Análise de Desempenho de Lista Simplesmente Encadeada e Árvore Binária de Pesquisa

Autor: Pedro Schuck de Azevedo
Componente Curricular: Estrutura de Dados
Docente: Renata de Matos Galante

Porto Alegre,
Dezembro de 2024

Introdução

Ao longo das últimas décadas, diferentes tipos de estruturas de dados foram criados, com os mais variados propósitos, porém sempre priorizando eficiência, velocidade e aplicabilidade. Tais estruturas contribuíram para a evolução da computação, otimizando e facilitando processos, pesquisas e programas que tornaram mais simples não só tarefas cotidianas para o grande público, como também problemas complexos para aqueles que trabalham e estudam na área. Entretanto, para alcançar os melhores resultados em cada uso, torna-se necessário analisar as múltiplas diferenças entre cada tipo de estrutura, definindo o contexto mais apropriado para empregar cada uma. Com este objetivo em mente, propõem-se a comparação entre uma Lista Simplesmente Encadeada (LSE) e uma Árvore Binária de Pesquisa (ABP) no tratamento de uma base de dados do tipo usuário-senha.

Estrutura de dados e geração de dados

As estruturas de dados escolhidas têm entre si a semelhança de utilizarem ponteiros para conectar seus elementos e serem capazes de alocar memória dinamicamente, sendo estes os únicos aspectos comuns a ambas. Enquanto que uma LSE une seus nodos em forma de lista, onde cada um apresenta uma ligação somente com o próximo nodo, acessando-os sempre sequencialmente, uma ABP possui nós hierarquizados, bem como uma raiz cuja ela própria e seus descendentes seguem a regra de ter no máximo dois filhos e caso um valor, denominado de chave, de um filho for maior que o de seu pai, este será posicionado à direita, sendo válido o inverso para posicionamentos à esquerda, percorrendo-os através da busca binária. Por isso, uma ABP possui dois ponteiros, um para o filho à esquerda e outro para à direita, em contraste com uma LSE, que apresenta apenas um ponteiro para o elemento subsequente da lista. Tendo em mente todas essas diferenças, estas estruturas foram as escolhidas para serem comparadas, pela crença de que tamanha disparidade gere resultados explícitos e variados.

Para gerar os dados usuário-senha foi feita uma função para criar um arquivo de texto contendo em cada linha um número seguido de uma senha composta de letras e algarismos gerados aleatoriamente. Dessa forma, obteve-se três arquivos, um de mil linhas, outro de dez mil linhas e o terceiro com cem mil linhas, todos em ordem crescente de número do usuário. Para produzir os arquivos desordenados, utilizou-se de um site para embaralhar as linhas dos três arquivos, salvando o resultado em outros novos arquivos. Depois, para fabricar conjuntos de senhas com 50% e 80% de erros, outro site foi usado para trocar todas as letras de algumas das senhas pelo carácter “?”, tornando-as incorretas perante a base de dados original. Assim, valendo-se de uma função escrita em C e dois sites, conseguiu-se adquirir todos os dados necessários para a análise geral entre as duas estruturas.

Metodologia

O equipamento empregado para a realização das comparações foi um computador com 4 Gigabytes de memória RAM e um processador Intel Pentium N3530 em conjunto com o sistema operacional Windows 10 e o ambiente de desenvolvimento integrado (IDE) CodeBlocks para manipular o código. Além disso, foram utilizados os sites “unitarios.com” e “browserling.com” para auxiliar na geração dos dados, bem como as estruturas de dados disponíveis no moodle para as implementações e o site “charts.livegap.com” para criar os gráficos. Para realizar esta análise, o seguinte algoritmo foi usado:

1. Selecionar estruturas de dados;
2. Encontrar implementações para cada uma;
3. Criar um novo projeto no CodeBlocks;
4. Transferir as implementações para o novo projeto;
5. Fazer alterações para que elas suportem o tipo de dado usuário-senha;
6. Escrever o código para leitura de arquivos txt;
7. Posicionar variáveis para contabilizar o número de comparações no código e o tempo de inserção e consulta dos dados;
8. Criar o código para gerar arquivos de texto contendo as bases de dados em ordem;
9. Obter as bases de dados ordenadas;
10. Utilizar um site para embaralhar a ordem e produzir as bases de dados desordenadas;
11. Selecionar uma parte das bases de dados obtidas;
12. Alterar 50% e 80% das senhas destas partes usando um site de troca de caracteres;
13. Salvar os dados obtidos em novos arquivos;
14. Rodar o programa com as bases de dados adquiridas anteriormente;
15. Anotar os resultados observados;
16. Ilustrar os resultados por meio de tabelas e gráficos;
17. Redigir as conclusões tidas através dos resultados.

Análise dos resultados

Para analisar as duas estruturas interagindo com as diferentes bases de dados, foram aplicadas duas medidas principais, a do número de comparações que ocorrem na consulta e o tempo em segundos que levou para executar as operações de inserir e consultar nas estruturas de dados. Em relação ao número de comparações, um ponteiro foi passado como parâmetro das funções de consulta e, dentro delas, ele era incrementado em uma unidade logo após quaisquer comparações, sendo estas realizadas principalmente pelas instruções if e while. Em contrapartida, conseguiu-se o tempo executando a função “clock” antes e depois dos loops responsáveis pela inserção e consulta de dados nas estruturas, armazenando estes valores, subtraindo os finais do iniciais e dividindo-os pela constante “clocks_per_second” para resultar no tempo em segundos gasto pelas duas operações. A seguir encontram-se tabelas que reúnem o conjunto de informações encontradas a partir do testes em ambas estruturas com cada base de dados e base de usuários.

