Unidade X:
Árvores TRIE

Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

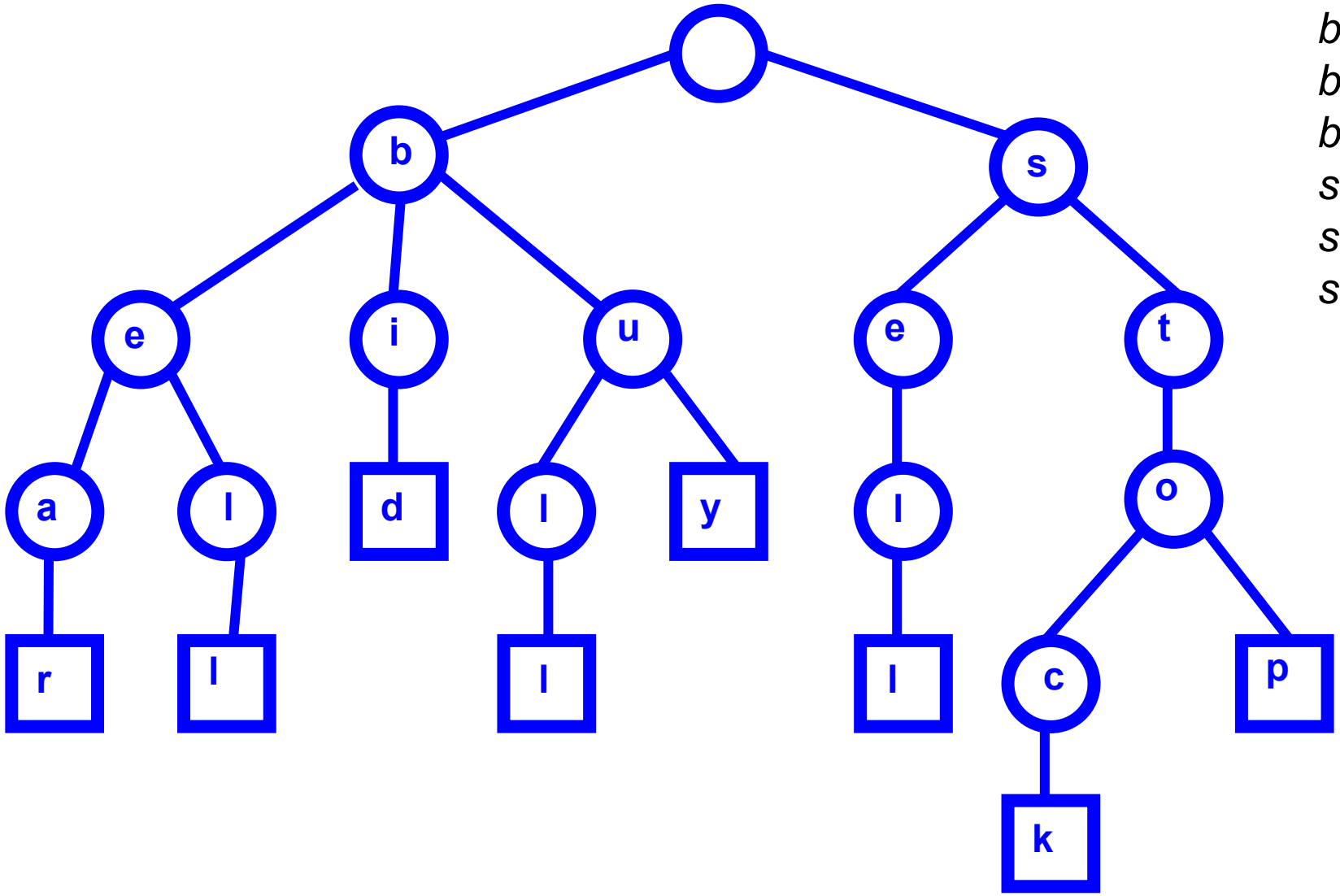
Árvores *Trie*

- Estruturas de dados para a procura rápida de padrões
- Usadas em aplicações de pré-processamento do texto
- Nome derivado da palavra *retrieval* (recuperação)
- Aplicadas, por exemplo, em: índices, armazenamento de palavras (dicionários) e busca 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

