Árvores Digitais

Letícia Rodrigues Bueno

UFABC

Busca Digital

Busca Digital

- **Problema geral de busca:** conjunto de chaves S e chave x a localizar em S;

Busca Digital

- **Problema geral de busca:** conjunto de chaves S e chave x a localizar em S;
- Assumido até agora:

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;
  - tamanho da chave permite armazenamento em memória de forma eficiente;

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;
  - tamanho da chave permite armazenamento em memória de forma eficiente;
  - chaves têm mesmo tamanho;

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;
  - tamanho da chave permite armazenamento em memória de forma eficiente;
  - chaves têm mesmo tamanho;
- **E se a busca consistir em frases em texto literário?**

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;
  - tamanho da chave permite armazenamento em memória de forma eficiente;
  - chaves têm mesmo tamanho;
- **E se a busca consistir em frases em texto literário?**
- Utilizamos **busca digital**;

Busca Digital

- **Problema geral de busca:** conjunto de chaves *S* e chave *x* a localizar em *S*;
- Assumido até agora:
  - chaves são elementos indivisíveis;
  - tamanho da chave permite armazenamento em memória de forma eficiente;
  - chaves têm mesmo tamanho;
- **E se a busca consistir em frases em texto literário?**
- Utilizamos **busca digital**;
- Estrutura utilizada: **árvore digital**;

Árvores Digitais ou *Trie*

Árvores Digitais ou *Trie*

- **Trie:** originado de *“information re**TRIE**val”* devido a aplicação em recuperação de informação;

Árvores Digitais ou *Trie*

- **Trie:** originado de *“information re**TRIE**val”* devido a aplicação em recuperação de informação;
- Na comparação de chaves:

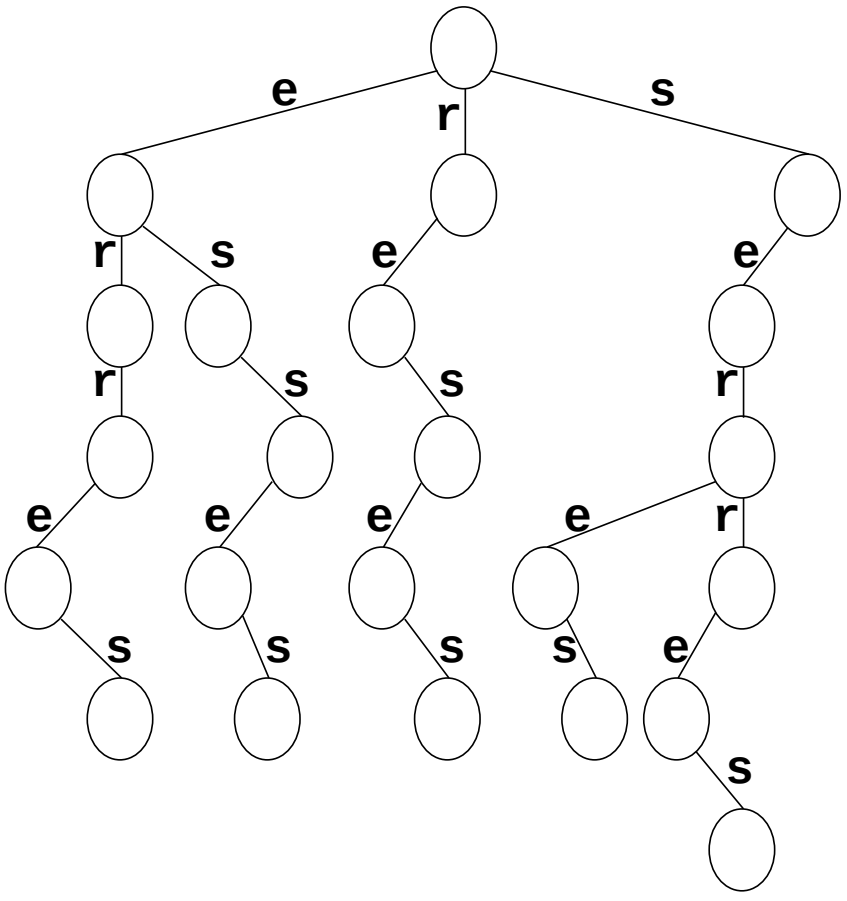
Árvores Digitais ou *Trie*

- **Trie:** originado de *“information re**TRIE**val”* devido a aplicação em recuperação de informação;
- Na comparação de chaves:
  - **NÃO** compara chave procurada com chaves do conjunto armazenado;

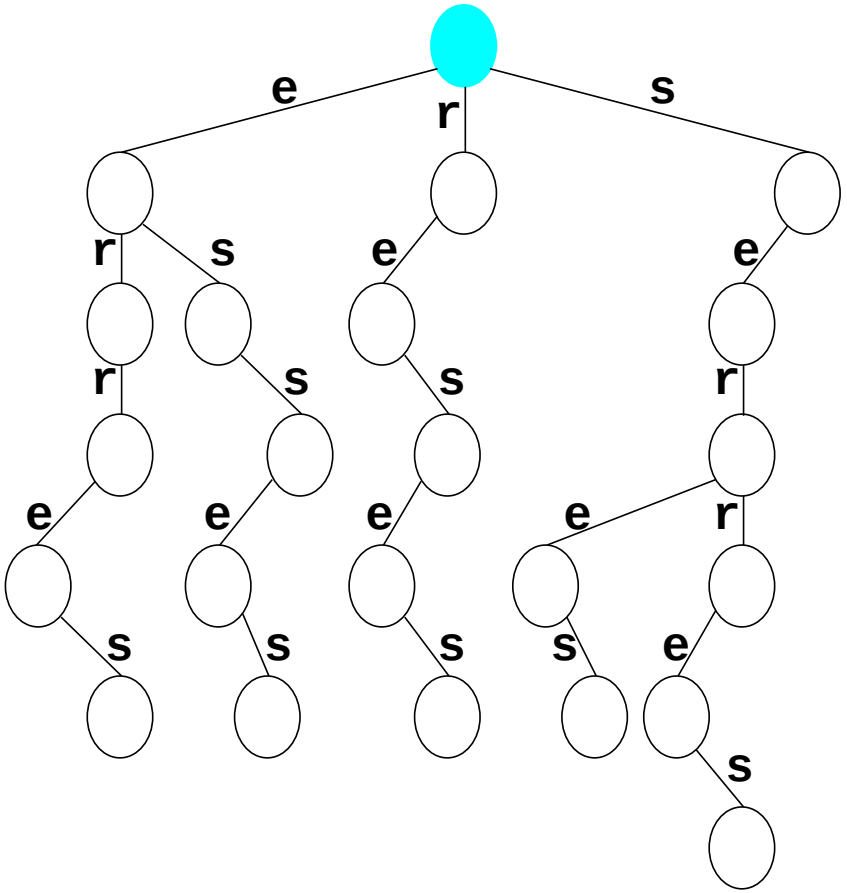
Árvores Digitais ou *Trie*

- **Trie:** originado de *“information re**TRIE**val”* devido a aplicação em recuperação de informação;
- Na comparação de chaves:
  - **NÃO** compara chave procurada com chaves do conjunto armazenado;
  - **SIM,** compara dígitos da chave individualmente. Número de passos igual tamanho da chave;

