Árvores Trie e Patricia

Márcio Bueno
ed2tarde@marciobueno.com / ed2noite@marciobueno.com

Árvores Trie

- Definida em 1960 por Edward Fredkin
- Vêm de Re**trie**val (Relacionado à Recuperação de Informações)
- Para distinção com tree pronuncia-se try
- Cada nó contém informações sobre um ou mais símbolos do alfabeto utilizado
- O Alfabeto pode abranger: {0,1} , {A,B,C,D...} ou {0,1,2,3,4...} e mais o caracter nulo (ou branco)

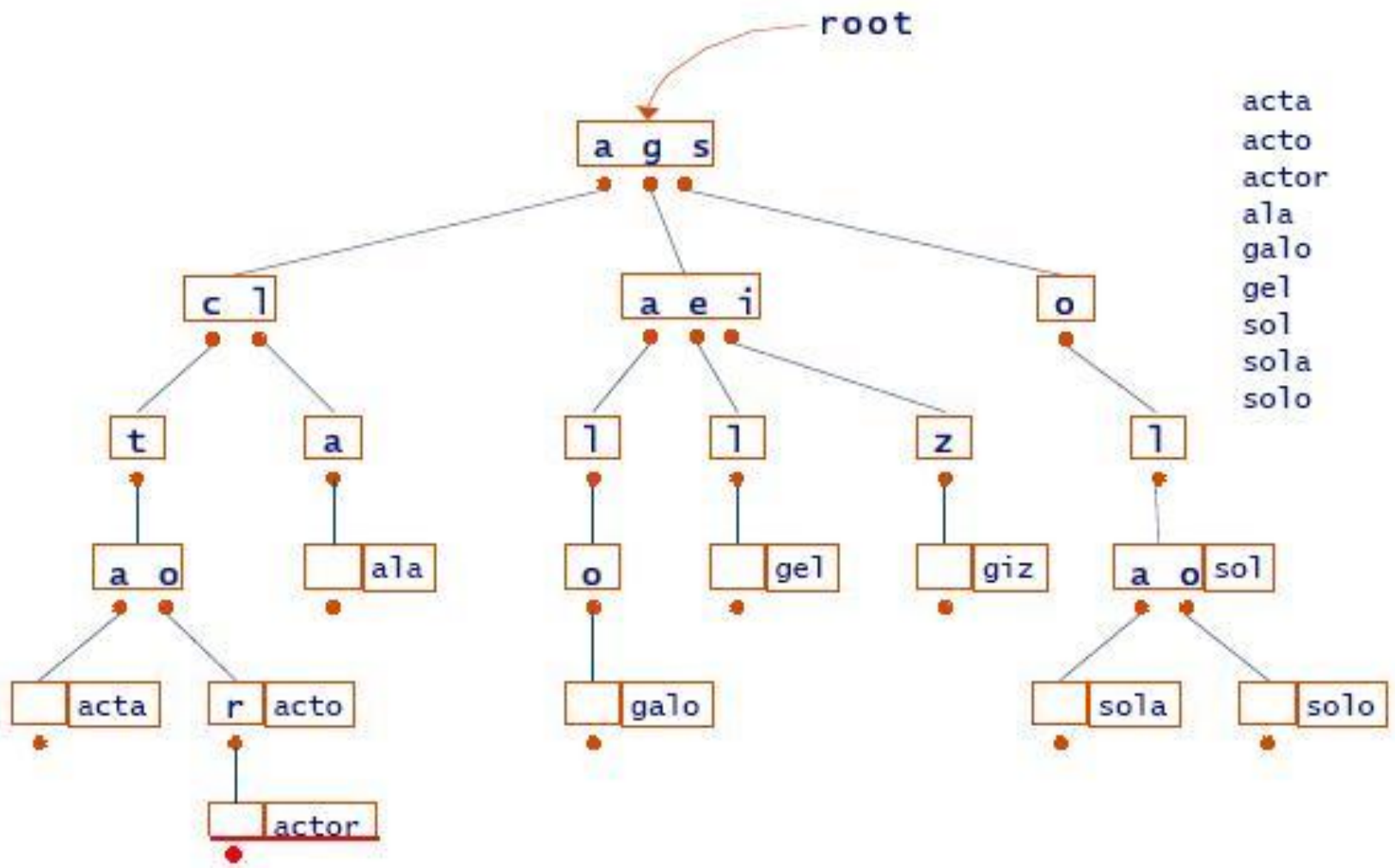
Árvores Trie

- As Tries são boas para suportar tarefas de tratamento lexicográfico, tais como:
  - manuseamento de dicionários;
  - pesquisas em textos de grande dimensão;
  - construção de índices de documentos;
  - expressões regulares (padrões de pesquisa).

