Pesquisa Digital

- A pesquisa digital está baseada na representação das chaves como uma seqüência de caracteres de um alfabeto
- 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

Algoritmos e Estruturas de Dados II

José Augusto Baranauskas
Departamento de Física e Matemática – FFCLRP-USP
augusto@ffclrp.usp.br
http://dfm.ffclrp.usp.br/~augusto

Árvorie

- Árvorie (*trie*) é uma estrutura de índice útil quando as chaves são de tamanho variável
- Uma árvorie é uma árvore de busca em **m**-vias, **m**≥2, na qual a ramificação em qualquer nível é determinada por apenas uma parte do valor de chave e não pela chave inteira

Árvorie

- O termo **trie** surgiu nos anos 60 e tem origem na palavra *re**trie**val*, uma vez que essa estrutura é usada basicamente na recuperação de dados
- Na estrutura de dados **trie** as chaves são representadas caractere por caractere
- **Tries** são usadas para fazer uma rápida busca em um texto grande
- Cada chave é formada por uma combinação específica de símbolos do alfabeto
- As combinações são de comprimento variável e ilimitado; portanto, as chaves podem ser palavras, seqüências binárias, códigos numéricos, etc

Árvorie

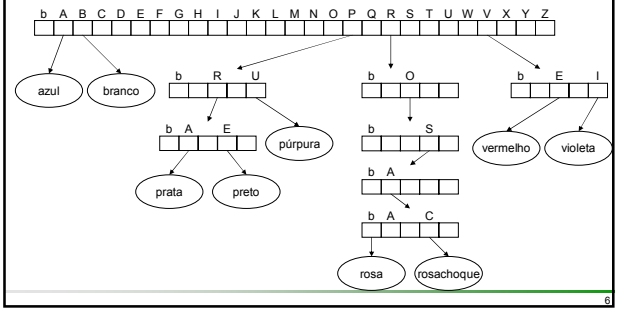
- Em uma árvorie, as chaves são armazenadas e manipuladas de uma forma especial, pois são parcialmente compartilhadas entre os elementos
- Ao invés de se comparar chaves inteiras entre si, as comparações são feitas componente a componente
- Adicionalmente, as chaves são decompostas e as partes comuns entre elas são fundidas

Árvorie

- As árvories são boas para suportar tarefas de tratamento lexicográfico, tais como:
  - manuseamento de dicionários
  - pesquisas em textos de grande dimensão
  - construção de índices de documentos
  - expressões regulares (padrões de pesquisa)

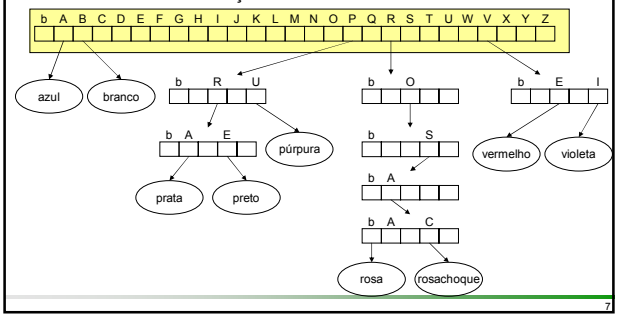
Árvorie: Estrutura

- Considere a árvore seguinte que contém dois tipos de nós: nó de desvio e nó de informação



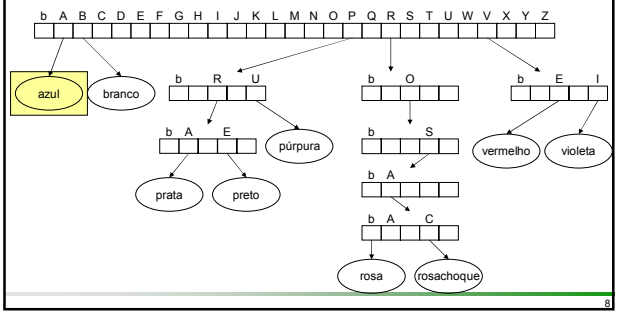
Árvorie: Estrutura

- Considere a árvore seguinte que contém dois tipos de nós: **nó de desvio** e nó de informação