Exemplo de Árvore Digital

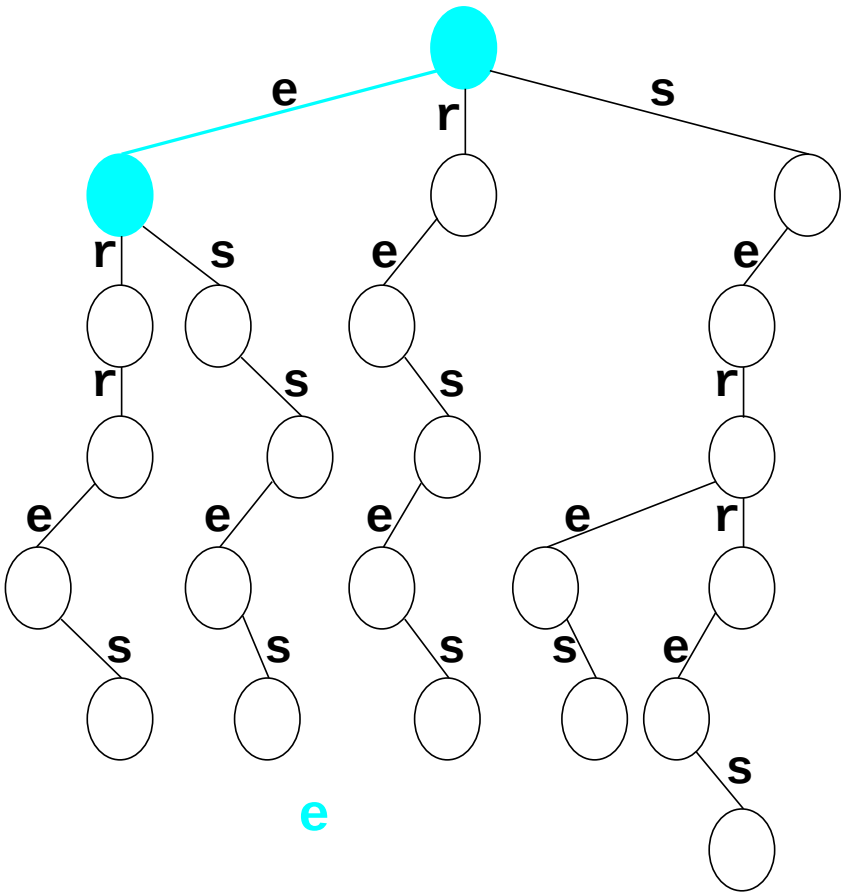


Exemplo de Árvore Digital



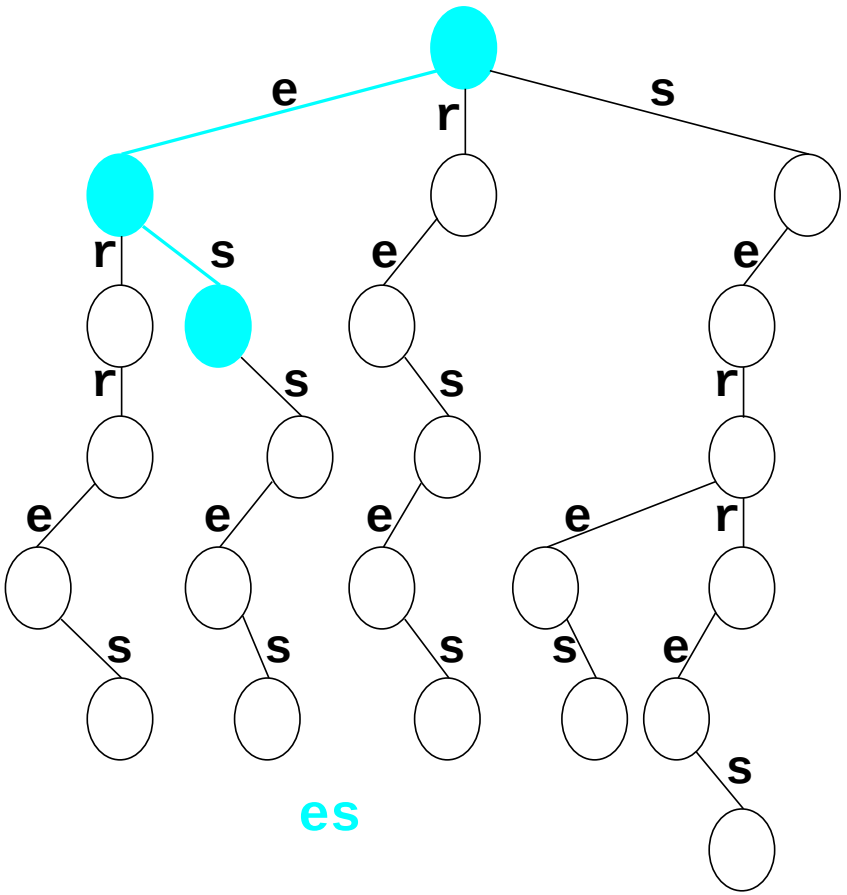
Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



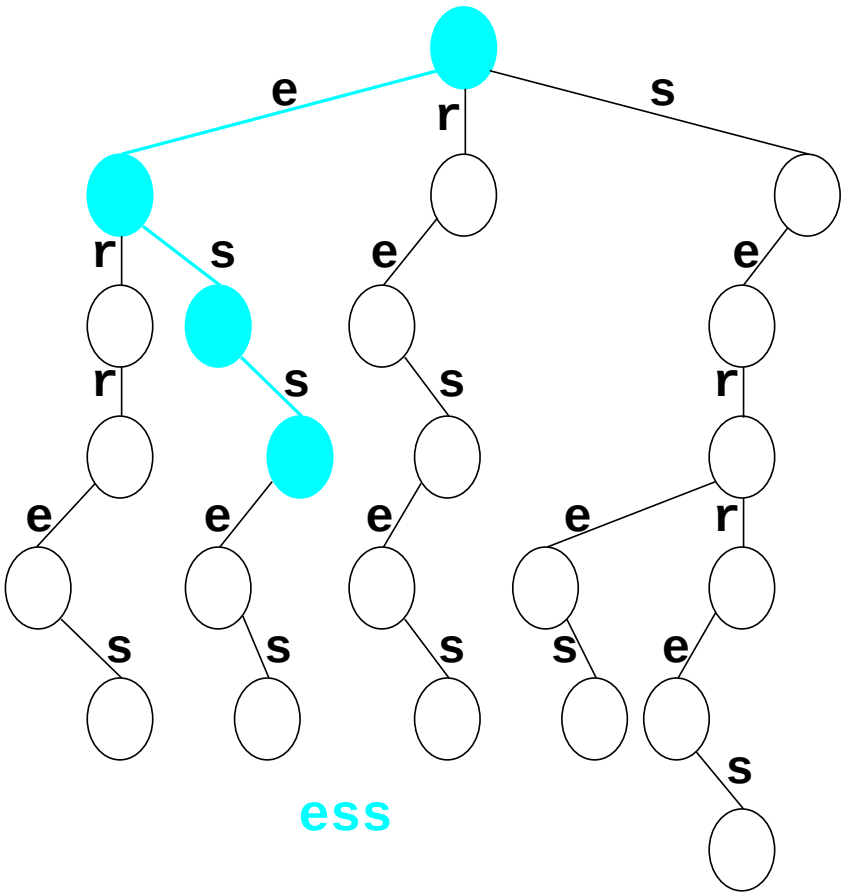
Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



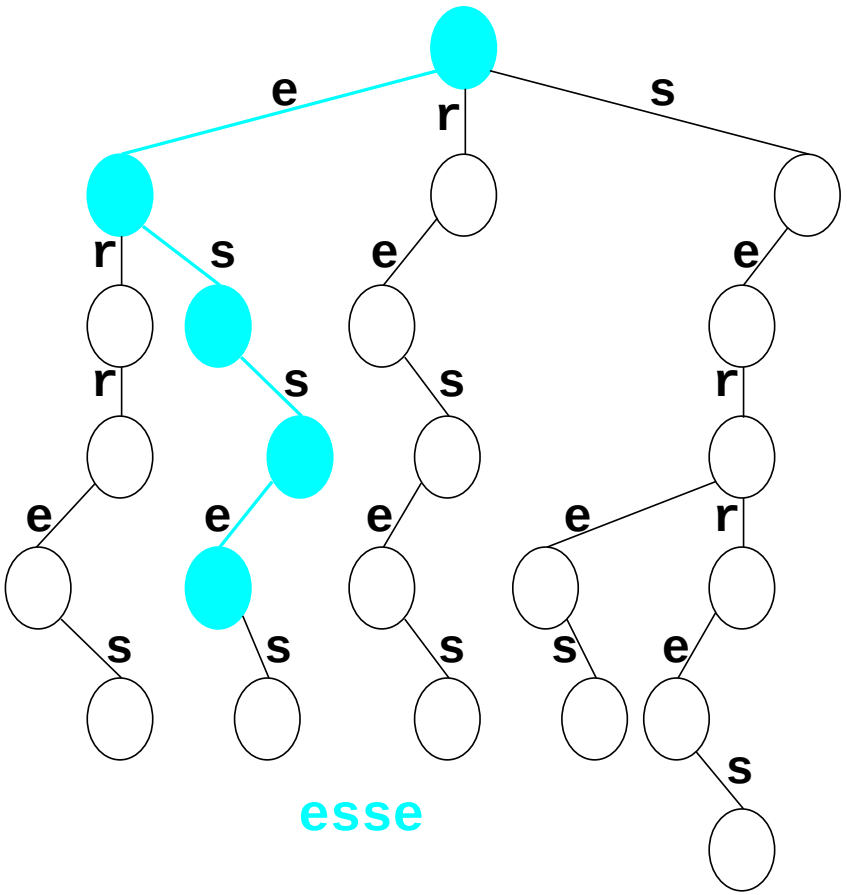
Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



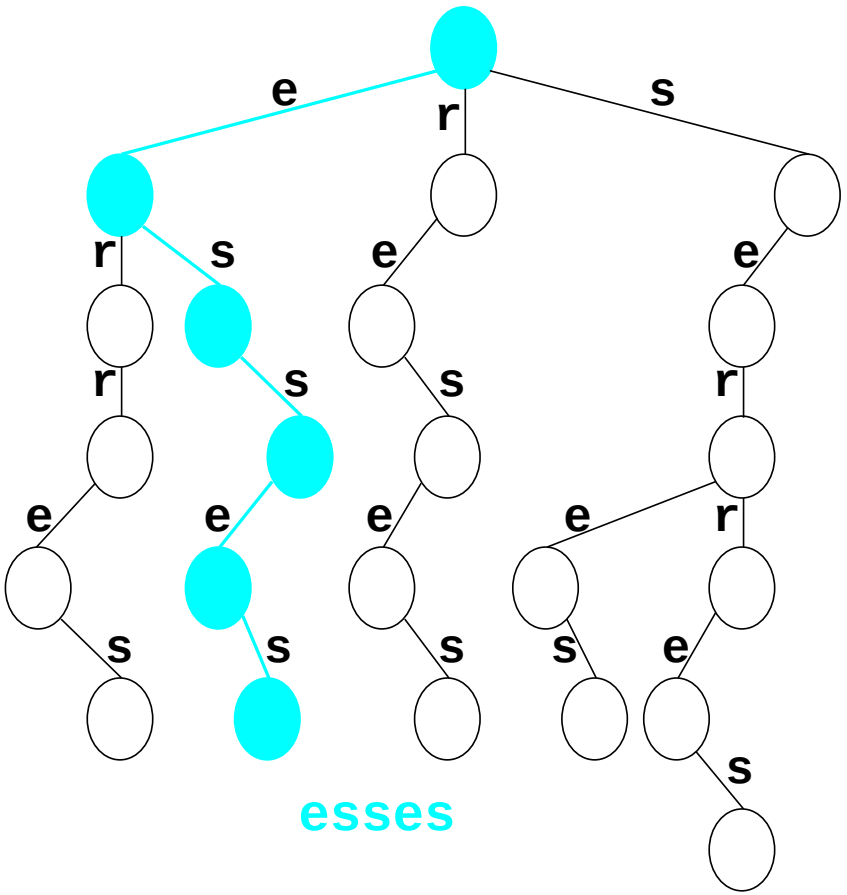
Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