- Tempo em segundos (inserção e consulta):

ABP	1.000 dados	10.000 dados	100.000 dados
Desordenada 50%	0,015 e 0	0,093 e 0,016	1,140 e 0,079
Desordenada 80%	0,016 e 0	0,062 e 0,031	0,625 e 0,078
Ordenada 50%	0,015 e 0	0,672 e 0,125	203,742 e 23,827
Ordenada 80%	0,016 e 0	0,640 e 0,110	200,626 e 24,503

LSE	1.000 dados	10.000 dados	100.000 dados
Desordenada 50%	0,001 e 0	0,384 e 0,134	193,619 e 41,311
Desordenada 80%	0,002 e 0	0,385 e 0,141	190,406 e 38,519
Ordenada 50%	0,012 e 0	0,367 e 0,084	183,985 e 21,455
Ordenada 80%	0,016 e 0	0,402 e 0,093	184,201 e 20,693

- Número de comparações:

ABP	1.000 dados	10.000 dados	100.000 dados
Desordenada	17.209	63.281	846.815
Ordenada	722.407	15.214.487	1.515.334.856

LSE	1.000 dados	10.000 dados	100.000 dados
Desordenada	251.000	19.002.000	1.900.020.000
Ordenada	481.438	10.142.658	1.010.219.904

A partir destas informações reunidas, pode-se perceber certos padrões e comportamentos que, em geral, podem ser explicados por meio da análise de como cada estrutura é. Por exemplo, o número de comparações significativamente maior em ABP com base de dados ordenada, se comparado com a desordenada (pode-se visualizar esta diferenças nas imagens 1, 3 e 5 do anexo), deve-se ao fato de que ao inserir elementos em ordem crescente em uma ABP, que não possui nenhum mecanismo de rotação para alterar seu balanceamento, ela tornar-se-á uma árvore totalmente pendente para a esquerda, sem um nodo sequer na direita, e com altura igual ao número de dados nela inserido, assemelhando-se muito a uma lista, o que também corrobora para que o número de comparações em uma ABP ordenada esteja mais próximo dos valores relativos às comparações em LSE.

Outro fator presente, é o de que na base de mil dados, a ABP é responsável por praticamente metade do total de comparações (imagem 2), porém, nas bases de dez e cem mil dados, a LSE passa a gerar mais da metade das comparações (imagens 4 e 6). Isso ocorre, pois, ainda que a ABP seja fortemente prejudicada, no quesito de comparações, por dados ordenados, ela lida consideravelmente melhor com os desordenados, ao passo que a LSE mantém sempre a mesma faixa de quantias altas em termos de comparação, sendo prejudicada na mesma medida em ambos os casos, e portanto, ultrapassando a metade do total de comparações conforme a base de dados aumenta. Nota-se também, que enquanto os valores das bases de dados crescem sendo um 10 vezes maior que o antecessor, os números de comparação da ABP ordenada e LSE crescem sendo o próximo quase que cem vezes maior que o da base anterior, passando da exorbitante marca de 1 bilhão na base de cem mil dados, o que evidencia que a busca em uma lista, ou algo parecida com uma, como no caso de uma árvore pendente somente para um lado, é extremamente custosa, em especial quanto maior for o volume de dados a ser tratado, afinal, neste processo é necessário comparar os elementos um a um até encontrar o desejado.

De maneira complementar, é perceptível que com o avançar das bases de dados, a ABP deixa de consumir mais da metade do tempo total das operações de inserção e consulta, cedendo este cargo para a LSE (imagens 7, 8 e 9). Isso se deve pela rapidez com que se pode inserir e consultar dados em uma ABP, já que ambos processos envolvem a busca binária, na qual, dentre dois nodos escolhe-se aquele com a chave que levará o mais próximo do nodo procurado, assim evitando ter que percorrer todos os elementos da estrutura, diferentemente do que acontece em listas com seu lento procedimento de busca sequencial. Por mais que na quantidade de comparações não haja diferença entre a base de testes com 50% e

80% de senhas erradas, porque independente da validade da senha, ela precisa ser verificada, no tempo empregado na consulta é possível ver uma leve diferença, que provavelmente deve ser fruto de alguma pequena instabilidade do computador no momento de execução do programa.

Adicionalmente, constata-se que a consulta é inicialmente tão veloz que a função clock da linguagem de programação C não consegue perceber a variação de tempo que ela exerce. E ainda que esta variação passe a ser percebida nas bases de dados maiores, é notável a característica de que ela sempre é significativamente menor que o tempo usado pela inserção (imagens 10, 11 e 12). Este fator não é somente explicado pelas bases de dados serem sempre maiores que as de teste, que possuem 500, mil e dez mil usuários, mas também porque na inserção sempre é preciso percorrer toda a estrutura até chegar a um ponteiro nulo para que o novo elemento possa ser inserido, sendo este um processo simples, porém que em grande escala, para volumosos conjuntos de dados, acaba por tornar-se demasiado lento em comparação com a consulta, em que há uma boa chance do elemento procurado ser encontrado antes de se chegar ao final da estrutura.