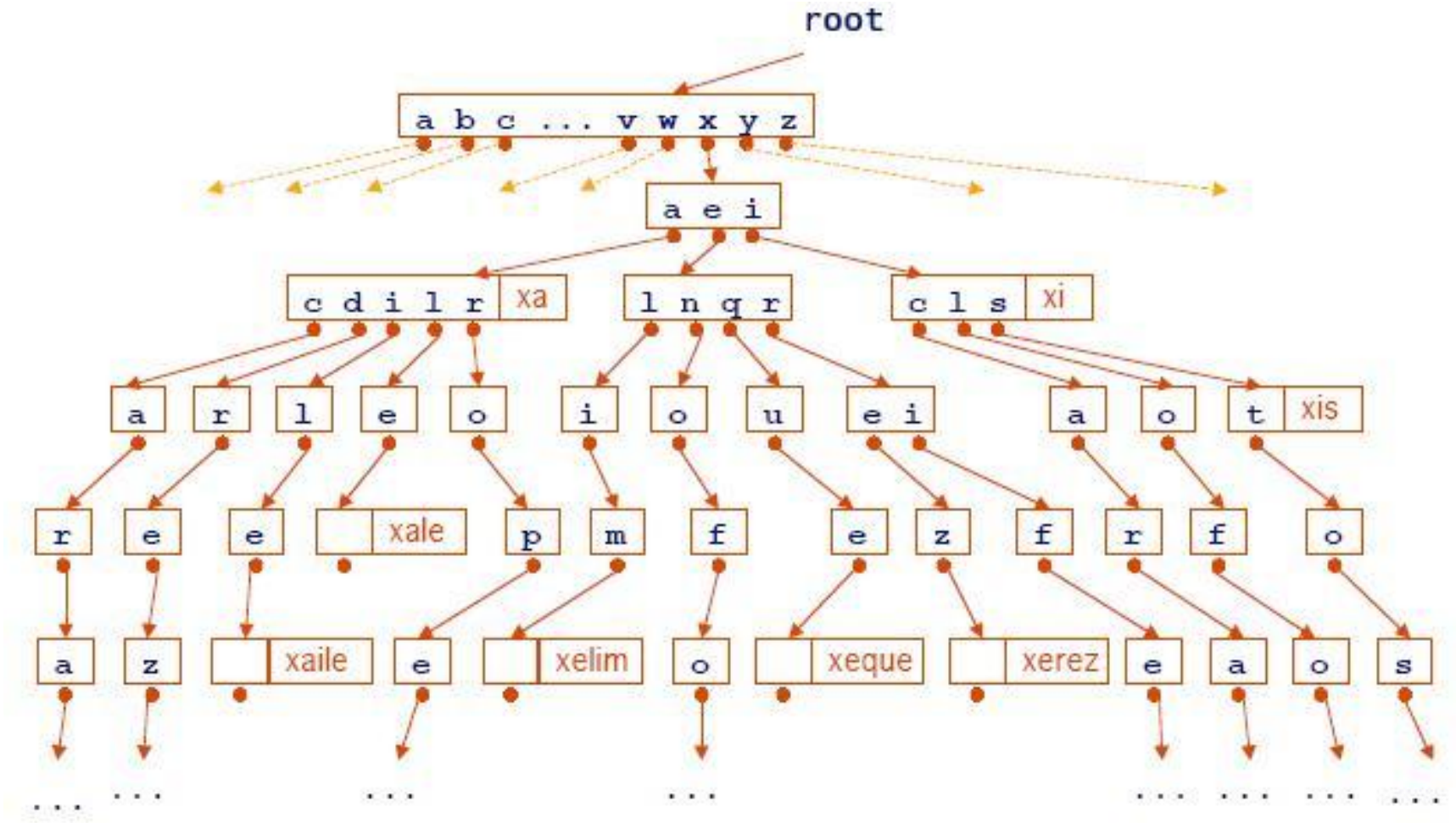
Árvores Trie

- O caminho da raiz (root) da trie para qualquer outro nó em representa um prefixo de uma string
- Em Tries Compactas todos os descendentes diretos do mesmo pai são agrupados
- No último nodo, o último caracter da palavra sendo procurada deverá ter associado a si (como seu apontador) a posição da palavra no texto

Árvores Trie



Árvores Trie



Árvores Trie

- Portanto:

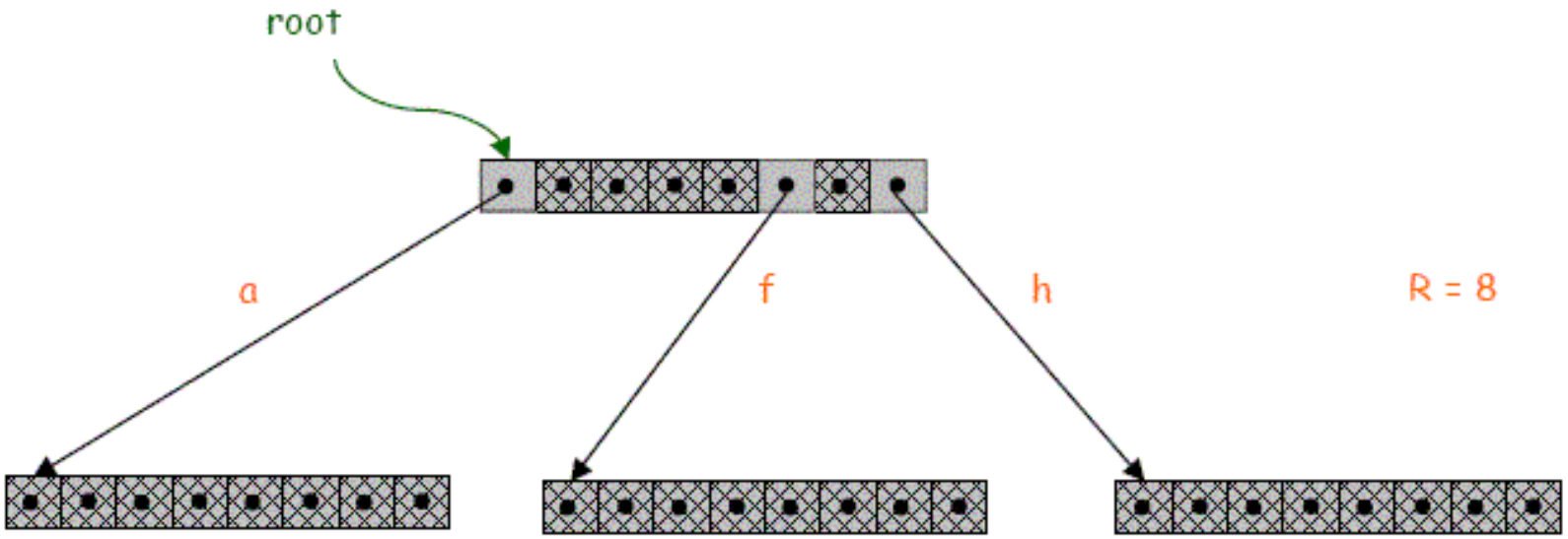
  Cada nível da árvore que se desce, corresponde a avançar um elemento na chave;

  Cada nó pode conter informação sobre um ou mais símbolos do alfabeto utilizado.

  Assim: uma dada sequência de arestas pode formar qualquer palavra (chave) possível com base nesse alfabeto; não existe limite para o tamanho de uma sequência (e portanto para o tamanho de uma chave); as sequências têm comprimento variável.

R-Way Trie

- Cada nó aloca espaço para todos os caracteres do alfabeto. Quase sempre há desperdício de espaço.



R = número de letras do alfabeto.

Aplicações de Trie

- **Busca**: localizar um dado que corresponde a chave informada;
- **Problema:** seria em um sistema de cadastros de pessoas, onde quando temos nomes com grafias semelhantes (Manuel/Manoel, Elaine/Elayne, Luis/Luiz), podem ocorrer erros na entrada desses dados, ou seja, de não ser que sejam testados os possíveis erros cometidos.

