UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

RELATÓRIO DE ANÁLISE EMPÍRICA DE IMPLEMENTAÇÃO DE ESTRUTURAS ABSTRATA DE DADOS

NATAL/RN
2020

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

DIEGO FILGUEIRAS BALDERRAMA
VINÍCIUS OLIVEIRA DA SILVA

RELATÓRIO DE ANÁLISE EMPÍRICA DE IMPLEMENTAÇÃO DE ESTRUTURAS ABSTRATA DE DADOS

Trabalho referente à nota parcial da Unidade III da disciplina DIM0119 - Estrutura de Dados Básica I - T02, orientado pelo Prof. Mr. Guilherme Fernandes de Araújo.

NATAL/RN
2020

SUMÁRIO

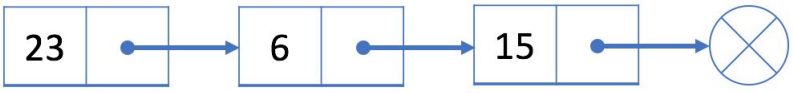
1. INTRODUÇÃO

Um Tipo Abstrato de Dados, ou simplesmente TAD, é como um modelo matemático formado por um modelo de dados (mais conhecidos como atributos) e um conjunto de procedimentos (mais conhecidos como métodos ou operações) que manipulam os dados que o compõem. Alguns exemplos de TADs que podemos dar são as listas encadeadas, filas, pilhas, dicionários, árvores etc.

Os atributos de um TAD são manipulados exclusivamente pelos métodos implementados no seu conjunto de procedimentos, o que é de grande utilidade e segurança, visto que códigos externos não têm, dessa forma, permissão de alterar seus estados diretamente.

Neste trabalho, implementamos a lista duplamente encadeada e a fila. Agora, vejamos brevemente o comportamento desses Tipos Abstratos de Dados.

A primeira TAD implementada foi a lista duplamente encadeada. Para entender o que ela é, precisamos antes ver como uma lista simplesmente encadeada funciona. Esse TAD é composto por vários nós onde são armazenados dois dados: um valor e um ponteiro, que geralmente, aponta para o próximo nó da lista.



Exemplo de lista simplesmente encadeada (Fonte: LeetCode)

A lista duplamente encadeada é um pouco mais completa que a simplesmente encadeada, pois ela armazena mais um ponteiro que aponta para o nó estabelecido na posição anterior.

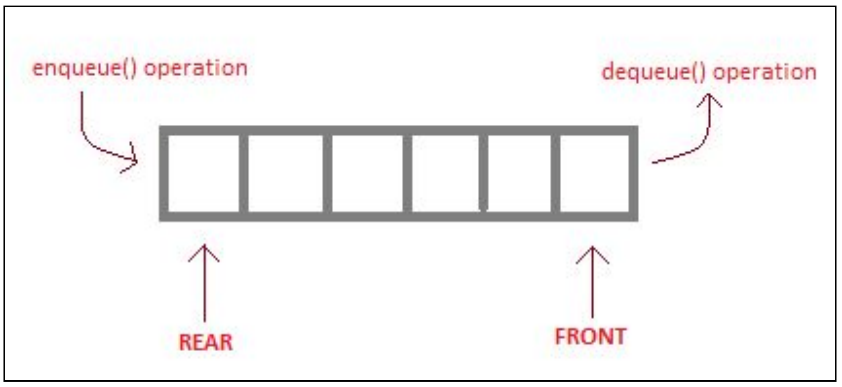
Exemplo de lista duplamente encadeada (Fonte: LeetCode)

Algumas vantagens de usar uma lista duplamente encadeada em detrimento de um array, por exemplo, são:

- A inserção de um novo dado só exige a alocação de um nó;

- O tamanho da lista armazenada é alinhado ao tamanho da lista real;
- Inserir um novo elemento entre outros dois não envolve deslocamentos.

