FAETERJ - FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO

CURSO TECNOLÓGICO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ARTHUR LOURENÇO MACHADO
SÉRGIO RAPHAEL AQUINO SAMPAIO
VITÓRIA MENDES PILOTO RODRIGUES

ESTRUTURA DE DADOS

ÁRVORE PATRICIA

RIO DE JANEIRO, RJ
2022

ARTHUR LOURENÇO MACHADO
SÉRGIO RAPHAEL AQUINO SAMPAIO
VITÓRIA MENDES PILOTO RODRIGUES

ESTRUTURA DE DADOS

ÁRVORE PATRICIA

Trabalho da disciplina de 3ESD apresentado à Faculdade de Educação Tecnológica do Estado do Rio de Janeiro, como parte das avaliações.

RIO DE JANEIRO, RJ
2022

SUMÁRIO

1.Introdução

PATRICIA (Practical Algorithm to Retrieve Information Coded in Alphanumeric) é um algoritmo de busca em árvores, originado das árvores RADIX (Árvore de Prefixos) e Tries. É capaz de oferecer meios flexíveis de armazenamento, indexação e recuperação de informações em um arquivo grande, contendo frases e títulos, por exemplo, economizando tempo e espaço.

Basicamente, consiste em uma árvore de caracteres em que seus caminhos internos que possuem apenas um filho que serão compactados em apenas um nó contendo os caracteres sucessores da palavra formada a partir do seu nó pai. Assim, a inserção e a remoção de palavras deve tomar o cuidado de garantir que os prefixos existentes na raíz estejam sempre de acordo com a sucessão das folhas e que não haja nó com apenas um filho.

Surgiu através da motivação de otimizar a busca de títulos em bibliotecas, e em seu uso mais atual, aplica-se nos Sistemas Gerenciadores de Banco de Dados (SGBD), e atua no ganho de eficiência em arquivos XML, além disso, originou a tecnologia utilizada na Blockchain.

2.Desenvolvimento

2.1.História

O termo foi cunhado por Donald R. Morrison, num trabalho de casamento de cadeias, aplicado à recuperação de informação em arquivos de grande porte. A primeira aparição do termo foi no jornal tecnológico ACM (Journal of the Association of Computing Machinery) em 1968. Como mencionado anteriormente, a motivação de Donald era otimizar e automatizar a busca de títulos em bibliotecas, as quais naquela época eram responsáveis por guardar mais do que apenas livros – elas armazenavam também listas telefônicas, dicionários científicos e técnicos, arquivos, entre outros tipos de documentos –. Portanto, havia a necessidade tanto dos bibliotecários em aprimorar a busca, tornando-a mais prática e intuitiva, quanto dos clientes que desejavam encontrar o título em menos tempo.

A busca anterior era baseada em índices e catálogos que eram guardados no ambiente físico e ocupavam espaço. Cada título era identificado por uma chave única, e a cada adição de novas obras no acervo, era necessário incluir uma nova chave ou alterar uma já existente, o que limitava muito a capacidade operacional da biblioteca. Assim, foi projetado o algoritmo PATRICIA para as bibliotecas que possuíam computadores, implementarem essa construção de busca para achar os alvos através de suas determinadas chaves usando o índice e o

modificando de acordo com novas inserções ou alterações, refletindo essa alteração no sistema e consequentemente na biblioteca.

A solução chegou quando a capacidade de processamento das máquinas, como a memória e processadores, eram mínimas e demoravam muito tempo para obter um retorno. Contudo, o algoritmo em sua forma original foi capaz de solucionar o problema proposto quando ele parecia ser impossível, cumprindo a busca ser feita de forma rápida e automatizando o processo. Tudo isso foi possível através da utilização de números que delimitavam a posição, dispensando textos, no índice da PATRICIA. Então, a limitação de processamento para achar um título no índice depende linearmente somente do tamanho da chave.

