Tries R-Way, Ternária, Patricia. Arquivos Invertidos.

Eduardo Furlan Miranda

2024-02-06

Adaptado dos materiais dos Profs. Paulo Feofiloff, Marcio Bueno, e Mauro S. P. Fonseca

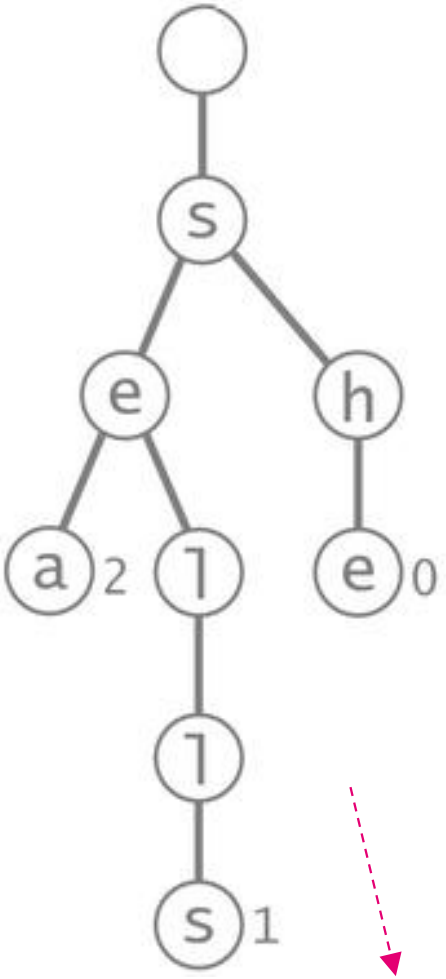
R-way Tries

R-way Tries

- Também conhecidas como árvores digitais ou árvores de prefixos
- Uma Trie (pronúncia "trái") é um tipo de árvore usado para implementar tabelas de símbolos de strings
- “R”-way Trie → cada nó pode ter até “R” filhos
- Aplicações:
  - Autocorretores: Para sugerir palavras com base em prefixos digitados pelo usuário
  - Sistemas de busca: Para indexar e recuperar palavras-chave
  - Dicionários: Para armazenar definições e palavras

Exemplo

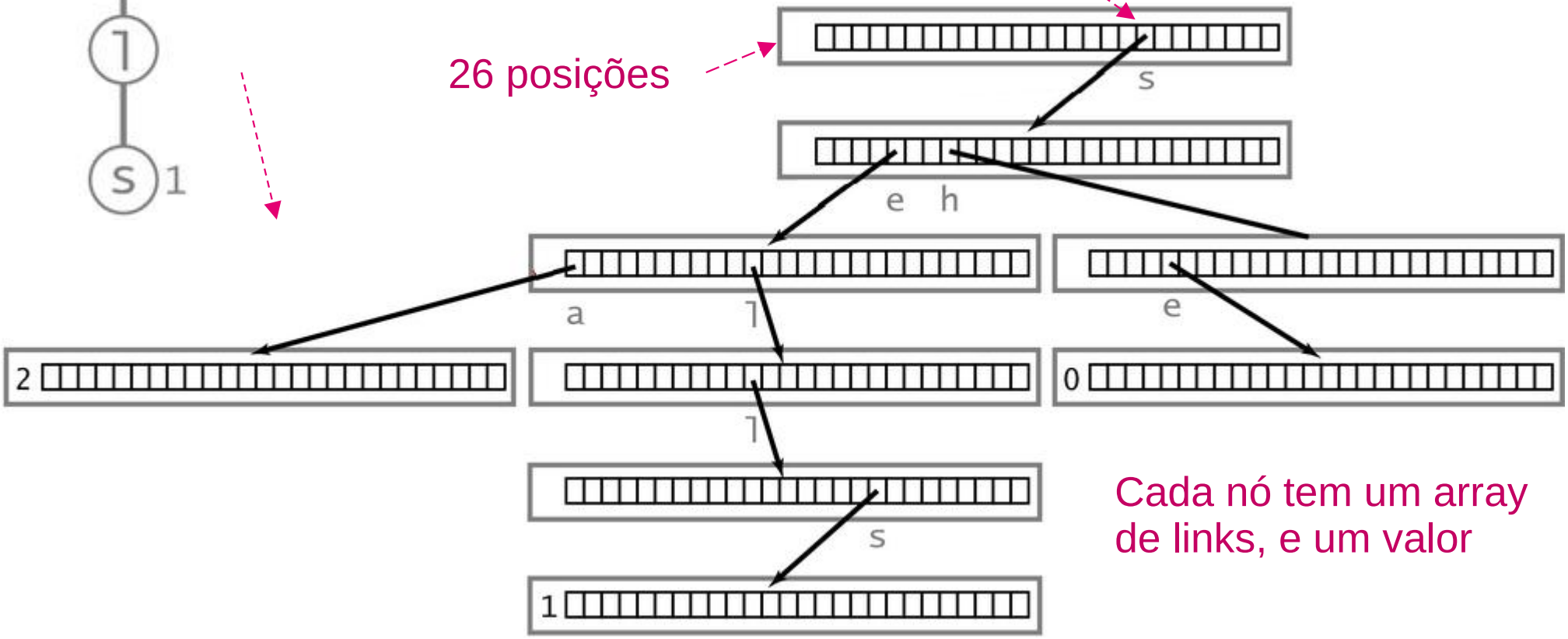
she sells sea (chave)
0 1 2 (valor)



- Trie para um alfabeto de R=26 caracteres

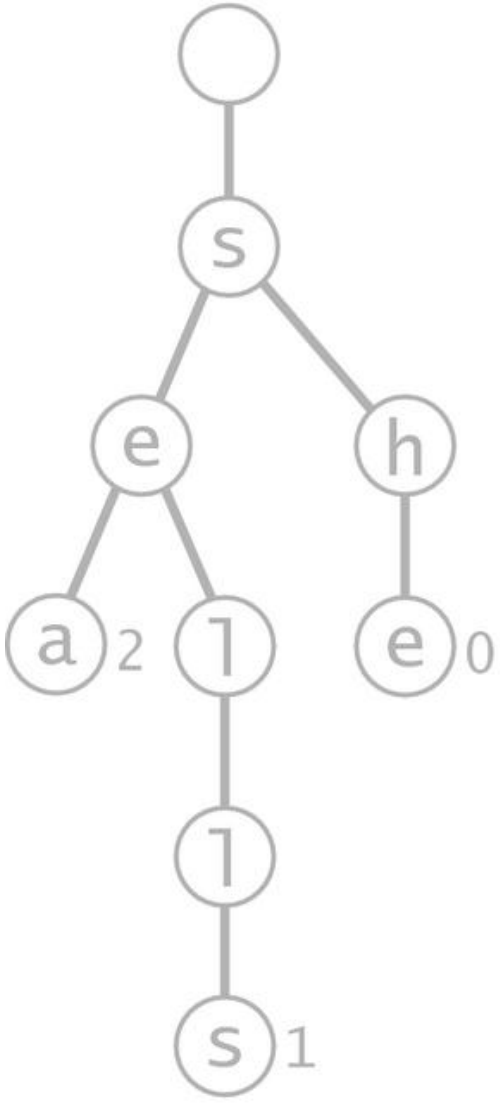
Os caracteres são implicitamente definidos pelos índices dos links

