

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Programa de Pós-Graduação em Ciência da Computação

ÁRVORES TRIE E PATRICIA

Nelson Cruz Sampaio Neto
nelsonneto@ufpa.br

18 de setembro de 2018

Introdução

- Definida em 1960 por Edward Fredkin.
- TRIE vem de “RE**TRIE**VAL” (recuperação).
- A pronúncia: “tri” ou “traí”.
- Também conhecida como **árvore digital**, é um tipo de estrutura onde partes das chaves são usadas na definição do caminho.
- Muito utilizada para armazenar cadeias de caracteres e suportar uma rápida procura de padrões.

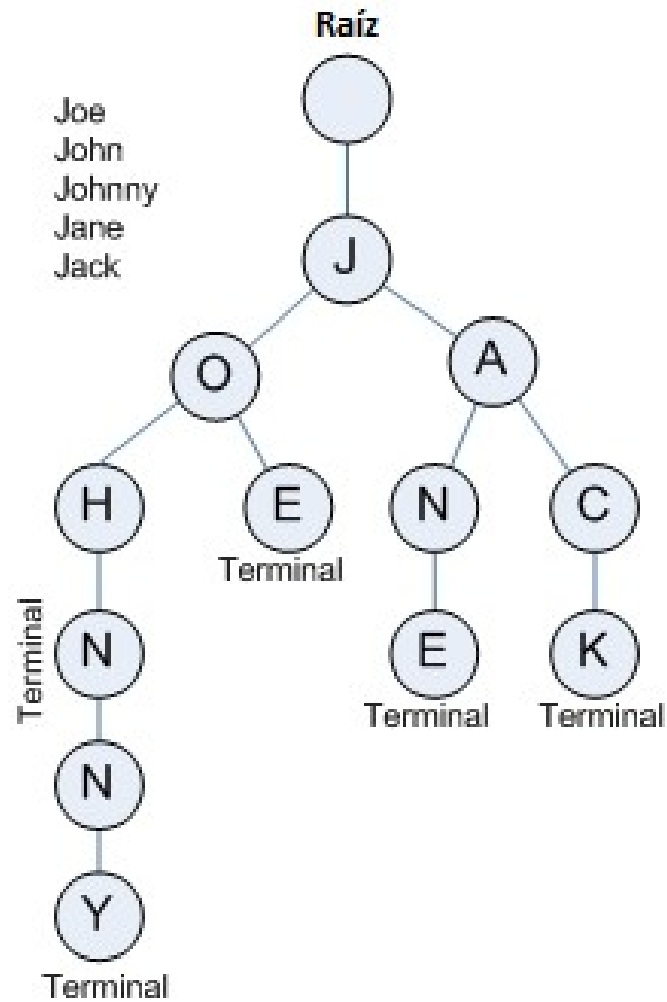
Introdução

- Na pesquisa digital, as chaves são formadas por um conjunto de dígitos ou caracteres sobre um alfabeto.
- As chaves têm tamanho variável e sem limitação explícita quanto ao tamanho.
- Exemplos de alfabetos: $\{0,1,2,3,4...\}$, $\{A,B,C,D,E,F...\}$, $\{0,1\}$.
- Exemplos de chaves:
ABABBBABABA 19034717 Maria
010101010000000000101000000001010

Introdução

- A diferença entre a busca digital e a busca estudada até agora é que a chave não é tratada como um elemento indivisível.
- Em vez de se comparar a chave procurada com as chaves do conjunto armazenado, a comparação é efetuada entre os dígitos que compõem as chaves, dígito a dígito.
- O método de pesquisa digital é análogo à pesquisa manual em dicionários: com a primeira letra da palavra são determinadas todas as páginas que contêm as palavras iniciadas por aquela letra e assim por diante.

Exemplo



Exemplo

Chaves de 6 *bits* :

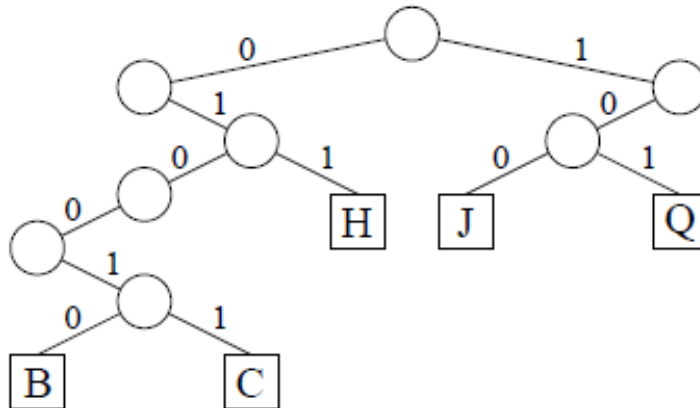
B = 010010

C = 010011

H = 011000

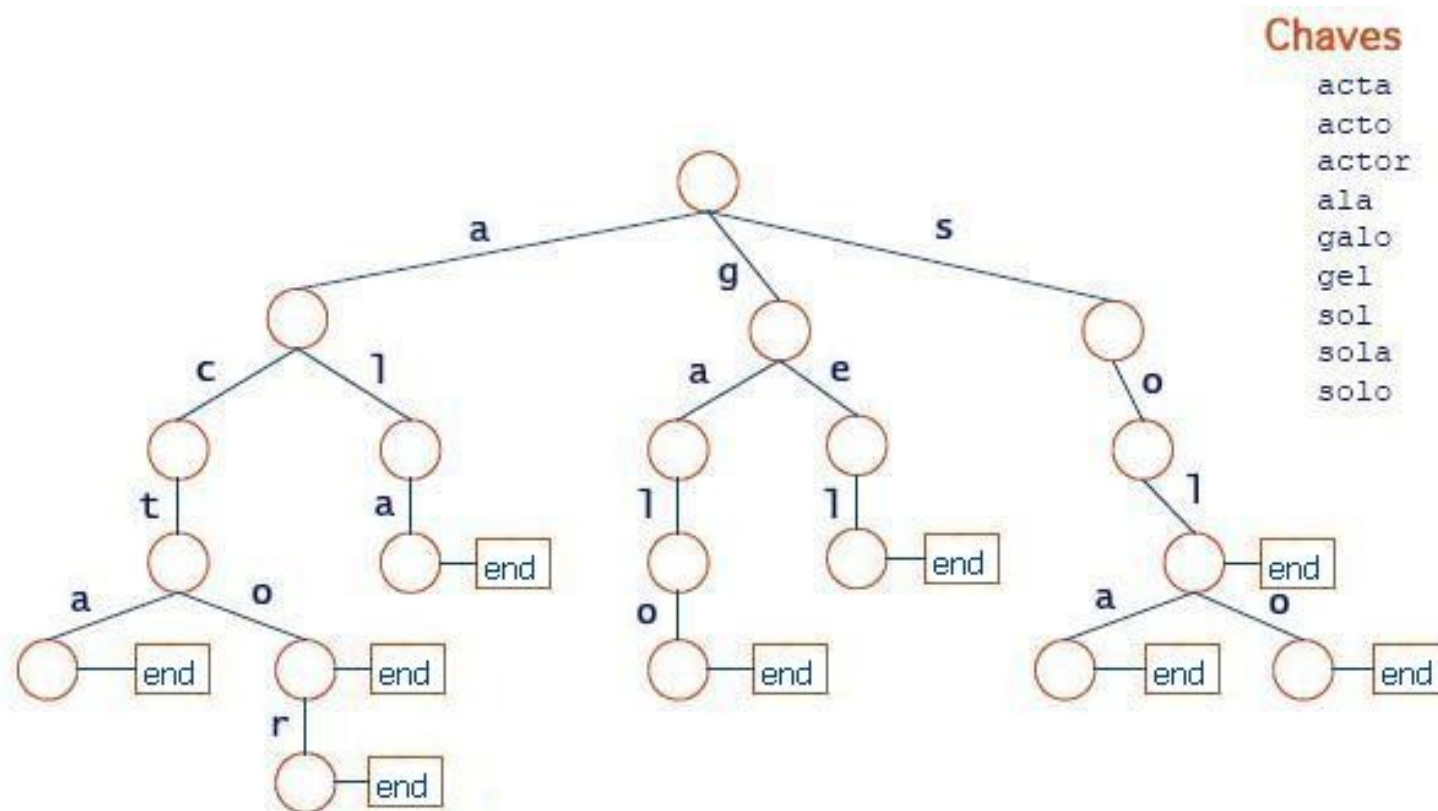
J = 100001

Q = 101000



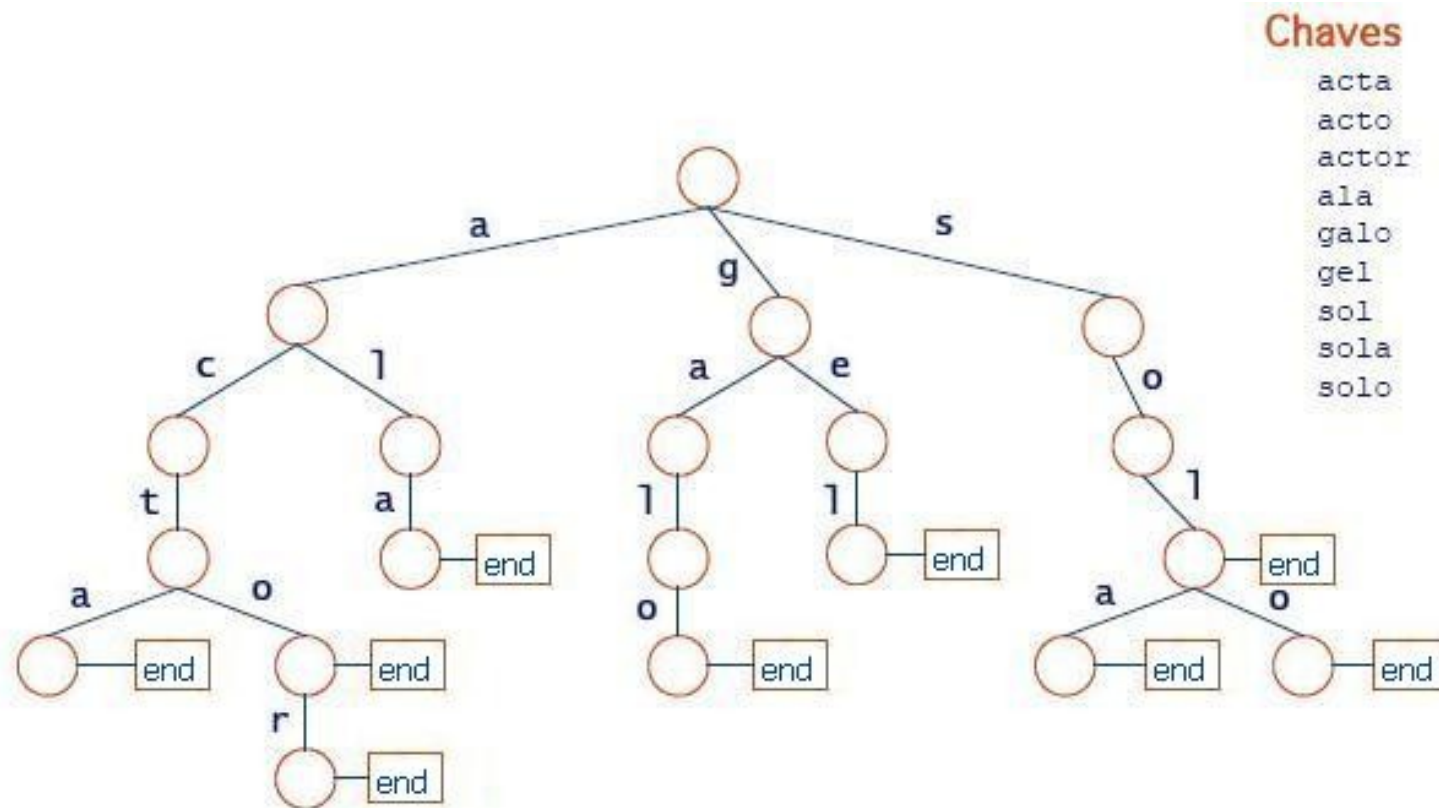
Estrutura de uma TRIE

- Árvore ordenada e n-ária.



