**FUNDAÇÃO CENTRO DE ANÁLISE, PESQUISA E INOVAÇÃO TECNOLÓGICA - FUCAPI**

**RELATÓRIO DE ORDENAÇÃO EXTERNA COM MERGE SORT**

# Manaus - Amazonas

**Abril, 2016**

**FUNDAÇÃO CENTRO DE ANÁLISE, PESQUISA E INOVAÇÃO TECNOLÓGICA - FUCAPI**

Relatório do trabalho prático da disciplina Projeto e Analise de Algoritmos sobre ordenação externa utilizando algoritmo merge sort.

# Manaus - Amazonas

**Abril, 2016**

**A – Nomes e Matriculas**

Evelyn Barbosa | RA:

Genilson Ferreira | RA: 070877

Guilherme Mateus | RA:

Thays Almeida | RA:

**B - Descrição sobre os objetivos do trabalho**

Utilizar ordenação externa para ordenar um arquivo de 6GB, dado os seguintes parâmetros: memoria disponível e K. Memória disponível servirá para limitar o tamanho do arquivo para ordenação na memória interna e o K, servirá para indicar quantos arquivos serão ordenados, ex: K = 2 . serão combinados arquivos de dois em dois, K = 3, serão combinados os arquivos de 3 em 3.

**C – Descrição de implementação**

A implementação foi dividida nas etapas abaixo:

1. – Descobrir quantos arquivos de “N”MegaBytes de memória vamos precisar para ordenar um arquivo de “N”MegaBytes .
   1. – Converter tamanho do arquivo a ser ordenado e o tamanho de memória disponível para bytes .
   2. - dividir o tamanho do arquivo a ser ordenado em bytes pelo tamanho da memória disponível em bytes. Como resultado temos a quantidade de arquivos de “N”bytes que será necessária criar para dividir o arquivo a ser ordenado.

REFERÊNCIA.

\PAAMergeSort\PAAMergeSort\controllers\ArquivoController.cs

Metodo: lerEscreverArquivo(string caminhoArquivoLeitura, string caminhoArquivoEscrita, string tamArquivoPrincipal, string tamArquivoSecundario, string KVetores)

1. – Dividir o arquivo principal em NPartes ordenadas
   1. – Com o resultado do passo 1.2 é criado um loop, que indica quantas vezes precisaremos ler o arquivo principal para divido-lo.
   2. – A leitura é feita parte a parte, primeiro é feito a leitura no arquivo da posição 0 á NBytes(memória disponível), criasse um novo arquivo com esse resultado já ordenado utilizando MergeSort.
   3. – Repete-se o passo 2.3 até o loop terminar na quantidade de arquivos conforme passo 2.1.

REFERÊNCIA.

\PAAMergeSort\PAAMergeSort\controllers\ArquivoController.cs

Metodo: separarArquivoSecundarios(string caminhoArquivoLeitura, string caminhoArquivoEscrita, byte[] bytesToRead, int quantidadeArquivosSecudarios)

**2.4 -** Após arquivos separados, é feito a combinação entre eles utilizando Listas (Pilha) de números inteiros

2.5 – Ler-se umas parte do primeiro e do segundo arquivo com a memória disponivel e alocamos em duas pilhas.

2.6 – Compara-se posição por posição das pilhas, e armazenamos o resultado numa terceira pilha resultante.

2.7 – Durante a comparação, os números menores são armazenados numa terceira pilha que será armazenada num arquivo resultante no final da comparação.

2.8 – O números maiores são armazenados numa quarta pilha para ser comparado com as segundas partes dos arquivos

2.9 – repete-se o passos 2.5 a 2.8 até a posição de leitura dos dois arquivos forem iguais ao seu tamanho.

3.0 – Repete-se os passos 2.5 a 2.9 até ((quantidade de arquivos\*2) -1) obtidos no passo 1.2.

3.1 - Após passo 3.0 é gerado um arquivo ordenado

REFERÊNCIA.

\PAAMergeSort\PAAMergeSort\controllers\ArquivoController.cs

Metodo: combinarArquivosSecundariosByKVetores2(int quantidadeArquivosCriados, int KVetores, string caminhoArquivoLeitura, string caminhoArquivoEscrita, byte[] bytesToRead)

**D – Link da implementação no GITHUB**

[**https://github.com/g3ferreira/REP-PAAMergeSort.git**](https://github.com/g3ferreira/REP-PAAMergeSort.git)

**E – Tutorial de compilação das questões 1, 2 e 4.**

**Questão 1:**

**Questão 2:**

**Questão 3:**

**F – Descrição dos resultados obtidos**

**Questão 1- Descreva em alto nível um algoritmo de ordenação externa por**

**intercalação (Merge-Sort) multivias como entrada;**

1. – É realizada a primeira leitura sobre o arquivo, quebrando-o em blocos do tamanho da memória interna.
   1. – Cada bloco é ordenado na memória interna
   2. – os blocos são intercalados, gerando blocos cada vez maiores até que todo arquivo esteja ordenado

**Questão 2 – Apresente a complexidade de tempo do seu algoritmo considerando o**

**ambiente de execução em memória secundária;**

Dado: 1 – Arquivo Principal de 20Mb e 5 Mb de memória interna

Temos:

N = Tamanho do Arquivo;

M = Tamanho memória interna;

X = Quantidade de arquivo ordenados;

Leitura no arquivo principal de: N/M

N = 20Mb

D

C

X= N/M ( 4 - 5Mb)

A

B

B1

A1

B(N-1)

A(N-1)

(X\*2) -1 repetições

C

AB

ABC

D

ABCD

**Questão 3 – Baseado em seu algoritmo, implemente um programa para ordenar**

**arquivos cujos registros são inteiros de 32 bits. O programa resultante (msort) dev**

**receber os seguintes parâmetros:**

**Questão 4 - Usando o programa implementado, execute e apresente gráficos dos**

**seguintes experimentos de desempenho (tempo e execução) (Válida somente com a entrega da questão 3).**