MATA54 - Estruturas de Dados e Algoritmos II

Tries e Árvores PATRICIA

Flávio Assis
Versão gerada a partir de slides do Prof. George Lima

IC - Instituto de Computação

Salvador, outubro de 2021

Motivação: busca de sequências de caracteres

Indexação por chaves alfanuméricas de tamanho variável

- Estrutura de busca: caractere-a-caractere (ou *bit-a-bit*)
- Busca exata ou aproximada
- Aplicações: dicionários, correção ortográfica, busca em páginas *web*, etc.
- Uma vez encontrada uma palavra, dados sobre esta palavra podem ser acessados: ocorrências em um texto, em páginas *web*, classe gramatical, ...

Motivação: busca de sequências de caracteres

Indexação por chaves alfanuméricas de tamanho variável

- Estrutura de busca: caractere-a-caractere (ou *bit-a-bit*)
- Busca exata ou aproximada
- Aplicações: dicionários, correção ortográfica, busca em páginas *web*, etc.
- Uma vez encontrada uma palavra, dados sobre esta palavra podem ser acessados: ocorrências em um texto, em páginas *web*, classe gramatical, ...

Estrutura de dados a serem estudadas

- TRIE: o termo vem de reTRIEval
- Árvore Patricia: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*

TRIES

TRIE

Representação básica:

- Uma floresta de árvores k-ária em que se representam nos nós da árvore os caracteres que ocorrem em um conjunto de palavras

TRIE

Representação básica:

- Uma floresta de árvores k-ária em que se representam nos nós da árvore os caracteres que ocorrem em um conjunto de palavras
- Prefixos comuns são aproveitados

TRIE

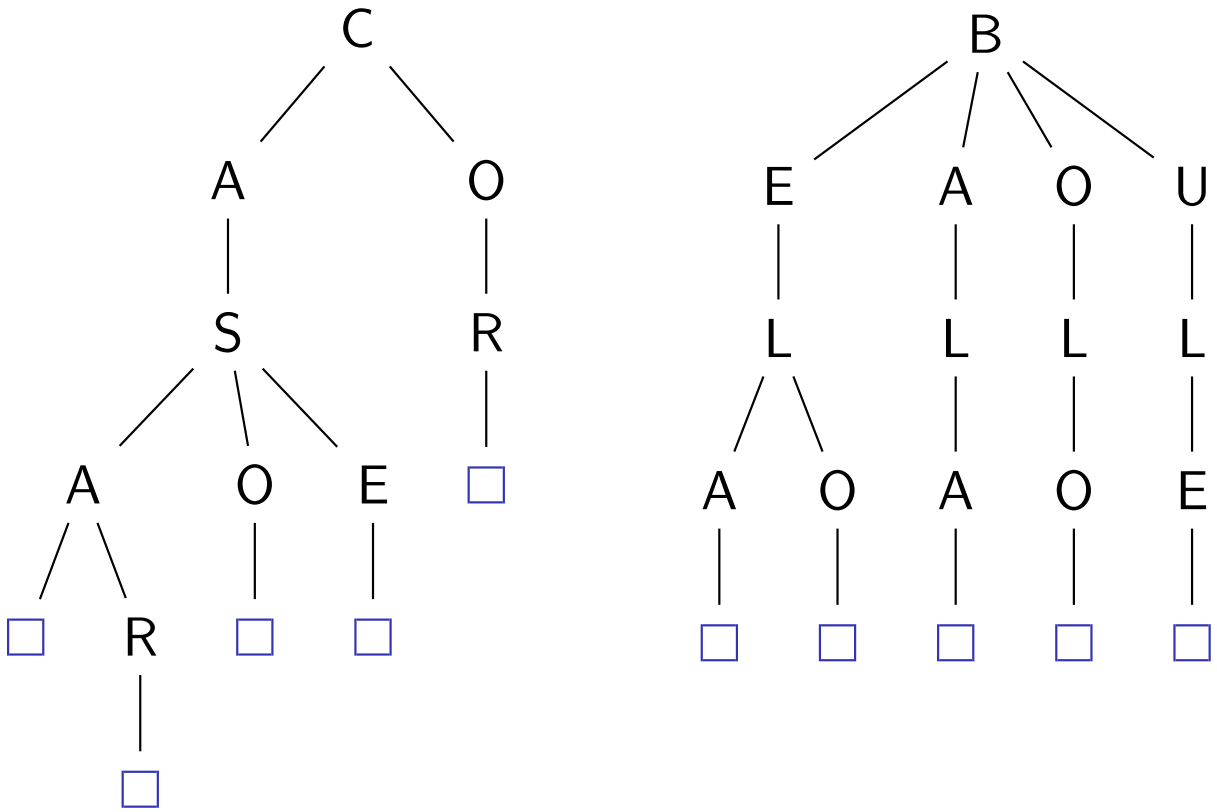
Representação básica:

- Uma floresta de árvores k-ária em que se representam nos nós da árvore os caracteres que ocorrem em um conjunto de palavras
- Prefixos comuns são aproveitados
- Uma árvore para cada caractere inicial

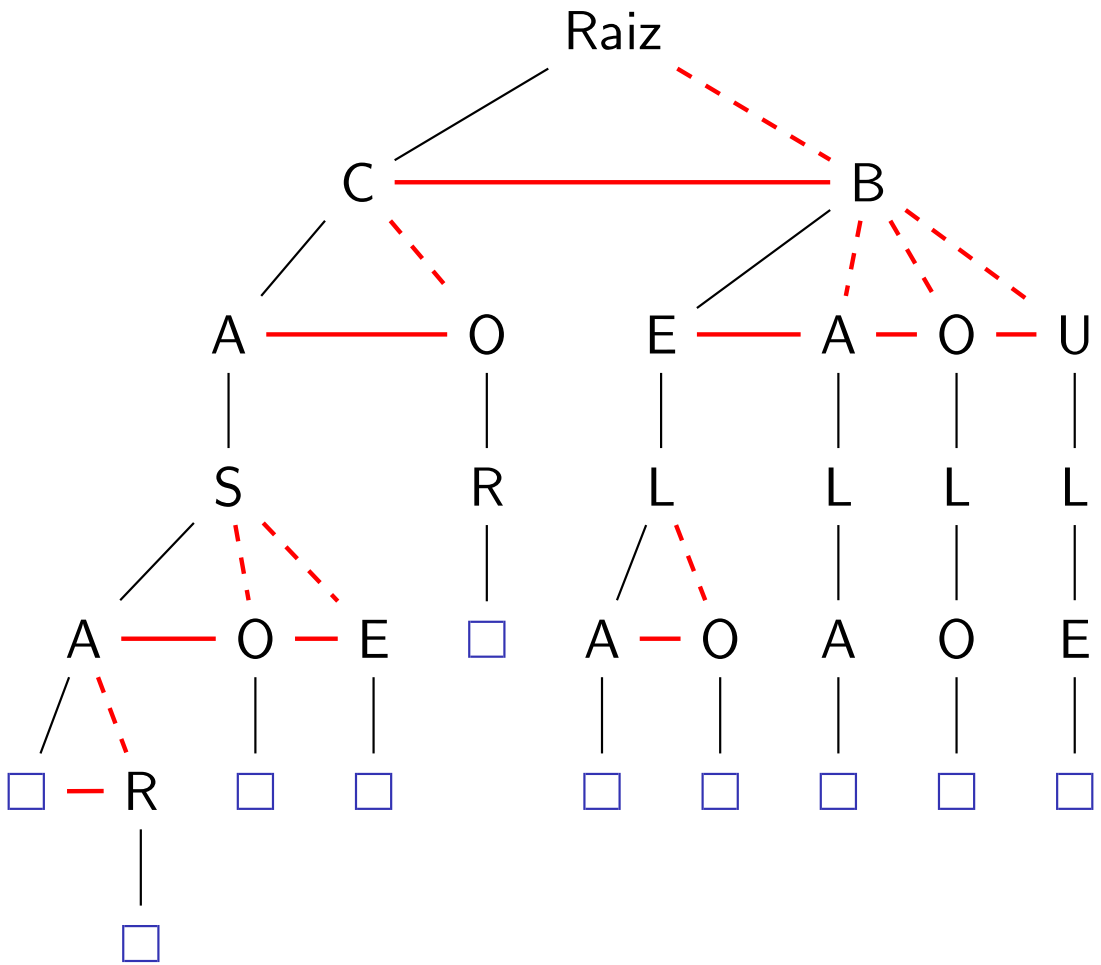
TRIE

Exemplo:

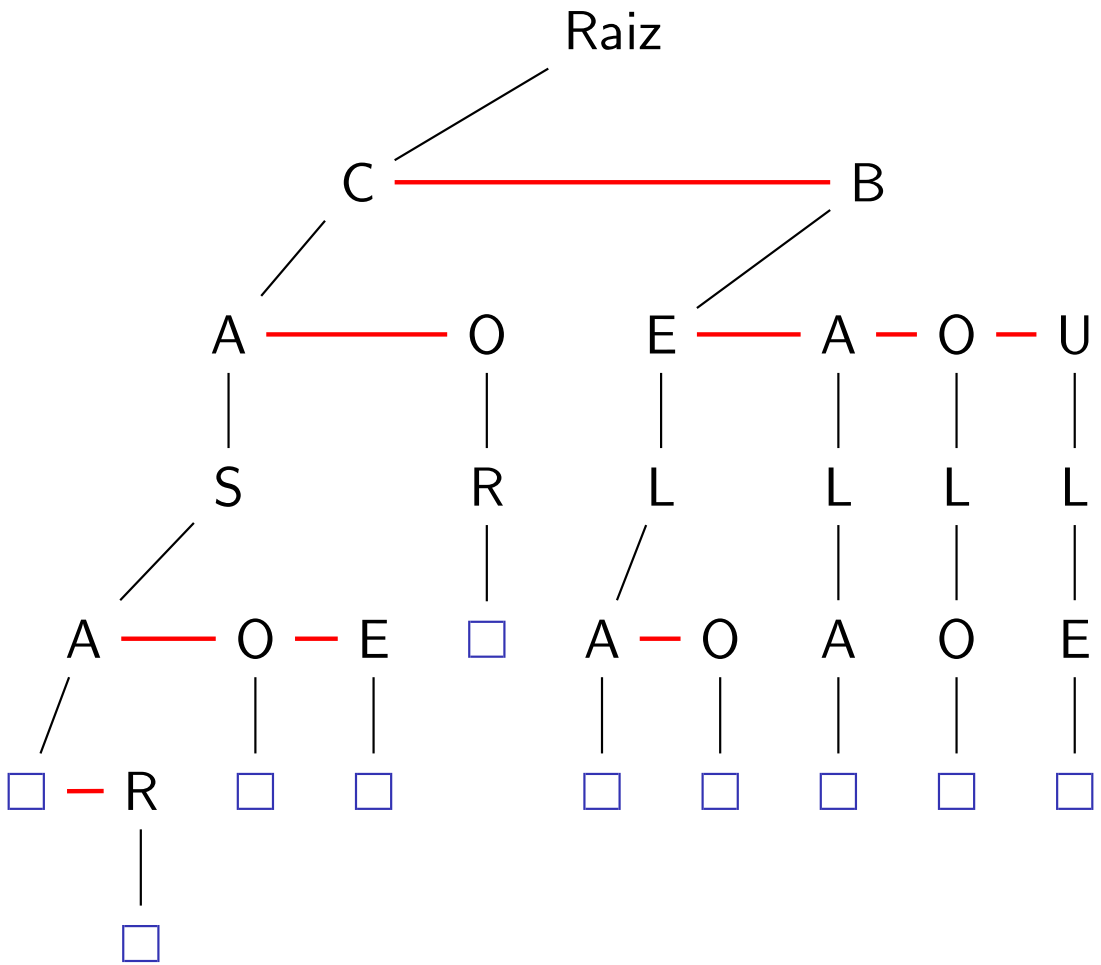
Trie com as palavras: **CASA, CASO, CASE, CASAR, COR, BELA, BELO, BALA, BOLO, BULE**



