

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
ENGENHARIA DE COMPUTAÇÃO

GUSTAVO MACHADO SILVA
YASMIN AGNES SIMAO

AVALIAÇÃO DE DESEMPENHO DE ESTRUTURAS DE DADOS

Disciplina: Estrutura de Dados (INF01203)

Professora: Renata Galante

Porto Alegre, Janeiro de 2025.

Sumário

1 INTRODUÇÃO	3
2 ESTRUTURA DE DADOS E GERAÇÃO DE DADOS	4
2.1 Estruturas de Dados Escolhidas	4
2.1.1 Lista Simplesmente Encadeada	4
2.1.2 Árvore de Pesquisa Binária	4
2.2 Geração de Dados	4
3 METODOLOGIA	6
3.1 Hardware Utilizado	7
3.1.1 MacBook Air M1	7
3.1.2 MacBook Air M2	7
3.2 Execução do Experimento	8
4 ANÁLISE DOS RESULTADOS	10
4.1 Mil Dados	10
4.2 Dez Mil Dados	12
4.3 Cem Mil Dados	14
4.4 Discussão dos resultados	16
5 CONCLUSÃO	18
REFERÊNCIAS	19

1 INTRODUÇÃO

O objetivo deste trabalho foi comparar experimentalmente duas estruturas de dados com o intuito de avaliar seu desempenho na execução de uma aplicação. A aplicação em questão é do tipo usuário-senha, a partir de uma entrada de um usuário e uma senha, o programa procura na estrutura de dados aquele usuário e confere se a relação usuário-senha está correta. Foram utilizadas diferentes quantidades de dados para também avaliar como ela afeta a eficiência de cada estrutura em resolver o problema.

Desta forma, buscamos compreender como a escolha da estrutura de dados afeta o funcionamento de um programa e o quão grande é a importância desta escolha na hora de desenvolver uma aplicação.

Todos os códigos e documentos deste trabalho podem ser encontrados no seguinte repositório: <https://github.com/gustavomachadosilva/final-project-data-structure>.

2 ESTRUTURA DE DADOS E GERAÇÃO DE DADOS

As estruturas foram escolhidas de acordo com a preferência dos autores levando em consideração o caso mais interessante de análise. Os dados para essa análise foram criados pelos autores através de um método que será explicado no decorrer deste relatório.

2.1 Estruturas de Dados Escolhidas

Com a intenção de explorar dois grupos de estruturas de dados diferentes, optamos por utilizar uma lista e uma árvore na execução da análise. A motivação dessa escolha foi a de poder comparar o ganho de desempenho que é tido ao utilizarmos uma árvore em comparação com uma lista e como isso pode afetar a execução do problema proposto.

2.1.1 Lista Simplesmente Encadeada

A lista simplesmente encadeada tem como característica a o encadeamento de seus elementos, ou seja, cada elemento possui um ponteiro em sua estrutura que aponta para o elemento seguinte. Essa estrutura de dados é de grande utilidade em casos onde é necessário inserir elementos no meio lista ou quando temos um número de elementos não previsível previamente.

2.1.2 Árvore de Pesquisa Binária

A árvore de pesquisa binária, ou ABP, é formada por uma estrutura de dados com dois ponteiros, um para esquerda e outro para direita e uma chave, ela é organizada de forma que, para qualquer nodo, todos os nodos a sua esquerda tem chaves menores que a dele e todos a direita chaves maiores. Dessa forma as consultas aos valores são realizadas como busca binária, um método muito eficiente, entretanto a rapidez da procura depende da ordem de inserção dos nodos, uma vez que a árvore não possui nenhum método de balanceamento. Operações de inserir e remover também são bastante eficientes nesta implementação.

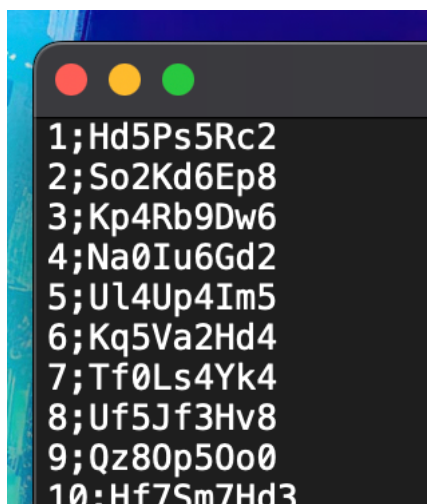
2.2 Geração de Dados

Para a geração de dados, os autores optaram por elaborar o próprio método de geração de senhas para um melhor aproveitamento dos dados. Para isso foi criado um projeto em C chamado "dataGenerator" que está disponível no repositório deste trabalho. Este gerador de dados gera cinco por vez com a quantidade de dados solicitada pelo usuário.