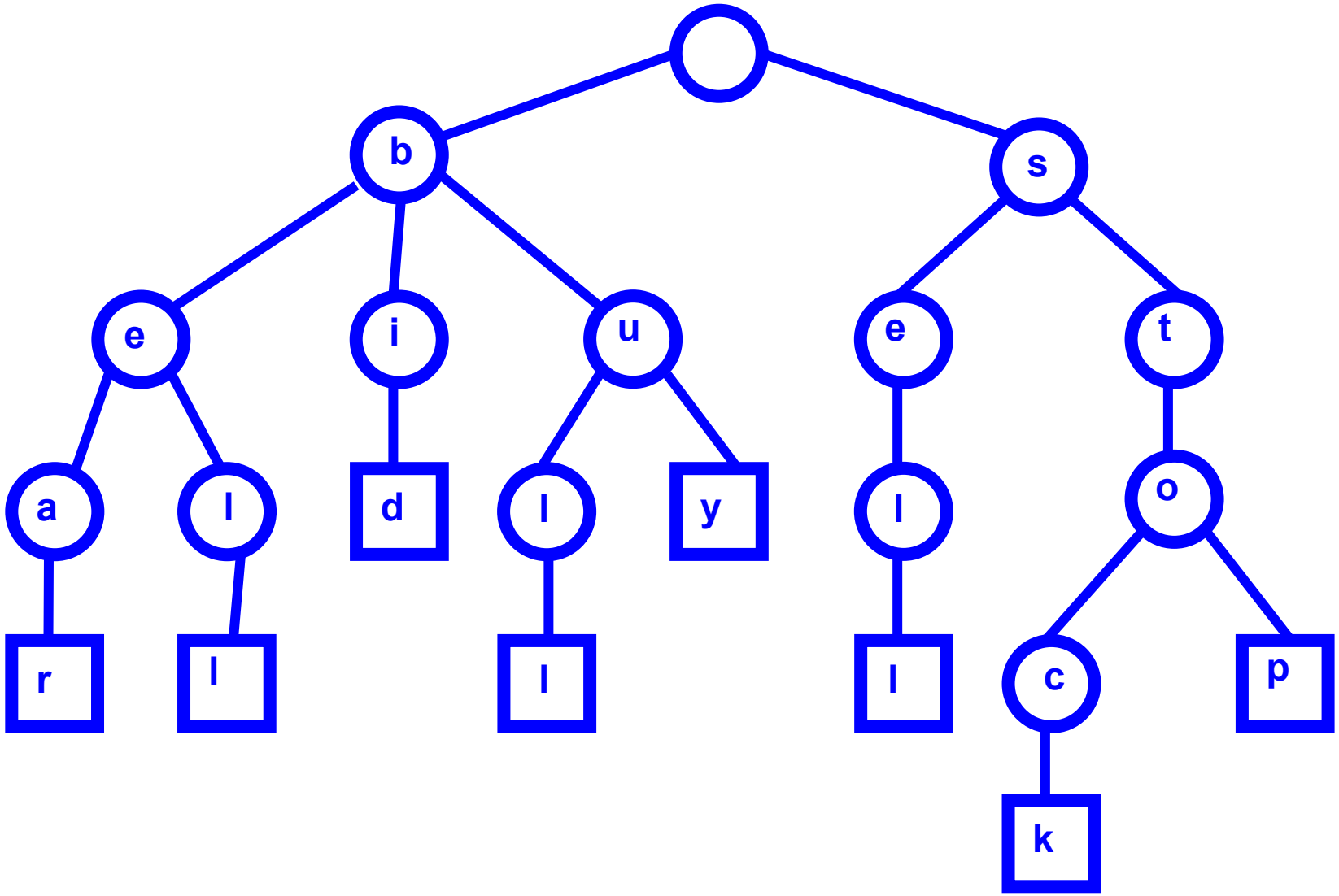
Exemplo



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