Representação como Árvore Binária

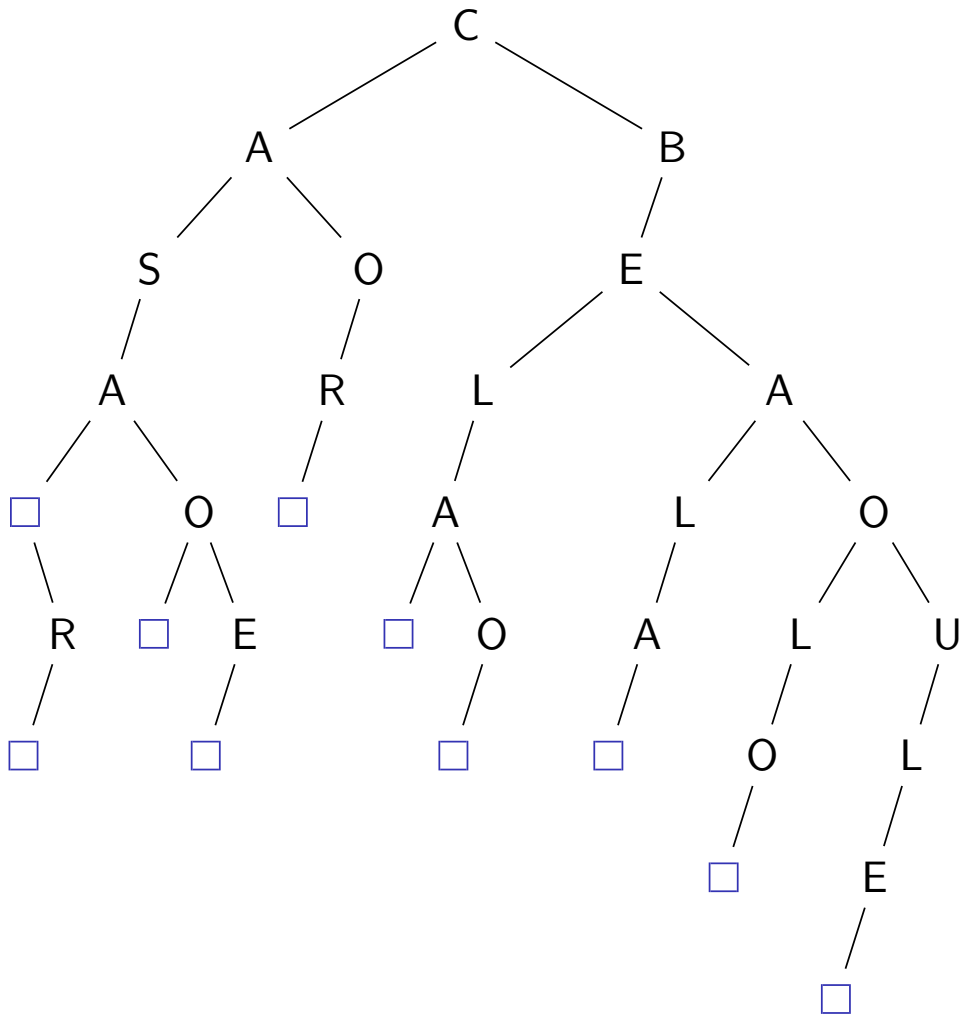


Representação como Árvore Binária



Representação como Árvore Binária

Eliminando a raiz e redesenhando:



Operações sobre a Representação Binária de Tries

Consulta

- A consulta de uma palavra consiste em, a partir da raiz, procurar a primeira letra da palavra no caminho formado apenas pelos filhos à direita dos nós visitados
- Se não encontrar, não há a palavra
- Se encontrar, caminha-se para o filho à esquerda do nó que corresponde à letra. Repete-se o mesmo procedimento para as demais letras
- Se encontrar todas as letras e, em seguida, a indicação de fim de palavra, a palavra está na *trie*.

