

# Unidade VI:

## Árvore *Trie*

Prof. Max do Val Machado



PUC Minas

Instituto de Ciências Exatas e Informática  
Curso de Ciência da Computação

- As árvores *trie* são estruturas de dados para a procura rápida de padrões
- Elas são usadas em aplicações de pré-processamento do texto
- O nome *trie* é derivado da palavra *retrieval* (recuperação)

# Exemplos de Aplicações

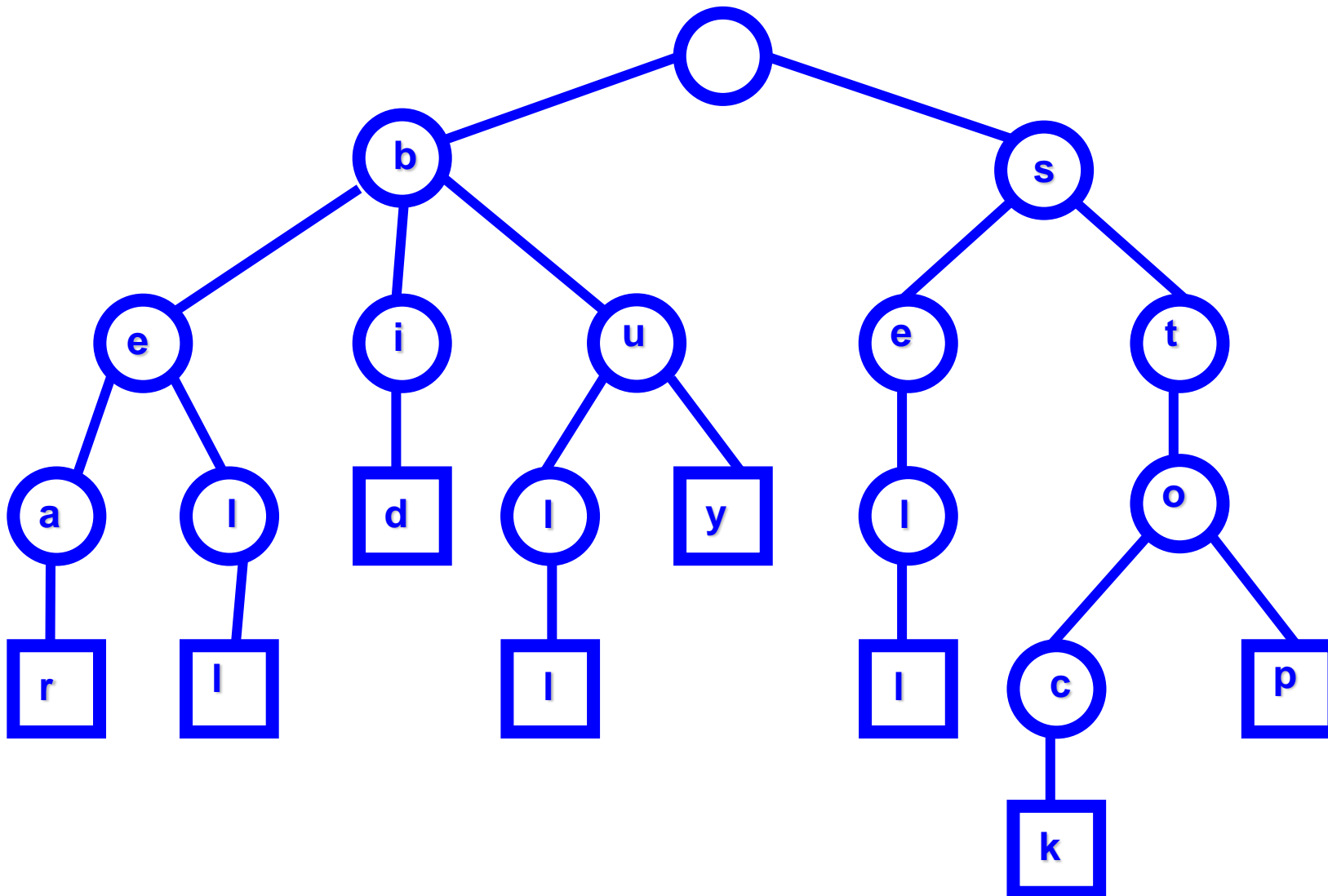
- Índices
- Armazenamento de Palavras (dicionários)
- Procura de uma sequência de DNA em uma base de genomas

# Definição

- Tem-se uma coleção de  $S$  cadeias de caracteres utilizando o mesmo alfabeto e as operações primárias suportadas são:
  - Procura de padrões
  - Procura de prefixos: Recebe-se uma cadeia  $X$  e retornam-se todas as cadeias que têm  $X$  como prefixo

