Indexed Search Tree (Trie)

Nelson Padua-Perez

Chau-Wen Tseng

Department of Computer Science

University of Maryland, College Park

- **Trie datastructure applications**

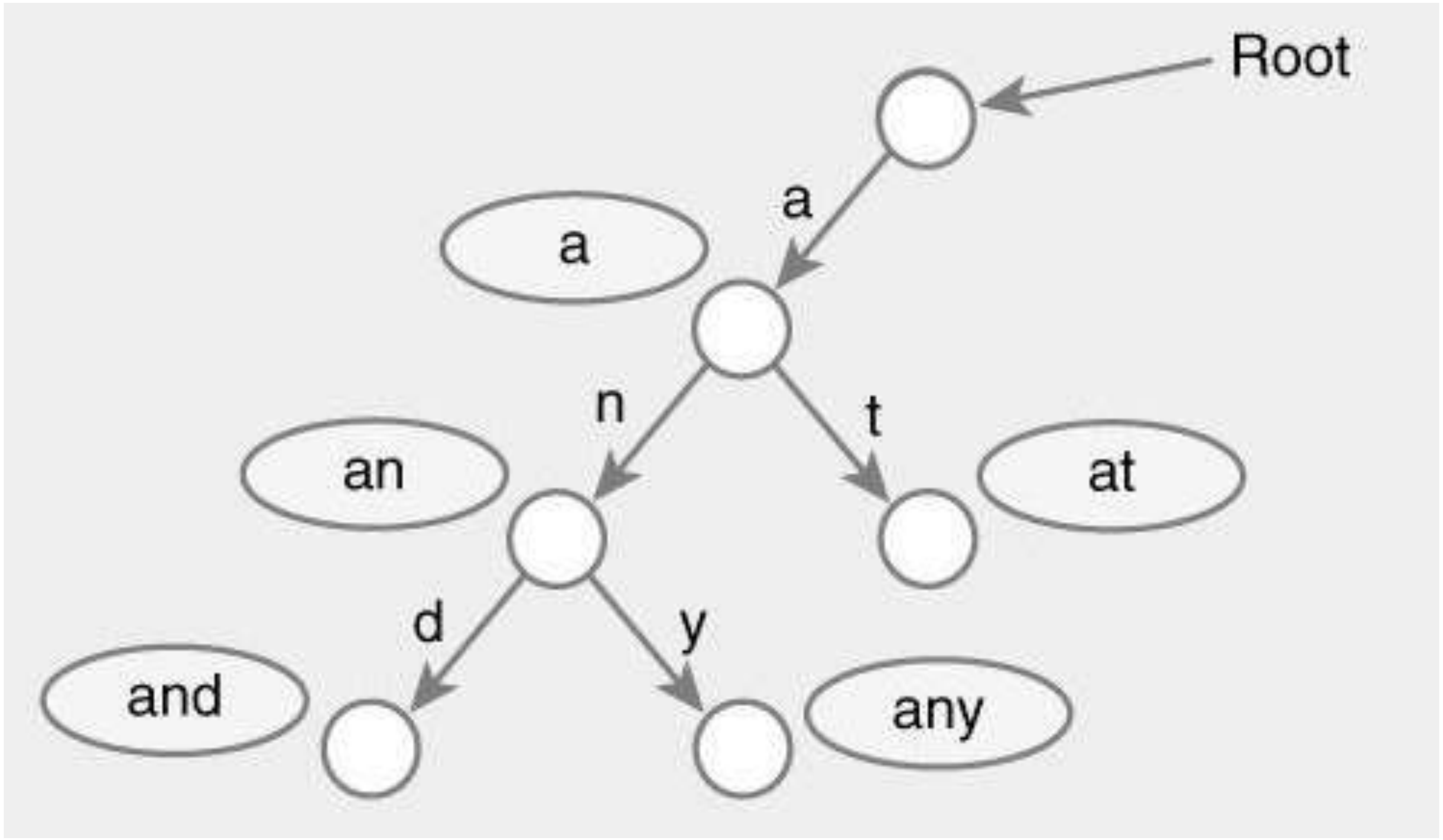
- **--auto completion**
- **--search engines**
- **--IP routing**