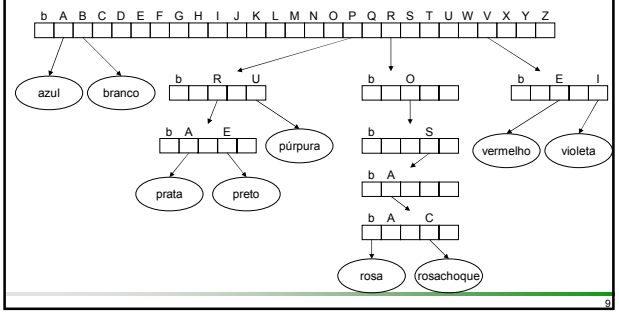
Árvorie: Estrutura

- Considere a árvore seguinte que contém dois tipos de nós: nó de desvio e **nó de informação**



Árvorie: Estrutura

- Cada nó de desvio tem m=27 campos de ligação (26 letras do alfabeto mais um branco que é usado para determinar uma chave)

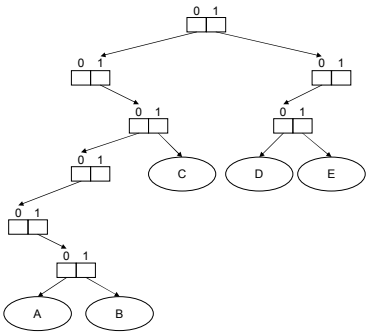


Árvorie: Estrutura

- O alfabeto pode ser qualquer conjunto, por exemplo considerando as chaves como seqüências de bits (**m**=2), o algoritmo de pesquisa digital é semelhante ao de pesquisa em árvore, exceto que, em vez de se caminhar na árvore de acordo com o resultado de comparação entre chaves, caminha-se de acordo com os bits da chave

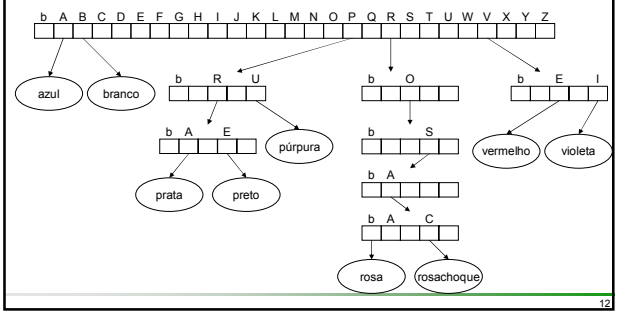
Árvorie: Estrutura

- A = 010010
- B = 010011
- C = 011000
- D = 100001
- E = 101000



Árvorie

- Retornando ao exemplo em que m=27...



Árvorie

- No nível 1, todas as chaves são particionadas em 27 classes disjuntas, de acordo com o primeiro caractere delas
- No **i**-ésimo nível, o desvio para o nó correspondente é determinado pelo **i**-ésimo caractere da chave
- Quando uma subárvore tem apenas uma chave, ela é substituída por um nó de informação
  - Neste nó inclui-se a chave juntamente com outras informações importantes, como o endereço de registro com esta chave etc
- Na representação gráfica, os nós de desvio são representados por retângulos, enquanto que para os de informação usa-se elipses

Algoritmo de Busca

- Para pesquisar em uma árvorie pela chave X, é necessário desdobrar X em caracteres (ou dígitos, dependendo do alfabeto utilizado) que a compõem e seguir a seqüência de desvios determinada por esses caracteres
- O algoritmo **TrieSearch** assume que **p**=0, não é um nó de desvio e que **p.key** é o valor da chave representada dentro de **p**, se ele é um nó de informação
- O algoritmo pesquisa pela chave X em árvorie de raiz T, admitindo-se, inicialmente, que o desvio no **i**-ésimo nível é determinado pelo **i**-ésimo caractere da chave
- **link(T,i)** retorna o ponteiro para a subárvore contendo todas as chaves iniciadas com o **i**-ésimo caractere, onde T é a raiz de árvorie