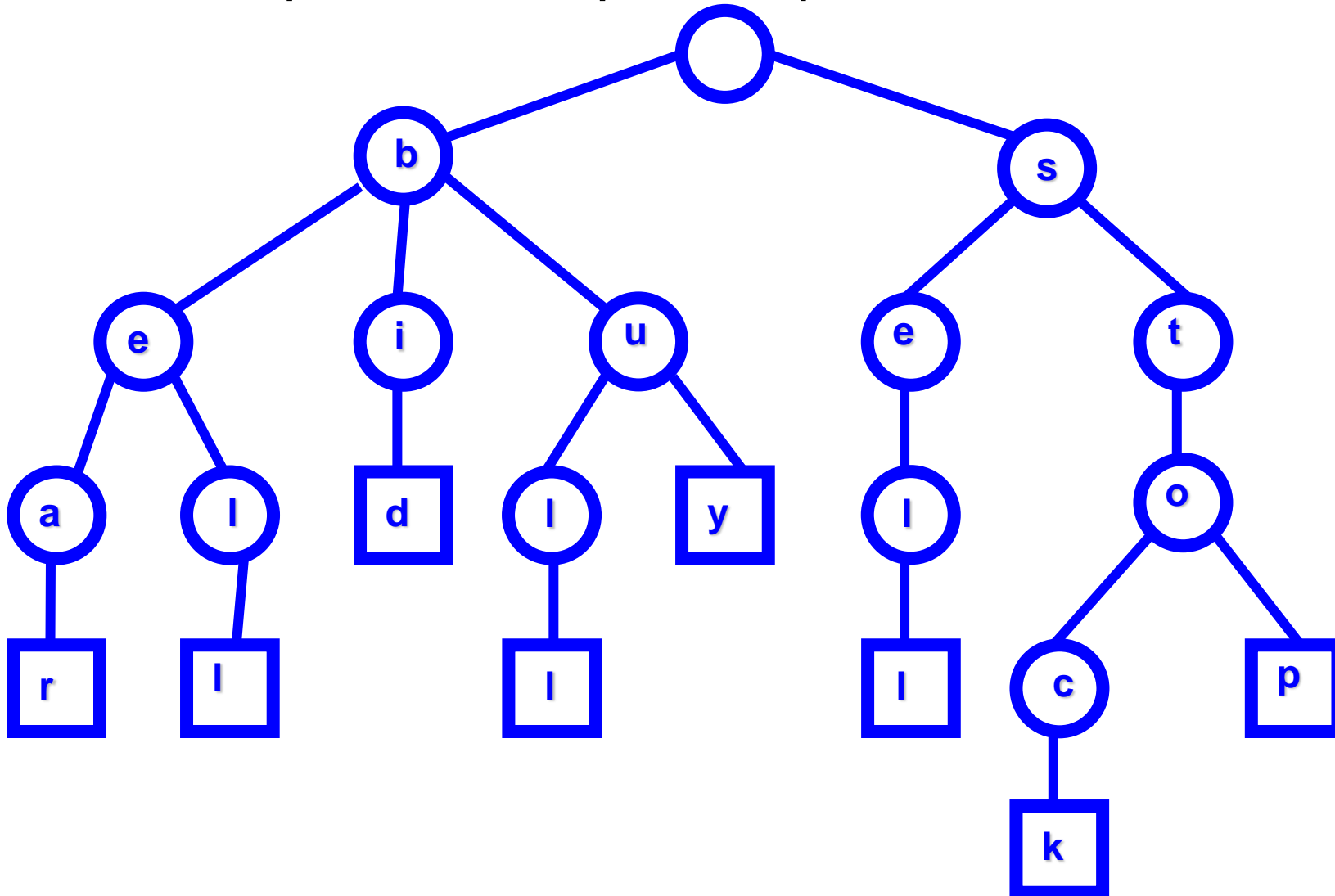
# Exemplo

*bear* - urso  
*bell* - sino  
*bid* - oferta  
*bull* - touro  
*buy* - compra  
*sell* - vende  
*stock* - ação  
*stop* - parar