Aplicações de Trie

- **Solução do problema**: 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.

Aplicações de Trie: Corretor Ortográfico

- **Aplicação usual de Trie** é o corretor ortográfico. Nesse tipo de programa as palavras são comparadas com um dicionário armazenado em arquivo, e se não são encontradas indica-se as opões para correção.

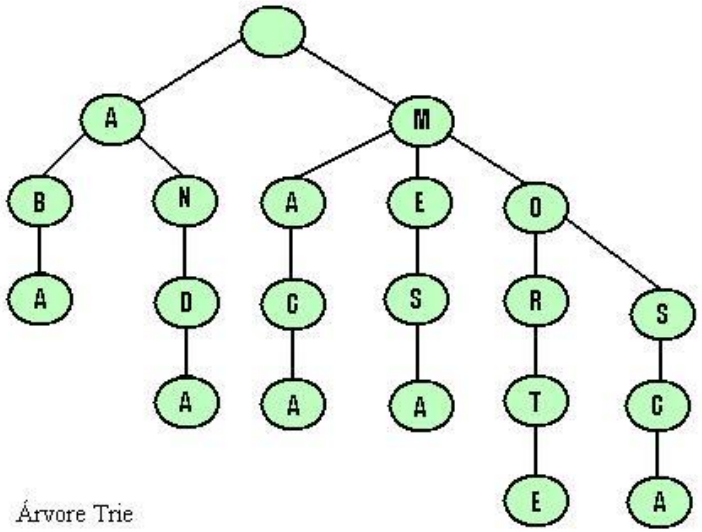
Aplicações de Trie: Corretor Ortográfico

- 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 tipos de erros para poder indicar as opções de correção.

Aplicações de Trie: Corretor Ortográfico

1. **Substituição -** avança um caracter na chave e avança um nível na árvore;
2. **Deleção -** avança um nível na árvore;
3. **Inserção -** avança um caracter na chave;
4. **Transposição -** avança um nível na árvore e testa a posição atual da chave, se coincidir, avança um caracter na chave e retrocede um nível na árvore para confirmar a inversão.

Aplicações de Trie: Corretor Ortográfico



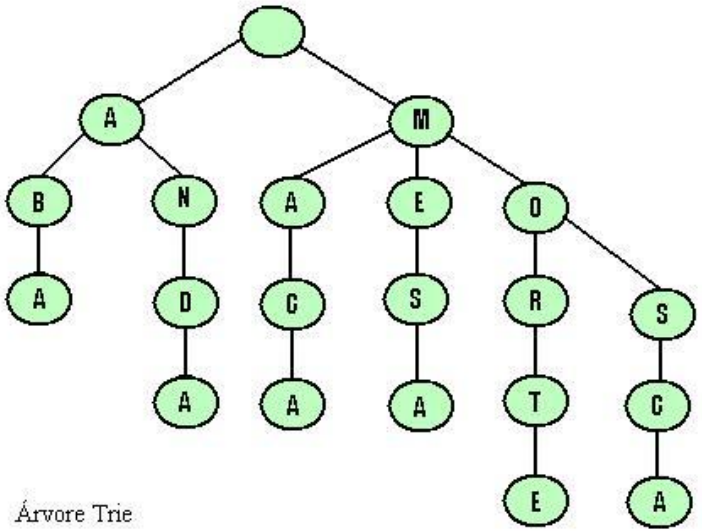
Árvore Trie

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

Aplicações de Trie: Corretor Ortográfico



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)

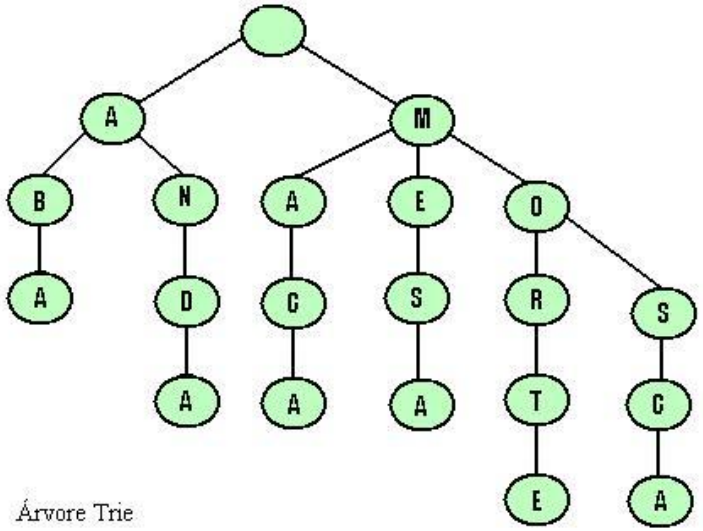
* D = A - erro * D = D - ok * A = A - ok

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

Aplicações de Trie: Auto-Preenchimento

Armazena palavras mais
usadas em uma **TRIE**
A medida que vai
digitando exibe as opções
possíveis de palavras já usadas