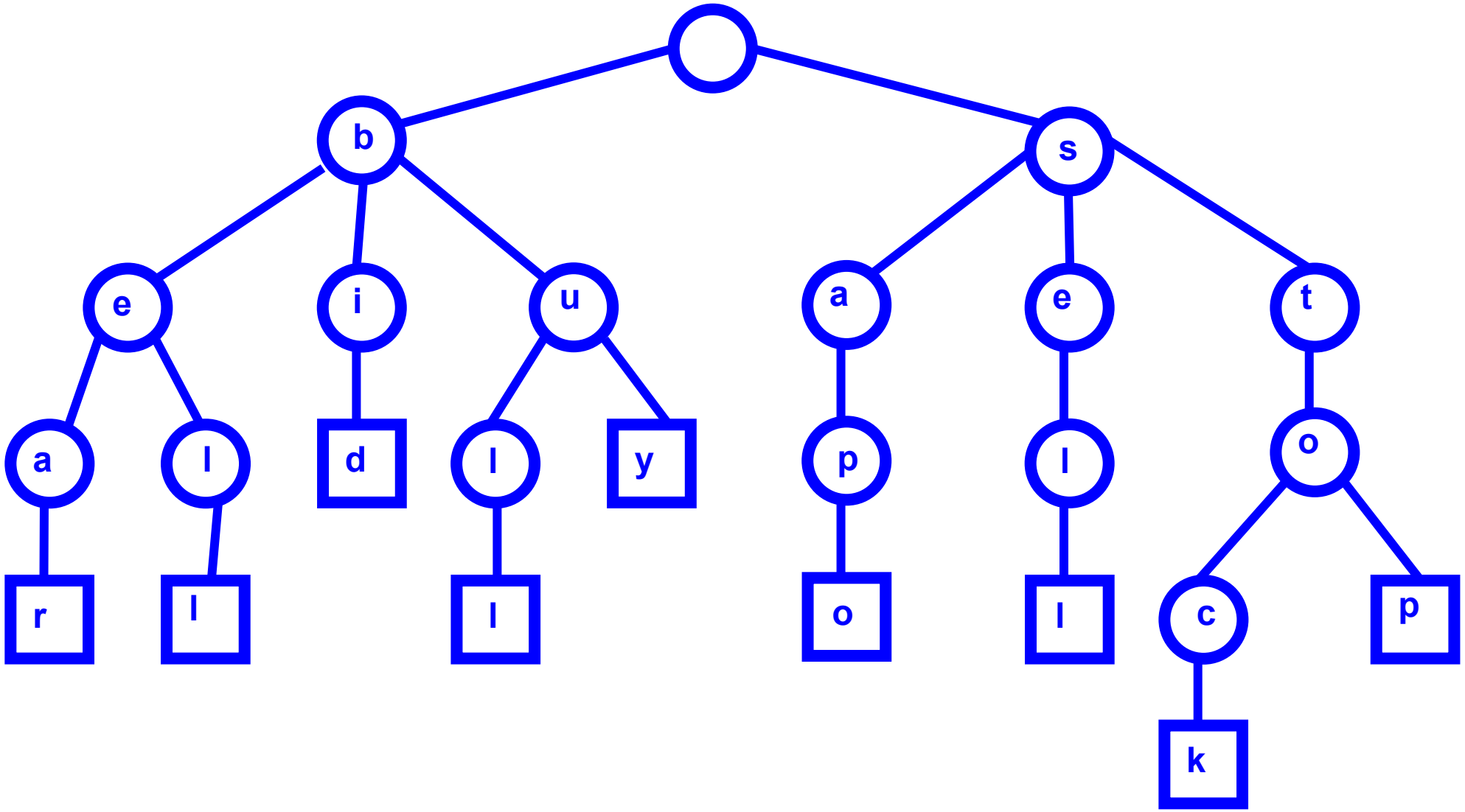
Exercício Resolvido (1)

