Algoritmos e Estruturas de Dados II

Capítulo da Aula 10: Processamento de Texto e Árvores Trie

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br
CEFET-MG - Campus Timóteo

Fevereiro de 2026

1. O Gargalo do Processamento de Strings

Na engenharia de sistemas contemporânea, estruturas balizadas por chaves numéricas simples (como Árvores AVL ou Hashes puros) colapsam ou se tornam terrivelmente ineficientes quando expostas a um oceano de **buscas textuais massivas**. Se você usar uma `BST` convencional para estocar um dicionário de 1 milhão de palavras, a operação de comparação não é mais $O(1)$. Comparar duas strings longas lexicograficamente exige um varredura caractere por caractere. A busca degradaria para $O(L \log N)$, onde L é o comprimento da palavra. Pior ainda: e se precisarmos buscar **todas as palavras que começam com o prefixo "auto"** (como no auto-completar do Google)? Hashes não preservam ordem, e BSTs exigiriam percursos pesados nas ramificações.

O domínio de processamento de texto exigia uma abstração morfológica superior projetada nativamente para navegar em "dígitos" ou "caracteres", fragmentando a chave. Nascia a **Trie** (Árvore de Prefixos).

2. Árvore Trie (Prefix Tree)

A Trie (derivada da palavra *reTRIEval*, recuperação) é uma estrutura fundadora de indexação onde a chave não está estocada de forma monolítica e isolada dentro de um nó, mas sim **desmembrada e espalhada pela trajetória do caminho** adotado partindo da raiz.

2.1. Anatomia do Nó Trie

Diferente da topologia binária restritiva, um nó Trie abandona o conceito de "esquerda/direita". Ele carrega em si um Array (vetor contíguo) de ponteiros dimensionado perfeitamente para suportar todo o limite geográfico do Alfabeto alvo. Se estivermos indexando caracteres ASCII em minúsculo, o nó ostentará **26 pontes vazadas** apontando para o vazio (Null). O caractere em si não fica necessariamente armazenado no nó; a própria posição (índice) do ponteiro lido é o caractere.

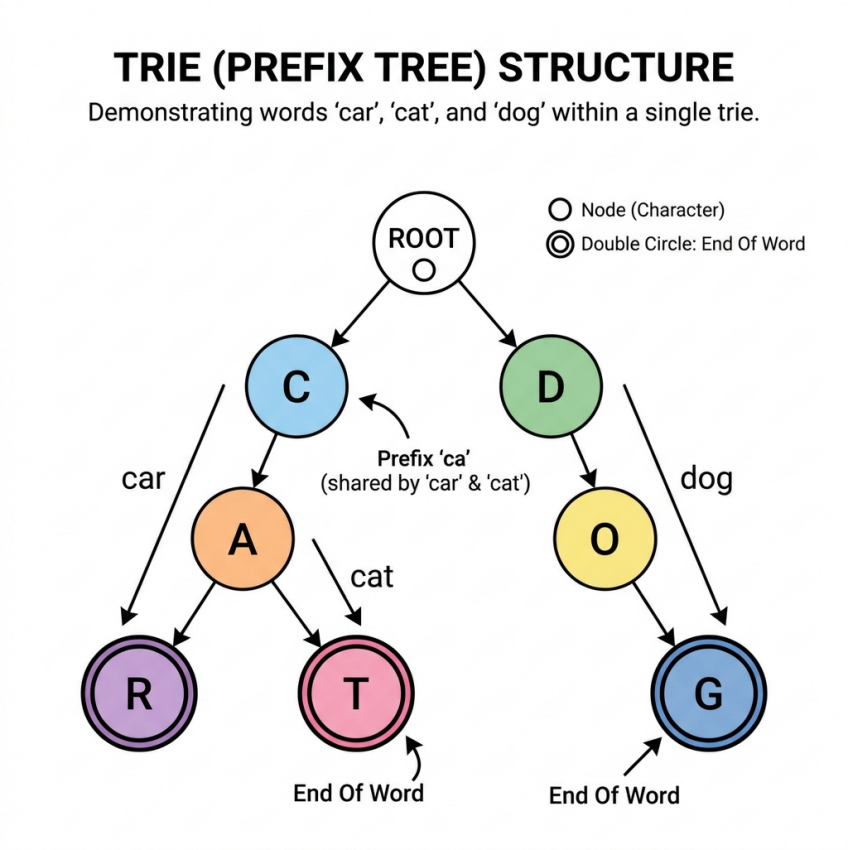

Figure 1: Morfologia da Trie: A palavra é reconstruída unindo as arestas no caminho navegado, finalizando na marcação End Of Word.