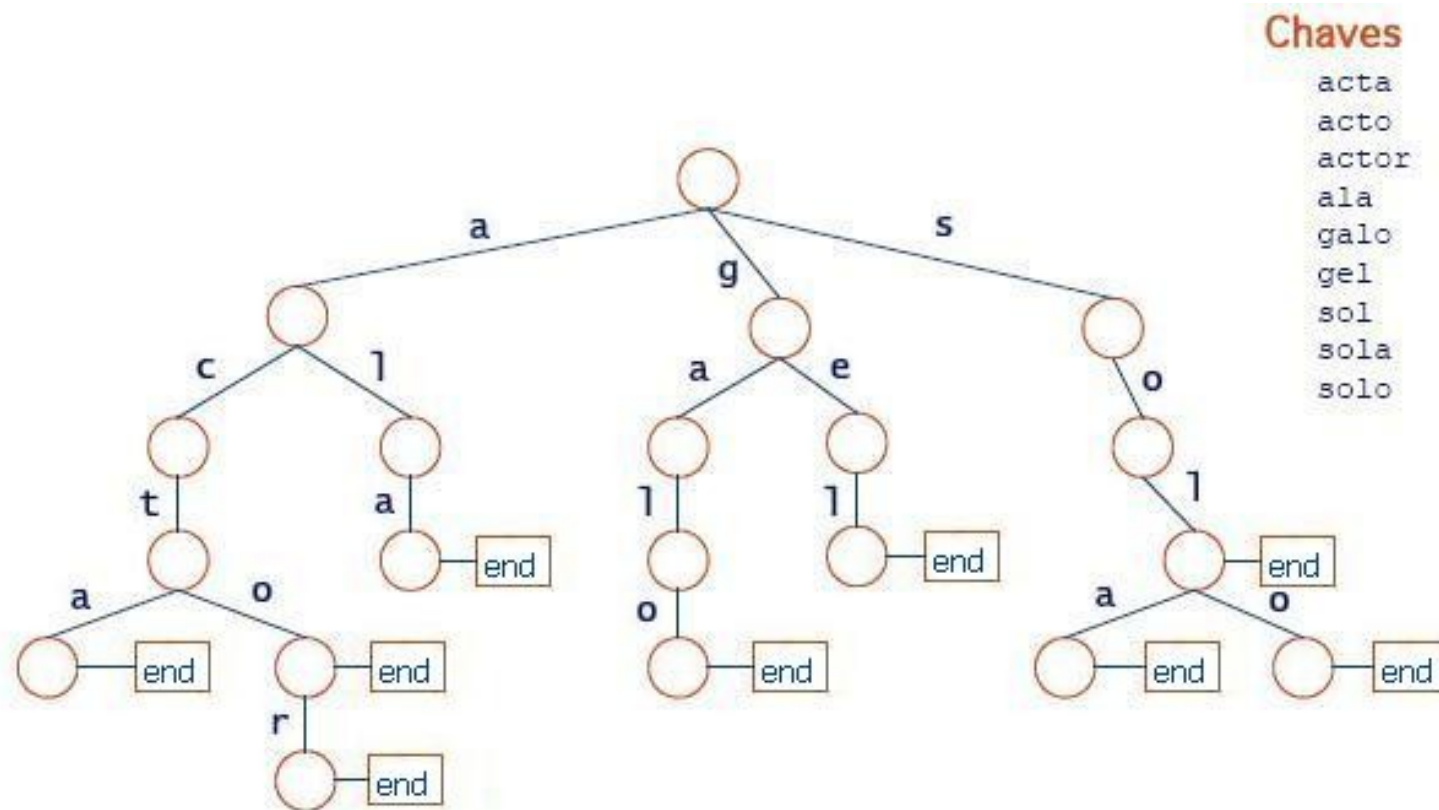
Estrutura de uma TRIE

- Raiz: cadeia vazia.



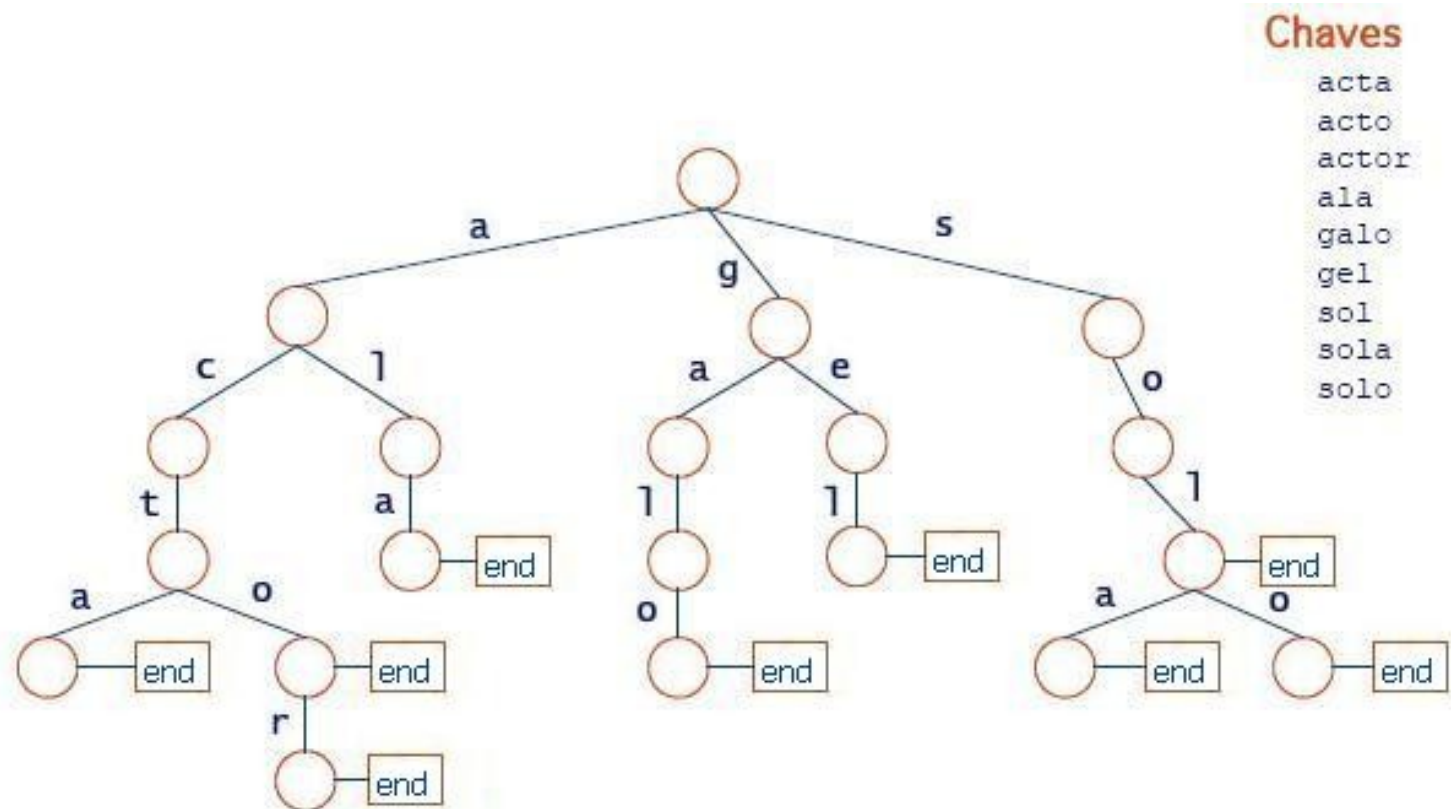
Estrutura de uma TRIE

- O caminho da raiz para qualquer outro nó é um prefixo de uma string.



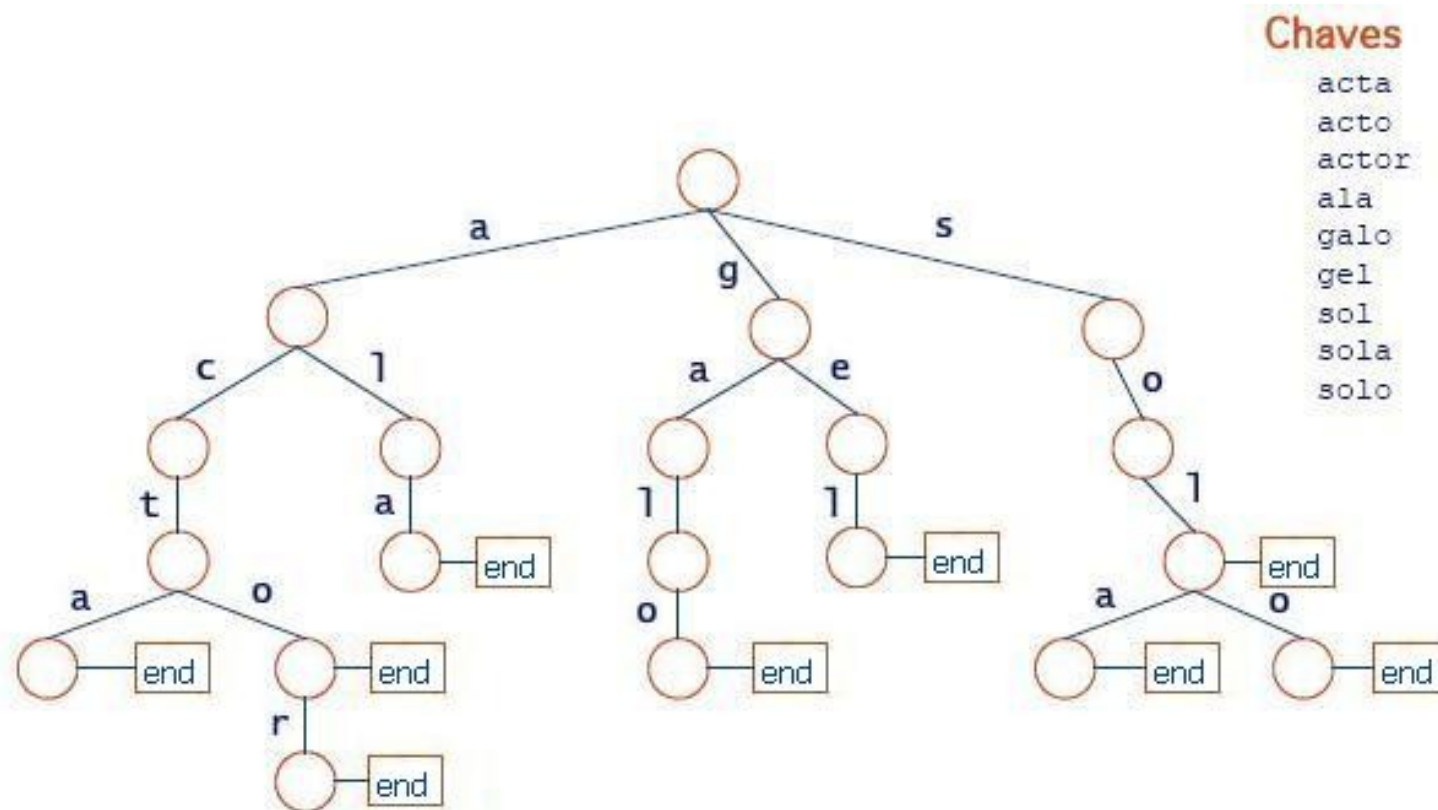
Estrutura de uma TRIE

- As chaves são associadas a folhas ou a nós internos de interesse, ditos terminais.



Estrutura de uma TRIE

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



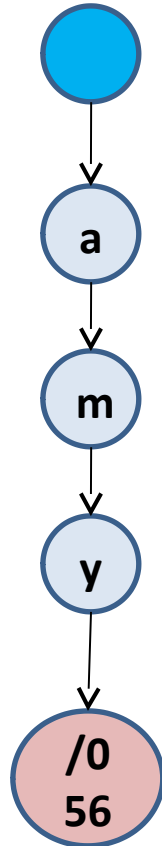
Montando uma árvore TRIE

- amy 56
- ann 15
- emma 30
- rob 27
- roger 52

Estes são pares que queremos inserir na árvore TRIE.

Montando uma árvore TRIE

- amy 56



<- Nível 0 (**RAIZ**)

<- Nível 1

<- Nível 2

<- Nível 3

<- Nível 4 (**TERMINAL**)