Anexos

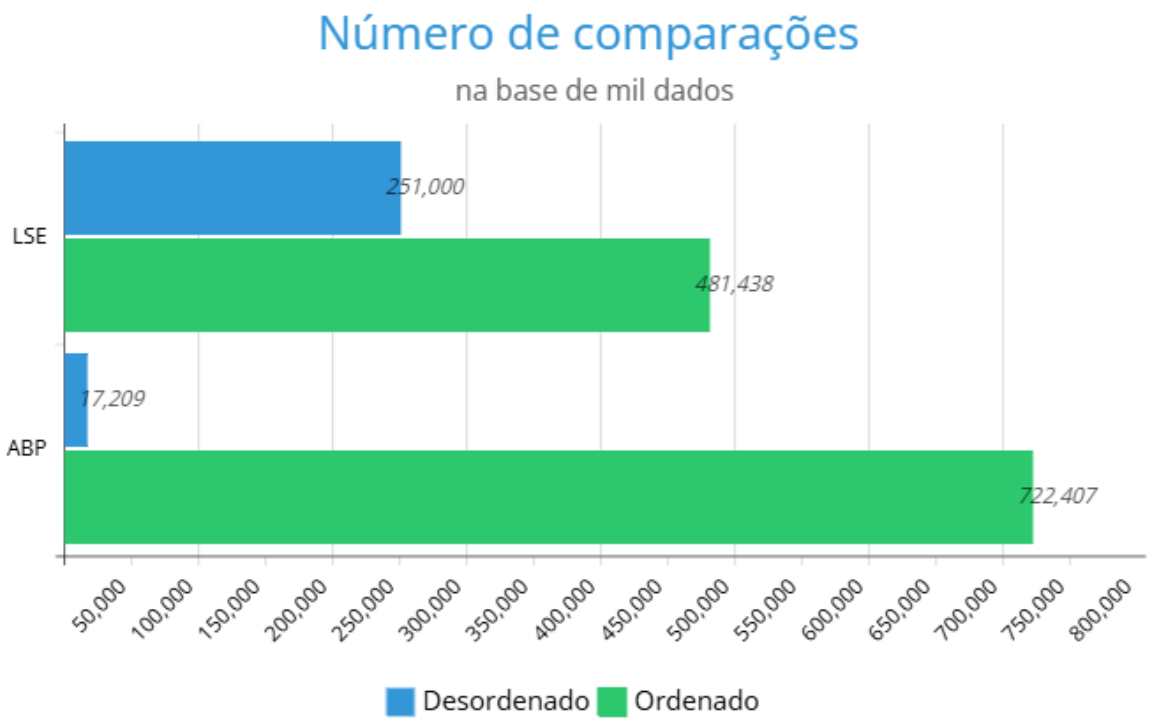

Imagem 1. Gráfico de barras sobre o número de comparações entre ABP e LSE na base de mil dados, desordenada e ordenada.

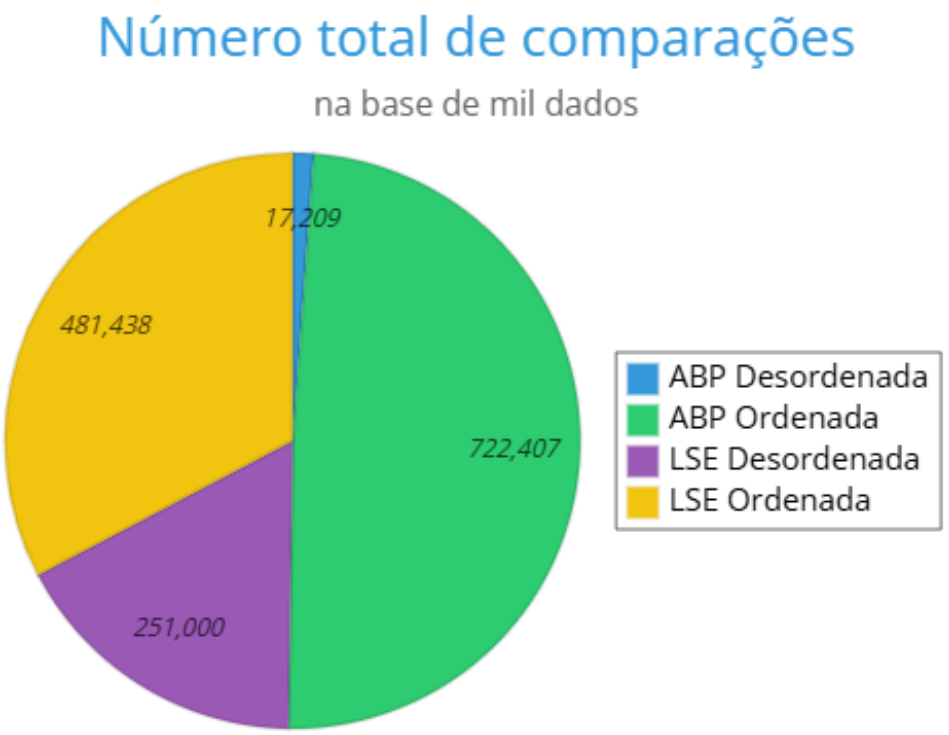

Imagem 2. Gráfico de setores sobre o número total de comparações entre ABP e LSE na base de mil dados, desordenada e ordenada.