Algoritmo de Busca

```
function TrieSearch(T,X)
 1  X ← X + " "; // concatene um branco no fim de X
 2  i ← 1; p ← T; q ← 0; // q é o pai de p
 3  while p é um nó de desvio do
 4    C ← i-ésimo caractere de X; q ← p;
 5    p ← link(p,C);
 6    i ← i + 1;
 7  endwhile
 8  if p = 0 or p.key ≠ X then
 9    return (q,i-1,false); // X não está na árvorie
10  endif
11  return (p,i,true);
end TrieSearch
```


Análise do Algoritmo TrieSearch

- O algoritmo de pesquisa para as árvores é muito progressivo e pode-se constatar prontamente que o tempo de pesquisa é no pior caso O(**h**), onde **h** é o número de níveis na árvore (incluindo ambos os tipos de nós de desvio e de informação)
- No caso de um índice, todos os nós vão residir em disco e por isso, para efetuar a busca precisarão ser feitos no máximo **h** acessos
- Tendo um conjunto de valores de chave, para ser representado num índice, o número de níveis na árvore dependerá obviamente, da estratégia ou da técnica de amostragem de chave, para determinar a ramificação em cada nível
- Isso pode ser definido por uma função de amostragem **sample(X,i)**, que classifica X corretamente para ser desviado até o **i**-ésimo nível

Funções de Amostragem

- No algoritmo de busca, linha 4, esta função era
  - (a) sample(X,i) = i-ésimo caractere de X = x_i
- Outras escolhas para essa função são (X = $x_1x_2...x_n$)
  - (b) sample(X,i) = x_{n-i+1}
  - (c) sample(X,i) = $x_{i/2}$ se i for par
    $x_{n-(i-1)/2}$ se i for ímpar
- Para cada destas funções, pode-se encontrar os conjuntos de valores de chave para os quais a função será mais adequada, isto é, para os quais resultará numa árvorie com o número mínimo de níveis

Funções de Amostragem

- Assumindo X = $x_1x_2...x_{10}$
- (a) sample(X,i) = x_i
- (b) sample(X,i) = x_{n-i+1}
- (c) sample(X,i) = $x_{i/2}$ se i for par
  $x_{n-(i-1)/2}$ se i for ímpar

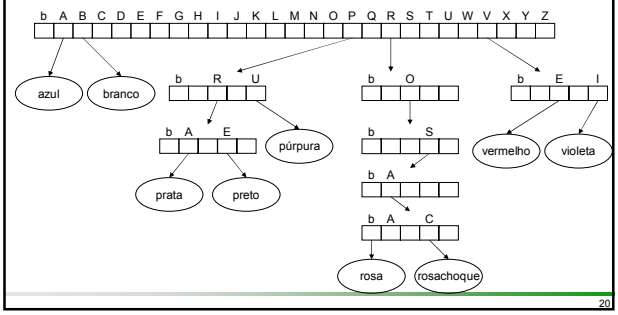
i	1	2	3	4	5	6	7	8	9	10
(a)	x_1	x_2	x_3	x_4	x_5	x_5	x_7	x_8	x_9	x_{10}
(b)	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1
(c)	x_{10}	x_1	x_9	x_2	x_8	x_3	x_7	x_4	x_6	x_5

Algoritmo de Busca

```
function TrieSearch(T,X)
 1  X ← X + " "; // concatene um branco no fim de X
 2  i ← 1; p ← T; q ← 0;  // q é o pai de p
 3  while p é um nó de desvio do
 4    C ← sample(X,i); q ← p;
 5    p ← link(p,C);
 6    i ← i + 1;
 7  endwhile
 8  if p = 0 or p.key ≠ X then
 9    return (q,i-1,false); // X não está na árvorie
10  endif
11  return (p,i,true);
end TrieSearch
```

