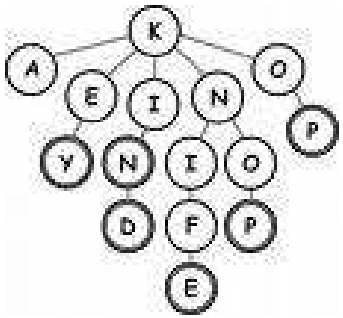
ÁRVORES TRIES

0. Introdução

- TRIE vem de RETRIEVAL – RECUPERAÇÃO
- A pronúncia: TRI ou TRAI
- É um tipo de árvore de busca como B, ISAM, quadtrees, e vermelho-preto.
- Idéia geral: usar partes das CHAVES como caminho busca
- Origem: anos 60 por Edward Fredkin

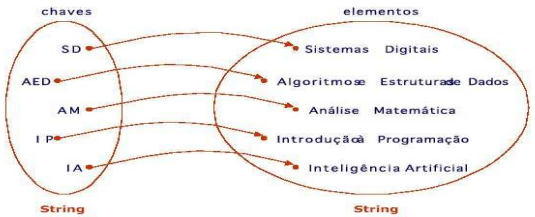
0. Introdução

Edward Fredkin (1934 -)



1. CHAVES: Características

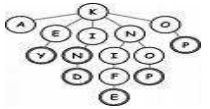
- Cada chave formada por palavras sobre um alfabeto
- Palavras com tamanho variável e ilimitado
- Em geral associam-se chaves a elementos ou registros



1. CHAVES: Características

- Cada chave é formada a partir de alfabeto de símbolos
- Exemplos de alfabetos:
  {0,1}, {A, B, C, D, E,...Z}, {0,1,2,3,4,5,...,9}
- Exemplos de chaves:
  ABABBBABABA 19034717 Maria
  01010101000000000101000000001010
- Chaves parcialmente partilhadas entre os elementos

2. Trie: A estrutura



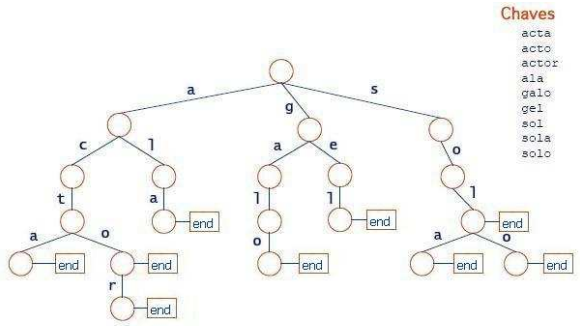
- Árvore ordenada e n-ária
- Chaves em geral caracteres
- Ao contrário da árvore de busca binária nenhum nó armazena a chave
- Chave determinada pela posição na árvore

2. Trie: A estrutura

- Descendentes de mesmo nó com mesmo prefixo
- Raiz: cadeia vazia
- Valores ou elementos associados a folhas ou a alguns nós internos de interesse
- O caminho da raiz para qualquer outro nó é um prefixo de uma *string*

