Estruturas de Dados - I

Estudo de tempo de execução em filas duplamente encadeadas de prioridade com e sem referência móvel

Alunos: Luciano Kohler e Pedro Stupp
Professor: Gilmário Barbosa dos Santos

Joinville - 2025

SUMÁRIO

1 INTRODUÇÃO

Uma Fila Duplamente Encadeada de Prioridade (abreviada *FDEP*) é um tipo de estrutura de dados comumente utilizada em softwares que usufruem de funcionalidades de uma fila comum, que implementa a metodologia *FIFO* (First In, First Out), com a adição de funcionalidades como a ordenação de cada elemento da fila por intermédio de um atributo *prioridade* que pode se diferenciar de estrutura para estrutura, além de possibilitar a busca de valores dentro da fila tanto pelo seu início (elemento de maior prioridade), quanto seu fim (elemento de menor prioridade), tudo graças ao seu encadeamento duplo.

Outra metodologia muito utilizada na implementação de uma FDEP é o uso de uma *referência móvel*, que guarda em si uma posição da fila para uso futuro, seja para inserção, ou remoção de objetos. O foco principal de uma referência móvel é a otimização nas buscas por um valor dentro da fila dando um ponto de partida para essa busca, assim, evitando que seja necessário começar pelo início da fila toda para cada busca, se a referência móvel está próxima deste elemento, é possível começar por ele e assim, reduzir os custos computacionais que o programa possa ter, coisa que uma fila sem uma referência não teria o luxo de ter, tendo que iniciar suas buscas no ponto inicial da fila para cada operação feita.

Esclarecendo outro ponto, uma implementação de referência móvel busca diminuir o *custo computacional* do programa, isto é, a quantidade de recursos de máquina necessários para cumprir os requisitos do programa, veja à seguir, uma representação gráfica dos diferentes custos computacionais denotada de *Big O Graphic*:

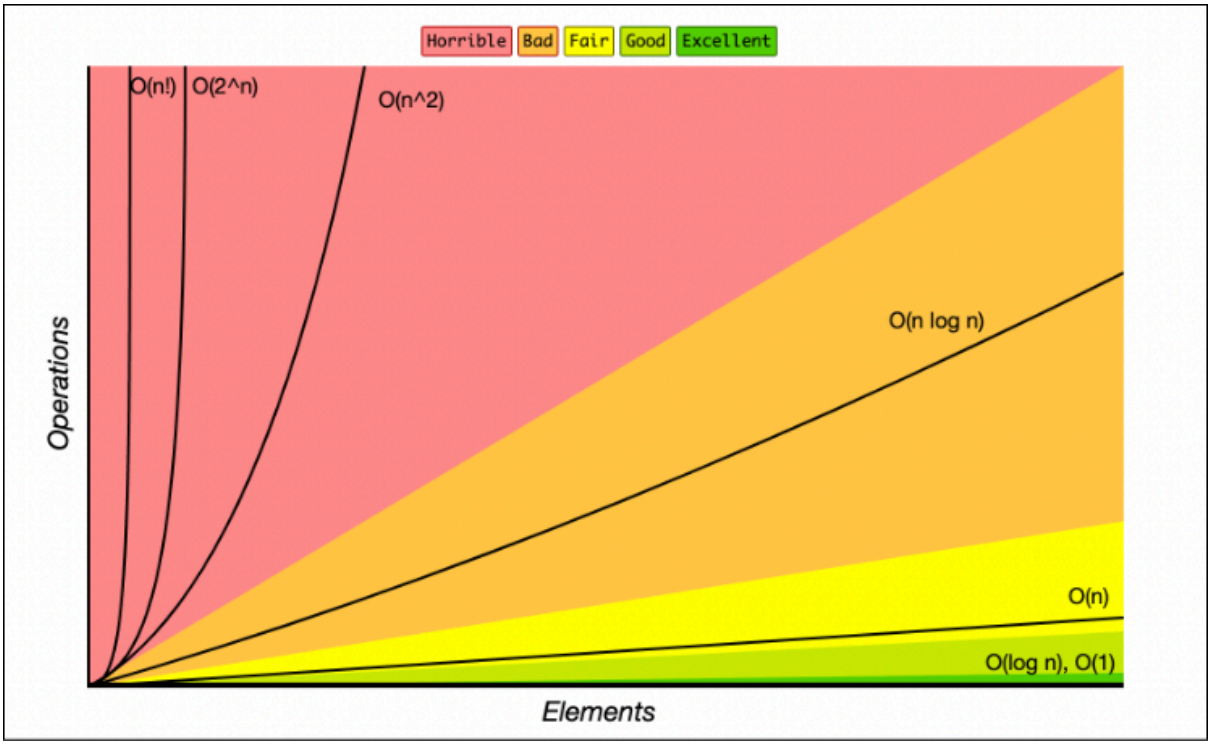

Figura 1: Gráfico de custos operacionais [1]

Como é possível ver, por exemplo, os de complexidade constantes (O(1)), ou logarítmicos (O(log n)) são ideias e devem ser buscados comumente, enquanto complexidades exponenciais (O(2^n)), ou fatoriais (O(n!)) são um veneno para o software,

pois indicam uma crescente falta de desempenho no sistema conforme o número de operações (n) aumentam.

