

Universidade Federal de Viçosa – Campus Florestal

Disciplina: CCF 211 - Algoritmos e Estrutura de Dados 1

Professora: Thais Regina de Moura Braga Silva

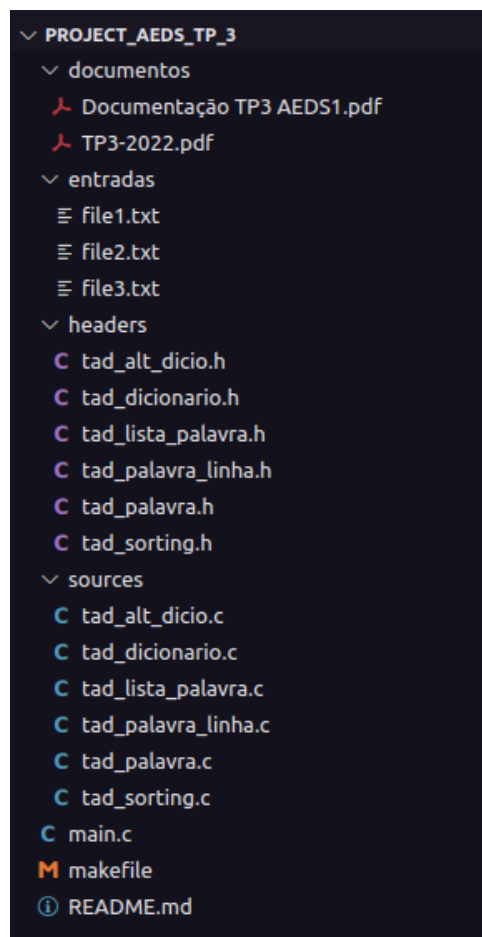
Alunos: Marcos Biscotto - 4236, Alan Araújo - 5096, Gabriel Marques - 5097

1. Introdução

Para esse Trabalho Prático, foi-nos solicitado incrementar o TAD Dicionário com a possibilidade de, após montar as listas das palavras separadas pela letra do alfabeto com a qual começam, oferecer ao usuário a possibilidade de visualizá-las ordenadas. Para tal, criamos TAD's e utilizamos funções e comandos da Linguagem C para realizar as ordenações das palavras de cada lista de palavras. Depois disso, medimos o tempo de execução de cada algoritmo de ordenação separadamente. Através de um menu interativo, exibimos opções para o usuário escolher de que forma utilizará o programa.

2. Organização

Em relação à organização e arquitetura do repositório do projeto, consideramos deixá-los da forma mais organizada possível. Novamente colocamos a documentação do trabalho em uma pasta dentro do repositório do projeto e também deixamos alguns arquivos de entrada como exemplo dentro de uma pasta no repositório. Separamos os arquivos “.h” e “.c” em pastas separadas e deixamos na pasta geral os arquivos main.c, o makefile do projeto e o arquivo “README”, como demonstrado na **Figura 1**. O arquivo makefile possui os comandos de compilação e execução do programa, estruturado de maneira a compilar e rodar ao introduzir o comando “make” no terminal ou IDE.



3. Desenvolvimento

Abaixo relatamos o processo de desenvolvimento do TP, desde sua interpretação até o que foi solicitado como produto final, além de descrições sobre as tomadas de decisões e complicações a respeito do desenvolvimento do programa.

3.1) Interpretação:

Diferente do último Trabalho Prático, a interpretação deste foi bastante simples, uma vez que os objetivos eram sucintos e as premissas já haviam sido atendidas nos outros trabalhos. Após tirarmos algumas dúvidas com a professora e os monitores, demos início ao esqueleto do projeto.

3.2) Headers e Sources:

Como o trabalho usa nosso Dicionário de Palavras desenvolvido no TP1, reciclamos os arquivos “.c” e “.h” dele e criamos outros 2, sendo eles o “alt_dicio” e “sorting” que contém, respectivamente, o processo criação do novo dicionário e os algoritmos de ordenação estudados na disciplina. Diferentemente do TP2, utilizamos outros arquivos “.c” além da main para deixar a visualização das funções de forma mais simples.

3.3) Tomada de Decisões:

Primeiramente, mudamos a parte referente ao TP1 que dizia respeito às Listas Encadeadas e substituímos por Vetores de itens através do TAD “alt_dicio”. Assim, geramos as listas de palavras iniciadas com cada letra do alfabeto e, após isso, oferecemos opções dentre as quais estão o ordenamento dessas listas.

