CI1057: Algoritmos e Estruturas de Dados III
Árvores Patricia

Profa. Carmem Hara

Departamento de Informática/UFPR

24 de abril de 2024

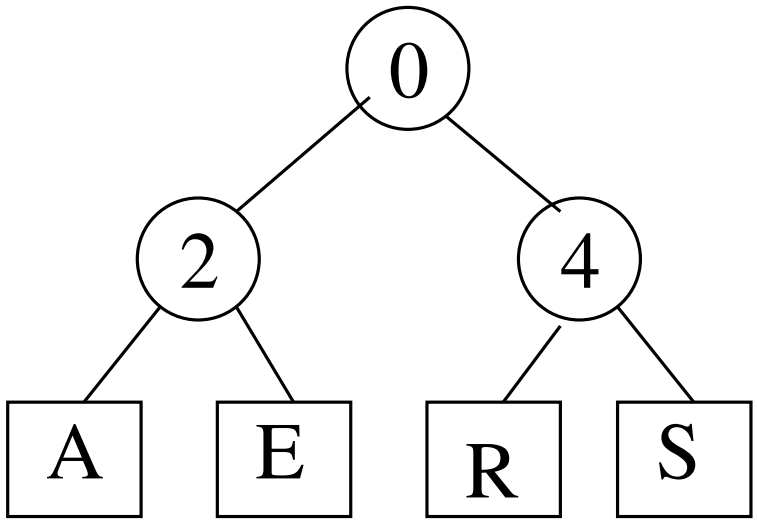
Árvore Patricia

- *Practical Algorithm To Retrieve Information Coded in Alphanumeric*
- Não tem as desvantagens da Trie:
  - quantidade de nós corresponde à quantidade de chaves
  - não há caminhos de uma única direção
- Criada por Morrison (1968) para recuperação de informação em textos
- Estendido por Knuth (1973), Sedgewick (1988), Gonnet e Baeza-Yates (1991)

Árvore Patricia Simplificada

- 2 tipos de nodos (folhas e não-folhas)

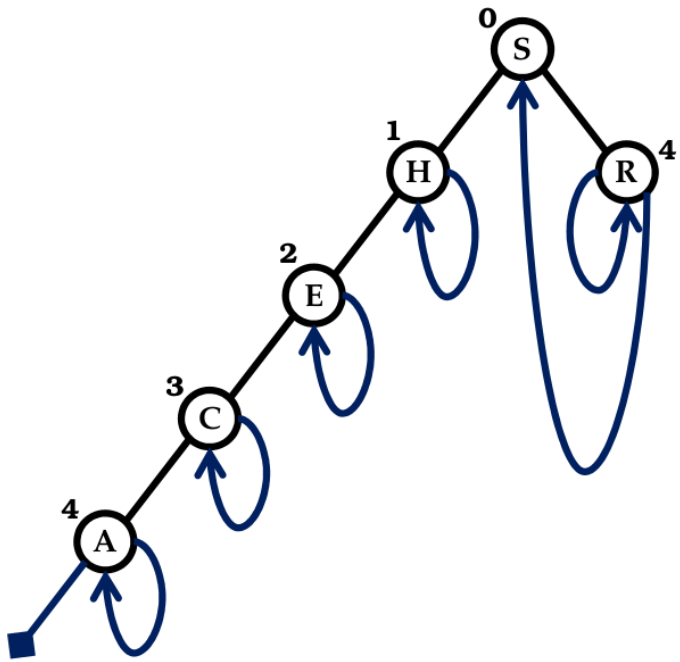
```
A 00001
C 00011
E 00101
H 01000
R 10010
S 10011
```



Árvore Patricia

- Um único tipo de nodo
- Cada nodo contém uma chave e um índice (número do bit) que deve ser testado

```
A 00001
C 00011
E 00101
H 01000
R 10010
S 10011
```



Árvore Patricia

Uma árvore Patricia é uma árvore binária na qual cada nodo n possui uma chave e um índice de bit k, cujo valor é definido da seguinte forma:

- k é o primeiro bit no qual a chave difere da chave do seu pai;
- seja kp o índice bit do pai. Se o bit kp da chave é igual a 0 entao n é o filho esquerdo do pai; caso contrário, ele é o filho direito.

