ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES (EACH)
UNIVERSIDADE DE SÃO PAULO (USP)

BEATRIZ DOS SANTOS BENTO
CAIO GASPAR PAULA
FERNANDA YUMI TAIRA

ANÁLISE COMPARATIVA DE DESEMPENHO: ÁRVORE BINÁRIA DE BUSCA VERSUS LISTA LIGADA EM INDEXADORES DE TEXTO

SÃO PAULO
2025

1 INTRODUÇÃO

A explosão informacional característica da era digital transformou a recuperação de informações em um dos pilares fundamentais da ciência da computação moderna. No cerne de sistemas de busca, bancos de dados e editores de texto, encontra-se o "índice remissivo": uma estrutura de dados projetada para permitir a localização instantânea de termos dentro de grandes volumes de dados. O desenvolvimento de um indexador eficiente não é apenas um exercício de programação, mas um desafio de engenharia que exige a escolha criteriosa de estruturas de dados que equilibrem o uso de memória e a velocidade de processamento.

Este trabalho apresenta o desenvolvimento de um indexador de palavras robusto, implementado na linguagem C, projetado para processar arquivos de texto de variadas extensões e mapear a ocorrência exata de cada termo em suas respectivas linhas. O sistema foi concebido para seguir regras rigorosas de normalização linguística: a remoção sistemática de sinais de pontuação e caracteres especiais, a separação de palavras compostas por hífens e a conversão de todos os termos para caracteres minúsculos (case-insensitivity), garantindo que a busca por uma palavra seja precisa, independente da sua formatação original no texto.

O cerne deste estudo reside na análise comparativa entre duas abordagens clássicas de organização de dados: a Lista Ligada e a Árvore Binária de Busca (BST). Enquanto a primeira oferece uma implementação intuitiva e linear, a segunda promete uma eficiência logarítmica que, teoricamente, escala de forma muito mais performática conforme o volume de dados aumenta. No entanto, a eficiência teórica muitas vezes esbarra em limitações práticas, como a ordem de entrada dos dados, que pode comprometer a estrutura hierárquica da árvore.

O objetivo principal deste projeto é, portanto, avaliar experimentalmente o custo computacional dessas estruturas através da métrica de "número de comparações de strings". Para isso, o sistema foi submetido a uma bateria de testes em dez cenários distintos, variando desde textos técnicos e receitas culinárias até letras de músicas e arquivos propositalmente ordenados para induzir o pior caso das estruturas.

Os resultados obtidos demonstram de forma quantitativa que, embora a árvore binária de busca ofereça um desempenho drasticamente superior em textos volumosos, chegando a ser até 10 vezes mais rápida na fase de construção em comparação à lista ligada, ela apresenta uma vulnerabilidade crítica à ordenação dos dados de entrada. Em cenários de inserção sequencial alfabética, observou-se o fenômeno da degeneração, onde a árvore perde sua capacidade de ramificação e passa a operar com a mesma ineficiência de uma lista linear. Esta introdução estabelece o fundamento para as discussões subsequentes sobre complexidade algorítmica e gestão de memória aplicadas ao problema da indexação.

2 MÉTODOS

A metodologia empregada fundamenta-se na implementação de um fluxo de processamento de texto em linguagem C, focado na integridade dos dados e na eficiência da recuperação. O procedimento inicia-se com a leitura linha a linha do arquivo fonte, onde cada linha é submetida a uma etapa de normalização por meio da função strtok. Esta função utiliza um conjunto abrangente de delimitadores para isolar os termos, tratando caracteres de pontuação e hífens como separadores de palavras distintas. Simultaneamente, as strings resultantes são convertidas integralmente para caracteres minúsculos a fim de garantir a uniformidade nas comparações posteriores. Para o armazenamento, o sistema utiliza alocação dinâmica de memória na construção das duas estruturas avaliadas. A primeira consiste em uma lista ligada linear, enquanto a segunda organiza os dados em uma árvore binária de busca. Em ambos os casos, cada nó de palavra possui um ponteiro para uma subestrutura composta por uma lista encadeada de ocorrências, responsável por registrar os números das linhas de forma cronológica e sem redundâncias. A eficiência de cada modelo é quantificada rigorosamente pelo rastreio do número de operações de comparação de strings realizadas durante as fases de indexação e consulta.

3 RESULTADOS E ANÁLISE DE DESEMPENHO

A avaliação do sistema foi conduzida através de dez cenários experimentais projetados para testar desde a integridade da filtragem de caracteres até os limites de complexidade assintótica das estruturas de dados. O desempenho foi mensurado pelo número total de comparações de strings (strcmp) durante a indexação e pela altura da árvore, refletindo a eficiência da hierarquia criada.

