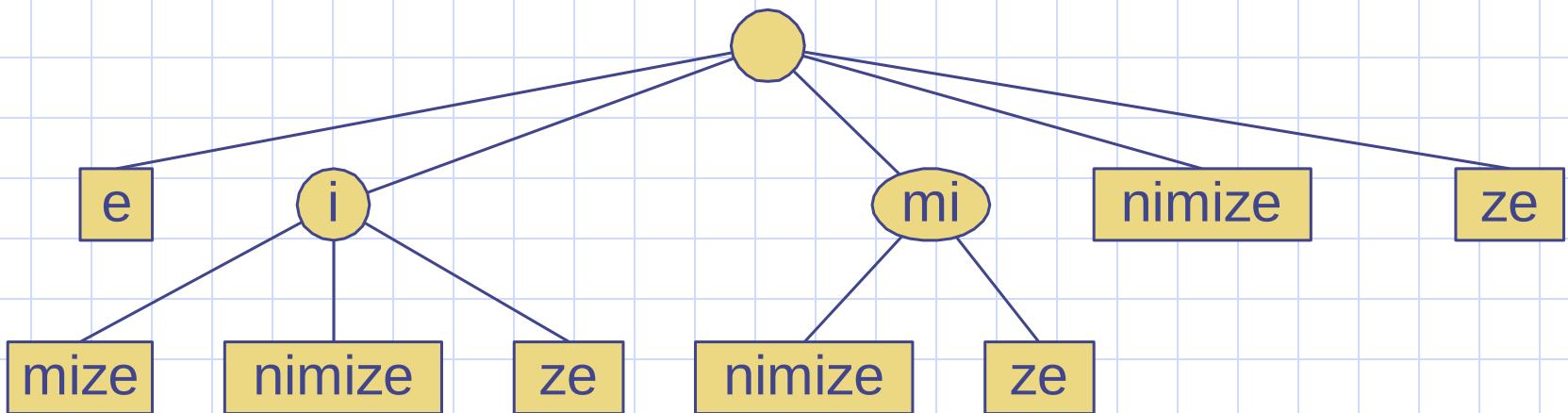


Tries

Suffix Trie

- ◆ The suffix trie of a string X is the compressed trie of all the suffixes of X

m	i	n	i	m	i	z	e
0	1	2	3	4	5	6	7



Analysis of Suffix Tries

- ◆ Compact representation of the suffix trie for a string X of size n from an alphabet of size d
 - Uses $O(n)$ space
 - Supports arbitrary pattern matching queries in X in $O(dm)$ time, where m is the size of the pattern
 - Can be constructed in $O(n)$ time

