Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: Estrutura de Dados I – BCC 202
Aluno: Felipe Fontenele de Ávila Magalhães
Matrícula: 15.1.4331

1. b) O tempo de execução do algoritmo usando notação assintótica das operações *push()* e *pop()* é dado pela tabela:

Operações	**Notação Ω (Melhor Caso)**	**Notação O (Pior Caso)**
Push()	Ω(1)	O(n)
Pop()	Ω(1)	O(n)

Pela tabela, percebemos que a operação *push()* no pior caso (Big O) é O(n) devido ao fato de passar os itens da fila 1 para a fila 2 com o custo O(n), logo após inserção do item digitado pelo usuário no final da fila 1 com custo O(1) e por fim devolver os itens da fila 2 para a fila 1 com o custo O(n), considera-se a complexidade de maior grau permanecendo O(n). Já o melhor caso (Ômega) da operação *push()* é Ω(1), considerando quando não houvesse nenhum item na pilha, fazendo a inserção do primeiro item digitado pelo usuário no final da fila 1 com o custo Ω(1). A operação *pop()* no pior caso (Big O) é O(n) pois retira um item no início da fila 1 com o custo n do tamanho da pilha. Já o melhor caso (Ômega) da operação *pop()* é Ω(1), considerando a pilha com somente um item, havendo um custo Ω(1).

2.b) A complexidade de tempo do algoritmo de ordenação é **$O(n^2)$**,pois foi utilizado o método BubbleSort(apesar de não ser a melhor opção), entretanto a entrada dados é pequena sendo o motivo da utilização do mesmo pela fácil implementação. O algoritmo divide a lista em 02 vetores, 01 contendo somente palavras e outro com números. A partir disso, aplica-se a ordenação BubbleSort em cada um dos vetores, e por fim é feito a intercalação dos vetores mantendo a sequência original digitada da lista. Para manter a sequência original foi criada um vetor de inteiros, para ser a minha "flag", 1 para palavras e 2 para números. Portanto, a complexidade do algoritmo é de ordem quadrática **$O(n^2)$**, já que aplica-se duas vezes a ordenação BubbleSort.

3.d) Percebe-se nitidamente que o programa obteve uma longa demora na execução com relação na ordenação em todos os casos pedidos, em especial, nos métodos de ordenação BubbleSort, SelectionSort e InsertionSort, levando bastante tempo para a execução do código. O pior caso registrado foi a entrada estar em ordem decrescente,

gerando muitas comparações e movimentações em alguns dos métodos. Os métodos MergeSort, QuickSort e ShellSort apresentaram um tempo de execução melhor, número de comparações e movimentações menores em comparação com os demais métodos. Por fim, este programa foi o que obteve uma maior demora na execução de todos os outros feitos.