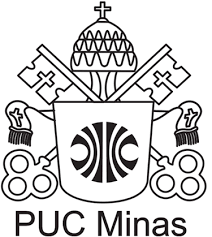
****

**Análise assintótica:**

***Gerenciador de Prateleira***

**Integrantes:** Amaury Alexandrino da Costa - 499960

Caroline Francisco Magalhães dos Santos - 499951

Felipe Henrique da Silva Rocha - 495779

João Fernando de Souza – 499972

**Turma:** 3º período – Sistemas de Informação(Noite)

**1) Análise de complexidade**

**1.1) Lista**

*1.1.1) Verificar se a Lista está vazia*

Ordem de complexidade O(1), pois a instrução será realizada apenas uma vez, que é nada mais que uma única comparação para saber se a Lista está vazia.

*1.1.2) Buscar posição do elemento na Lista*

Somente a busca, possui ordem de complexidade O(n), pois no pior das hipóteses, se a posição de inserção do elemento for a última, vai-se percorrer até o final da Lista fazendo as devidas comparações. Então a Lista será percorrida no máximo n vezes, que seria o seu tamanho.

*1.1.3) Inserção e remoção na Lista*

A inserção é de ordem O(1), pois realiza poucas operações.

A remoção é de ordem O(n), pois o loop procurando o elemento a ser removido, será executado no máximo n vezes, que seria o tamanho da Lista.

**1.2) Fila**

*1.2.1) Verificar se a Fila está vazia*

Ordem de complexidade O(1), pois a instrução será realizada apenas uma vez, que é nada mais que uma única comparação para saber se a Fila está vazia.

*1.2.2) Inserção e remoção na Fila*

A inserção é de ordem O(1), pois realiza poucas operações.

A remoção é de ordem O(n), pois o loop procurando o elemento a ser removido, será executado no máximo n vezes, que seria o tamanho da Fila.

**1.3) Prateleira**

*1.3.1) Método FIFO*

Na pior das hipóteses, a ordem de complexidade será O(n²), pois contém a operação de ler o arquivo (1 vez – O(1)), depois tem de percorrer todo o registro pegando caractere por caractere dentro do arquivo (O(n)), além de posteriormente percorrer a prateleira pra saber se o elemento(Produto) já se encontra nela (O(n) – no pior das hipóteses).

Logo, 1\*(n\*n) = n² => O(n²).

*1.3.2) Método LRU*

Assim como no FIFO, o LRU possui ordem de complexidade O(n²), onde contém a operação de ler o arquivo (1 vez – O(1)), depois tem de percorrer todo o registro pegando caractere por caractere dentro do arquivo (O(n)), além de fazer chamada à função busca da classe Lista, que procura a posição ordenada na Lista – que no caso é ordenada pelo tempo de acesso de um produto na prateleira, onde no pior das hipóteses a prateleira é percorrida totalmente (O(n)).

Logo, 1\*(n\*n) = n² => O(n²).

*1.3.3) Método Segunda Chance*

O algoritmo Segunda Chance possui ordem de complexidade O(n²), onde contém a operação de ler o arquivo (1 vez – O(1)), depois tem de percorrer todo o registro pegando caractere por caractere dentro do arquivo (O(n)), além de posteriormente percorrer a prateleira pra saber se o elemento(Produto) já se encontra nela (O(n)). Caso necessite dar uma segunda chance a algum elemento, o algoritmo procura algum produto em alguma página antiga que ainda não tenha sido referenciada. No pior das hipóteses (todos os produtos tiverem sido referenciados) a prateleira será percorrida totalmente, reinserindo produto no final da fila até encontrar um bitR = 0 (O(n)).

Nesse caso, apesar de uma ordem de complexidade parecida com a do FIFO e LRU, no Segunda Chance o número de operações é maior, pois tem a possibilidade de se percorrer a Lista duas vezes

Logo, 1\*((n\*n) + (n\*n)) = 2\*n² => O(n²).

----------------------------------------------------------------------------------------------------------