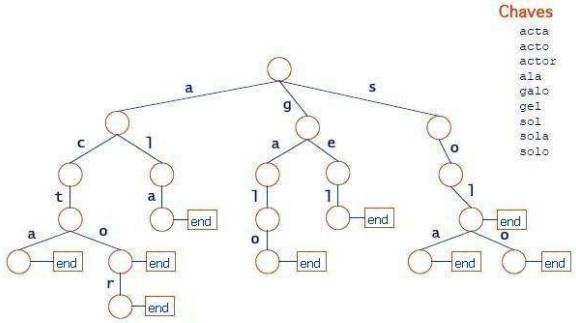
2. Trie: A estrutura

- Árvore ordenada e n-ária



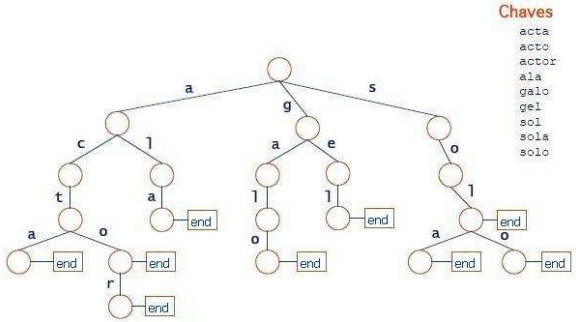
2. Trie: A estrutura

- Chaves em geral caracteres



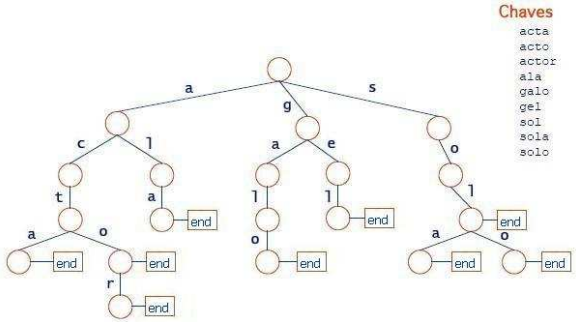
2. Trie: A estrutura

- Ao contrário da árvore de busca binária nenhum nó armazena a chave



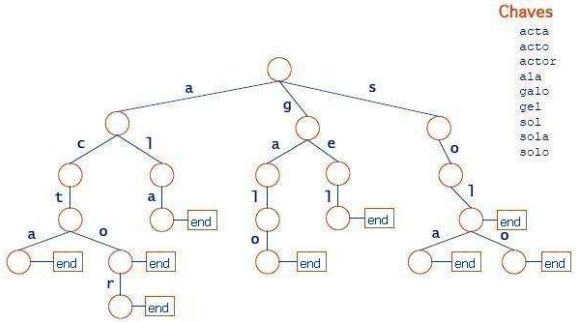
2. Trie: A estrutura

- Chave determinada pela posição na árvore



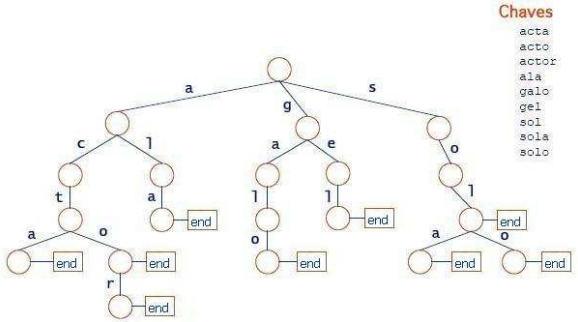
2. Trie: A estrutura

- Descendentes de mesmo nó com mesmo prefixo



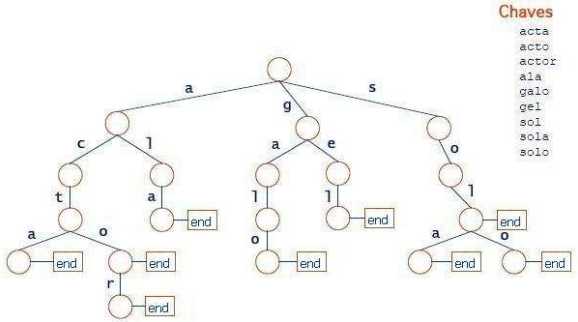
2. Trie: A estrutura

- Raiz: cadeia vazia



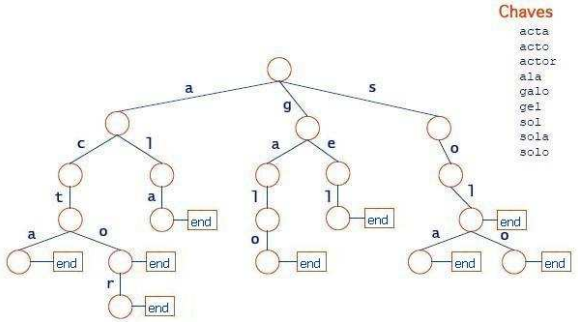
2. Trie: A estrutura

- Valores ou elementos associados a folhas ou a alguns nós internos de interesse



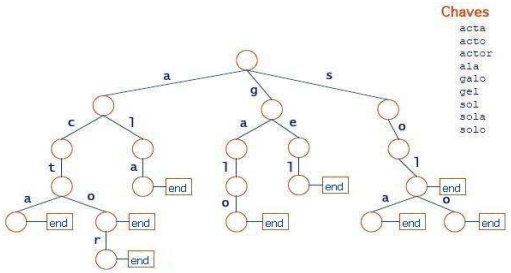
2. Trie: A estrutura

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



2. Trie: A estrutura

- O grau corresponde ao tamanho do alfabeto
- Cada nível que se desce corresponde a avançar um elemento na chave