3.4) Construção do Dicionário

A implementação inicial do novo dicionário ocorreu corretamente. Foram implementados os métodos Bubble, Selection, Insertion e Shell sem nenhum empecilho. Entretanto, a implementação do Quick e Heap ocorreram problemas, que resultaram em “segmentation fault”. Para solucionar esse problema foi preciso implementar novamente o TAD “alt_dicio”.

3.5) O problema do Heap Sort:

Por algum motivo, o algoritmo de ordenação Heap Sort apresentado no slide da disciplina teve complicações para ser implementado. Portanto, optamos por utilizar um Heap Sort estruturado de maneira alternativa encontrada na internet. Tal código está devidamente referenciado no final da documentação.

4. Resultados

Como resultado, temos dois menus interativos funcionais (**Figura 2 e Figura 3**) que apresentam ao usuário algumas opções de manuseio do código, recebe um comando do mesmo e realiza uma operação baseada na escolha. Com a execução do código para os diversos algoritmos de ordenação para o mesmo arquivo de entrada, plotamos gráficos de execuções para as operações e fizemos tabelas para apresentar de forma mais simplificada a média de tempo, comparações e movimentações das operações, comparando todos os algoritmos de ordenação para um mesmo arquivo de texto.

```
----- DICIONÁRIO DE AEDS -----
Opções do dicionário:
Nome do arquivo:
1. Inserir nome do arquivo
2. Construir o dicionário do texto
3. Exibir a lista de palavras de determinada letra
4. Exibir toda a lista de palavras do texto
5. Verificar o número de palavras do texto
6. Verificar se determinada palavra está na lista de palavras do texto
7. Remover determinada palavra de uma lista de palavras
8. Ordenação
9. Instruções para usar o programa
10. Sair do programa
-----
Opção desejada: █
```

Figura 2 - Menu Interativo Dicionário

```
----- ORDENAÇÃO DO DICIONÁRIO -----
Opções de ordenação:
Nome do arquivo:
1. Construir o dicionário do texto
2. Ordenar e exibir a lista de palavras de determinada letra
3. Ordenar e exibir toda a lista de palavras do texto
4. Instruções e observações para realizar a ordenação
5. Sair da ordenação e voltar ao menu inicial
-----
Opção desejada: █
```

Figura 3 - Menu Interativo Ordenação

As saídas do “DICIONÁRIO DE AEDS” são as mesmas do TP1. A saída do “ORDENAÇÃO DO DICIONÁRIO” (**Figura 4 e Figura 5**) pode apresentar uma lista específica ou toda a lista de palavras, exibindo a lista desordenada, informações relevantes sobre a ordenação e por fim o resultado da ordenação.

```

-----
UNSORTED:

Letra |B| :

[bonita]

[bolo]

[bom]

[bem]

-----

Ordenação com Quick Sort!

-----

Letra |B|
-----
Palavras atuais: 4.000000
Comparações atuais: 2.000000
Movimentações atuais: 3.000000
Tempo atual: 0.000003
-----
Total de palavras: 4.000000
Total de listas de palavras: 1.000000
Total de comparações: 2.000000
Total de movimentações: 3.000000
Tempo total: 0.000003
-----
Média de palavras/lista: 4.000000
Média de comparações/lista: 2.000000
Média de movimentações/lista: 3.000000
Média de tempo/lista: 0.000003
-----

SORTED:

Letra |B| :

[bem]

[bolo]

[bom]

[bonita]

-----

```

Figura 4 - Informações de Ordenação

```

-----
Ordenação com Heap Sort!

-----

Letra |A|
-----
Palavras atuais: 8.00
Comparações atuais: 16.00
Movimentações atuais: 7.00
Tempo atual: 0.000005
-----
Total de palavras: 8.00
Total de listas de palavras: 1.00
Total de comparações: 16.00
Total de movimentações: 7.00
Tempo total: 0.000005
-----
Média de palavras/lista: 8.00
Média de comparações/lista: 16.00
Média de movimentações/lista: 7.00
Média de tempo/lista: 0.000005
-----

Letra |B|
-----
Palavras atuais: 4.00
Comparações atuais: 4.00
Movimentações atuais: 3.00
Tempo atual: 0.000002
-----
Total de palavras: 12.00
Total de listas de palavras: 2.00
Total de comparações: 20.00
Total de movimentações: 10.00
Tempo total: 0.000007
-----
Média de palavras/lista: 6.00
Média de comparações/lista: 10.00
Média de movimentações/lista: 5.00
Média de tempo/lista: 0.000004
-----