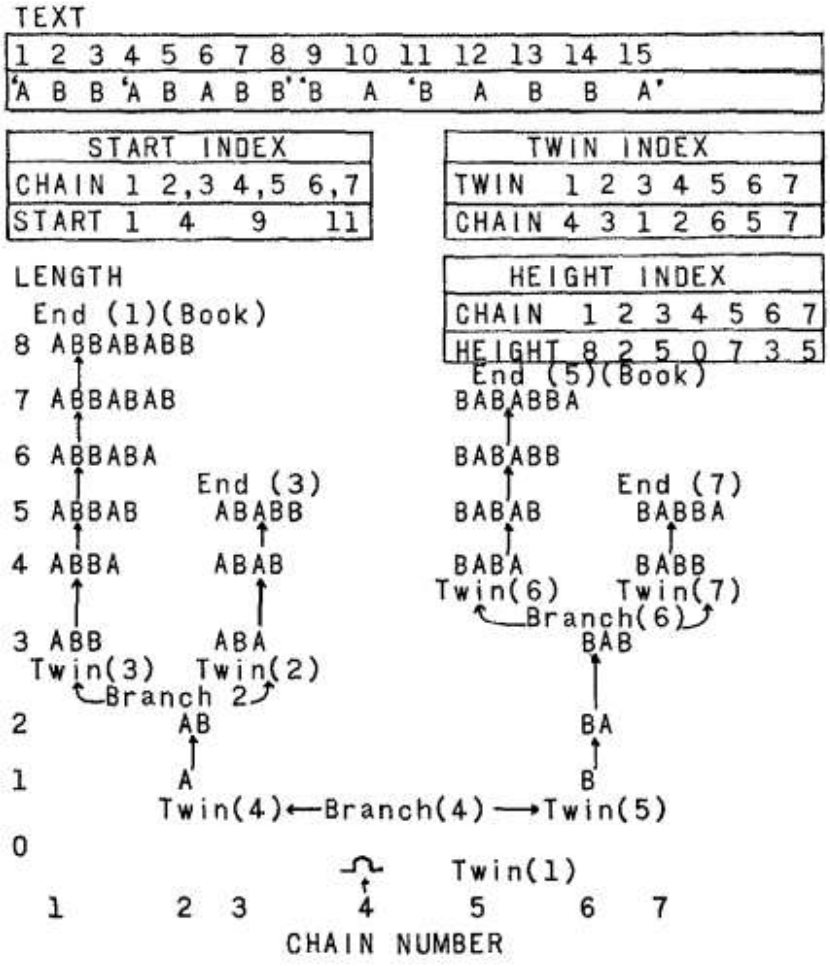

Figura 1: representação da árvore patrícia inicial no contexto da biblioteca

2.2.Características e propriedades

Contudo, com o passar do tempo e a modernização dos computadores, a representação mudou e foi atualizada. Segundo Ziviani (1999, p. 129):

> Knuth (1973) deu um novo tratamento ao algoritmo, apresentando-o de forma mais clara como um caso particular de pesquisa digital, essencialmente, um caso de árvore trie binária. Sedgewick(1988) apresentou novos algoritmos de pesquisa e de inserção baseados nos algoritmos propostos por Knuth. Gonnet e Baeza-Yates (1991) propuseram também outros algoritmos

Os nós atuais passaram a ser formados por sequências de caracteres (Strings) e identificados por meio da sua altura e ponteiros (Figura 2). Assim como nas árvores Tries, pelo mesmo motivo, as *strings* devem ter como nó raíz um operador nulo (NULL) e terminarem com o caracter terminador (na Figura 2, indicado pelo símbolo $). Mas mesmo entre tantas similaridades entre a PATRICIA e as Tries, a principal diferença entre uma e outra é a ausência de nós com somente um filho na PATRICIA, inclusive essa é a sua principal característica: cada nó é uma folha e contém pelo menos dois filhos. Isso implica imediatamente que o número de nós internos (sem serem folhas) não excedam o número de folhas. Considerando que cada folha corresponde a uma *string*, que é armazenada na árvore PATRICIA, então se a árvore PATRICIA armazena n strings, o armazenamento total usado pelos nós é O($n * |\Sigma|$). Obviamente, isso requer que também o armazenamento separado das *strings* a um custo de O(N). (N indica o comprimento total de todas as strings armazenadas na árvore Patricia.)