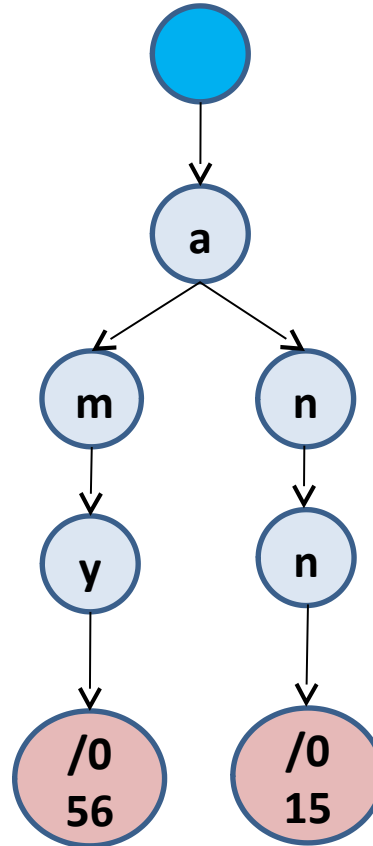
Montando uma árvore TRIE

- INSIRA

ann 15

Montando uma árvore TRIE

- ann 15



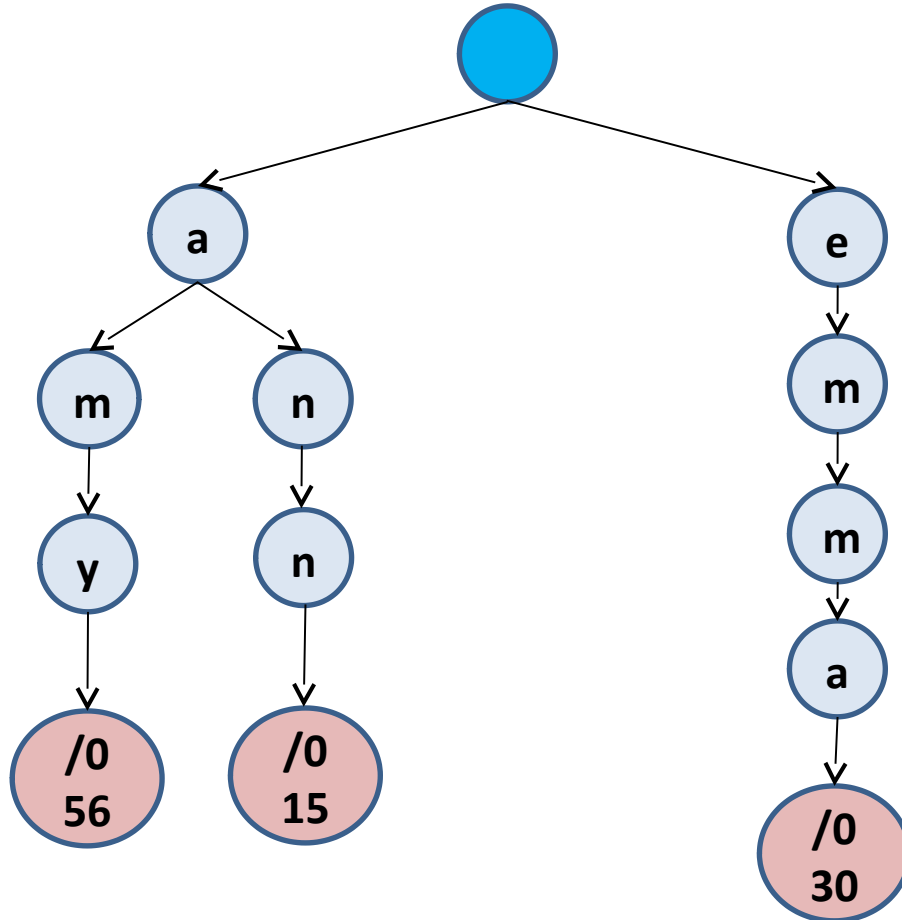
Montando uma árvore TRIE

- INSIRA

emma 30

Montando uma árvore TRIE

- emma 30



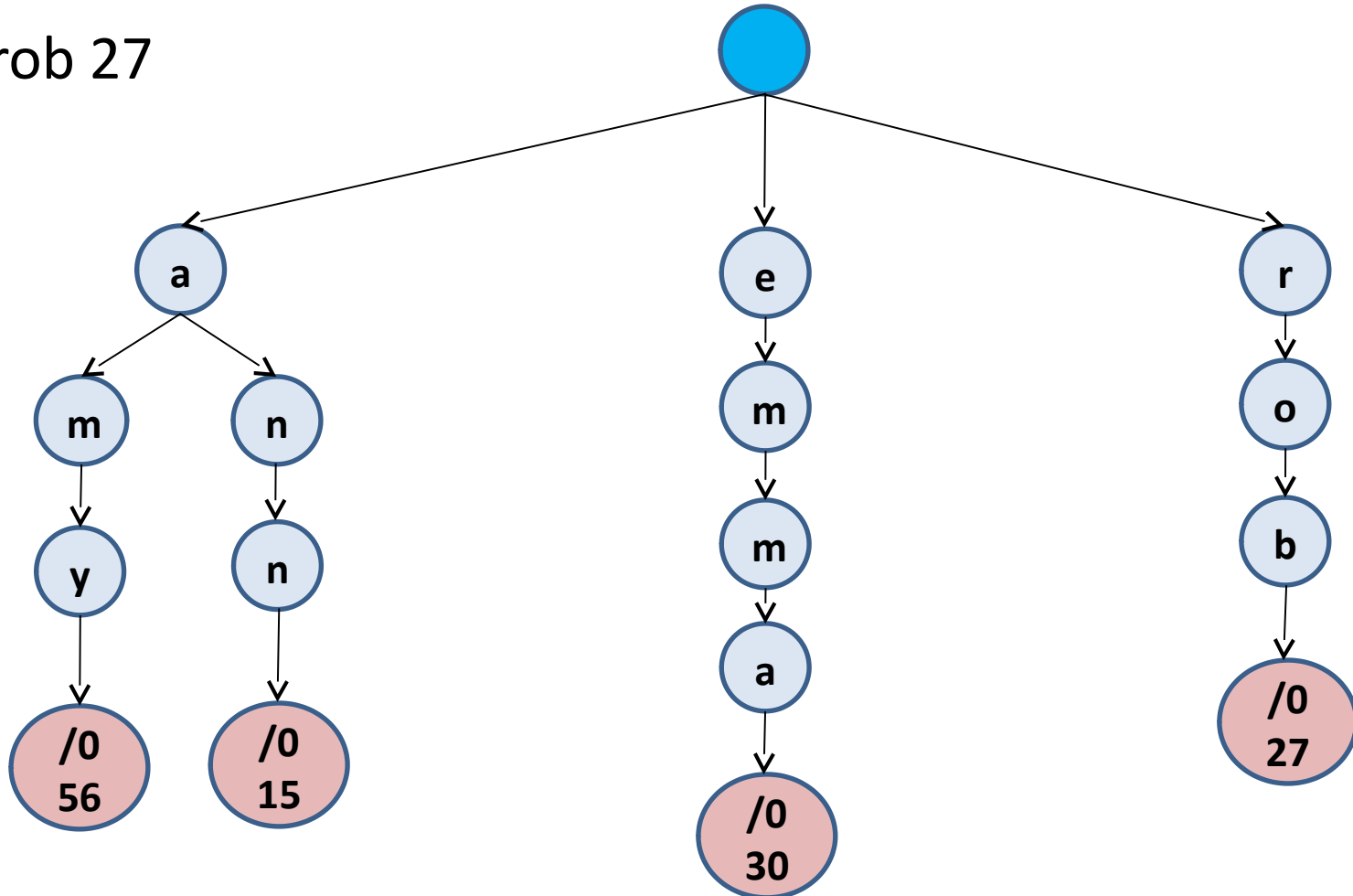
Montando uma árvore TRIE

- INSIRA

rob 27

Montando uma árvore TRIE

- rob 27



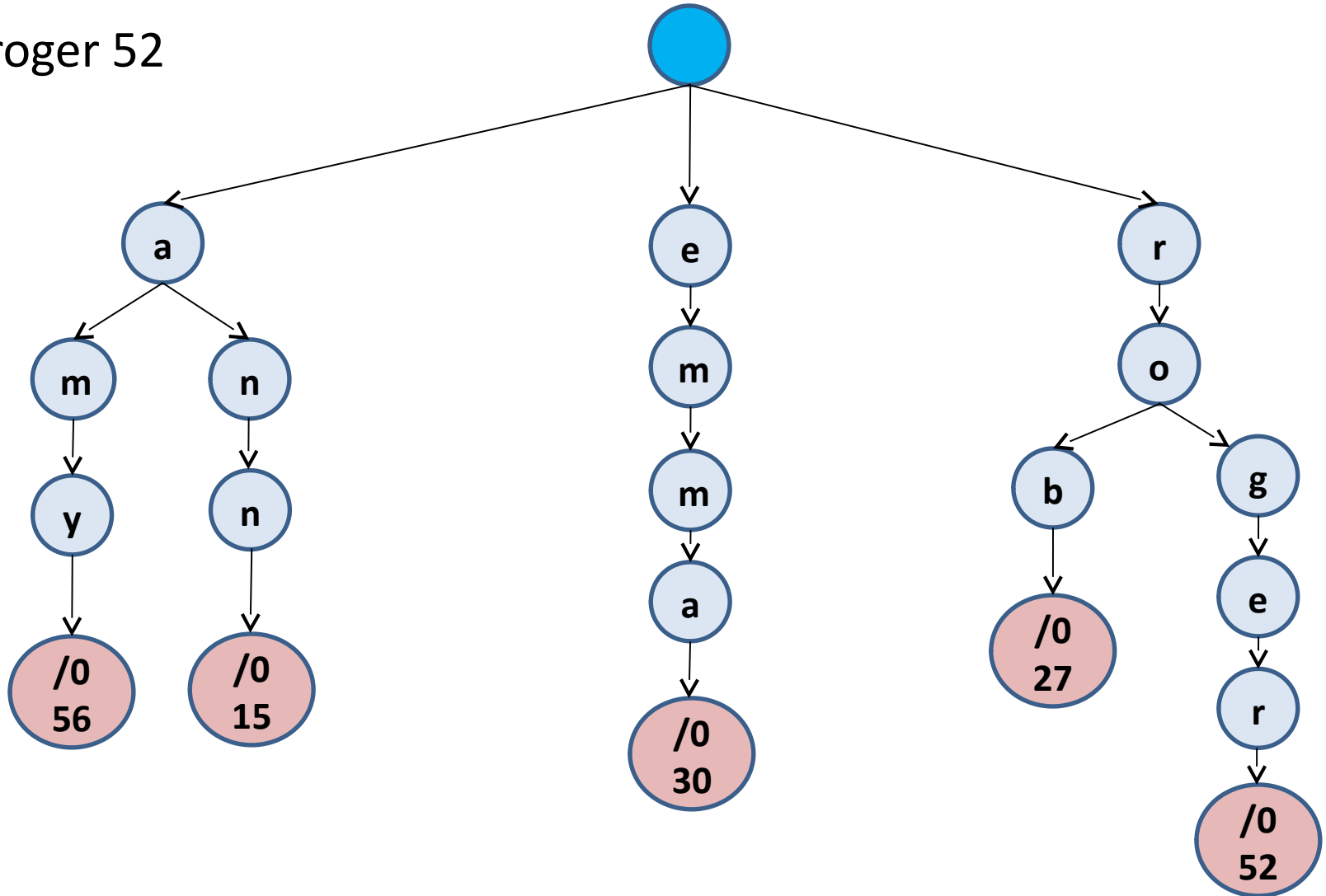
Montando uma árvore TRIE

- INSIRA

roger 52

Montando uma árvore TRIE

- roger 52



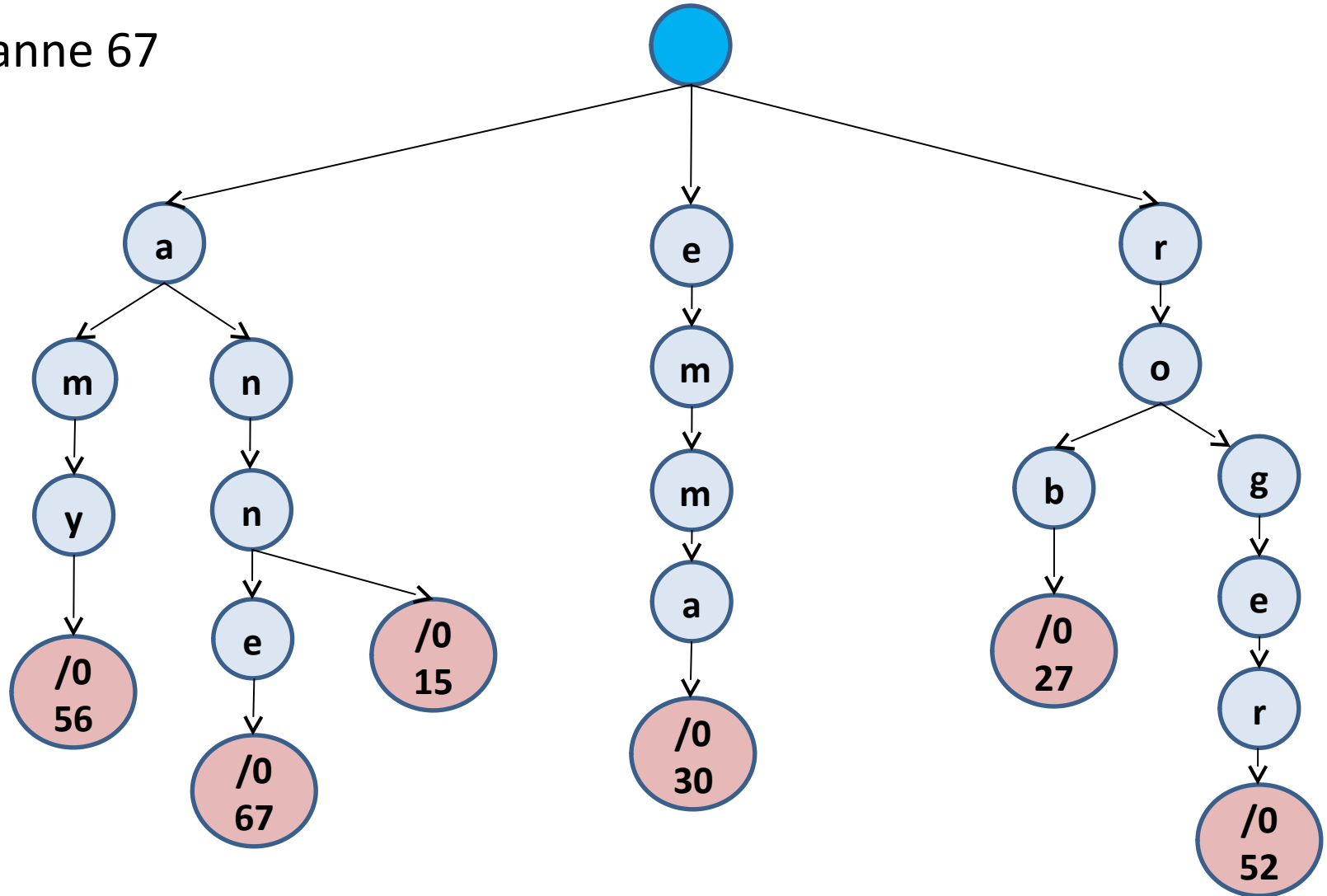
Montando uma árvore TRIE

- INSIRA

anne 67

Montando uma árvore TRIE

- anne 67