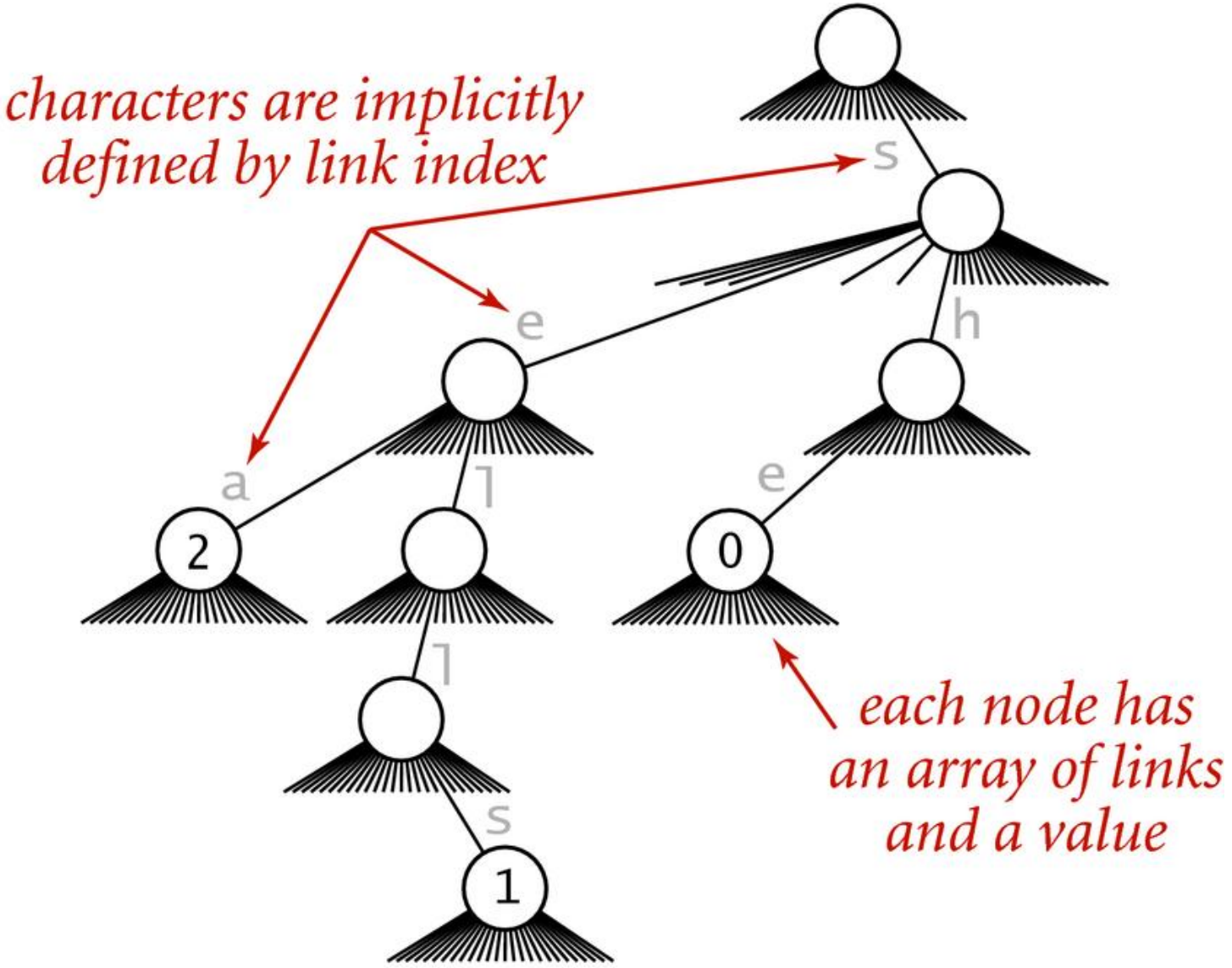
26 posições



Cada nó tem um array de links, e um valor

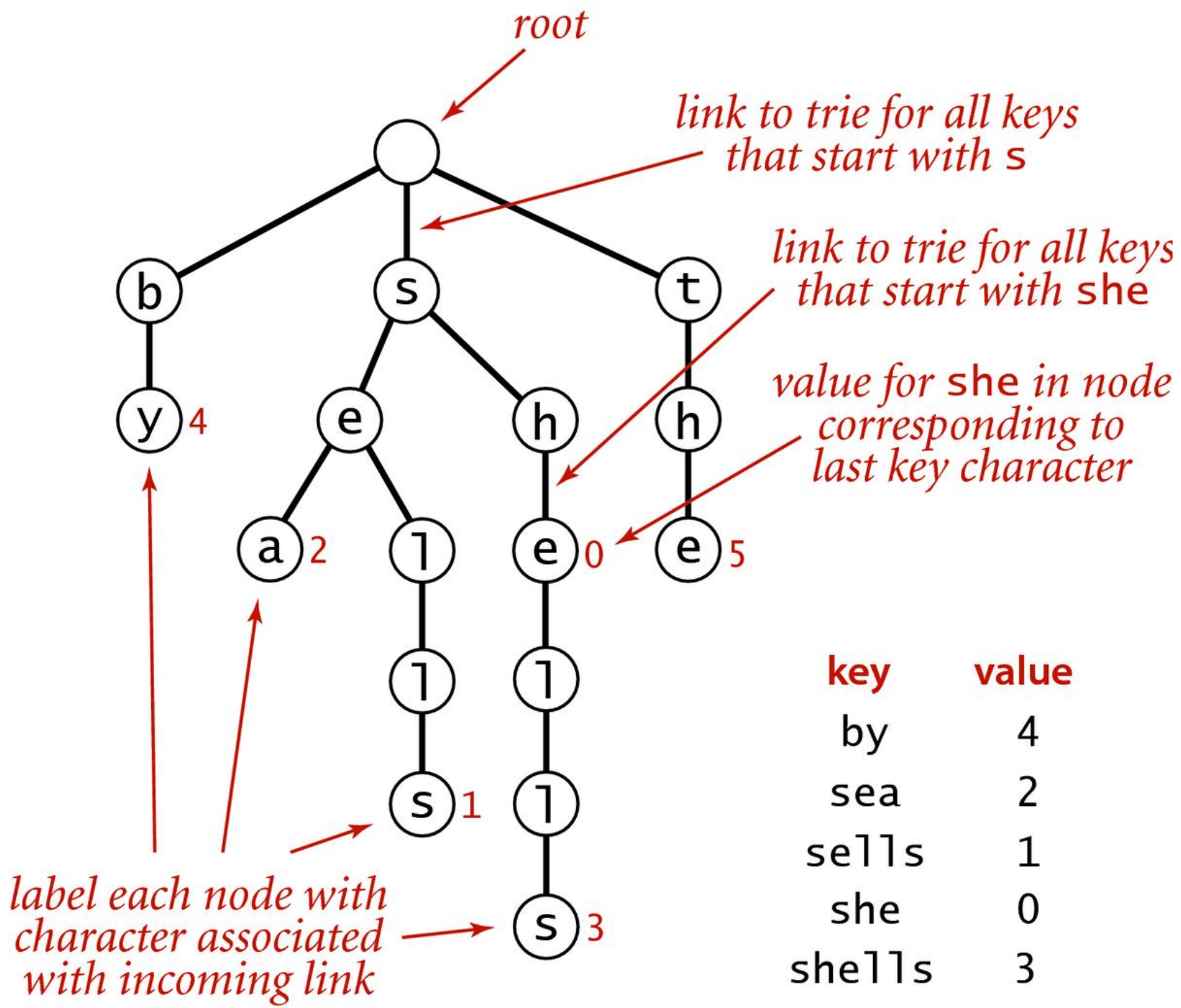
Outra representação da Trie anterior





Trie representation

- Nesse exemplo, o conjunto de chaves é sea, sells, she
- As strings sh e sell, por exemplo, estão representadas na Trie mas não são chaves
- O único nó não apontado por nenhum link é a raiz da Trie

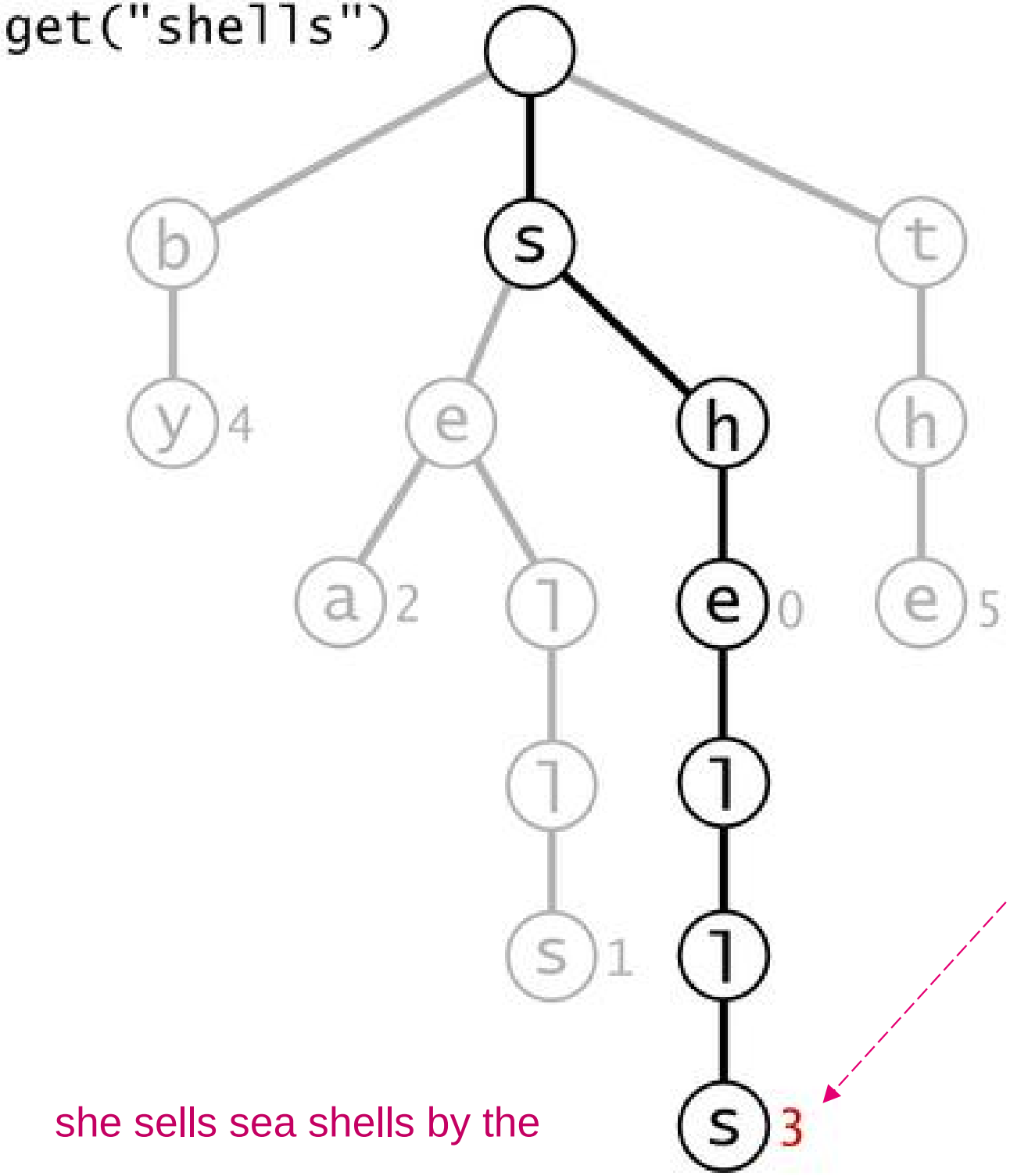


key	value
by	4
sea	2
sells	1
she	0
shells	3
the	5

- As chaves não são armazenadas explicitamente
  - Ficam codificadas nos caminhos que começam na raiz
- Todos os prefixos de chaves estão representados na Trie
  - Ainda que alguns prefixos não sejam chaves
- Os links da estrutura correspondem a caracteres
  - E não a chaves
- O caractere escrito dentro do nó é o caractere do link que entra no nó