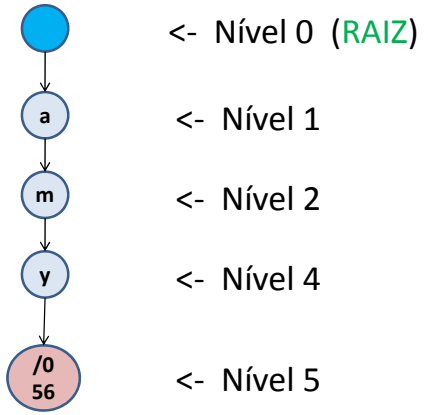
3. Montando uma árvore TRIE

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

Estes são pares que queremos colocar na árvore TRIE

3. Montando uma árvore TRIE

- amy 56



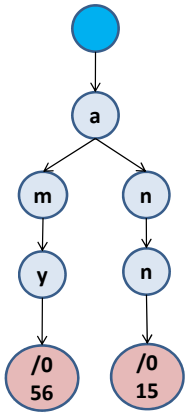
3. Montando uma árvore TRIE

- INSIRA

ann 15

3. Montando uma árvore TRIE

- ann 15



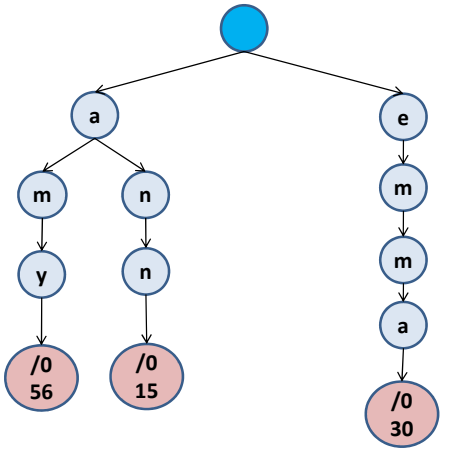
3. Montando uma árvore TRIE

- INSIRA

emma 30

3. Montando uma árvore TRIE

- emma 30



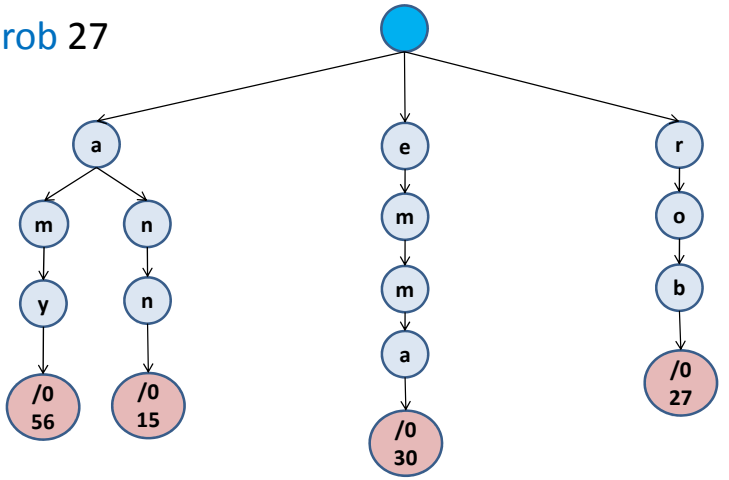
3. Montando uma árvore TRIE

