Estrutura de Dados I

ÁRVORE TRIE

Isabela Romeiro S. C. Rosa
Tayrone Cordeiro de M. Marques
Thaynnara Luciano dos Santos

28/11/2017 – Equipe C

Tópicos

1. O que é uma árvore trie?
2. Representação
3. Aplicações
4. Características das chaves da árvore Trie
5. Principais Operações
6. Questionário
7. Referências Bibliográficas

1

O QUE É UMA ÁRVORE TRIE?

“Uma árvore trie (pronuncia-se “trái”) é um tipo de árvore usado para implementar tabelas de símbolos de strings.

É também conhecida como árvore digital ou árvore de prefixos.

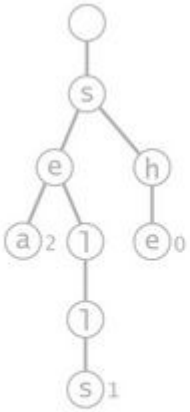
UTILIZAÇÃO DAS TRIES

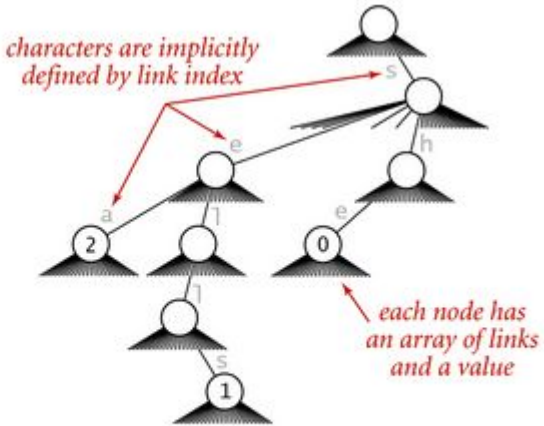
- As Tries são boas para suportar tarefas de tratamento lexicográfico, tais como:
  - Manuseamento de dicionários;
  - Pesquisas em textos de grandes dimensões;
  - Construção de índices de documentos e;
  - Expressões regulares (padrões de pesquisa);

2

REPRESENTAÇÃO

- Cada nó contém informações sobre um ou mais símbolos do alfabeto utilizado e, também, o caracter nulo (ou branco);
- Uma dada sequência de arestas pode formar qualquer palavra (chave) possível com base nesse alfabeto;
- Não existe limite para o tamanho de uma sequência (comprimento variável);
- O único nó não apontado por nenhum link é a raiz da trie.
- 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.



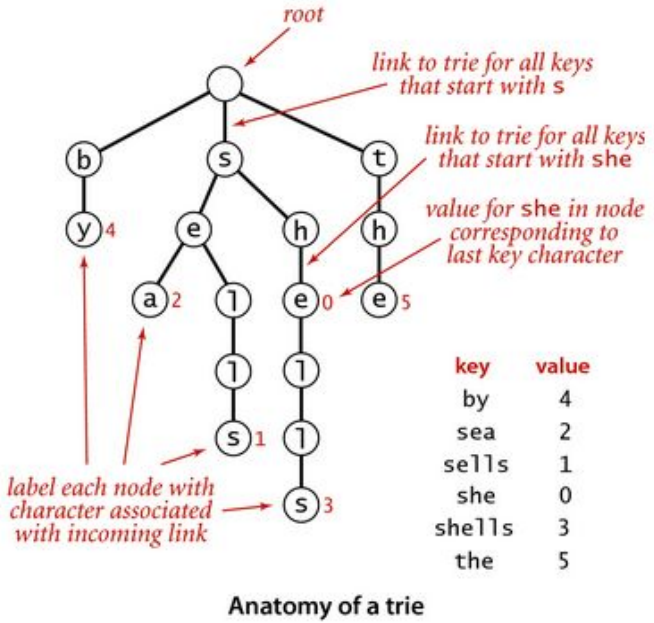


Trie representation

OBSERVAÇÕES IMPORTANTES

1) As chaves não são armazenadas explicitamente. Elas ficam codificadas nos caminhos que começam na raiz.

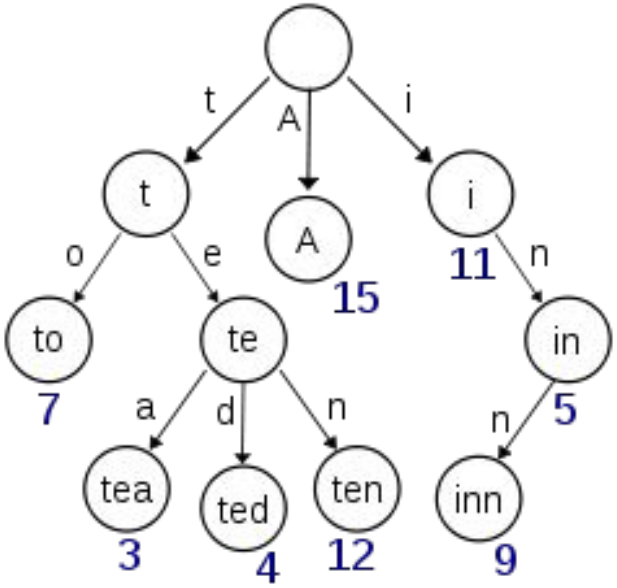
2) Todos os prefixos de chaves (que nem sempre são chaves) estão representados na trie.



