**Referências bibliográficas em formato ABNT para o estudo de Árvores binárias.**  
  
FREDKIN, Edward. Trie memory. *Communications of the ACM*, New York, v. 3, n. 9, p. 490-499, 1960.

MORRISON, Donald R. PATRICIA — Practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, New York, v. 15, n. 4, p. 514-534, 1968.

ZIVIANI, Nivio. *Projeto de algoritmos com implementações em Pascal e C*. 3. ed. São Paulo: Cengage Learning, 2011.

SEDGEWICK, Robert; WAYNE, Kevin. *Algorithms*. 4. ed. Boston: Addison-Wesley, 2011.

KNUTH, Donald E. *The art of computer programming. Volume 3: Sorting and Searching*.

**Trechos e partes mais relevantes dos artigos acima.**

**📄 Artigos científicos**

**FREDKIN, Edward. Trie Memory. CACM, 1960.**

* Introduz formalmente a **árvore digital (trie)**.
* Pontos principais:
  + Estrutura hierárquica para **armazenar e recuperar chaves (strings)**.
  + Cada nível da árvore representa **um caractere da chave**.
  + Busca é realizada seguindo o caminho da raiz até a folha.
* Frase importante (tradução livre):  
  *“The trie provides an efficient representation for a set of words, where retrieval is achieved by traversing the tree structure character by character.”*

**MORRISON, Donald R. PATRICIA — Practical Algorithm to Retrieve Information Coded in Alphanumeric. JACM, 1968.**

* Apresenta a **árvore Patricia** (Practical Algorithm to Retrieve Information Coded in Alphanumeric).
* Pontos principais:
  + É uma **variação compacta da árvore digital binária**.
  + Remove nós intermediários que têm apenas um filho, economizando espaço.
  + Operações de **busca, inserção e remoção** são feitas de forma eficiente.
* Frase importante (tradução livre):  
  *“The Patricia tree eliminates unnecessary nodes in the digital search tree, thus reducing storage requirements while maintaining efficient retrieval.”*

**📚 Livros**

**ZIVIANI, Nivio. Projeto de Algoritmos com Implementações em Pascal e C. 2011.**

* Livro brasileiro, muito usado em universidades.
* **Capítulo 9 – Árvores Digitais**.
* Conteúdo principal:
  + Introdução às árvores digitais (*tries*).
  + Implementação em Pascal e C.
  + Discussão sobre **árvores Patricia** como versão compacta.
  + Exemplos de busca de chaves em dicionários.

**SEDGEWICK, Robert; WAYNE, Kevin. Algorithms. 2011.**

* **Capítulo 5 – Strings**.
* Conteúdo principal:
  + Apresenta **Tries** como solução para busca eficiente de palavras em dicionários.
  + Explica a aplicação em **autocompletar, compressão e tabelas de símbolos**.
  + Discute também a **Patricia trie** (chamada de “compressed trie”).

**KNUTH, Donald E. The Art of Computer Programming, Vol. 3. 1998.**

* **Seção 6.3 – Digital Searching**.
* Conteúdo principal:
  + Explica o conceito matemático de **árvores digitais (digital search trees)**.
  + Descreve o funcionamento de **tries e Patricia** em detalhe.
  + Analisa complexidade de tempo e espaço.
* Considerada a referência mais **rígida e formal** sobre o assunto.

✅ Ou seja:

* Para **definição e origem**, use **Fredkin (1960)**.
* Para **otimização (Patricia)**, cite **Morrison (1968)**.
* Para **implementação prática**, use **Ziviani (2011)**.
* Para **aplicações modernas**, use **Sedgewick & Wayne (2011)**.
* Para **fundamentação teórica rigorosa**, cite **Knuth (1998)**.

Árvores Binárias

Introdução

Árvore digital, também conhecida como Trie, é uma estrutura de dados do tipo árvore ordenada. Essa estrutura é especialmente útil para armazenar e buscar palavras, permitindo operações rápidas de inserção, busca e exclusão.   
 A árvore é formada por nodos que contêm os caracteres componentes de uma cadeia, ficando armazenado nas folhas as informações sobre a cadeia. Um nodo da arvore é formado por um vetor no qual cada elemento é formado por três campos: campo que contém o caractere representado pelo nodo; campo que indica o sucessor desse nodo (apontando para um nodo contendo o caractere que o sucede dentro da cadeia); campo que aponta para um registro associado a cadeia (no caso de este ser o último caractere da cadeia). A estrutura permite inserção ou pesquisa de uma cadeia em um número pré-definido de acessos dado pelo comprimento da cadeia. Isto garante um tempo de acesso pequeno em estrutura com grande número de palavras armazenada pois a quantidade de palavras mantidas na estrutura não altera a performance do método. Além disso o método permite a otimização do tamanho total da estrutura pois o utiliza o prefixo das palavras já existentes para formar uma nova palavra.

**🌳 Árvores Digitais (Trie)**

**Introdução**

A **árvore digital**, também conhecida como **Trie**, é uma estrutura de dados do tipo árvore ordenada, utilizada principalmente para **armazenar e buscar cadeias de caracteres (strings)**. Sua principal vantagem está na eficiência das operações de **inserção, busca e exclusão**, cujo tempo de execução depende apenas do comprimento da cadeia, e não da quantidade de palavras armazenadas.

**Estrutura básica**

A árvore é formada por **nodos (ou nós)**, que representam os caracteres componentes de uma cadeia. Os **prefixos comuns** entre palavras são compartilhados, o que permite economia de memória e organização compacta dos dados.

Um **nó da árvore digital** pode ser representado por:

* **Caractere** → símbolo que o nodo representa.
* **Apontador para o próximo nodo** → indicando o sucessor dentro da cadeia.
* **Indicador de fim de cadeia** → marcação (ou ponteiro) para um registro associado, caso o caractere seja o último da palavra.

As **folhas** da árvore representam palavras completas, enquanto os **nós internos** representam prefixos intermediários.

**Funcionamento**

* **Inserção**: percorre-se a árvore caractere por caractere, criando novos nós quando necessário.
* **Busca**: percorre-se a sequência de nós; se todos os caracteres forem encontrados e o nó final for marcado como terminal, a palavra existe.
* **Exclusão**: remove-se a marca de terminal ou, caso não haja mais dependências, apagam-se nós desnecessários.

**Vantagens**

* **Eficiência**: a busca e a inserção ocorrem em tempo proporcional ao tamanho da palavra (*O(m)*, onde *m* é o comprimento da cadeia).
* **Escalabilidade**: o número de palavras armazenadas não altera significativamente a perfo