- Um arquivo com os dados ordenados para alimentar a estrutura de dados.
- Um arquivo com os dados desordenados para alimentar a estrutura de dados.
- Um arquivo com os dados desordenados para servir como consulta.
- Um arquivo com os dados desordenados e 50% dos valores incorretos para servir como consulta.
- Um arquivo com os dados desordenados e 80% dos valores incorretos para servir como consulta.

Estes cinco arquivos gerados foram planejados para serem aplicados no código de análise da estrutura de dados de acordo com o funcionamento do código. O padrão de senha utilizado nesses arquivos segue três conjuntos de uma letra maiúscula, uma letra minúscula e um número. A figura 1, a seguir, demonstra como os dados ficam no arquivo.

**Figura 1 - Exemplo de
Arquivo de Dados**



```
1;Hd5Ps5Rc2
2;So2Kd6Ep8
3;Kp4Rb9Dw6
4;Na0Iu6Gd2
5;U14Up4Im5
6;Kq5Va2Hd4
7;Tf0Ls4Yk4
8;Uf5Jf3Hv8
9;Qz80p50o0
10:Hf7Sm7Hd3
```

Fonte: Os autores.

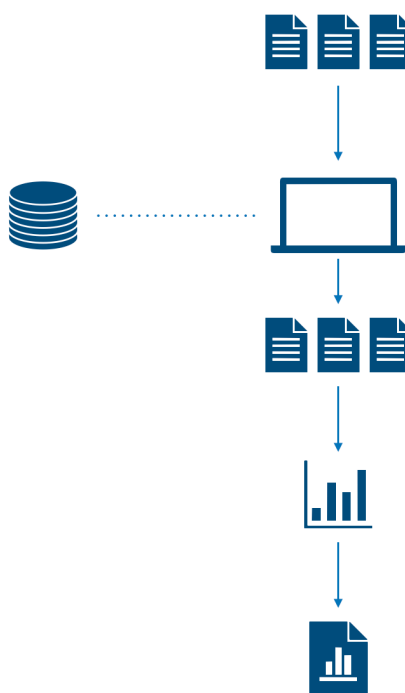
A Figura 1 mostra o padrão de arquivo de dados utilizado na execução do trabalho onde vemos que para cada linha temos um identificador e uma senha separados pelo caractere ‘;’.

3 METODOLOGIA

Para realizar o experimento foram escritos programas para teste na linguagem C utilizando a IDE Xcode, disponível para macOS. Foram feitos 2 projetos, um para cada estrutura avaliada, contendo a implementação da estrutura de dados, a implementação da aplicação usuário-senha junto com a geração de arquivos com os resultados das análises e o main.c, com a alimentação das estruturas e a chamada da aplicação. Em seguida foi realizado o agrupamento dos dados recolhidos para a avaliação pelos autores. Todos os códigos estão disponíveis no repositório deste trabalho.

O processo deste experimento consistiu em um processo de entrada e saída de arquivos que consistiam em arquivos de consulta, arquivos de resultados e arquivos de alimentação. A Figura 2, a seguir demonstra como foi pensado este processo.

Figura 2 - Esquema de Funcionamento do Experimento



Fonte: Os autores.

A Figura 2 mostra que o experimento foi realizado com o sistema recebendo os arquivos de consulta e consultando os valores na base de dados. Após isso, o sistema gerou os arquivos de saída e a partir desses arquivos e utilizando uma ferramenta de análise de dados como o Numbers foi possível gerar os gráficos e tabelas.

3.1 Hardware Utilizado

Para este trabalho, foram utilizados dois computadores para a o desenvolvimento e execução do experimento. Ambos computadores utilizando o sistema operacional MacOS porém com hardware diferentes. As especificações técnicas de ambos computadores podem ser vistas a seguir. No primeiro foram realizados os testes com a implementação de lista encadeada e no segundo com a ABP.

3.1.1 MacBook Air M1

O primeiro computador foi um MacBook Air M1 de 8GB modelo de 2020, utilizando o macOS Sequoia versão 15.2. As informações técnicas do processador podem ser visualizadas na figura 3 a seguir.

Figura 3 - Especificações Técnicas Chip M1

Chip	<ul style="list-style-type: none">• M1 da Apple<ul style="list-style-type: none">◦ CPU de 8 núcleos (4 de desempenho e 4 de eficiência)◦ GPU de 7 núcleos, GPU de 8 núcleos◦ Neural Engine de 16 núcleos
-------------	--

Fonte: Apple.