Aplicações de Trie: Auto-Preenchimento



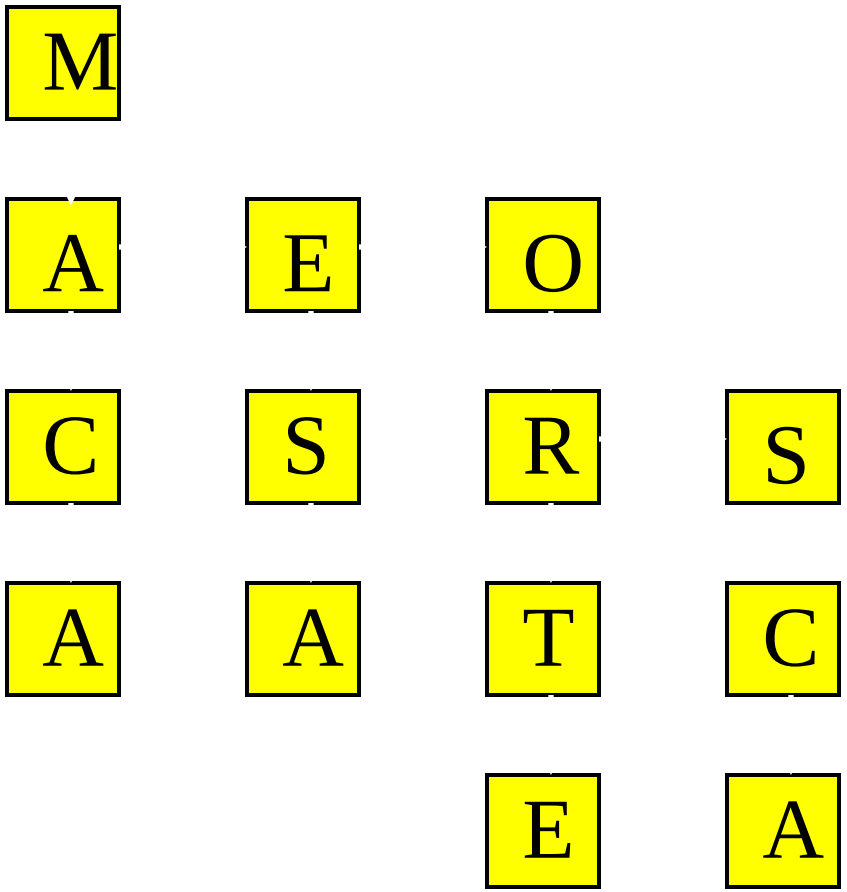
Árvore Trie

Utilização desta aplicação: **Browsers; Programas de e-mail**: Gmail e o Yahoo! mail, até **linguagens de programação.**

Série de endereços que já foram usados (browser/e-mail) ou os comandos disponíves (linguagem de programação).

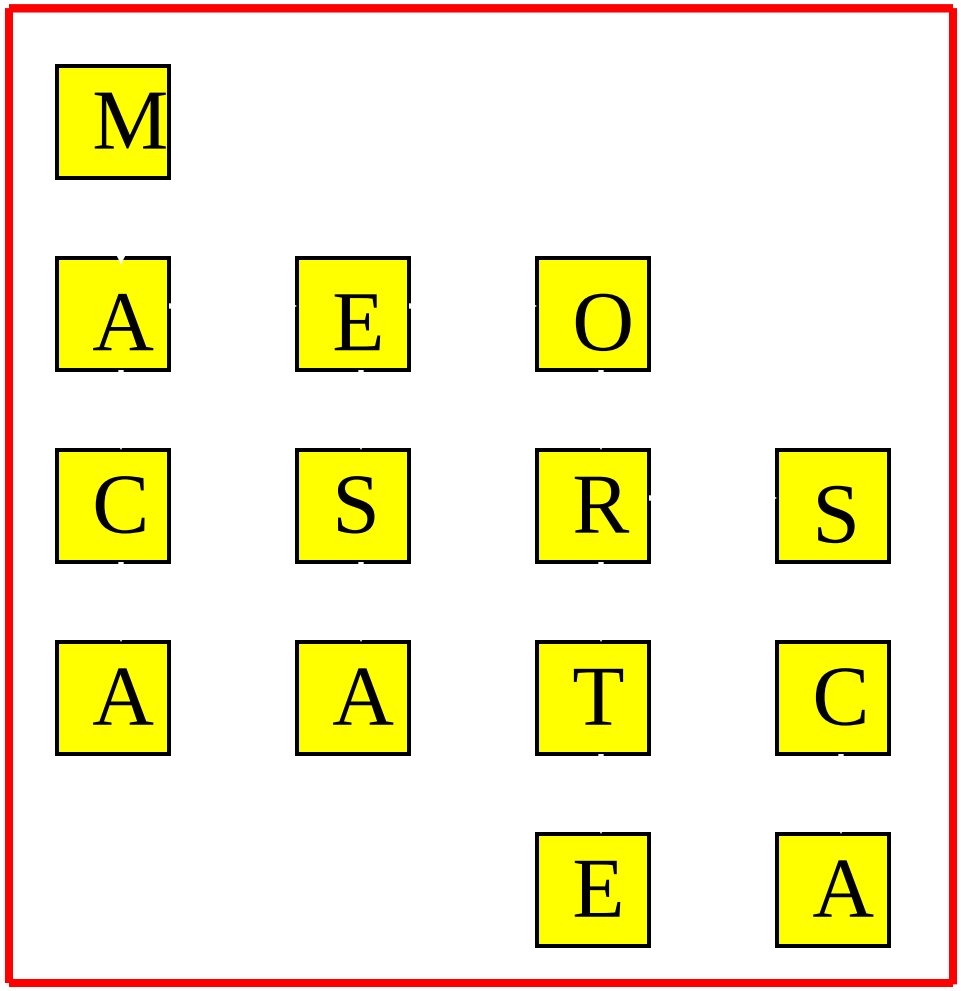
São armazenados em tries e a medida que é digitada uma seqüência de caracteres o algoritmo vai comparando a existência de correspondências na estrutura. A cada caractere digitado, são apresentadas as opções de preenchimento, e no momento em que só existir um caminho possível a ser seguido na trie ocorre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