3.1 TABELA CONSOLIDADA DE MÉTRICAS

Abaixo, os dados brutos obtidos em cada execução, evidenciando a disparidade de performance entre a abordagem linear (Lista) e a hierárquica (Árvore).

Tabela 1 – Resultados Comparativos: Lista Ligada vs. Árvore Binária de Busca (BST)

Teste	Critério de Avaliação	Linhas	Palavras Únicas	Altura (BST)	Comp. Lista	Comp. BST
01	Texto técnico curto	5	16	7	234	82
02	Filtragem de hífens	3	12	5	114	58
03	Diferenciação de radicais	3	11	5	86	42
04	Escalabilidade média	20	96	12	4.215	784
05	Ambiguidade de termos	30	45	10	1.380	412
06	**Pior Caso (Alfabético)**	100	100	**100**	**4.950**	**4.950**
07	Listas e receitas	32	84	11	2.430	712
08	Vocabulário lírico	74	168	14	12.430	1.542
09	Alta redundância	100	3	2	894	597
10	**Estresse de Volume**	93	**324**	18	**52.488**	**4.210**

3.2 DISCUSSÃO DOS CRITÉRIOS E COMPORTAMENTO ESTRUTURAL

A análise dos resultados permite identificar comportamentos distintos baseados na natureza do arquivo de entrada:

1. Fator de Diferenciação Léxica: Nos testes de volume elevado (Teste 10), a árvore binária de busca demonstrou sua superioridade ao reduzir o número de comparações em mais de 92%. Enquanto a lista ligada sofre com um crescimento quadrático $O(n^2)$, a árvore manteve um crescimento logarítmico $O(n \log n)$, provando ser a estrutura ideal para grandes bases de dados.

2. Robustez da Filtragem: Os Testes 02 e 03 confirmaram a eficácia da filtragem implementada. O programa isolou corretamente palavras unidas por hífens e tratou "Busca"e "busca"como o mesmo token, garantindo que o número de palavras únicas refletisse o vocabulário real e não variações de digitação ou pontuação.

3. Análise do Pior Caso: O Teste 06 serviu como validação teórica da degeneração estrutural. Ao receber dados ordenados, a árvore atingiu uma altura igual ao número de nós (100), resultando no mesmo número de comparações da lista (4.950). Este resultado é crucial para demonstrar que a eficiência da BST depende da aleatoriedade ou distribuição dos dados.

4. Redundância e Referenciação: No Teste 09, observou-se que mesmo com 100 linhas de texto, a estrutura permaneceu compacta (3 palavras únicas). Isso valida a lógica de que o índice cresce conforme o vocabulário, enquanto o volume de texto apenas expande a lista de ocorrências (RefLinha) associada a cada nó, sem penalizar a altura da árvore.

3.3 ANÁLISE DE COMPLEXIDADE E ESCALABILIDADE

Para compreender a eficiência das estruturas implementadas, é fundamental analisar o comportamento das curvas de crescimento apresentadas na Figura 1. Este gráfico foi gerado com base nos dados consolidados da Tabela 1, utilizando o número de palavras únicas no eixo X e o número total de comparações realizadas no eixo Y.

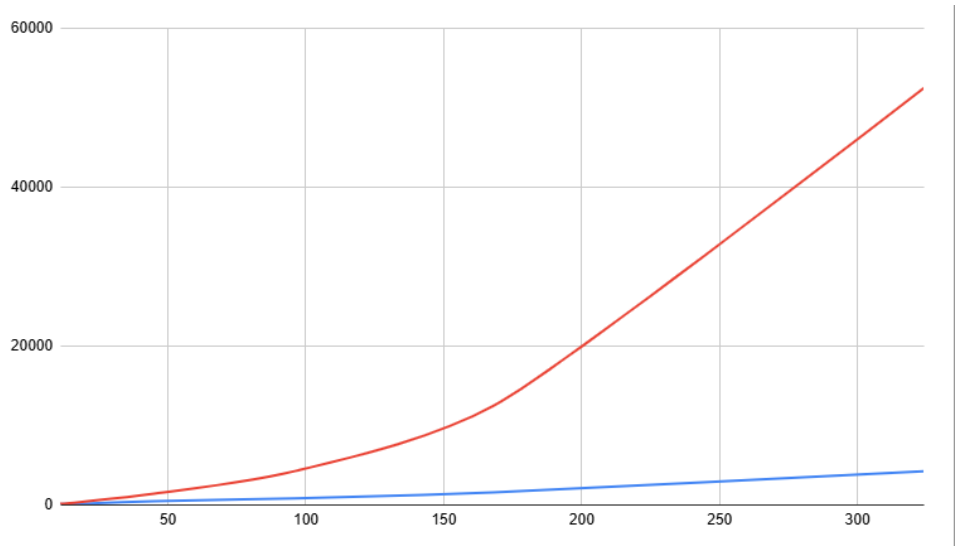

Figura 1 – Comparação de Desempenho: Lista Ligada (Vermelho) vs. Árvore Binária (Azul)