- Insira as palavras sapo e sapato na árvore abaixo



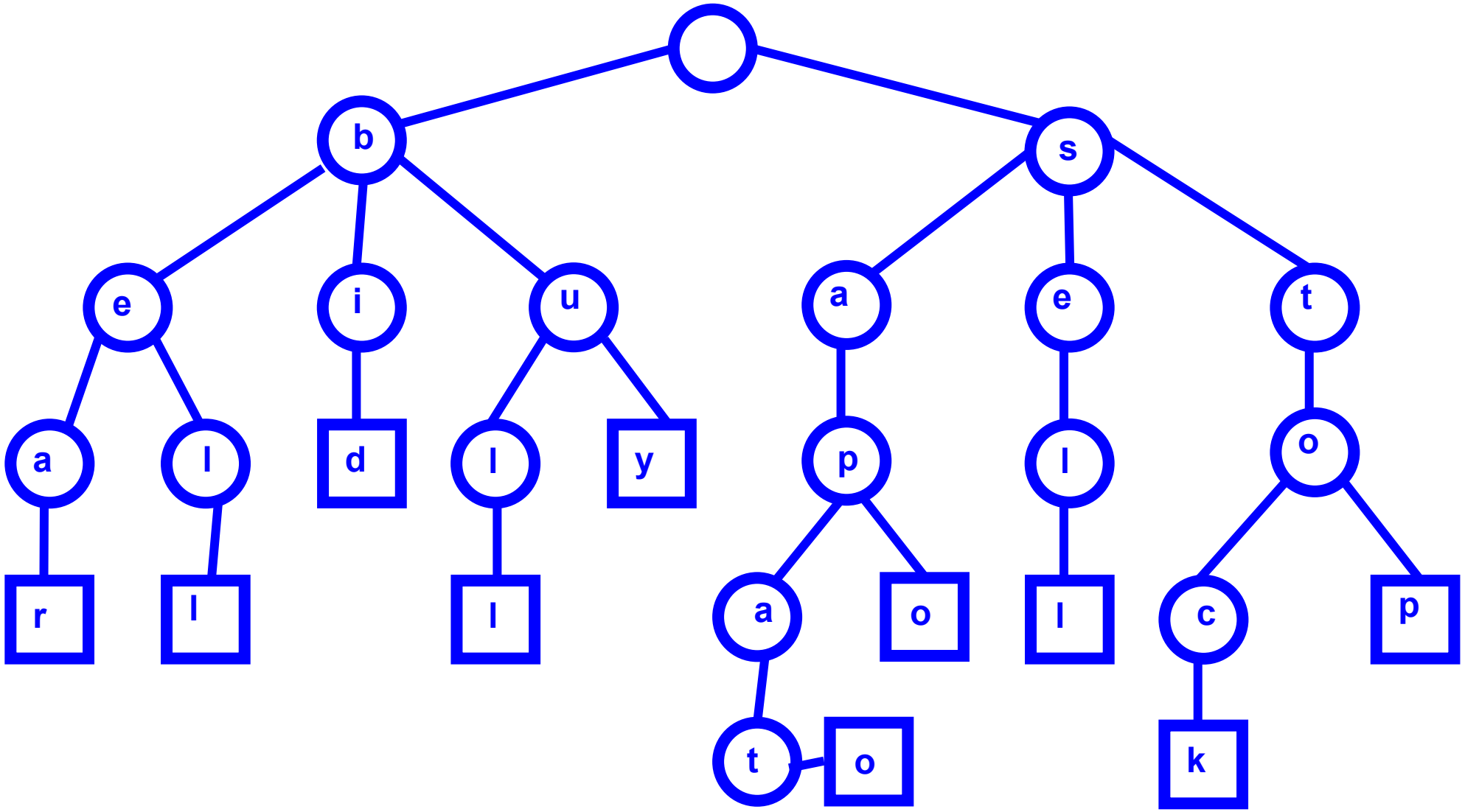
Exercício Resolvido (1)

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



Exercício Resolvido (1)

- 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 Σ
- A árvore tem s folhas, um para cada cadeia de S
- A concatenação dos rótulos em um caminho da raiz até uma folha, resulta na cadeia de S associada a essa folha

Propriedades

- Em geral, a *trie* é uma árvore múltipla (0...d filhos)
- Se o Σ 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 $\Theta(n)$ 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