Aplicações de Trie: Auto-Preenchimento



M O S C A

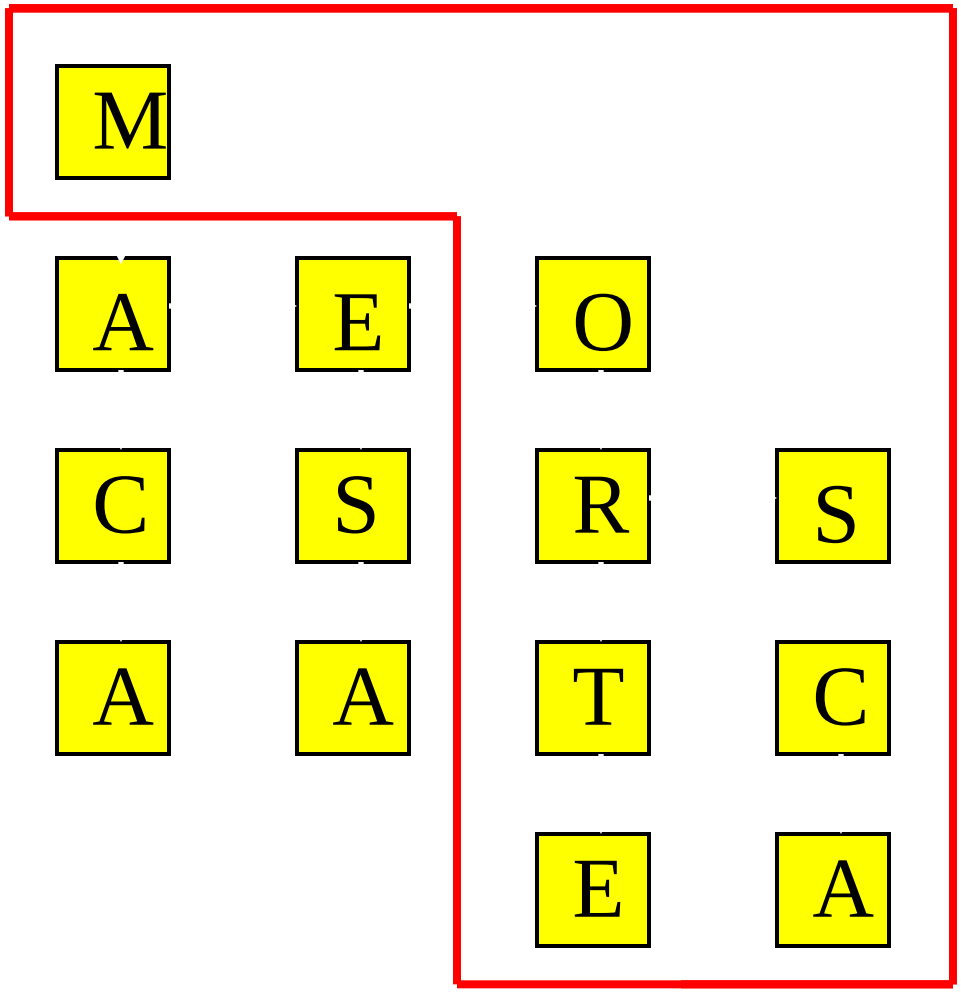
Aplicações de Trie: Auto-Preenchimento



M O S C A

maca
mesa
morte
mosca

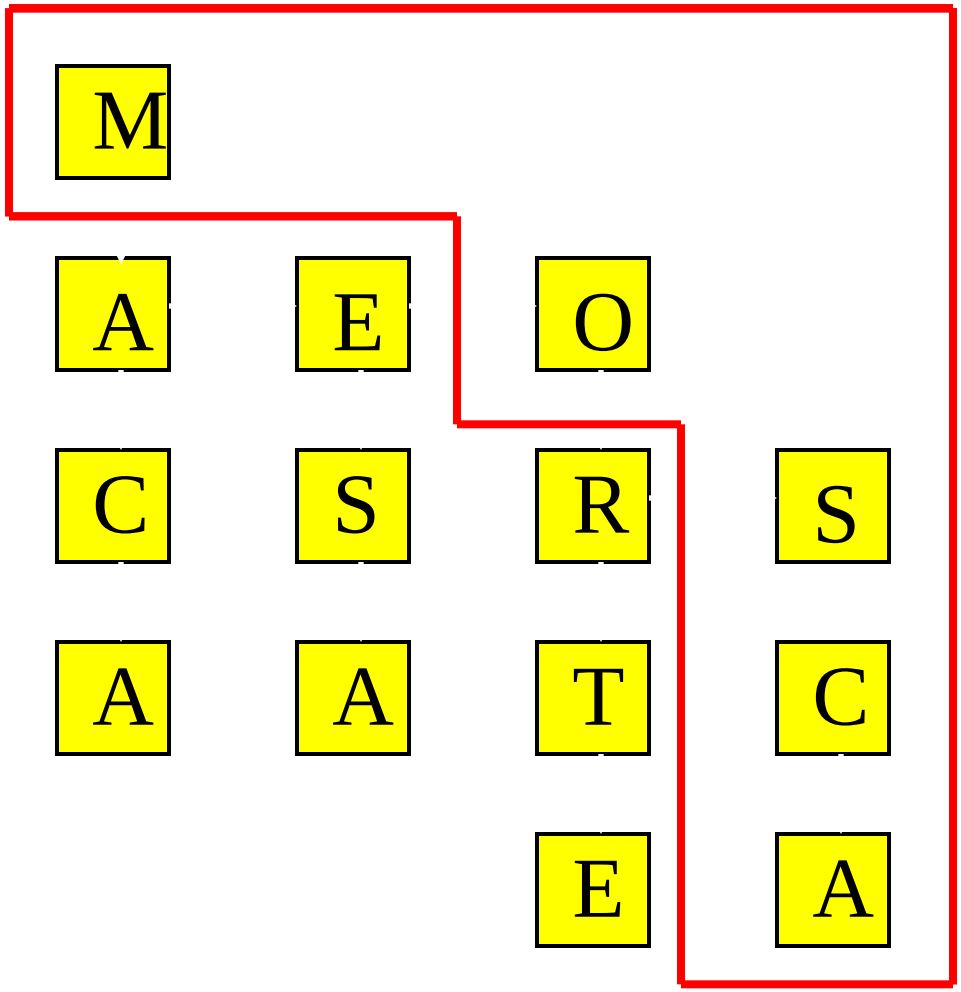
Aplicações de Trie: Auto-Preenchimento



M O S C A

morte
mosca

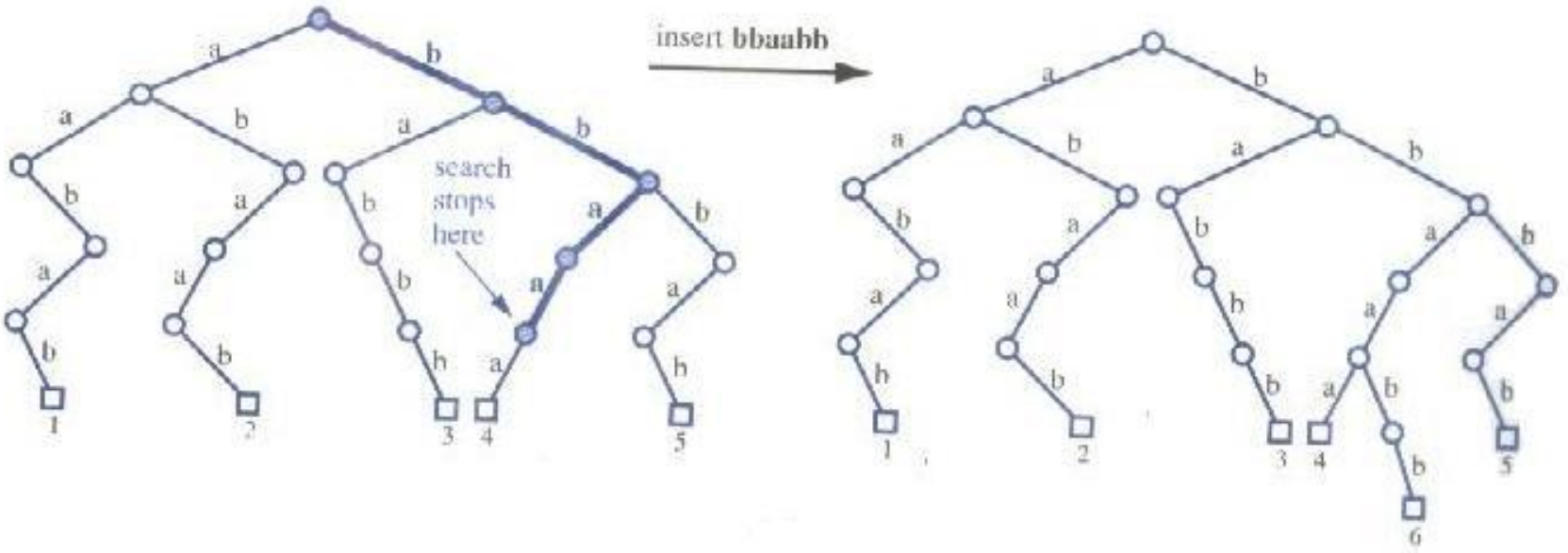
Aplicações de Trie: Auto-Preenchimento



M O S C A

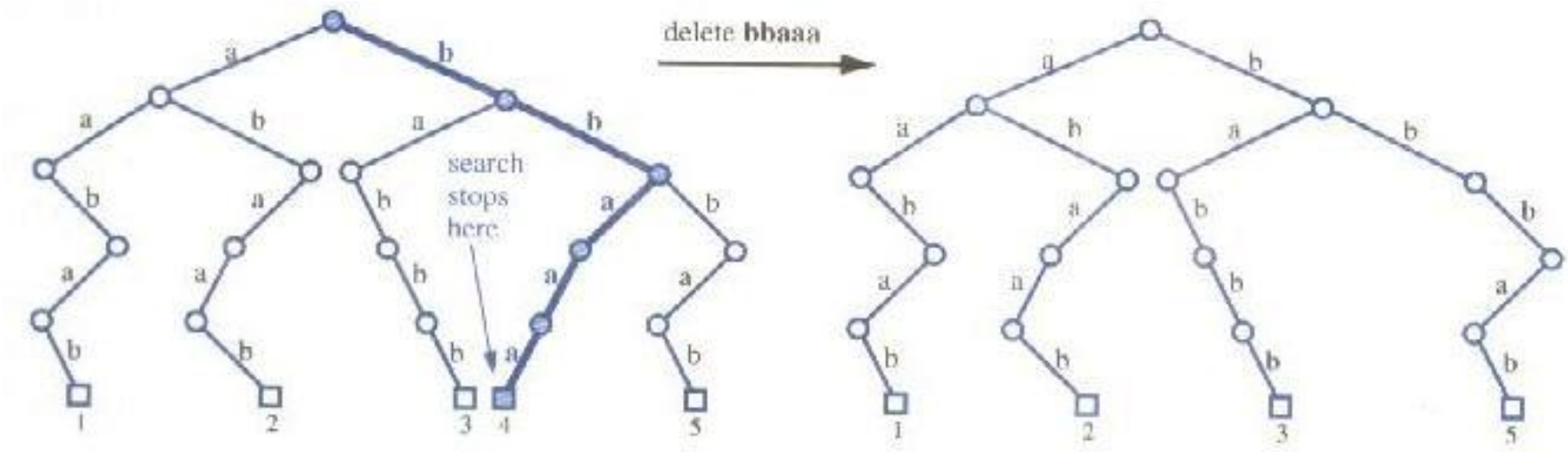
Árvores Trie – Inserção

É feita uma busca pela palavra a ser inserida, se ela já existir na trie nada é feito, caso contrário, é recuperado