A segunda TAD implementada foi a fila. As filas são Tipos Abstratos de Dados que podem ser implementados usando vetores ou listas encadeadas e representam muito bem o conceito de fila que conhecemos no dia a dia, onde há a ideia de *First In First Out*, ou seja, o primeiro a chegar é o primeiro a sair. Além disso, por padrão os novos elementos são adicionados ao final da fila e existem dois ponteiros indicando o início e o fim da fila.



Exemplo de fila (Fonte: Studytonight)

A vantagem do seu uso é que a fila pode se adaptar melhor a um certo tipo de problema do que outros Tipos Abstratos de Dados que não seguem a regra do *First In First Out*.

2. DESENVOLVIMENTO

No presente trabalho, realizamos a análise empírica dos métodos de inserção e deleção da lista duplamente encadeada e da fila. Os algoritmos foram implementados em C++, utilizando o compilador g++ (9.3.0) e a IDE Microsoft Visual Studio Code (1.51.1). Além disso, foram utilizadas as ferramentas Git e GitHub para versionamento de código e o Gnuplot versão 5.2 para plotar os gráficos a partir dos dados obtidos. O repositório no GitHub pode ser acessado para consulta por meio do link a seguir: https://github.com/viniciu21/TadAnalysis.

Para os dois as dua operações, inserção e remoção, o tamanho da amostra variou de 100 a 10000 elementos, aumentando de 100 em 100, e para cada tamanho foram feitos 50 testes diferentes, nos quais foram aplicados uma média aritmética para diminuir os picos de processamento paralelo e gerar resultados mais consistentes e homogêneos.

Para implementar a lista duplamente encadeada, primeiramente foi implementada a classe do nó que compõe a lista. Observe abaixo seus atributos e métodos:

```
1   #include <iostream>
2
3   using namespace std;
4
5   class Node {
6      private:
7       int key;
8       Node *prev;
9       Node *next;
10
11     public:
12      Node();
13      void set_key(int x);
14      void set_prev(Node *x);
15      void set_next(Node *x);
16      int get_key();
17      Node *get_prev();
18      Node *get_next();
19
20      ~Node();
21  };
```

Atributos e métodos da classe Node

Assim, em seguida usamos essa classe para implementar a lista duplamente encadeada de fato, como mostrado a seguir:

```
1   #include <iostream>
2
3   #include "./Node.hpp"
4
5   using namespace std;
6
7   class Doubly_Linked_List {
8      private:
9       Node *head = new Node();
10
11     public:
12      Doubly_Linked_List();
13      void insert(int x);
14      void removeFirst();
15      void print_nodes();
16  };
```

Atributos e métodos da classe da lista duplamente encadeada

Como dito anteriormente, é possível implementar uma fila usando vetores ou lista encadeada. Neste trabalho, a fila foi implementada utilizando vetores. Além disso, para manter uma coerência na comparação da operação de remoção na lista duplamente encadeada e na fila, foi implementado um método para remover o último elemento da fila, no qual ele remove um elemento por vez partindo do início da fila, até chegar ao final, removendo, assim, o último elemento.

Na implementação da fila, foi preciso somente de uma classe, a classe Queue, como descrita a seguir:

```
#ifndef __QUEUE_
#define __QUEUE__

class queue {
    int *arr;
    int capacity;
    int front;
    int rear;
    int count;

  public:
    queue(int size);
    ~queue();

    void dequeue_all();
    void enqueue(int x);
    int peek();
    int size();
    bool isEmpty();
    bool isFull();
};

#endif
```