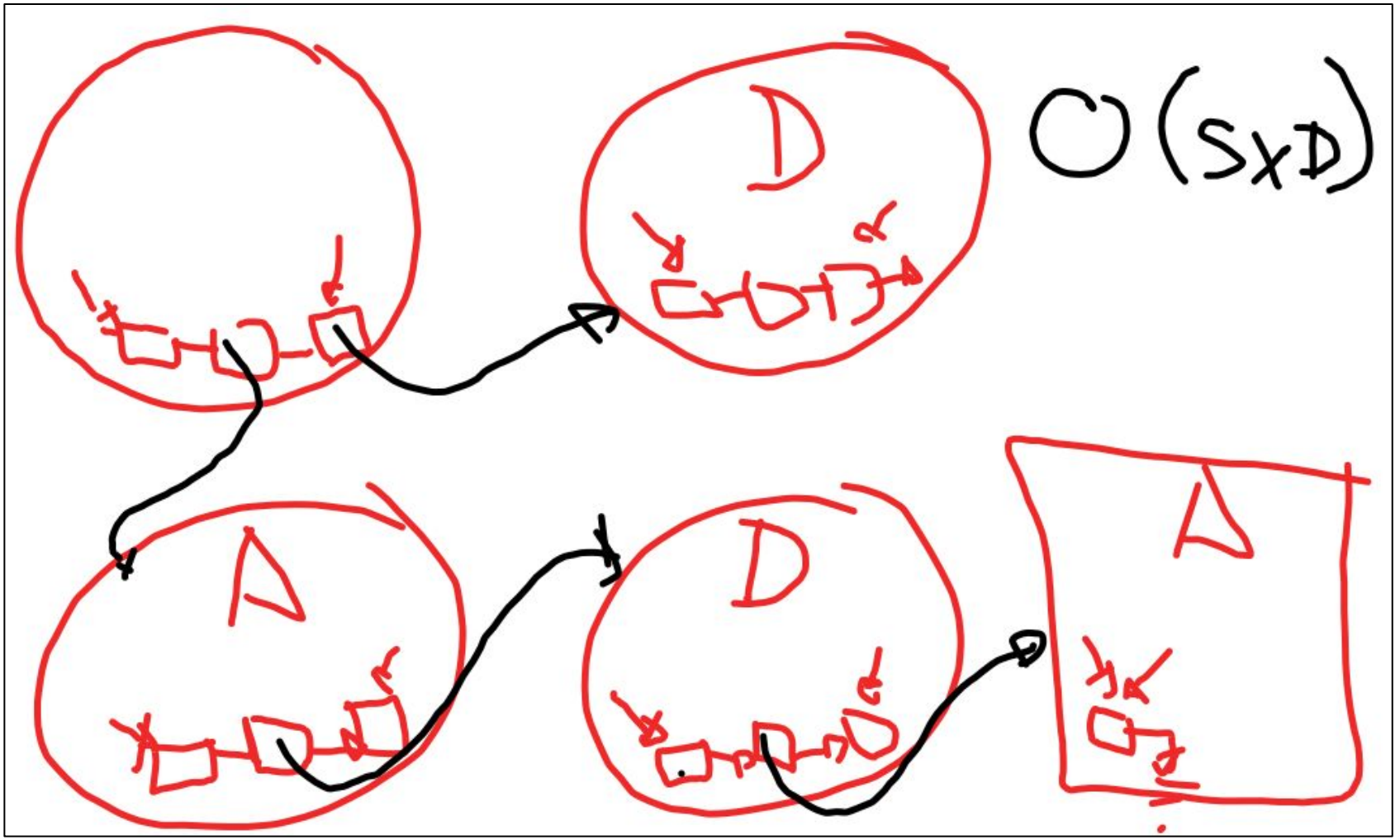
- Por definição, um caminho sempre termina em uma folha

Exercício Resolvido (2)

