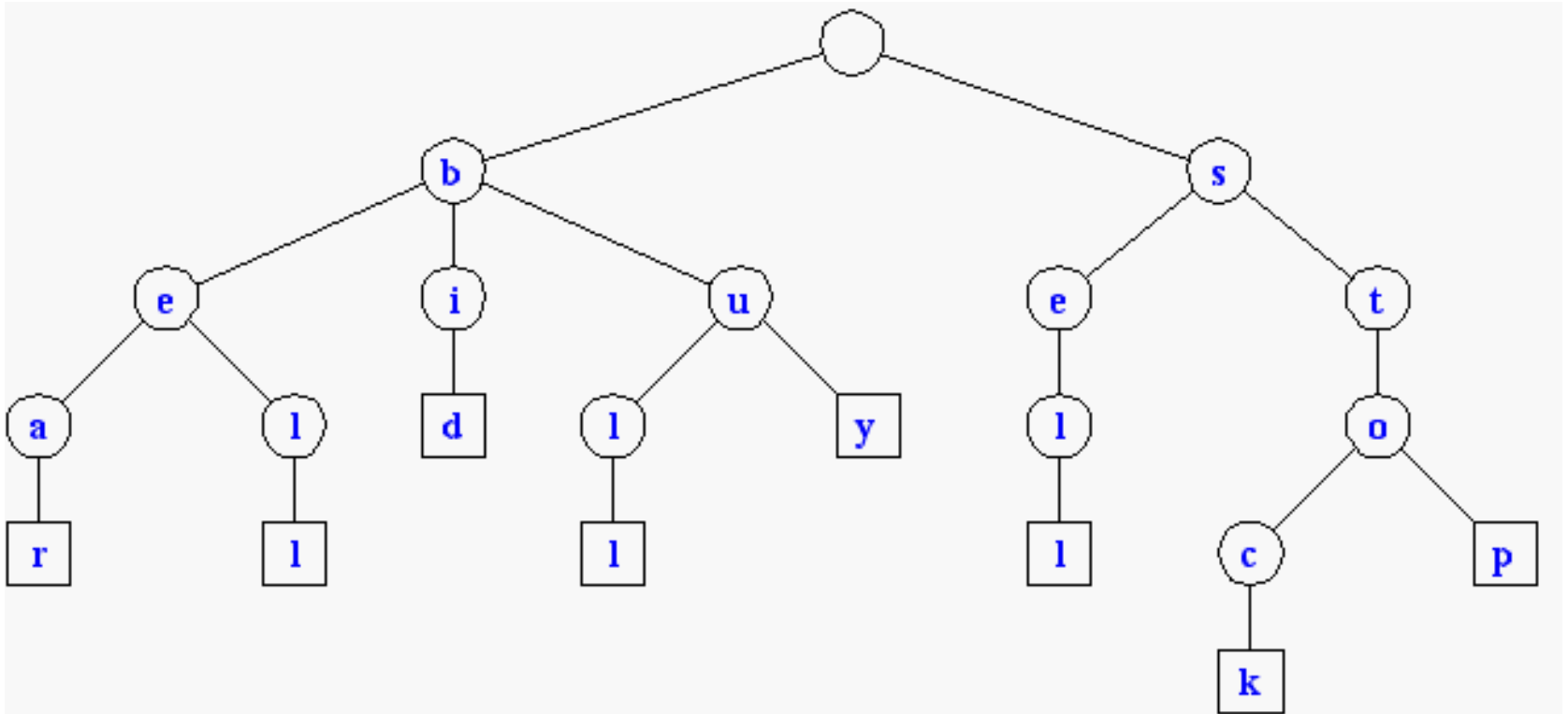


Tries

- Standard Tries
- Compressed Tries
- Suffix Tries

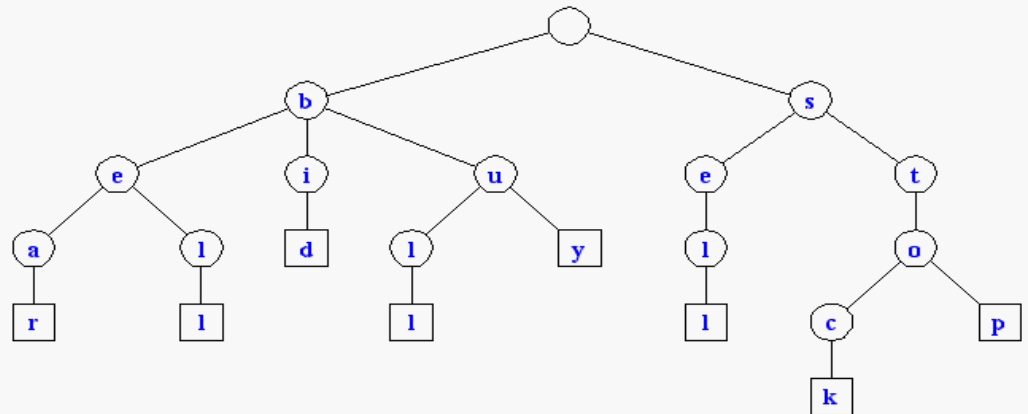


Standard Tries

- 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

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



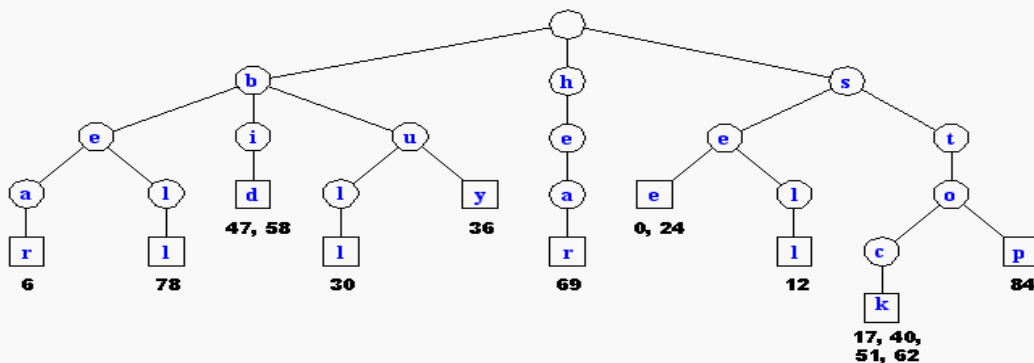
- A standard trie uses $O(n)$ space. Operations (**find**, **insert**, **remove**) take time $O(dm)$ each, where:

- n = total size of the strings in S ,
- m = size of the string parameter of the operation
- d = alphabet size,

Applications of Tries

- A standard trie supports the following operations on a preprocessed text in time $O(m)$, where $m = |X|$