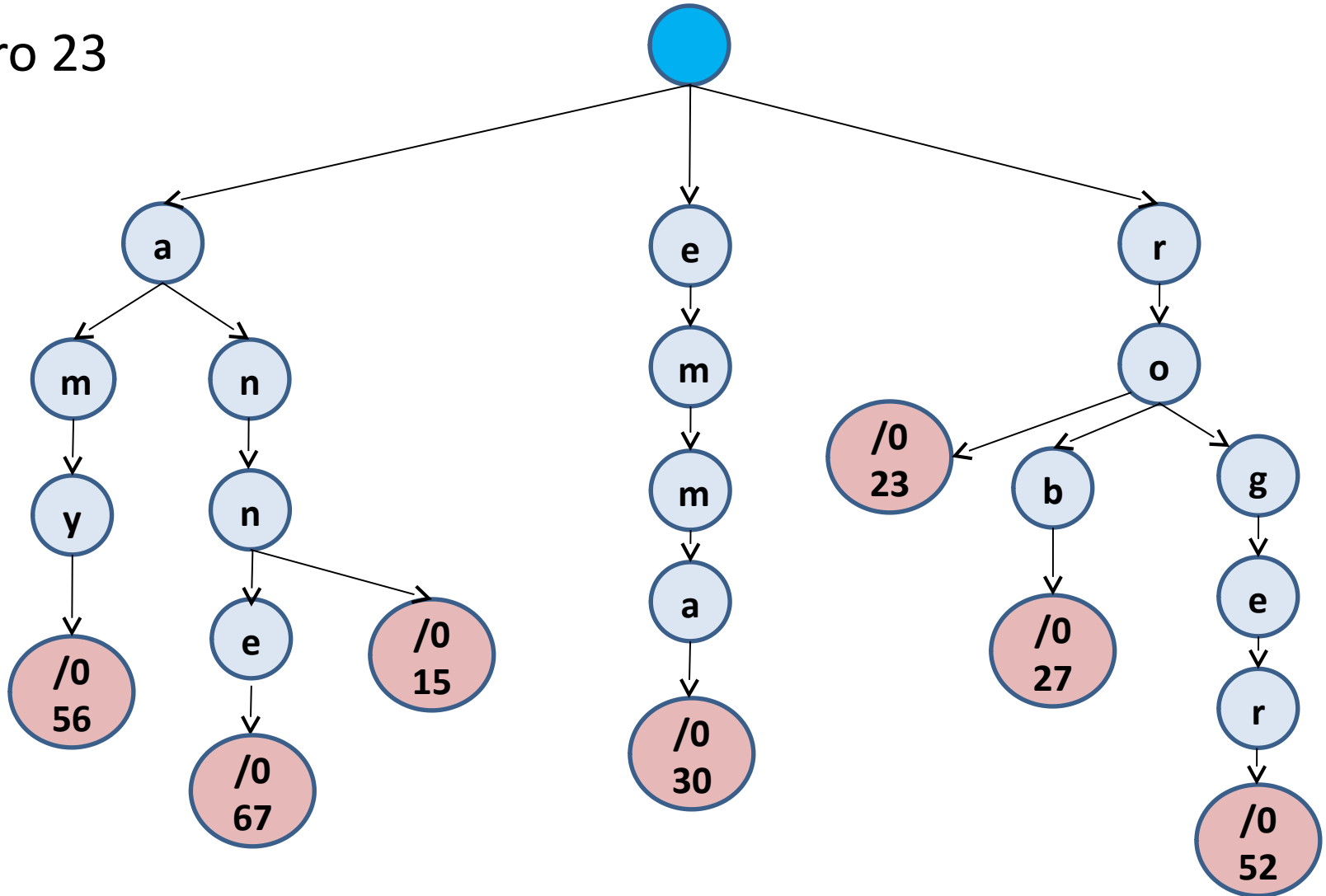
Montando uma árvore TRIE

- INSIRA

ro 23

Montando uma árvore TRIE

- ro 23



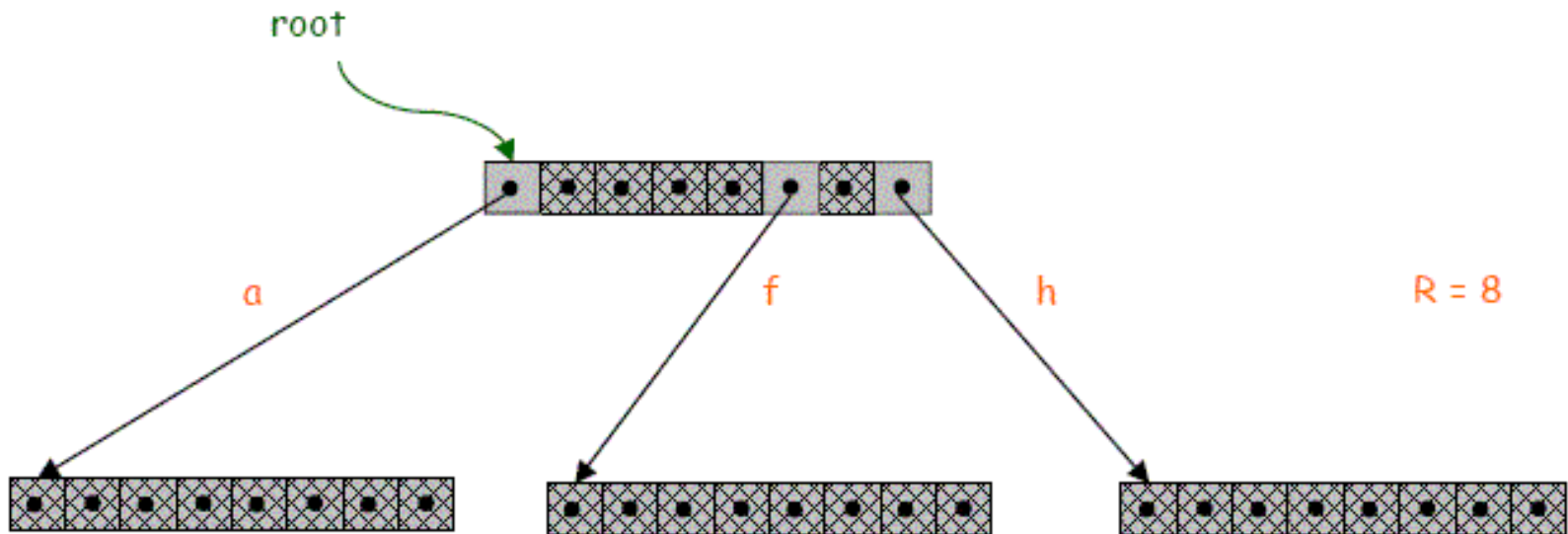
Tipos de TRIES

- Existem muitas variantes e tipos de TRIES:
 - R-WAY
 - TST
 - Patricia
 - ...

Tipos de TRIES

- R-WAY

Cada nó aloca espaço para todo o alfabeto.
Há desperdício de espaço.



Tipos de TRIES

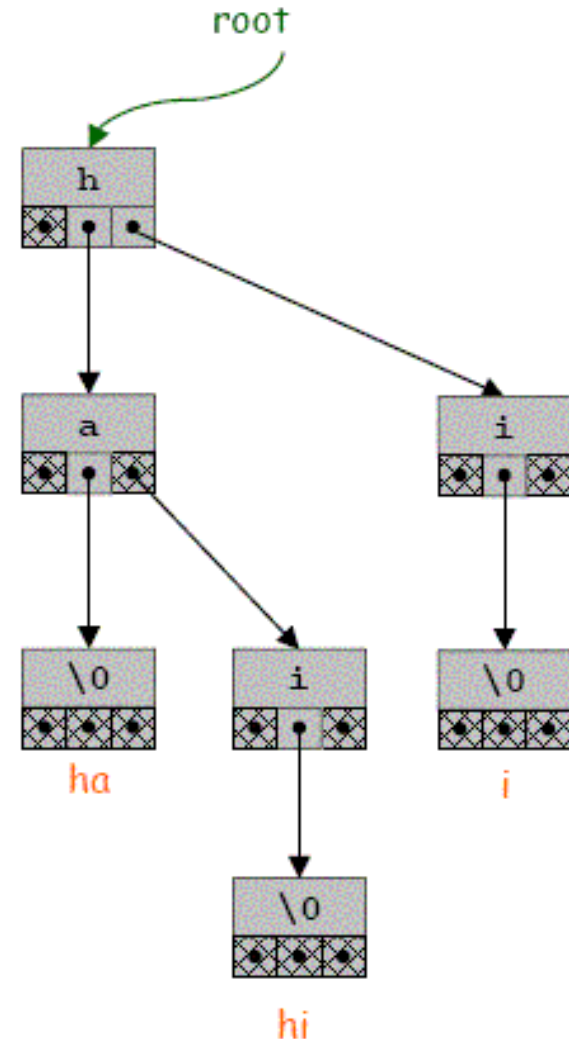
- TST- Ternary Search Tree

Cada nó aloca três ponteiros.

Centro: caractere seguinte.

Filhos da esquerda e direita:
caracteres alternativos.

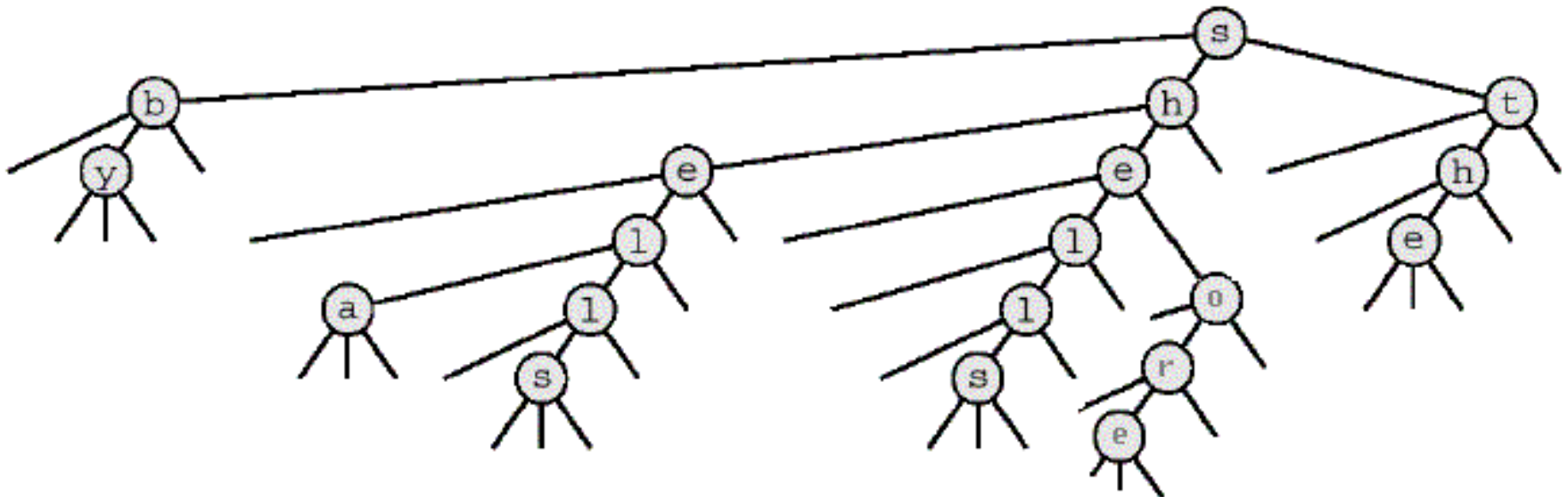
Tem desempenho melhor
que o R-WAY no que se refere
a espaço.