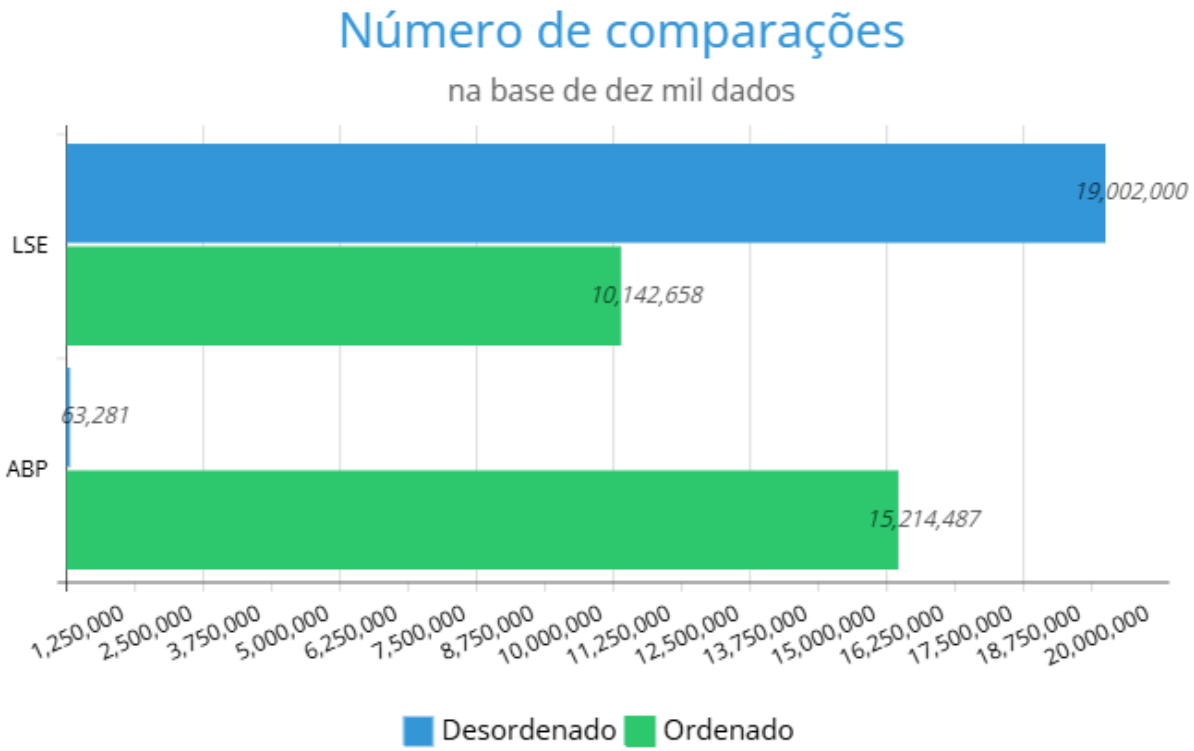

Imagem 3. Gráfico de barras sobre o número de comparações entre ABP e LSE na base de dez mil dados, desordenada e ordenada.

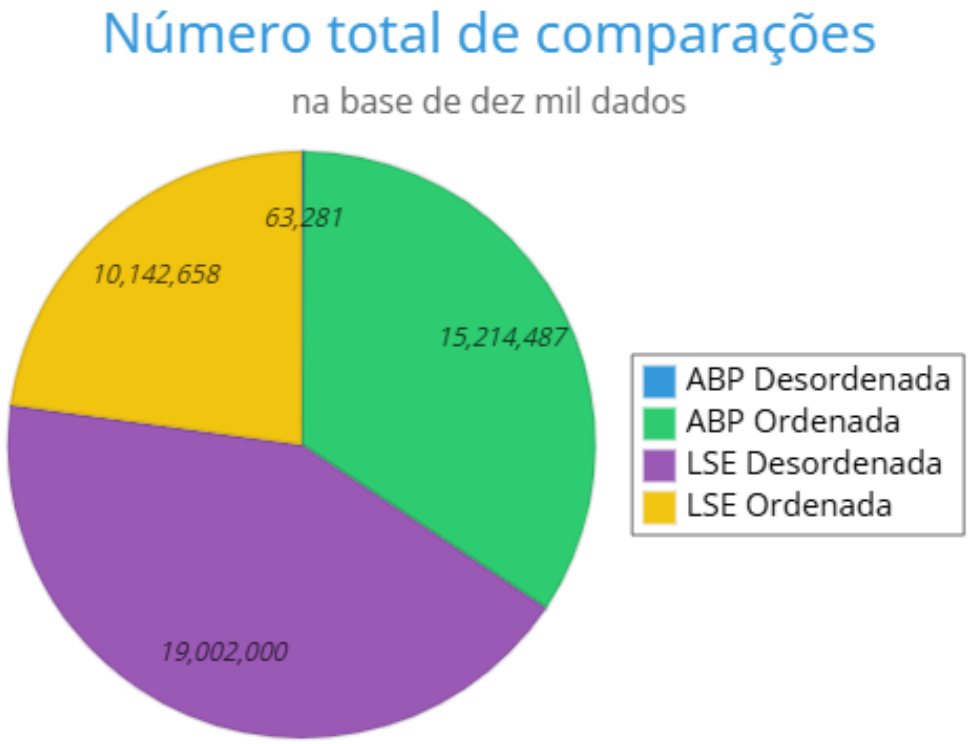

Imagem 4. Gráfico de setores sobre o número total de comparações entre ABP e LSE na base de dez mil dados, desordenada e ordenada.