 - word matching*: find the first occurrence of word X in the text
 - prefix matching*: find the first occurrence of the longest prefix of word X in the text
- Each operation is performed by tracing a path in the trie starting at the root

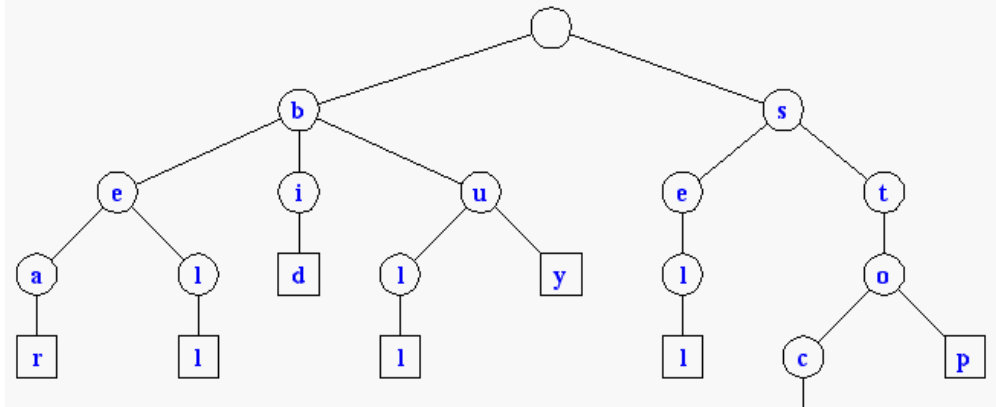
s	e	e		a		b	e	a	r	?		s	e	l	l		s	t	o	c	k	!		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
s	e	e		a		b	u	l	l	?		b	u	y		s	t	o	c	k	!			
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46		
b	i	d		s	t	o	c	k	!		b	i	d		s	t	o	c	k	!				
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68			
h	e	a	r		t	h	e		b	e	l	l	?		s	t	o	p	!					
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88					