- Ao descer da raiz até um nó “x”, soletramos uma string “s”
  - Dizemos que s leva ao nó x
  - Dizemos que o nó x é localizado pela string s
  - Ex.: o nó localizado pela string vazia é o root
- A string que leva a um nó x é uma chave se o valor armazenado em x não é nulo
- **SubTries**: cada nó “x” da Trie é a raiz de uma subTrie “X”
  - A subTrie X representa o conjunto de todas as chaves da Trie que têm como prefixo a string que leva da raiz da Trie até x

Busca - hits

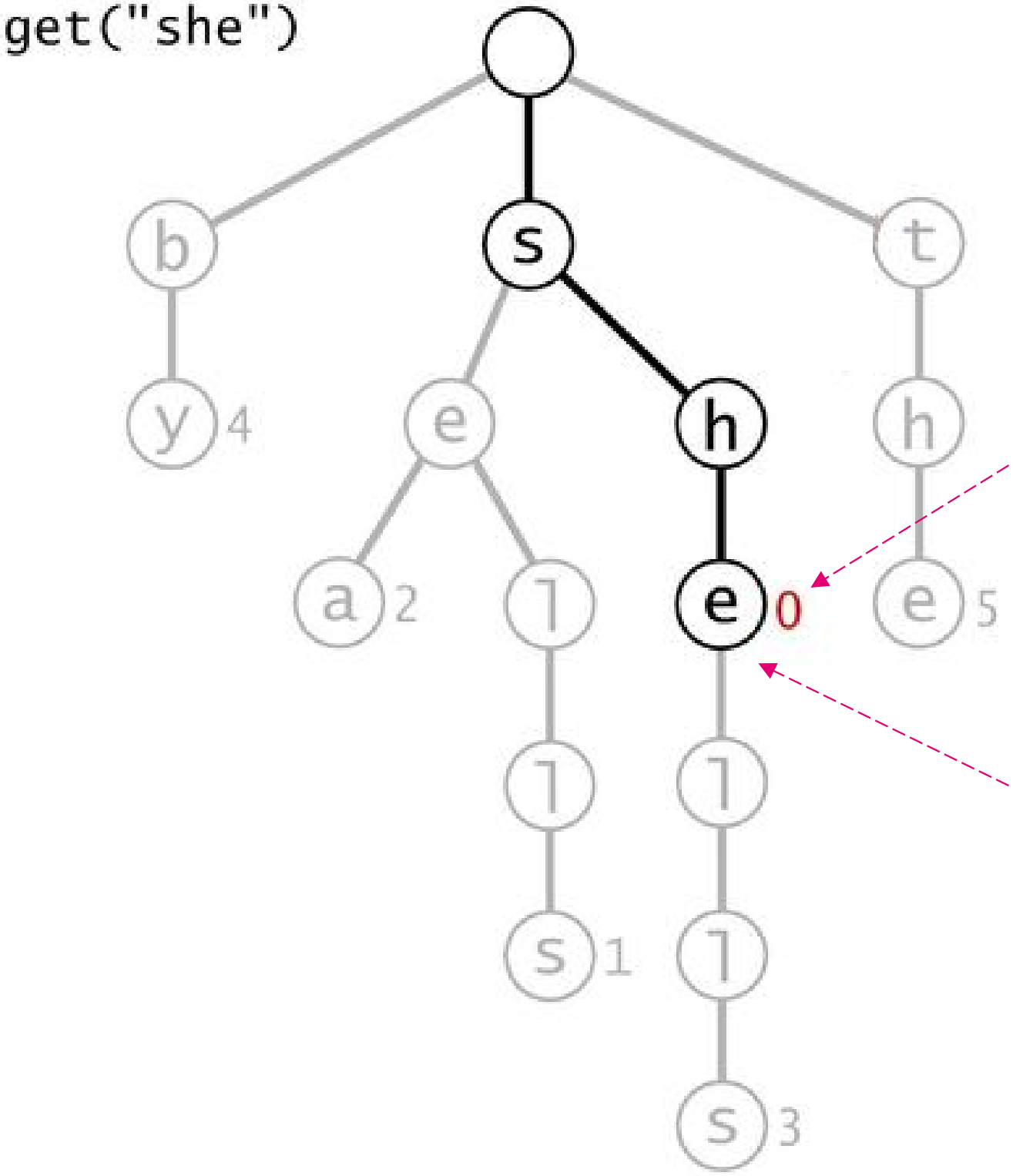


mesma árvore anterior

- Retorna o valor do nó correspondente ao último caractere chave

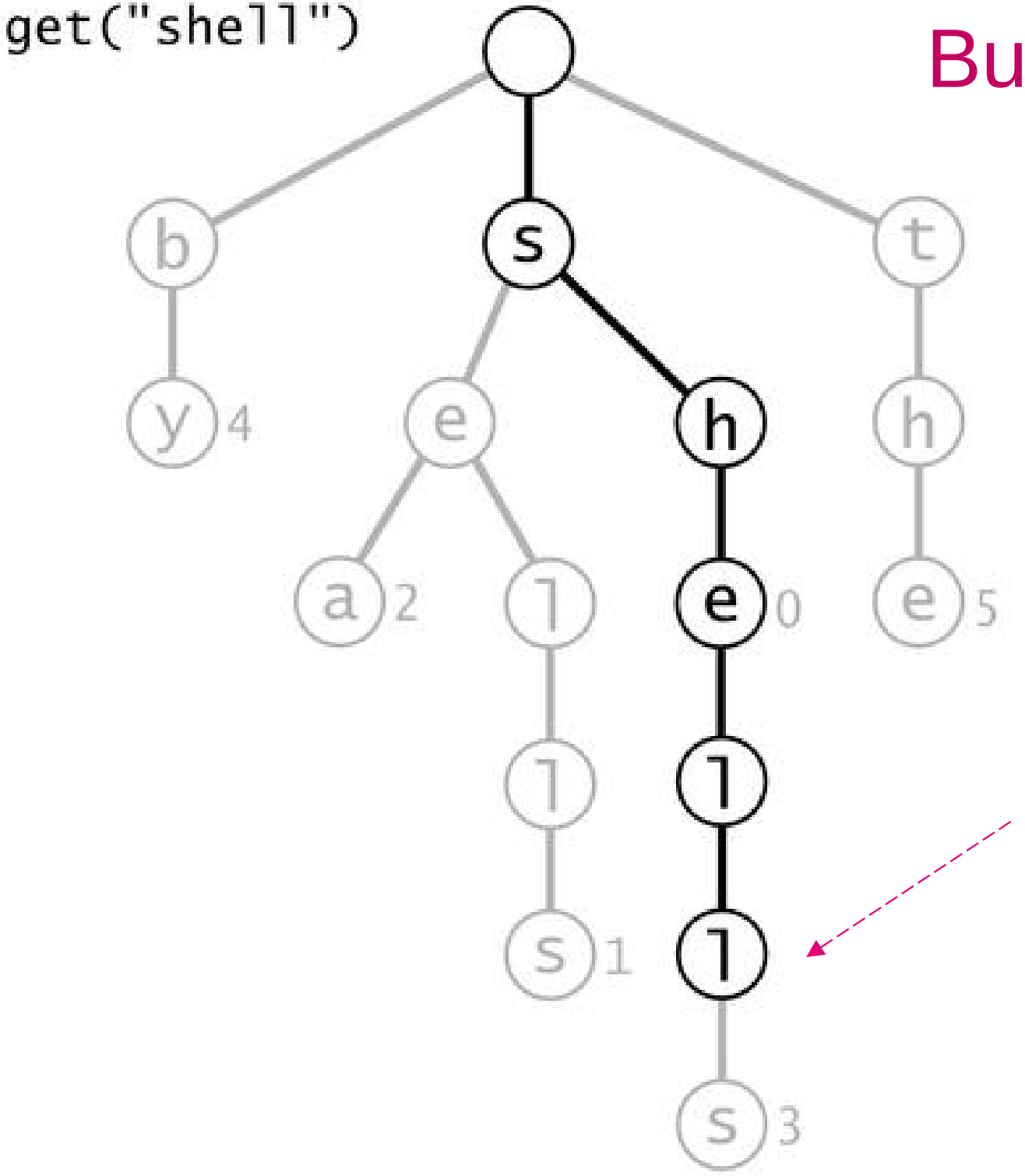
she sells sea shells by the

Busca - hits



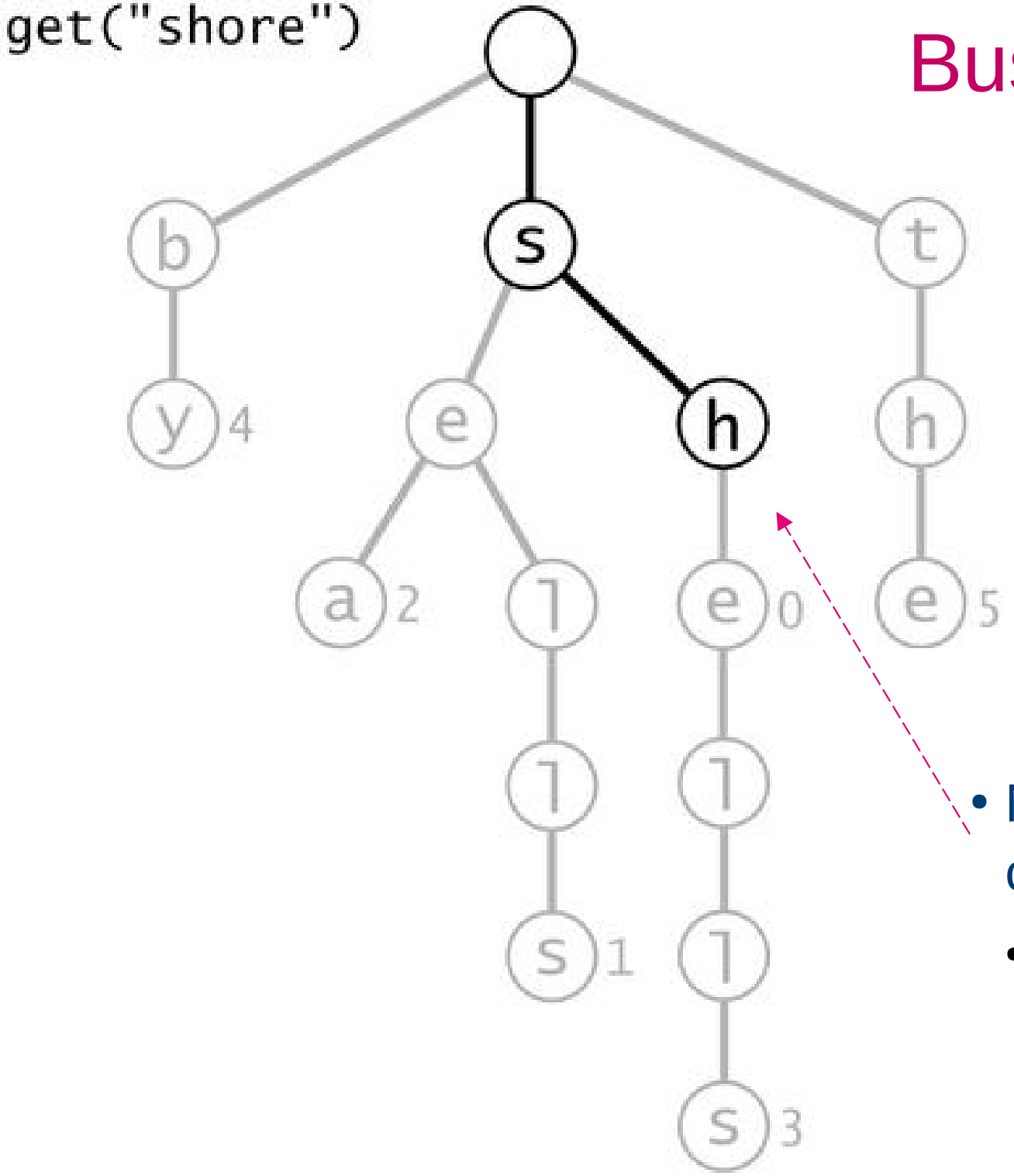
- Retorna o valor do nó correspondente ao último caractere chave
- A procura pode terminar em um nó interno

Busca - misses