3.1.2 MacBook Air M2

O segundo computador foi um MacBook Air M2 de 8GB modelo de 2022, utilizando o macOS Sonoma versão 14.4. As informações técnicas do processador podem ser visualizadas na figura 4 a seguir.

Figura 4 - Especificações Técnicas Chip M2

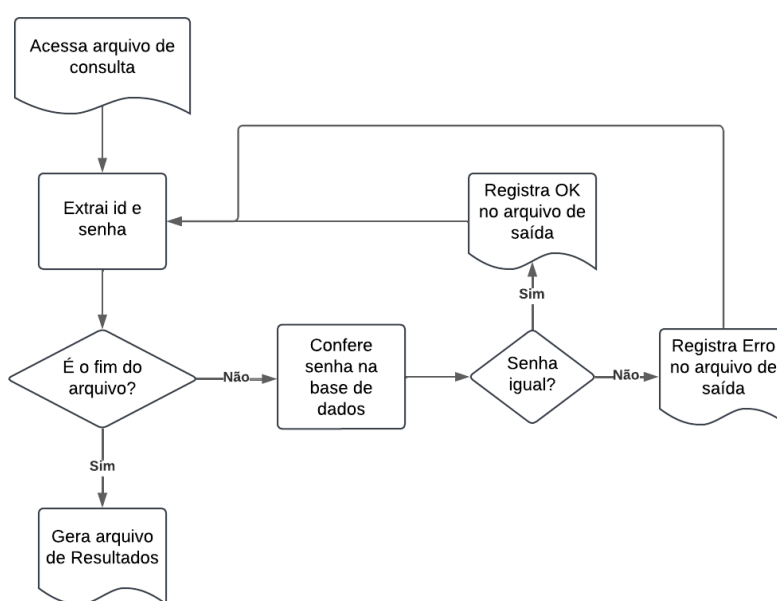
Chip	M2 da Apple <ul style="list-style-type: none">CPU de 8 núcleos (4 de desempenho e 4 de eficiência)GPU de 8 núcleosNeural Engine de 16 núcleos
-------------	--

Fonte: Apple.

3.2 Execução do Experimento

A execução deste experimento ocorreu com a criação de um programa em C para ser aplicado no processo apresentado anteriormente. Foi importante planejar como esse programa deveria se comportar e o que deveríamos esperar da sua execução. A Figura 5, a seguir, demonstra o comportamento do programa.

Figura 5 - Comportamento do Programa



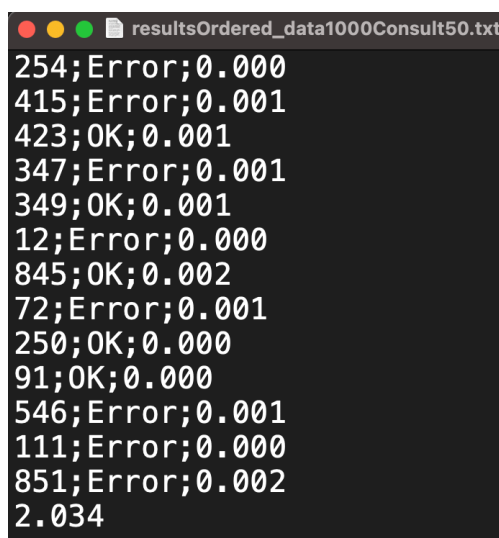
Fonte: Os autores.

O programa teste de cada estrutura de dados foi executado três vezes, uma para cada quantidade de dados avaliada, são elas, mil, dez mil e cem mil. Em cada execução a função main alimenta duas estruturas, uma com o arquivo de dados ordenado e outra com o não ordenado, faz isso utilizando as funções de manipulação de arquivos disponíveis na biblioteca stdio.h e a implementação das estruturas escritas pelos autores.

Após isso, chama a função “AnalyzesPassword” seis vezes, cada arquivo de consulta para as duas estruturas geradas anteriormente. Esta função cria um arquivo do tipo .txt e lê o arquivo de consulta passado como parâmetro procurando cada combinação usuário-senha na estrutura, também passada como parâmetro. Então escreve no arquivo criado se foi possível encontrar a combinação e quanto tempo, em milissegundos, foi necessário, escreve também, no final do arquivo, o tempo

total da análise do arquivo de consulta completo. O arquivo gerado com os resultados está organizado da maneira exposta na figura 6. Os outros arquivos citados estão especificados na seção 2.2 deste trabalho.