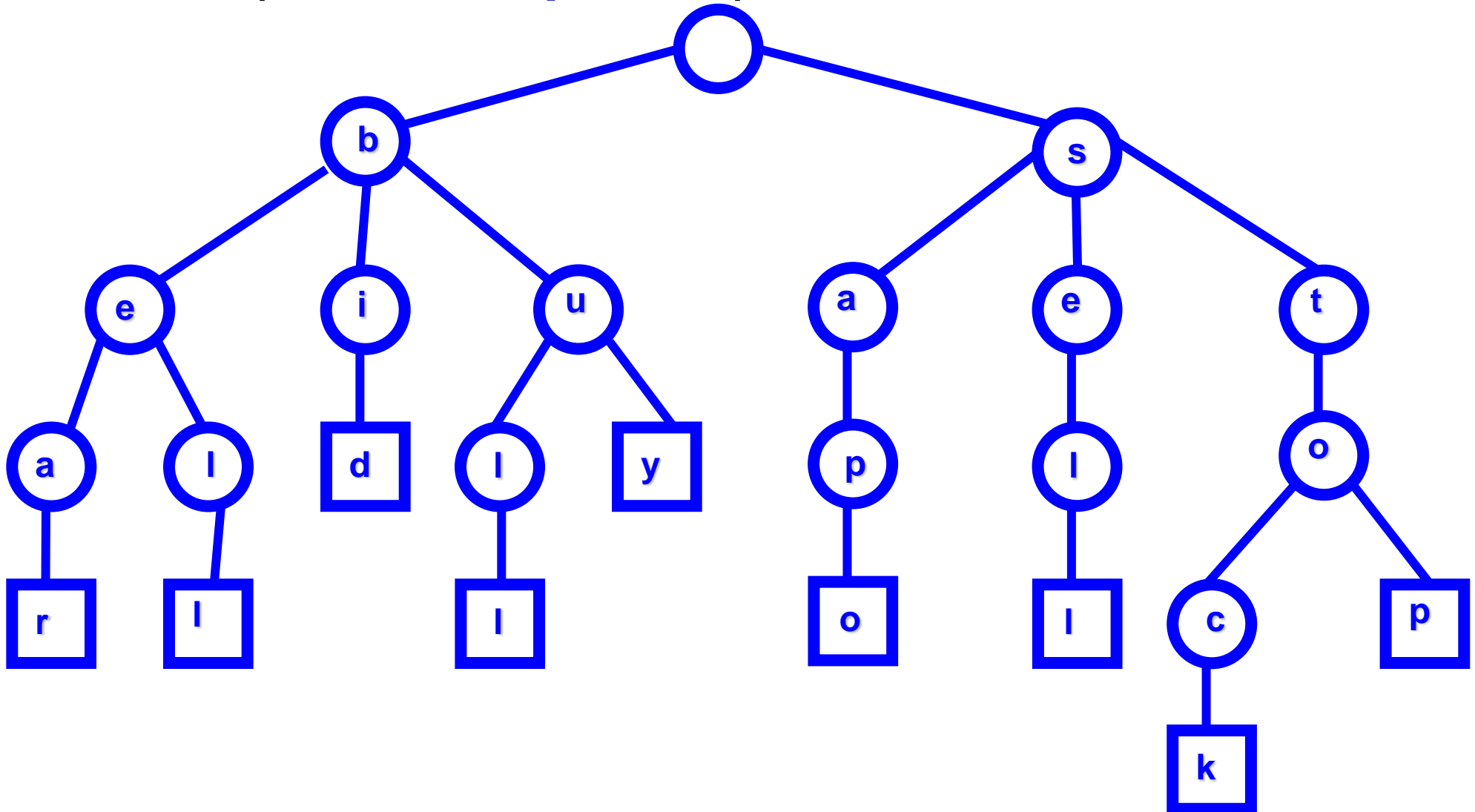
## Exercício

- Insira as palavras sapo e sapato na árvore abaixo



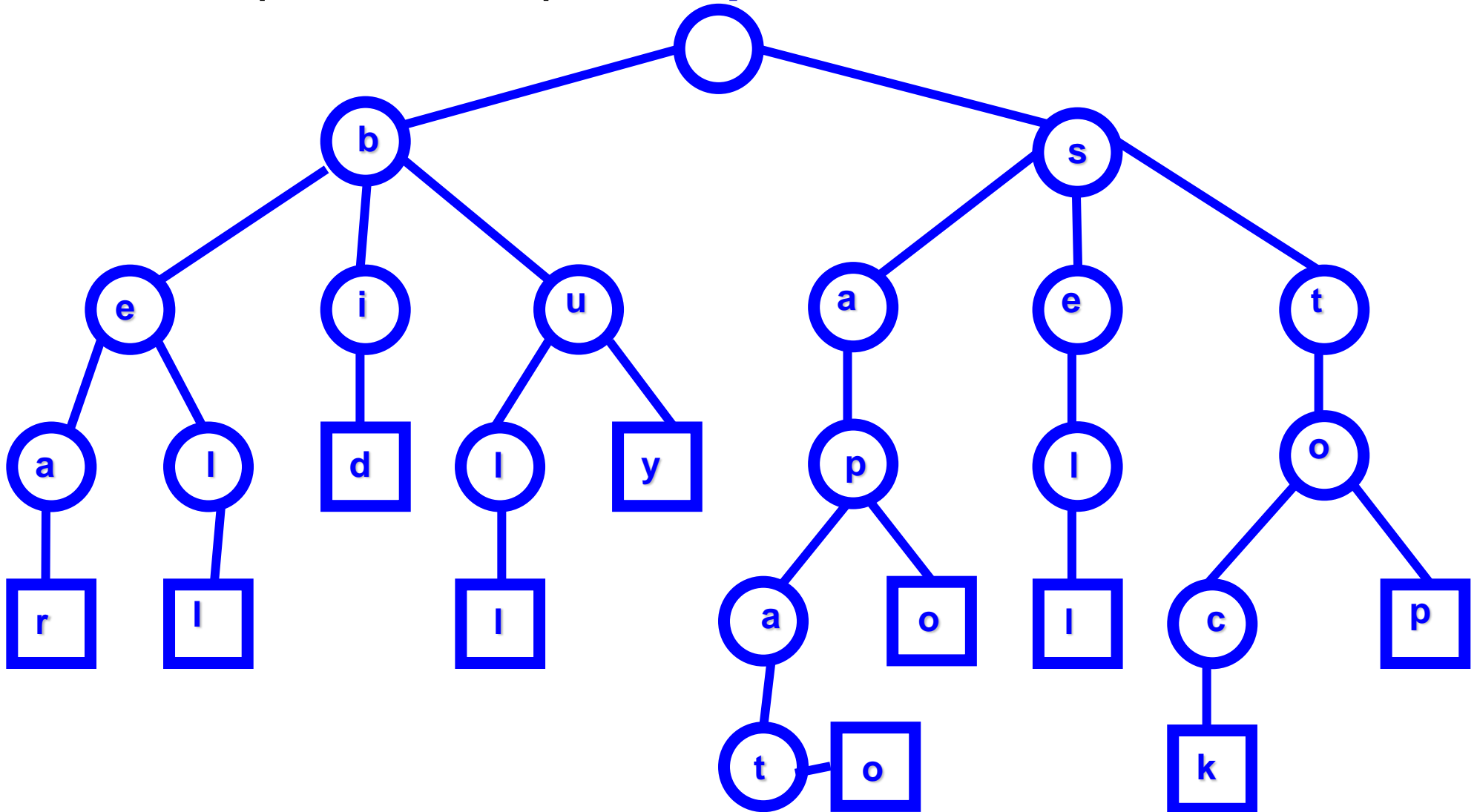
## Exercício

- Insira as palavras **sapo** e sapato na árvore abaixo



## Exercício

- Insira as palavras sapo e sapato na árvore abaixo





# Propriedades

- Nenhuma cadeia de  $S$  é prefixo de outra cadeia
- Cada nó (exceto a raiz) é rotulado com um caractere do  $\Sigma$
- A árvore tem  $s$  nós externos, um para cada cadeia de  $S$
- A concatenação dos rótulos em um caminho da raiz até um nó externo  $v$ , resulta na cadeia de  $S$  associada ao nó  $v$

# Propriedades

- Em geral, a *trie* é uma árvore múltipla (1...d filhos)
- Se o  $\Sigma$  tem tamanho d igual a 2, a *trie* será uma árvore binária
- Cada nó interno tem no máximo d filhos
- A altura da árvore é igual ao tamanho da maior cadeia em S

# Propriedades

- O número de nós é  $O(n)$  sendo  $n$  o comprimento total de  $S$
- O pior caso para o número de nós acontece quando não existe qualquer prefixo comum entre as cadeias, fazendo com que todos os nós internos (exceto a raiz) tenham um filho

# Pesquisar por uma Cadeia de Caracteres

- A partir da raiz, verificamos caractere a caractere se existe um caminho na árvore correspondendo à cadeia desejada - por definição, um caminho sempre termina em um nó externo
- Se cada nó tiver uma tabela hash perfeita para endereçar seus filhos, o tempo de pesquisa é  $O(m)$  onde  $m$  é o tamanho da cadeia a ser procurada