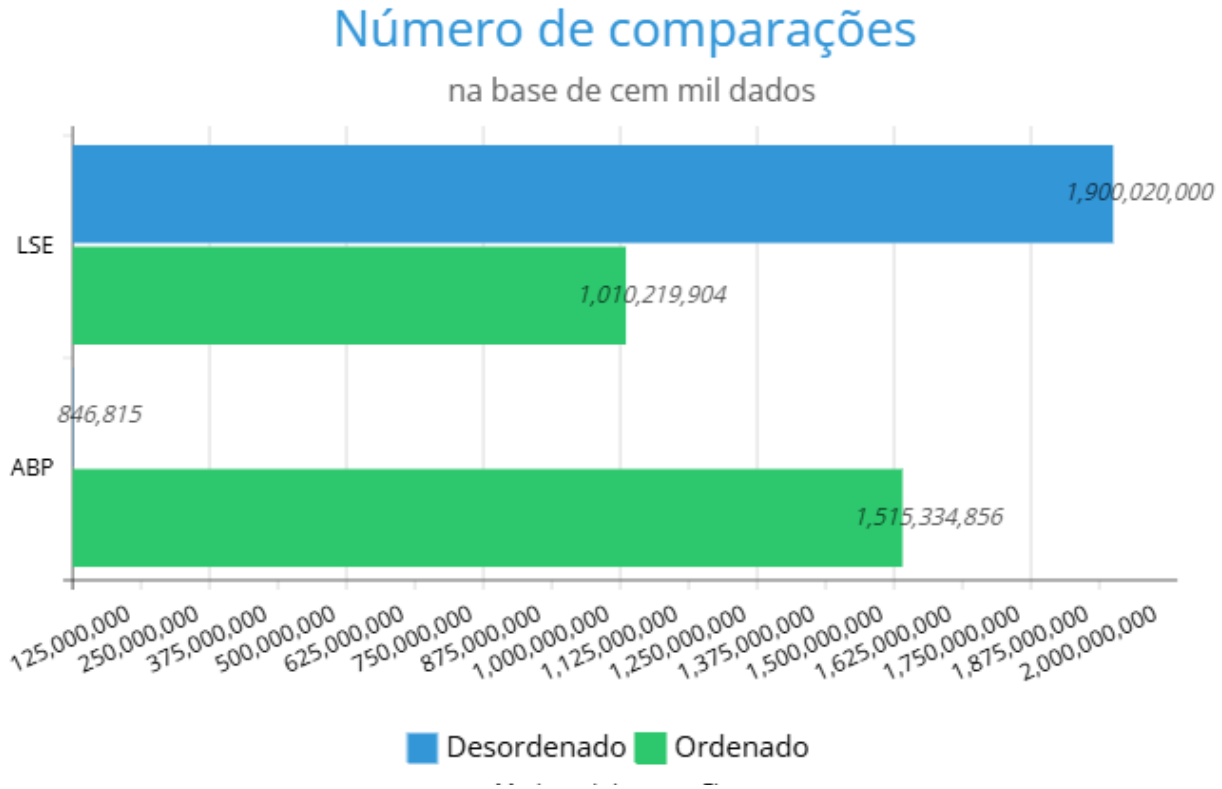

Imagem 5. Gráfico de barras sobre o número de comparações entre ABP e LSE na base de cem mil dados, desordenada e ordenada.

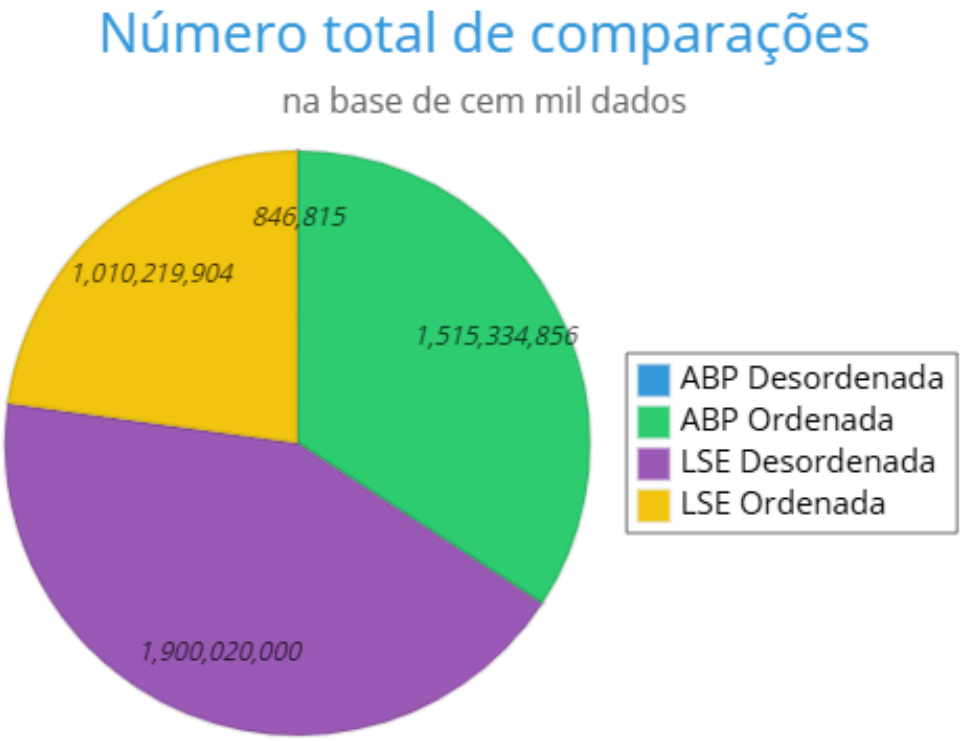

Imagem 6. Gráfico de setores sobre o número total de comparações entre ABP e LSE na base de cem mil dados, desordenada e ordenada.

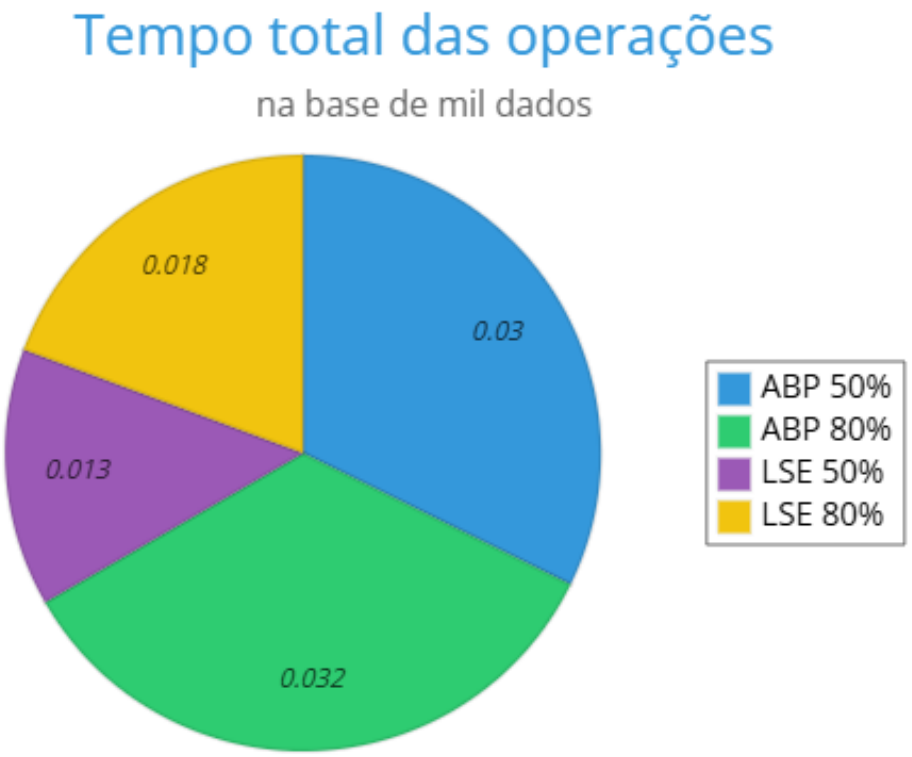

Imagem 7. Gráfico de setores sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de mil dados, com os testes de 50% e 80% de senhas erradas.