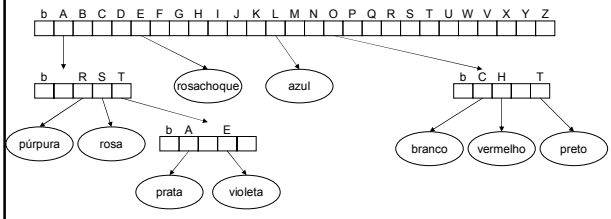

Funções de Amostragem

- (a) sample(X,i) = x_i



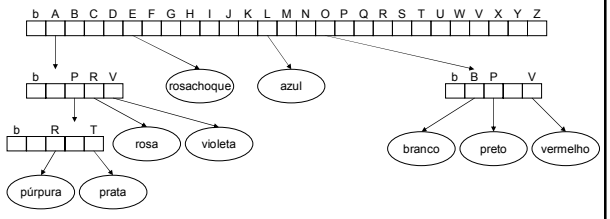
Funções de Amostragem

- (b) sample(X,i) = x_{n-i+1}



Funções de Amostragem

- (c) sample(X,i) = $x_{i/2}$ se i for par; $x_{n-(i-1)/2}$ se i for ímpar



Funções de Amostragem

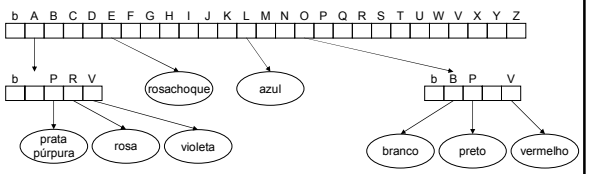
- A árvore usando a função de amostragem (a) tem 6 níveis
- Usando as funções (b) e (c), para as mesmas chaves, as árvores correspondentes têm apenas 4 níveis

Funções de Amostragem

- A quantidade máxima de níveis de uma árvore pode ser mantida baixa, adotando uma estratégia diferente para os nós de informação
- Esses nós podem ser projetados para manter mais do que uma chave
- Se o número máximo permitido de níveis for **h**, então todos os valores de chave que continuam sinônimos até o nível (**h**-1) entram no mesmo nó de informação
- Se a função de amostragem for escolhida corretamente, haverá apenas alguns sinônimos em cada nó de informação, que portanto, será suficientemente pequeno para ser processado na memória principal
- Nos próximos *slides* assumiremos que a função de amostragem usada é (a) e que não há restrições quanto à quantidade de níveis na árvore

Árvorie Limitada em Altura

- Exemplo de árvorie com máximo de 3 níveis permitidos

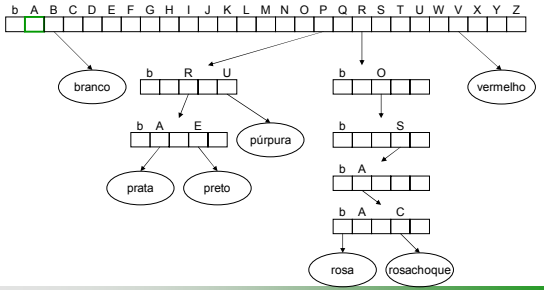


Inserção

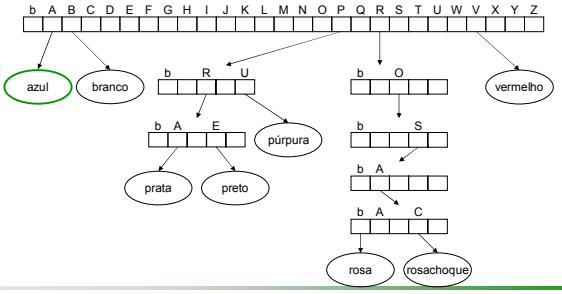
- A inserção na árvorie é progressiva: inicialmente, é feita uma busca pela chave a ser inserida
  - Se ela já existir na árvorie nada é feito
  - Caso contrário, é recuperado o nó até onde acontece a maior parte da chave a ser inserida, sendo o restante dos seus caracteres adicionados na árvorie a partir daquele nó, por meio de nós de desvios adicionados
- No algoritmo de inserção, assume-se que a árvorie vazia é criada com um único nó de desvio, sem nenhuma chave inserida e que um branco foi adicionado ao final da chave pelo algoritmo de busca