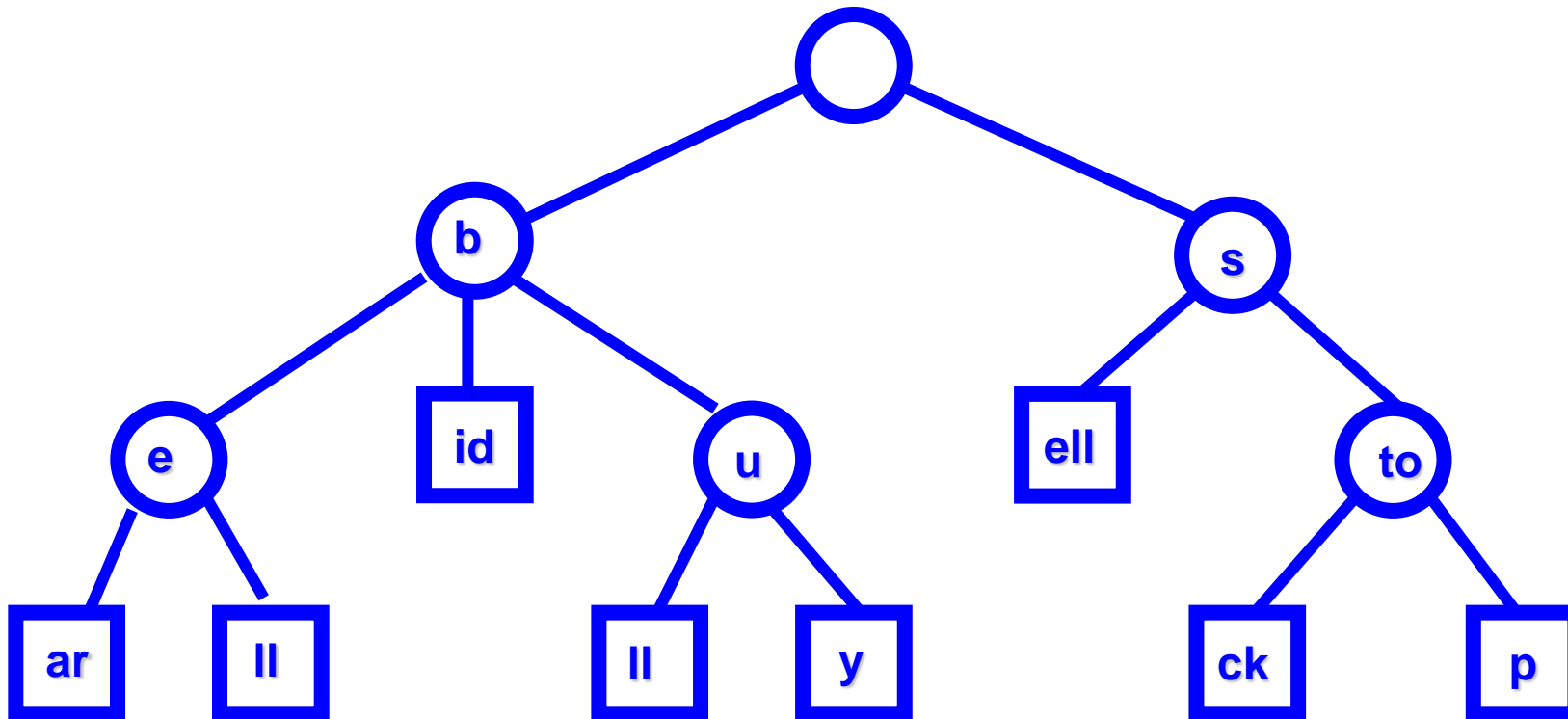
# Inserção de uma Cadeia de Caracteres

- Caminhamos na *trie* casando cada caractere da nova cadeia
- Quando não existe um nó para um caractere, criamos o nó e repetimos esse passo para os demais caracteres da cadeia
- Lembrando que nenhuma cadeia é prefixo de outra
- O tempo de inserção é  $O(m)$  e a construção total da árvore é  $O(n)$ , onde  $n = |S|$

- *Practical Algorithm To Retrieve Information Coded In Alphanumeric (PATRICIA)*
- O fato dos nós da *trie-padrão* terem apenas um filho faz com que as trie padrão sejam ineficientes em termos de espaço
- A Trie-Patricia elimina os nós redundantes fazendo com que todos os nós (exceto a raiz) tenham pelo menos dois filhos

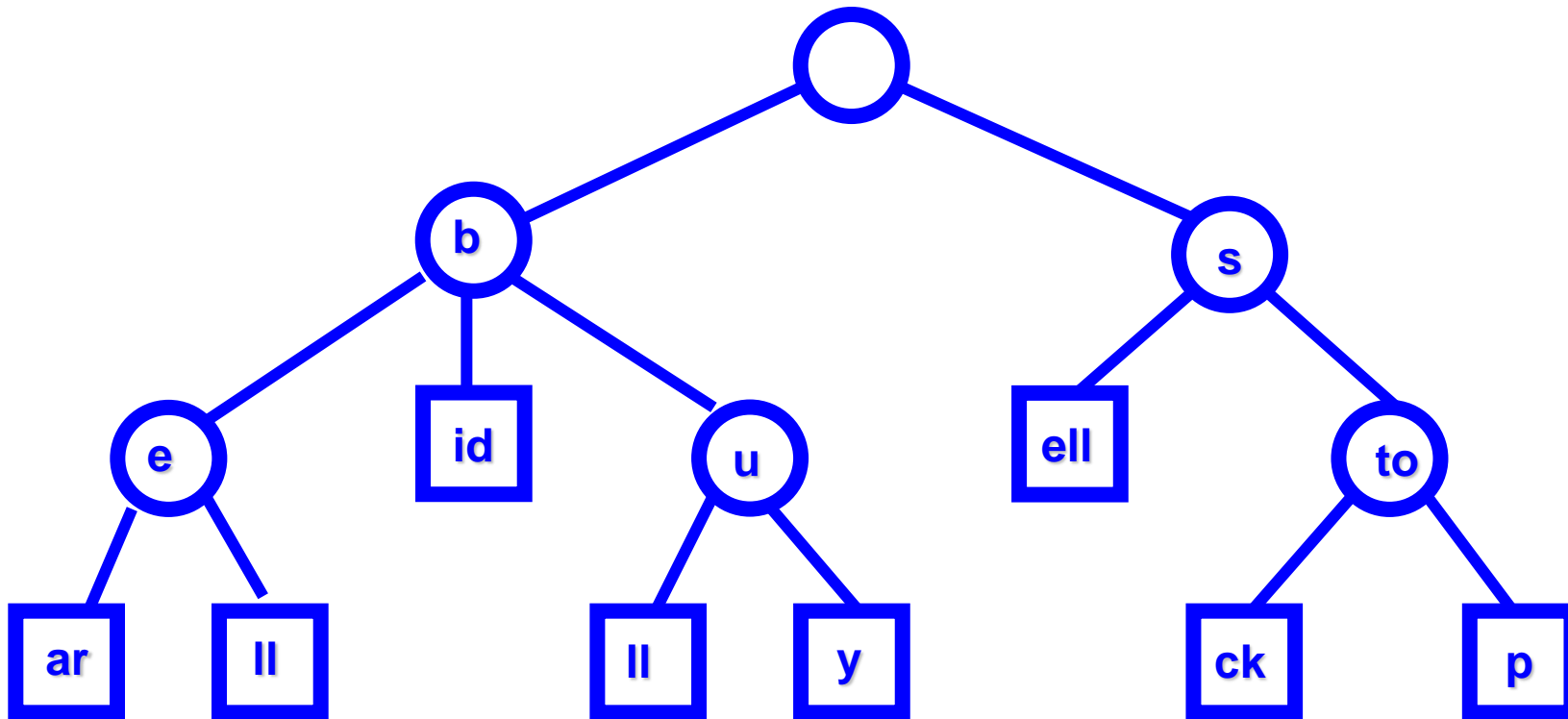
# Exemplo de *Trie Patricia*

*bear* - urso  
*bell* - sino  
*bid* - oferta  
*bull* - touro  
*buy* - compra  
*sell* - vende  
*stock* - ação  
*stop* - parar