Anatomy of a trie

Ao descer da raiz até um nó x, soletramos uma string, digamos, S. Dizemos que S leva ao nó x. Dizemos também que o nó x é localizado pela string S. (Por exemplo, o nó localizado pela string vazia é root).

A string que leva a um nó x é uma chave se e somente se x.val != NULL.



SUBTRIES

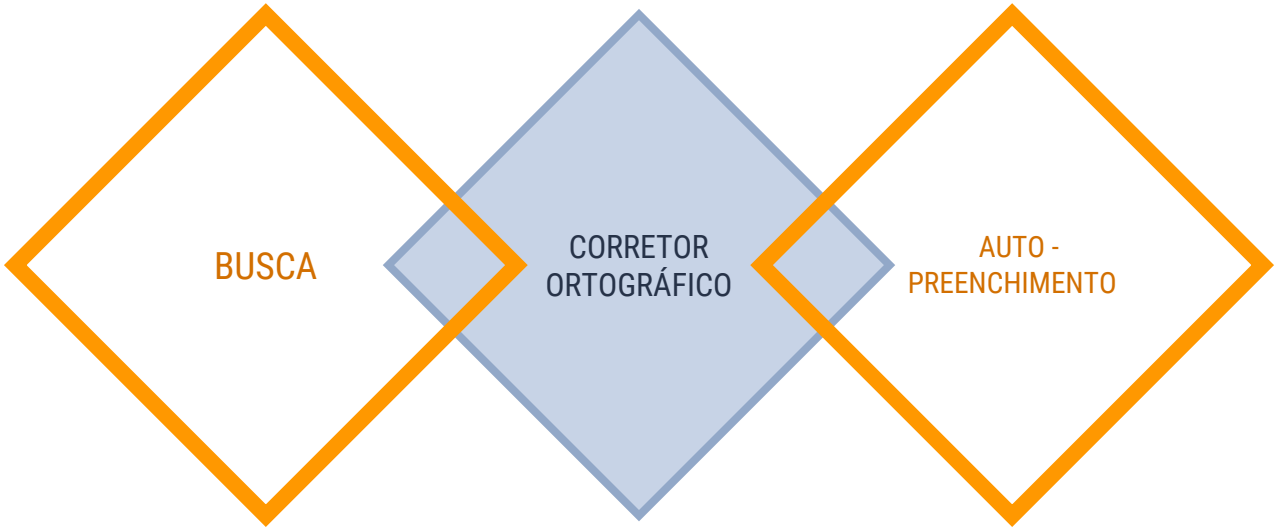
Cada nó x da trie é a raiz de uma subtrie, digamos X.

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.

3

APLICAÇÕES

APLICAÇÕES



BUSCA

- Objetivo: localizar um dado que corresponde a chave informada.
- Problema: podem ocorrer erros na entrada dos dados a serem buscados, por exemplo, palavras com grafias semelhantes.
- Solução: existe um método de busca por aproximação de correspondência, onde podemos localizar dados que são semelhantes a uma chave informada. Pela estrutura de representação de caractere a caractere usada nas tries, elas acabam tendo um desempenho muito bom nesse tipo de aplicação.

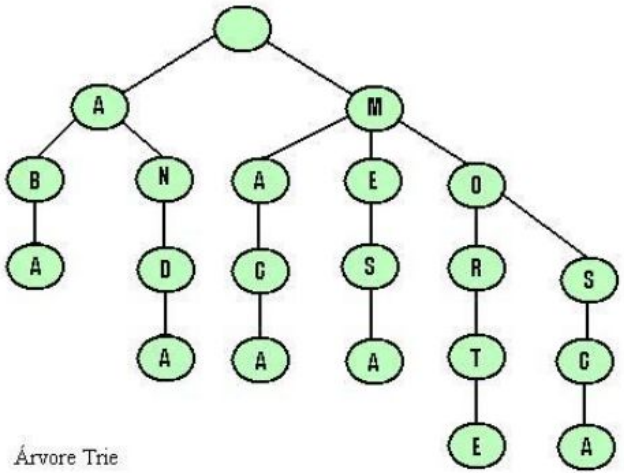
CORRETOR ORTOGRÁFICO

- As palavras são comparadas com um dicionário armazenado em arquivo, e se não são encontradas indica-se as opções para correção.
- Com o dicionário armazenado numa trie, pode-se percorrer essa estrutura letra por letra para encontrar, ou não, a palavra testada.
- Com base na chave informada o algoritmo vai percorrer a árvore que contém o dicionário, enquanto as letras da chave e alguma letra de cada nível da árvore coincidirem.
- Caso seja detectado um erro na chave o algoritmo verifica a possibilidade de ocorrência de cada um dos tipos de erros para poder indicar as opções de correção.

