

# **IMPLEMENTAÇÃO DE UMA ESTRUTURA TRIE PARA VERIFICAÇÃO E SUGESTÃO DE PALAVRAS EM PROCESSAMENTO DE TEXTO**

DAVI LEMOS E HEDUARDO WITKOSKI



# INTRODUÇÃO



- Corretores ortográficos e sistemas de autocompletar exigem soluções rápidas e eficientes devido ao aumento do volume de palavras.
- Métodos tradicionais de busca se tornam ineficazes em grandes volumes de dados.
- A Trie organiza palavras hierarquicamente, otimizando a busca e sugestões a partir de prefixos, ideal para sistemas que exigem respostas rápidas.

## ●●● DEFINIÇÃO DA PROBLEMÁTICA

A problemática abordada envolve a verificação de palavras e sugestão de alternativas em sistemas interativos, como corretores ortográficos e autocompletar, especificamente no contexto da língua inglesa. Com o aumento constante de dados, torna-se crucial a utilização de estruturas de dados eficientes para buscar rapidamente em grandes volumes de palavras, garantindo uma experiência de usuário ágil e precisa.

# ... METODOLOGIA

- Implementação realizada em Java.
- As palavras em inglês foram carregadas a partir de um arquivo de texto, onde as palavras são organizadas para buscas eficientes.





# ACESSO AO ARQUIVO



```
// Carregar palavras de um arquivo para a Trie
public void carregarDicionario(String caminhoArquivo) {
    try (BufferedReader leitor = new BufferedReader(new FileReader(caminhoArquivo))) {
        String palavra;
        int indice = 0;
        while ((palavra = leitor.readLine()) != null) {
            dicionario.inserir(palavra.trim().toLowerCase(), indice++);
        }
        System.out.println("Dicionário carregado com sucesso.");
    } catch (IOException erro) {
        System.err.println("Erro ao carregar o dicionário: " + erro.getMessage());
    }
}
```


**INSERÇÃO**

# ... INSERÇÃO DAS PALAVRAS

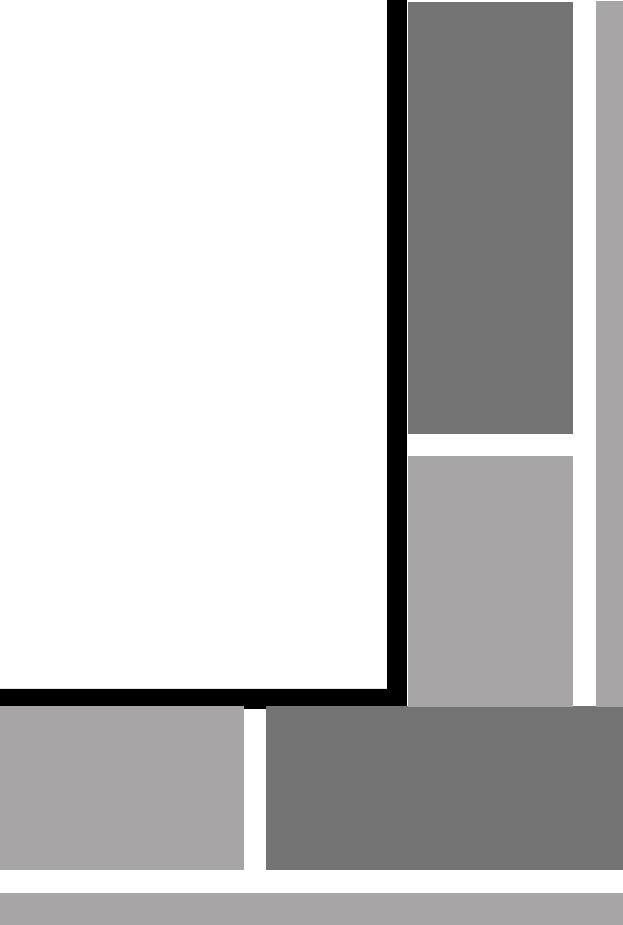


```
// Inserir uma palavra na Trie
public void inserir(String chave, Valor valor) {
    No atual = raiz;
    for (char caractere : chave.toCharArray()) {
        atual.proximos.putIfAbsent(caractere, new No());
        atual = atual.proximos.get(caractere);
    }
    atual.valor = valor;
}
```


# BUSCA



```
// Buscar uma palavra na Trie
public Valor buscar(String chave) {
    No atual = raiz;
    for (char caractere : chave.toCharArray()) {
        if (!atual.proximos.containsKey(caractere)) {
            return null;
        }
        atual = atual.proximos.get(caractere);
    }
    return (Valor) atual.valor;
}
```



# SUGESTÕES



```
// Obter todas as palavras com um prefixo
public Iterable<String> chavesComPrefixo(String prefixo) {
    LinkedList<String> resultados = new LinkedList<>();
    No atual = raiz;
    for (char caractere : prefixo.toCharArray()) {
        if (!atual.proximos.containsKey(caractere)) {
            return resultados; // Retorna vazio se o prefixo não existir
        }
        atual = atual.proximos.get(caractere);
    }
    coletar(atual, new StringBuilder(prefixo), resultados);
    return resultados;
}

// Método auxiliar para coletar palavras
private void coletar(No no, StringBuilder prefixo, LinkedList<String> resultados) {
    if (no == null) return;
    if (no.valor != null) resultados.add(prefixo.toString());
    for (char caractere : no.proximos.keySet()) {
        coletar(no.proximos.get(caractere), prefixo.append(caractere), resultados);
        prefixo.deleteCharAt(prefixo.length() - 1);
    }
}
}
```



# RESULTADOS



Dicionário carregado com sucesso.