Types of Tries

- Standard
  - Single character per node
- Compressed
  - Eliminating chains of nodes
- Compact
  - Stores indices into original string(s)
- Suffix
  - Stores all suffixes of string

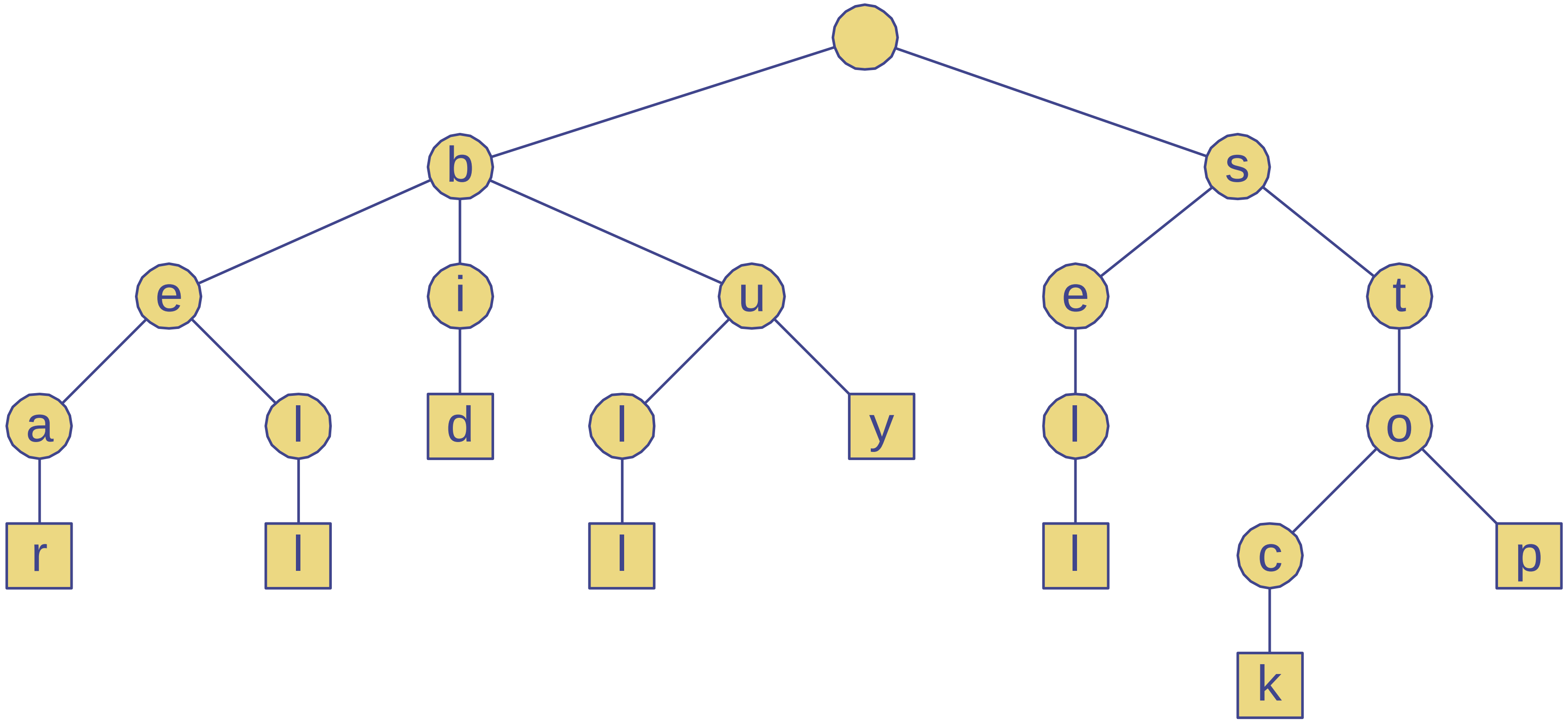
Standard Trie Example

- **For strings**
  - **{ a, an, and, any, at }**



Standard Trie Example

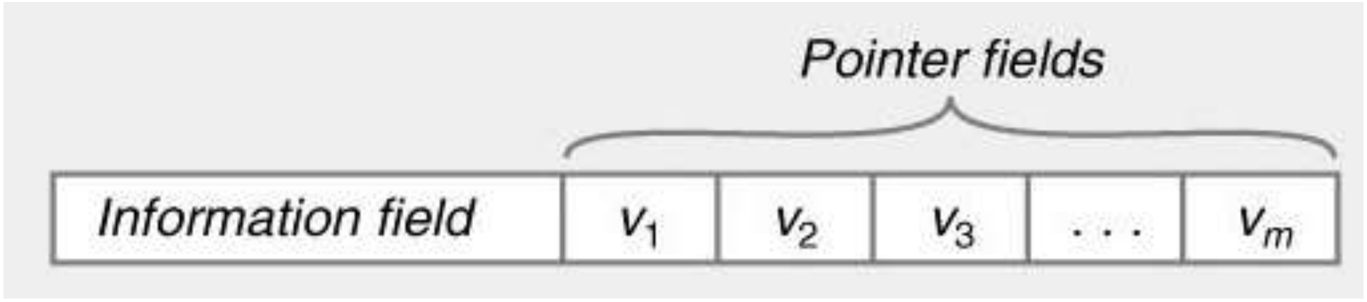
- For strings
  - { bear, bell, bid, bull, buy, sell, stock, stop }



Standard Tries

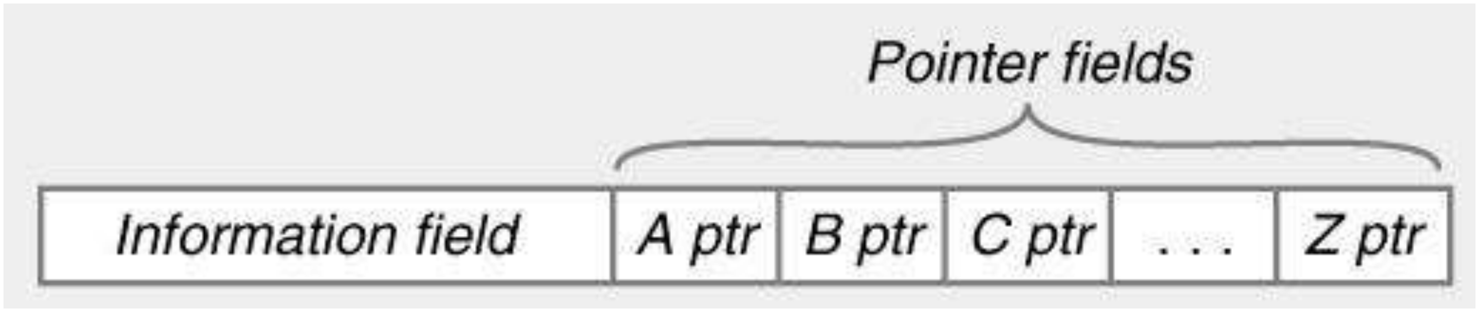
- Node structure
  - Value between 1…m
  - Reference to m children
    - Array or linked list
- Example

```
Class Node {
    Letter value;          // Letter V = { V1, V2, … Vm }
    Node child[ m ];
}
```



Standard Tries

- **Efficiency**
  - **Uses O(n) space**
  - **Supports search / insert / delete in O(d×m) time**
  - **For**
    - **n total size of strings indexed by trie**
    - **d length of the parameter string**
    - **m size of the alphabet**



Word Matching Trie

- **Insert words into trie**
- **Each leaf stores occurrences of word in the text**