CORRETOR ORTOGRÁFICO (cont.)

- **MÉTODOS DE CORREÇÃO:**
  - Substituição: avança um caractere na chave e avança um nível na árvore;
  - Deleção: avança um nível na árvore;
  - Inserção: avança um caractere na chave;
  - Transposição: avança um nível na árvore e testa a posição atual da chave, se coincidir, avança um caractere na chave e retrocede um nível na árvore para confirmar a inversão;

CORRETOR ORTOGRÁFICO (cont.)



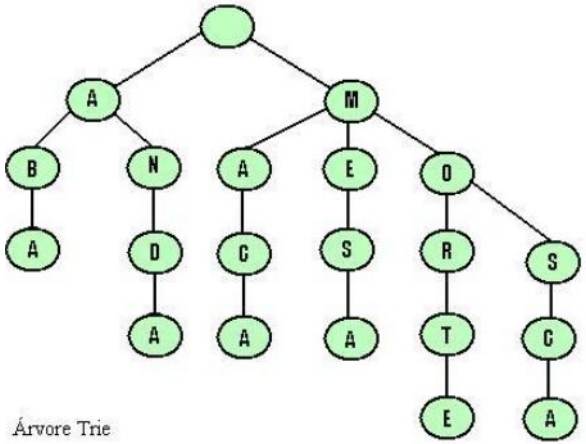
Árvore Trie

Com as seguintes palavras: **ABA, ANDA, MACA, MESA, MORTE, MOSCA.**

Digamos que a chave a ser testada seja ADA, onde ocorreu erro na tentativa de escrever ABA. Será realizada a seguinte seqüência de testes :

* A = A - ok
* D = B - **erro**
* D = N - **erro**
* próximo passo avança na chave e na árvore (substituição)
* A = A - ok

CORRETOR ORTOGRÁFICO (cont.)



Árvore Trie

Detectado erro de substituição, onde a letra B foi substituída por D. Nesse ponto o algoritmo pode parar e apresentar as opções de correção, ou continuar verificando ocorrência dos outros tipos de erros a partir do ponto em que foi encontrada divergência entre o dicionário e a chave.

Vamos então analisar o teste de erro de deleção para a mesma chave.

* A = A - ok * D = B - erro * D = N - erro

* próximo passo avança somente na árvore (deleção)

* D = A - erro * D = D - ok * A = A - ok

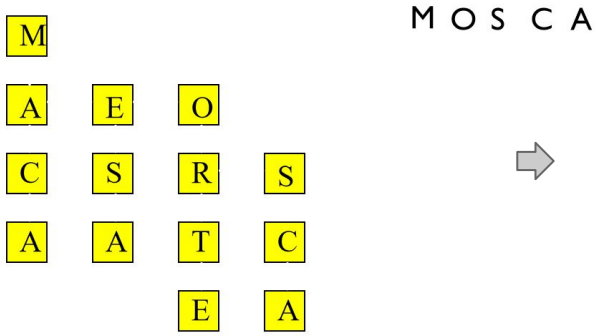
Detectado erro de deleção, onde a letra N foi suprimida da chave.

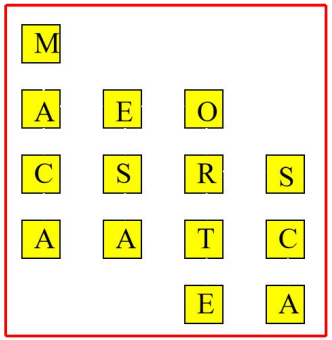
AUTO-PREENCHIMENTO

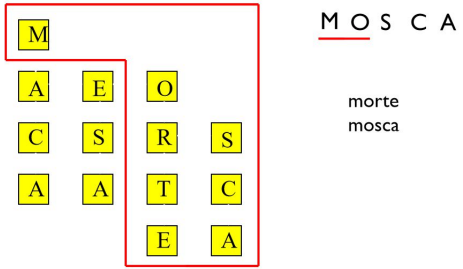
- As palavras mais utilizadas pelo usuário são armazenados em tries e a medida que é digitada uma sequência de caracteres o algoritmo vai comparando a existência de correspondências na estrutura. A cada caractere digitado, são apresentadas as opções de preenchimento, e no momento em que só existir um caminho possível a ser seguido na trie, ocorre o preenchimento automático.

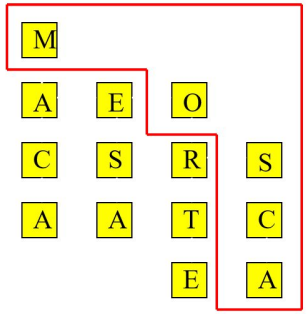
- Utilização em browsers, programas de e-mail, aplicativos para smartphones, entre outros.