- Implemente a Classe Nó da Árvore Trie

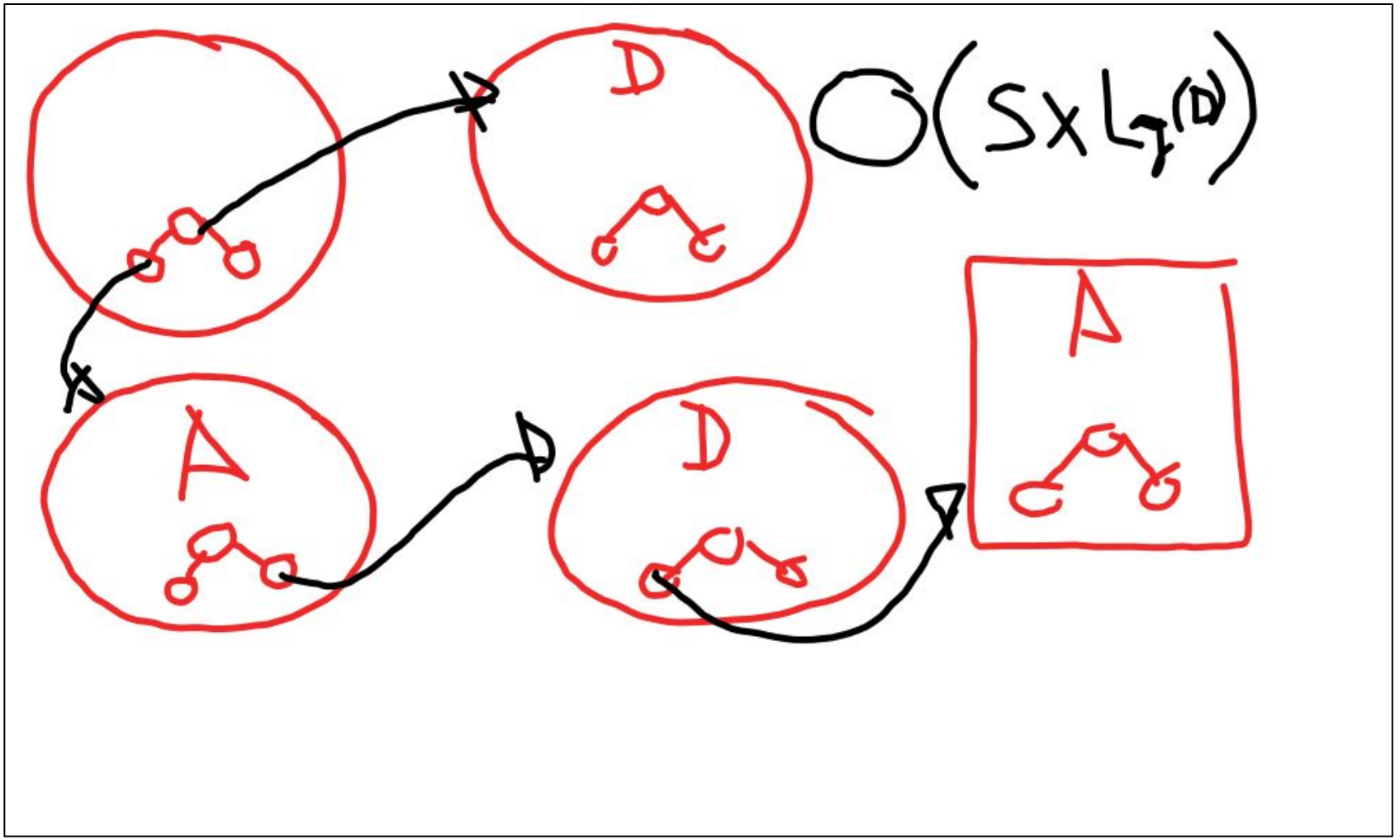
Exercício Resolvido (2)