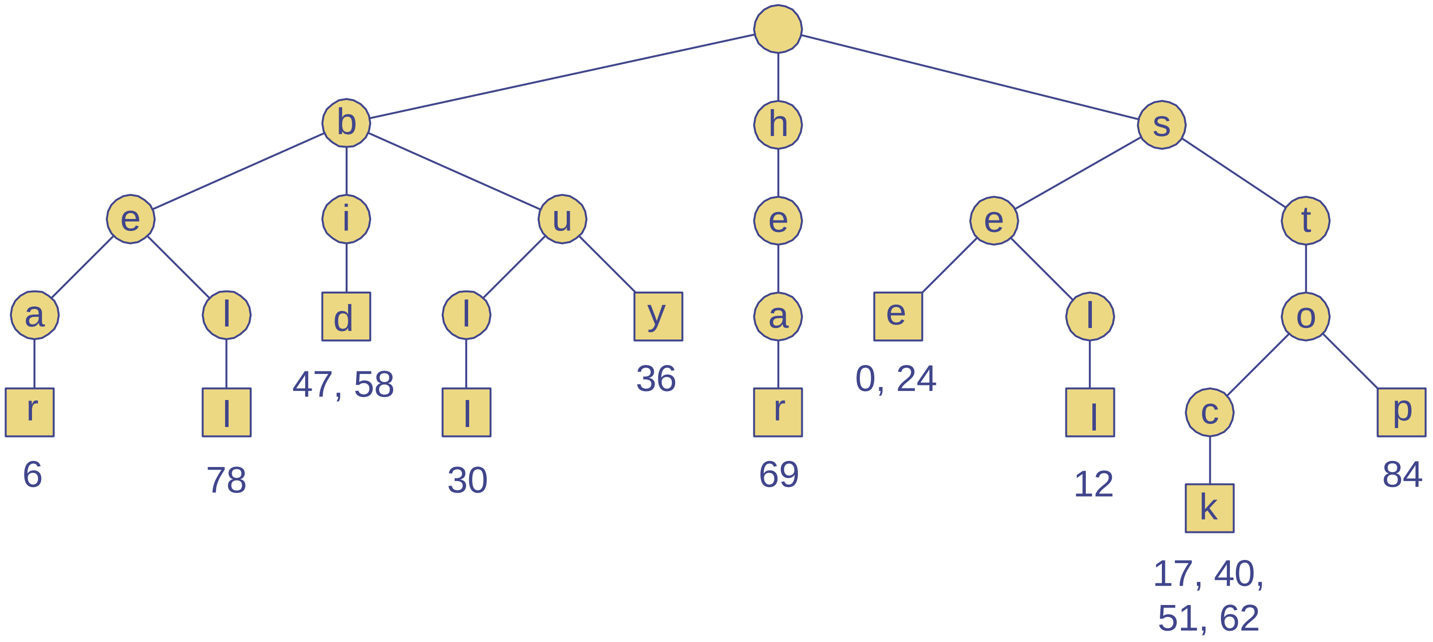
s	e	e		a		b	e	a	r	?		s	e	l	l		s	t	o	c	k	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

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

Deletion?