AUTO-PREENCHIMENTO (cont.)









4

Caracteristícas das chaves da árvore Trie

Chave

Na árvore Trie cada chave é tratada como um elemento divisível, ou seja cada chave é composta por um conjunto de caracteres ou dígitos contidos em um alfabeto de símbolos.

EX:

ARARA → A R A R A

↓

(5 caracteres tratados que serão tratados individualmente em cada nó).

Chave (alfabetos)

Exemplos de alfabetos que podem conter nas chaves:

➢ {0,1}:
Palavras: 0101010100000000001010000000001010

➢ {A, B, C,...Z, a, b, c,..., z}:
Palavras: ABABBBABABA, Maria

➢ {0,1,2,3,4,5,...,9}:
Palavras: 19034717

5

Principais operações

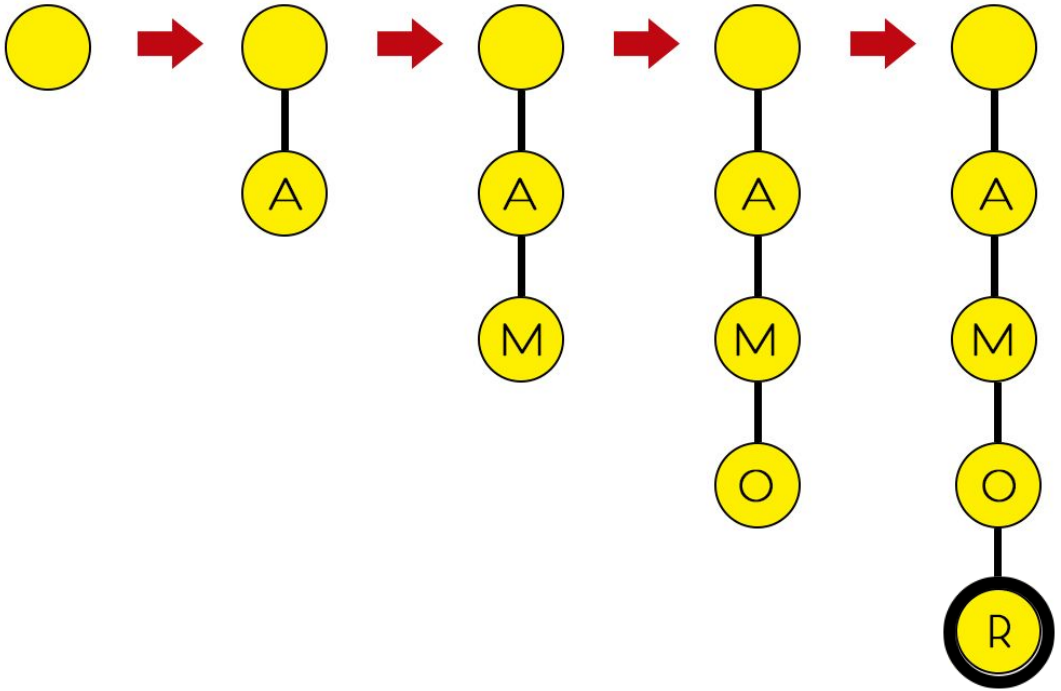
Inserção

Para realizar a inserção é realizado o seguinte método:

- Realiza-se uma busca pela árvore para descobrir onde será inserida a nova chave, para verificar a existência de um nó ou chave já existente.
- Na já existência da chave a ser inserida na árvore na TRIE nada é feito.
- Caso contrário, é recuperado o nó até onde acontece a maior substring da palavra a ser inserida.
- O restante dos seus caracteres são adicionados na TRIE a partir daquele nó.