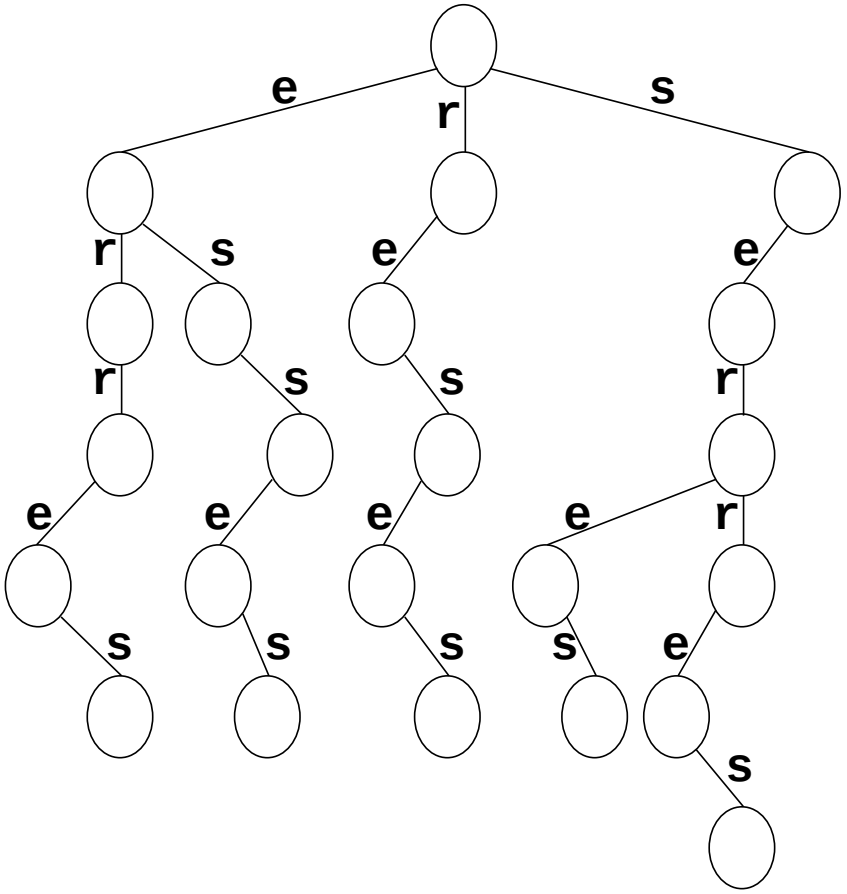
Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



Alfabeto:
$\{e, r, s\}$

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

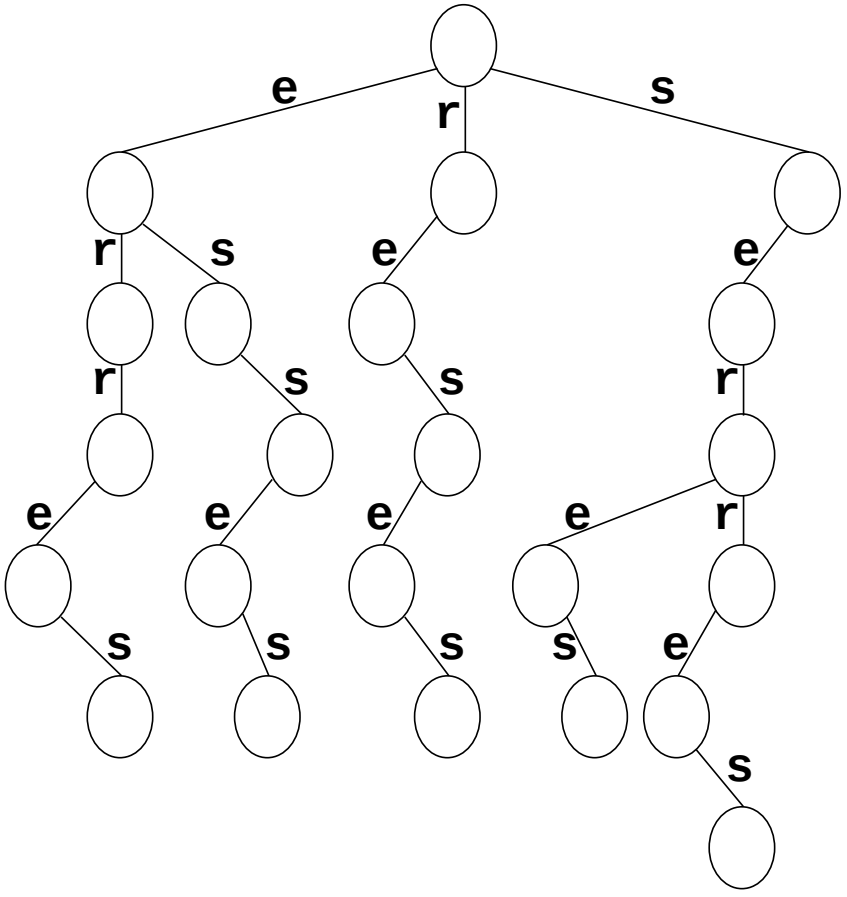
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

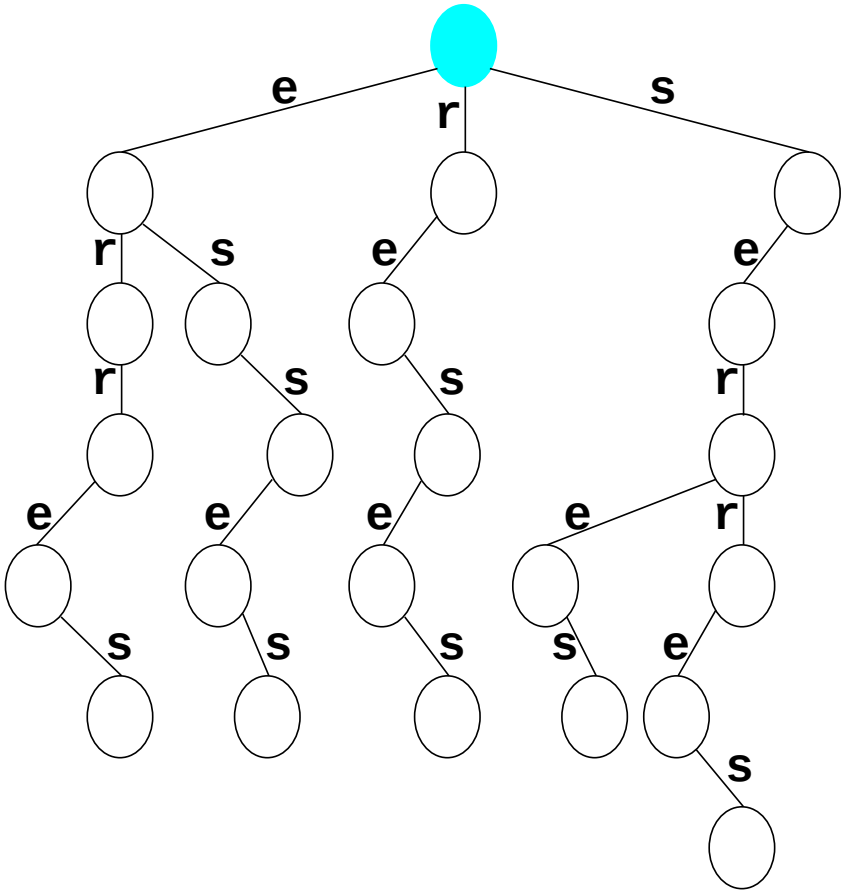
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

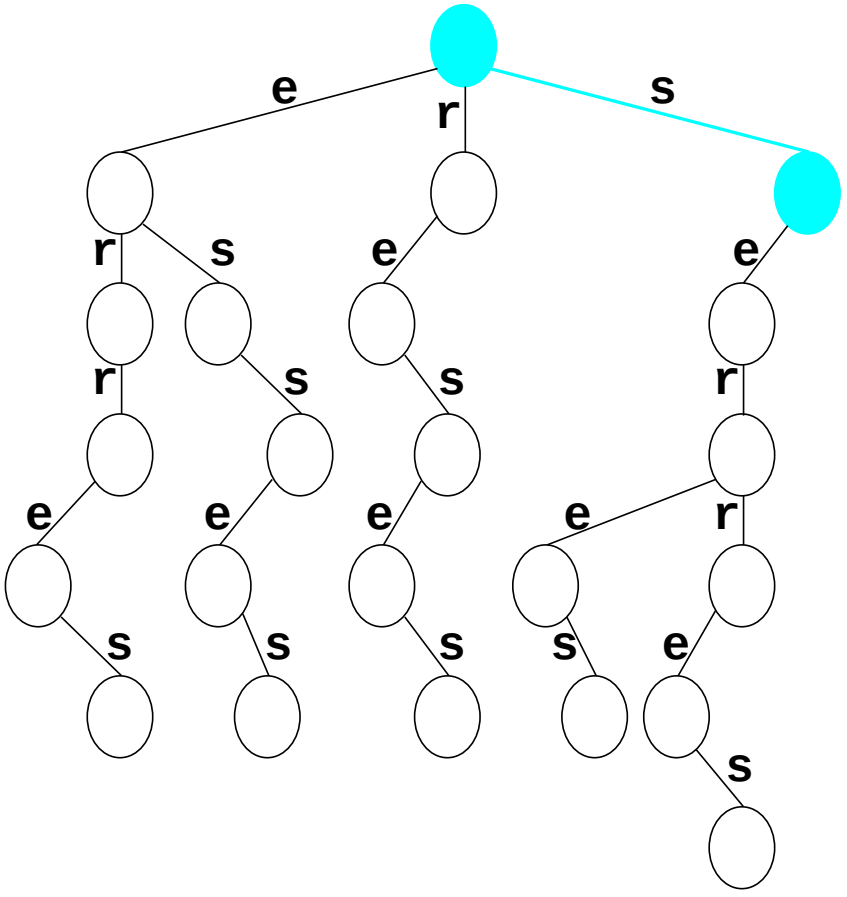
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