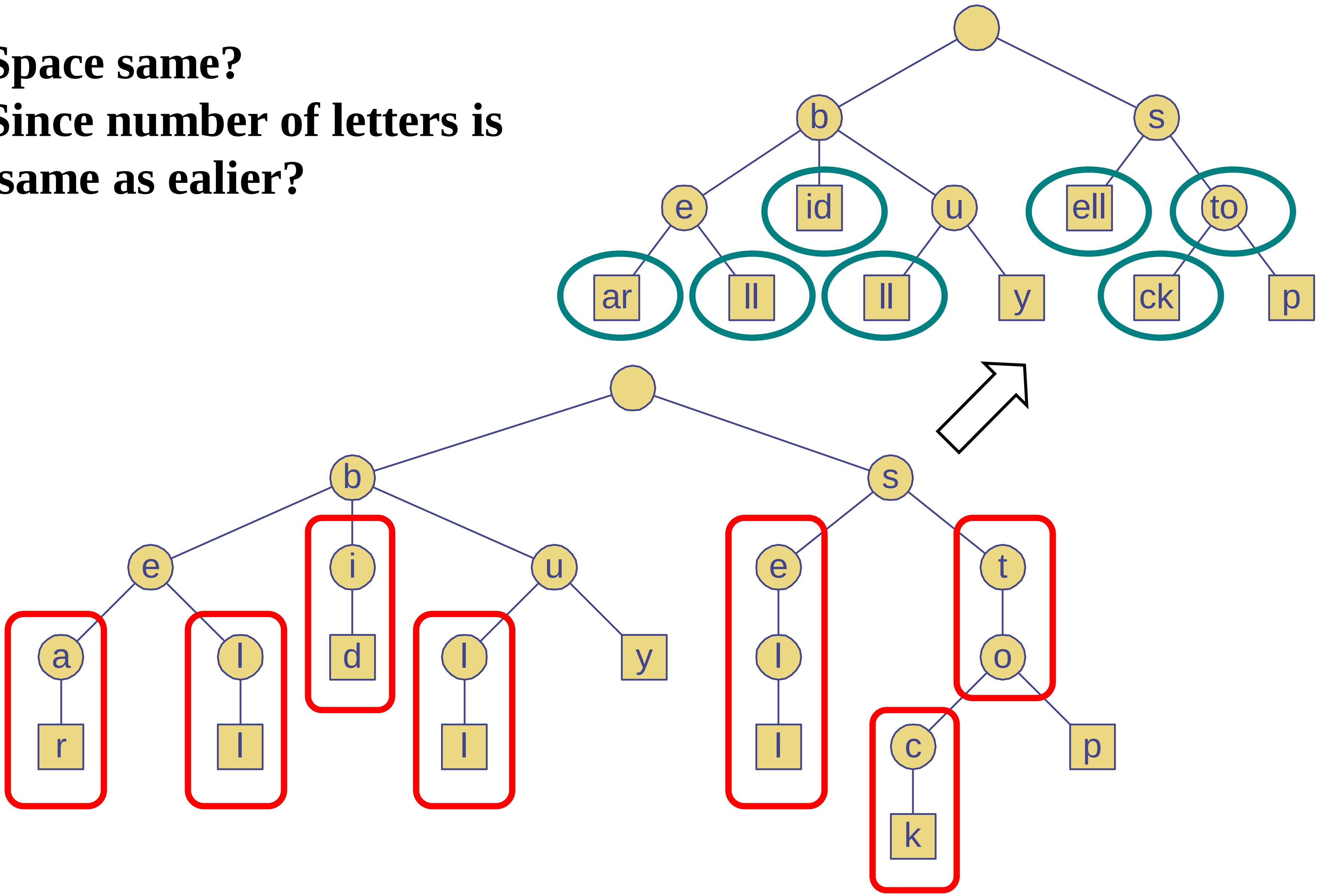


Compressed Trie

- Observation
  - Internal node v of T is redundant if v has one child and is not the root
- Approach
  - A chain of redundant nodes can be compressed
    - Replace chain with single node
    - Include concatenation of labels from chain
- Result
  - Internal nodes have at least 2 children
  - Some nodes have multiple characters

Compressed Trie

Space same?
Since number of letters is
same as ealier?