- INSIRA

rob 27

3. Montando uma árvore TRIE

- rob 27



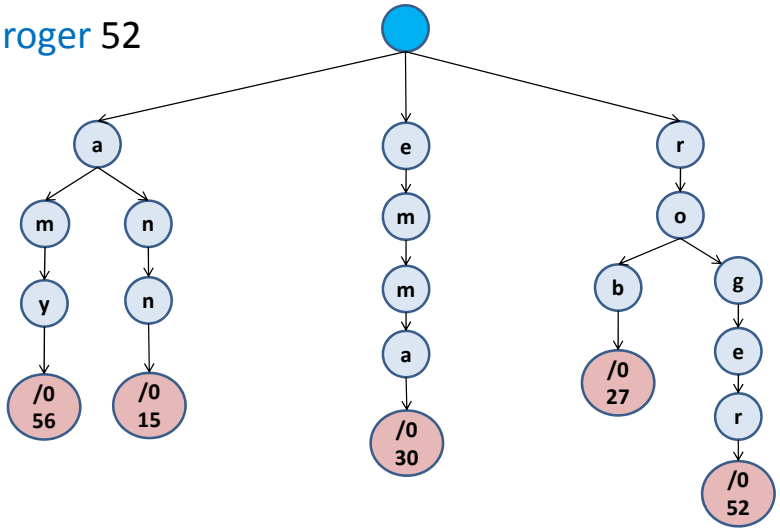
3. Montando uma árvore TRIE

- INSIRA

roger 52

3. Montando uma árvore TRIE

- roger 52



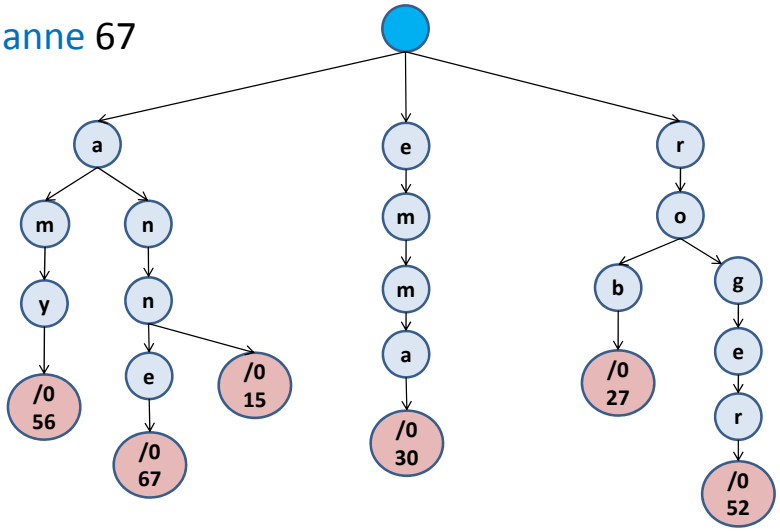
3. Montando uma árvore TRIE

- INSIRA

anne 67

3. Montando uma árvore TRIE

- anne 67



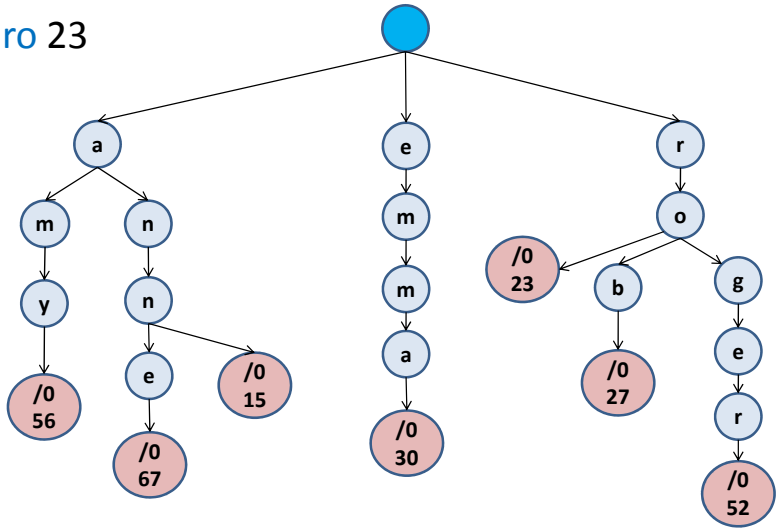
3. Montando uma árvore TRIE

- INSIRA

ro 23

3. Montando uma árvore TRIE

- ro 23



4. Operações em TRIES

- INSERÇÃO
- ALGORITMO "É MEMBRO"
- REMOÇÃO

4. Operações em TRIES

- INSERÇÃO