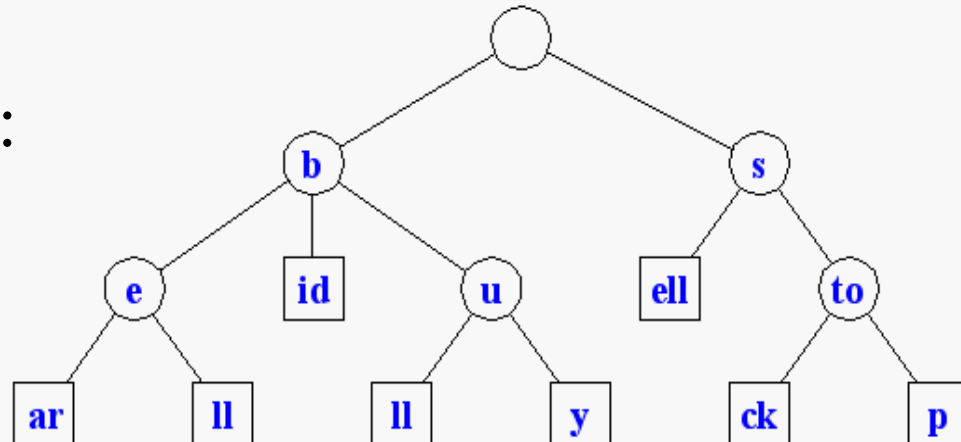
Compressed Tries

- Trie with nodes of degree at least 2
- Obtained from standard trie by compressing chains of *redundant nodes*

Standard Trie:

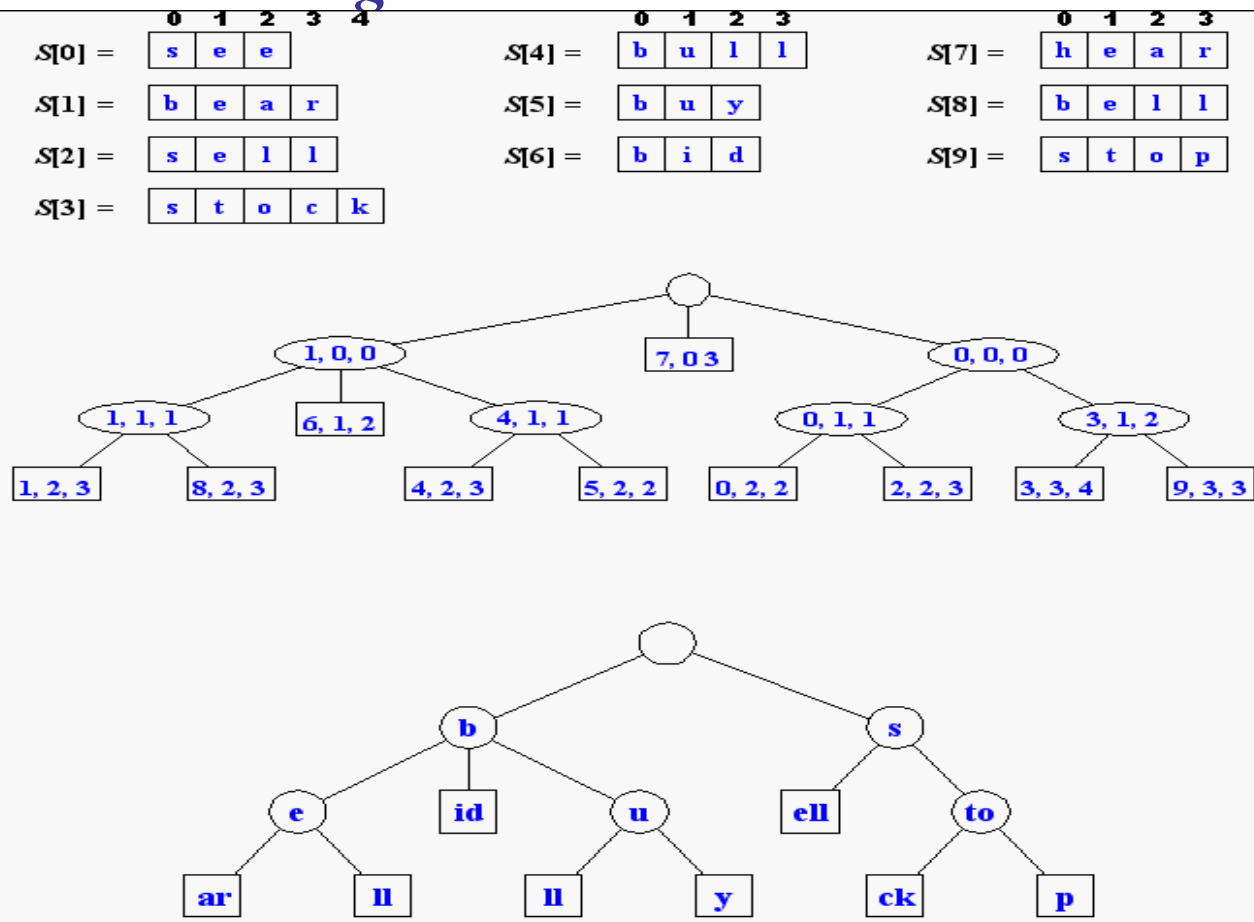


Compressed Trie:

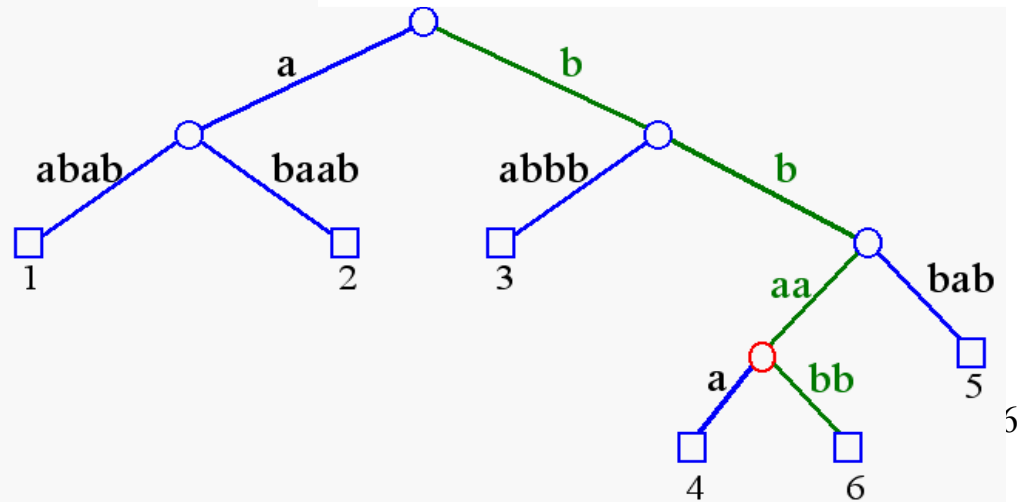
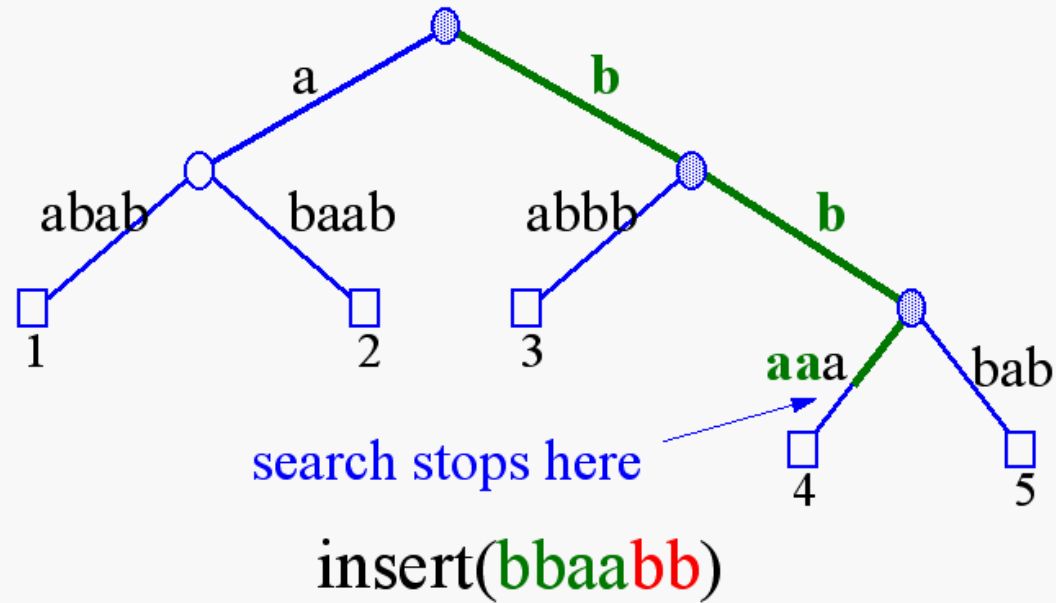