> **Theorem 20**: Árvore PATRICIA suporta inserção ou remoção de qualquer string s em O($|s| + |\Sigma|$) tempo e busca para s em O($|s|$) tempo. Se N é o comprimento total de todas as strings e n é o número de strings armazenadas na árvore PATRICIA, então o armazenamento utilizado é O($n * |\Sigma| + N$). (MORIN, 2014, p. 50)

Além disso, a árvore PATRICIA também suporta *prefix-matches (*ou correspondência por prefixos), nos quais podem retornar uma lista de todas as strings que possuem uma terminação não nula para a string s como prefixo, ou seja, retornam todas as ocorrências de palavras que possuem como prefixo a string s. Em termos de implementação, é feito através de uma busca comum (na qual veremos a seguir), verificando todos os caracteres em s até o final. Nesse ponto, cada folha na subárvore em que a pesquisa terminou, corresponde a uma string que começa com a string s, ou seja, que a tem como prefixo. A complexidade de tempo atribuída a essa operação é de O(|s| + k), onde s é a string procurada como prefixo e k o número de folhas.

Se o interesse é encontrar apenas uma *string* com o prefixo s, é possível verificar o começo da *string* do último nó verificando no caminho até lá a busca por s. Esse nó é representado através da representação utilizando altura e ponteiro (Figura 2) e esse ponteiro

aponta para uma string completa que possui s como prefixo. Por exemplo, considere o prefixo /ple$ da Figura 2, no qual aponta para o segundo“p” de /apple$. Portanto, esse nó final pode andar para os nós anteriores até chegar na string final. É como se fosse o caminho inverso do caso anterior.

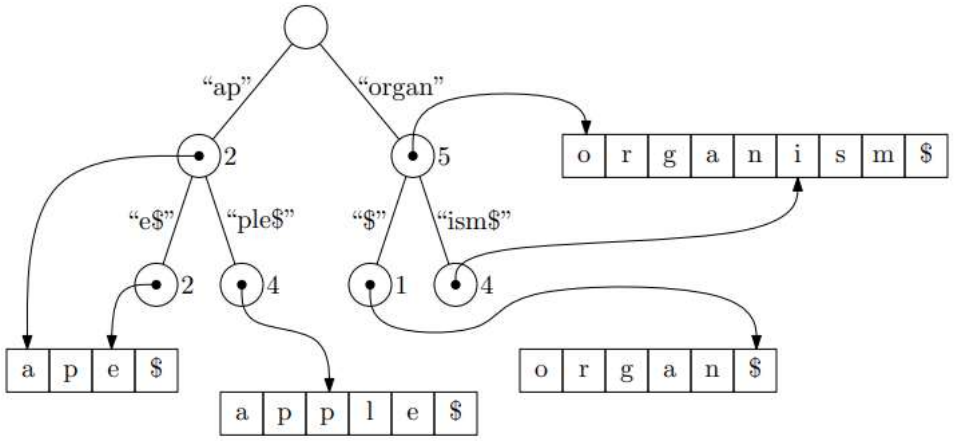

Figura 2: Representação atual da Árvore PATRICIA.

2.3.Busca

A busca na árvore PATRICIA é similar a busca na Trie, exceto pelo fato da Trie comparar substrings inteiras. Assim, a busca é feita através de cada nó comparando apenas as substrings, o que otimiza muito mais a busca, sem a necessidade de comparações desnecessárias, dado a configuração dos nós. Dessa forma, caso uma substring coincida com o outro, o nó é avançado, caso contrário, a busca falha sem encontrar a string s. Em qualquer caso, isso tomará O(|s|) de tempo. O pior caso é percorrer toda a árvore para encontrar o desejado.

2.4.Inserção

Inserir uma *string* em uma árvore PATRICIA é similar a realizar buscas em que a pesquisa chega até ao final, pois a *string* s não está na árvore. Se a busca é travada no meio de uma aresta A, então A se divide e dá origem a duas novas arestas unidas por um novo nó N. O remanescente da string s se torna rótulo da aresta, que vai de N até uma folha nova. No entanto, se a pesquisa travar no nó N, então um nó filho é criado e o restante da nova chave é usado como rótulo para aresta entre os dois. Ambos os casos tem complexidade de tempo de O($s + |\Sigma|$), em que s é a *string* que será inserida. Em pseudocódigo, o algoritmo fica:

A. 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.

B. 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 D.

C. 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.

D. 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

E. O caminho de inserção é percorrido novamente de baixo para cima, subindo com o par de nós criados no passo D até chegar a um nó de desvio cujo índice seja menor que o índice i determinado no passo B: este é o ponto de inserção e o par de nós é inserido.