Um nodo externo pode corresponder a NULL ou ao endereço de um nodo a, que é igual a n ou um ancestral de n.
Sejam $n_1, \ldots, n_q$ os nodos da raiz até a com indices bit $b_1, \ldots, b_q$, respectivamente. A chave de a é a unica da árvore que satisfaz a seguinte condição. Para todo i em $[2, q]$:

- se n_i é filho esquerdo de n_{i-1} então o bit b_{i-1} da chave é igual a 0;
- se n_i é filho direito de n_{i-1} então o bit b_{i-1} da chave é igual a 1.

Árvore de Pesquisa Digital Binária - Interface

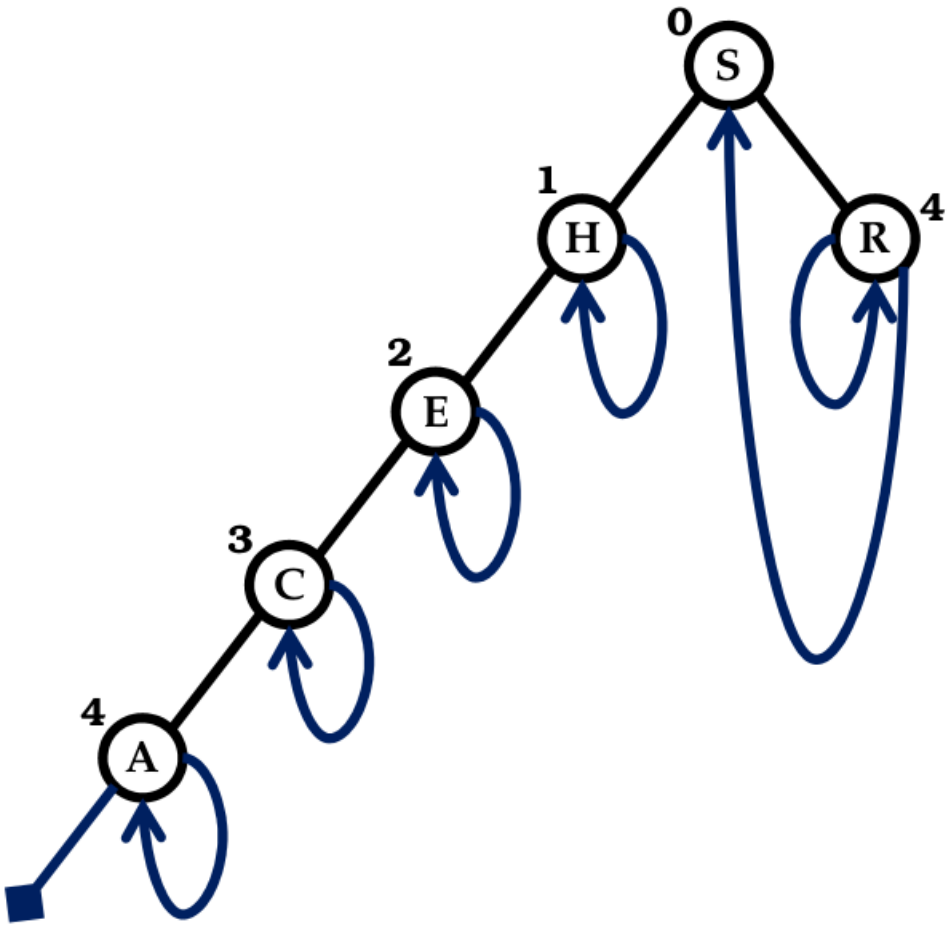
```
1  #define ITEMNULO 0
2
3  typedef long TipoItem;
4  typedef struct Nodo *ApNodo;
5  typedef struct Nodo {
6    TipoItem item;
7    ApNodo esq, dir;
8    int bit;
9  } Nodo;
10
11 ApNodo criaArv();
12 void freeArv( ApNodo raiz );
13 ApNodo busca( TipoItem v, ApNodo raiz);
14 ApNodo insere( TipoItem v, ApNodo raiz );
15 ApNodo remove( TipoItem v, ApNodo raiz );
```

Árvore Patricia - `criaArv()`

- para simplificar a implementação, a raiz da arvore é sempre um nodo *r* com "chave nula" (`ITEMNULO`) (todos os bits iguais a 0, um valor nao utilizado como valor de chave) e campo bit igual -1;
- caso a árvore contenha pelo menos uma chave, `r->esq` aponta para um nodo que contém uma chave.

```
1 ApNodo criaArv(){
2   ApNodo raiz;
3
4   raiz= criaNodo( ITEMNULO, -1 );
5   raiz->esq = raiz;
6   raiz->dir = raiz;
7   return raiz;
8 }
```