Tipos de TRIES

- TST- Ternary Search Tree

Chaves: by, sea, sells, shells, shore, the



Operações em TRIES

- SELEÇÃO: “É MEMBRO?”
 - INSERÇÃO
 - REMOÇÃO

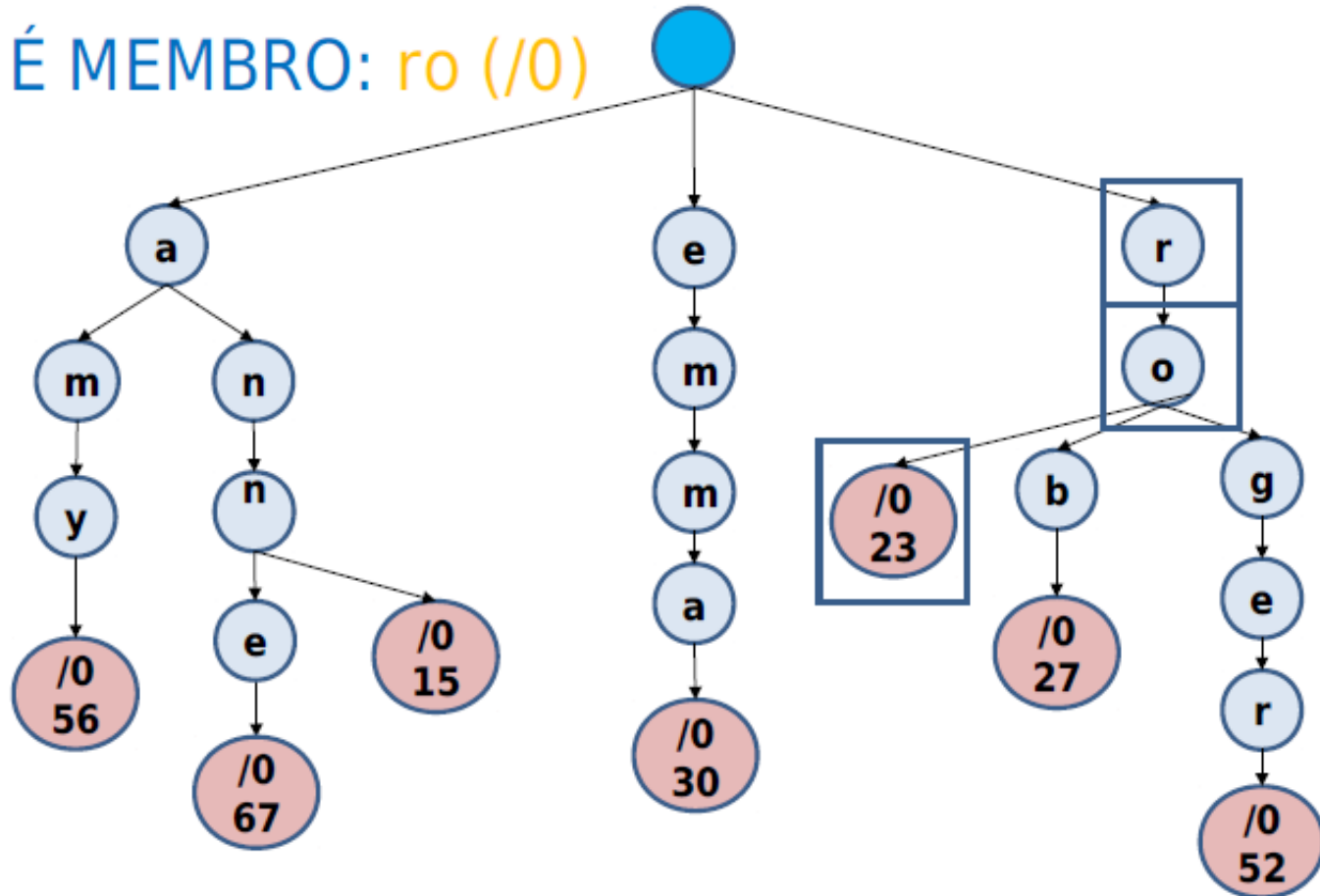
Operações em TRIES

- É MEMBRO?

1. Busca o nó que confere com o primeiro (ou atual) caractere da chave
2. Se nenhum, retorna FALSO
Senão
3. Se o caractere que confere é \0, retorna VERDADEIRO
Senão
4. Move para a subTRIE que confere com esse caractere
5. Avança para o próximo caractere na chave
6. Vá para Passo 1

Operações em TRIES

- É MEMBRO: ro (/0)



Operações em TRIES

- INSERÇÃO

Faz-se uma busca pela palavra a ser inserida.

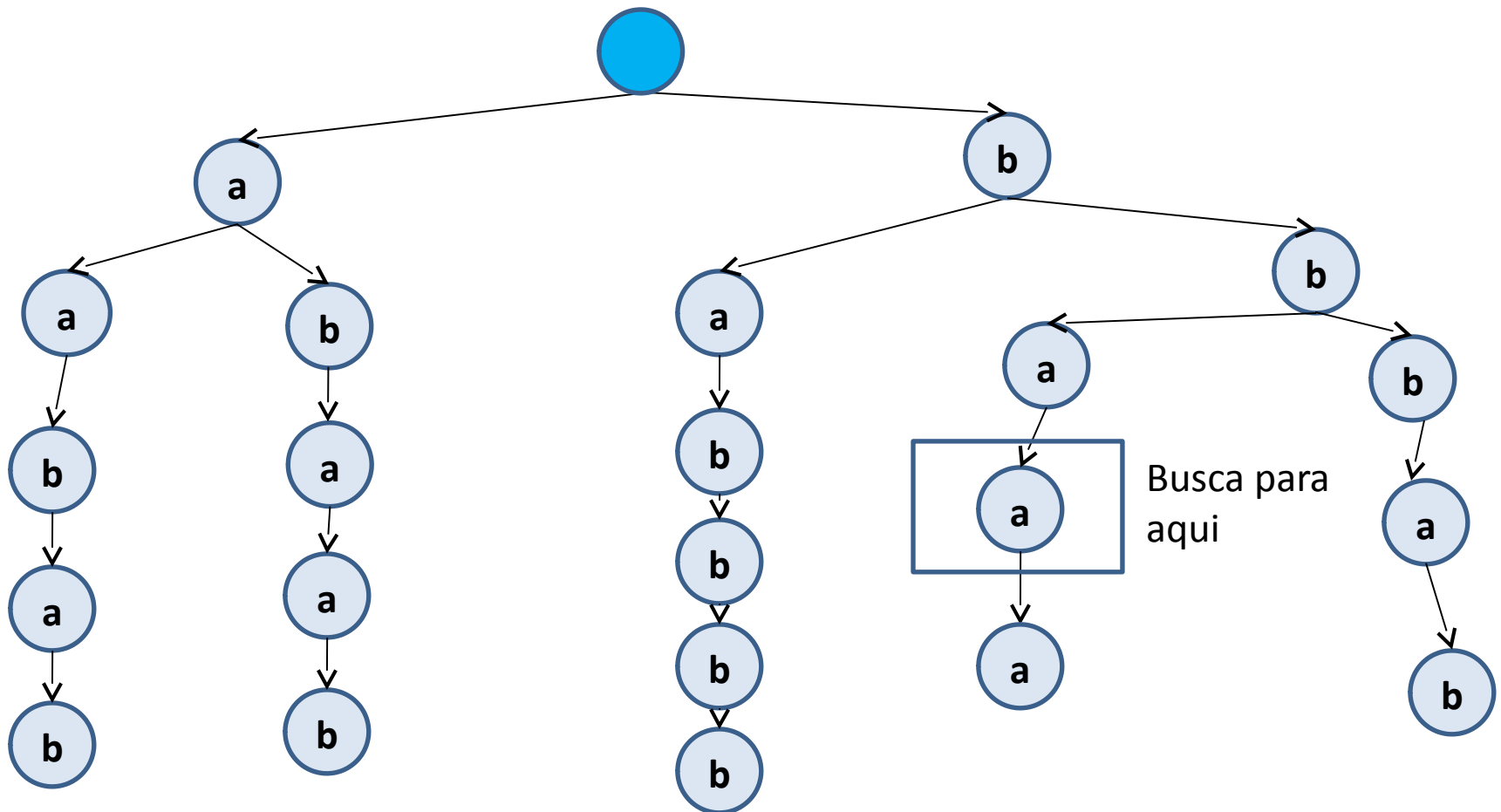
Se ela já existir na TRIE nada é feito, ou adiciona-se alguma informação no terminal.

Caso contrário, é recuperado o nó até onde acontece a maior prefixo da palavra a ser inserida.

O restante dos seus caracteres são adicionados na TRIE a partir daquele nó.

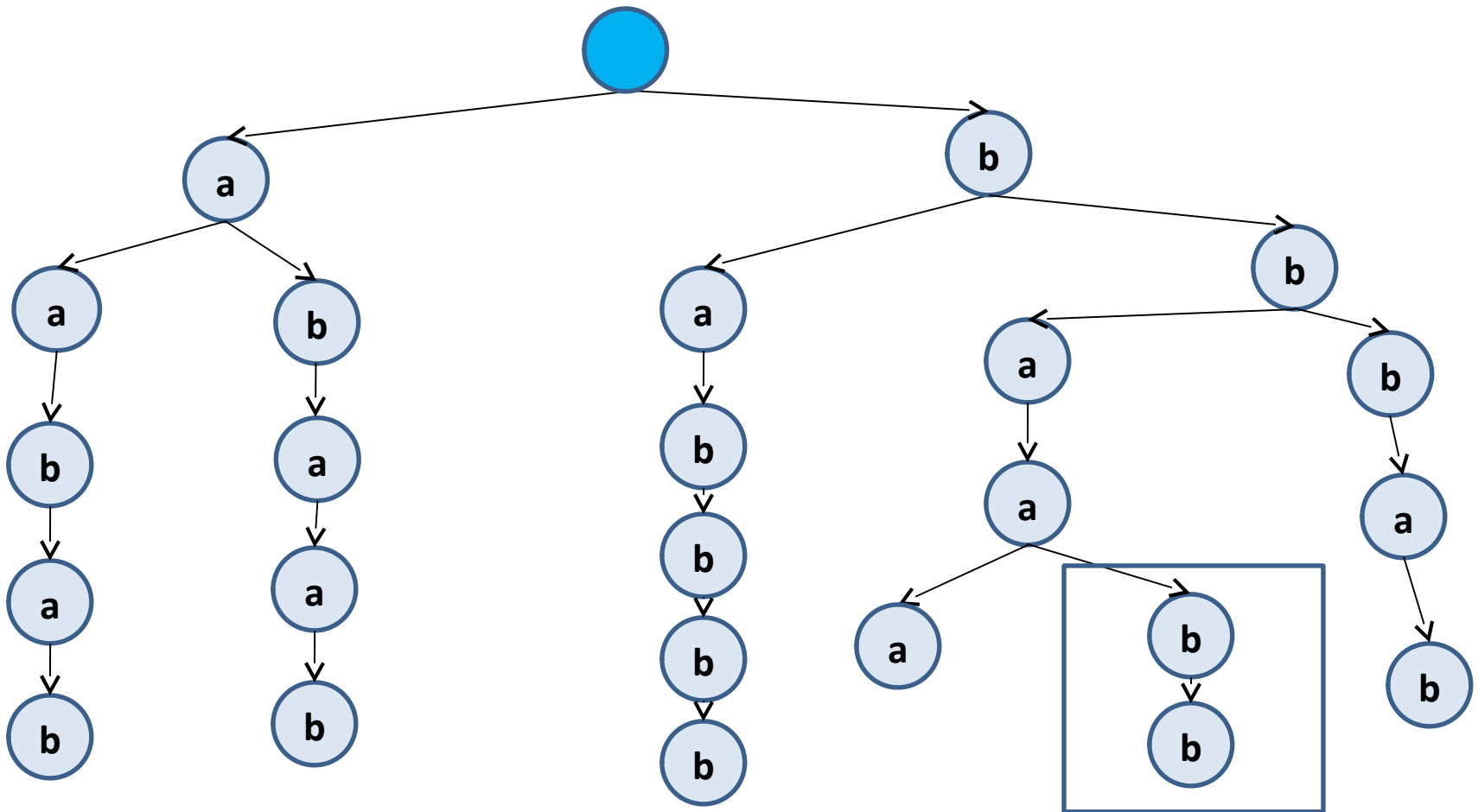
Operações em TRIES

Inserção : bbaabb



Operações em TRIES

Inserção : bbaabb



Operações em TRIES

- REMOÇÃO

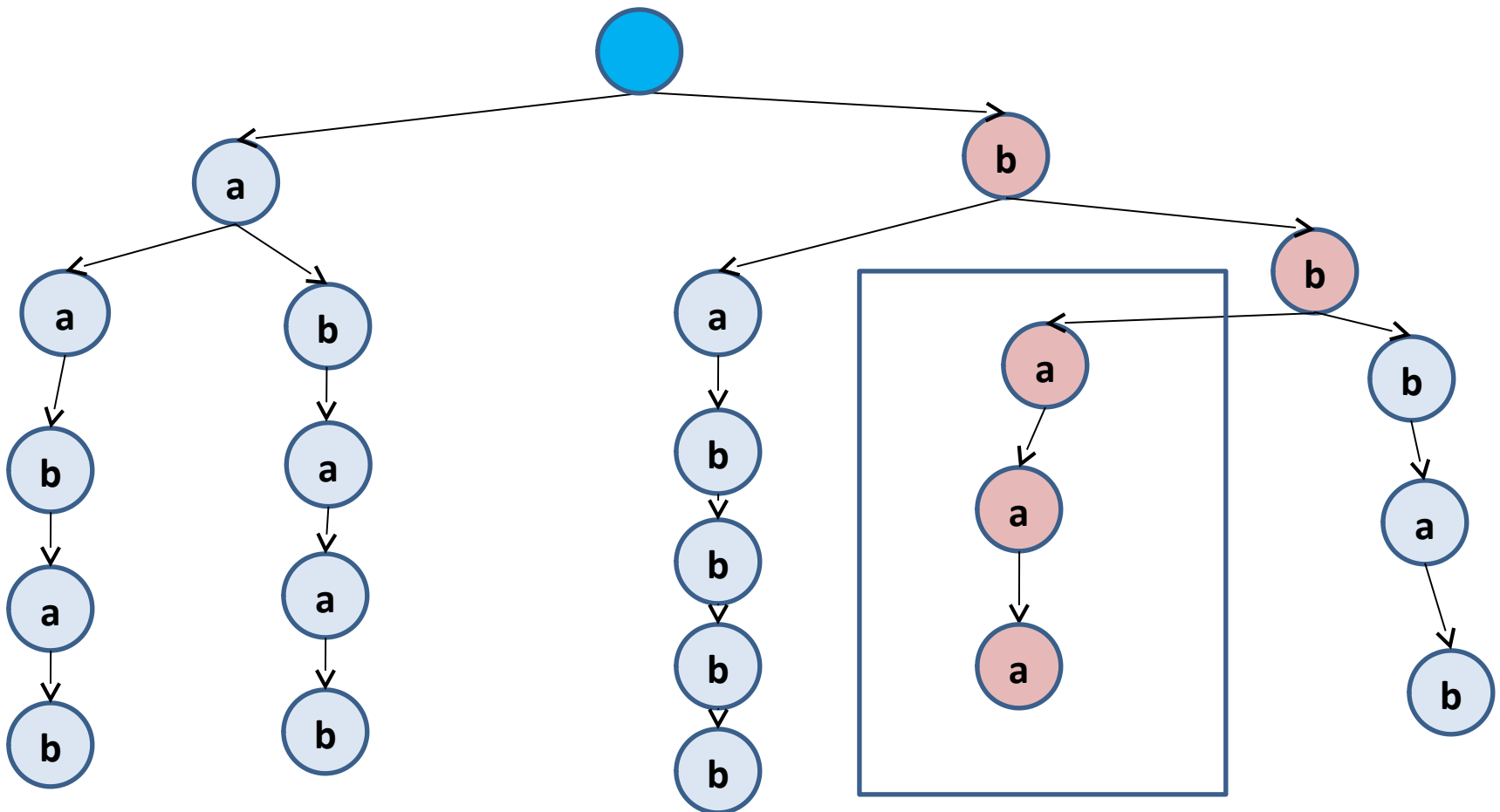
Busca-se o nó que representa o final da palavra a ser removida.

