Relatório do EP

Indexador e buscador de palavras

Integrantes: Manuela Campos de Amorim (NUSP: 16887191) e Sabrina Cristan (NUSP: 16837632)

Disciplina: Algoritmos e Estruturas de Dados I

1. Visão geral

Com o alto crescimento da produção de conteúdos textuais, ferramentas que auxiliem na indexação e busca de palavras são essenciais. Nesse contexto, esse exercício programa tem como objetivo o desenvolvimento de um sistema que carregue arquivos de texto, processe seu conteúdo e permita que o usuário consulte se um termo está presente na entrada enviada.

Para a consolidação do trabalho, foram implementadas duas abordagens distintas para armazenar o índice de palavras: uma baseada em listas sequenciais e a outra em árvores binárias de busca. Ambos os métodos são funcionais e permitem a verificação da existência de palavras e a recuperação das linhas de ocorrência, mas possuem características de desempenho divergentes.

Além da implementação funcional, esse relatório visa realizar uma análise comparativa. Para tal, foram realizados testes com arquivos de texto de diferentes tamanhos para medir o custo computacional de cada abordagem. O objetivo final é exibir as vantagens e desvantagens do uso de cada estrutura.

2. Metodologia

Para realizar a comparação de desempenho entre as estruturas de lista sequencial e árvore binária de busca, foram submetidas três amostras de texto de diferentes volumes de dados. O primeiro trecho foi extraído do enunciado, o segundo é um capítulo de um livro e o terceiro, um livro inteiro. Os arquivos utilizados estão disponíveis no repositório do projeto.

2.1. Especificações da implementação do programa

O sistema foi projetado para não distinguir letras maiúsculas e minúsculas, operando no modo *case insensitive*. Dessa forma, termos como "coutinho", "COUTINHO" e "Coutinho" são normalizados e indexados como uma única entrada. A insensibilidade às letras maiúsculas e minúsculas só não é garantida em palavras acentuadas. Por exemplo, “Árvore” e “árvore” são consideradas duas palavras distintas. Essa limitação decorre do fato de que a função utilizada para padronizar as palavras em letras minúsculas (`tolower`) segue o padrão da tabela ASCII e implementar a normalização completa de Unicode exigiria a utilização de bibliotecas externas, o que fugiria do escopo do trabalho.

Além disso, foi considerado importante o reconhecimento de palavras acentuadas, visto que o público-alvo da ferramenta consiste majoritariamente em falantes da língua portuguesa. Nesse sentido, o programa é *accent sensitive*, ou seja, palavras com grafias distintas, como “borogodó” e “borogodo”, são tratadas como entradas diferentes no índice, exigindo que o usuário realize buscas com a acentuação correta.

Observação: o programa remove corretamente toda a pontuação padrão ASCII (hifens, pontos, vírgulas, aspas retas). No entanto, devido ao suporte implementado para caracteres UTF-8 (caracteres com acento), certos caracteres de formatação estendida, como aspas curvas ou travessões longos, podem ser interpretados como parte de palavras, já que compartilham a faixa de valores de bytes utilizada por caracteres acentuados. Em versões futuras do projeto essa limitação será contornada, mas, por agora, por se tratar de um caso específico, só é orientado tomar ciência e evitá-lo.

Tabela 1: Informações das amostras

Amostra	Conteúdo textual selecionado	Quantidade de linhas	Total de palavras únicas indexadas
Texto 1	Trecho disponibilizado pelo professor (**curto**).	13	70
Texto 2	Capítulo 1 do livro “Frankenstein ou o prometeu moderno” (**médio**).	163	743
Texto 3	Livro “Orgulho e preconceito” (**longo**).	14.909	7.556
Texto 4	Livro “Moby Dick” (**muito longo**).	22.308	19.596
Texto 5	Lista de frutas com cada letra do alfabeto (**muito curto**).	1	26

3. Resultados obtidos

Visualização de tabelas e gráficos

Para avaliar a eficiência das estruturas de dados implementadas, foram realizados experimentos variando o tamanho da entrada. Utilizou-se o número de comparações entre palavras, conforme solicitado na especificação do projeto, pois esta métrica não é afetada pela velocidade do computador ou por outros programas rodando ao mesmo tempo, oferecendo uma medida precisa da complexidade algorítmica.