## Exercício

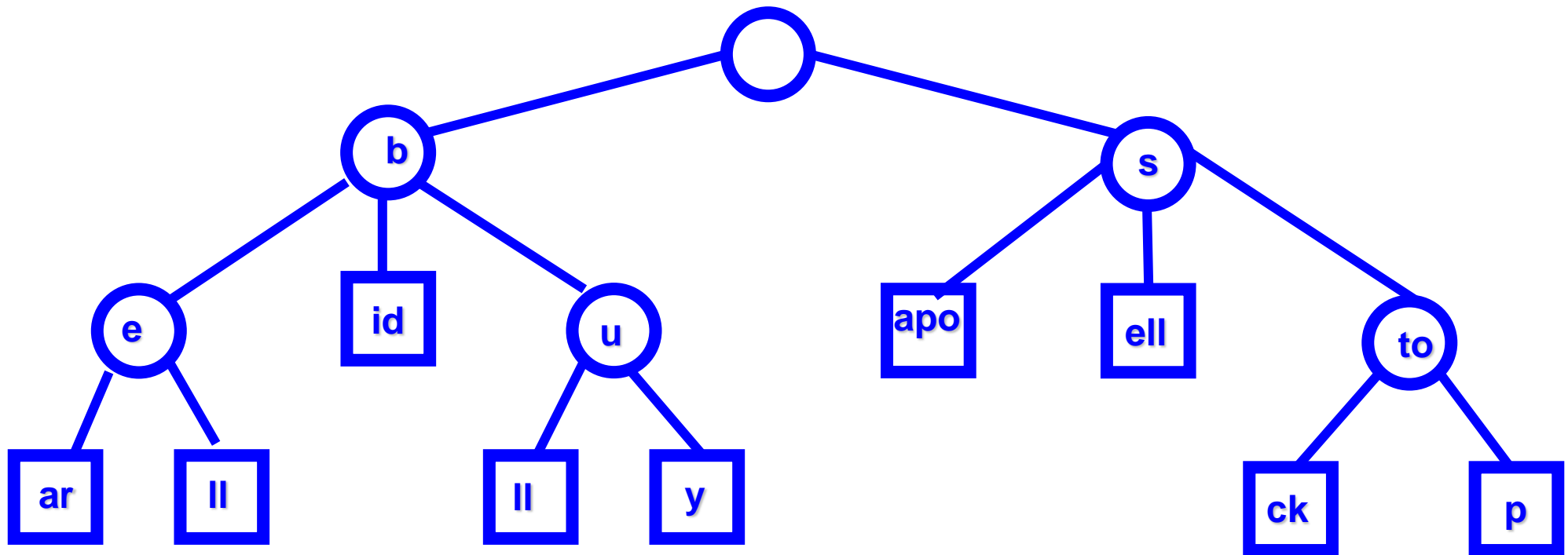
- Insira as palavras sapo e sapato na árvore abaixo