Figura 6 - Exemplo de Arquivo de Resultados



```
resultsOrdered_data1000Consult50.txt
254;Error;0.000
415;Error;0.001
423;OK;0.001
347;Error;0.001
349;OK;0.001
12;Error;0.000
845;OK;0.002
72;Error;0.001
250;OK;0.000
91;OK;0.000
546;Error;0.001
111;Error;0.000
851;Error;0.002
2.034
```

Fonte: Os autores.

4 ANÁLISE DOS RESULTADOS

Para analisar os resultados gerados pelo programa criado, utilizamos o software Numbers com o objetivo de gerar tabelas e gráficos que tivessem relevância para o estudo proposto. Separando os dados em os que foram consultados em uma estrutura ordenada e os que foram consultados em uma estrutura não ordenada, conseguimos obter informações relevantes para o experimento.

4.1 Mil Dados

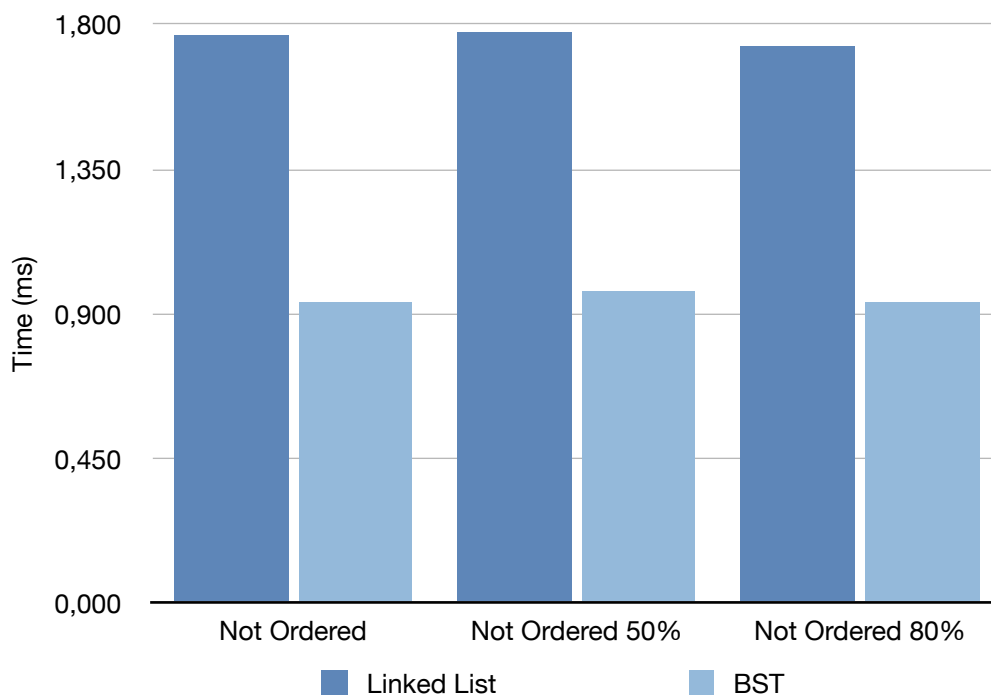
Começando pela menor quantidade de dados utilizada na execução deste experimento, para essa quantidade de dados foi possível ter uma primeira ideia da comparação das estrutura de dados. A Figura 7, a seguir, representa uma tabela com o total de tempo (milissegundos) que se levou para fazer cada consulta na respectiva estrutura de dados.

Figura 7 - 1.000 Dados - Tempo Total Por Consulta (ms)