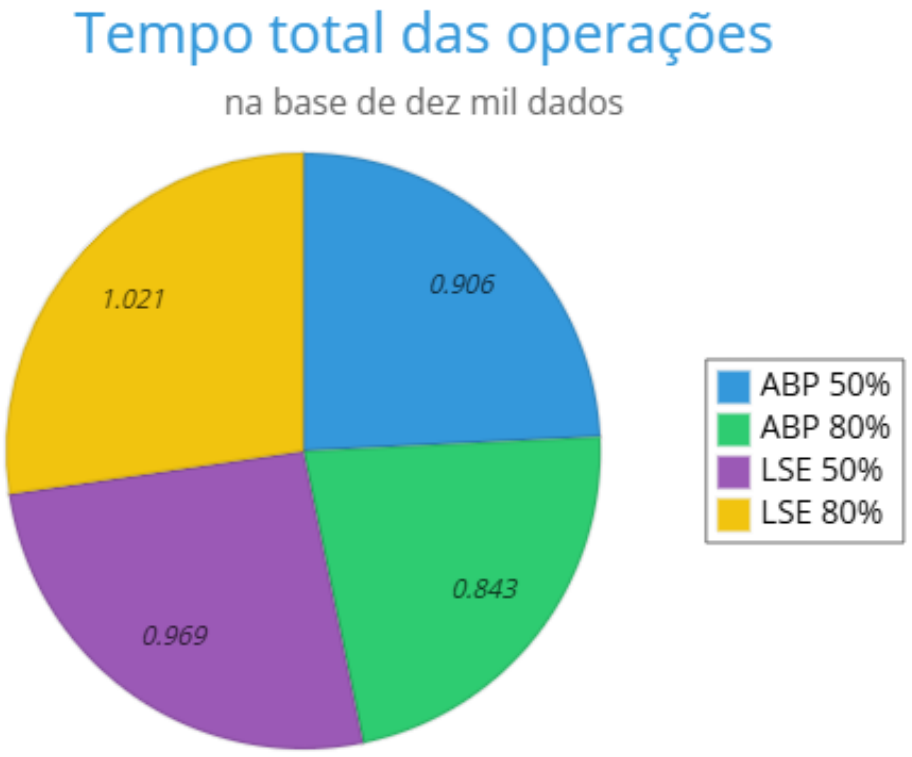

Imagem 8. Gráfico de setores sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de dez mil dados, com os testes de 50% e 80% de senhas erradas.

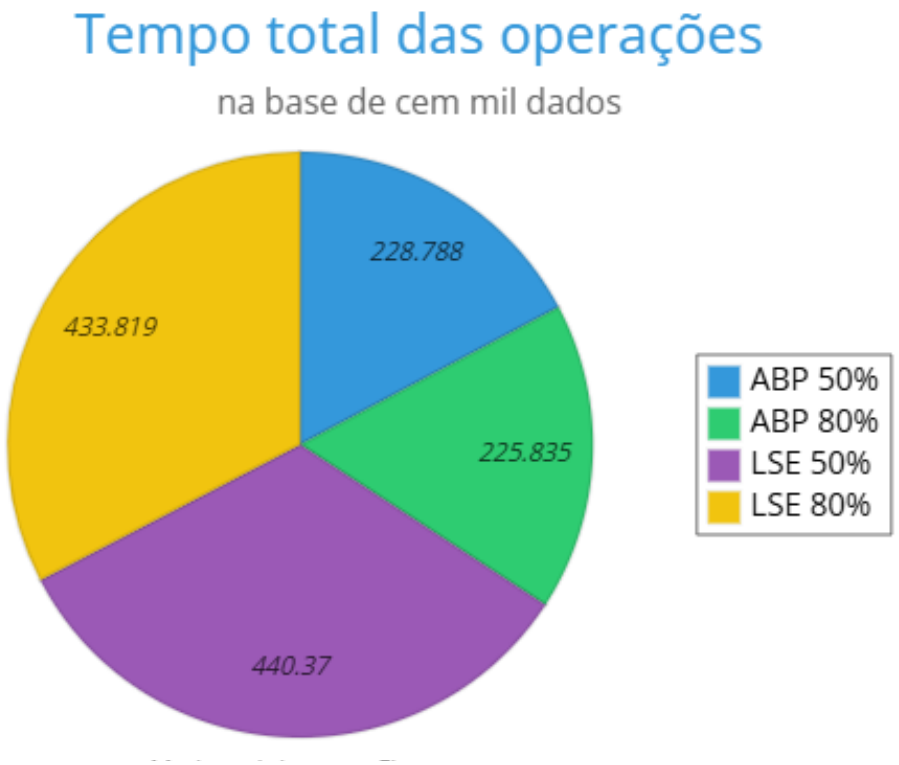

Imagem 9. Gráfico de setores sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de cem mil dados, com os testes de 50% e 80% de senhas erradas.