São removidos os nós que possuem apenas um filho pelo caminho ascendente.

A remoção é concluída quando se encontra um nó com mais de um filho – considerar os nós terminais.

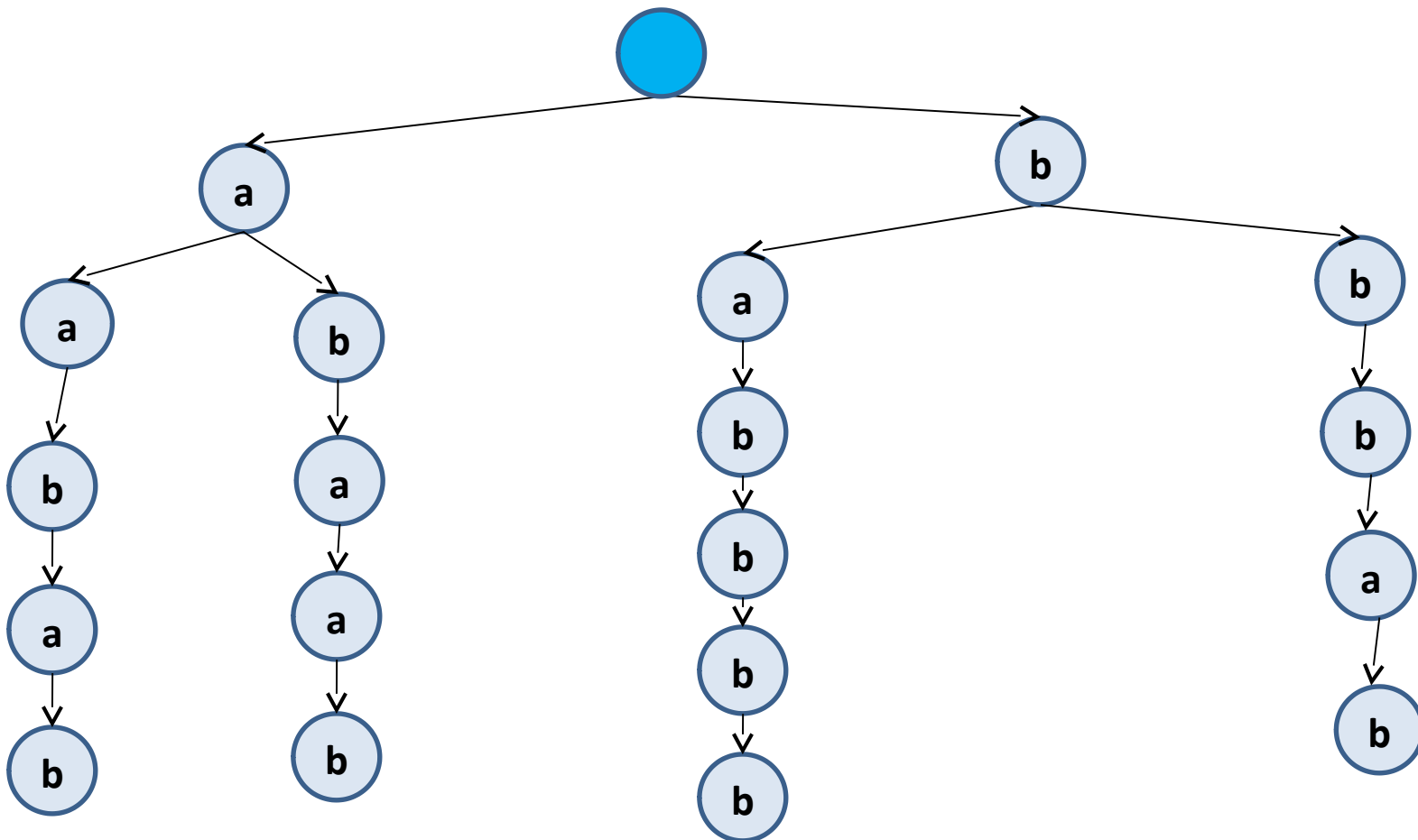
Operações em TRIES

Remoção : bbaaa



Operações em TRIES

Remoção : bbaaa



Análise

- A altura da árvore TRIE é igual ao comprimento da chave mais longa, com isso, o tempo de execução das operações **não** depende do número total de chaves.
- De fato, visita-se no máximo $K + 1$ nós da árvore TRIE e gasta-se tempo $O(A)$ em cada nó para determinar a posição do dígito corrente na ordenação do alfabeto.

Complexidade: $O(KA)$

K = tamanho da chave

A = tamanho do alfabeto

Análise

- Usando busca binária, a **seleção** pode ser realizada em tempo de resposta logarítmico em relação ao tamanho do alfabeto.

Complexidade: $O(K \log(A))$

- A mesma implementação pode ser usada na **inserção**:

Complexidade: $O(K_1 \log(A) + K_2 A)$

$K = K_1 + K_2$: tamanho da chave a ser inserida;

K_1 : tamanho do maior prefixo comum a chave a ser inserida e a alguma outra chave da árvore;

K_2 : número de nós a serem (efetivamente) incluídos na árvore.

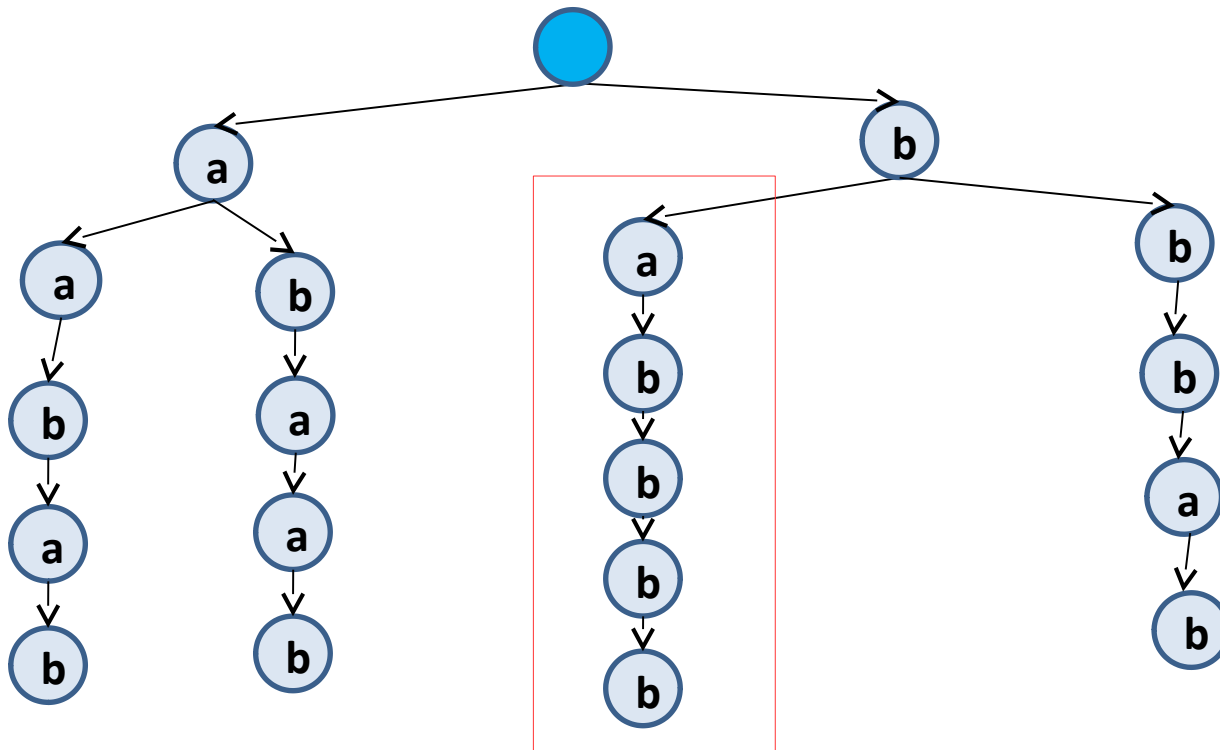
- A **remoção** depende do tamanho da chave, no máximo.

Considerações

- A pesquisa digital é uma base importante para tratamento de grandes arquivos e quando as chaves têm tamanho arbitrário e variável. Por exemplo, na área de linguística.
- Existem potencialmente muitos nós na árvore TRIE que possuem apenas um filho, e a existência desses nós é um desperdício. Em alguns casos, o total de nós poderá ser maior que o número de palavras no texto.
- Assim, a técnica da pesquisa digital é tão mais eficiente quanto maior for a quantidade de chaves com prefixos comuns. Uma TRIE com muitos “zigue-zagues” é quase sempre ineficiente.

Considerações

- Um zigue-zague de uma árvore é uma subárvore cujos nós possuem um único filho.



Aplicações das TRIES

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

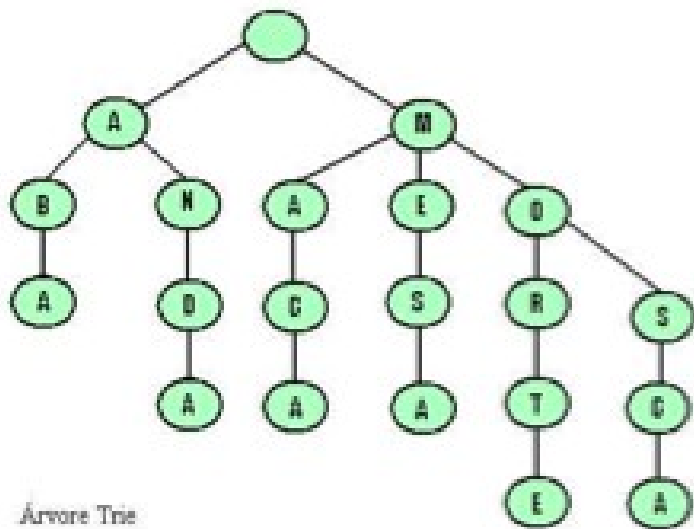
Corretor ortográfico

- Com o dicionário armazenado numa TRIE, pode-se percorrer essa estrutura letra por letra para encontrar, ou não a palavra testada.
- Caso seja detectado um “erro”, o algoritmo verifica a possibilidade de ocorrência de cada um dos tipos de erros para poder indicar as opções de correção.

Corretor ortográfico

1. **Substituição** - avança um caractere na chave e avança um nível na árvore.
2. **Deleção** - avança um nível na árvore.
3. **Inserção** - avança um caractere na chave.
4. **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



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

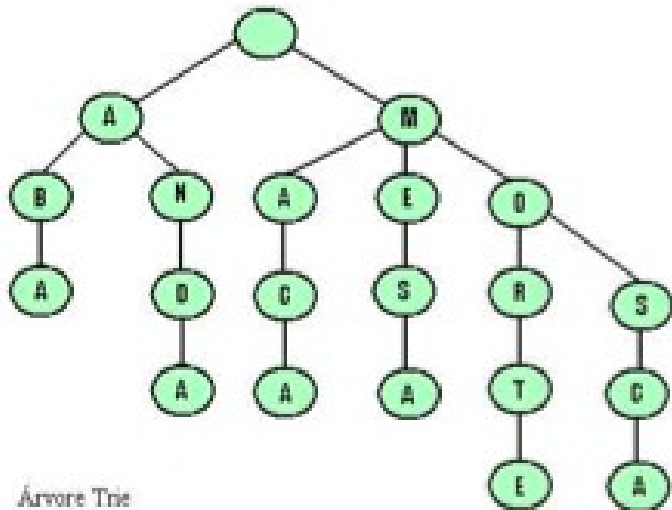
* D = B - erro

* D = N - erro

* próximo passo avança na chave e na árvore (substituição)

* A = A - ok

Corretor ortográfico



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 das TRIES

Autopreenchimento

- Armazena as palavras mais usadas em uma árvore TRIE. A medida que vai digitando exibe as opções possíveis de palavras já usadas.
- Utilizado em várias aplicações: browsers, programas de e-mail, IDEs de linguagens de programação, etc.