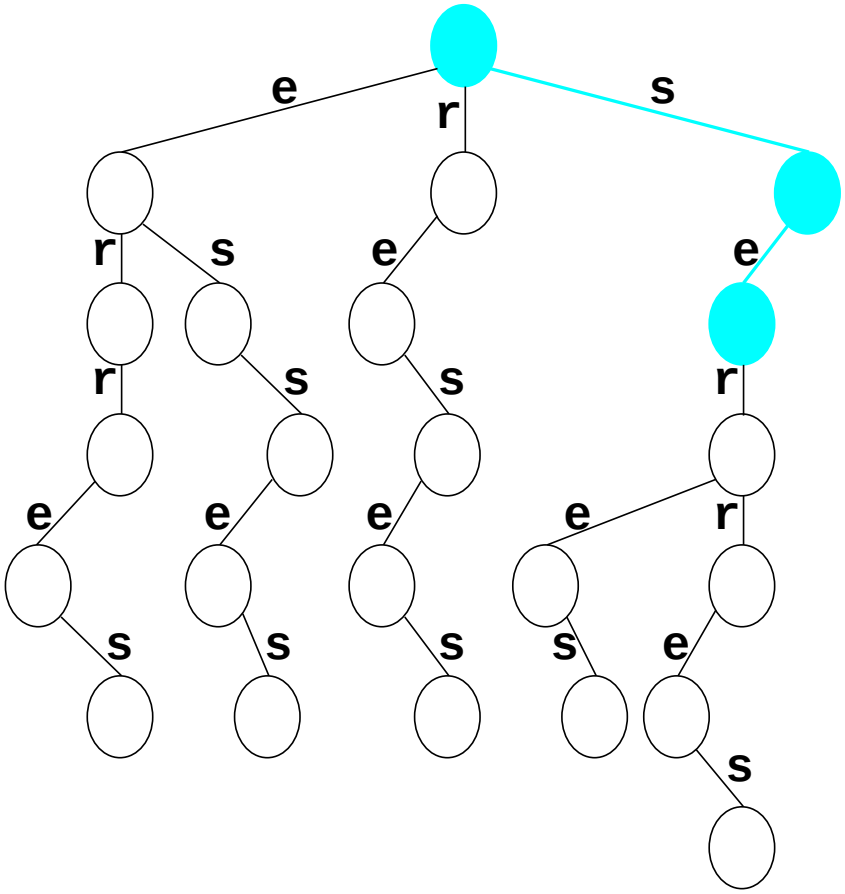
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

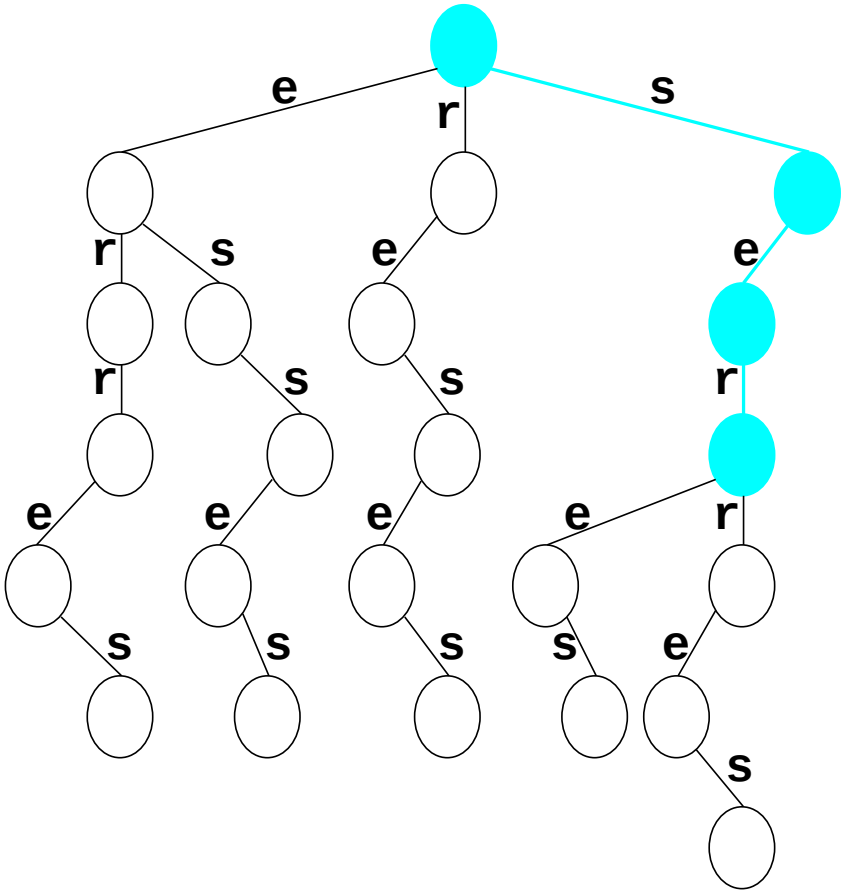
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

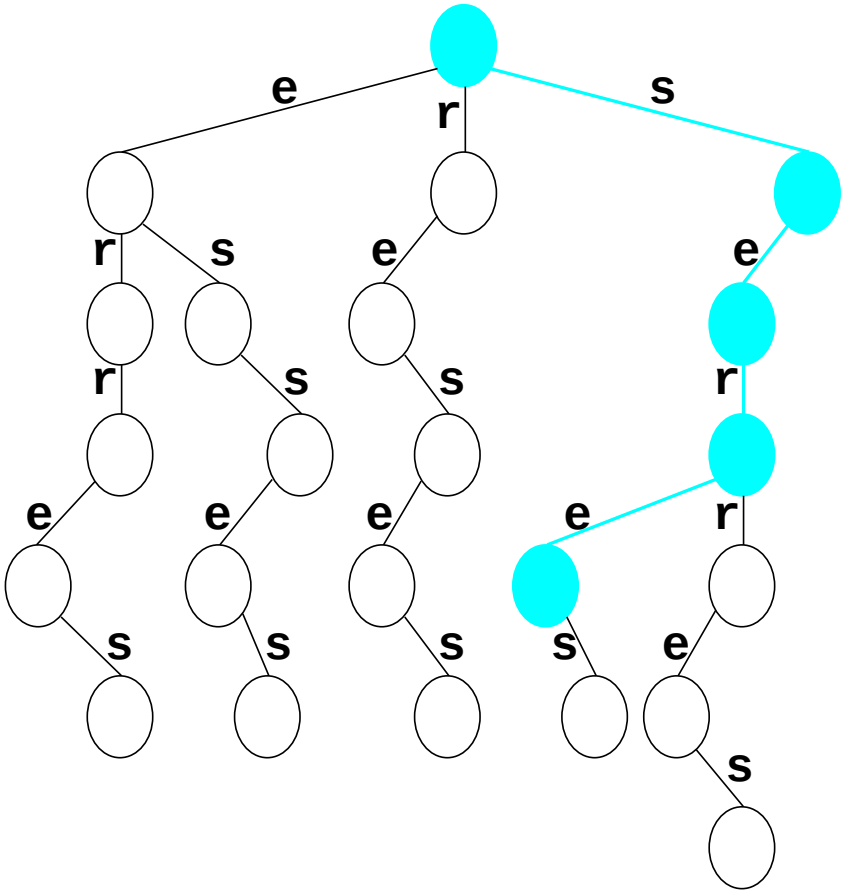
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

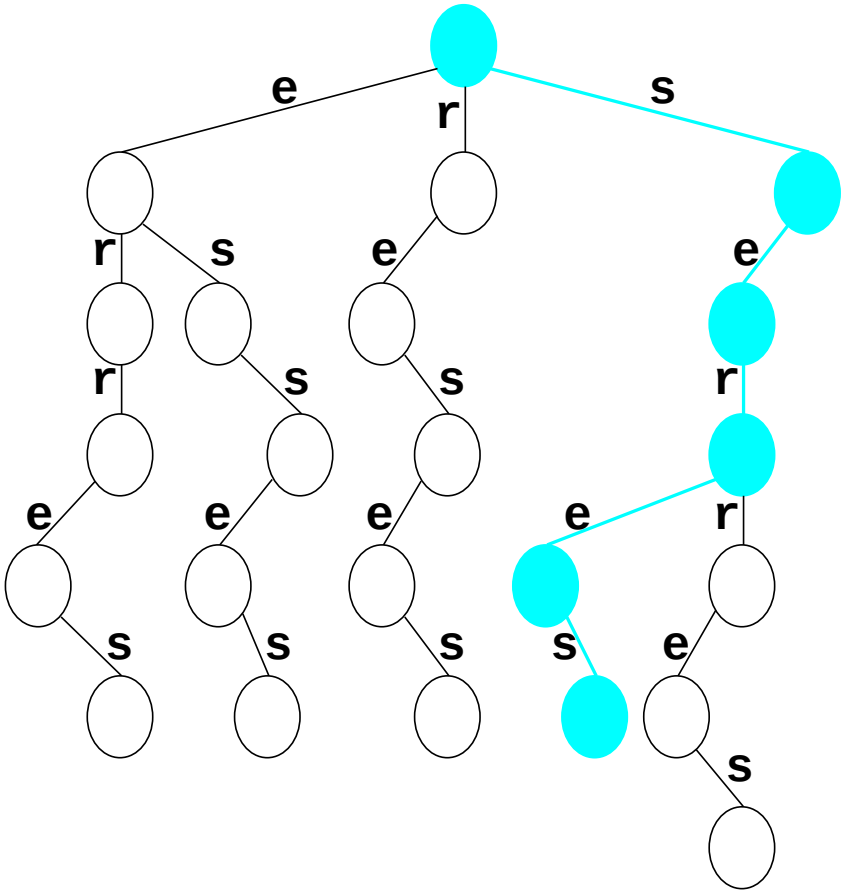
rese

reses

serres

seres

Exemplo de Árvore Digital



erre

erres

es

esse

esses

se

ser

serre

re

res

rese

reses

serres

seres

Árvores Digitais ou *Trie*

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;
- Alfabeto de S: $d_1 < d_2 < \ldots < d_m$;

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;
- Alfabeto de S: $d_1 < d_2 < \ldots < d_m$;
- p primeiros dígitos de chave: *prefixo*;

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;
- Alfabeto de S: $d_1 < d_2 < \ldots < d_m$;
- p primeiros dígitos de chave: *prefixo*;
- Árvore digital é árvore m-ária T não vazia onde:

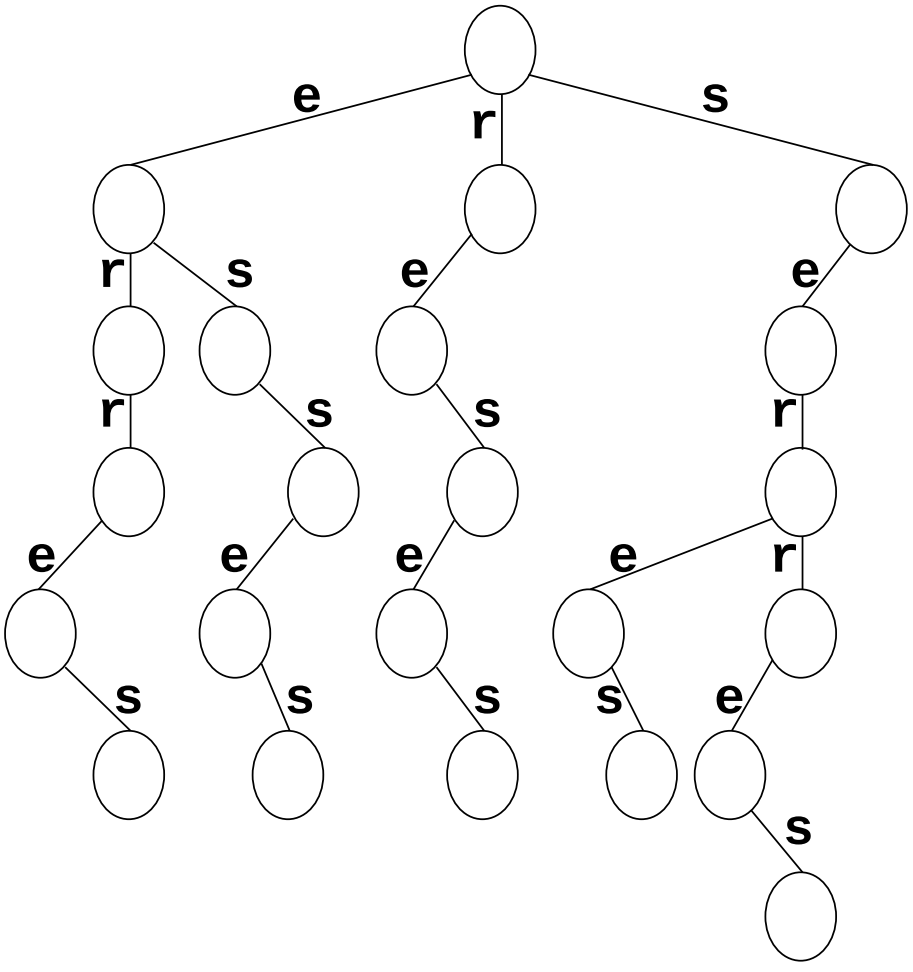
Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;
- Alfabeto de S: $d_1 < d_2 < \ldots < d_m$;
- p primeiros dígitos de chave: *prefixo*;
- Árvore digital é árvore m-ária T não vazia onde:
  1. Nó v é j-ésimo filho de seu pai $\Rightarrow$ v corresponde dígito d_j;

Árvores Digitais ou *Trie*

- Conjunto chaves: $S = \{s_1, s_2, \ldots, s_n\}$;
- s_i é sequência dígitos d_j;
- Alfabeto de S: $d_1 < d_2 < \ldots < d_m$;
- p primeiros dígitos de chave: *prefixo*;
- Árvore digital é árvore m-ária T não vazia onde:
  1. Nó v é j-ésimo filho de seu pai $\Rightarrow$ v corresponde dígito d_j;
  2. Sequência de dígitos da raiz até um nó corresponde a prefixo de alguma chave de S.

Exemplo de Árvore Digital



Árvore ternária;

Alfabeto:
$\{e, r, s\} \therefore e < r < s$
$m = 3$

$S = \{erre, erres, es,$
$esse, esses, se,$
$ser, serre, re,$
$res, rese, reses,$
$serres, seres\}$

Árvores Digitais: algoritmo de busca

```
1  buscaDigital(x, pt, l, a):
2      se l < k então
3          seja j a posição de d(l + 1)
               na ordenação do alfabeto
4          se pt.pont[j] ≠ null então
5              pt ← pt.pont[j];
6              l ← l + 1;
7              buscaDigital(x, pt, l, a);
8      senão se pt.terminal = true então
9          a ← 1;
```

Árvores Digitais: algoritmo de busca

```
1  buscaDigital(x, pt, l, a):
2      se l < k então
3          seja j a posição de d(l + 1)
               na ordenação do alfabeto
4          se pt.pont[j] ≠ null então
5              pt ← pt.pont[j];
6              l ← l + 1;
7              buscaDigital(x, pt, l, a);
8      senão se pt.terminal = true então
9          a ← 1;
```

Análise da complexidade:

Árvores Digitais: algoritmo de busca

```
1  buscaDigital(x, pt, l, a):
2      se l < k então
3          seja j a posição de d(l + 1)
               na ordenação do alfabeto
4          se pt.pont[j] ≠ null então
5              pt ← pt.pont[j];
6              l ← l + 1;
7              buscaDigital(x, pt, l, a);
8      senão se pt.terminal = true então
9          a ← 1;
```

Análise da complexidade:

- **Linha** 3: gasta $O(\log m)$ usando busca binária;

Árvores Digitais: algoritmo de busca

```
1  buscaDigital(x, pt, l, a):
2      se l < k então
3          seja j a posição de d(l + 1)
               na ordenação do alfabeto
4          se pt.pont[j] ≠ null então
5              pt ← pt.pont[j];
6              l ← l + 1;
7              buscaDigital(x, pt, l, a);
8      senão se pt.terminal = true então
9          a ← 1;
```

Análise da complexidade:

- **Linha** 3: gasta $O(\log m)$ usando busca binária;
- **Complexidade total**: $O(k \cdot \log m)$;

Árvores Digitais: algoritmo de busca

```
1  buscaDigital(x, pt, l, a):
2      se l < k então
3          seja j a posição de d(l + 1)
               na ordenação do alfabeto
4          se pt.pont[j] ≠ null então
5              pt ← pt.pont[j];
6              l ← l + 1;
7              buscaDigital(x, pt, l, a);
8      senão se pt.terminal = true então
9          a ← 1;
```

x é a chave procurada com k dígitos.

Chamada inicial:

$l \leftarrow 0$; $a \leftarrow 0$; $buscaDigital(x, raiz, l, a)$

Análise da complexidade:

- **Linha 3**: gasta $O(\log m)$ usando busca binária;
- **Complexidade total**: $O(k \cdot \log m)$;
- Representação binária de dígitos faz complexidade: $O(k)$;

Árvores Digitais: algoritmo de inserção

```
 1  insereDigital(x, pt):
 2      pt ← ptraiz; l ← a ← 0;
 3      buscaDigital(x, pt, l, a);
 4      se a = 0 então
 5          para h = l + 1, ..., k faça
 6              seja j a posição de
                    d(h) no alfabeto;
 7              ocupar(ptz);
 8              para i = 1, ..., m faça
 9                  ptz.pont[i] ← null;
10              pt.pont[j] ← ptz;
11              ptz.terminal ← false;
12              pt ← ptz;
13          pt.terminal ← true;
14      senão "inclusão inválida";
```

Árvores Digitais: algoritmo de inserção

```
 1  insereDigital(x, pt):
 2      pt ← ptraiz; l ← a ← 0;
 3      buscaDigital(x, pt, l, a);
 4      se a = 0 então
 5          para h = l + 1, ..., k faça
 6              seja j a posição de
                    d(h) no alfabeto;
 7              ocupar(ptz);
 8              para i = 1, ..., m faça
 9                  ptz.pont[i] ← null;
10              pt.pont[j] ← ptz;
11              ptz.terminal ← false;
12              pt ← ptz;
13          pt.terminal ← true;
14      senão "inclusão inválida";
```

Análise da complexidade:

Árvores Digitais: algoritmo de inserção

```
1  insereDigital(x, pt):
2      pt ← ptraiz; l ← a ← 0;
3      buscaDigital(x, pt, l, a);
4      se a = 0 então
5          para h = l + 1, ..., k faça
6              seja j a posição de
                   d(h) no alfabeto;
7              ocupar(ptz);
8              para i = 1, ..., m faça
9                  ptz.pont[i] ← null;
10             pt.pont[j] ← ptz;
11             ptz.terminal ← false;
12             pt ← ptz;
13         pt.terminal ← true;
14     senão "inclusão inválida";
```

Análise da complexidade:

- Seja $k_1 + k_2 = k$;

Árvores Digitais: algoritmo de inserção

```
1  insereDigital(x, pt):
2      pt ← ptraiz; l ← a ← 0;
3      buscaDigital(x, pt, l, a);
4      se a = 0 então
5          para h = l + 1, ..., k faça
6              seja j a posição de
                   d(h) no alfabeto;
7              ocupar(ptz);
8              para i = 1, ..., m faça
9                  ptz.pont[i] ← null;
10             pt.pont[j] ← ptz;
11             ptz.terminal ← false;
12             pt ← ptz;
13         pt.terminal ← true;
14     senão "inclusão inválida";
```

Análise da complexidade:

- Seja $k_1 + k_2 = k$;
- **Linha** 3: gasta $O(k_1 \cdot \log m)$;

Árvores Digitais: algoritmo de inserção

```
 1  insereDigital(x, pt):
 2      pt ← ptraiz; l ← a ← 0;
 3      buscaDigital(x, pt, l, a);
 4      se a = 0 então
 5          para h = l + 1, ..., k faça
 6              seja j a posição de
                    d(h) no alfabeto;
 7              ocupar(ptz);
 8              para i = 1, ..., m faça
 9                  ptz.pont[i] ← null;
10              pt.pont[j] ← ptz;
11              ptz.terminal ← false;
12              pt ← ptz;
13          pt.terminal ← true;
14      senão "inclusão inválida";
```

Análise da complexidade:

- Seja $k_1 + k_2 = k$;
- **Linha** 3: gasta $O(k_1 \cdot \log m)$;
- **Linha** 8: executa m vezes. Portanto, inserir gasta $O(k_2 \cdot m)$;