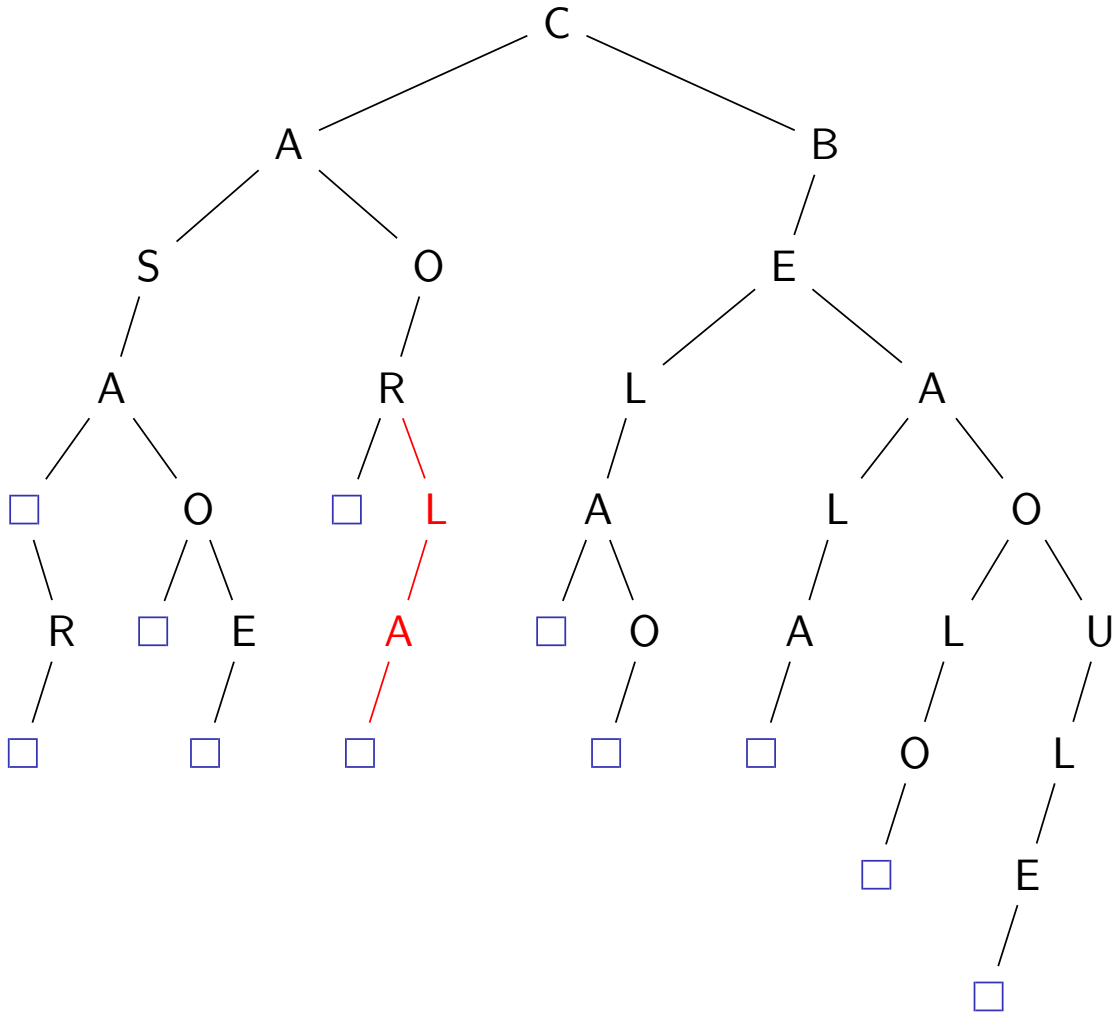
Operações sobre a Representação Binária de Tries

Inserção

- Os nós filhos de um nó x na representação de uma *trie* como árvore k-ária correspondem, na representação em árvore binária, ao filho à esquerda do nó que representa x e aos nós no caminho formado seguindo-se sempre os filhos à direita (até não haver mais)
- O procedimento de inserção corresponde a buscar o maior prefixo que está na árvore (seguindo o procedimento de consulta). Seja c o último caractere do prefixo encontrado. Insere-se então o caractere que segue c na palavra como um novo filho de c e, depois, os demais caracteres sempre como filho à esquerda do caractere anterior. Se não houver prefixo encontrado, faz-se este procedimento a partir da raiz.

Exemplo: Como seria a inserção da palavra COLA na *trie* apresentada anteriormente?

Inserção da Palavra COLA



Remoção

A remoção de uma palavra consiste em remover o maior sufixo da palavra que não é comum a outras palavras.