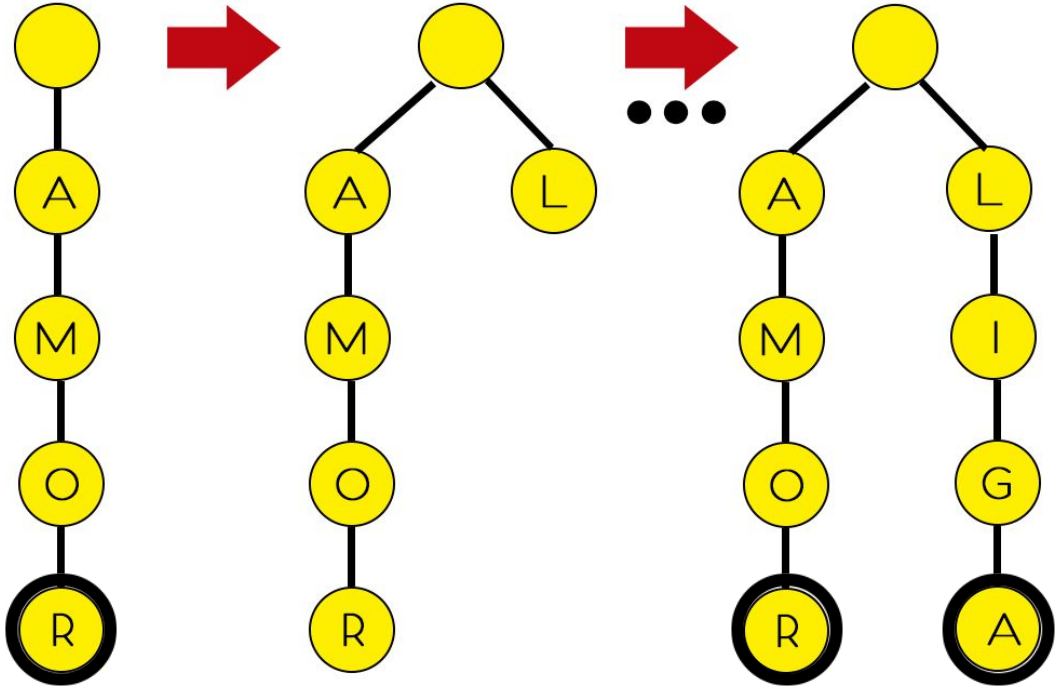
Inserção

- Inserção: **AMOR**



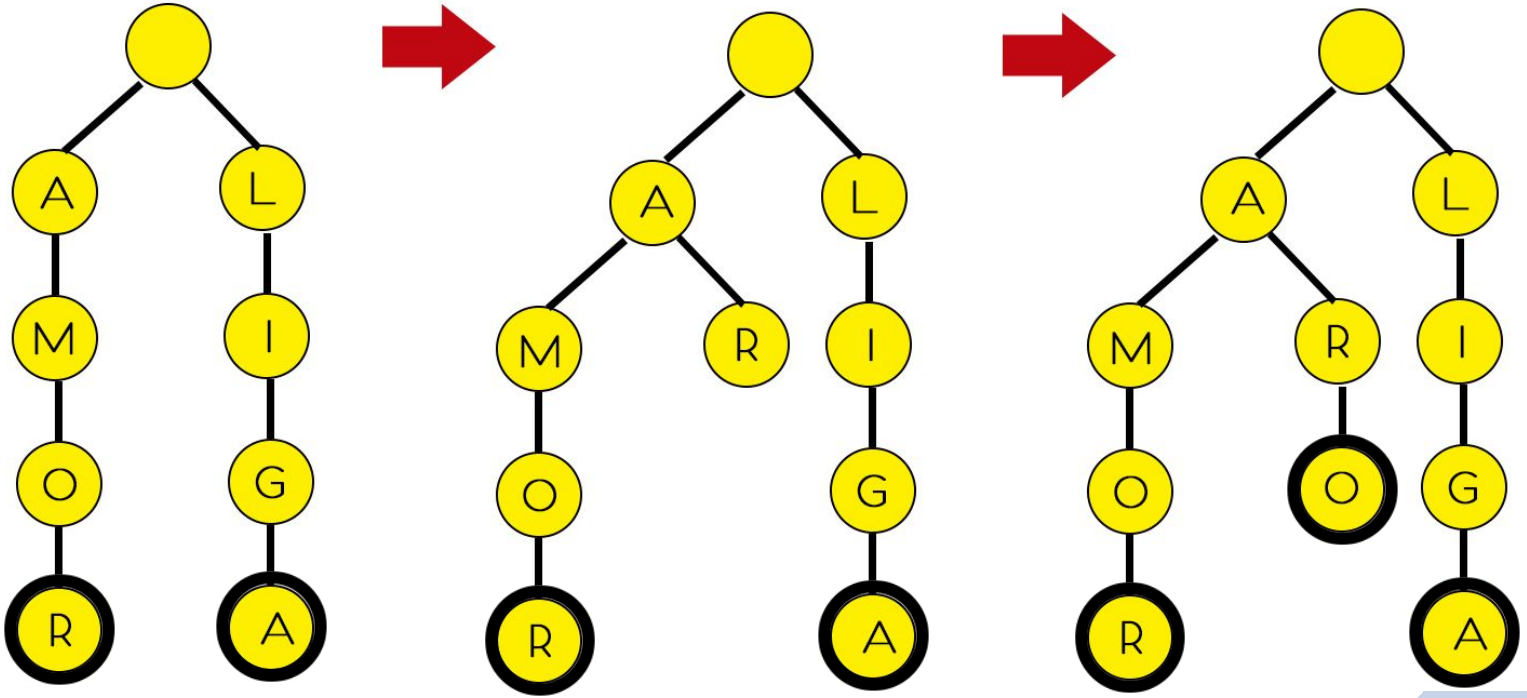
Inserção

- Inserção: **LIGA**



Inserção

- Inserção: **ARO**



Busca

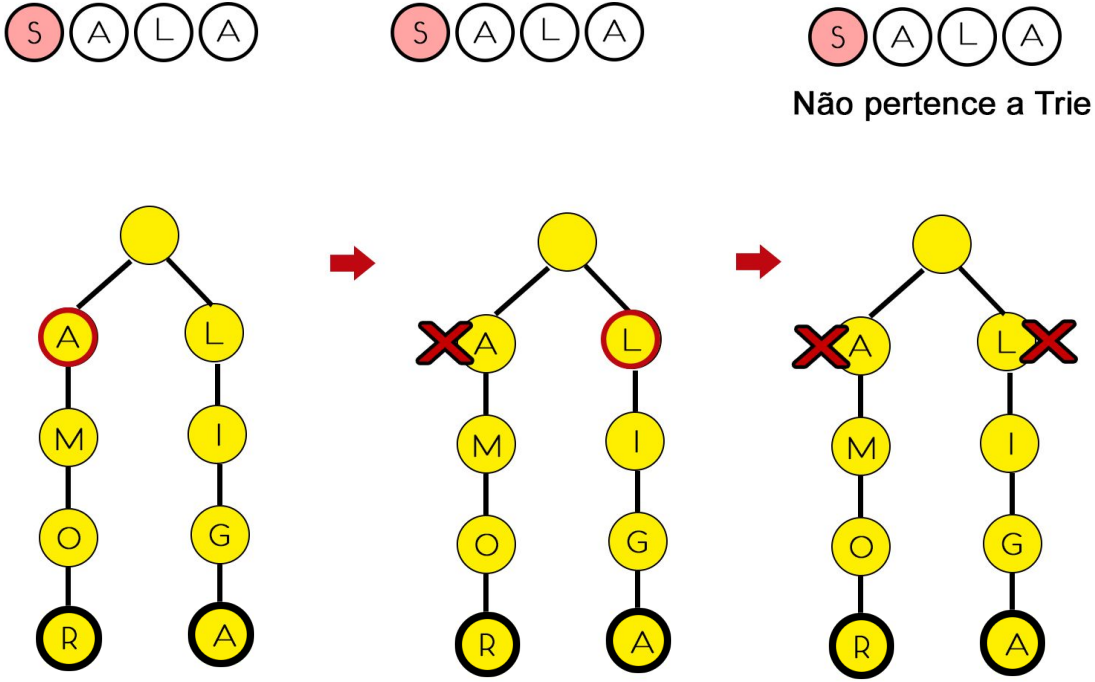
Para pesquisar por uma chave na árvore Trie é realizada uma busca caractere por caractere para verificar se a chave pesquisada pertence a Trie, seguindo o seguinte método:

- Caso o caractere inicial não pertence a árvore Trie, toda a chave passa a não pertencer a Trie;

- Caso o caractere pertença a Trie, o próximo caractere é verificado. Encontrando todos os caracteres, a chave pesquisada pertence a Trie.