```

Figura 4 - Informações de Ordenação

O algoritmo foi executado em uma máquina com as seguintes especificações:

- Sistema operacional: Ubuntu 22.10
- Processador: Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz com 4 núcleos e 4 threads
- Memória RAM: 12GB DDR3 1600MHz
- Armazenamento: SSD 240GB Kingston A400
- Placa de vídeo: NVIDIA GeForce GTX 960 SC 2GB
- Placa-mãe: B85M-E/BR (ASUSTeK COMPUTER INC.)

Os seguintes resultados foram obtidos:

Tabela 1: Tabela das Médias para o Arquivo 1

| X | Bubble Sort | Selection Sort | Insertion Sort | Shell Sort | Quick Sort | Heap Sort |
|------------------------|-------------|----------------|----------------|------------|------------|-----------|
| Média de Tempo | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 |
| Média de Comparações | 11.24 | 11.24 | 6.12 | 5.12 | 7.18 | 8.82 |
| Média de Movimentações | 4.47 | 2.82 | 3.82 | 4.12 | 3.35 | 2.82 |

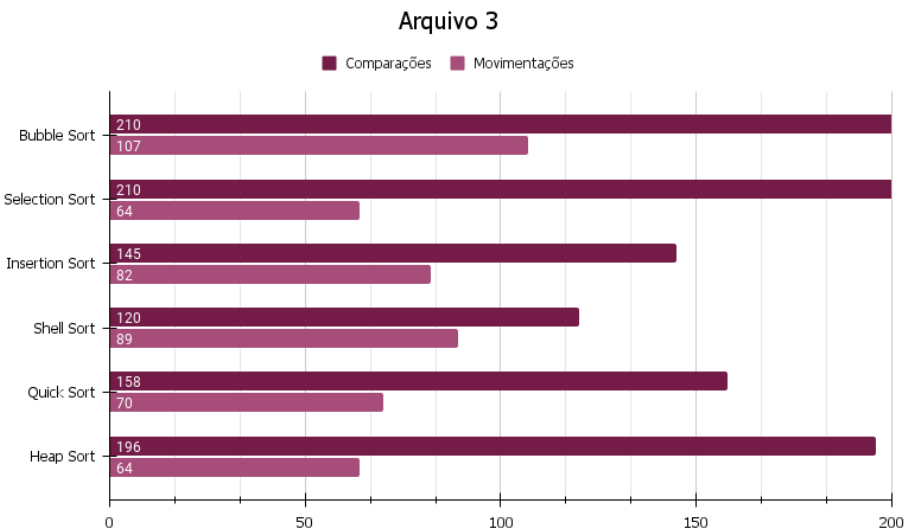
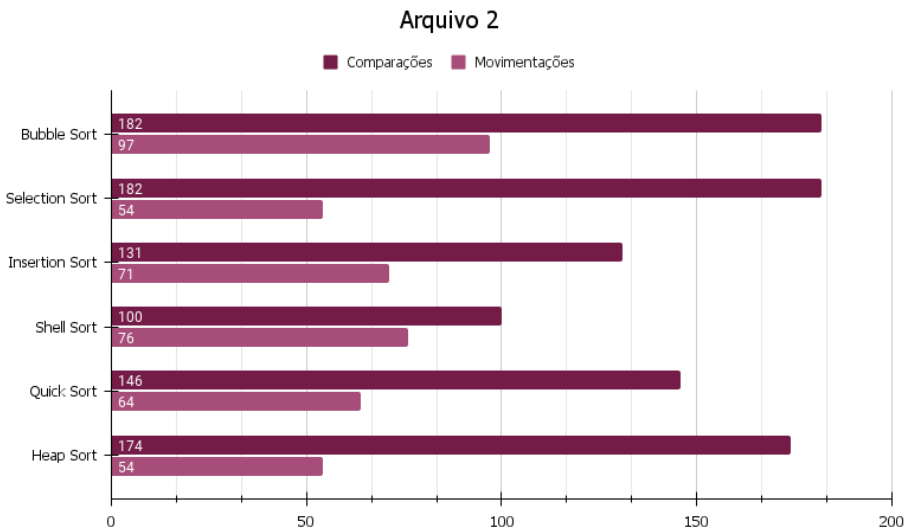
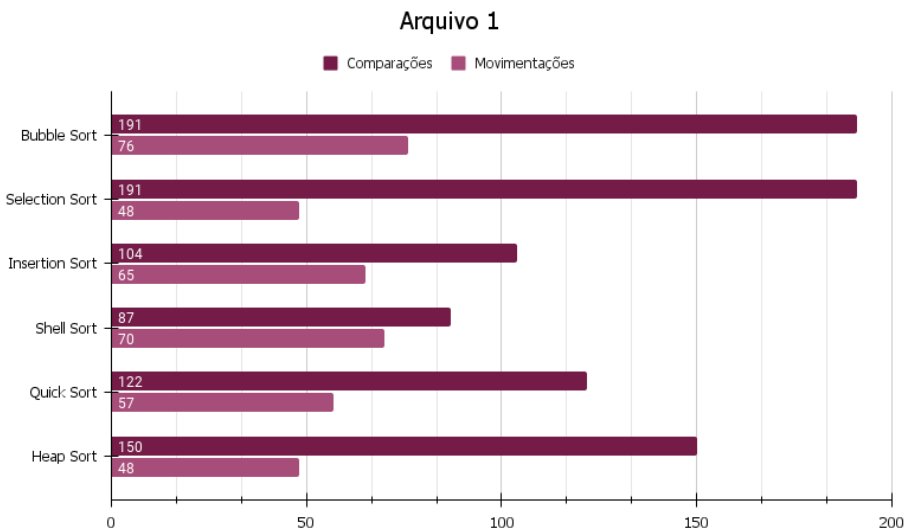
Tabela 2: Tabela das Médias para o Arquivo 2