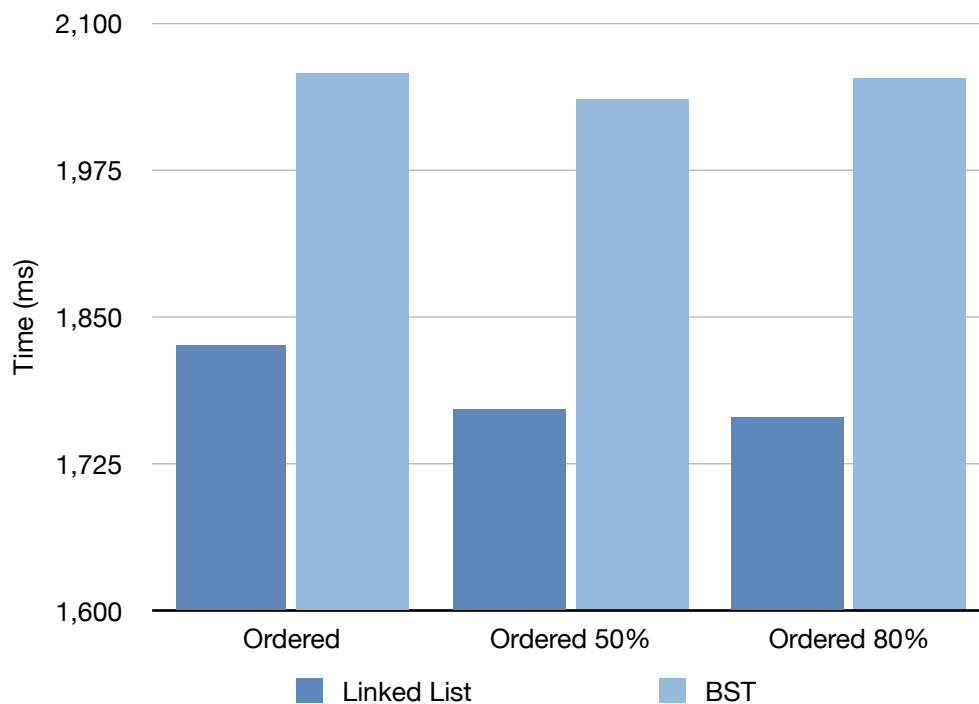
Type of Query	Linked List	BST
Not Ordered	1,768	0,933
Not Ordered 50%	1,774	0,968
Not Ordered 80%	1,735	0,935
Ordered	1,825	2,058
Ordered 50%	1,771	2,034
Ordered 80%	1,765	2,054

Fonte: Os autores.

Através dos dados apresentados na Figura 7 foi possível extrair gráficos separando o comportamento das estruturas para uma base de dados ordenada e uma base não ordenada. As figuras 8 e 9, a seguir, mostram esses comportamentos.

Figura 8 - 1.000 Dados - Consulta Não Ordenada

Fonte: Os autores.

Figura 9 - 1.000 Dados - Consulta Ordenada

Fonte: Os autores.

Nas figuras 8 e 9 podemos ver um comportamento interessante em relação ao tempo de execução das consultas. Na consulta não ordenada podemos ver que a Arvore de Pesquisa Binária possui uma vantagem de quase metade do tempo de execução. Já na consulta ordenada podemos ver que a vantagem no tempo de execução fica com a Lista Encadeada.

4.2 Dez Mil Dados

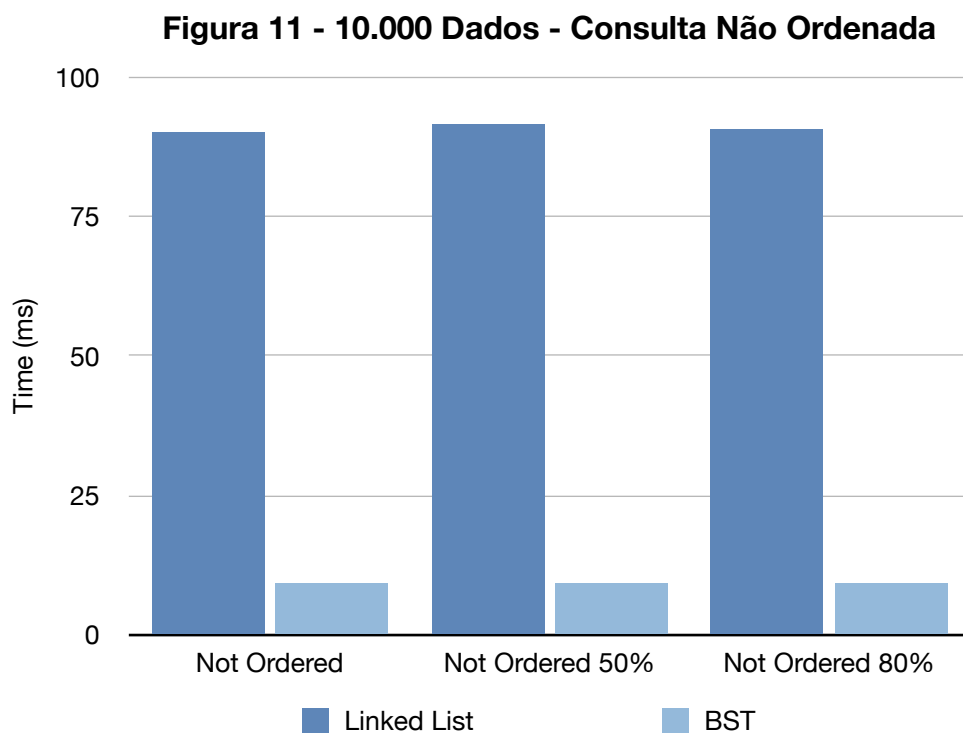
Passando agora pra uma quantidade de dados mais significativa utilizada na execução deste experimento, para essa quantidade de dados foi possível observar um comportamento mais acentuado das estruturas. A Figura 10, a seguir, representa uma tabela com o total de tempo (milissegundos) que se levou para fazer cada consulta na respectiva estrutura de dados.