- O valor no nó correspondente ao último caractere é null
  - Então retorna nulll

Busca - misses



- Não existe o caractere “o”
  - Retorna null

Árvore Ternária de Busca (*Ternary Search Trie* - TST)

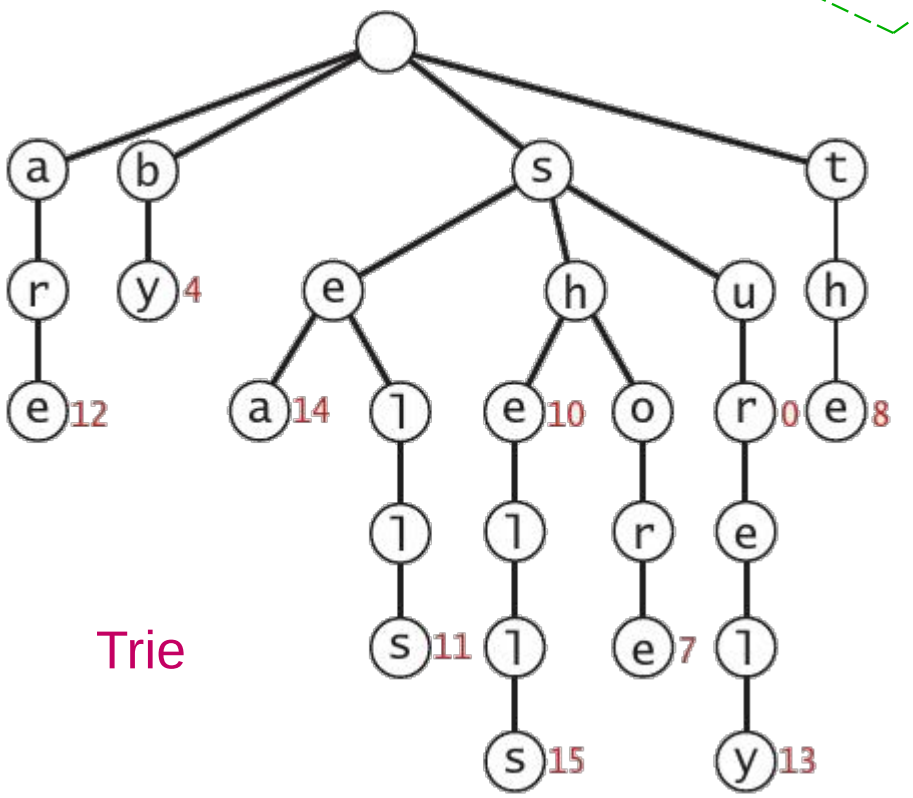
TST

- É um tipo de Trie (às vezes chamado de árvore de prefixos)
- Os nós são organizados em uma forma semelhante a uma árvore de busca binária
  - Cada nó tem 3 filhos
    - menor (esquerda), igual (meio), maior (direita)
- Armazena caracteres e valores em nós (e não chaves)
- Funciona apenas para strings (ou chaves digitais)
- Examina apenas os caracteres-chave suficientes
- O erro de busca pode envolver apenas alguns caracteres
- Suporta operações ordenadas de tabela de símbolos

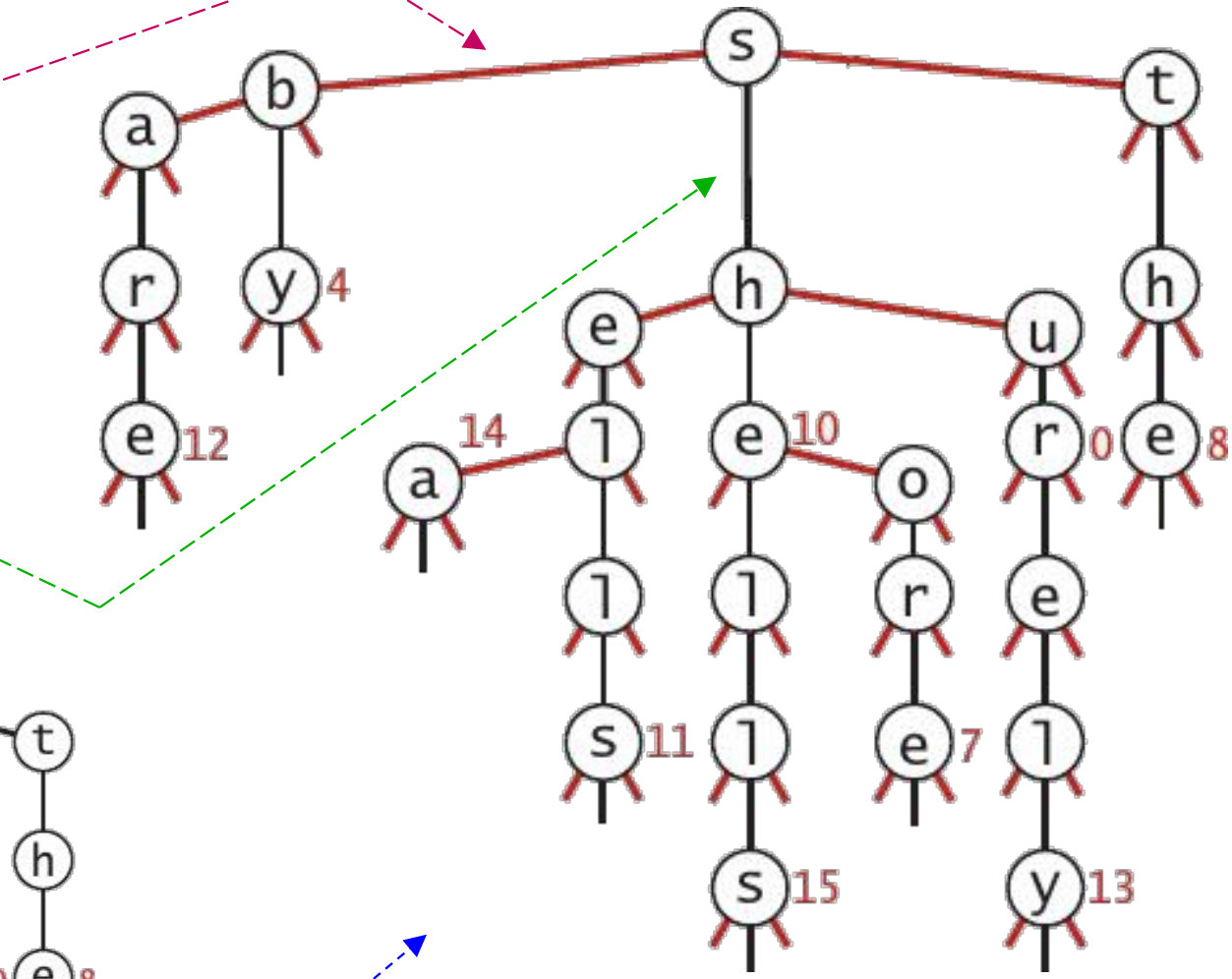
TST

Link para o TST, para todas as chaves que iniciam com a letra **menor** que “s”