Atributos e métodos da classe Queue

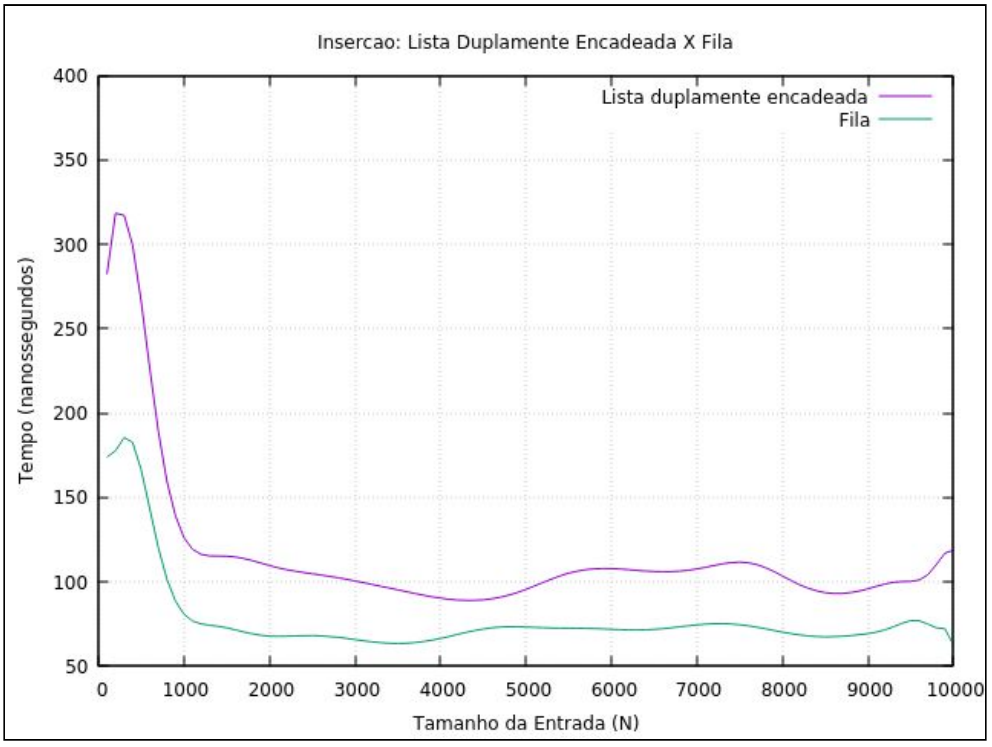
Após a execução dos testes, o tempo médio de execução de cada operação dos dois Tipos Abstratos de Dados foram salvos em arquivos externos para que, em seguida, fosse possível a geração dos gráficos por meio do Gnuplot.

3. RESULTADOS E CONSIDERAÇÕES FINAIS

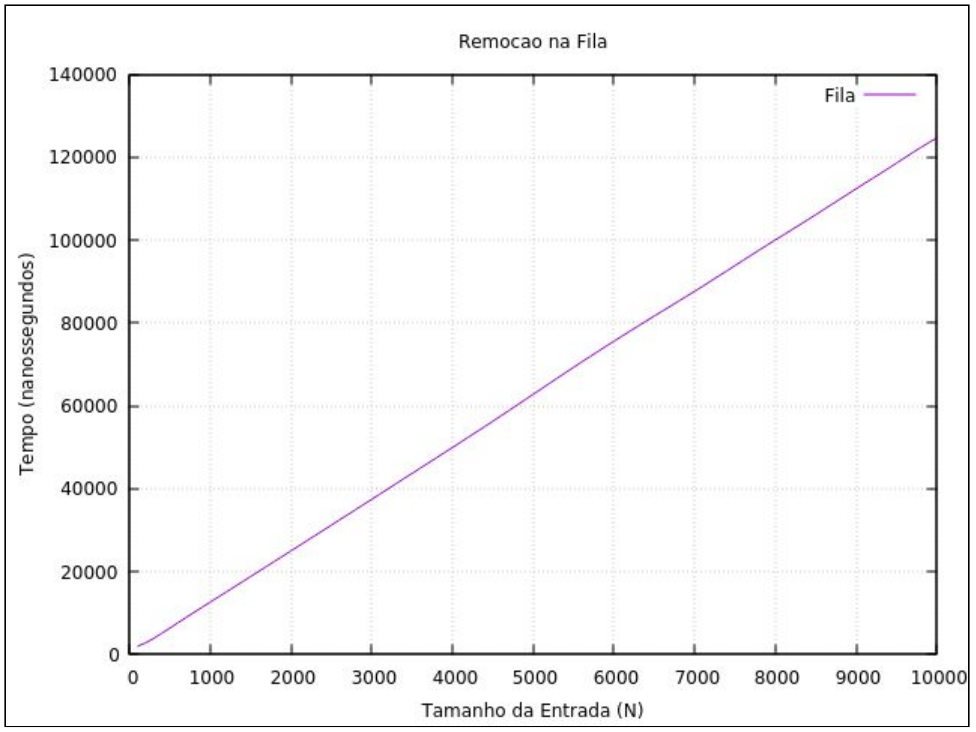
Para rodar o programa e gerar os gráficos foi utilizado um computador com as seguintes especificações e ferramentas:

- Sistema Operacional: Ubuntu 20.04.
- Processador: Intel Core i5-7200U 3.1 GHz.
- Memória RAM: 8GB.
- Arquitetura: x86_64.
- Linguagem: C++11.
- Compilador: G++ 9.3.0.
- Gnuplot: 5.2.8

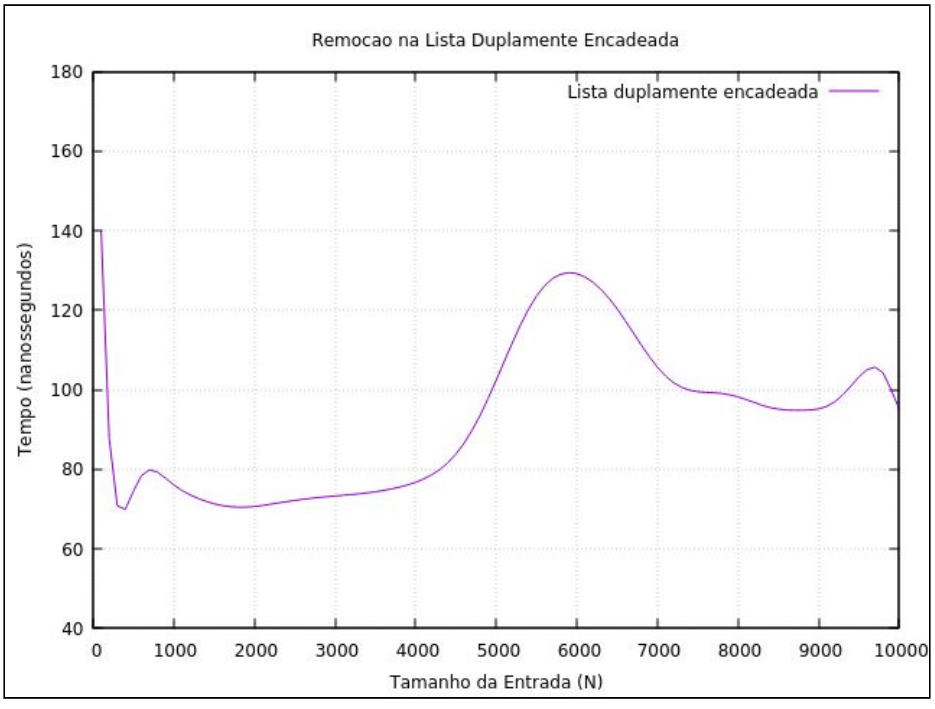
A começar pela inserção, os dois Tipos Abstratos de Dados, tiveram quase o mesmo resultado. Podemos observar no gráfico abaixo que, no início, houve um pico no tempo de inserção dos primeiros elementos, provavelmente devido a algum pico de processamento na própria máquina, mas que após esse momento, o tempo se não apresentou nenhum crescimento definido. Isso se dá pelo fato da inserção, tanto da lista duplamente encadeada quanto na fila, ser O(1), logo o tamanho da entrada não interferiu no tempo que levado para inserir elementos nesses Tipos Abstratos de Dados.