Árvores Digitais: algoritmo de inserção

```
 1  insereDigital(x, pt):
 2      pt ← ptraiz; l ← a ← 0;
 3      buscaDigital(x, pt, l, a);
 4      se a = 0 então
 5          para h = l + 1, ..., k faça
 6              seja j a posição de
                    d(h) no alfabeto;
 7              ocupar(ptz);
 8              para i = 1, ..., m faça
 9                  ptz.pont[i] ← null;
10              pt.pont[j] ← ptz;
11              ptz.terminal ← false;
12              pt ← ptz;
13          pt.terminal ← true;
14      senão "inclusão inválida";
```

Análise da complexidade:

- Seja $k_1 + k_2 = k$;
- **Linha** 3: gasta $O(k_1 \cdot \log m)$;
- **Linha** 8: executa m vezes. Portanto, inserir gasta $O(k_2 \cdot m)$;
- **Complexidade total**: $O(k_1 \log m + k_2 m)$;

x é chave a inserir. **Chamada inicial:** $insercaoDigital(x, ptraiz)$

Árvores Digitais: complexidade da busca

Árvores Digitais: complexidade da busca

- Diferente de métodos clássicos de busca pois:

Árvores Digitais: complexidade da busca

- Diferente de métodos clássicos de busca pois:
  - independe do número total de chaves (e de tamanho arquivo);

Árvores Digitais: complexidade da busca

- Diferente de métodos clássicos de busca pois:
  - independe do número total de chaves (e de tamanho arquivo);
  - depende do tamanho chave procurada e do alfabeto;

Árvores Digitais: complexidade da busca

- Diferente de métodos clássicos de busca pois:
  - independe do número total de chaves (e de tamanho arquivo);
  - depende do tamanho chave procurada e do alfabeto;
- **Alternativa implementação**: lista circular para ponteiros para evitar inúmeros ponteiros nulos (otimiza memória);

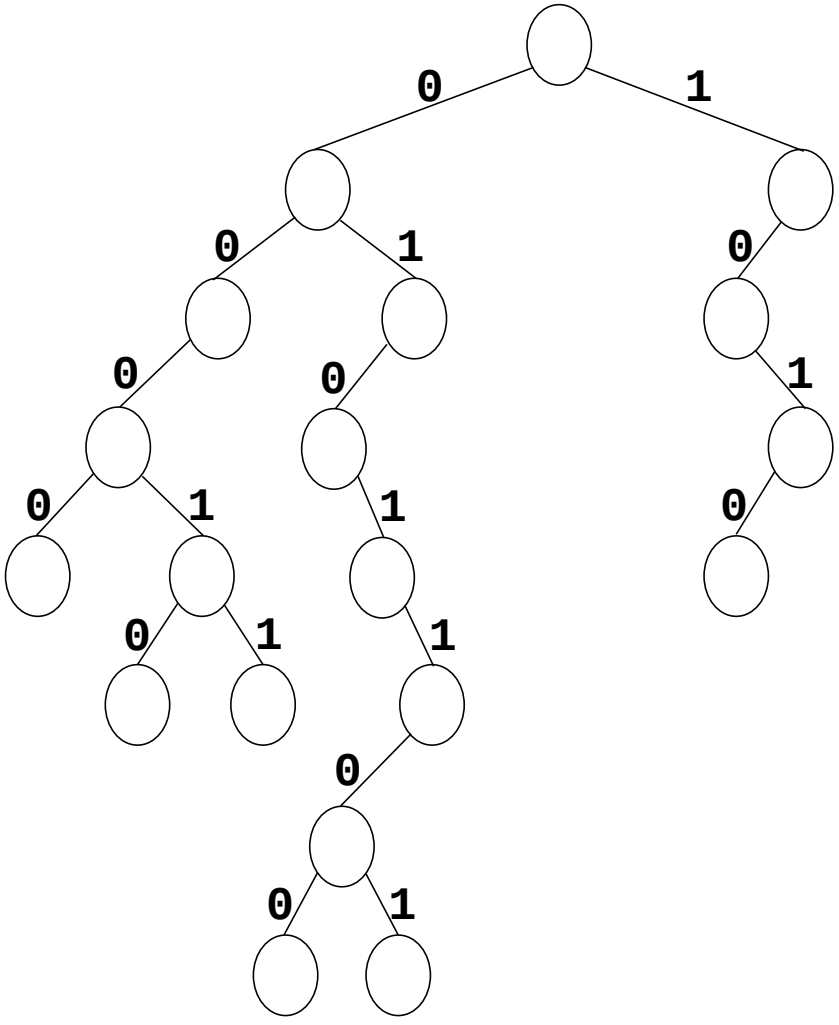
Árvores Digitais: complexidade da busca

- Diferente de métodos clássicos de busca pois:
  - independe do número total de chaves (e de tamanho arquivo);
  - depende do tamanho chave procurada e do alfabeto;
- **Alternativa implementação**: lista circular para ponteiros para evitar inúmeros ponteiros nulos (otimiza memória);
- *Trie* é tão mais eficiente quanto maior quantidade de chaves com prefixos comuns;

Árvores Digitais: complexidade da busca

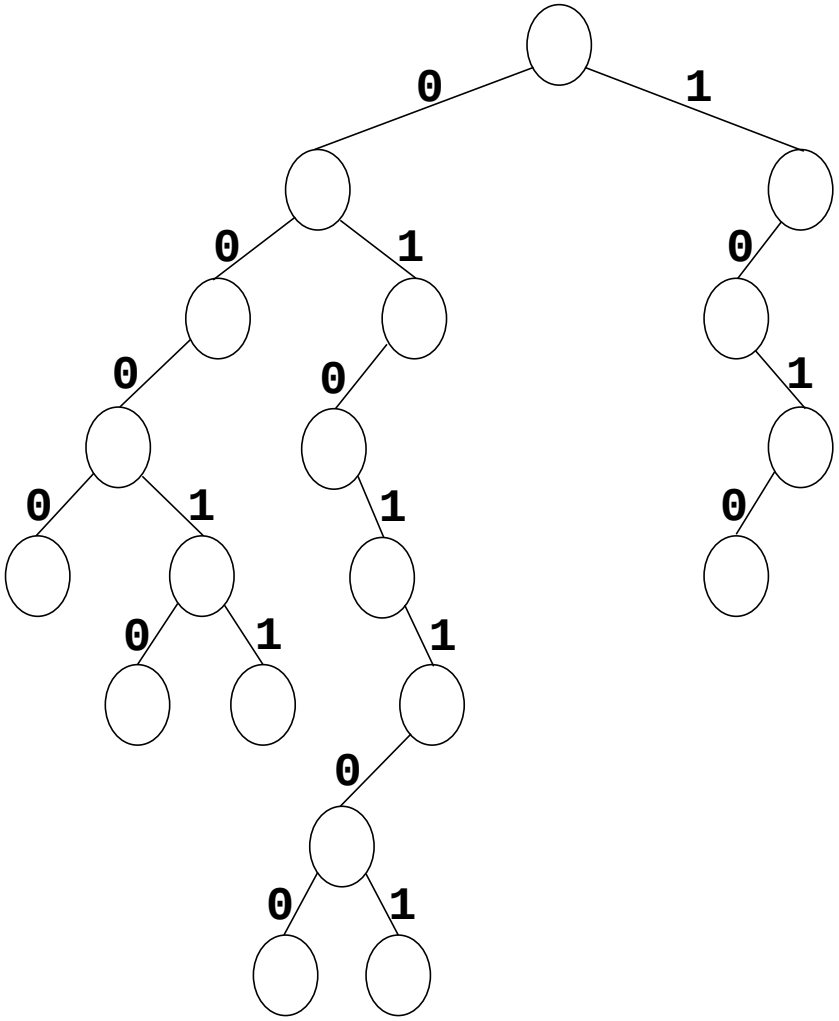
- Diferente de métodos clássicos de busca pois:
  - independe do número total de chaves (e de tamanho arquivo);
  - depende do tamanho chave procurada e do alfabeto;
- **Alternativa implementação**: lista circular para ponteiros para evitar inúmeros ponteiros nulos (otimiza memória);
- *Trie* é tão mais eficiente quanto maior quantidade de chaves com prefixos comuns;
- *Trie* com muitos ziguezagues é quase sempre ineficiente;

Árvores Digitais Binárias



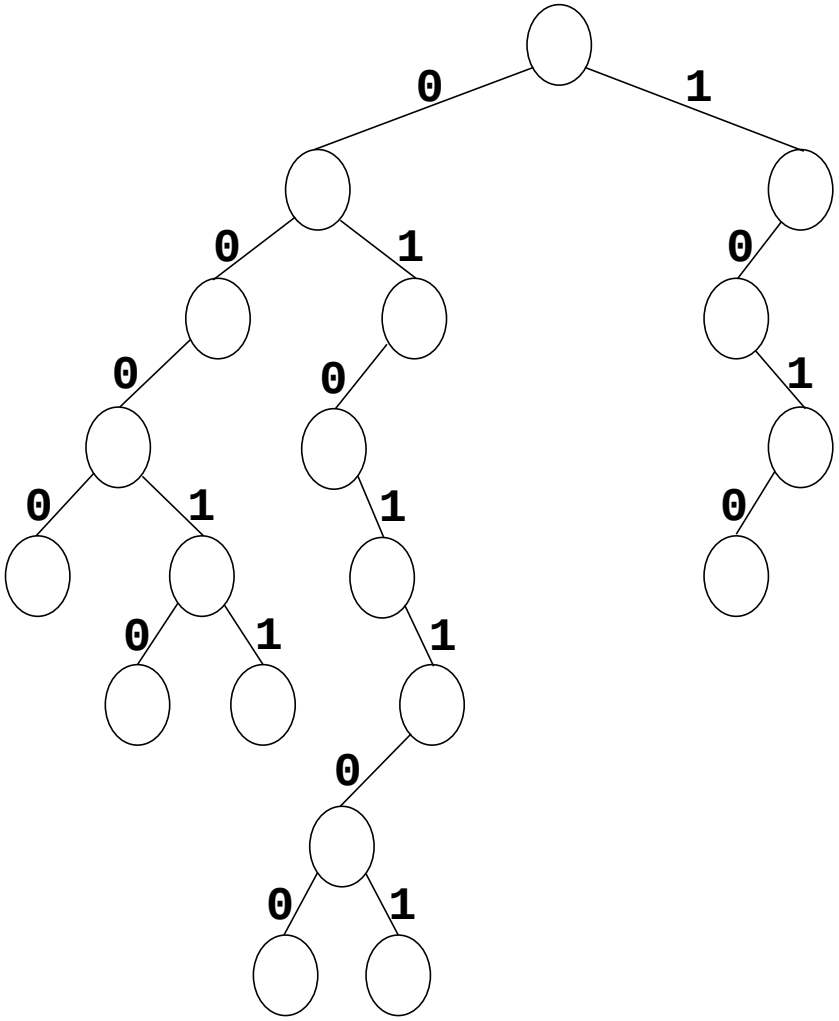
- Árvore digital binária com alfabeto $\{0, 1\}$;