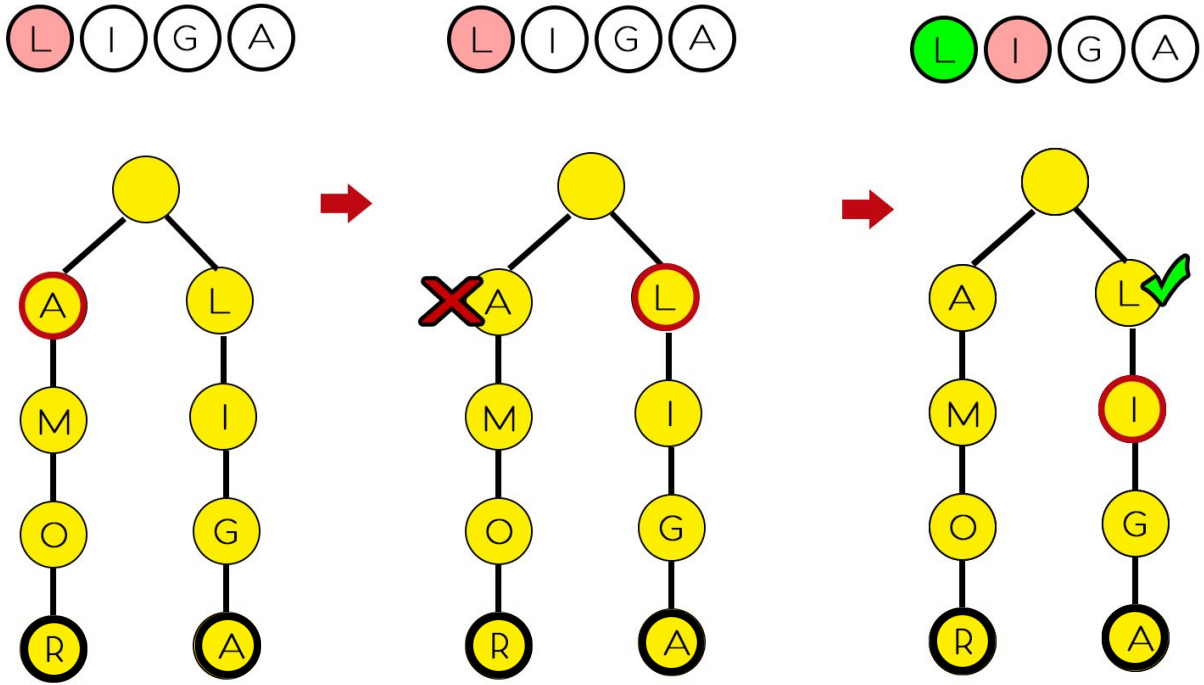
Busca

- Busca: **SALA**



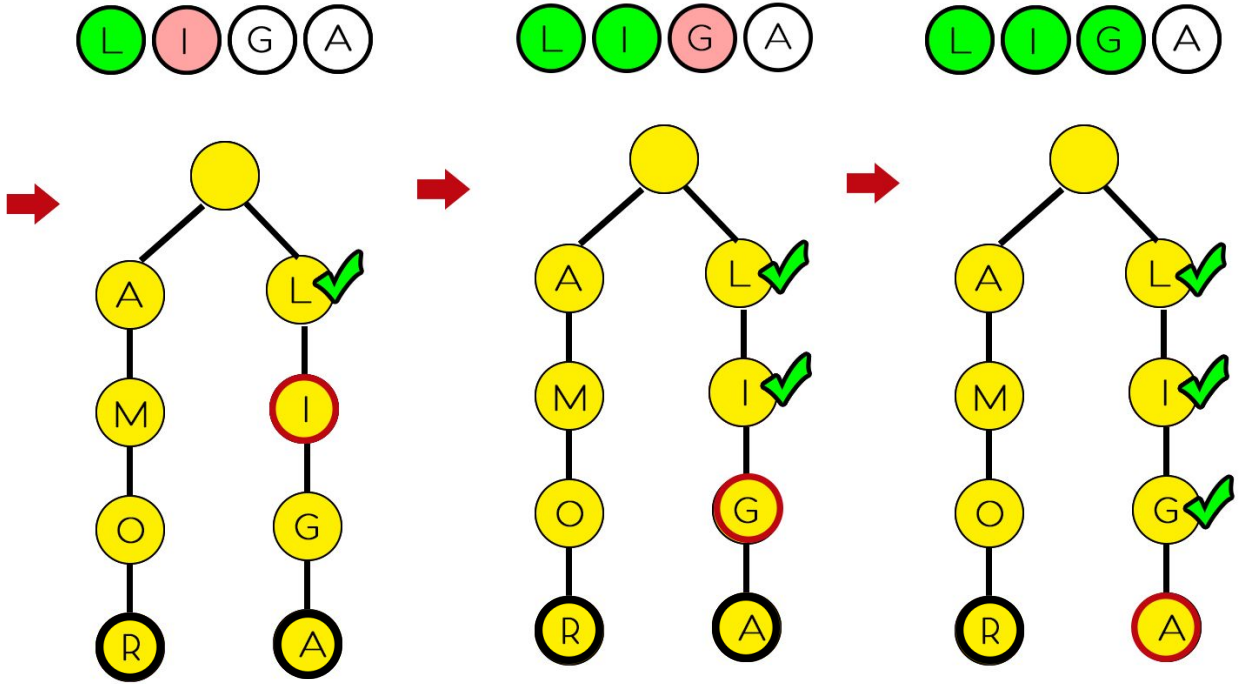
Busca

- Busca: **LIGA**



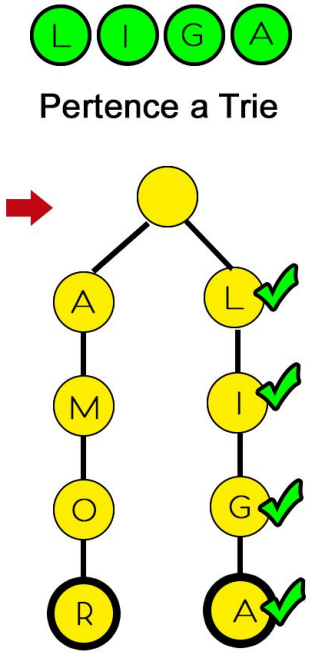
Busca

- Busca: **LIGA**



Busca

- Busca: **LIGA**



Remoção

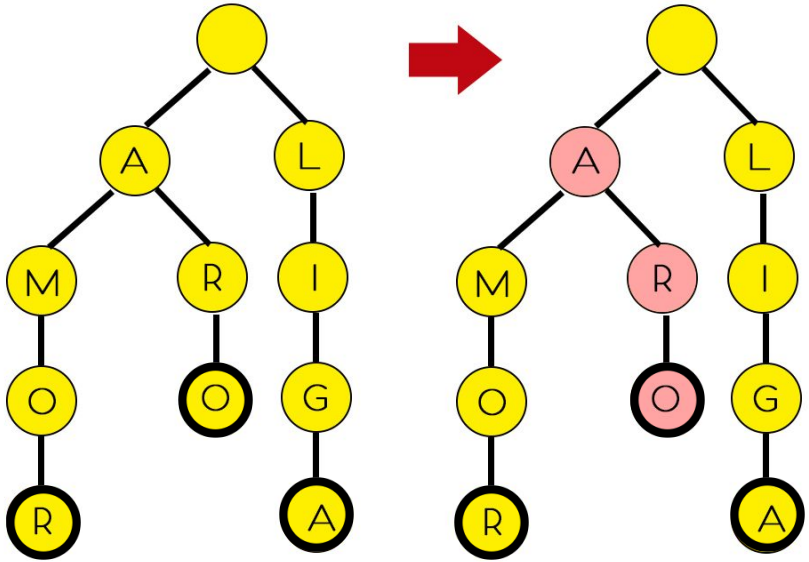
Para remover uma chave na árvore Trie é seguido o seguinte método:

- Realiza-se uma busca pelo nó que representa o final da palavra a ser removida.

- Realiza-se a remoção até uma das situações ocorrer:
  - Finaliza a remoção quando chegar ao nó raiz;
  - Finaliza a remoção ao encontrar um nó com mais de um filho;

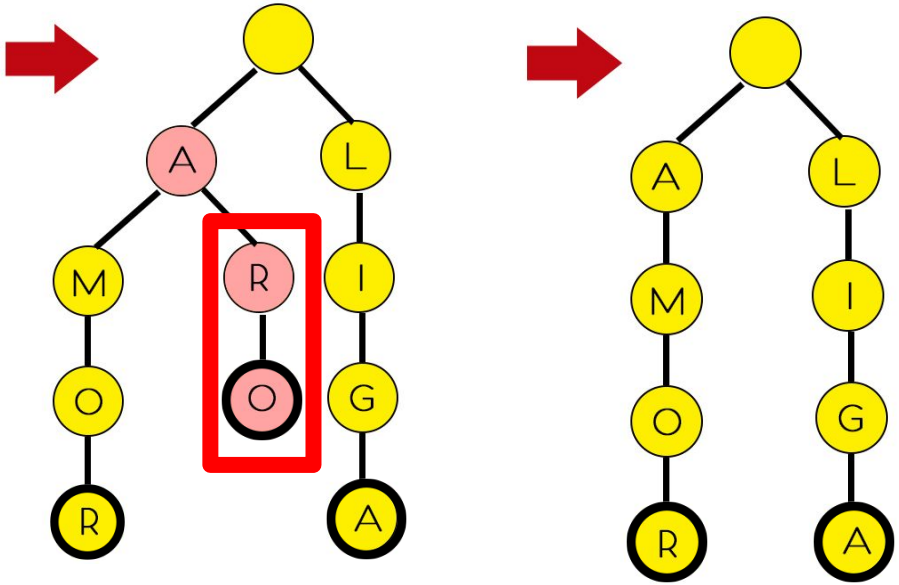
Remoção

- Remoção: **ARO**



Remoção

- Remoção: **ARO**



6

QUESTIONÁRIO

Questão 1

Comparando com outras estruturas de busca, qual a vantagem da Árvore Trie?

Questão 1 - Resposta

Na busca, a comparação é efetuada individualmente entre os dígitos que compõem as chaves (dígito a dígito).

Questão 2

Qual a vantagem da Árvore Trie em relação à Árvore Binária de Pesquisa?

Questão 2 - Resposta

Ao contrário de uma árvore de busca binária, nenhum nó desta árvore armazena a chave associada a ele; ao invés disso, ela é determinada pela sua posição na árvore.