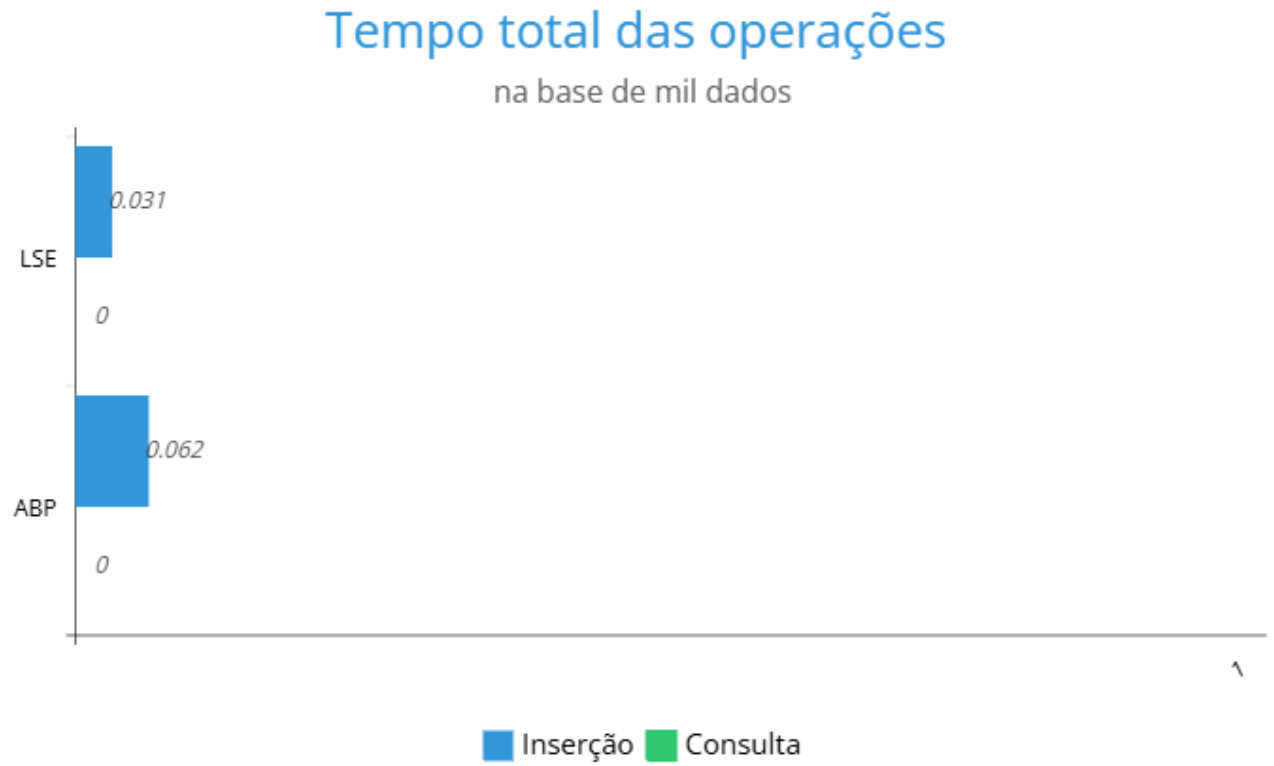

Imagem 10. Gráfico de barras sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de mil dados.

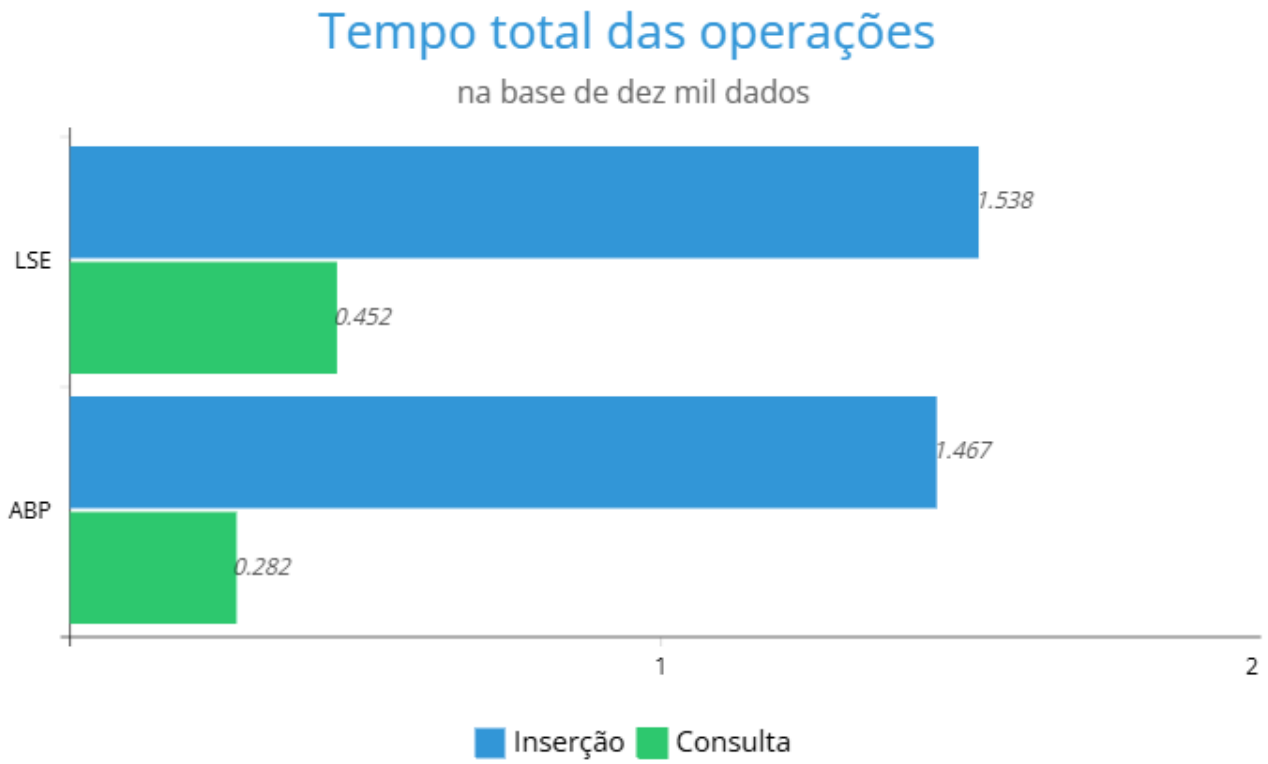

Imagem 11. Gráfico de barras sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de dez mil dados.