- Implemente a Classe Nó da Árvore Trie **(Lista Flexível)**



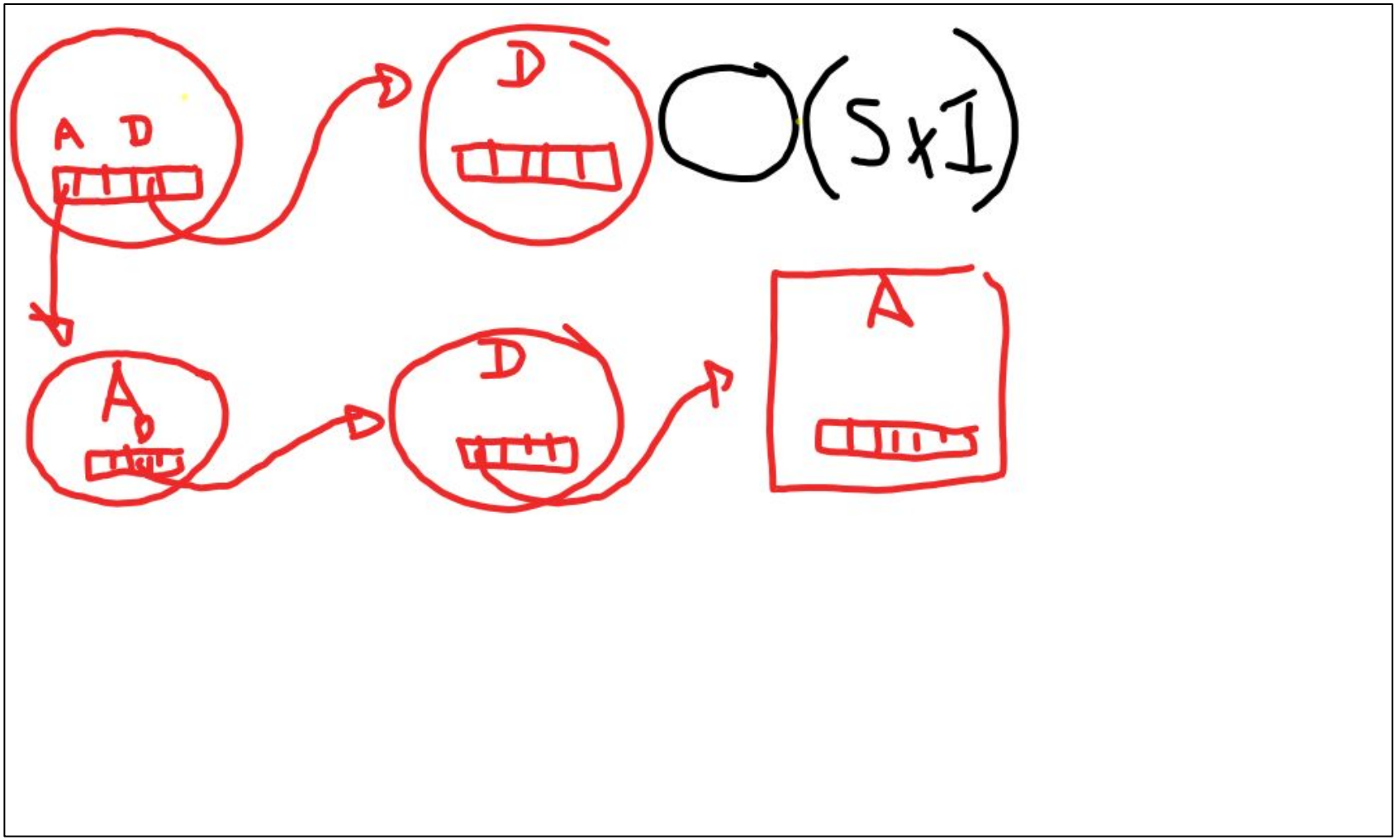
Exercício Resolvido (2)

- Implemente a Classe Nó da Árvore Trie **(AB Balanceada)**



Exercício Resolvido (2)

- Implemente a Classe Nó da Árvore Trie **(*Hash* Perfeita)**



Pesquisar por uma Cadeia de Caracteres

- Se cada nó tiver uma tabela *hash* perfeita para endereçar seus filhos, o tempo de pesquisa é $\Theta(s)$ onde s 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 $\Theta(s)$ e a construção total da árvore é $\Theta(n)$, onde n = |S|

Exercício (1)

- Implemente a árvore *trie* usando uma **tabela *hash* perfeita**

em seus nós: construtor, pesquisar, inserir e mostrar

Exercício (2)

- Implemente a árvore *trie* usando uma **árvore binária balanceada** em seus nós

- Implemente a árvore *trie* usando uma **lista flexível** em seus nós

Exercício (3)

- Implemente a árvore *trie* usando uma **tabela *hash* perfeita** em seus nós **aceitando a inserção de prefixos**