Figura 10 - 10.000 Dados - Tempo Total Por Consulta (ms)

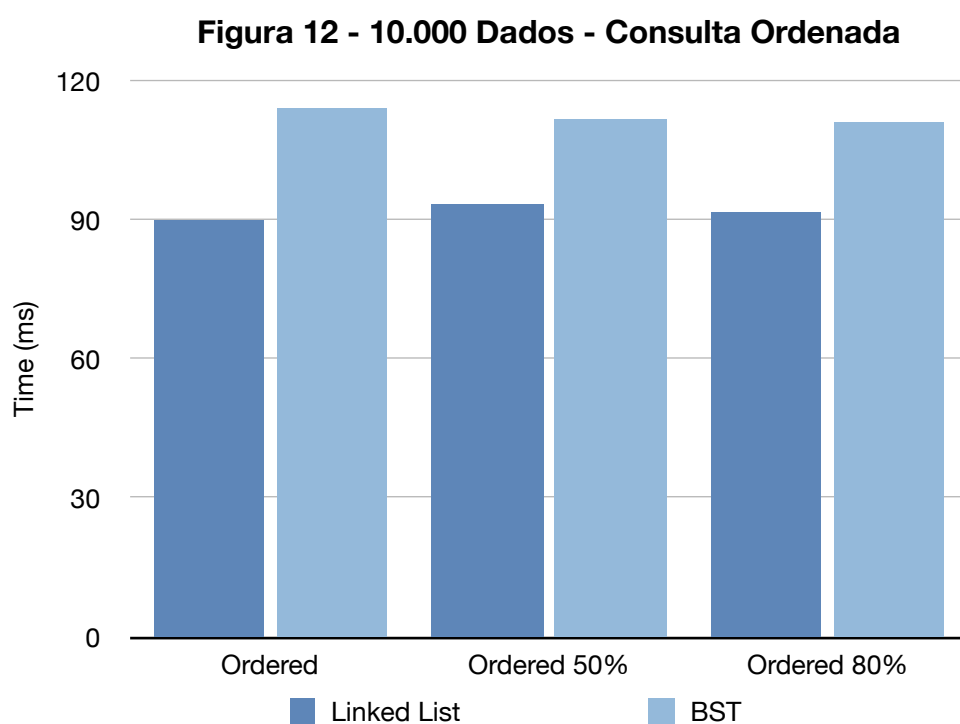
Type of Query	Linked List	BST
Not Ordered	90,231	9,428
Not Ordered 50%	91,325	9,447
Not Ordered 80%	90,569	9,371
Ordered	89,844	114,329
Ordered 50%	93,107	111,580
Ordered 80%	91,645	111,040

Fonte: Os autores.

Através dos dados apresentados na Figura 10 foi possível extrair gráficos separando o comportamento das estruturas para uma base de dados ordenada e uma base não ordenada. As figuras 11 e 12, a seguir, mostram esses comportamentos.



Fonte: Os autores.



Fonte: Os autores.

Nas figuras 11 e 12 podemos ver um comportamento interessante em relação ao tempo de execução das consultas e os valores apresentados para a quantidade de dados anterior. Na consulta

não ordenada podemos ver que a Arvore de Pesquisa Binária possui uma vantagem ainda maior do que a apresentada anteriormente. Já na consulta ordenada, a Lista Encadeada ainda apresenta uma vantagem no tempo de execução mas não tão destacável quanto anteriormente.