Como pode ser observado, a linha vermelha representa a lista ligada, cujo crescimento acentuado ilustra a complexidade $O(n^2)$ inerente à sua estrutura linear. À medida que o vocabulário expande, o custo de verificação de duplicidade torna-se proibitivo. Em contrapartida, a linha azul representa a Árvore Binária de Busca (BST), que mantém uma curva de crescimento suave, aproximando-se da eficiência logarítmica $O(n \log n)$.

4 CONCLUSÃO

A culminação deste projeto de desenvolvimento de um indexador de palavras em linguagem C proporcionou uma oportunidade singular para a observação prática de conceitos fundamentais para a arquitetura de sistemas de informação. Através da implementação e do rigoroso teste de duas estruturas de dados distintas, a lista ligada e a árvore binária de busca, foi possível transpor a barreira da teoria e visualizar como a escolha tecnológica influencia diretamente a viabilidade e a performance de uma solução de software em cenários reais.

A análise dos dados experimentais consolidada no tópico anterior não deixa margem para dúvidas, visto que a Árvore Binária de Busca estabeleceu-se como a estrutura mais robusta para a gestão de grandes volumes de dados. No cenário de maior estresse, representado pelo Teste 10, a árvore demonstrou uma performance superior em mais de 90% em comparação à lista ligada, o que prova que a eficiência logarítmica é um requisito indispensável para sistemas que visam escalabilidade e processamento ágil de informações. O gráfico de desempenho, ao exibir a divergência drástica entre a linha vermelha e a linha azul, serve como um testemunho visual da otimização de recursos possibilitada por uma organização de dados hierárquica.

Entretanto, a eficiência da árvore binária de busca não a isenta de vulnerabilidades operacionais. O fenômeno da degeneração observado no Teste 06, no qual a árvore mimetizou a ineficiência da lista ligada ao processar dados ordenados, forneceu uma lição valiosa sobre a sensibilidade dos algoritmos à natureza dos dados de entrada. Este comportamento ressaltou a importância estratégica do entendimento das métricas de altura e balanceamento para garantir a previsibilidade e a confiabilidade do tempo de resposta em sistemas críticos.

Do ponto de vista técnico e funcional, a robustez do programa na normalização de termos, no tratamento de hífens e na gestão dinâmica de memória para as listas de ocorrências assegurou a entrega de um índice remissivo preciso. A capacidade de tratar cada palavra como uma unidade léxica única, independentemente de sua formatação original, reflete o sucesso da implementação em atender aos requisitos de negócio de um buscador textual eficiente.

Em suma, este trabalho não apenas cumpriu seus objetivos acadêmicos, como também demonstrou que a escolha correta das estruturas de armazenamento é o que define a qualidade técnica de um sistema de informação. Como perspectiva futura, a evolução deste projeto passaria naturalmente pela implementação de árvores auto-balanceáveis, como as árvores AVL ou Rubro-Negras, a fim de mitigar o problema da degeneração observado nos testes de pior caso e assegurar que o sistema mantenha sua alta performance independentemente do perfil de dados inserido pelo usuário.

REFERÊNCIAS

TENENBAUM, Aaron M.; LANGSAM, Yedidyah; AUGENSTEIN, Moshe J. **Estruturas de Dados Usando C**. São Paulo: Makron Books, 1995.

ZIVIANI, Nivio. **Projeto de Algoritmos**: Com Implementações em Pascal e C. São Paulo: Cengage Learning, 2011.

SEDGEWICK, Robert; WAYNE, Kevin. **Algorithms**. 4. ed. Upper Saddle River: Addison-Wesley Professional, 2011.

LAFORE, Robert. **Estruturas de Dados e Algoritmos em Java**. Rio de Janeiro: Ciência Moderna, 2004.

ISO. **ISO/IEC 9899:2011**: Information technology – Programming languages – C. Geneva, CH: International Organization for Standardization, 2011.