3.1. Análise teórica da complexidade temporal

A tabela abaixo exibe a complexidade teórica das duas abordagens. Em relação à construção do índice, nas listas sequenciais, esse processo ocorre de forma quadrática. Exemplificando, em um texto com um total de 50 palavras, sendo todas elas distintas, a cada nova palavra inserida, é necessário realizar a

comparação com as palavras que já foram inseridas antes. Essas comparações realizadas na operação de inserção resultam numa progressão aritmética, então a complexidade temporal de construção pode ser dada por:

$$\frac{(n-1)\cdot n}{2} = \frac{n^2-n}{2} = \Theta(n^2)$$

Por outro lado, na abordagem utilizando árvore binária de busca, a complexidade de construção é muito reduzida. Como a árvore divide o espaço de busca a cada comparação, a altura da estrutura tende a crescer de forma logarítmica (h ≈ log n). Assim, para inserir cada uma das n palavras, o custo médio é proporcional à altura da árvore, resultando em uma complexidade total de construção dada por:

$$\Theta(n \log n)$$

Quanto à operação de busca, em listas sequenciais, no pior dos casos, a palavra procurada é a última indexada ou inexistente, sendo necessário realizar n comparações para encontrar a palavra, resultando em um limite inferior igual a:

$$O(n)$$

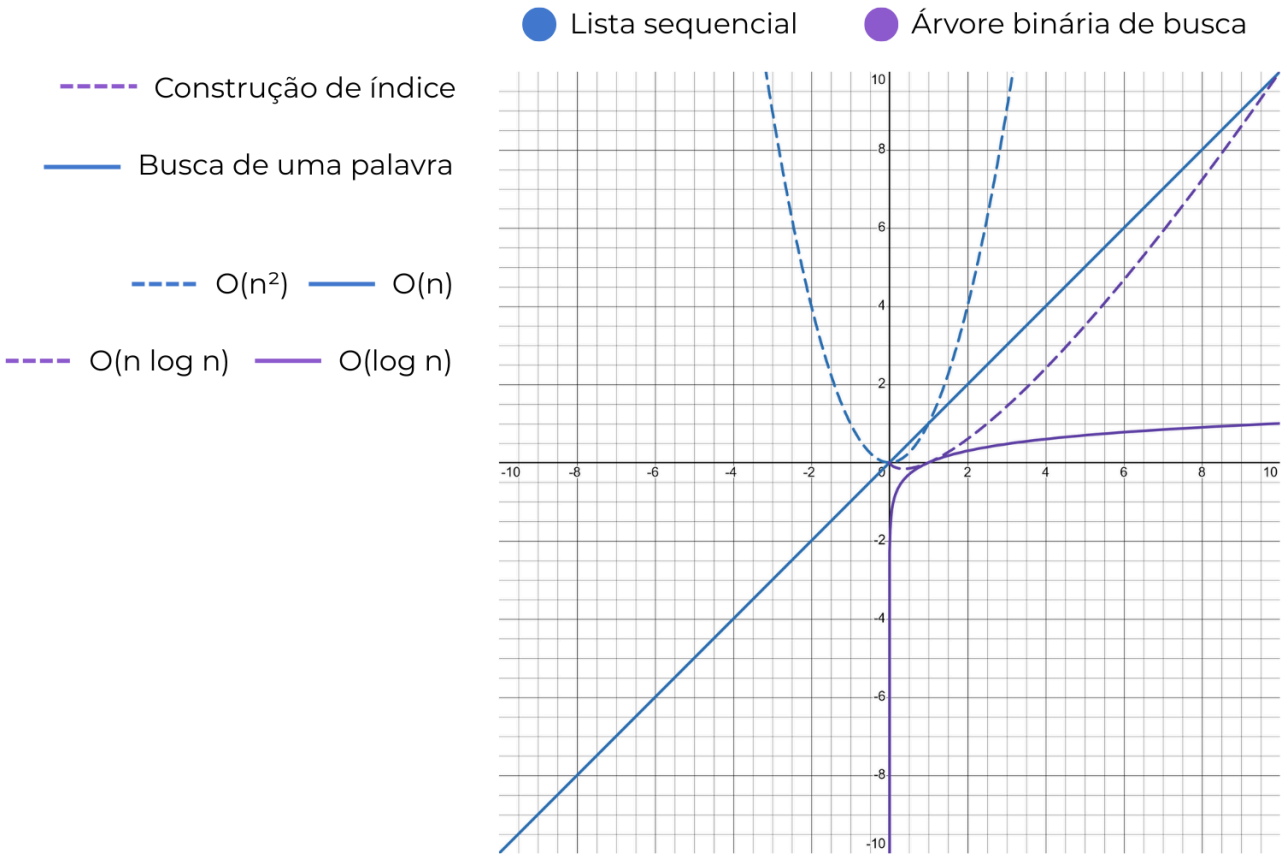
Já em árvores binárias de busca, a procura parte da raiz e desce apenas por um dos ramos, eliminando metade das opções restantes a cada passo. Em uma árvore balanceada, a complexidade é:

$$O(\log n)$$

Tabela 2: Complexidade assintótica das estruturas utilizadas

Estrutura	Construção do índice	Busca de uma palavra
Lista sequencial	O(n²)	O(n)
Árvore binária de busca	O(n log n)	O(log n)

Gráfico 1: Comparação da complexidade assintótica das estruturas abordadas



3.2. Análise dos resultados obtidos

Tabela 3: Desempenho na construção do índice

Amostra	Total de palavras únicas indexadas	Comparações (lista sequencial)	Comparações (árvore binária de busca)	Altura da árvore
Texto 1	70	3.412	691	11
Texto 2	743	394.160	15.047	17
Texto 3	7.556	133.819.206	1.510.606	32
Texto 4	19.596	549.683.440	2.614.063	39
Texto 5	26	325	325	25

Os dados obtidos confirmam a disparidade de eficiência prevista na seção anterior. Nas listas sequenciais, observa-se o comportamento quadrático esperado. Em contrapartida, o crescimento das comparações na árvore manteve-se próximo da linearidade (n log n).

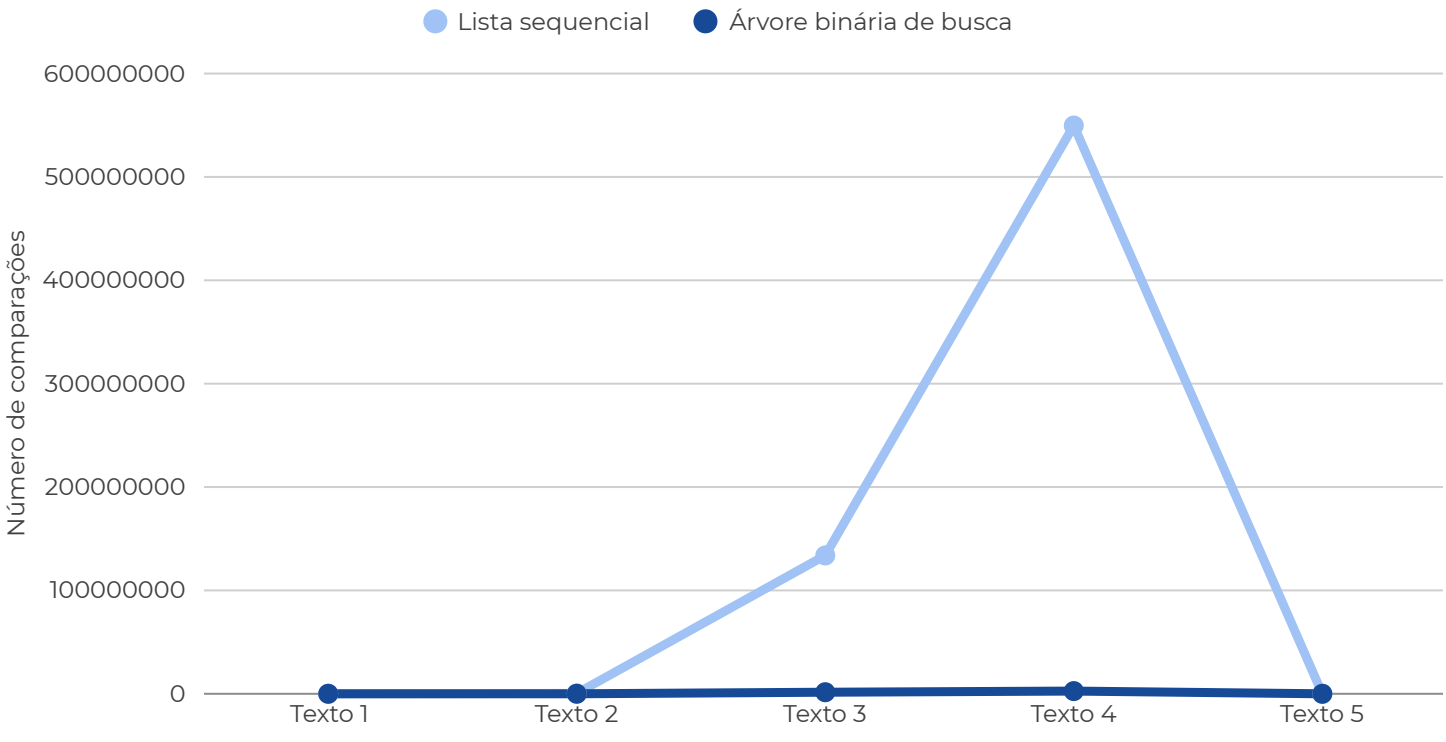
Em relação ao balanceamento de estrutura de árvore binária de busca, na maioria dos casos, a altura da árvore se manteve baixa. Apenas no último exemplo proposital, cujas palavras estavam em ordem alfabética, a árvore tornou-se uma estrutura semelhante à lista sequencial, reduzindo a eficiência.

Tabela 4: Desempenho na busca de palavras no texto 4

Cenário de teste	Palavra buscada	Comparações (lista sequencial)	Comparações (árvore binária de busca)
Melhor caso	the	1	1
Caso médio	prophesied	10.242	12
Outro caso médio	bowditch	9.799	21
Pior caso	newsletter	19.596	18
Palavra inexistente	inexistente	19.596	19