o nó até onde acontece a maior substring da palavra a ser inserida, sendo o restante dos seus caracteres (palavra - prefixo) adicionados na trie a partir daquele nó.



Árvores Trie - Remoção

Tendo a busca encontrado o nó que representa o final da palavra a ser removida, são removidos os nós que possuem apenas um filho seguindo o caminho ascendente. A remoção é concluída quando se encontra um nó com mais de um filho.



Árvores PATRICIA

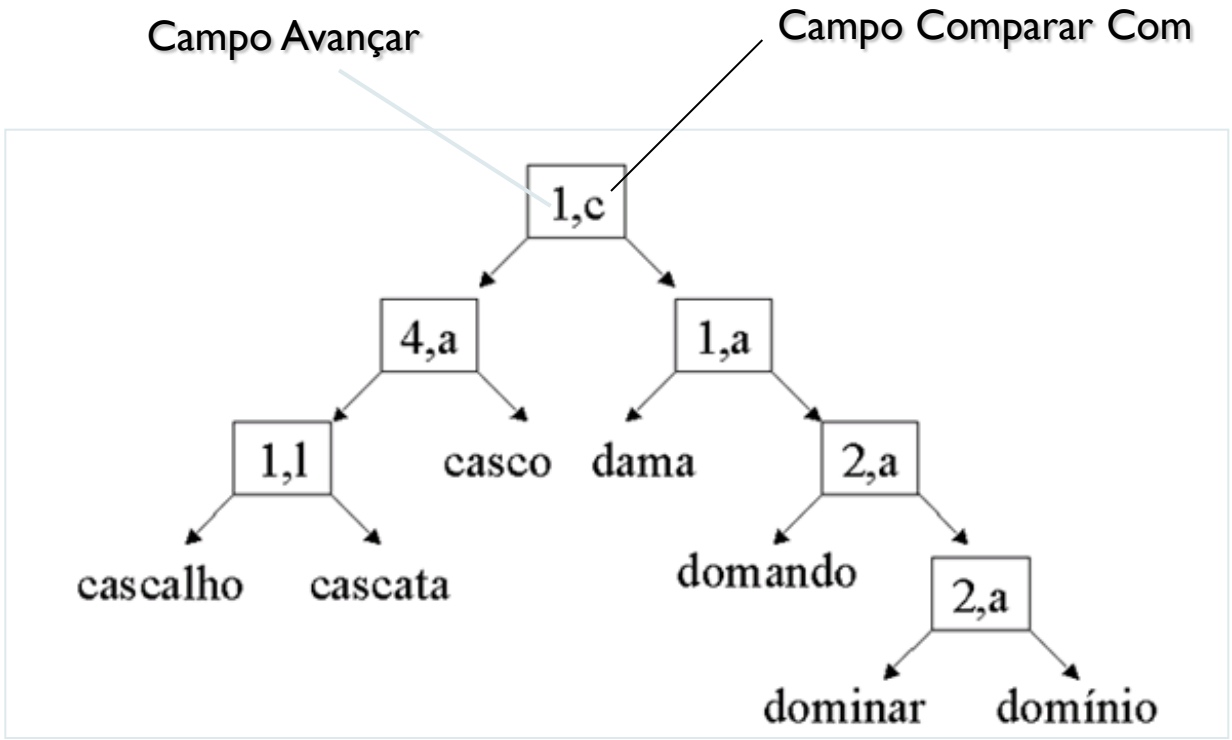
Pratical
Algorithm
To
Retrieve
Information
Coded
In
Alphanumeric