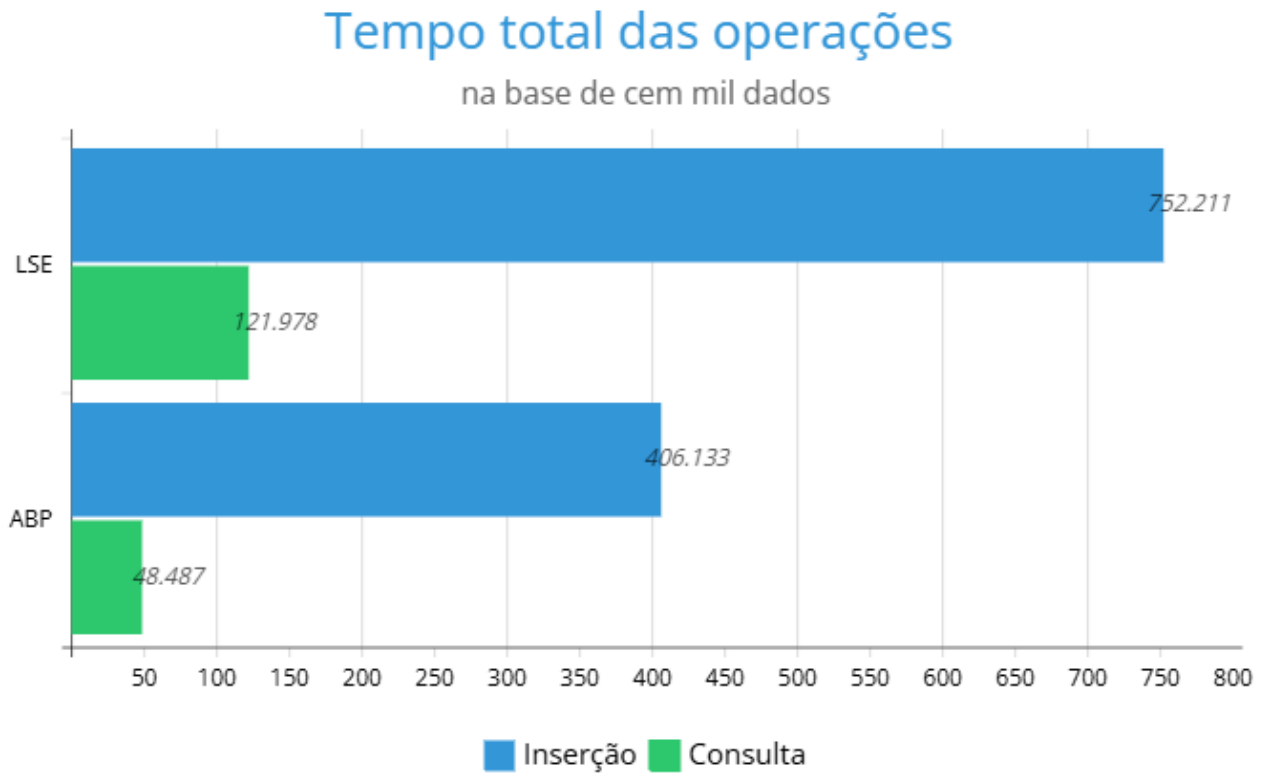

Imagem 12. Gráfico de barras sobre tempo total gasto pelas operações de inserção e consulta entre ABP e LSE na base de cem mil dados.

Conclusão

Após realizada a análise dos dados obtidos da comparação entre LSE e ABP, certos fatos puderam ser concluídos. O primeiro deles foi o de que a ABP apresenta um comportamento fortemente similar a LSE quando nela são inseridos dados ordenados, tanto em termos de número de comparações quanto em tempo de operações. Justamente por isso, percebe-se o quão essencial é o conceito de balanceamento em uma árvore, para que ela possa manter-se eficiente mesmo frente à enormes volumes dados, como ocorreu com a ABP que recebeu dados desordenados. O segundo foi que, apesar de realizar um número cada vez mais gigantesco de comparações e usar períodos de tempo ainda maiores nas operações conforme o tamanho das bases de dados aumentavam, a LSE demonstrou ser perfeitamente adequada para tratar pequenos conjuntos de dados, superando a ABP em boa parte das medidas envolvendo a base de mil dados.

E o terceiro, e principal, é que lidar com tamanhos avassaladores de dados é uma tarefa extremamente difícil e problemática, tanto logisticamente quanto em termos de processamento, ocupação de memória e tempo. Por exemplo, algoritmos que usam de recursão para tratar com tantos dados tendem a ser eficientes e rápidos, porém ocupam um espaço massivo devido às múltiplas chamadas recursivas que empregam, podendo inclusive resultar em erros de compilação como “stack overflow”. Por outro lado, remover a recursão elimina qualquer possibilidade de tais erros, porém ocasiona uma considerável demora no programa, que dependendo do computador onde é aplicado, pode levar vários minutos para terminar de executar. Em suma, é necessária muita atenção e paciência para tratar grandes grupos de dados, levando-se em consideração o espaço e tempo disponíveis, além é claro, da estrutura de dados mais eficaz para armazená-los.

Referências bibliográficas

- CRIAR seu gráfico. Disponível em: https://charts.livegap.com/?lan=pt#TypesofCharts. Acesso em: 20 dez. 2024.
- EGGEN, Joop. **Binary Seach Tree Insertion C without recursion**. Disponível em: https://stackoverflow.com/questions/36795065/binary-search-tree-insertion-c-without-recursion. Acesso em: 21 dez. 2024.
- EMBARALHE sua lista de nome (ordem aleatória). Disponível em: https://www.unitarios.com.br/ferramentas/lista-em-ordem-aleatoria. Acesso em: 18 dez. 2024.
- FIND and Replace Text. Disponível em: https://www.browserling.com/tools/text-replace. Acesso em: 19 dez. 2024.
- GALANTE, Renata de Matos. **TAD ABP - implementação em C.** Disponível em: https://moodle.ufrgs.br/. Acesso em: 2 dez. 2024.
- GALANTE, Renata de Matos. **TAD LSE em C.** Disponível em: https://moodle.ufrgs.br/. Acesso em: 2 dez. 2024.