Árvores Digitais Binárias



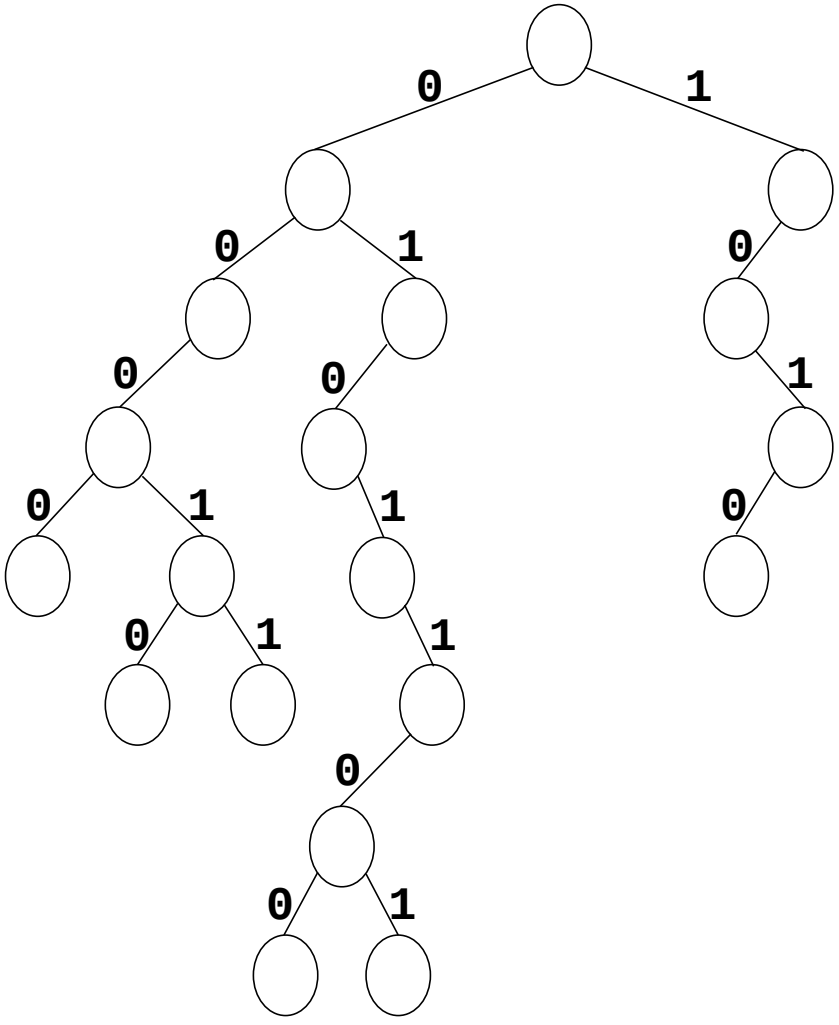
- Árvore digital binária com alfabeto $\{0, 1\}$;
- Chaves são sequência binária;

Árvores Digitais Binárias



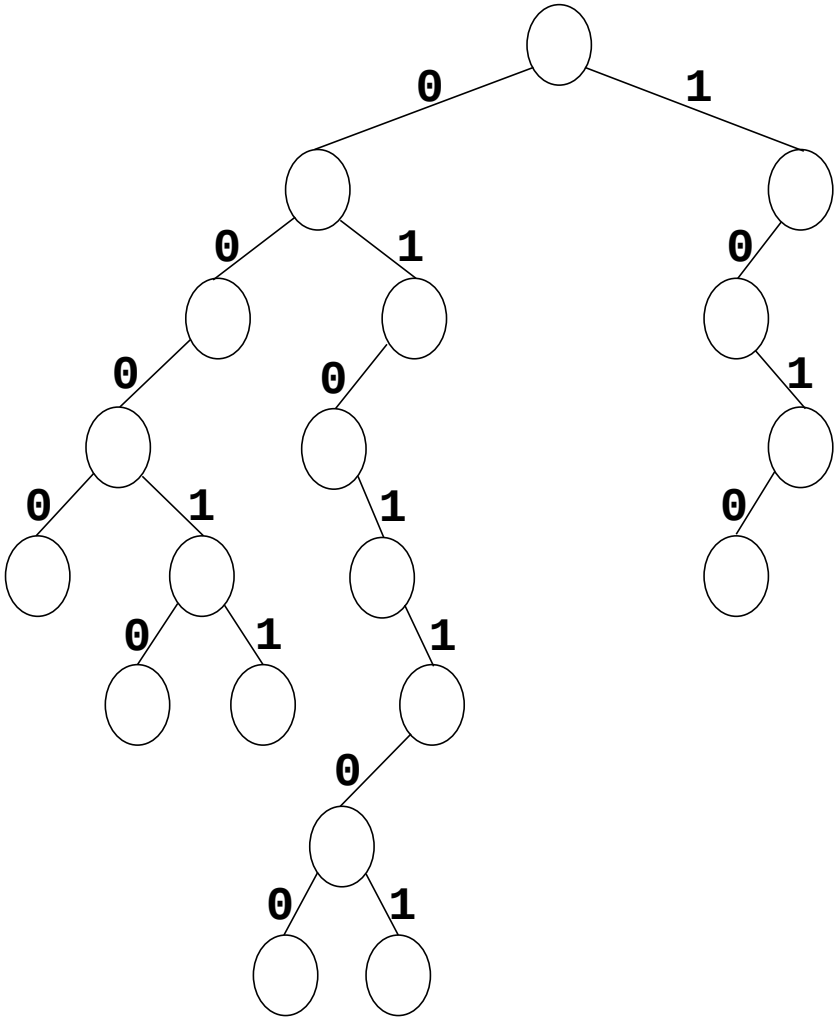
- Árvore digital binária com alfabeto $\{0, 1\}$;
- Chaves são sequência binária;
- Filho esquerdo: 0;

Árvores Digitais Binárias



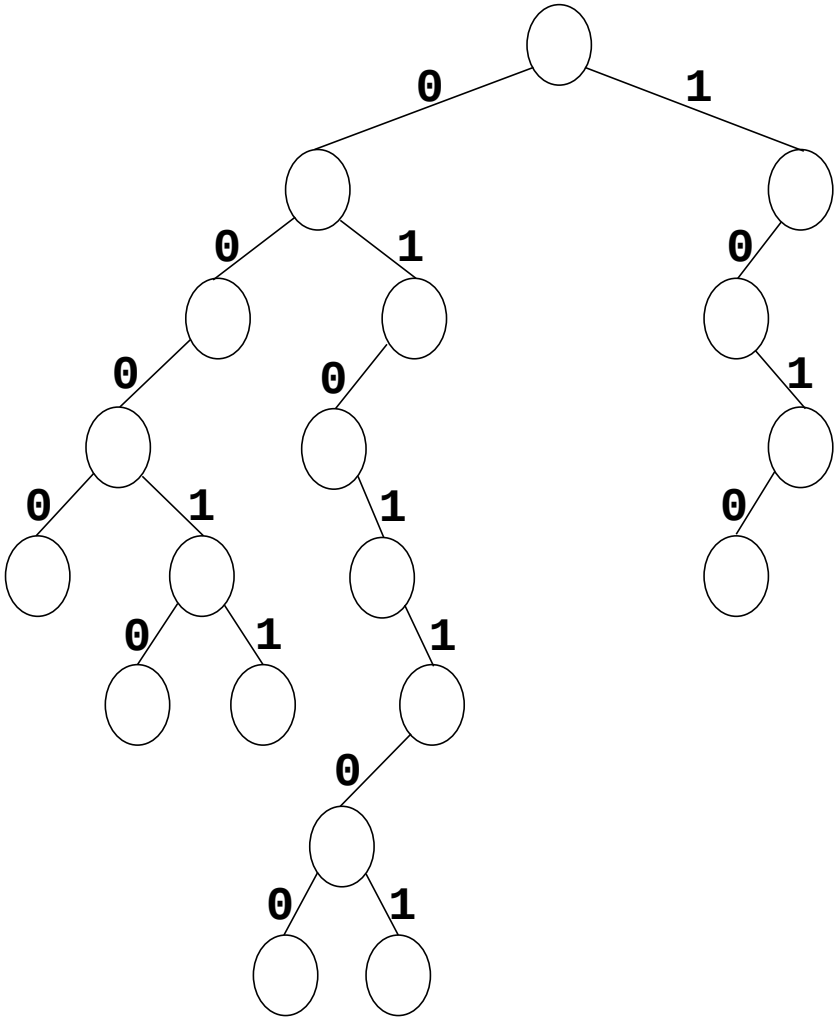
- Árvore digital binária com alfabeto $\{0, 1\}$;
- Chaves são sequência binária;
- Filho esquerdo: 0;
- Filho direito: 1;

Árvores Digitais Binárias



- Árvore digital binária com alfabeto $\{0, 1\}$;
- Chaves são sequência binária;
- Filho esquerdo: 0;
- Filho direito: 1;
- $S = \{00, 0000, 00010, 00011, 0101100, 0101101, 10, 101, 1010\}$

Árvores Digitais Binárias



- Árvore digital binária com alfabeto $\{0, 1\}$;
- Chaves são sequência binária;
- Filho esquerdo: 0;
- Filho direito: 1;
- $S = \{00, 0000, 00010, 00011, 0101100, 0101101, 10, 101, 1010\}$
- Maior utilização de árvores digitais é caso binário;