Link para o TST, para todas as chaves que iniciam com a **mesma** letra “s”



Trie



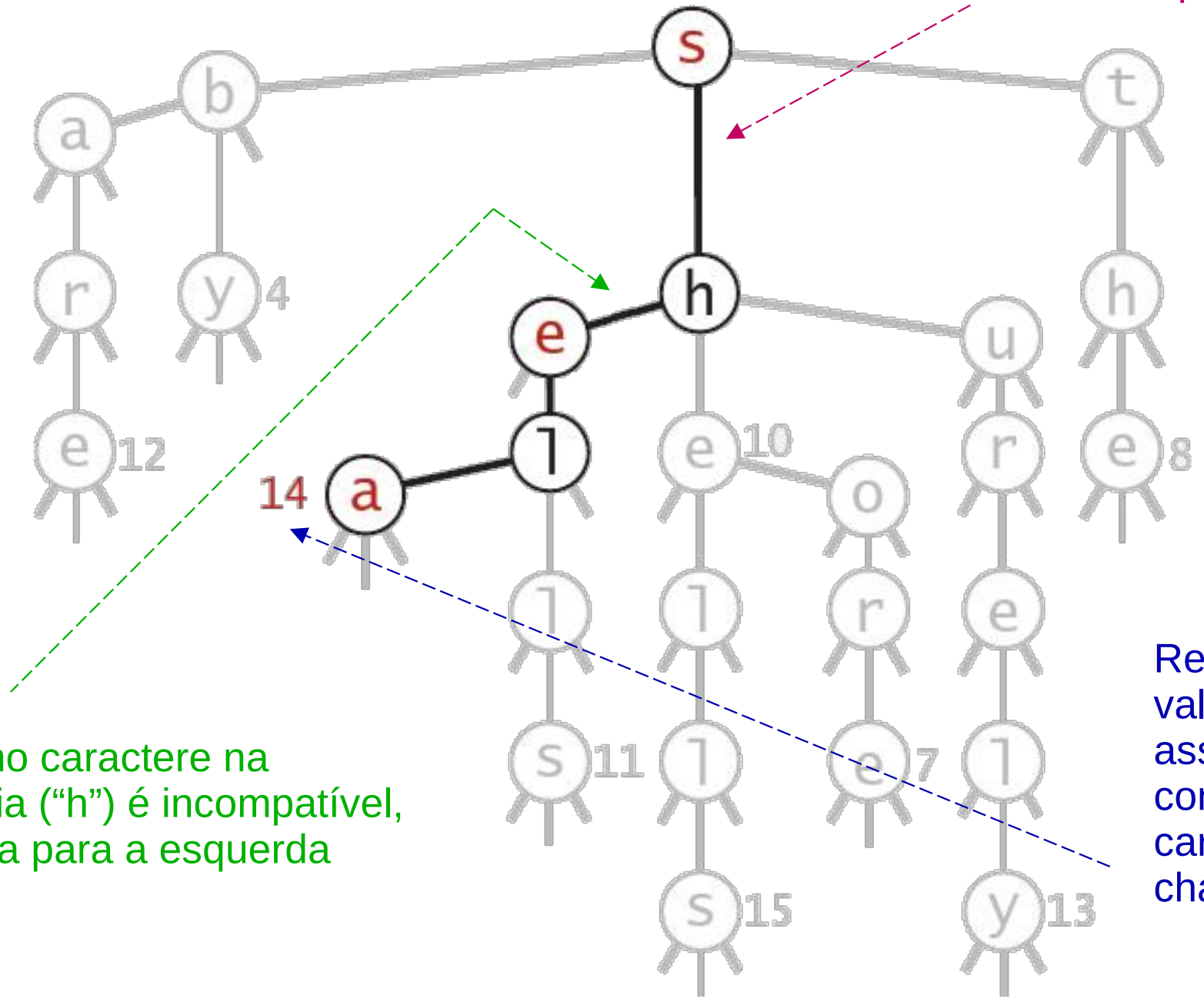
TST

Pesquisa em uma TST

- Percorrer os links correspondentes a cada caractere na chave
  - **Se menor**: pegar o link à esquerda
  - **Se maior**: pegar o link à direita
  - **Se igual**: pegar o link do meio
- Mover para o próximo caractere da chave
- Acerto (*search hit*)
  - O nó onde a procura termina tem um valor não-nulo
- Erro (*search miss*)
  - Encontra um link nulo, ou
  - O nó onde a pesquisa termina tem valor nulo

get("sea")

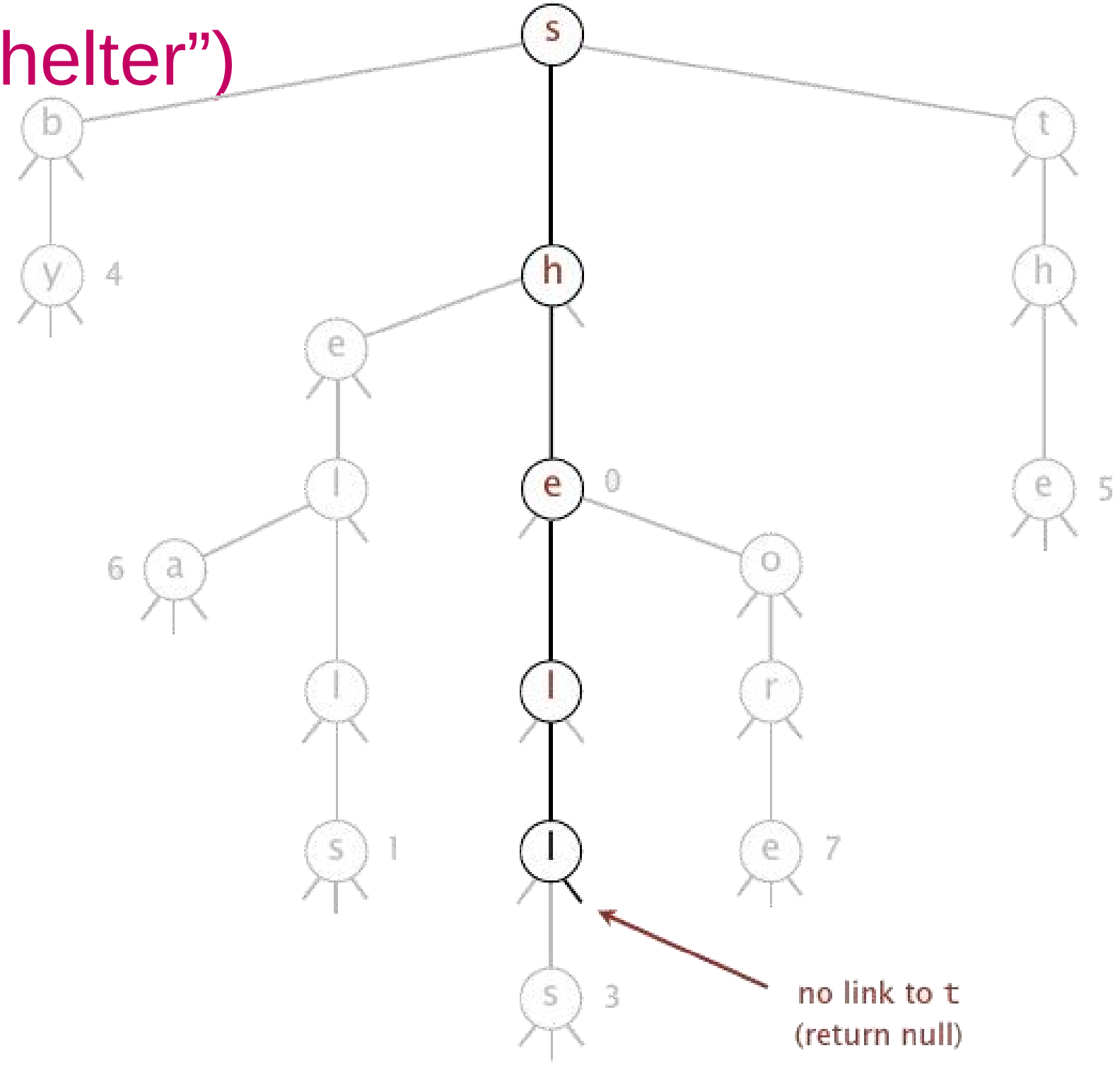
O primeiro caractere ("s") corresponde, então pega o caminho do meio ("igual") e continua neste link até o próximo



O próximo caractere na sequência ("h") é incompatível, então vira para a esquerda