Inserção X = “azul”

- Como link(T,’A’)=0, X ainda não está em T e pode ser ali inserido

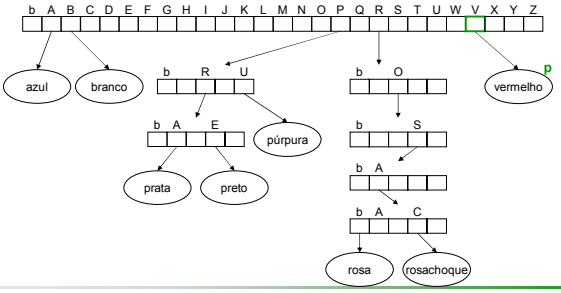


Inserção X = “azul”



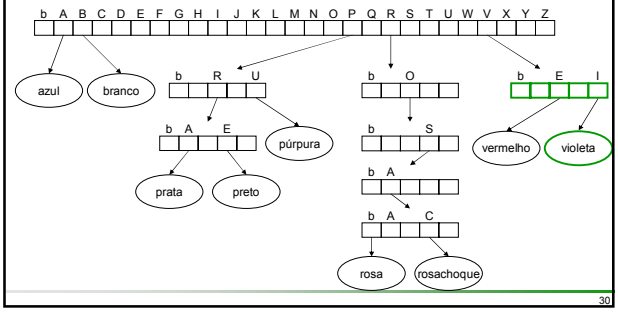
Inserção X = “violeta”

- Como link(T,’V’)≠0 e **p**.key≠X, onde **p**=link(T,’V’), X pode ser inserido em T, mas deve-se buscar um caractere que diferencie X de **p**.key



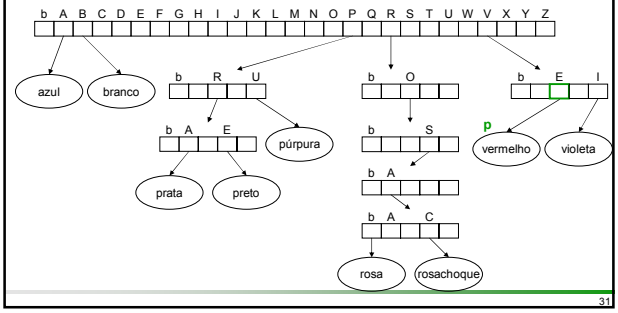
Inserção X = “violeta”

- Nesse caso, basta criar apenas um novo nó de desvio

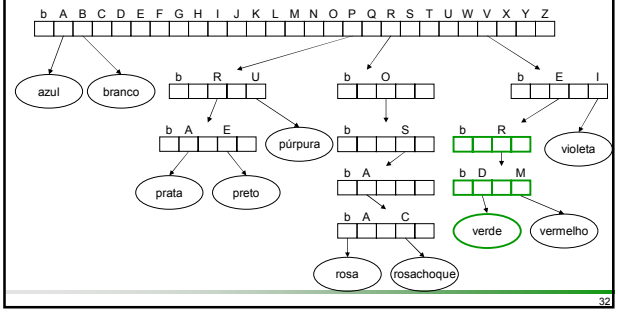


Inserção X = “verde”

- Novamente, deve-se buscar um caractere que diferencie X de **p**.key, criando-se nós de desvios adicionais para isso