Para medir a complexidade das duas filas à serem comparadas (FDEP com e sem referência móvel), foi utilizado uma metodologia de “bases”, onde foram escolhidos bases aleatórias de 500 a 9000 elementos de um banco de dados em .csv, e inseridos esses dados aleatoriamente em duas filas, uma com, e outra sem uma referência móvel, tudo isso enquanto era medido o número de iterações (quantas vezes cada inserção teve que percorrer a lista) de cada fila, assim gerando uma base de dados que possibilita a comparação de velocidade e otimização de ambas as estruturas, que por fim, pode mostrar se é realmente válido a implementação de uma FDEP com referência móvel.

Por fim, seja deixado claro aqui que nossos comparadores principais na criação da nossa base de dados são a *quantidade de iterações* que cada elemento teve que percorrer para ser inserido na lista. Esse dado é diretamente ligado com o ***n*** utilizado na representação de complexidade vista à cima, pois o mesmo indica a quantidade de elementos inseridos, fator que, ao crescer e aumentar a quantidade de dados à serem comparados, também aumenta a quantidade de iterações que a função tem que realizar para encontrar o local correto do elemento.

2 OBJETIVOS

GERAL

- Realizar uma análise comparativa de desempenho das FDEP com e sem referencial móvel.

ESPECÍFICOS

- Implementar uma visualização gráfica dos diferentes desempenhos feitos por cada fila, destacando a quantidade de iterações que cada estrutura teve que realizar nas diferentes bases aleatoriamente geradas.
- Comparar os diferentes resultados gerados pela implementação a fim de afirmar com clareza, a porcentagem de melhoria que uma fila com referência móvel tem em cima de uma fila sem uma referência.
- Entender de forma prática, a preocupação com custos computacionais na hora do desenvolvimento, destacando os graus de complexidade que podem afetar o desempenho no software ao trabalhar com cargas de dados pesadas.

3 METODOLOGIA

3.1 Recursos para o experimento

- Utilização de um notebook Dell Inspiron 15:
    - Memória RAM: 16GB
    - Processador: Intel Core i5-1235U 10 núcleos
    - Tipo de unidade de armazenamento: SSD M.1 512GB

- Implementação da estrutura de dados Descritor, que possui atributos responsáveis pela descrição da FDEP:
    - Tamanho da fila
    - Tamanho do dado

    - Frente da fila
    - Cauda da fila
    - Referência móvel

- Utilização de duas funções de inserção distintas baseadas em algoritmos passados nas aulas de Estruturas de Dados pelo professor:
    - inserirFilaPrioridadeREFMOVEL(), que insere os dados em uma fila usufruindo da funcionalidade da referência móvel
    - inserirFilaPrioridadeSEMREFMOVEL(), que insere os dados em uma fila sem o uso da referência móvel
- Uma base de dados aleatória contando com 10000 registros diferentes de alunos

3.2 Procedimento experimental

- Inicialmente, é **compilado** o código por meio do compilador GCC V14.2.0, com o terminal da IDE Visual Studio Code no SO Windows 11, informações técnicas da máquina especificadas no tópico anterior.
- Ao rodar o código, é criado uma fila encadeada denominada **bancoDeAlunos**, que contém **10000 registros** de alunos retirados de um arquivo .csv providenciado pelo professor
- Após isso, é chamada a função **embaralhaFila()**, que como o nome indica, pega os 10000 registros e embaralha-os por meio do algoritmo de *Fisher Yates* [2]
- Após isso, é iniciado um **loop** que inicia com o valor base **500**, valor esse que é então aplicado em **duas funções distintas** de inserção de valores em **filas também distintas**, uma função **utilizará** a referência móvel para auxiliar na inserção dos 500 valores, e a outra irá **ignorar** a existência do referencial móvel, inserindo os valores de forma menos eficiente.
- Após a chamada das funções, é extraído por meio de cada uma, uma informação **numIter**, que indica a quantidade de iterações que cada função desempenhou, esses valores são armazenados para uso futuro no programa.
- Ao final das inserções, o loop **incrementa** seu valor base em 500, e todas as outras informações, como o estado da fila e o numIter são **reiniciados** para gerar dados novos, o loop então **aplica novamente** a função com a nova base, e em seguida se incrementa em 500 unidades novamente, até chegar na base máxima 9000.
- Nota-se que a quantidade de iterações **cresce diretamente** com o crescimento da base de alunos usada, o que implica que quanto mais alunos houverem, maior será a **complexidade** da inserção, fazendo novamente uma referência ao nível de complexidade já visto em tópicos passados.
- Após tudo isso, é gerado uma tabela que mostra a quantidade de iterações que cada metodologia (com/sem ref. Móvel) fez para completar a ação em cada uma das bases, além de mostrar também a **porcentagem** de melhoria no desempenho da inserção utilizando referencial móvel:

RELATORIO DE DESEMPENHO DA FILA DE PRIORIDADE COM/SEM REFERENCIA MOVEL

BASE	TIPO INSERCAO	ITERACOES	ITERACOES MEDIAS	MELHORIA (%)
500	COM REF:	41584	83.17	
	SEM REF:	63604	127.21	34.62%
1000	COM REF:	159012	159.01	
	SEM REF:	242534	242.53	34.44%
1500	COM REF:	369941	246.63	
	SEM REF:	554098	369.40	33.24%
2000	COM REF:	662658	331.33	
	SEM REF:	988311	494.16	32.95%
2500	COM REF:	1017023	406.81	
	SEM REF:	1523632	609.45	33.25%
3000	COM REF:	1468311	489.44	
	SEM REF:	2209355	736.45	33.54%
3500	COM REF:	2032264	580.65	
	SEM REF:	3033208	866.63	33.00%
4000	COM REF:	2682421	670.61	
	SEM REF:	3998587	999.65	32.92%
4500	COM REF:	3347890	743.98	
	SEM REF:	5025321	1116.74	33.38%
5000	COM REF:	4145141	829.03	
	SEM REF:	6252794	1250.56	33.71%
5500	COM REF:	4995686	908.31	
	SEM REF:	7566777	1375.78	33.98%
6000	COM REF:	5947426	991.24	
	SEM REF:	8986930	1497.82	33.82%
6500	COM REF:	6973630	1072.87	
	SEM REF:	10522647	1618.87	33.73%
7000	COM REF:	8110210	1158.60	
	SEM REF:	12243821	1749.12	33.76%
7500	COM REF:	9261832	1234.91	
	SEM REF:	13954506	1860.60	33.63%
8000	COM REF:	10559549	1319.94	
	SEM REF:	15898154	1987.27	33.58%
8500	COM REF:	11955562	1406.54	
	SEM REF:	18007949	2118.58	33.61%
9000	COM REF:	13408804	1489.87	
	SEM REF:	20181966	2242.44	33.56%

Figura 2: Tabela de iterações em filas com Vs. sem referencial móvel, autoria própria

- Ao fim do código, as informações foram guardadas e o código, executado novamente para **embasar** melhor a amostra, e garantir que **não ocorram anomalias** na hora da depuração dos resultados do experimento.

4 RESULTADOS E DISCUSSÃO

Ao final da fase de geração de dados, foi feito um gráfico com os dados depurados com o uso da plataforma Canva:

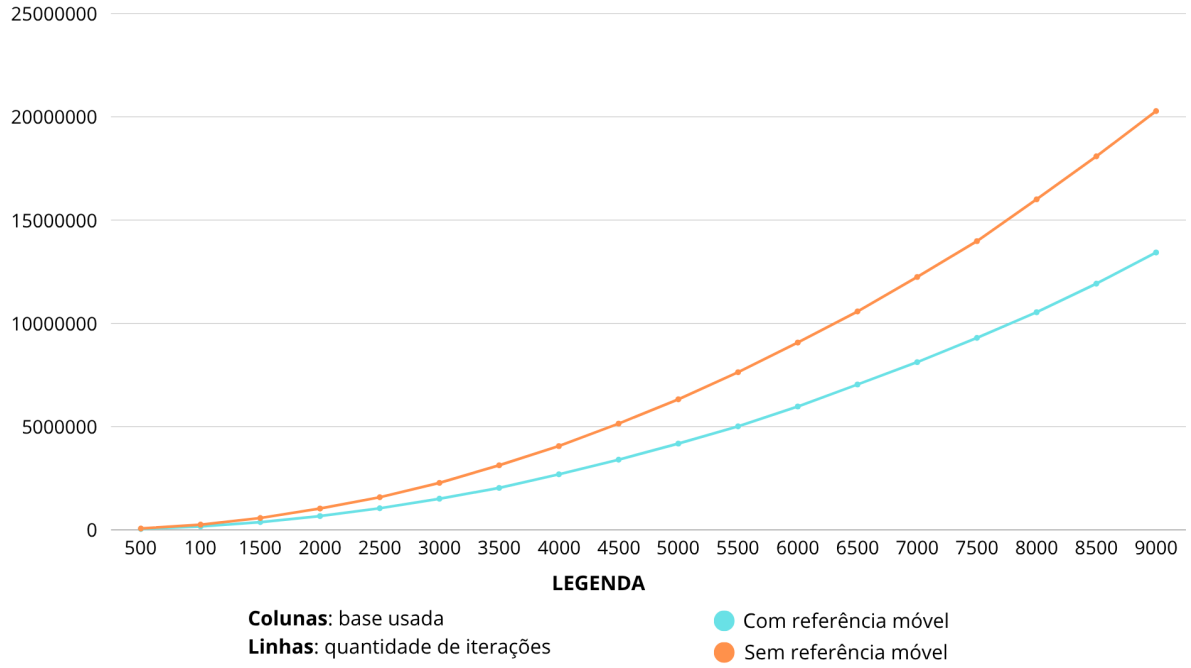

Figura 3: Depuração dos dados ao fim do experimento, feito via Canva

Com o gráfico em mãos, podemos identificar diversos pontos interessantes:

- Não ocorreram casos anômalos no gráfico, é possível notar que o gráfico tanto da inserção com ref. móvel, quanto sem, refletem uma linha suave e condizente com a sua complexidade.
- Ambos os gráficos seguem um padrão visto no gráfico Big O, e de fato, é possível perceber que o uso de uma referência móvel ajudou e muito, na otimização do software, diminuindo o crescimento da complexidade do código em aproximadamente 33% (veja Figura 2)
- Enquanto o gráfico da inserção sem referência móvel segue um padrão similar ao da complexidade O(n log n), a inserção com referência móvel também segue o mesmo comportamento, porém com uma taxa de crescimento muito menor do que a sua comparativa

Com um estudo mais teórico do funcionamento de uma FDEP com/sem referencial móvel, também é possível destacar os melhores e piores cenários que cada tipo de fila pode passar por, afinal, o uso de uma referência não garante que a inserção seja otimizada, mesmo que, na maioria esmagadora dos casos, existe sim uma otimização, vejamos:

- **FDEP com referencial móvel:**

    - Melhor cenário: A inserção é feita em ordem de prioridade crescente, assim o referencial móvel será sempre usado e o novo elemento é inserido logo após o referencial.
    - Pior cenário: Uma inserção em que os valores de prioridade estejam muito espaçados, assim, a referência móvel não terá efeitos significativos na otimização, fazendo com que a fila seja similar à uma FDEP comum.
- **FDEP sem referencial móvel:**
    - Melhor cenário: A inserção é feita em ordem de prioridade decrescente, assim, cada elemento sempre é inserido na primeira comparação de elementos.
    - Pior cenário: A lista é inserida em ordem crescente, pois assim, o elemento a ser inserido terá de percorrer a lista inteira até encontrar seu lugar na fila.

5 CONCLUSÃO

- A implementação da visualização gráfica foi um grande sucesso, principalmente levando em conta o uso final dos dados gerados, que ao final, nos deram uma curva extremamente similar às curvas vistas no *Big O Chart*, indicando que a implementação foi válida.
- A comparação dos resultados gerados pela implementação nos deram embasamento suficiente para afirmar que a implementação de uma referência móvel em um sistema de FDEP pode e irá reduzir os custos computacionais do software em aproximadamente 33% comparado à implementação sem o uso de um referencial móvel.
- A realização do trabalho nos deu informações suficientes para afirmar que é vital sim, pensar em custos computacionais na hora de implementar um código que se submeterá à bases muito densas de dados, deve se notar que a amostra de dados (10000), mesmo sendo grande, ainda não se compara aos verdadeiros bancos de dados que muitos sistemas têm que guardar em seus servidores, então a programação voltada à redução de complexidade se apresenta ainda mais vital em um cenário real e de grande porte.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] OLAWANLE, J. *Big O cheat sheet – time complexity* chart. FreeCodeCamp, 5 out. 2022. Disponível em: <https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/>. Acesso em: 30 maio 2025.

[2] ESTEVES, R. *Algoritmo de Fisher-Yates para embaralhamento de arrays*. Medium, 3 Jul. 2018. Disponível em: <https://raullesteves.medium.com/algoritmo-de-fisher-yates-para-embaralhamento-de-arrays-ba13a0542e88>. Acesso em: 28 maio 2025