Aplicações Típicas de TRIE

- Dicionário de palavras
- Índice de registros por palavras
- Complementação de texto ao digitar prefixo de palavras
- Busca aproximada de padrões em texto (corretor ou verificador ortográfico)

Corretor Ortográfico usando TRIE

Possível abordagem:

Assume-se **um erro** por palavra (o mais comum) entre os tipos: **substituição**, **omissão**, **inserção** e **transposição**

Corretor Ortográfico usando TRIE

Possível abordagem:

Assume-se **um erro** por palavra (o mais comum) entre os tipos: **substituição**, **omissão**, **inserção** e **transposição**

1. **Substituição**: quando um caractere não é encontrado, continua-se a busca considerando outro caractere entre os filhos do último caractere da palavra

Corretor Ortográfico usando TRIE

Possível abordagem:

Assume-se **um erro** por palavra (o mais comum) entre os tipos: **substituição**, **omissão**, **inserção** e **transposição**

1. **Substituição**: quando um caractere não é encontrado, continua-se a busca considerando outro caractere entre os filhos do último caractere da palavra
2. **Omissão**: continua-se a busca a partir do próximo caractere

Corretor Ortográfico usando TRIE

Possível abordagem:

Assume-se **um erro** por palavra (o mais comum) entre os tipos: **substituição**, **omissão**, **inserção** e **transposição**

1. **Substituição**: quando um caractere não é encontrado, continua-se a busca considerando outro caractere entre os filhos do último caractere da palavra
2. **Omissão**: continua-se a busca a partir do próximo caractere
3. **Inserção**: continua-se a busca ignorado o suposto caractere adicionado

Corretor Ortográfico usando TRIE

Possível abordagem:

Assume-se **um erro** por palavra (o mais comum) entre os tipos: **substituição**, **omissão**, **inserção** e **transposição**

1. **Substituição**: quando um caractere não é encontrado, continua-se a busca considerando outro caractere entre os filhos do último caractere da palavra
2. **Omissão**: continua-se a busca a partir do próximo caractere
3. **Inserção**: continua-se a busca ignorado o suposto caractere adicionado
4. **Transposição**: troca-se a ordem do caractere corrente da palavra com o próximo e retoma-se a busca

Observações

- Os nós com marcadores de fim de palavra podem ser usados para armazenar informações ou apontadores para outros locais onde estão informações sobre a palavra: classe gramatical, ocorrências em um texto, tradução, etc.

Observações

- Os nós com marcadores de fim de palavra podem ser usados para armazenar informações ou apontadores para outros locais onde estão informações sobre a palavra: classe gramatical, ocorrências em um texto, tradução, etc.

- Pode-se usar algoritmos para eliminar representações duplicadas de sufixos em comum

Árvores PATRICIA

Árvores PATRICIA

- **PATRICIA**: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*

Árvores PATRICIA

- **PATRICIA**: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*

- Corresponde a uma representação compacta de uma *trie*, em que se condensam nós com apenas um filho

Árvores PATRICIA

- **PATRICIA**: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*
- Corresponde a uma representação compacta de uma *trie*, em que se condensam nós com apenas um filho
- É utilizada em buscas de cadeias longas de caracteres

Árvores PATRICIA

- **PATRICIA**: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*
- Corresponde a uma representação compacta de uma *trie*, em que se condensam nós com apenas um filho
- É utilizada em buscas de cadeias longas de caracteres
- Em cadeias de caracteres/*bits*, armazena em que caracteres/*bits* as cadeias se diferenciam

Árvores PATRICIA

- **PATRICIA**: *Practical Algorithm to Retrieve Information Coded In Alphanumeric*
- Corresponde a uma representação compacta de uma *trie*, em que se condensam nós com apenas um filho
- É utilizada em buscas de cadeias longas de caracteres
- Em cadeias de caracteres/*bits*, armazena em que caracteres/*bits* as cadeias se diferenciam
- É um tipo específico (ou usada como sinônimo) de árvore **radix**

PATRICIA: Idéia Geral

Árvore PATRICIA podem ser usadas com diferentes representações das cadeias de caracteres. Usaremos *bits* no exemplo.

Exemplo:

Sejam as representações binárias para as letras:

A=01001 B=00011 E=00101 L=11100 O=01101 U=10101

Teríamos as seguintes representações binárias para as palavras abaixo:

BOLA: 00011 01101 11100 01001
BELO: 00011 00101 11100 01101
BULE: 00011 10101 11100 00101
BALA: 00011 01001 11100 01001
BULA: 00011 10101 11100 01001
LUA: 11100 10101 01001

PATRICIA: Exemplo

Inserção de:

BOLA: 00011 01101 11100 01001

PATRICIA: Exemplo

Inserção de:

BOLA: 00011 01101 11100 01001

A árvore estava vazia. Cria-se um nó:

PATRICIA: Exemplo

Inserção de:

BOLA: 00011 01101 11100 01001

A árvore estava vazia. Cria-se um nó:

Inserção de:

BELO: 00011 00101 11100 01101

PATRICIA: Exemplo

Inserção de:

BOLA: 00011 01101 11100 01001

A árvore estava vazia. Cria-se um nó:

BOLA

Inserção de:

BELO: 00011 00101 11100 01101

Ao se chegar em um nó folha, verifica-se onde as palavras se diferem. Os nós internos indicam em que *bit* as palavras na subárvore se diferenciam.