  Faz-se uma busca pela palavra a ser inserida.

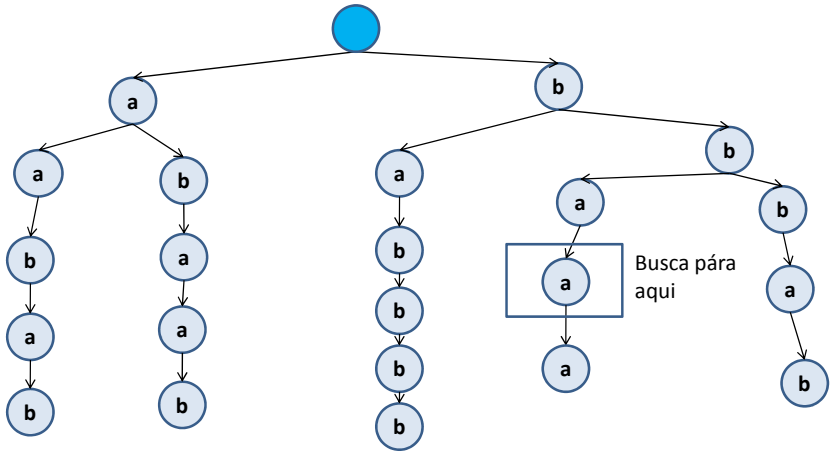
  Se ela já existir na TRIE nada é feito.

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

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

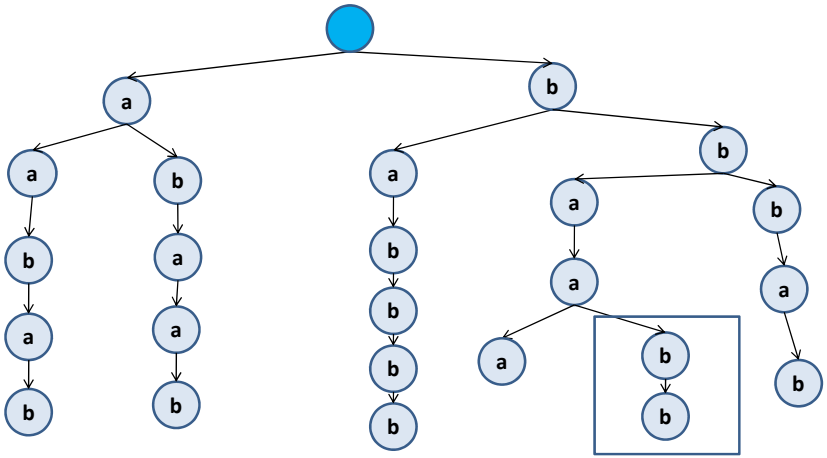
4. Operações em TRIES

Inserção : bbaabb



4. Operações em TRIES

Inserção : bbaabb



4. Operações em TRIES

- É MEMBRO
  1. Busca no nível superior o nodo que confere com o primeiro caractere (corrente) 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