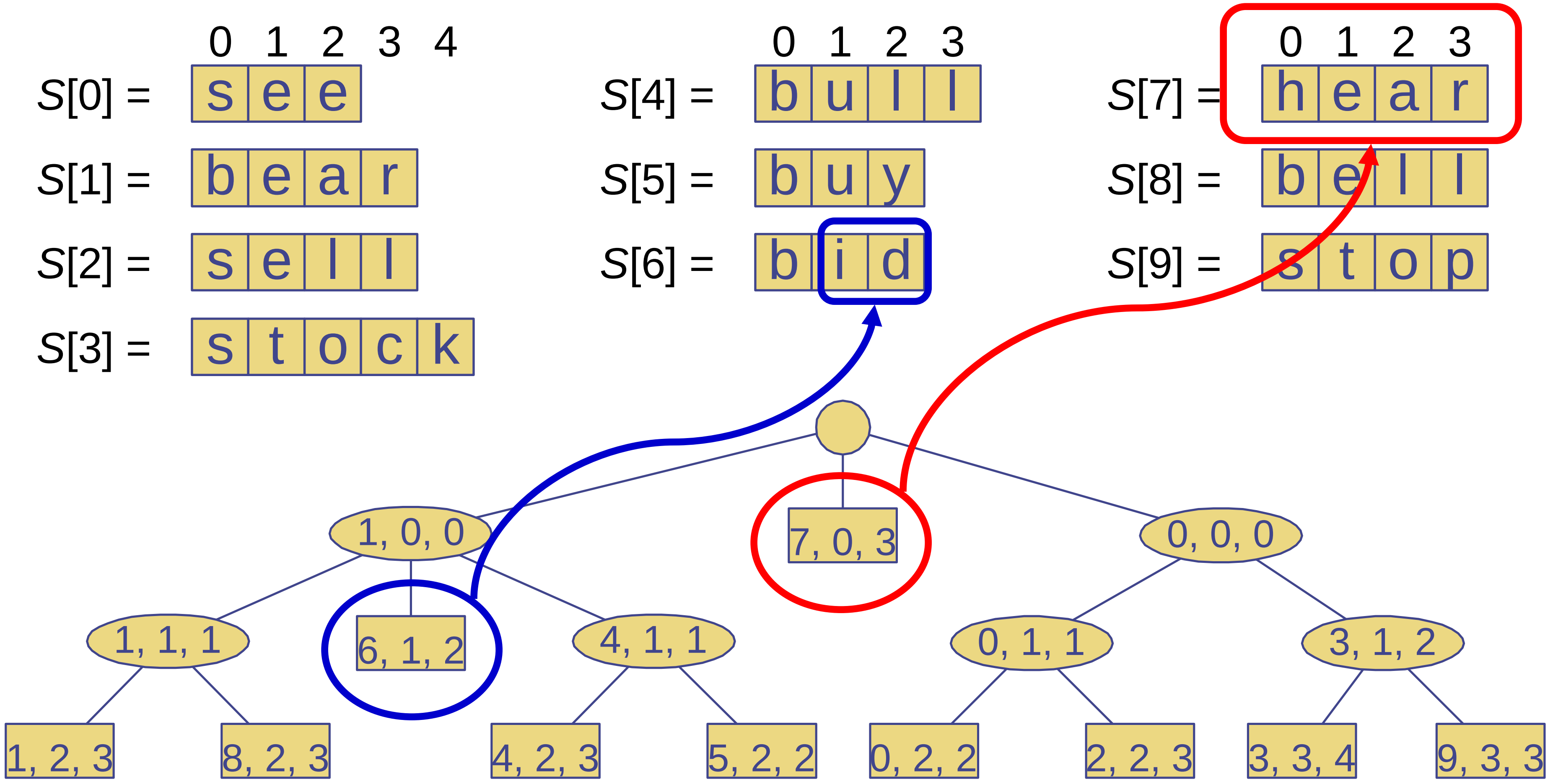
Compact Tries

- Compact representation of a compressed trie
- Approach
  - For an array of strings S = S[0], … S[s-1]
  - Store ranges of indices at each node
    - Instead of substring
  - Represent as a triplet of integers (i, j, k)
    - Such that X = s[i][j..k]
  - Example: S[0] = “abcd”, (0,1,2) = “bc”
- Properties
  - Uses O(s) space, where s = # of strings in the array
  - Serves as an auxiliary index structure