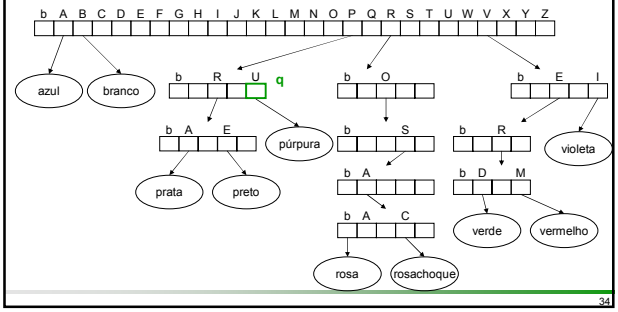
Inserção X = “verde”



Algoritmo de Inserção

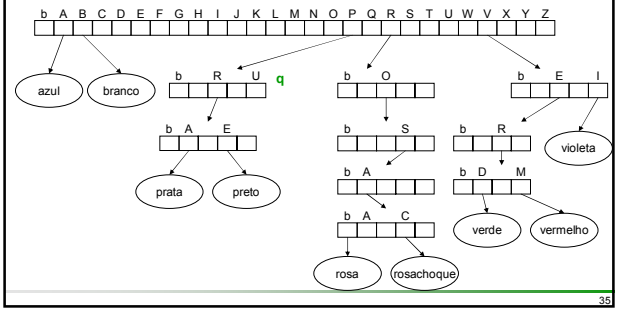
```
procedure TrieInsert(T,X)
 1 (q,i,achou) ← TrieSearch(T,X);
 2 if achou then return; endif // X está em T
 3 C ← sample(X,i); p ← link(q,C);
 4 if p = 0 then
 5   crie nó de informação F;
 6   F.key ← X; link(q,C) ← F; return;
 7 endif
 8 while sample(X,i) = sample(p.key,i) do
 9    crie nó de desvio R; link(q,C) ← R;
10    q ← R; i ← i + 1; C ← sample(X,i);
11 endwhile
12 crie nó informação F; F.key ← X; link(q,C) ← F;
13 C ← sample(p.key,i); link(q,C) ← p;
end TrieInsert
```


Remoção X = “púrpura”



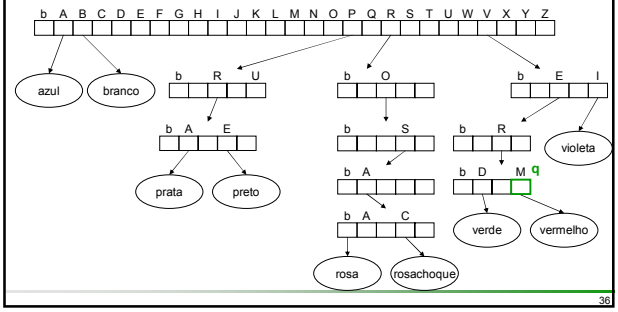
Remoção X = “púrpura”

- Neste caso, basta posicionar link(**q**,’U’)=0



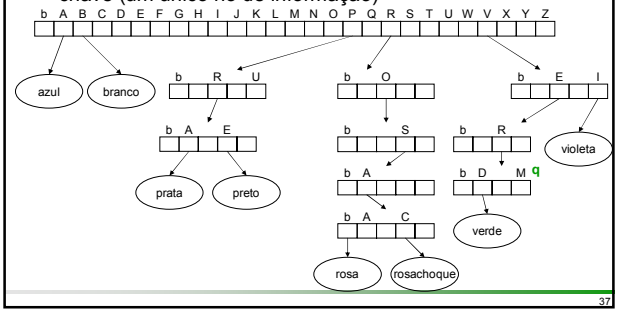
Remoção X = “vermelho”

- Ao posicionar link(**q**,’M’)=0...