PATRICIA: Exemplo

Inserção de:

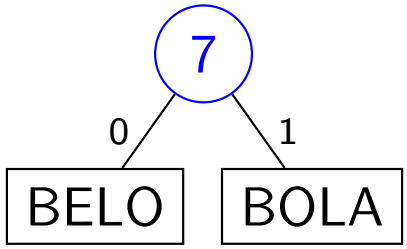
BOLA: 00011 01101 11100 01001

A árvore estava vazia. Cria-se um nó:

Inserção de:

BELO: 00011 00101 11100 01101

Ao se chegar em um nó folha, verifica-se onde as palavras se diferem. Os nós internos indicam em que *bit* as palavras na subárvore se diferenciam.



PATRICIA: Exemplo

Inserção de:

BULE: 00011 10101 11100 00101

PATRICIA: Exemplo

Inserção de:

BULE: 00011 10101 11100 00101

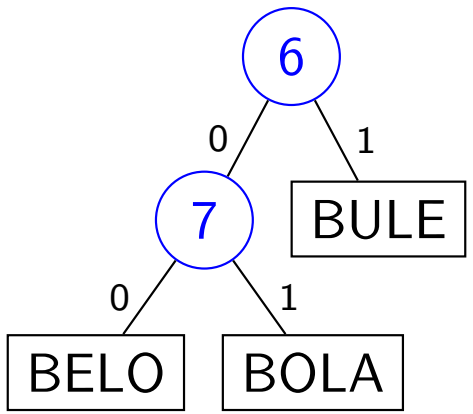
Como o sétimo *bit* é 0, segue-se à esquerda. Chega-se na folha e não é BULE. Comparam-se as duas sequências e atualiza-se a árvore:

PATRICIA: Exemplo

Inserção de:

BULE: 00011 10101 11100 00101

Como o sétimo *bit* é 0, segue-se à esquerda. Chega-se na folha e não é BULE. Comparam-se as duas sequências e atualiza-se a árvore:

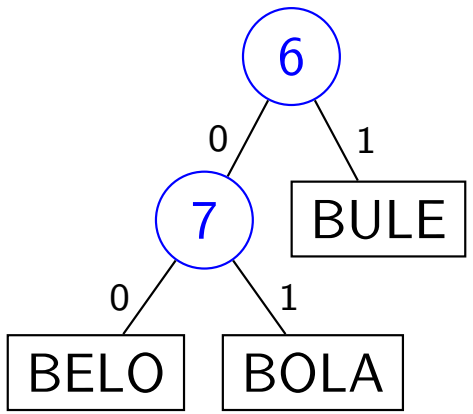


PATRICIA: Exemplo

Inserção de:

BULE: 00011 10101 11100 00101

Como o sétimo *bit* é 0, segue-se à esquerda. Chega-se na folha e não é BULE. Comparam-se as duas sequências e atualiza-se a árvore:



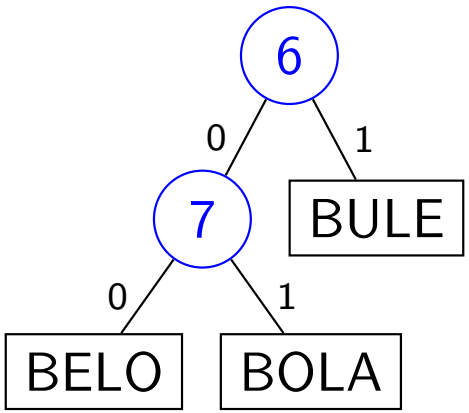
- A árvore mantém a informação sobre prefixos em comum

PATRICIA: Exemplo

Inserção de:

BULE: 00011 10101 11100 00101

Como o sétimo *bit* é 0, segue-se à esquerda. Chega-se na folha e não é BULE. Comparam-se as duas sequências e atualiza-se a árvore:



- A árvore mantém a informação sobre prefixos em comum
- Por que a subárvore existente permanece consistente?