| X | Bubble Sort | Selection Sort | Insertion Sort | Shell Sort | Quick Sort | Heap Sort |
|------------------------|-------------|----------------|----------------|------------|------------|-----------|
| Média de Tempo | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 |
| Média de Comparações | 10.71 | 10.71 | 7.71 | 5.88 | 8.59 | 10.24 |
| Média de Movimentações | 5.71 | 3.18 | 4.18 | 4.47 | 3.76 | 3.18 |

Tabela 2: Tabela das Médias para o Arquivo 3

| X | Bubble Sort | Selection Sort | Insertion Sort | Shell Sort | Quick Sort | Heap Sort |
|------------------------|-------------|----------------|----------------|------------|------------|-----------|
| Média de Tempo | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 | 0.000001 |
| Média de Comparações | 11.67 | 11.67 | 8.06 | 6.67 | 8.78 | 10.89 |
| Média de Movimentações | 5.94 | 3.56 | 4.56 | 4.94 | 3.89 | 3.56 |

Resultado em gráfico para os valores totais de ordenação para cada arquivo:



5. Conclusão

Por fim, com o término do trabalho, obtivemos um resultado que ao nosso ver atende às especificações anteriormente prescritas. Com a utilização de funções da linguagem C importadas de bibliotecas como `time.h` e dos algoritmos de ordenação a nós apresentados em aula, conseguimos realizar as operações necessárias e medir o tempo das mesmas no término de suas execuções. Também fizemos a leitura de um terceiro arquivo de texto para obtermos mais análises e enriquecermos a documentação. Com isso, pudemos ter uma noção na prática de como funcionam cada algoritmo de ordenação, seus custos computacionais e das vantagens e desvantagens de cada um.

6. Referências

[1] Github. Disponível em: <<https://github.com/>> Último acesso em: 08 de dezembro de 2022

[2] Gitlab. Disponível em: <<https://gitlab.com/>> Último acesso em 07 de dezembro de 2022

[3] Stack Overflow. Disponível em <<https://stackoverflow.com/>> Último acesso em 10 de dezembro de 2022

[4] Geeks for Geeks. Disponível em <<https://www.geeksforgeeks.org/>> Último acesso em 08 de dezembro de 2022

[5] ZIVIANI, Nivio. **Projeto de Algoritmos com Implementações em Pascal e C**. 3ªEd. Cengage Learning, 23 junho 2010.

[6] Programação Descomplicada (para implementação do Heap Sort). Disponível em <<https://programacaodescomplicada.wordpress.com/>> Último acesso em 11 de dezembro de 2022.