Digite uma palavra para verificar (ou 'sair' para encerrar): *Hello*

A palavra 'Hello' está correta.

Digite uma palavra para verificar (ou 'sair' para encerrar): *World*

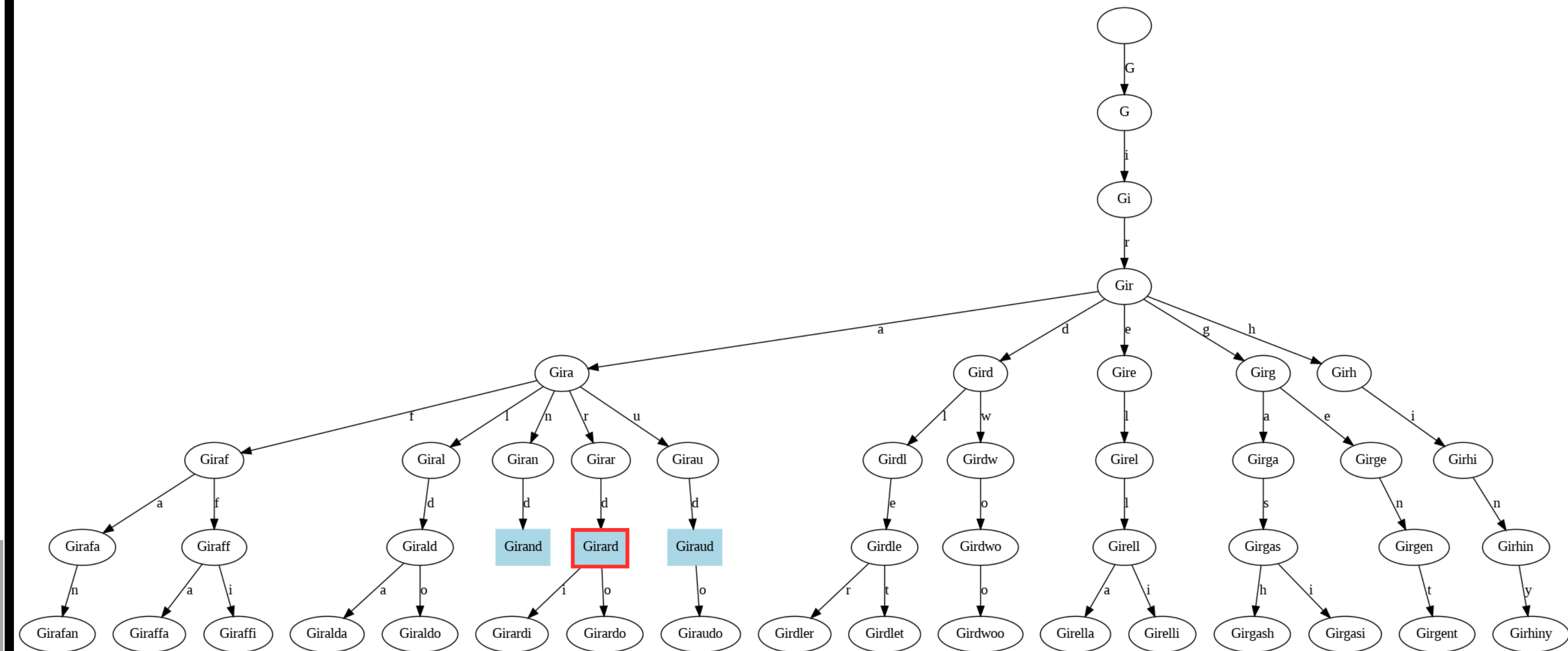
A palavra 'World' está correta.

Digite uma palavra para verificar (ou 'sair' para encerrar): *Gir*

A palavra 'Gir' não está no dicionário.

Sugestões:

- girard
- girardi
- girardo
- girasol





## CONSIDERAÇÕES FINAIS

- A Trie é eficaz para verificação e autocompletar em grandes listas de palavras.
- O trabalho mostra o potencial da Trie para melhorar a performance em sistemas de busca e sugestões.
- Perspectivas incluem escalabilidade e aplicação em outros contextos.



# Obrigado!

Dúvidas e Sugestões:

[davilemos.aluno@unipampa.edu.br](mailto:davilemos.aluno@unipampa.edu.br)

[heduardorochoa.aluno@unipampa.edu.br](mailto:heduardorochoa.aluno@unipampa.edu.br)