4. Operações em TRIES

- REMOÇÃO

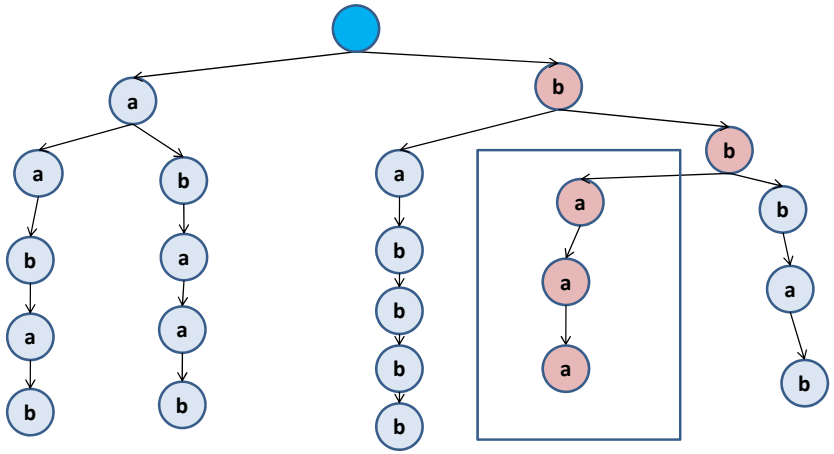
  Busca-se o nó que representa o final da palavra a ser removida.

  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

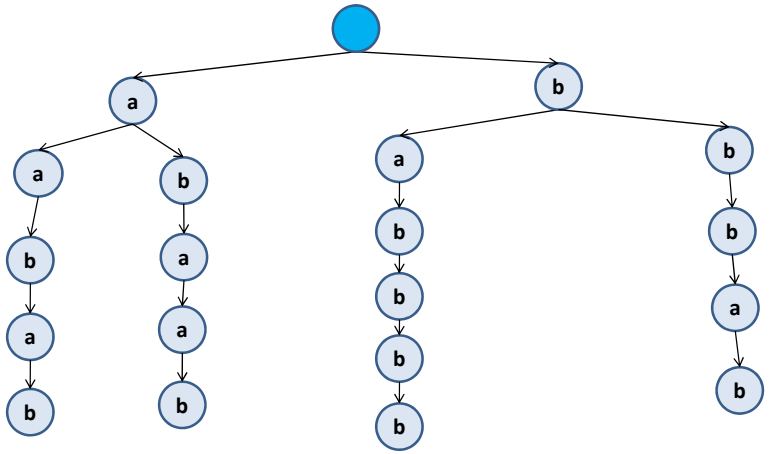
4. Operações em TRIES

Remoção : bbaaa



4. Operações em TRIES

Remoção : bbaaa



4. Operações em TRIES

- COMPLEXIDADE

  A altura da árvore é igual ao comprimento da chave mais longa

  O tempo de execução das operações não depende do número de elementos da árvore

  Complexidade: O (AK)

  A = tamanho do alfabeto

  K = tamanho da chave

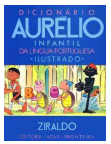
  A utilização de uma TRIE só compensa se o acesso aos componentes individuais das chaves for bastante rápido

  Quanto maior a estrutura mais eficiente uso do espaço

  Para enfrentar o desperdício de espaço com estruturas pequenas foram criadas as árvores PATRÍCIA

7. Aplicações das TRIES

Dicionários (telefone celular)



Corretores Ortográficos

Programas para compreender Linguagem Natural

Auto-preenchimento:

browsers,

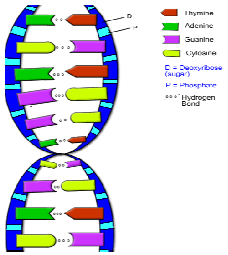
e-mail,

linguagens de programação

7. Aplicações das TRIES

*Compressão de dados

*Biologia computacional



* Tabelas de roteamento para endereços IP
* Armazenar e consultar documentos XML
* Fundamental para o Burstsort (o método mais rápido de ordenação de strings em memória/cache)
* Tabelas de símbolos em compiladores