Os resultados explicitam que a eficiência da lista sequencial depende inteiramente da ordem de inserção, enquanto que a árvore demonstra uma performance superior, variando de modo desprezível o número de comparações.

Gráfico 2: Desempenho das estruturas na construção de índice



Nota: devido à grandeza dos valores da lista, a curva da árvore parece linear ou nula, mas na verdade possui crescimento logarítmico.

Observe no gráfico acima a disparidade de escala entre as duas estruturas. Enquanto o número de comparações na lista sequencial cresce de forma opressiva, a curva da árvore binária permanece praticamente constante próxima ao eixo horizontal. Isso evidencia visualmente que o custo de construção da árvore é insignificante se comparado ao esforço computacional exigido pela lista.

4. Conclusão

Os experimentos realizados neste trabalho evidenciaram as acentuadas diferenças práticas entre estruturas de dados lineares e hierárquicas. A partir disso, percebeu-se que uma lista sequencial em contextos de larga escala cresce de maneira explosiva, enquanto que o aumento das comparações em estrutura de árvore é controlado e mais eficiente. Portanto, conclui-se que no cenário de um indexador e/ou buscador de palavras, a estrutura árvore de busca binária se fez mais apta no decorrer dos testes.

4.1. Referências

Notas de aula da disciplina Algoritmos e Estruturas de Dados I. EACH-USP, 2025.

C REFERENCE. *C Standard Library documentation*. Disponível em: https://en.cppreference.com/w/c. Acesso em: dez. 2025.

PROJECT GUTENBERG. Free eBooks - Project Gutenberg. Disponível em: https://www.gutenberg.org. Acesso em: dez. 2025.