2.2. Inserção Geométrica Assintótica ($O(L)$)

A mágica suprema da Trie é que o tempo de inserção de uma palavra **independe** da quantidade brutal de chaves N já englobadas no índice global da árvore. A inserção caminha cega e fluida, alocando novos nós pontualmente apenas para cada letra final desbravada. Ao injetar a palavra "carro" (5 letras), gastaremos parcos e ínfimos **5 passos diretos de ponteiro** $O(L)$, enveredando no tecido base e finalizando com o estalo do carimbo da *flag* de encerramento (`isEndOfWord = true`).

3. Implementação Matricial (Java Padrão)

Para evitar a complexidade do uso de tabelas hash explícitas, podemos ancorar as pontes num vetor puro indexado onde 'a' equivale ao índice 0, 'b' a 1, etc:

```
class TrieNode {
    TrieNode[] children;
    boolean isEndOfWord;

    public TrieNode() {
        children = new TrieNode[26];
        isEndOfWord = false;
```

```
    }
}

class Trie {
    private TrieNode root;

    public Trie() {
        root = new TrieNode();
    }

    // Insercao cravada em O(L)
    public void insert(String word) {
        TrieNode current = root;
        for (int i = 0; i < word.length(); i++) {
            int index = word.charAt(i) - 'a'; // Mapeamento ascii logico 0.
            if (current.children[index] == null) {
                current.children[index] = new TrieNode();
            }
            current = current.children[index];
        }
        current.isEndOfWord = true;
    }

    // Busca Exata massificada em O(L)
    public boolean search(String word) {
        TrieNode current = root;
        for (int i = 0; i < word.length(); i++) {
            int index = word.charAt(i) - 'a';
            if (current.children[index] == null) return false;
            current = current.children[index];
        }
        return current.isEndOfWord;
    }
}
```

4. O Dilema: Velocidade Suprema vs Sangria de Memória

A Trie tradicional é inquestionável quando o assunto é o trânsito da CPU (tempo de travessia logístico computacional puro estático $O(L)$). Entretanto, ela cobra um ágio sombrio no pedágio da Memória RAM: a famigerada **Fragmentação Ociosa Estéril**.

Analise a palavra "algoritmo". O nó atrelado ao sufixo longo e direto "ritmo" forjará uma cascata morta vertical cega pendente em linha reta de 5 nós solitários onde 25 das 26 pontes alocadas passivamente em `children[26]` apontarão friamente para escombros ociosos espaciais marginais nulos (`null`). O desperdício arquitetural limiar de memória explode. Para contra-atacar e aplacar essa fumaça vazada e letalidade, a indústria avança aplicando pesadamente a topologia **Patricia Trie (Radix Tree)** que consolida brutal-

mente os cordões de nódulos solitários em blocos textuais fundidos encapsulados engessados inteiros maciços, compactando e ramificando arestas engatilhadas só e estritamente quando acorre colisão divergente nas raias restritas cruzeiras cegas do prefixo.