Autopreenchimento

M

M O S C A

A

E

O

C

S

R

S

A

A

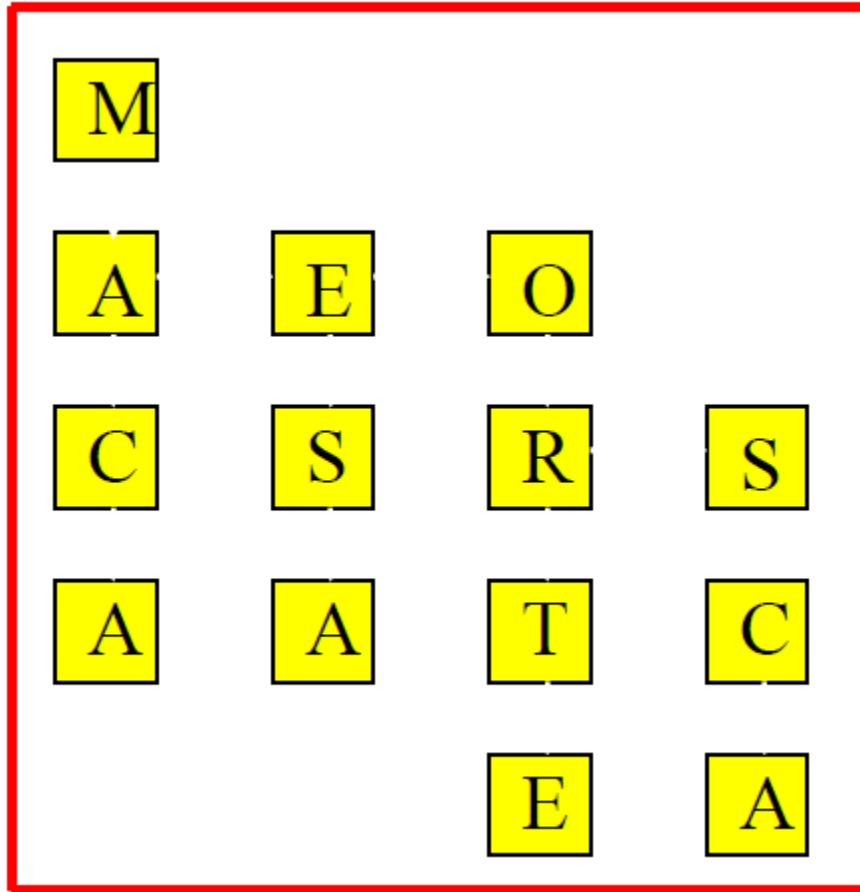
T

C

E

A

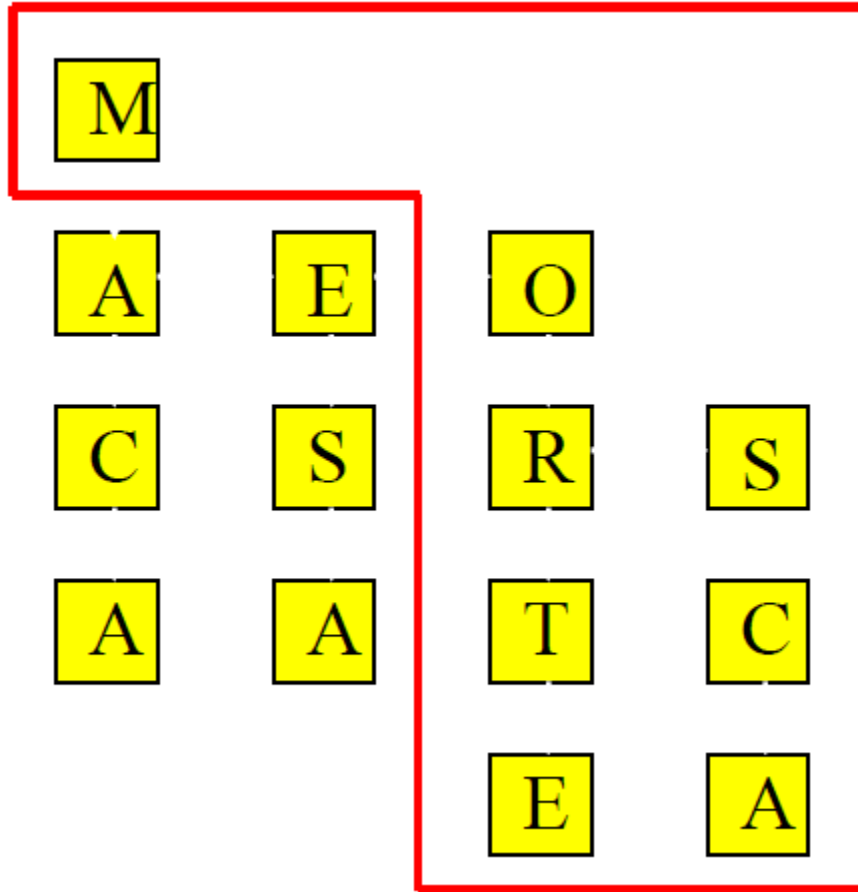
Autopreenchimento



M O S C A

maca
mesa
morte
mosca

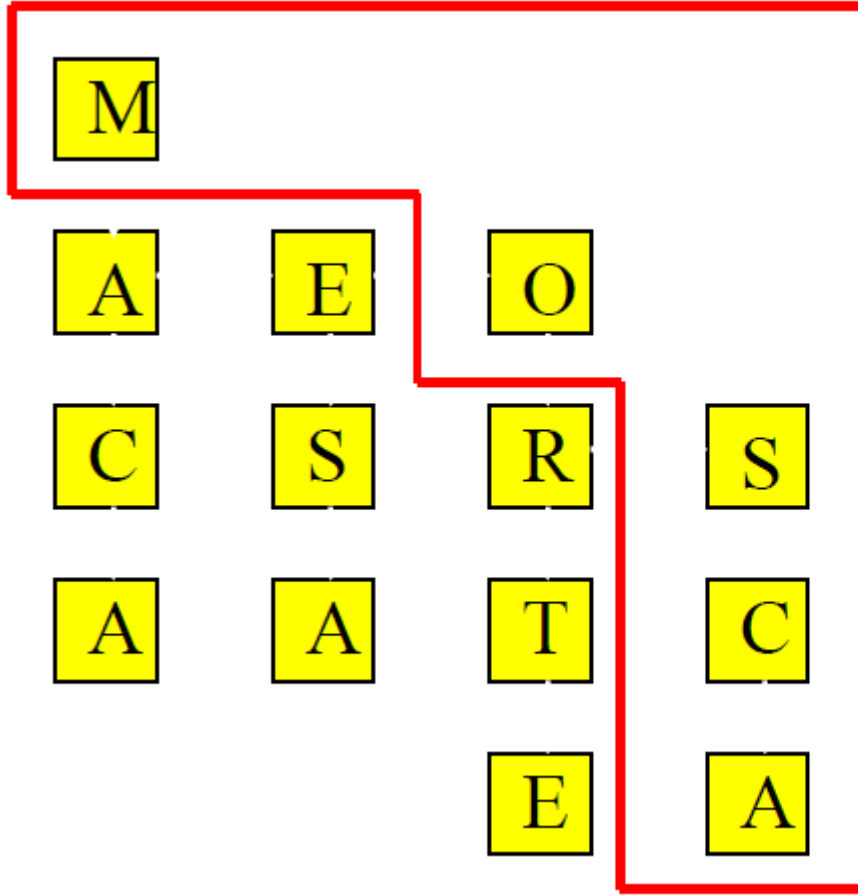
Autopreenchimento



M O S C A

morte
mosca

Autopreenchimento



M O S C A

Árvore PATRICIA

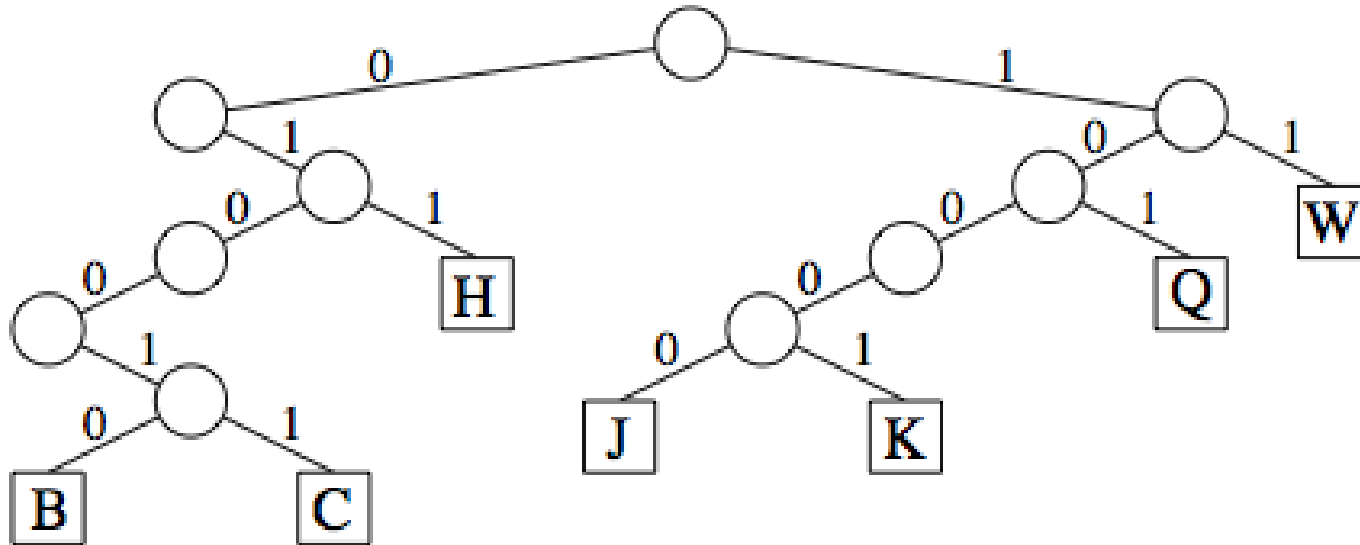
- Definida em 1968 por Donald Morrison: Practical Algorithm To Retrieve Information Coded In Alphanumeric.
- É um caso particular de árvore TRIE binária.
- Essa estrutura foi pensada para resolver uma das grandes desvantagens da árvore TRIE:

“Formação de caminhos de uma só direção para chaves com um grande número de *bits* em comum, os zigue-zagues.”
- **Solução:** cada nó interno da árvore contém o índice do *bit* a ser testado para decidir qual ramo tomar.

Árvore PATRICIA

- A PATRICIA é estritamente binária e construída a partir de uma **árvore binária de prefixo**, ou seja, nenhuma chave é prefixo da outra.
- Uma propriedade da árvore binária de prefixo é que há uma correspondência entre o conjunto das chaves e o das folhas da árvore.
- Isto é, cada chave é unicamente representada por uma folha e a codificação binária dessa chave corresponde ao caminho da raiz até essa folha.

Exemplo



Se duas chaves diferirem somente no último *bit*, elas formarão um caminho cujo comprimento é igual ao tamanho delas, gerando nós e comparações desnecessários. Por exemplo, o caminho para localizar as chaves B e C.

Exemplo

- Vamos obter a árvore PATRICIA para as chaves abaixo.
- A ordem de inserção será: B, J, H, Q, C, K.

B : 010010

J : 100001

H : 011000

Q : 101000

C : 010011

K : 100010

Exemplo

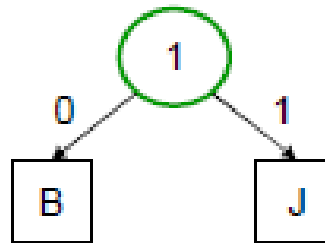
- A chave B é a primeira a ser inserida na árvore.



B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

Exemplo

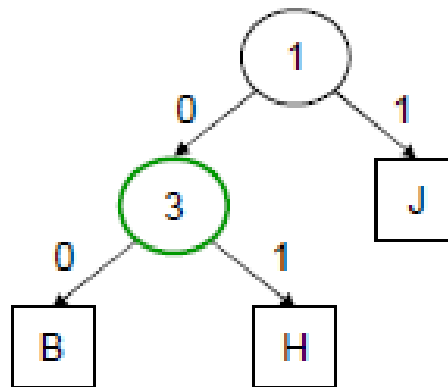
- A chave J é inserida na árvore.



B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

Exemplo

- A chave H é inserida na árvore.

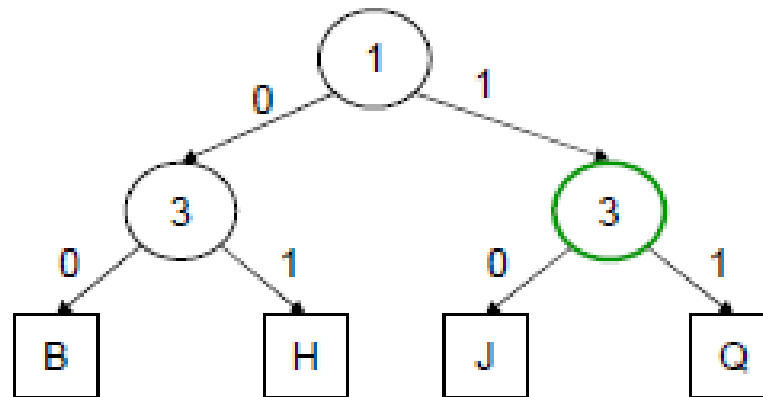


B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

B e H seguem o padrão
0xxxxx.
O 3o. bit diferencia B de H

Exemplo

- A chave Q é inserida na árvore.