Árvore Patricia - Busca

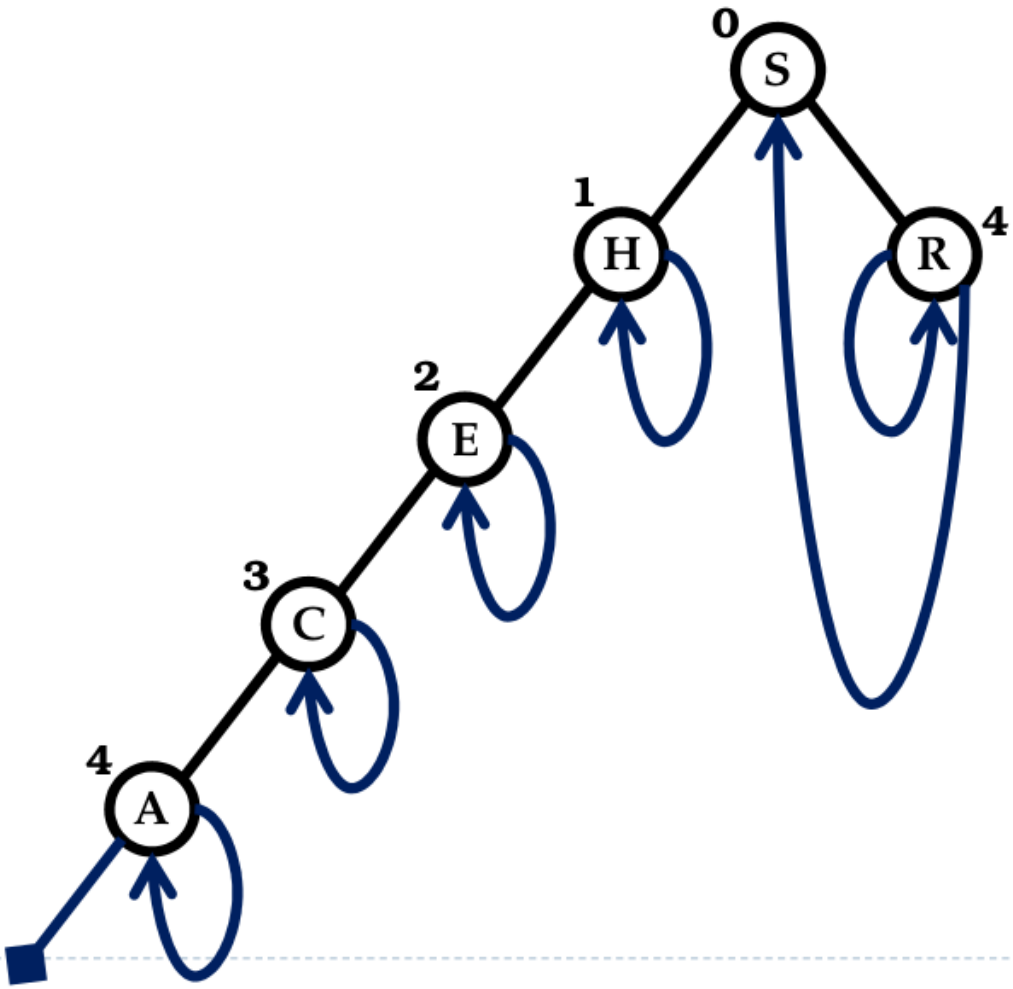


Busca por
R = 10010
I = 01001

Árvore Patricia - Busca

```
1  ApNodo busca( TipoItem k, ApNodo raiz ){
2    ApNodo p;
3
4    p = buscaR( k, raiz->esq, -1 );
5    if( p->item == k ) return p;
6    else retorn NULL;
7  }
8
9  ApNodo buscaR( TipoItem k, ApNodo p, int bit ){
10   if( p->bit <= bit )
11     return p;
12   if( digito( k, p->bit) == 0 )
13     return buscaR( k, p->esq, p->bit );
14   else
15     return buscaR( k, p->dir, p->bit );
16 }
```