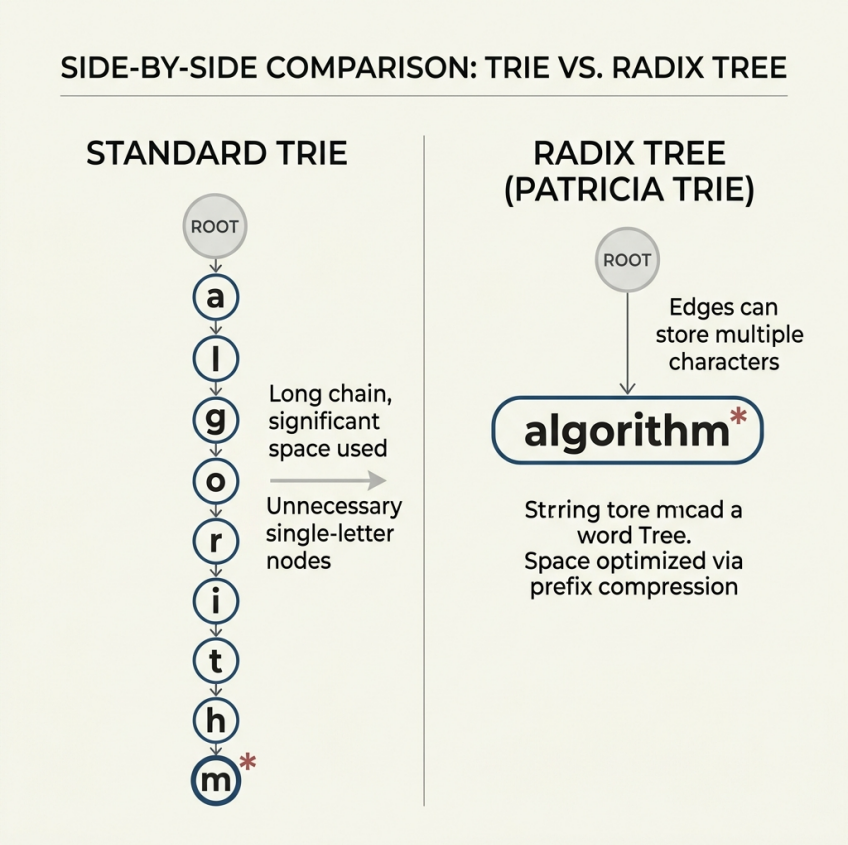

Figure 2: Trie Tradicional ladeando as otimizações brutais mortais encolhedoras da Radix Tree, dizimando blocos vazios inertes laterais de ponteiros nulos.

5. Aplicações de Classe Global (Engenharia Contemporânea)

Tries e suas vertentes radicais compõem a espinha dorsal escondida que ampara os processamentos logísticos cegos em redes contíguas limpas atreladas a alta transação:

- **Autocompletar (Auto-Suggest):** Google Busca e VSCode utilizam Tries densificadas mapeadas onde a simples inserção limpa do usuário ancora-se direto na borda do prefixo passivo vivo alvo matricial estático transversal cravado e, via percurso em varredura BFS passiva estanque contíguo passiva limpa cega limpa, brota e irradia varrendo entregando estaticamente a completude lateral contígua.
- **Roteamento IP Base de Redes Puras (Longest Prefix Match):** Os pacotes de rotas varridos puramente na crosta dos colossais Roteadores Backbone da Cisco e BGP processam nativamente redes em IP emulados cruzados num vetor matricial vital passivo de percurso Trie convertendo a máscara da rede numa teia atarracada passiva.

- **Extirpação Lexical Plena (Corretores):** Varredura plena textuais pesadas atômicas brutas engessadas submetem o bloco de caracteres cegos massivos inteiros da tela para cair e tracionar vital plácido vivo passivo engatilhado nas garras logísticas limpas validadoras de Tries operantes matriciais em tempo contíguo assombrado constante invisível.

6. Exercícios Analíticos e Teóricos

6.1. Simulações Teóricas

1 Desenhe a morfologia da malha hierárquica baseada pura para alocação nativa Trie contendo exatamente as palavras limitadas: `bola, bolo, bolacha, casa, carro`. Sinalize precisamente em círculo duplo os nós cravados com a etiqueta `isEndOfWord=true`.

2 Em tese abstrata absoluta para seu cenário hipotético, some todos as pontuais pontes alocáveis vitais engessadas limpas operacionais passivas abertas e ejetáveis de um casulo puro da matriz global pregressa do exercício `"1"` se a capacidade estrutural nativa da árvore foi forjada a ferro com array restritivo fechado alfanumérico padrão isolando limite alocado passivo universal nulo operante limpo fechado isolado de tamanho 26 (`PonteirosTrie[26]`). E após, cite qual o peso fantasma do desperdício de vetores `null` puros ociosos plenos.

3 Em Engenharia Sistêmica de Alta Camada Logística, avalie o suicídio estrutural passivo alocado e limite contíguo logístico puro passivo vivo de se processar dicionários universais de alfabetos asiáticos unificados vivos de mais de 15.000 ideogramas estáticos puros se tentassem submeter os nós à matriz rígida limpa base transversal limpa cega estúpida estática da `Array[15000]` alocado e engessado num sistema operante limpo passivo. Quais são as soluções viáveis atreladas para amenizar este dolo isolador puro pleno?