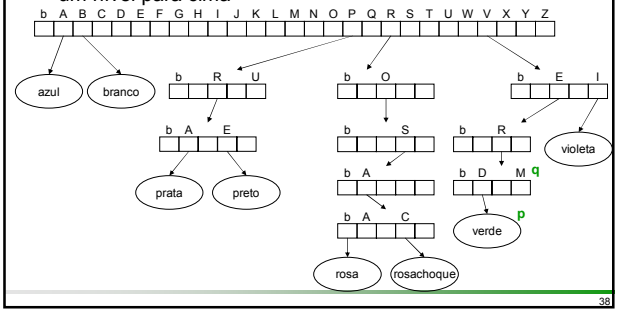
Remoção X = “vermelho”

- Ao posicionar link(**q**,’M’)=0, o nó **q** fica com uma única chave (um único nó de informação)



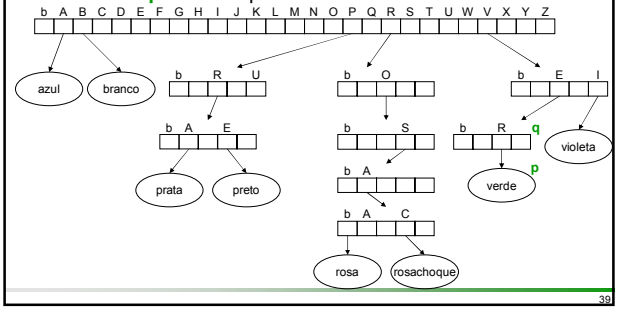
Remoção X = “vermelho”

- Isto significa que o nó **q** pode ser eliminado e **p** movido um nível para cima



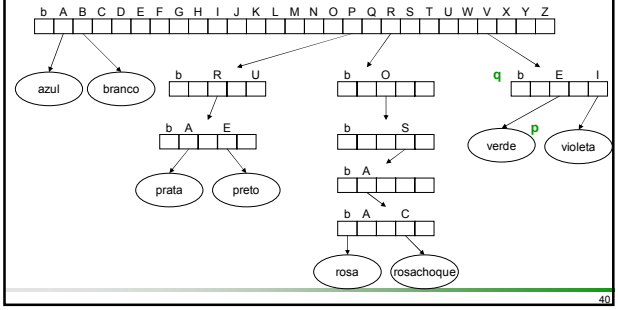
Remoção X = “vermelho”

- Novamente, **q** tem uma única chave e pode ser eliminado movendo **p** um nível para cima



Remoção X = “vermelho”

- Como **q** agora tem mais do que uma chave, o processo termina



Remoção

- Para facilitar as remoções, é útil acrescentar um campo **count** (contador) em cada nó de desvio
- Esse contador armazena quantidade de nós de informação na subárvore da qual ele é a raiz
- A escrita formal do algoritmo de remoção em árvories é deixada como exercício

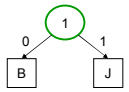
Árvores Patricia

- Patricia é abreviatura de
  - ***P**ractical **A**lgorithm **T**o **R**etrieve **I**nformation **C**oded **I**n **A**lphanumeric*
  - Algoritmo Prático para Recuperar Informação Codificada em Alfanumérico
- O algoritmo para construção da árvore Patricia é baseado no método de pesquisa digital, mas sem apresentar o inconveniente das árvories
- O problema de caminhos de uma só direção é eliminado por meio de uma solução simples e elegante: cada nó interno da árvore contém o índice do caractere a ser testado para decidir qual subárvore seguir

Inserção X = B

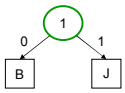
B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010

Inserção X = J



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

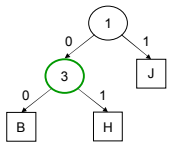
Inserção X = H



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

B e H seguem o padrão
0xxxxx.

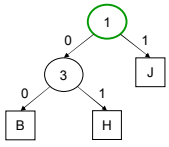
Inserção X = H



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

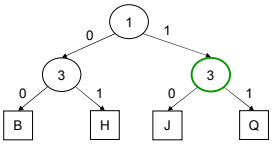
Inserção X = Q



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

J e Q seguem o padrão
1xxxxx.