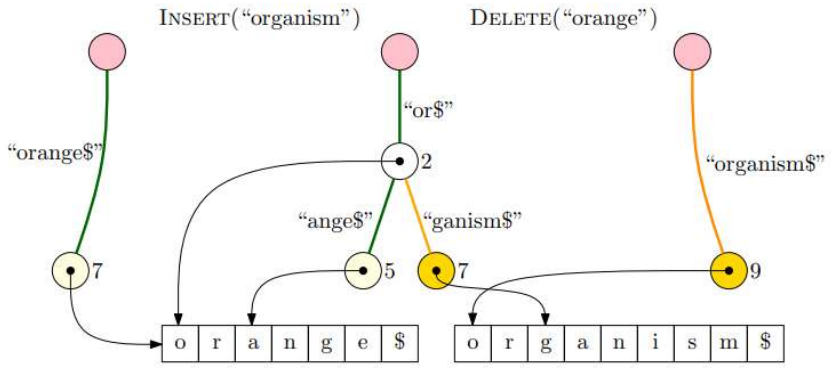

Figura 3: Inserção e remoção de dados na árvore PATRICIA.

2.5.Remoção

A remoção é o oposto da inserção. Primeiramente, é necessário localizar a folha correspondente à string **s** e removê-la da árvore. Se os parentes **u** dessa folha estão localizados à esquerda com apenas um filho **w**, então **u** é removido e substituído por uma única aresta: unindo o pai de **u** a **w**. O tempo de operação depende de quanto tempo leva para excluir dois nós de tamanho |Σ|. Se considerarmos que essa exclusão é uma operação de tempo constante, então o tempo de execução é O(|s|). Se considerarmos que essa exclusão leva tempo O(N),

então o tempo de execução é O(|s| + N).

2.6.Aplicações

A árvore PATRICIA é largamente utilizada na Blockchain, em especial na criptomoeda Ethereum para guardar dados, como:

1. stateRoot
2. storageRoot
3. transactionRoot
4. receiptRoot

A aplicação da PATRICIA nesse caso é mesclada à Merkle Tree e Radix Tree, para ser capaz de armazenar os dados da tabela hash. Essa solução também é encontrada para a melhoria dos arquivos XML e nos Sistemas Gerenciadores de Banco de Dados, otimizando a busca.

3.Conclusão

Portanto, como vimos, o algoritmo de busca PATRICIA ainda é muito utilizado atualmente, sendo usado em união à tabela hash. Ao longo dos anos houve várias melhorias e a tendência é que cada vez mais seu uso seja aprimorado, a fim de atingir soluções rápidas e eficientes aos problemas futuros, dado que o volume de dados aumenta exponencialmente a cada dia e esse mecanismo foi criado para lidar com essa quantidade de dados.

4.Referências bibliográficas

MORRISON, Donald R. 1968. **PATRICIA - Practical Algorithm to Retrieve Information Coded in Alphanumeric**. Journal of the Association of Computing Machinery. Vol. 15, No. 4, pp. 514-534.

MORIN, P. Advanced Data Structures - Chapter 7: **Data Structures for Strings**. School of Computer Science, Carleton University. Ontario, 2014.

SOUZA, Jairo. **Busca Digital (Trie e Árvore Patrícia).** UFRJ, 2009. Disponível em: https://www.ufjf.br/jairo_souza/files/2009/12/6-Strings-Pesquisa-Digital.pdf. Acesso em: 18 nov. 2022.

ZIVIANI, Nivio. **Projeto de algoritmos**: com implementações Pascal e C. 4. ed. São Paulo: Pioneira Informática, 1999. 129-130 p.

Song, Huazhu & Sun, Wenting & Zhang, Mingzhi & Lu, Yi. (2009). **XML Index Algorithm Based on Patricia Tries**. Web Information Systems and Mining, International Conference on. 289-293. 10.1109/WISM.2009.67.