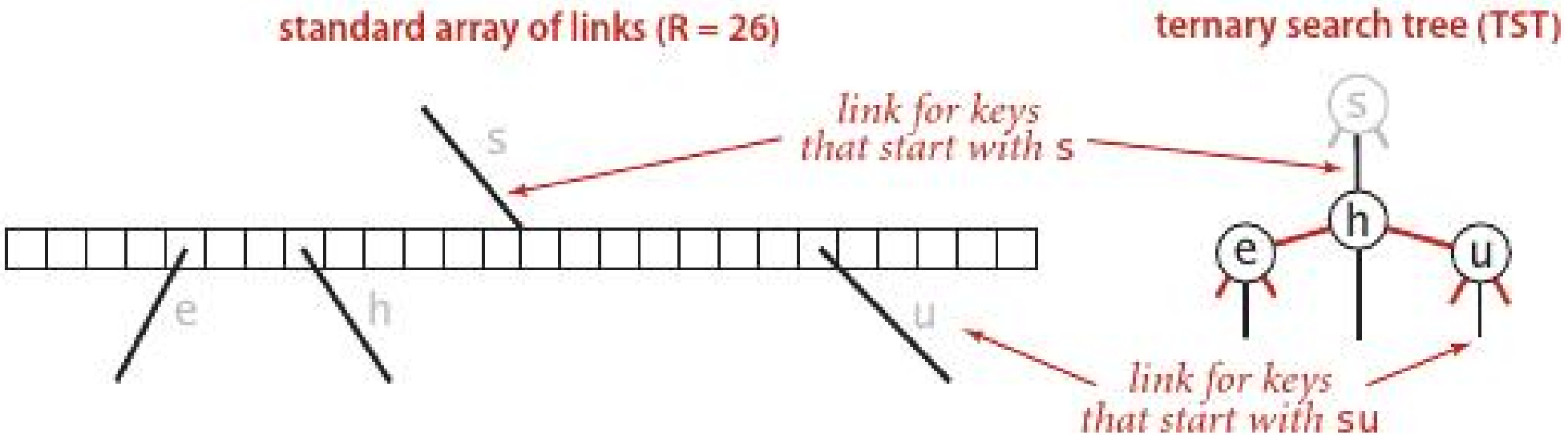
Retorna o valor associado com o último caractere da chave

get("shelter")



Representação TST

- Um nó TST tem 5 campos
  - 1 valor
  - 1 caractere
  - 3 referências, esquerda, meio, direita

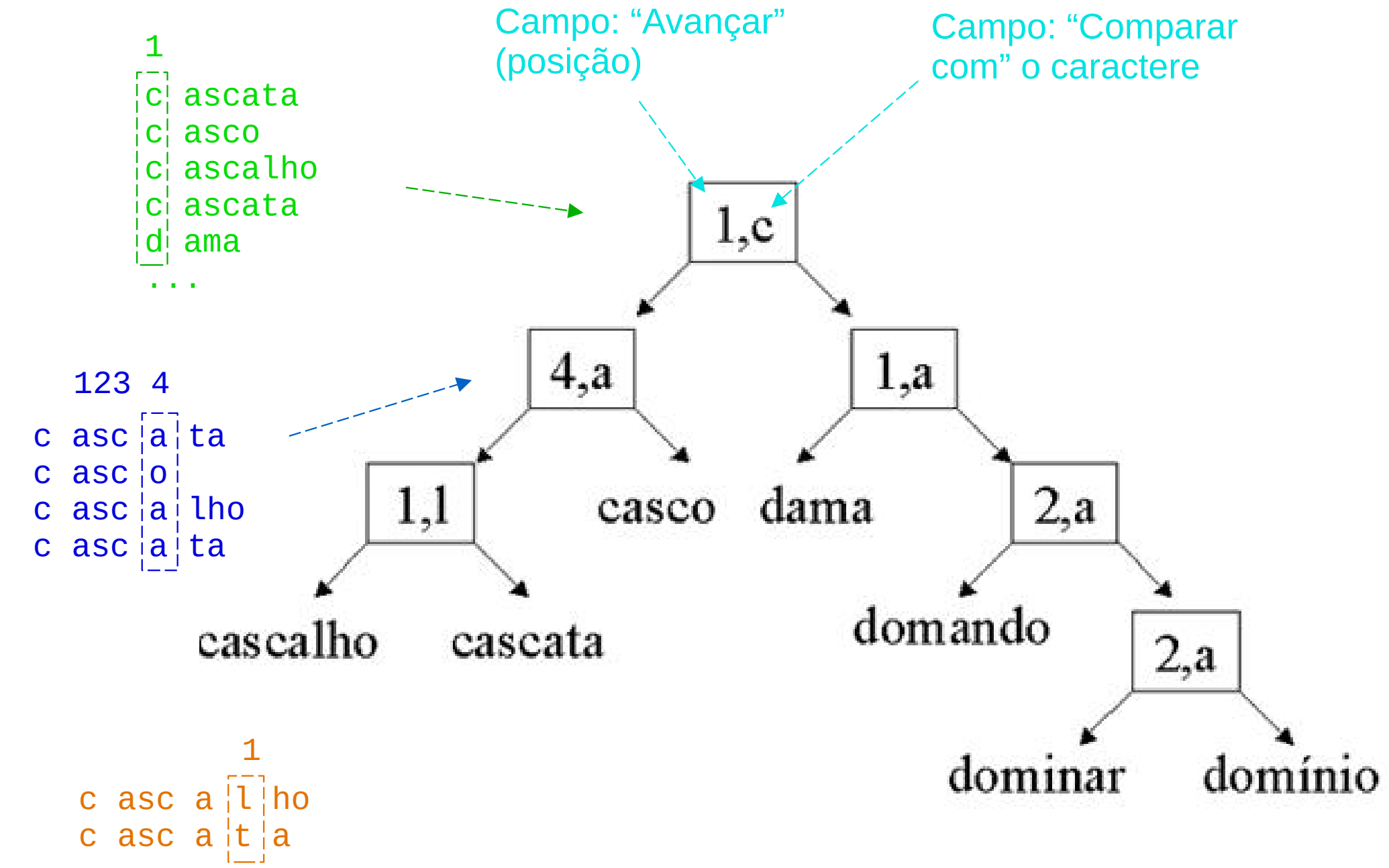


Árvore PATRICIA (*PATRICIA Trie*)

- Também conhecida como *crit-bit tree* e *radix tree*
- *PATRICIA = “Practical Algorithm To ReTrieve Information Coded In Alphanumeric”*

Adaptado do material do Prof. Marcio Bueno

Exemplo


1
c ascata
c asco
c ascalho
c ascata
d ama
...
Campo: “Avançar” (posição)
Campo: “Comparar com” o caractere
1,c
4,a
1,a
1,l
casco
dama
2,a
cascalho
cascata
domando
2,a
dominar
domínio
123 4
c asc a ta
c asc o
c asc a lho
c asc a ta
1
c asc a l ho
c asc a t a

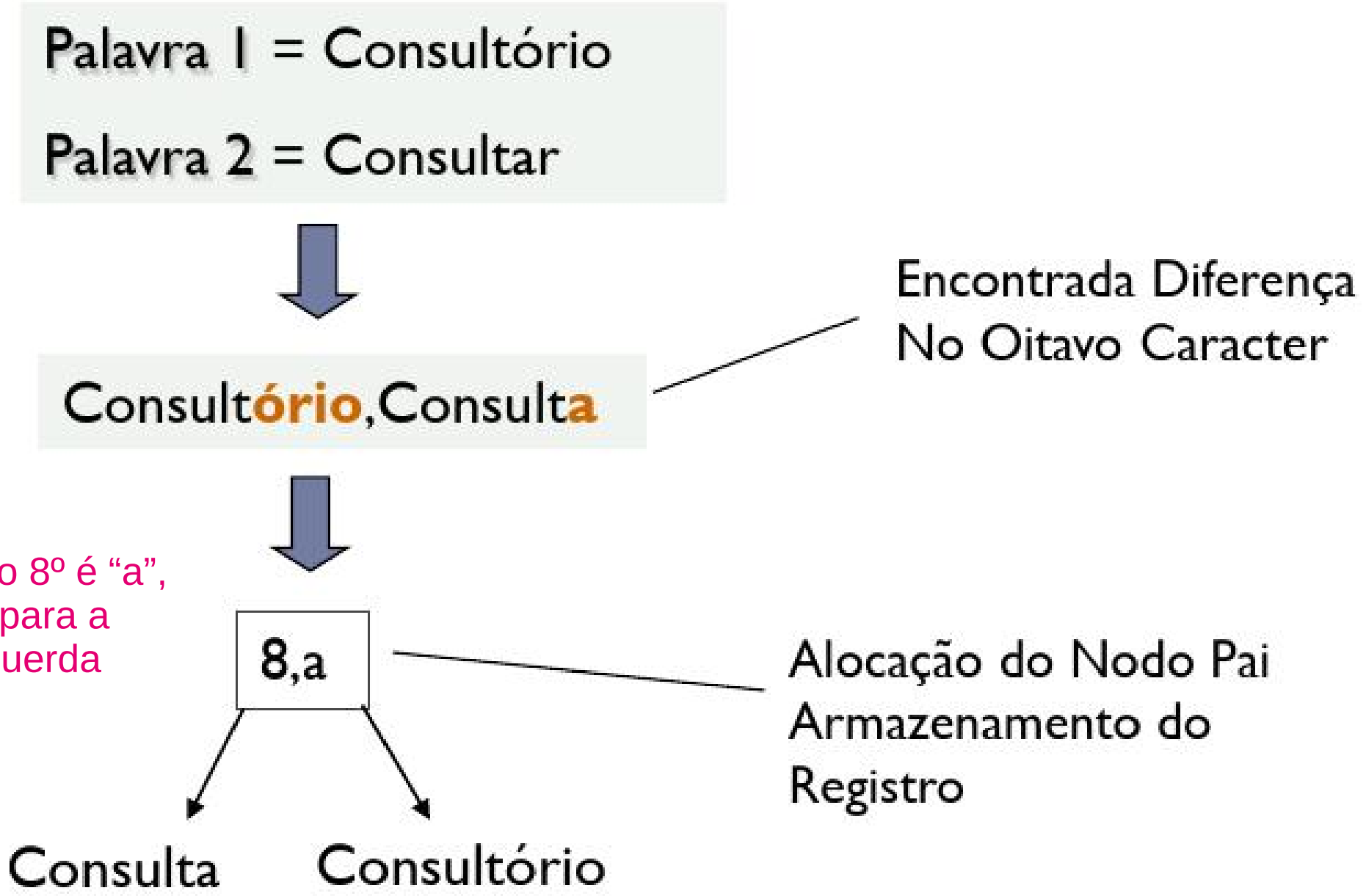
PATRICIA

- Trie compacta binária
- Não armazena informações nos nós internos
  - Apenas contadores e ponteiros para cada subárvore descendente
- Nós com 1 filho são agrupados nos seus antecessores
  - Cada nó representa uma sequência de caracteres
- Reduz o espaço ocupado por árvores Trie com chaves dispersas e apenas um descendente
- Escolhe um elemento da chave (armazenando a sua posição) para determinar a subárvore
  - Ao invés de usar cada uma das partes de uma chave