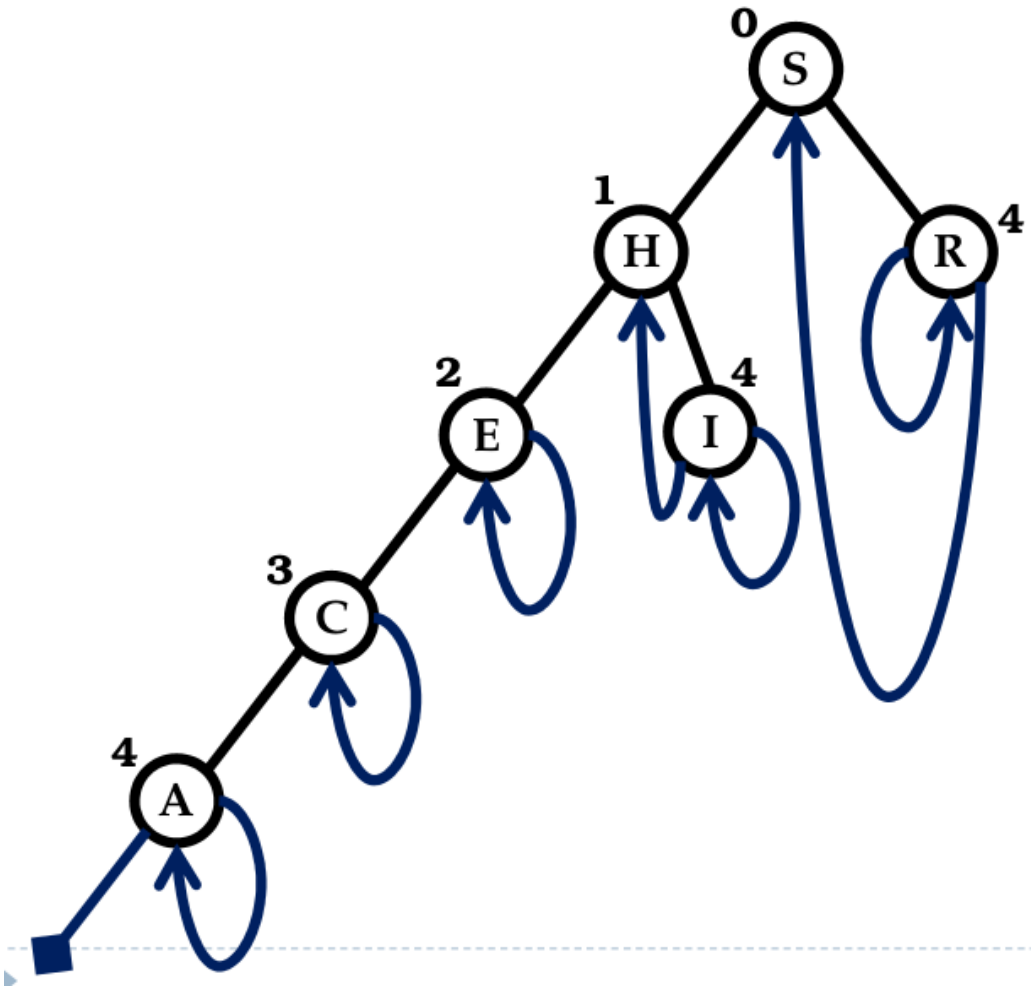
Árvore Patricia - Inserção



Inserindo
I = 01001

H = 01000

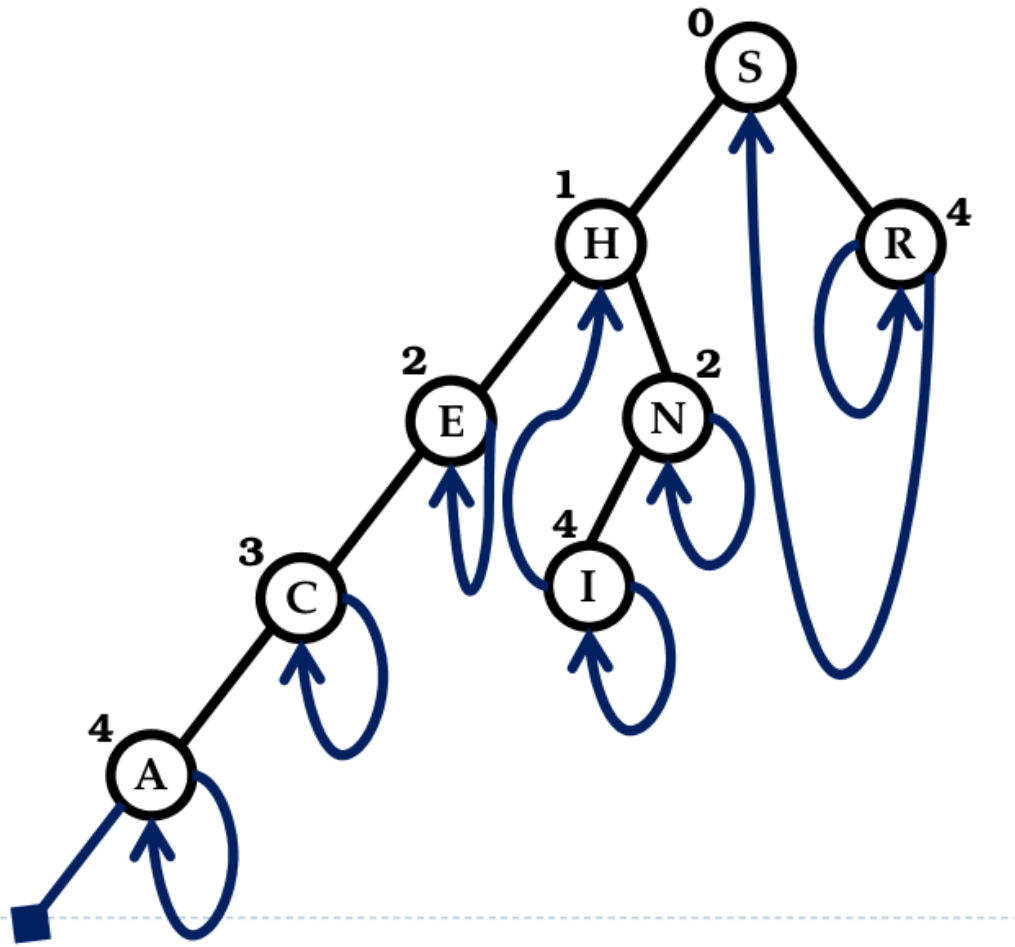
Árvore Patricia - Inserção - Caso 1



Inserindo
N = 01110

I = 01001

Árvore Patricia - Inserção - Caso 2



Árvore Patricia - Inserção

2 Casos:

- Caso 1: O novo nó substitui um nó externo
  - Acontece quando o bit que diferencia nova chave da chave encontrada não foi utilizada na busca
- Caso 2: O novo nó substitui um nó interno
  - Acontece quando o bit que diferencia a nova chave da chave encontrada foi pulado durante a busca

Árvore Patricia - Inserção

Exemplo: inserção das chaves A, S, E, R, C, H, I, N

```
A 00001
S 10011
E 00101
R 10010
C 00011
H 01000
I 01001
N 01110
```

Árvore Patricia - Inserção

```
1  void insere( TipoItem v, ApNodo raiz ){
2    ApNodo p;
3
4    p= buscaR( v, raiz->esq, -1 );
5    if( v == p->item ) return;    /* chave ja' esta' na arvore */
6    i = 0;                        /* procura bit que diferencia v da chave de p */
7    while( digito(v, i) == digito(p->item, i) ) i++;
8    raiz->esq = insereR (v, raiz->esq, i, raiz);
9    return;
10 }
11
12 ApNodo insereR( TipoItem v, ApNodo p, int bit, ApNodo paiP ){