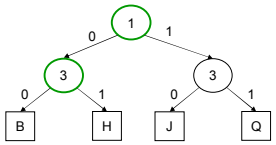
Inserção X = Q



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

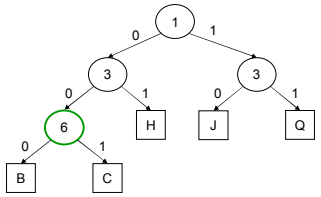
Inserção X = C



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

B e C seguem o padrão 0x0xxx.

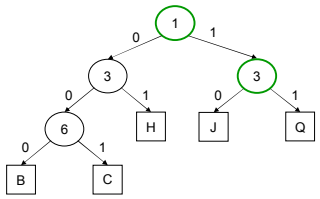
Inserção X = C



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

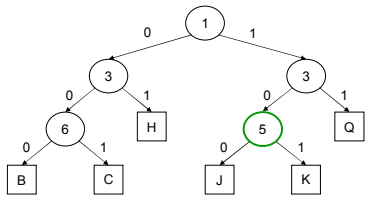
Inserção X = K



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

J e K seguem o padrão 1x0xxx.

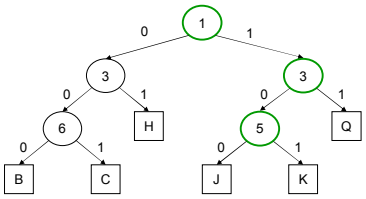
Inserção X = K



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

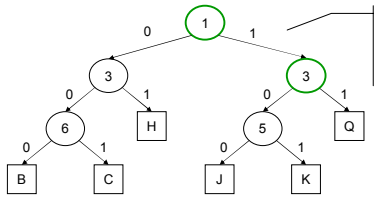
Inserção X = W



B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010
W = 110110

K e W seguem o padrão 1x0x1x.
O 2o. bit diferencia K de W

Inserção X = W

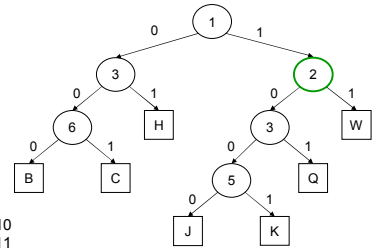


Nesse caso, o ponto de inserção será entre esses nós

B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010
W = 110110

K e W seguem o padrão 1x0x1x.
O 2o. bit diferencia K de W

Inserção X = W



B = 010010
C = 010011
H = 011000
J = 100001
Q = 101000
K = 100010
W = 110110

Algoritmo de Inserção

1. Se a subárvore atual for vazia, é criado um nó de informação com a chave X (isto ocorre somente na inserção da primeira chave) e o algoritmo termina
2. Se a subárvore atual for simplesmente um nó de informação, os bits da chave X são comparados, a partir do bit de índice imediatamente após o último índice da seqüência de índices consecutivos do caminho de pesquisa, com os bits correspondentes da chave X deste nó de informação, até encontrar um índice **i** cujos bits sejam diferentes
   - A comparação dos bits a partir do último índice consecutivo melhora o desempenho do algoritmo: se todos forem iguais, a chave já se encontra na árvore e o algoritmo termina; senão, vai para o passo 4
3. Se a raiz da subárvore atual for um nó de desvio, deve-se prosseguir para a subárvore indicada pelo bit da chave X de índice dado pelo nó atual, de forma recursiva
4. Criar um nó de desvio e um nó de informação: o primeiro contendo o índice **i** e o segundo a chave X. A seguir, o nó de desvio é ligado ao de informação pelo ponteiro de subárvore esquerda ou direita, dependendo se o bit de índice **i** da chave X seja 0 ou 1, respectivamente
5. O caminho de inserção é percorrido novamente de baixo para cima, subindo com o par de nós criados no passo 4 até chegar a um nó de desvio cujo índice seja menor que o índice **i** determinado no passo 2: este é o ponto de inserção e o par de nós é inserido

Resumo

- A pesquisa digital está baseada na representação das chaves como uma seqüência de caracteres ou de dígitos
- Os métodos de pesquisa digital são particularmente vantajosos quando as chaves são grandes e de tamanho variável