- **A tree with L leaf nodes in which every node has at least 2 children except the leaf nodes has atmost L-1 internal nodes.**

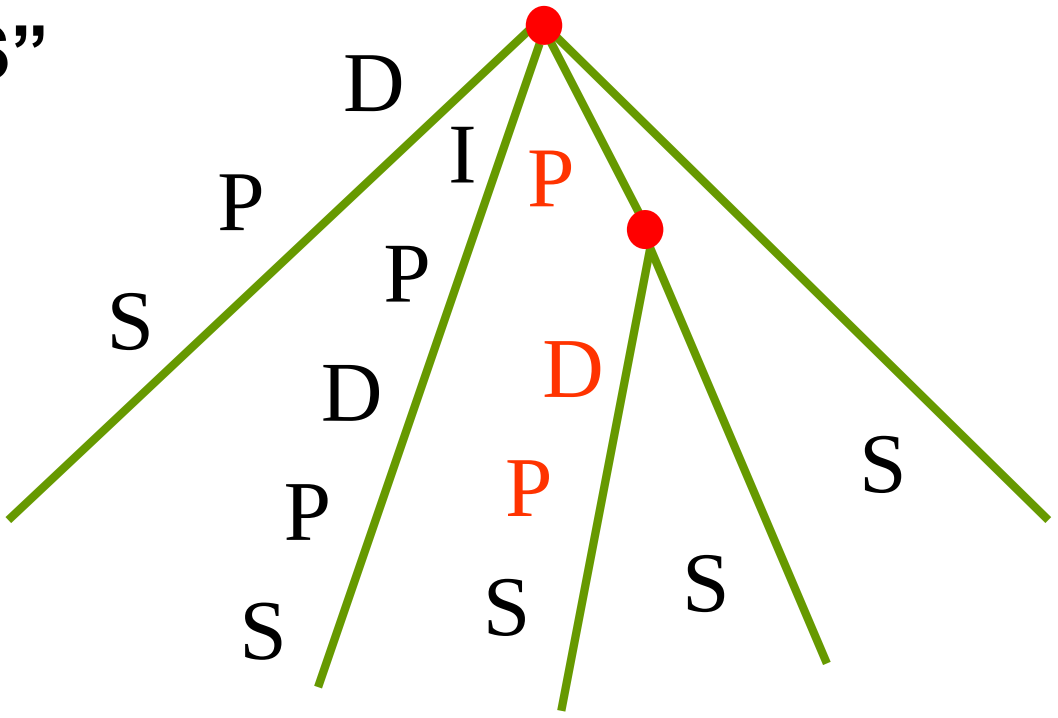
Compact Representation

- **Example**



Suffix Trie

- Compressed trie of all suffixes of text
- Example: “IPDPS”
  - Suffixes
    - IPDPS
    - PDPS
    - DPS
    - PS
    - S