PATRICIA

- Campo “Avançar”
  - Registro acumulativo que integra todos os nós exceto as folhas
  - Identifica qual a posição do caractere da chave informada que deve ser analisado
- Campo “Comparar com”
  - Apresenta o caractere que deve ser comparado ao caractere da chave informada
  - Como nas árvores binárias de busca, após a análise, se a chave
    - É menor ou igual ao nó
      - Ela é alocada/consultada à esquerda
    - Senão
      - À direita

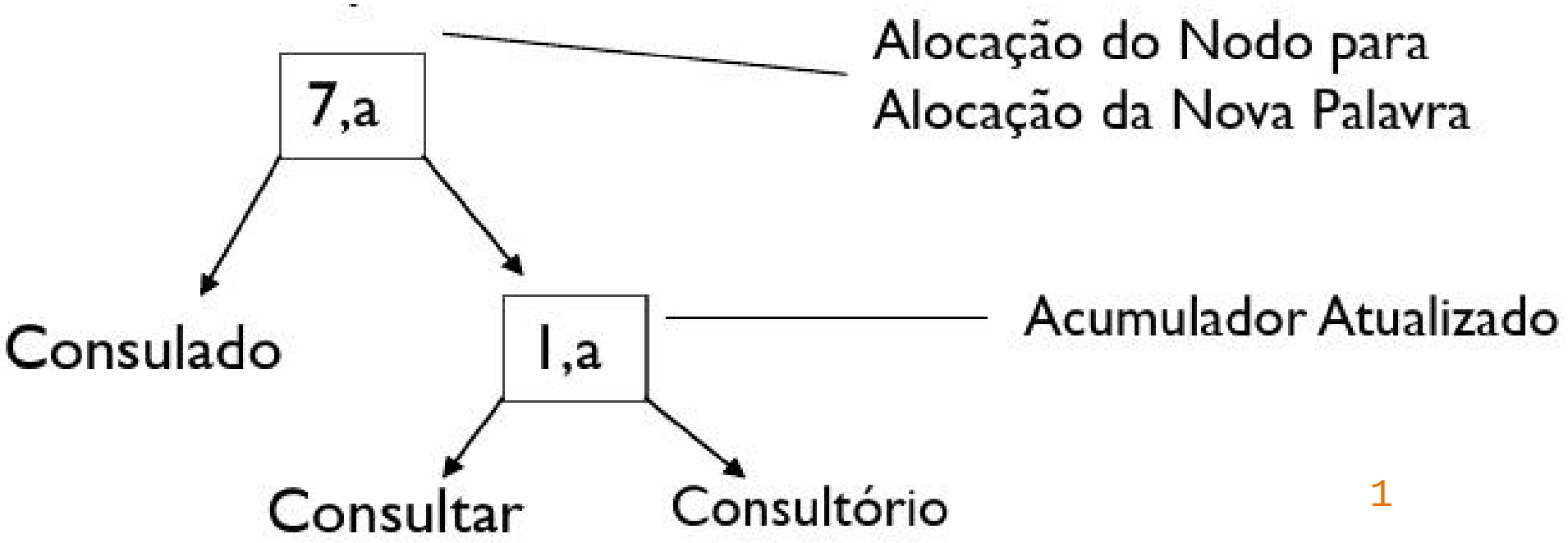
Exemplo de inserção



Exemplo 2 de inserção

7

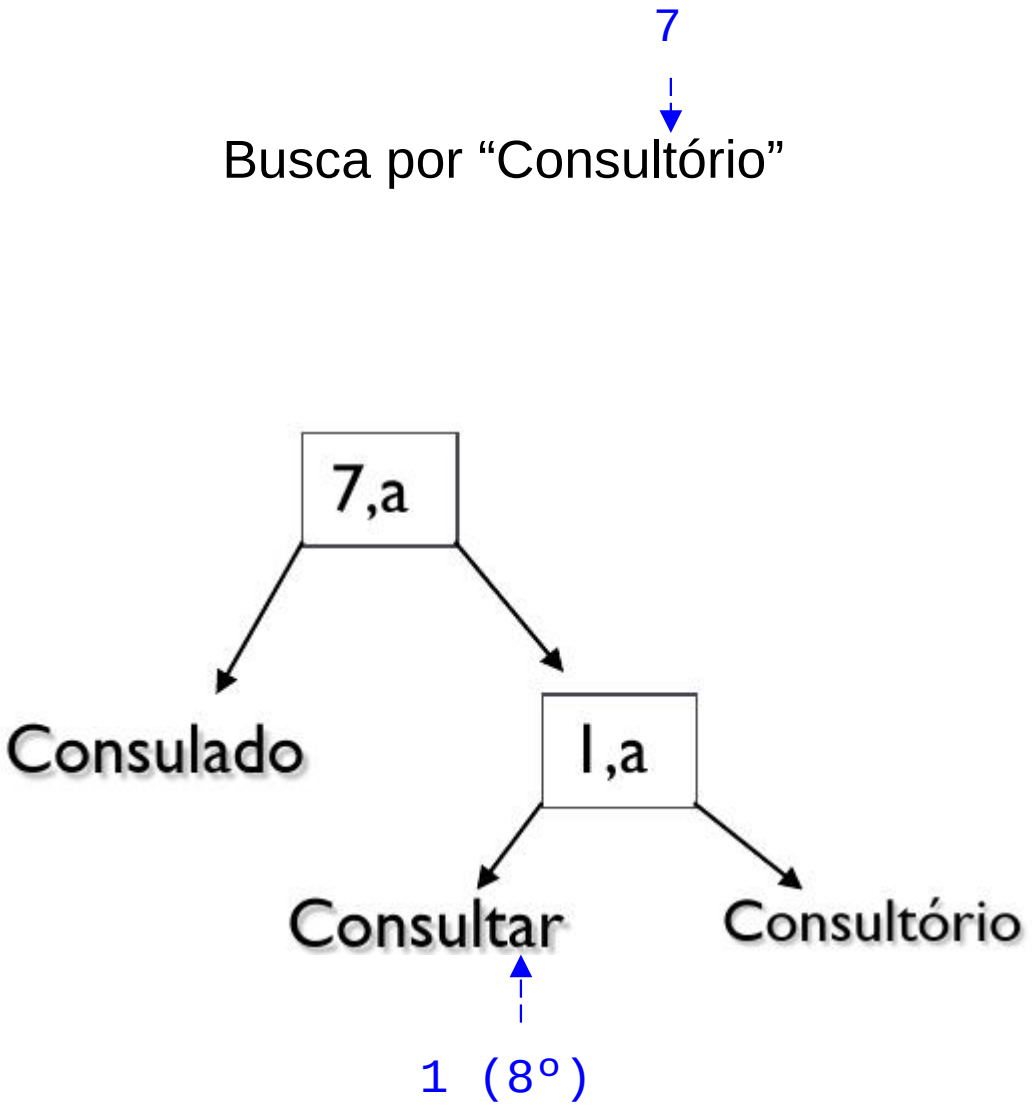
Consul a do
Consul t ar
Consul t ório



1

Consul t a r
Consul t ó rio

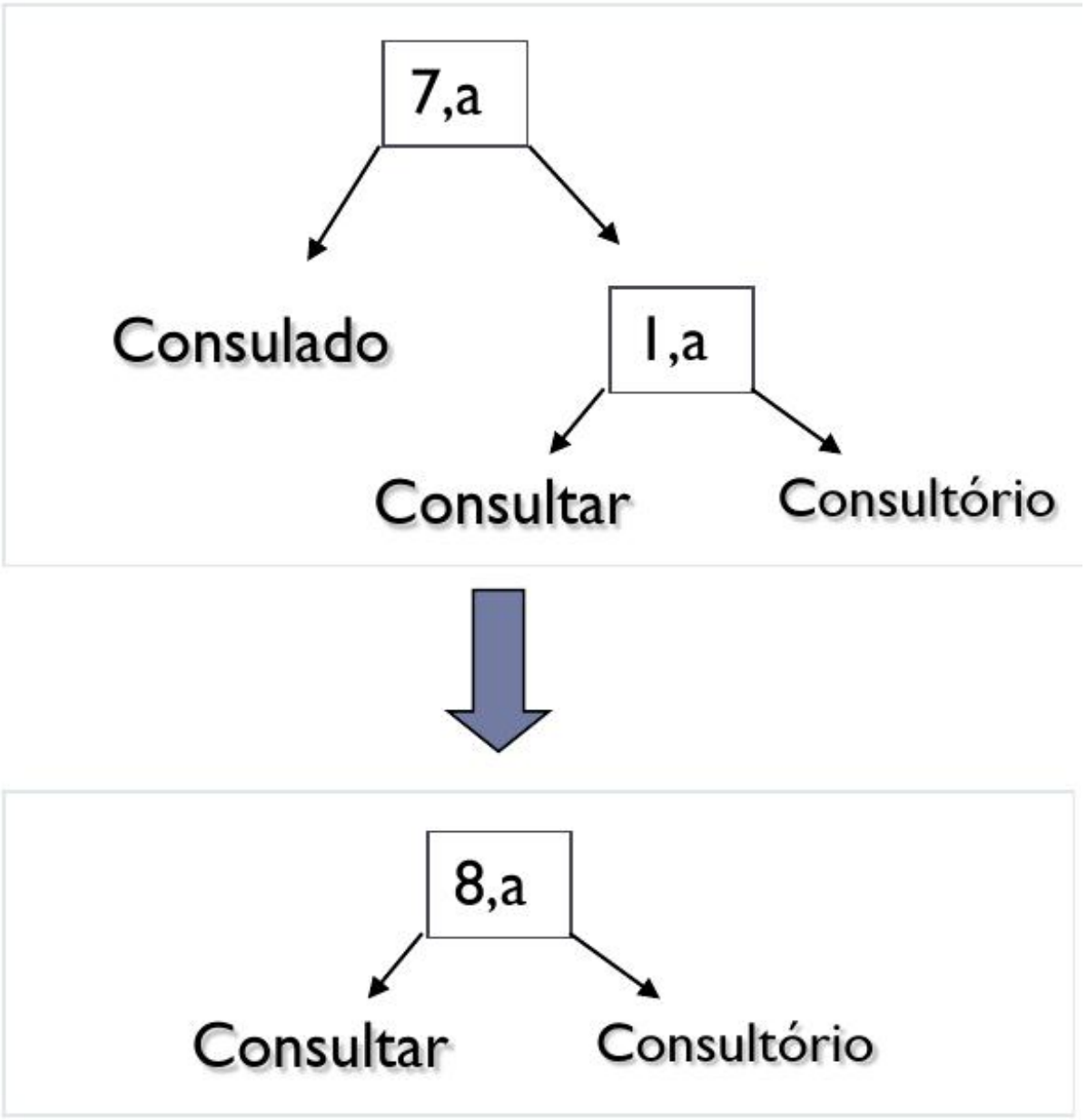
Exemplo de consulta



- O primeiro nó fala para pegar o 7º caractere (“t”) e comparar com “a”
- Como é maior vai para a direita
- Reseta o contador para 1
  - O 8º vira posição 1
- Na posição 1 agora pega o caractere “ó” e compara com “a”
- Como é maior, vai para a direita

Apagar

- Apagar “Consulado”
- Primeiro buscar a palavra
- Apagar a palavra
- Somar o valor do campo Avançar do nó pai a todos os nós filhos



Arquivos invertidos (*inverted file*)

- Também conhecido como *inverted index*

Adaptado do material do Prof. Mauro S. P. Fonseca

Exemplo

Texto exemplo. Texto tem palavras. Palavras exercem fascínio.

exemplo	7			
exercem	45			
fascínio	53			
palavrar	26	36		
tem	22			
texto	1	16		

Arquivo invertido

- Um arquivo invertido possui duas partes
  - Vocabulário
  - Ocorrências
- O vocabulário é o conjunto de todas as palavras distintas no texto
  - Para cada palavra distinta, uma lista de posições onde ela ocorre no texto é armazenada
- O conjunto das listas é chamado de ocorrências
- As posições podem referir-se a palavras ou caracteres

Pesquisa

- A pesquisa tem geralmente três passos
  - Pesquisa no vocabulário
    - Palavras e padrões da consulta são isoladas e pesquisadas no vocabulário
  - Recuperação das ocorrências
    - As listas de ocorrências das palavras encontradas no vocabulário são recuperadas