Compact Storage of Compressed Tries

- A compressed trie can be stored in space $O(s)$, where $s = |S|$, by using $O(1)$ space *index ranges* at the nodes



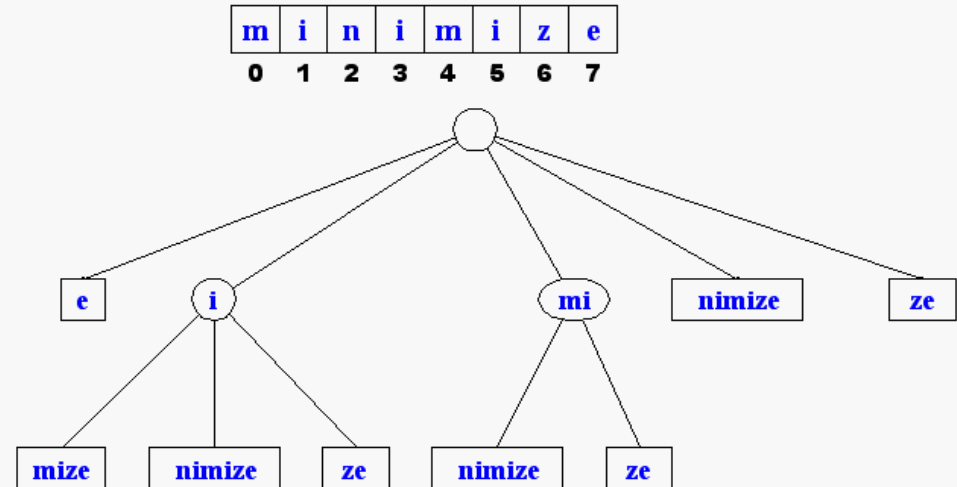
Insertion and Deletion into/from a Compressed Trie



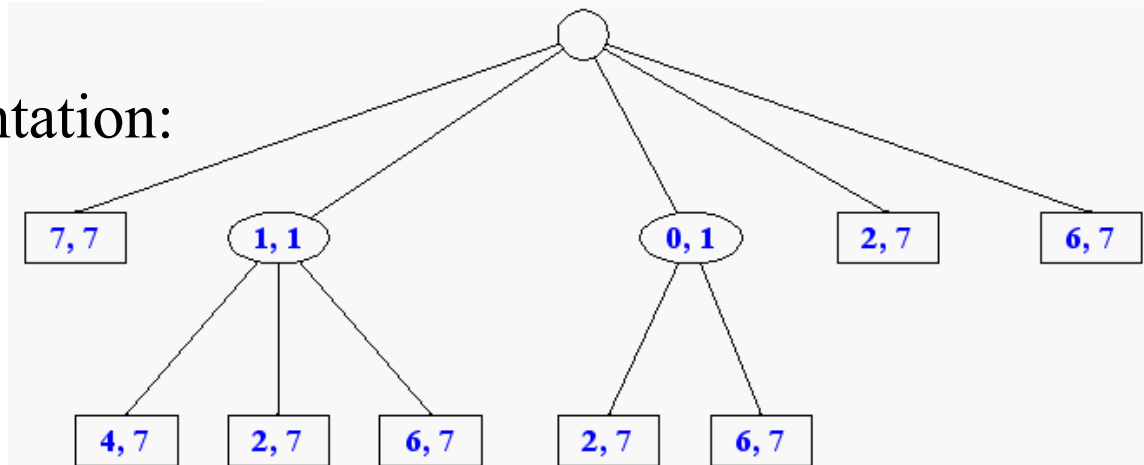
Suffix Tries

- A *suffix trie* is a compressed trie for all the suffixes of a text

Example:

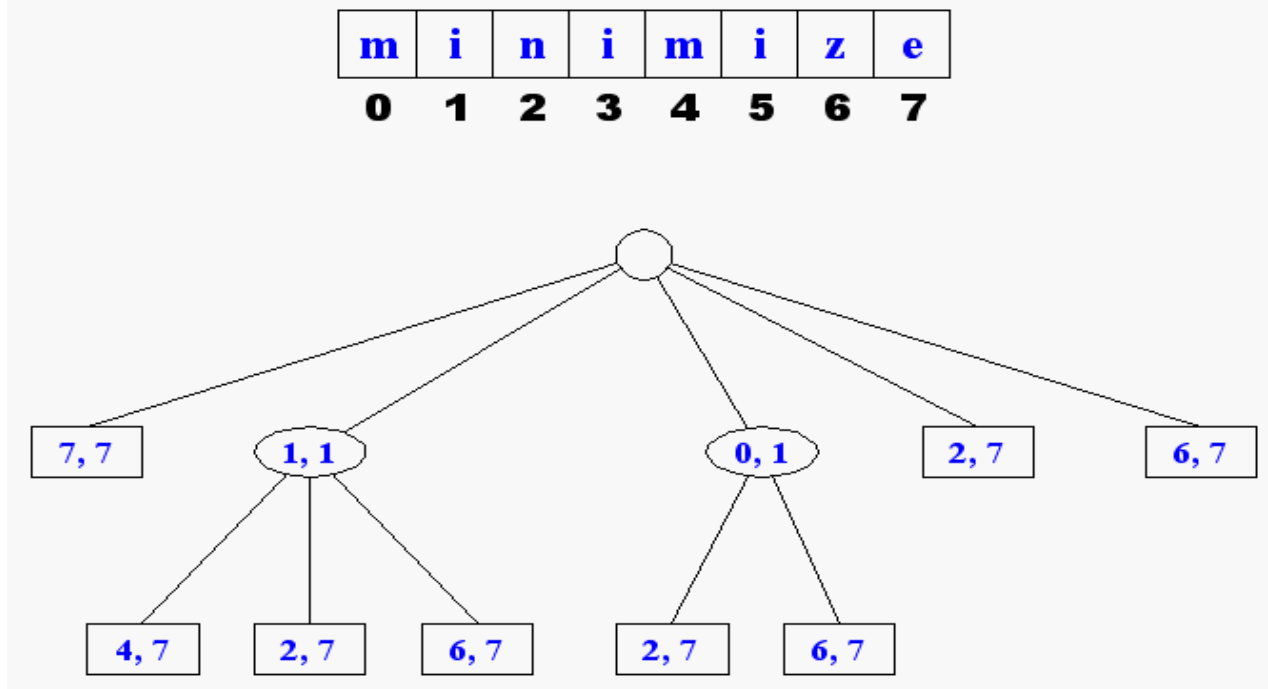


Compact representation:



Properties of Suffix Tries

- The *suffix trie* for a text X of size n from an alphabet of size d
 - stores all the $n(n-1)/2$ *suffixes* of X in $O(n)$ space
 - supports arbitrary *pattern matching* and prefix matching queries in $O(dm)$ *time*, where m is the length of the pattern
 - can be constructed in $O(dn)$ *time*



Tries and Web Search Engines

- The *index of a search engine* (collection of all searchable words) is stored into a compressed trie
- Each leaf of the trie is associated with a word and has a list of pages (URLs) containing that word, called *occurrence list*
- The trie is kept in internal memory
- The occurrence lists are kept in external memory and are ranked by relevance
- Boolean queries for sets of words (e.g., Java and coffee) correspond to set operations (e.g., intersection) on the occurrence lists
- Additional *information retrieval* techniques are used, such as
 - stopword elimination (e.g., ignore “the” “a” “is”)
 - stemming (e.g., identify “add” “adding” “added”)
 - link analysis (recognize authoritative pages)