4.3 Cem Mil Dados

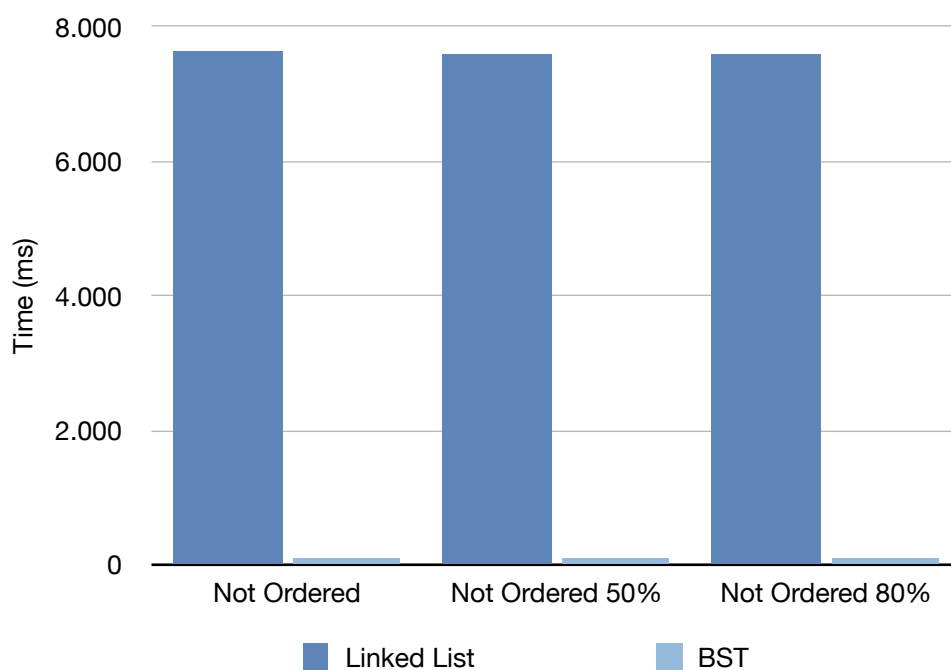
Por fim, analisamos a maior quantidade de dados utilizada na execução deste experimento, para essa quantidade de dados foi possível observar um comportamento ainda mais acentuado das estruturas. A Figura 13, a seguir, representa uma tabela com o total de tempo (milissegundos) que se levou para fazer cada consulta na respectiva estrutura de dados.

Figura 13 - 100.000 Dados - Tempo Total Por Consulta (ms)

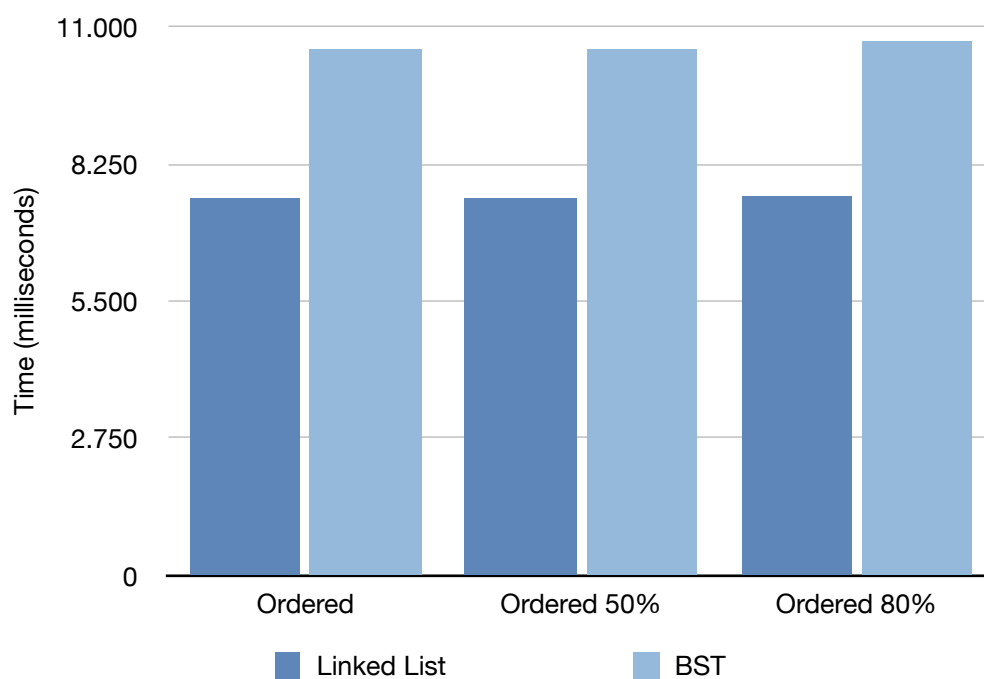
Type of Query	Linked List	BST
Not Ordered	7.636,569	110,338
Not Ordered 50%	7.599,255	107,795
Not Ordered 80%	7.618,001	112,976
Ordered	7.570,383	10.537,344
Ordered 50%	7.583,518	10.570,305
Ordered 80%	7.615,940	10.725,881

Fonte: Os autores.