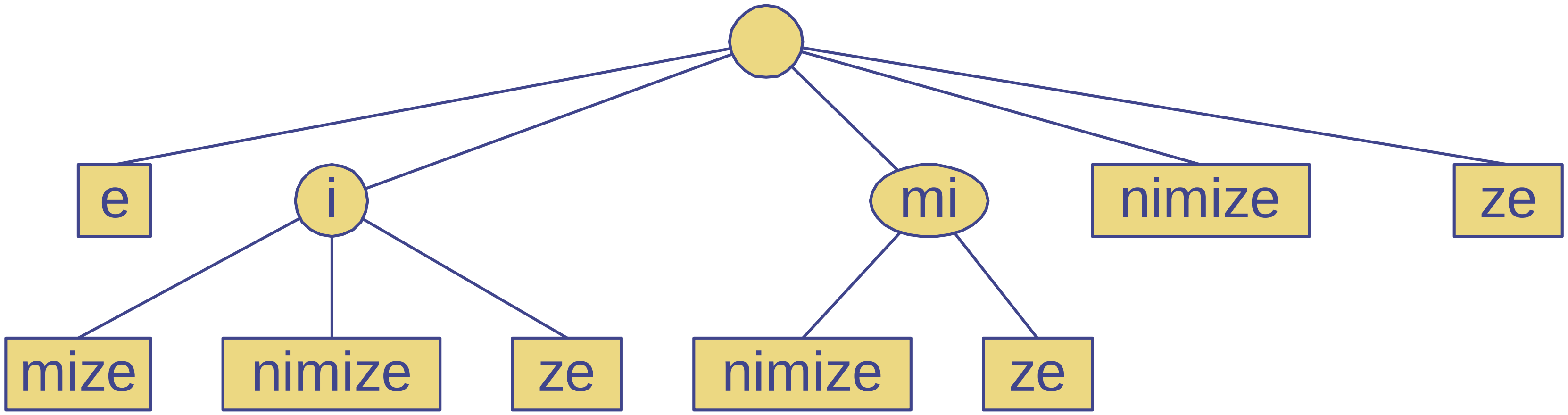
- Useful for finding pattern in any part of text
  - Occurrence ⇒ prefix of some suffix
  - Example: find PDP in IPDPS

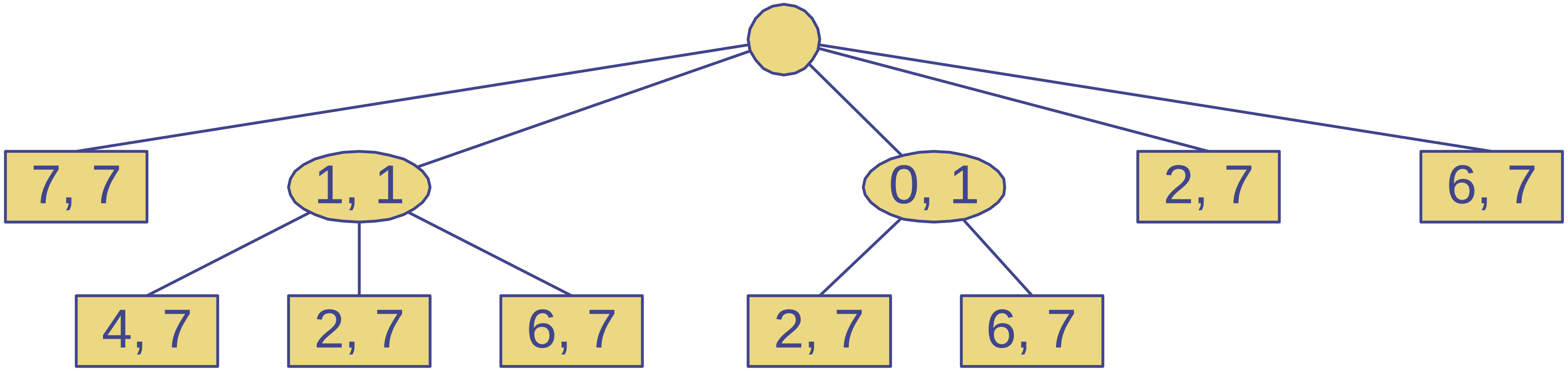
Suffix Trie

- **Properties**
  - **For**
    - **String X with length n**
    - **Alphabet of size m**
    - **Pattern P with length d**
  - **Uses O(n) space** (since we have O(n) leaves)
  - **Can be constructed in O(n) time**
  - **Find pattern P in X in O(d×m) time**
    - **Proportional to length of pattern, not text**

Suffix Trie Example

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





Tries and Web Search Engines

- Search engine index
  - Collection of all searchable words
  - Stored in compressed trie
- Each leaf of trie
  - Associated with a word
  - List of pages (URLs) containing that word
    - Called occurrence list
- Trie is kept in memory (fast)
- Occurrence lists kept in external memory
  - Ranked by relevance