Universidade Federal da Bahia
Instituto de Matemática e Estatística
Departamento de Ciência da Computação

Disciplina: EDA II	Docente: Flávio Morais de Assis	Ciência da Computação
Discentes: Bárbara Luz, Fabiano Fernandes, Mateus Barbosa		21/06/2019

ORDENAÇÃO EXTERNA

1. DECISÕES E DISCUSSÕES PRÉ IMPLEMENTAÇÃO

A equipe se reuniu e após algumas discussões optou-se por utilizar o método de intercalação balanceada de vários caminhos. Essa escolha foi feita, pois o algoritmo de intercalação balanceada de vários caminhos não extrapola a quantidade pré determinada de arquivos definidas, e acreditamos que esta seria a melhor opção visto que iríamos trabalhar com uma quantidade limitada de arquivos.

2. ORGANIZAÇÃO DO PROJETO

O programa para manipulação de arquivo foi desenvolvido em C++ com a utilização das bibliotecas `iostream`, `stdio.h`, `string.h`, `algorithm`, `sstream`, `iostream`, `fstream` que definem os tipos e métodos de entrada/saída, leitura/escrita em arquivos e ordenação interna utilizados ao longo do trabalho.

O projeto foi dividido em cinco arquivos:

- **main.cpp**: arquivo que possui o método main e é responsável por receber os comandos e chamar as funções correspondentes;
- **sort.h**: arquivo onde estão definidos os métodos de inserção, consulta, ordenação, inicialização de arquivos, além dos métodos auxiliares para concatenação e separação de strings, deleção de arquivos e comparação, utilizado na ordenação. Neste arquivo, também estão definidos os tipos abstratos de dado *Record*, composto por chave e conteúdo e *FileAttributes,* composto pelo nome do arquivo e linha que está sendo lida.

- **sort.cpp**: os métodos definidos em *sort.h* são implementados neste arquivo.
- **Makefile**: define diretivas de compilação, remoção de arquivos e população.
- **input-generator.py**: gerador de arquivo de entrada.

O programa salva os registros nos arquivos em formato de texto sequencialmente.

2.1 GERADOR DE ENTRADA

Com o intuito de facilitar o processo de testes, foi desenvolvido um gerador de entrada denominado `input-generator.py`, que é responsável por criar o arquivo `entrada.dat` no qual insere 50 registros com chaves de até 20 caracteres e conteúdo correspondente a um texto aleatório com até 50 caracteres.

2.2 MAKEFILE

O projeto possui um arquivo Makefile responsável por facilitar o processo de compilação e testes. O Makefile define os seguintes comandos:

- **make** - Compila o projeto.
- **make faker** - Instala a biblioteca necessária para executar o gerador de input e gera o arquivo com as entradas.

3. ALGORITMOS

Nessa seção serão descritos os principais algoritmos utilizados para manipulação de arquivo e ordenação. Rotinas auxiliares, portanto, não serão comentadas.

3.1. INSERÇÃO

O pseudocódigo a seguir representa a ideia geral do algoritmo de inserção de um registro num arquivo.

```
insert(Record record, ofstream &file) [@ sort.cpp]
```

```
string r = implode(record, ‘,’)
escreva r no arquivo
```

A inserção do registro no arquivo ocorre em tempo O(1), pois os registro são sempre adicionados no final do arquivo.

3.2. INITIALIZE FILES

Conforme descrito por Ziviani no algoritmo de intercalação polifásica de vários caminhos, o arquivo a ser ordenado deve ser separado em blocos de tamanho menor ou igual ao processável em memória primária, distribuídos em arquivos que serão posteriormente intercalados. Esta separação dos registros nos arquivos é realizada na função `initialize files`.
O pseudocódigo a seguir representa o algoritmo descrito:

```
initializeFiles()
     para i = 1 até maxArqs/2
          crie o arquivo Ai.dat

enquanto houver registros em entrada.dat
     distribua os registros em metade dos Ai
```

3.3. EXTERNAL SORT

O algoritmo para ordenação externa consiste na ordenação dos blocos e sua intercalação na outra metade dos arquivos. A fim de ordenar os blocos, foi utilizado um vetor, que foi passado como parâmetro para a função de ordenação.
A intercalação, no entanto, não foi bem sucedida.

4. DIVISÃO DE ATIVIDADES

Após a definição do algoritmo a ser implementado, as atividades subsequentes foram distribuídas da seguinte forma:

- **Bárbara:** responsável pela organização da estrutura dos arquivos, revisão e otimização de código, implementação do gerador de entradas em Python, implementação das funções auxiliares para concatenação e separação de

strings, deleção de arquivos e pela implementação da inserção de um registro no arquivo.

- **Fabiano:** responsável pela organização da estrutura dos arquivos, revisão e otimização de código.
- **Mateus:** responsável pela organização da estrutura dos arquivos, revisão e otimização de código, implementação da ordenação, implementação da função inicialização dos arquivos, implementação da função de consulta de um registro e implementação da função auxiliar de `explode`.

5. REFERÊNCIAS

ZIVIANI, Nivio. **Projeto de Algoritmos Com Implementações em Pascal e C.** 4. ed. São Paulo: Pioneira, 1999.

THARP, Alan L. **File Organization and Processing.** North Carolina: Editora Wiley, 1998.