Árvores Digitais Binárias

Árvores Digitais Binárias

- Chaves/códigos binários são mais empregados em computação;

Árvores Digitais Binárias

- Chaves/códigos binários são mais empregados em computação;
- Número de ponteiros vazios é menor;

Árvores Digitais Binárias

- Chaves/códigos binários são mais empregados em computação;
- Número de ponteiros vazios é menor;
- **Árvore binária de prefixo**:

Árvores Digitais Binárias

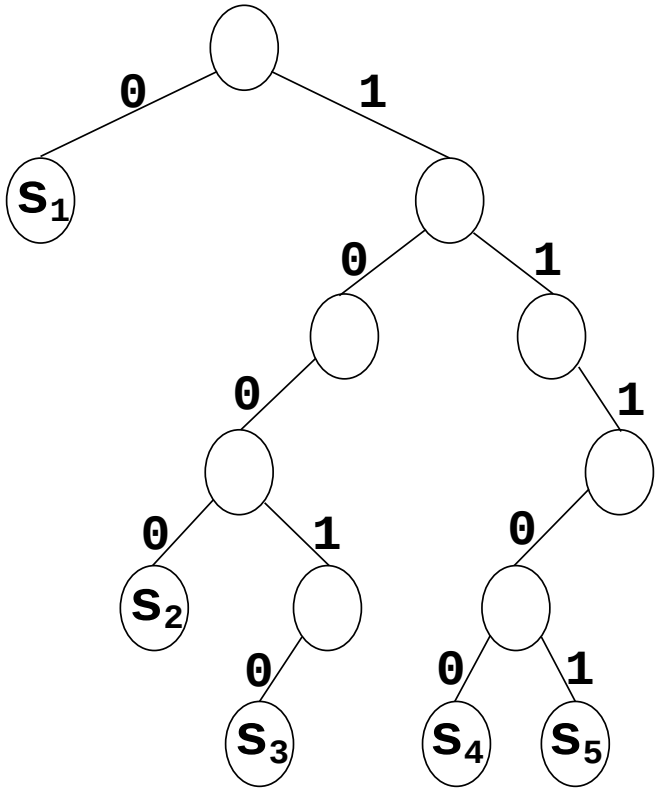
- Chaves/códigos binários são mais empregados em computação;
- Número de ponteiros vazios é menor;
- **Árvore binária de prefixo**:
  1. nenhum código é prefixo de outro;

Árvores Digitais Binárias

- Chaves/códigos binários são mais empregados em computação;
- Número de ponteiros vazios é menor;
- **Árvore binária de prefixo**:
  1. nenhum código é prefixo de outro;
  2. chaves representadas por folhas;

Árvores Digitais Binárias

- Chaves/códigos binários são mais empregados em computação;
- Número de ponteiros vazios é menor;
- **Árvore binária de prefixo**:
  1. nenhum código é prefixo de outro;
  2. chaves representadas por folhas;



Árvores Patricia: Introdução

- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;

Árvores Patricia: Introdução

- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;
- **Criador:** Donald Morrison, em 1968;

Árvores Patricia: Introdução

- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;
- **Criador:** Donald Morrison, em 1968;
- **Árvore Patricia:** árvore digital binária de prefixos;

Árvores Patricia: Introdução

- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;
- **Criador:** Donald Morrison, em 1968;
- **Árvore Patricia:** árvore digital binária de prefixos;
- Estritamente binária;

Árvores Patricia: Introdução

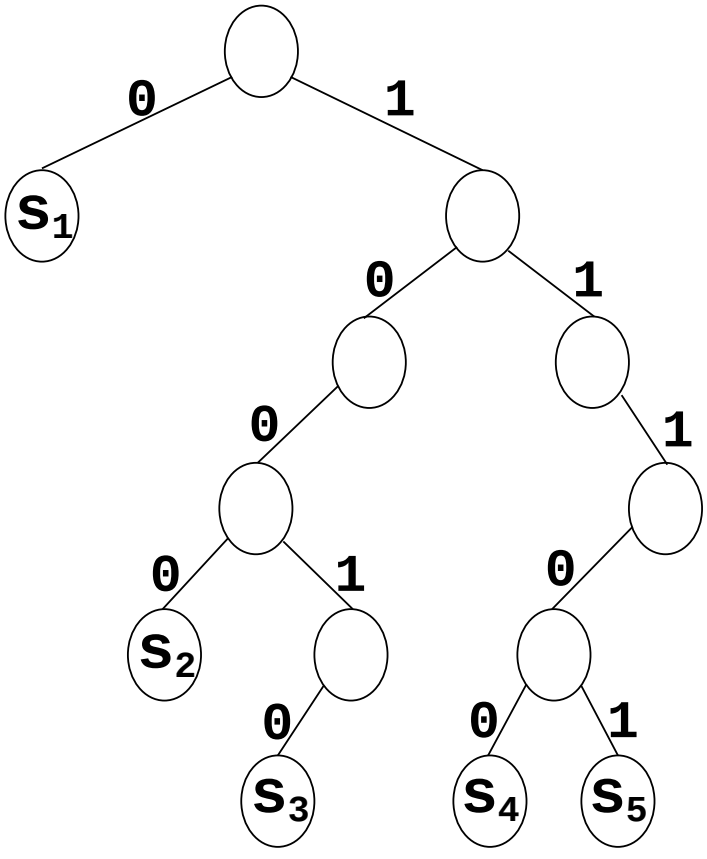
- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;
- **Criador:** Donald Morrison, em 1968;
- **Árvore Patricia:** árvore digital binária de prefixos;
- Estritamente binária;
- Sequência de nós com apenas um filho são compactados em um único nó;

Árvores Patricia: Introdução

- **PATRICIA:** acrônimo de ***P****ractical* ***A****lgorithm* ***T****o* ***R****etrieve* ***I****nformation* ***C****oded* ***I****n* ***A****lphanumeric*;
- **Criador:** Donald Morrison, em 1968;
- **Árvore Patricia:** árvore digital binária de prefixos;
- Estritamente binária;
- Sequência de nós com apenas um filho são compactados em um único nó;
- Nenhuma chave é prefixa de outra chave.

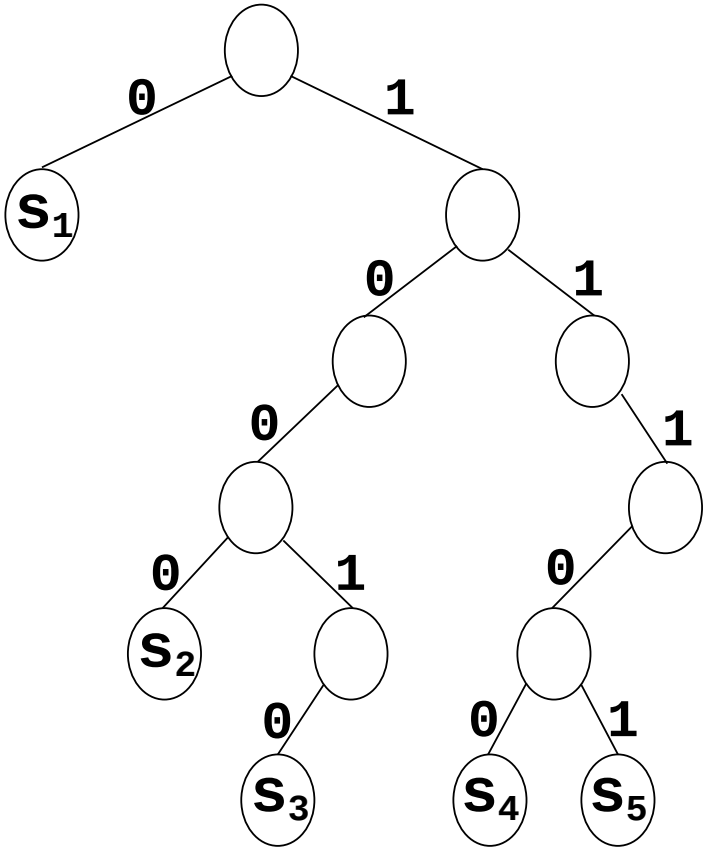
Exemplo de Árvore Patricia

Exemplo de Árvore Patricia

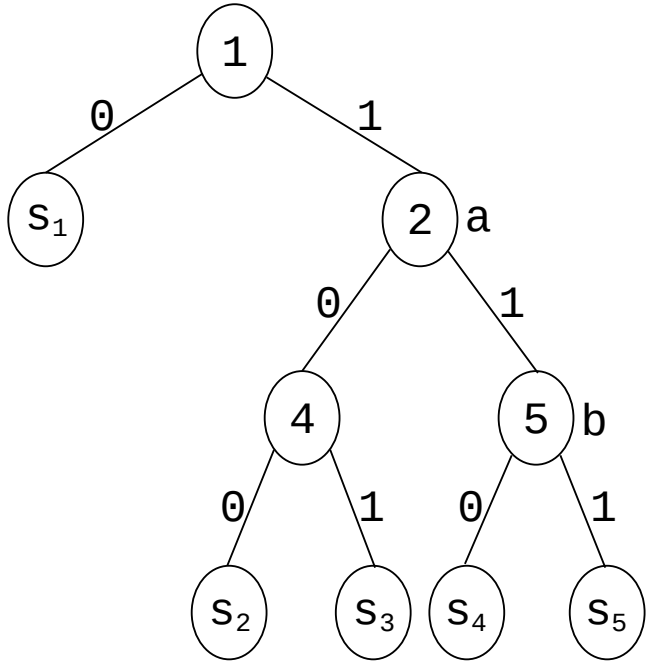


(c) Árvore Dig. Bin. Prefixo

Exemplo de Árvore Patricia



(e) Árvore Dig. Bin. Prefixo



(f) Árvore Patricia

Exercícios

1. Escreva o procedimento de inserção de uma chave em árvores *Trie*.

Exercícios

1. Escreva o procedimento de inserção de uma chave em árvores *Trie*.
2. Escreva o procedimento de remoção de uma chave em árvores *Trie*.

Bibliografia

SZWARCFITER, J. L. e MARKENZON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.

Perguntas?