13   ApNodo n;
14
15   if( p->bit >= bit || p->bit <= paiP->bit ){
16     n = criaNodo( v, bit );
17     if( digito(v, bit) == 0 )
18          { n->esq = n;  n->dir = p;}
19     else { n->esq = p;  n->dir = n; }
20     return n;
21   }
22   if( digito(v, bit) == 0 )
23     p->esq = insereR( v, p->esq, p->bit, p );
24   else
25     p->dir = insereR( v, p->dir, p->bit, p );
26   return p;
27 }
```

Árvore Patricia - Características

- Inserção de N chaves aleatórias requer aproximadamente $lg(N)$ comparações de bits no caso médio e $2lg(N)$ comparações no pior caso
- O número de comparações de bits é limitado pelo tamanho das chaves
- todos as folhas abaixo de um determinado nodo n com bit de indice k tem como prefixo os mesmos k bits
- para obter as chaves ordenadas, basta imprimir as chaves das folhas, percorrendo a árvore em-ordem

Árvore Patricia - Sort

```
1  void sort( ApNodo raiz ){
2    sortR( raiz->esq, -1 );
3  }
4
5  void sortR( ApNodo p, int bit ){
6    if( p == NULL ) return;
7    if( p->bit <= bit ) escreveItem( p->item );
8    sortR( p->esq, p->bit );
9    sortR( p->dir, p->bit );
10 }
```

Referências

- Livro Sedgewick Cap. 15 (Seção 15.3)
- Livro Nivio Ziviani – Patricia Simplificada