Através dos dados apresentados na Figura 13 foi possível extrair gráficos separando o comportamento das estruturas para uma base de dados ordenada e uma base não ordenada. As figuras 14 e 15, a seguir, mostram esses comportamentos.

Figura 14 - 100.000 Dados - Consulta Não Ordenada

Fonte: Os autores.

Figura 15 - 100.000 Dados - Consulta Ordenada

Fonte: Os autores.

Nas figuras 14 e 15 podemos ver um comportamento interessante em relação ao tempo de execução das consultas e os valores apresentados para as quantidades de dados anteriores. Na consulta não ordenada podemos ver que a Arvore de Pesquisa Binária possui uma vantagem muito mais significativa do que as amostragem anteriores. Já na consulta ordenada, a Lista Encadeada apresentou uma vantagem parecida com a apresentada anteriormente.

4.4 Discussão dos resultados

A partir dos gráficos apresentados é possível concluir que a estrutura mais eficiente depende da ordenação ou não ordenação dos dados. Isso se deve ao modo no qual as estruturas são construídas em memória ao longo da inserção dos dados, de maneira que, a ABP fica com uma estrutura similar à lista encadeada quando os dados são inseridos em ordem crescente.

Para a entrada não ordenada vemos uma vantagem muito grande da árvore de pesquisa binária, causada por seu método de consulta, enquanto a lista encadeada utiliza um algoritmo de busca linear, isto é, percorre todos os nodos até encontrar o procurado e a maior quantidade de comparações possíveis é o tamanho total da lista, a ABP utiliza a busca binária, que tem a quantidade máxima de comparações definidas pela altura máxima da árvore. Com os dados não ordenados, a medida que aumentamos sua quantidade a diferença entre o tamanho linear da lista e a altura da árvore é cada vez maior e mais relevante nos resultados.

Para a entrada ordenada temos que, o tamanho da lista e a altura da árvore são iguais, deste modo a ABP não apresenta mais a vantagem observada nas entradas não ordenadas. Além disso é observado que a lista encadeada tem um desempenho melhor que a ABP mesmo estando construídas da mesma forma. Essa diferença é explicada pela implementação das funções de consulta de cada estrutura e a quantidade de comparações que cada uma realiza. As figuras 16 e 17 mostram a função de consulta de cada implementação.

Figura 16 - Consulta da Lista Encadeada

```

37 Password* findPasswordById>Password* list, int id) {
38
39     Password *currentPassword = list;
40
41     while (currentPassword != NULL && currentPassword->id != id) {
42         currentPassword = currentPassword->next;
43     }
44
45     if (currentPassword == NULL) {
46         return NULL;
47     }
48     else {
49         return currentPassword;
50     }
51
52 }

```

Fonte: Os autores.

Figura 17 - Consulta da ABP

```

35 Password* findPasswordByKey>Password* tree, int key){
36     while (tree != NULL){
37         if (tree->key == key ){
38             return tree;
39         }
40         else{
41             if (tree->key > key)
42                 tree = tree->left;
43             else
44                 tree = tree->right;
45         }
46     }
47     return NULL; //not found
48 }

```

Fonte: Os autores.

Percebemos que, enquanto a primeira implementação apenas compara se o id procurado é o mesmo daquele nodo, a segunda também compara se ele é maior que o anterior, para seguir o algoritmo de busca binária. Com os dados não ordenados esta diferença é irrelevante frente ao método de busca utilizado, entretanto quando a altura da árvore é igual ao tamanho da lista esta comparação extra a cada execução do lado while é perceptível na análise do desempenho pelo tempo.

5 CONCLUSÃO

Estruturas de dados são parte importante do desenvolvimento de aplicações em computação, e neste trabalho observamos que a escolha de uma estrutura que se adapte às exigências da aplicação é relevante quando discutimos eficiência de execução.

A escolha da melhor estrutura de dados para a aplicação não deve ser baseada somente no tipo de aplicação mas também no tamanho da aplicação. Entender o propósito da aplicação e qual será a sua escala é o fator essencial para definir o melhor caminho para o projeto.

REFERÊNCIAS

APPLE. Support, 2025. Página de suporte. Disponível em: < <https://support.apple.com/pt-br/111883>>. Acesso em janeiro de 2025.

APPLE. Support, 2025. Página de suporte. Disponível em: <<https://support.apple.com/pt-br/111867>>. Acesso em janeiro de 2025.