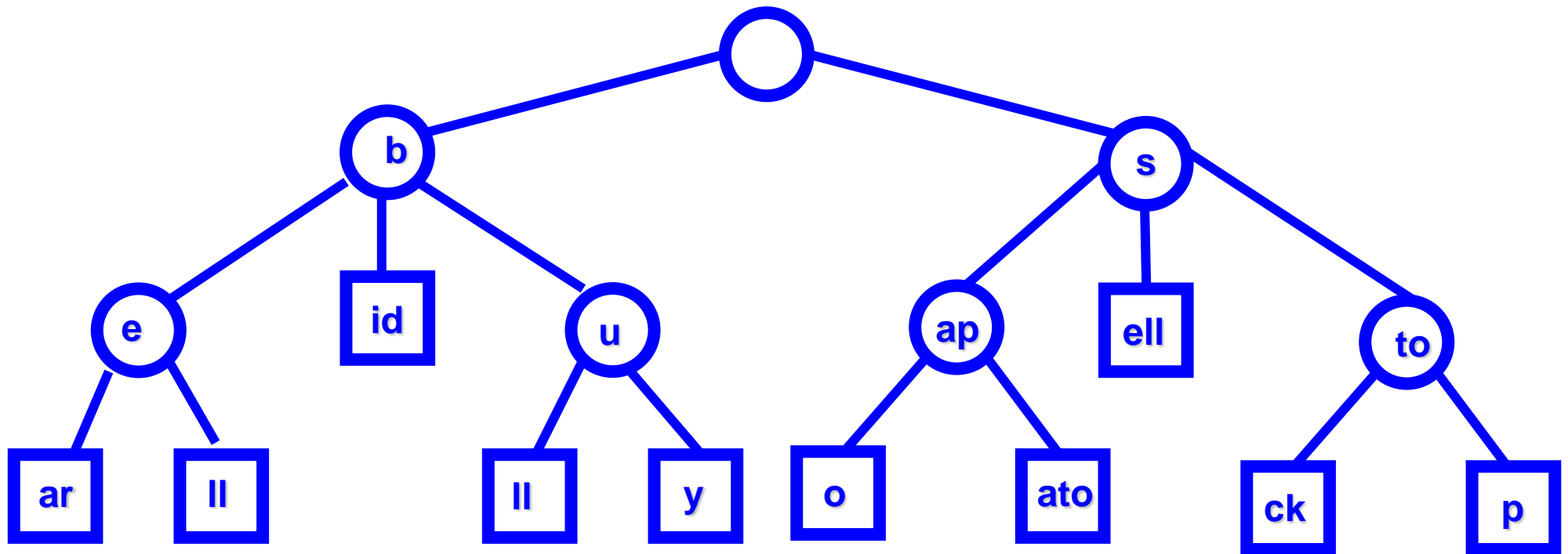
## Exercício

- Insira as palavras **sapo** e sapato na árvore abaixo



## Exercício

- Insira as palavras sapo e sapato na árvore abaixo



# Propriedades das *Trie Patricia*

- Os nós são rotulados por substrings das cadeias de caracteres pertencentes a  $S$
- O número de nós é proporcional ao número de cadeias existentes em  $S$  e não ao comprimento das mesmas

# Propriedades das *Trie Patricia*

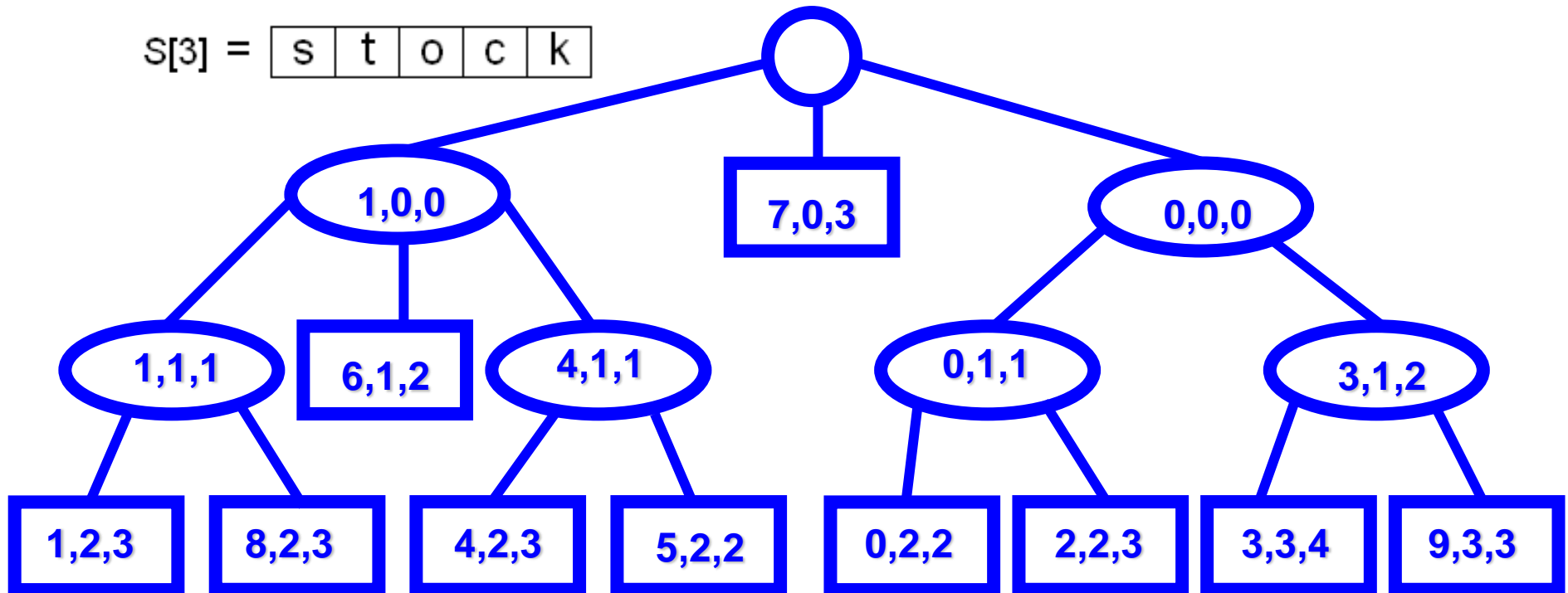
- Todo nó interno de  $T$  tem pelo menos 2 e no máximo  $d$  filhos
- $T$  tem  $S$  nós externos
- O número de nós de  $T$  é  $O(s)$ , onde  $s$  é o número de cadeias de  $S$

# Estrutura de Dados da *Trie Patricia*

- Cada nó armazena uma tripla de inteiros  $(i, j, k)$ , indicando o rótulo do nó de tal forma que  $S[i][j...k]$ , onde:
  - A coleção de cadeias  $S$  será  $S[0], S[1], \dots, S[s-1]$
  - $j$  e  $k$  representam, respectivamente, a primeira e última (inclusive) posições da cadeia  $S[i]$  que correspondem ao rótulo corrente