PATRICIA: Exemplo

Inserção de:

BALA: 00011 01001 11100 01001

PATRICIA: Exemplo

Inserção de:

BALA: 00011 01001 11100 01001

Segue-se na árvore até uma folha. Chega-se em BOLA:

BOLA: 00011 01101 11100 01001

Comparam-se as duas palavras e atualiza-se a árvore.

PATRICIA: Exemplo

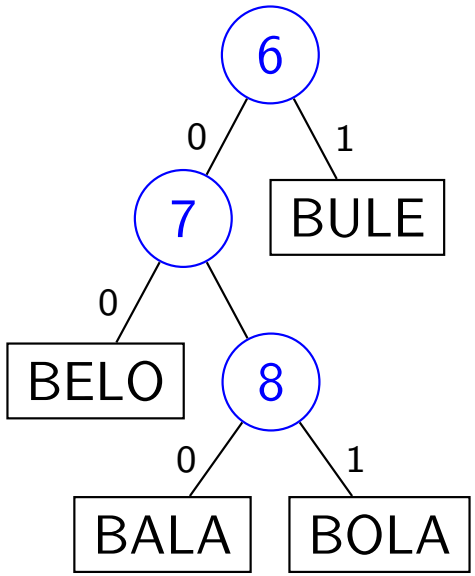
Inserção de:

BALA: 00011 01001 11100 01001

Segue-se na árvore até uma folha. Chega-se em BOLA:

BOLA: 00011 01101 11100 01001

Comparam-se as duas palavras e atualiza-se a árvore.



PATRICIA: Exemplo

Inserção de:

LUA: 11100 10101 01001

PATRICIA: Exemplo

Inserção de:

LUA: 11100 10101 01001

Segue-se na árvore até uma folha. Chega-se em BULE:

BULE: 00011 10101 11100 00101

Comparam-se as duas palavras e atualiza-se a árvore.

PATRICIA: Exemplo

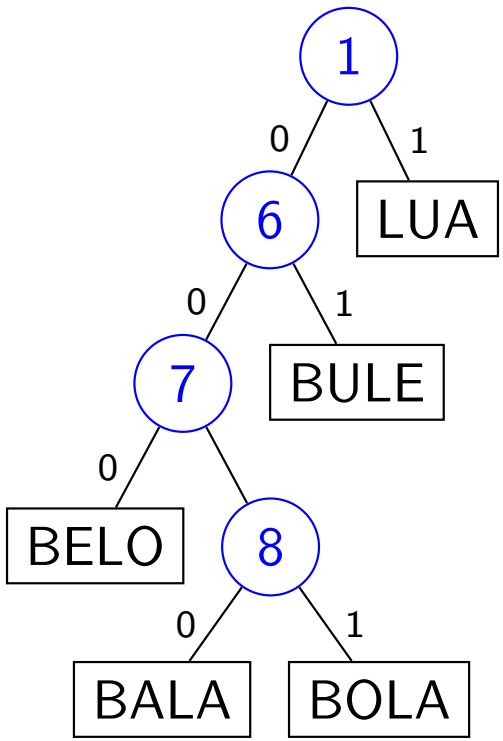
Inserção de:

LUA: 11100 10101 01001

Segue-se na árvore até uma folha. Chega-se em BULE:

BULE: 00011 10101 11100 00101

Comparam-se as duas palavras e atualiza-se a árvore.



PATRICIA: Exemplo

Inserção de:

BULA: 00011 10101 11100 01001

PATRICIA: Exemplo

Inserção de:

BULA: 00011 10101 11100 01001

Segue-se na árvore até uma folha. Chega-se em BULE:

BULE: 00011 10101 11100 00101

Comparam-se as duas palavras e atualiza-se a árvore.

PATRICIA: Exemplo

Inserção de:

BULA: 00011 10101 11100 01001

Segue-se na árvore até uma folha. Chega-se em BULE:

BULE: 00011 10101 11100 00101

Comparam-se as duas palavras e atualiza-se a árvore.