**UNIVERSIDADE FEDERAL DO MARANHÃO**

**CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA**

**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

**DISCIPLINA: ESTRUTURA DE DADOS II**

**ALUNOS: CAIO MEDEIROS E DANILO MEDEIROS**

**RELATÓRIO**

**ATIVIDADE COMPLEMENTAR**

**São Luís**

**2017**

1. **Introdução**

O problema apresentado foi desenvolver dois métodos de ordenação, utilizando os algoritmos de Ordenação InsertSort, SelectSort, ShellSort, QuickSort, HeapSort, MergeSort e CountSort. O primeiro método deveria ordenar, em ordem alfabética, apenas as palavras que possuíssem 4 ou mais caracteres.O segudo método deveria ordenar as palavras em ordem decrescente de acordo com o número de ocorrência de cada palavra no arquivo. A utilização de algoritmos de ordenação é muito comum, podendo ser facilmente encontrados quando se precisa ordenar uma lista de alunos, por exemplo.

1. **Implementação**

Para a resolução deste problema utilizou-se uma struct uma matriz de string. As principais funções utilizadas foram:

* SelectionSort,InsertionSort,ShellSort,HeapSort,QuickSort,MergeSort,CountingSort: As funções de ordenação utilizadas.
* preencheBuffer,preencheMatriz: Leiam o arquivo e coloca as strings em um vetor.
* escolheMet: escolhe o método em qual será ordenado o arquivo.

A maioria das funções recebe dados do tipo Char e retornam o mesmo tipo.

1. **Análise de Complexidade**

As funções de ordenação utilizadas são as mesmas estudadas em sala de aula, e seus algoritmos não foram alterados demasiadamente, portanto as complexidades dos algoritmos se mantiveram,em média, como a seguir:

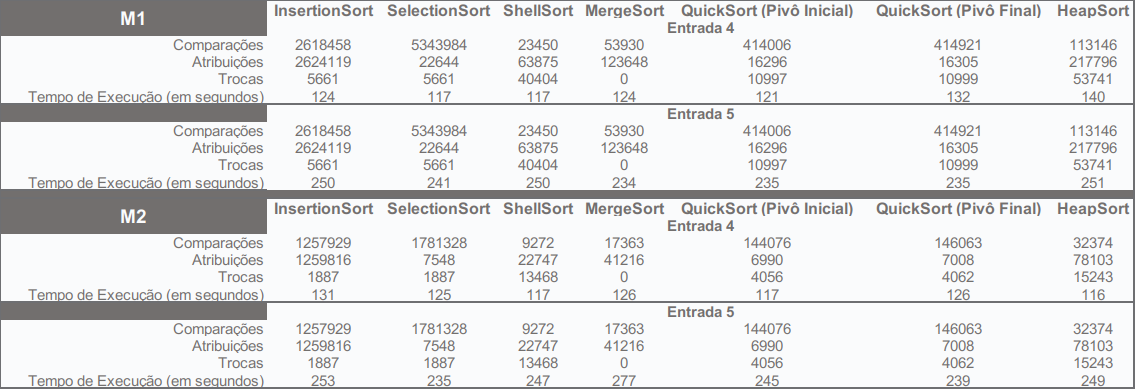
* Selection Sort: O(n2)
* InsertionSort: O(n2)
* ShellSort: Indefinida
* MergeSort: O(nlogn)
* QuickSort: O(nlogn)
* HeapSort: O(nlogn
* CountingSort: O(n)

Contudo, as funções criadas para desenvolver os métodos e a execução do programa, possuem complexidades distindas. Elas estão representadas em seguida:

* contaPalavras: O(n);
* Aloca: O(n);
* preencheMatriz: O(n2);
* escolheOrdena: O(8n);
* desaloca: O(n);
* Imprime: O(n);

1. **Testes e Análise dos Resultados**

Executou-se os algoritmos algumas vezes: os de ordenação em cada método. Como o algoritmo construído dos métodos exclui palvras repetidas da ordenação, os número de trocas, atribuições e comparações foi o mesmo em todas as entradas no método 1 e no método 2 também houve a memsa situação, mas com valores diiferentes dos apresentados no méotodo 1. Observou-se uma óbvia reduçãoo no tempo de execução, para os dois métodos, a medida que o tamanho da entrada também era reduzia, o que era esperado. Para visto de análise, comparou-se os algoritmos de ordenação em cada método com as entradas 4 e 5, por serem as maiores e apresentarem dados mais realistas, como pode-se visualizar na tabela 01.



**Tabela 01 - Dados da execução dos algoritmos de ordenação para entradas 04 e 05.**

Como observado nos dados apresentados na tabela, no método 1 o algoritmo de ordenação que apresentou melhor tempo de execução foi o MergeSort. Já no método 2, O QuickSort (com pivô inicial) e o SelecionSort foram os algoitmos mais rápidos, no geral.

1. **Conclusão**

O trabalho realizado apresentou dados não muito conclusivos. A diferença nos tempos de execução, para os dois métodos, não foi muito grande, podendo sugerir que a maior parte do tempo se deu apenas para ler a entrada. O tempo para ordenar em si as palavras pode ter ficado no menor detalhe. O hardware da máquina pode ter influenciado também no tempo de execução, visto que pode não ter havido poder de execução suficiente para otimizar a leitura das entradas.

Por fim, de qualquer forma, pôde-se concluir que o QuickSort foi, de forma geral, o algoritmo mais rápido, como esperado.

A maior dificuldade encontrada foi criar o método 2 de forma que lesse que a struct criada, esta construída no intuito de facilitar a ordenação no método 2. Todas as funções tiveram que ser adequadas para ler esta struct. Além disso, houve uma certa dificuldade em implementa o algoritmo do CountingSort, para o que foi solicitado.

**REFERÊNCIAS**

**CPLUSPLUS.http://www.cplusplus.com/reference/cstring/**