Questão 3

Qual a complexidade da busca na Árvore Trie?

Questão 3 - Resposta

Em todos os casos será O(m), sendo m o tamanho da string.

Questão 4

Qual a altura da Árvore Trie?

Questão 4 - Resposta

O comprimento da chave mais longa.

Questão 5

Qual a complexidade da criação de uma Árvore Trie?

Questão 5 - Resposta

O(n*l), onde n é número de strings e l é o tamanho médio das strings: deve-se fazer l operações sobre as para cada string.

7

REFERÊNCIAS BIBLIOGRÁFICAS

- BUENO, Márcio. **Estrutura de Dados II - Árvores Trie e Patricia**. < https://marciobueno.com/arquivos/ensino/ed2/ED2_06_Trie_Patricia.pdf >
- SOUZA, Jairo F. **Busca Digital (Trie e Árvore Patricia)**. <http://www.ufjf.br/jairo_souza/files/2009/12/6-Strings-Pesquisa-Digital.pdf >
- CUNHA, Ítalo. **Pesquisa digital**. <http://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/radixsearch.pdf >
- SCHREINER, Marcos A. **Árvores Trie e Patricia**. <https://profschreiner.files.wordpress.com/2015/01/arvoredigital.pdf >
- TECHIE DELIGHT.**Trie Implementation in C | Insertion, Searching and Deletion**. <http://www.techiedelight.com/trie-implementation-insert-search-delete/ >