  - Manipulação das ocorrências
    - As listas de ocorrências são processadas para tratar frases, proximidade, ou operações booleanas
- Como a pesquisa em um arquivo invertido sempre começa pelo vocabulário, é interessante mantê-lo em um arquivo separado
  - Na maioria das vezes, esse arquivo cabe na memória principal

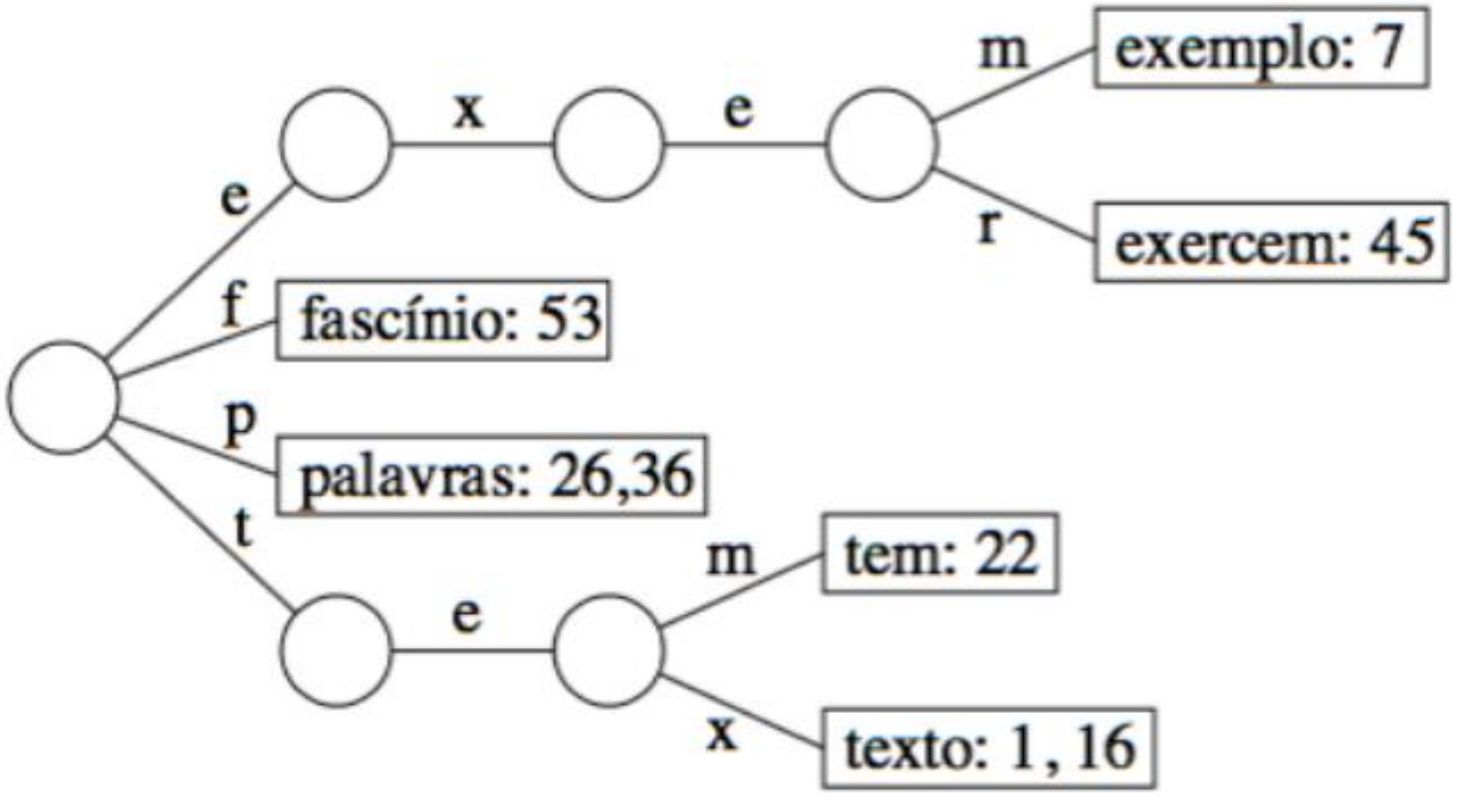
Pesquisa

- A pesquisa de palavras simples pode ser realizada usando qualquer estrutura de dados que torne a busca eficiente, como hashing, árvore Trie ou árvore B.
- Guardar as palavras na ordem lexicográfica é barato em termos de espaço e competitivo em desempenho.
- A pesquisa por frases usando índices é mais difícil de resolver.
  - Cada elemento da frase tem de ser pesquisado separadamente e suas listas de ocorrências recuperadas.
  - A seguir, as listas têm de ser percorridas de forma sincronizada para encontrar as posições nas quais todas as palavras aparecem em sequência.

Arquivo invertido usando Trie

1 7 16 22 26 36 45 53

Texto exemplo. Texto tem palavras. Palavras exercem fascínio.



- Cada nova palavra lida é pesquisada na Trie
  - Se a pesquisa é sem sucesso, então a palavra é inserida na árvore e uma lista de ocorrências é inicializada com a posição da nova palavra no texto
  - Senão, uma vez que a palavra já se encontra na árvore, a nova posição é inserida ao final da lista de ocorrências

Referências

FEOFILOFF, P. Tries (árvores digitais). [S. l.]: IME-USP, 2022. Disponível em: https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/Tries.html.

BUENO, M. Árvores Trie e Patricia. [S. l.]: Universidade Católica de Pernambuco, UNICAP, 2009. Disponível em: https://marciobueno.com/arquivos/ensino/ed2/ED2_06_Trie_Patricia.pdf.

FONSECA, M. S. P. Processamento de Cadeias de Caracteres. [S. l.]: Universidade Tecnológica Federal do Paraná, UTFPR, 2015. Disponível em: https://pessoal.dainf.ct.utfpr.edu.br/maurofonseca/lib/exe/fetch.php?media=cursos:if63c:if63ced_10_cadeias.pdf.

SEDGEWICK, R.; WAYNE, K. Algorithms. [S. l.]: Addison-Wesley Professional, 2011.