Árvores PATRICIA

- Definida em 1968 por Donald R. Morrison
- Trie Compactada Binária
- Caminhos que possuem nós com apenas 1 filho são agrupados em uma única aresta
- Diferente das Tries não armazena informações nos nodos internos, apenas contadores e ponteiros para cada sub-árvore descendente.

Árvores PATRICIA

Exemplo de Representação ::



Árvores PATRICIA

- Exemplo de Representação

- **Campo Avançar**

✓ Registro Acumulativo que Integra todos os Nodos Exceto os Folhas

✓ Identifica qual a Posição do Caracter da Chave Informada que deve ser analisado

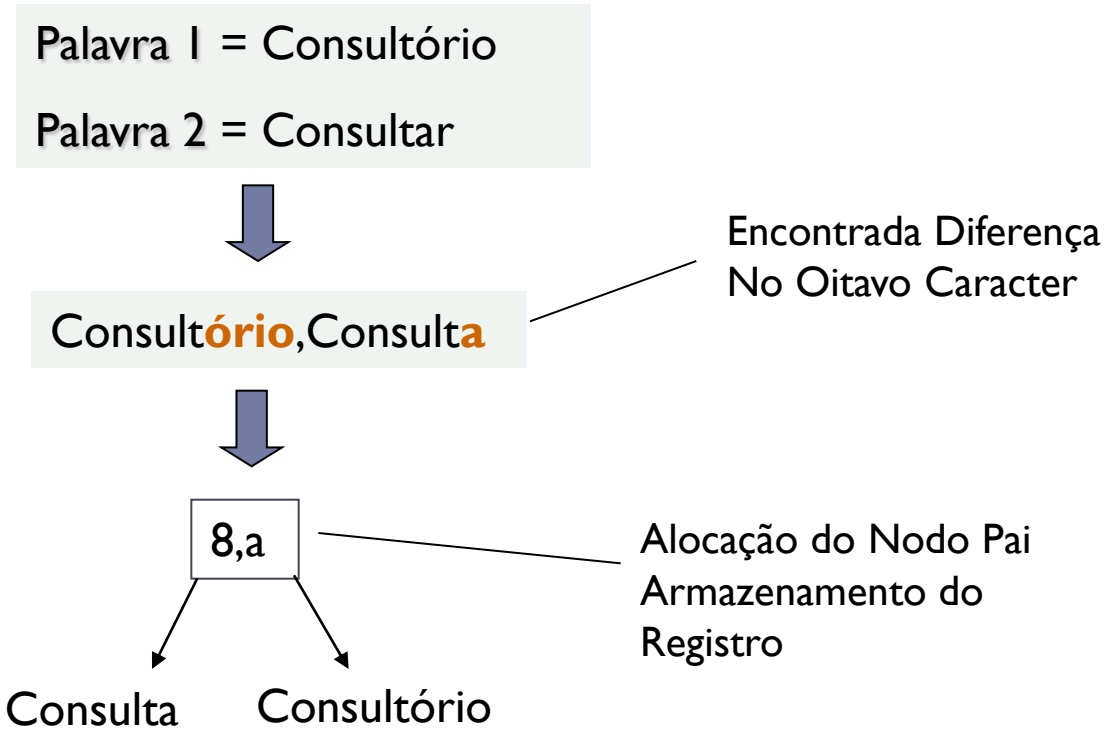
- **Campo Comparar Com**

✓ Apresenta o Caracter que deve ser Comparado ao Caracter da Chave Informada

✓ Como nas Árvores Binárias de Busca, após a análise, se a Chave é Menor ou Igual ao Nodo ela é Alocada/Consultada à Esquerda senão à Direita