Na remoção de elementos da fila, podemos ver claramente o seu crescimento linear mostrado pelo gráfico abaixo. Esse crescimento linear se dá pelo motivo de termos implementado um método para remover o último elemento da fila, conforme justificado anteriormente. Podemos dizer, portanto, que a complexidade da remoção implementada é O(N). Caso tivesse sido implementado o método de remoção usual da fila, sua complexidade seria O(1).

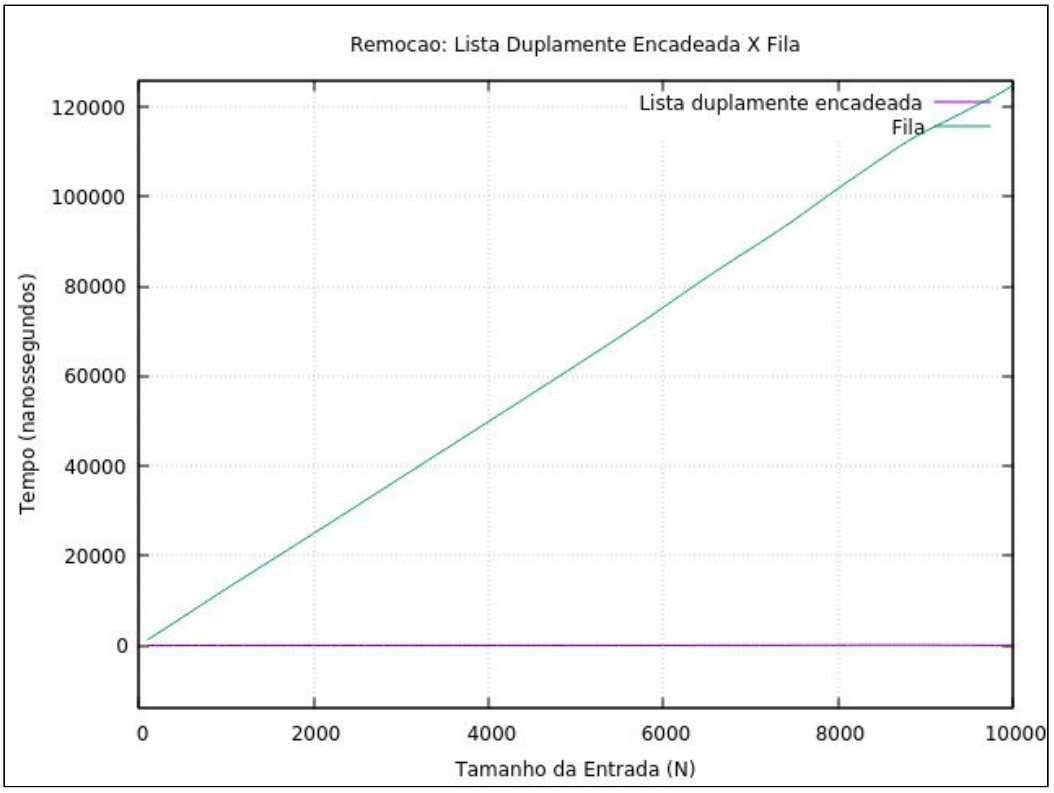


Já na remoção de elementos da lista duplamente encadeada, quando vemos o seu gráfico, pode ser difícil de identificar qual é a sua complexidade baseado no seu crescimento.



O tempo de remoção na lista duplamente encadeada variou entre 56.48 nanossegundos e 140.2 nanossegundos. Pode-se dizer que essa foi uma variação significativamente pequena, permanecendo praticamente constante, portanto podemos concluir que a complexidade da remoção na lista duplamente encadeada é O(1).

Quando comparamos o gráfico da fila junto do gráfico da lista duplamente encadeada, fica mais claro de perceber o quão melhor é a remoção do último elemento na lista duplamente encadeada em comparação com a fila.



REFERÊNCIAS

MAFFEO, Bruno. **Tipos Abstratos de Dados**: Definição e Exemplos. Departamento de Informática. PUC-Rio.