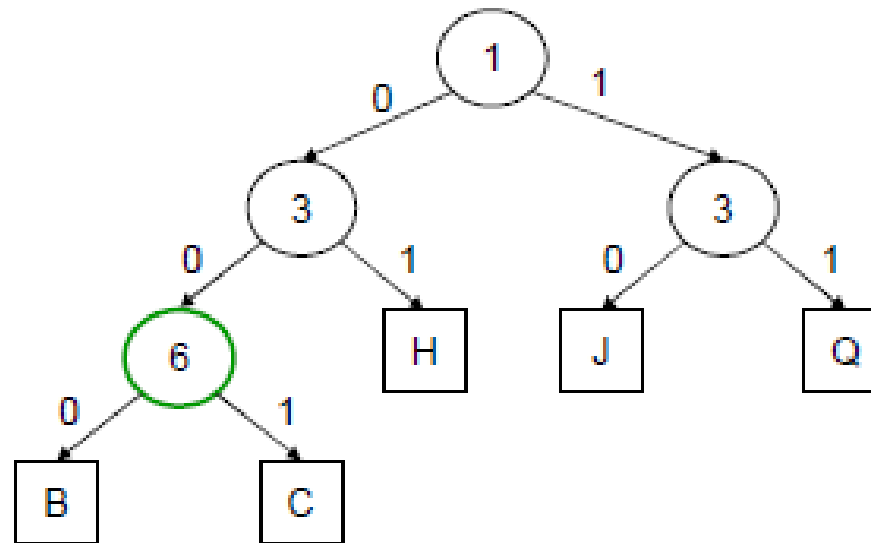


B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

J e Q seguem o padrão
1xxxxx.
O 3o. bit diferencia J de Q

Exemplo

- A chave C é inserida na árvore.



B = 010010

C = 010011

H = 011000

J = 100001

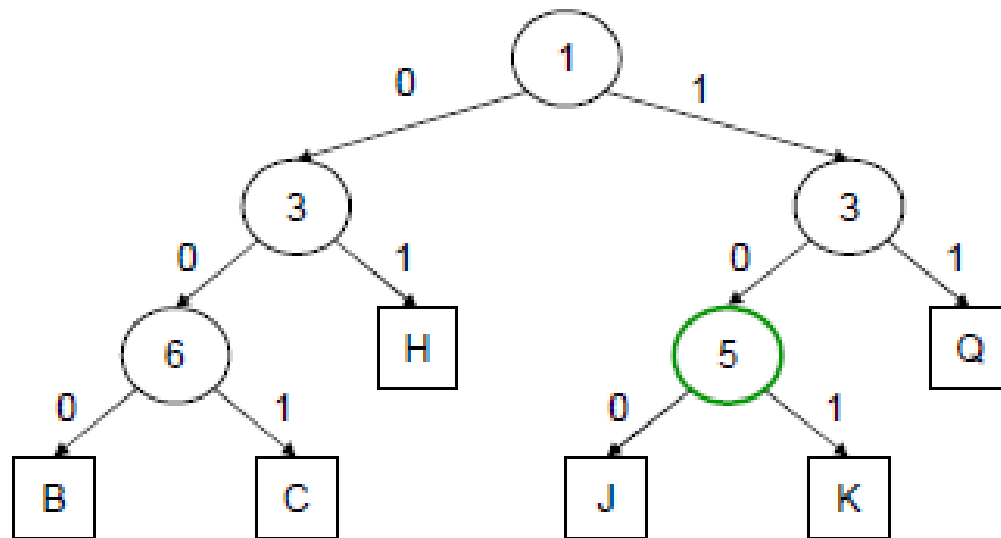
Q = 101000

K = 100010

B e C seguem o padrão
0x0xxx.
O 6o. bit diferencia B de C

Exemplo

- A chave K é inserida na árvore.



B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

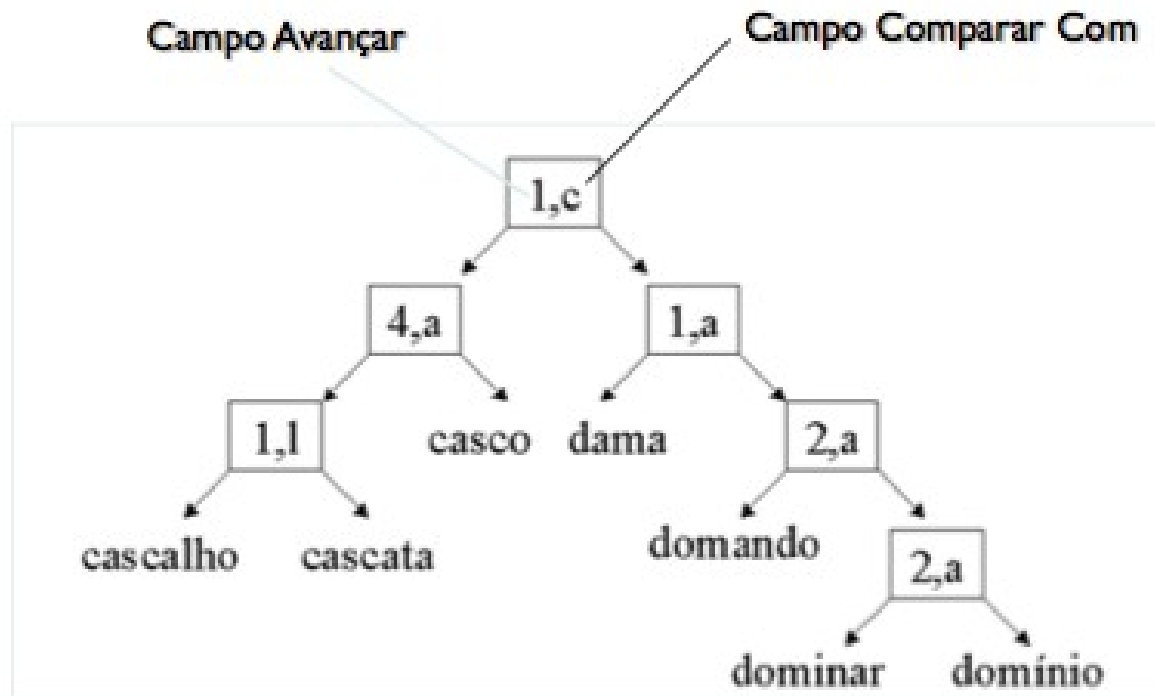
J e K seguem o padrão
1x0xxx.
O 5o. bit diferencia J de K

Considerações

- A eficiência (ou complexidade no tempo) das operações em uma árvore PATRICIA limita-se ao tamanho da maior chave.
- Logo, o ganho da árvore PATRICIA em relação à árvore digital binária fica restrito à memória.
- **Vantagem:** Compactação. Armazena um número de posições para qual é movido para a frente antes de fazer a próxima comparação, o que elimina comparações desnecessárias e melhora o desempenho na prática.
- **Desvantagem:** Produz apenas duas subárvores. Caso mais de duas chaves sejam distintas na mesma posição, será preciso adicionar nós extras para separá-las.

Exercícios

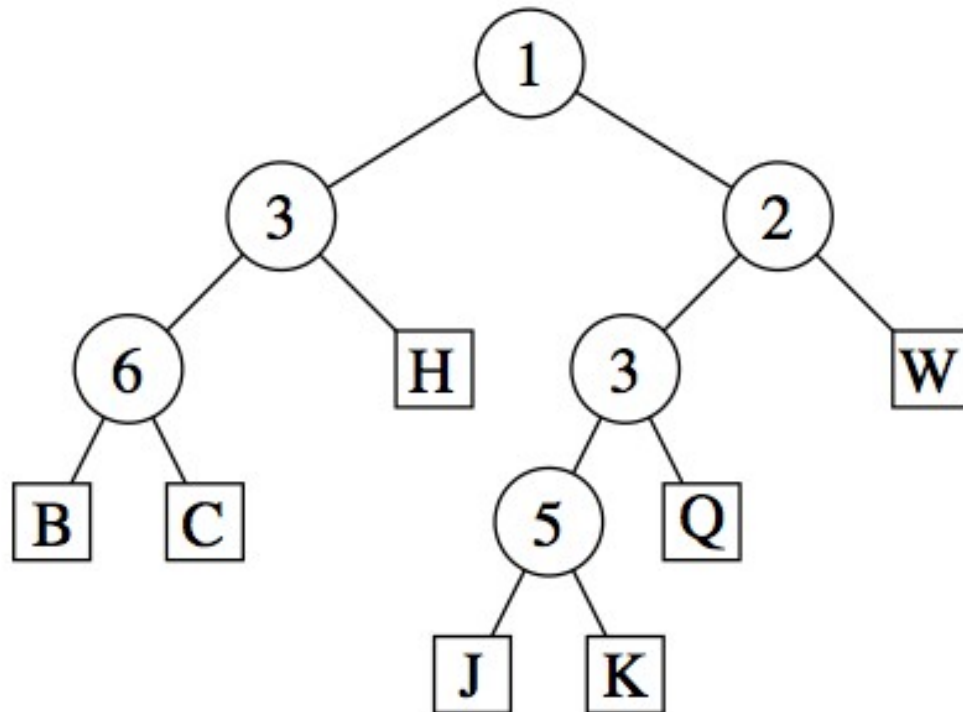
1. Insira a chave de 6 *bits* $W = 110110$ na árvore PATRICIA resultante do exemplo anterior.
2. Explique a representação da árvore PATRICIA abaixo.



Exercício 1

- A inserção da chave $W = 110110$ ilustra um outro aspecto.
- Os *bits* das chaves K e W são comparados a partir do primeiro para determinar em qual índice eles diferem, sendo, neste caso, no índice 2.
- Portanto, o ponto de inserção agora será no caminho de pesquisa entre os nós internos de índice 1 e 3.
- Cria-se um novo nó interno de índice 2, cujo descendente direito é um nó externo contendo W e o descendente esquerdo é a subárvore de raiz de índice 3.

Exercício 1

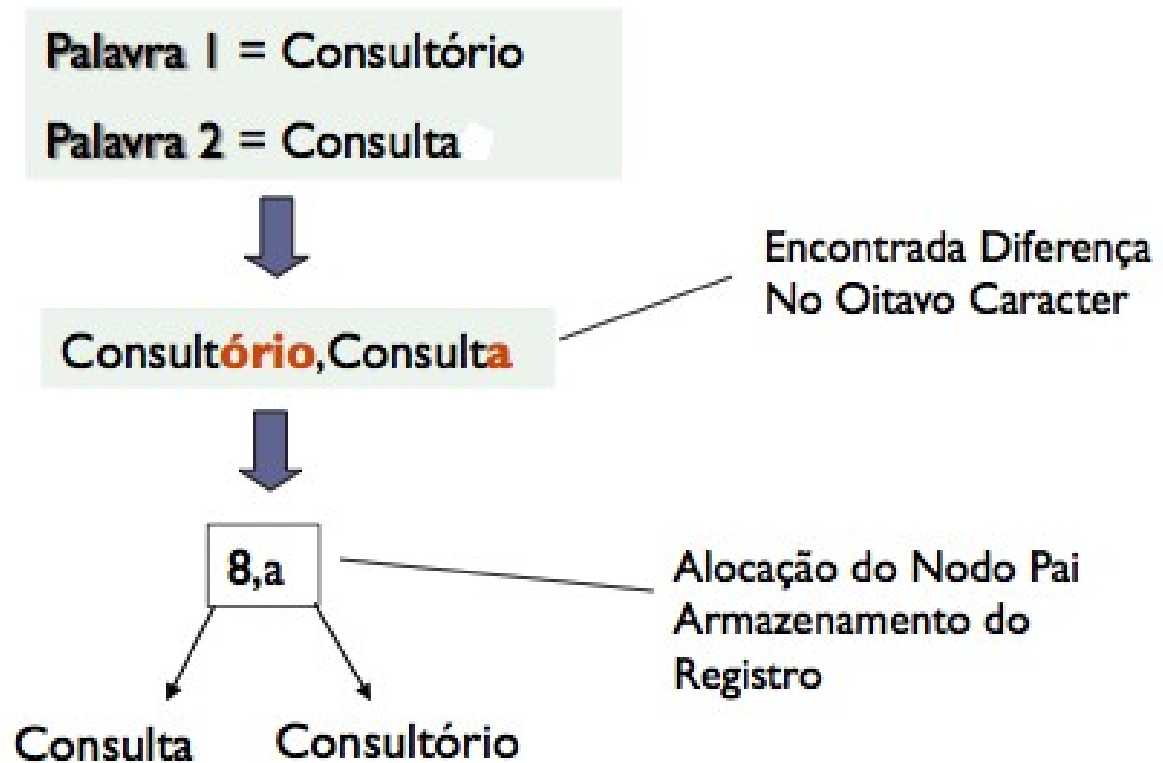


Exercício 2

- **Campo avançar:**
 - Registro acumulativo presente em todos os nós exceto nos nós folha.
 - Identifica a posição (ou caractere) da chave informada que deve ser analisada.
- **Campo comparar com:**
 - Apresenta o caractere que deve ser comparado ao caractere da chave informada.
 - Como nas árvores binárias de busca, se a chave é menor ou igual ao nó, a busca segue pela subárvore esquerda, senão pela subárvore direita.

Exercício 2

Exemplo de Inserção ::



Exercício 2

Exemplo de Inserção 2 ::