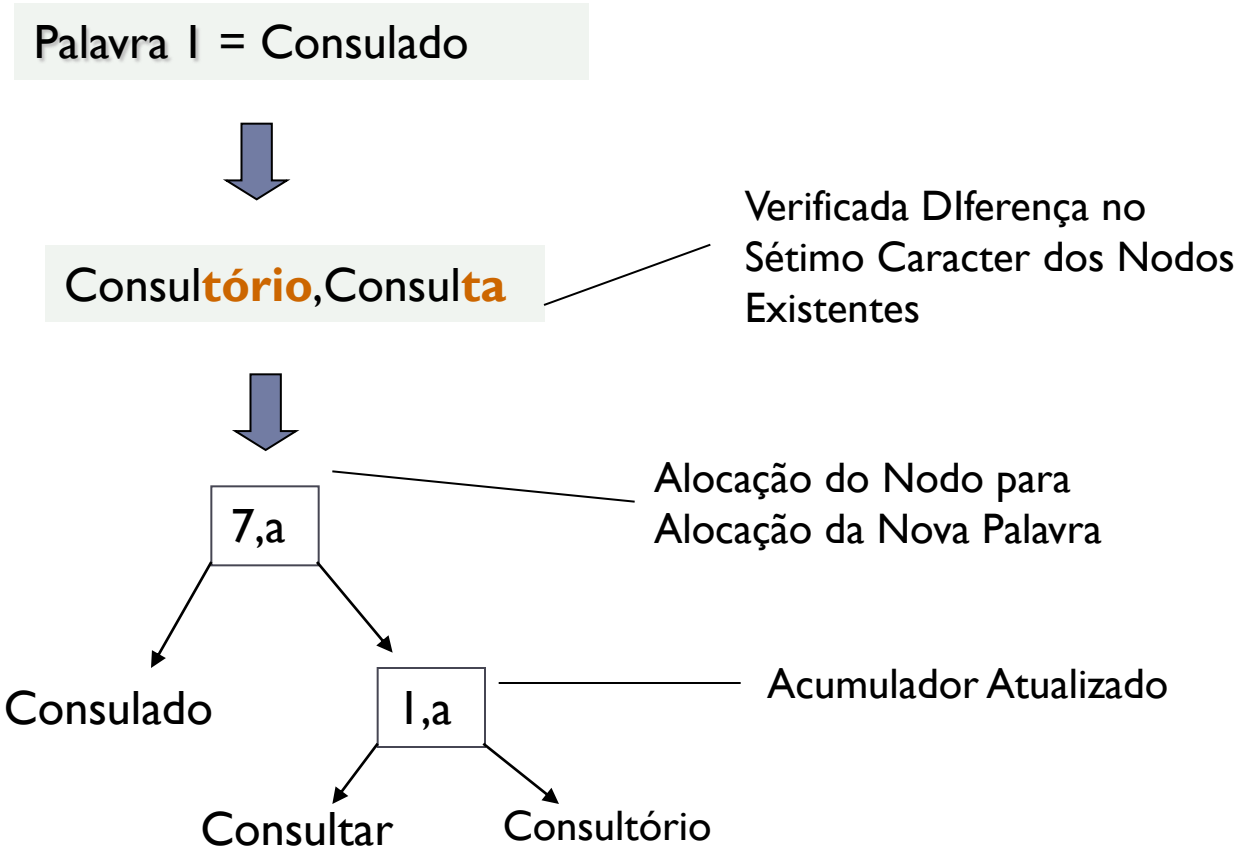
Árvores PATRICIA

Exemplo de Inserção ::



Árvores PATRICIA

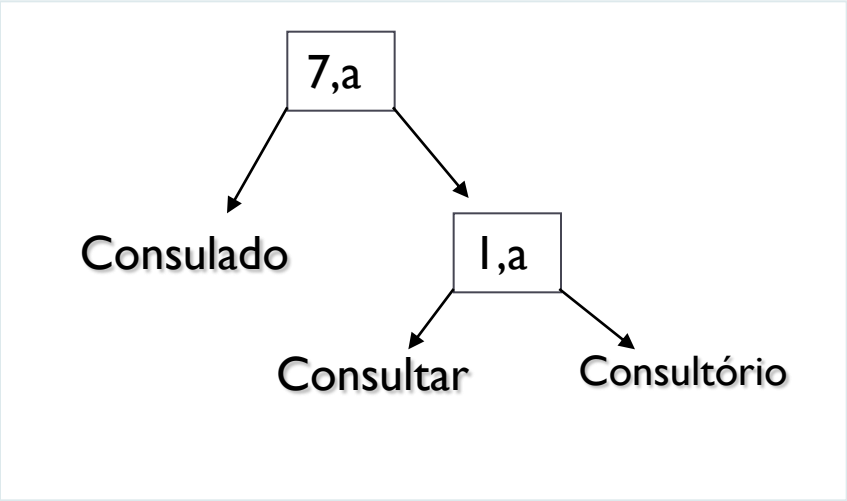
Exemplo de Inserção 2 ::



Árvores PATRICIA

Exemplo de Consulta ::

Busca por **"Consultório"**



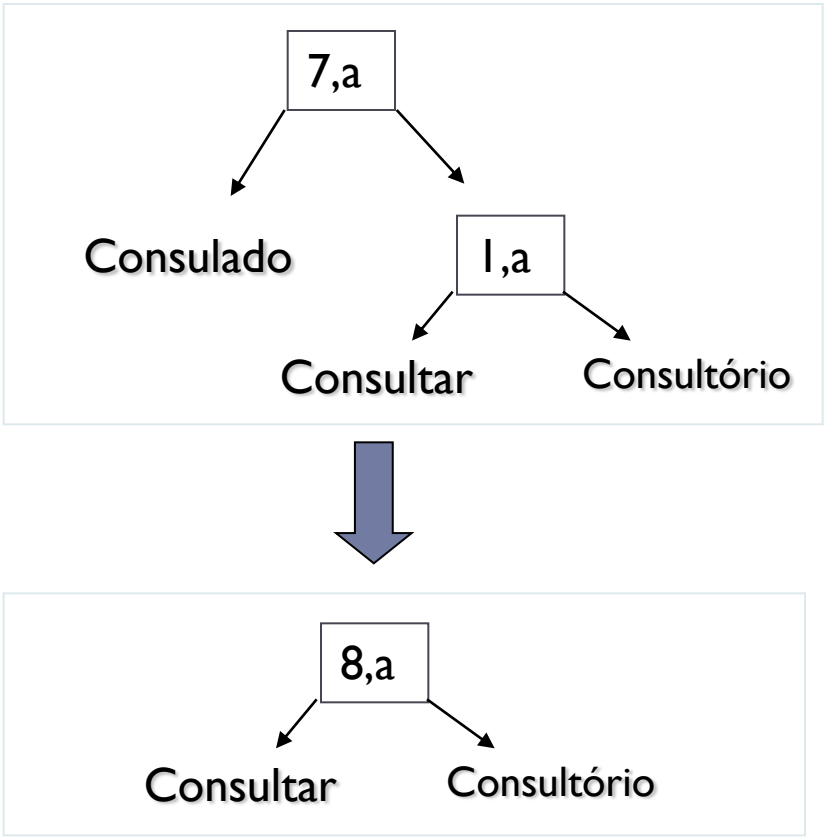
Etapas:

1) Primeiro Nodo Informa pra Comparar 7° Caracter da Palavra com "a".
2) Como "t" é maior que "a" desloca-se pra sub-árvore da direita.
3) Compara-se agora o Caracter 8° Caracter da Chave com "a"
4) Como "ó" é maior que a ele percorre a sub-árvore da direita e acha a palavra.

Árvores PATRICIA

Exemplo de Deleção ::

Apagar "**Consulado**"



Etapas:

1) Primeiro Busca-se e Apaga-se a Palavra "Consulado" da Árvore
2) Soma-se o valor do Campo Avançar do Nó Pai a Todos os nós FIlhos