# Exemplo da Estrutura de Dados da *Trie Patricia*

0 1 2 3 4	0 1 2 3	0 1 2 3											
s[0] = <table border="1"><tr><td>s</td><td>e</td><td>e</td></tr></table>	s	e	e	s[4] = <table border="1"><tr><td>b</td><td>u</td><td>l</td><td>l</td></tr></table>	b	u	l	l	s[7] = <table border="1"><tr><td>h</td><td>e</td><td>a</td><td>r</td></tr></table>	h	e	a	r
s	e	e											
b	u	l	l										
h	e	a	r										
s[1] = <table border="1"><tr><td>b</td><td>e</td><td>a</td><td>r</td></tr></table>	b	e	a	r	s[5] = <table border="1"><tr><td>b</td><td>u</td><td>y</td></tr></table>	b	u	y	s[8] = <table border="1"><tr><td>b</td><td>e</td><td>l</td><td>l</td></tr></table>	b	e	l	l
b	e	a	r										
b	u	y											
b	e	l	l										
s[2] = <table border="1"><tr><td>s</td><td>e</td><td>l</td><td>l</td></tr></table>	s	e	l	l	s[6] = <table border="1"><tr><td>b</td><td>i</td><td>d</td></tr></table>	b	i	d	s[9] = <table border="1"><tr><td>s</td><td>t</td><td>o</td><td>p</td></tr></table>	s	t	o	p
s	e	l	l										
b	i	d											
s	t	o	p										
s[3] = <table border="1"><tr><td>s</td><td>t</td><td>o</td><td>c</td><td>k</td></tr></table>	s	t	o	c	k								
s	t	o	c	k									



# Exercício

- Insira as palavras sapo e sapato na árvore abaixo

0 1 2 3 4

0 1 2 3

0 1 2 3

s[0] = s e e

s[4] = b u l l

s[7] = h e a r

s[1] = b e a r

s[5] = b u y

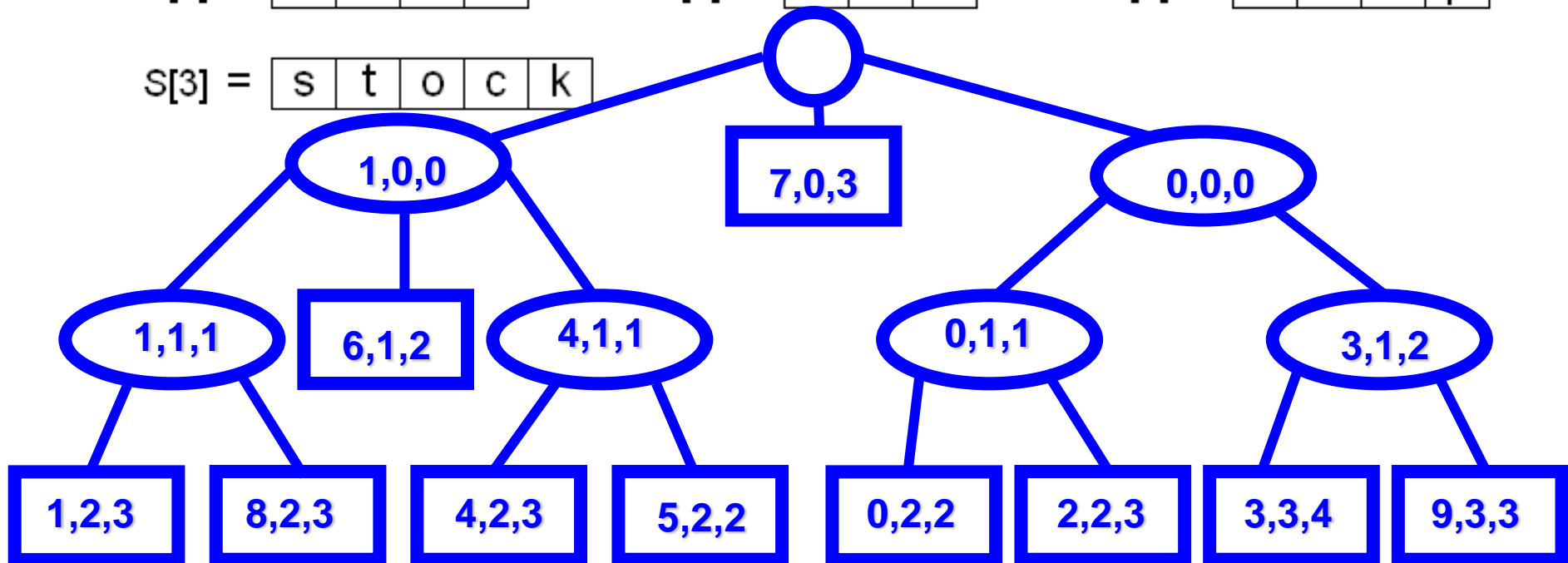
s[8] = b e l l

s[2] = s e l l

s[6] = b i d

s[9] = s t o p

s[3] = s t o c k



Análise da *Trie Patricia*

- Na trie-padrão, o espaço era  $O(n)$  e na trie-Patricia, é  $O(s)$
- Isso, descontando o espaço para armazenamento de  $S$ , contudo, o ganho do trie-Patricia é maior



# Referências

- GOODRICH, M. e TAMASSIA, R.; "Estrutura de Dados e Algoritmos em Java"; trad. Bernado Copstein e João Batista Oliveira; 2 Ed; Porto Alegre; Bookman; 2002
- SEDGEWICK, R.; "Algorithms in C++"; Addison-Wesley; 2nd edition ; 1988.