Fundação Centro de Análise Pesquisa e inovação – FUCAPI

Análise de Alorítmos

**Relatório do Trabalho Prático**

Jonas Cavalcante Macêdo

Milrak Pereira Pessoa

Rafael Soares da Costa

Yan Rodrigo da Silva Soares

Manaus

Maio de 2016

1. **Introdução**

Este trabalho apresenta uma análise feita em cima de um algorítmo de ordegenação multivias com armazenamento externo (utilizando a memória secundária), este método é utilizado quando temos muitos registros a serem ordenados e não existe espaço suficiente para a alocação em memória principal, sendo assim uilizaremos arquivos .txt com números inteiros de 32 bits gerados de forma aleatória, e vamos utilizar o método de ordenação Merge-Sort.

1. **Implementação**

Para demonstrámos a execução do algorítmo iremos utilizar a lingem c/c++ com o auxílio de algumas bibliotecas “padrões”, pois as mesmas já estão inclusas na IDE DevC, que será a tulizada no projeto.

Bibliotecas: stdio.h; stdlib.h; time.h; string.h; conio.h;

Iremos através dos ponterios alocar os números em memória principal de tal forma que não estressamos tanto o processador da máquina utilizada no experimento, pois iremos alterar somente o local para qual terminado ponteiro aponta, tendo no final uma lista de ponteiros ordenados e assim ordenando os resultados.

Para teste final iremos utilizar arquivos que ao final totalizaram 6GBs de espaço (no disco secundário), esses 6GBs estaram sendo divididos conforme parêmetro informado na execução do algorítmo, esses, a partir daí já começa o método de divisão e conquiesta utilizado no método de ordenação utilizado, pois iremos dividir o trabalho principal em trabalhos menores e assim ordenando os sub-resultados, para esse processo definimos uma constante na linha 15 do algorítmo.

#define TAMANHO 1288490190

Que informa ao programa o tamanho de 6GBs.

1. **Links no GitHub**

**Link 1** – Algorítmo de Ordenação.

<https://github.com/estilosoft/PAA>

**Link 2** – Algorítmo de Ordenação.

<https://github.com/estilosoft/PAA/blob/master/mergesort.c>

**Link 3** - Algorítmo para métrica de tempo.

<https://github.com/estilosoft/PAA/blob/master/Tempo%20da%20ordena%C3%A7%C3%A3o.cpp>

1. **Tutorial de Compilação**

A execução e testes foram realizadas em ambiente Windows 7 e podendo ser exetutado em outras plataformas, no entando que seja observado as biliotecas usadas no projeto, pode ser que ocorra imcompatibilidades na mudança de S.O. sendo necessário a utilização de uma biblioteca de terceiros que seja baseada na utilizada.

O projeto foi desenvolvido na IDE DevC, no entanto recomendamos a mesmo ide para compilação no entando não sendo obrigatório, se preferir pode ser utilizada outras ides, mas qualquer ide que tenha suporte a compiladores baseados em GCC pode ser tulizado.

Para compilação basta baixar o algorítmo no site do GitHub (informado no tópico anterior), ou clonar o mesmo através do aplicativo para desktop.

Antes de compilar o projeto se faz necessária a alteração de algumas linhas de comando para a nomeção de algumas variáveis, que iram guardar os “caminhos” dos arquivos de origem e os arquivos gerados.

Linha 112 -> arquivo= fopen("mergesort.txt", "a+");

Linha 127 -> arquivo = fopen("mergesort.txt", "rt");

Linha 128 -> arqjunta = fopen("juntaordenado.txt", "a+");

Após as seguintes alterações o prorama estará pronto para execução.

1. **Resultados**

Se todos os passo seguintos do tópico anterior estiverem sido seguidos de forma correta o algoritimo irá apresentar um arquivo final ordenado no tamanho informado (6GBs utilizados nesse exemplo), com os números ordenados. O programa irá pegar as partes dos arquivos e irá começar a ordenação, arquivo por arquivo, em seguida irá fazer a junção dos arquivos em um uníco arquivo gerando o txt final, por se tratar de um pacote de dados muito grande, o número de informaões repetidas é grande, pois não temos tantos numeros interiros de 32 bits para preencher 6Gbs de dados, mas por fim o algoritmo cumpre com o proposto no escopo do projeto.

1. **Tempo de execução**

O tempo para execusão dos teste foi relativamente satisfatório, pois tratamos de testar com pequenos pacotes de dados, e não com os 6GBs.

Após o avanço do projeto, fomos aumentando o número de dados nos pacotes até chegar no pacote final de 6Gbs, o tempo final varia de acordo com o desenpendo do procesador mas em média de 3 a 5 horas conseguimos fazer a execução completa dos dados, esse tempo variou quando mudamos de ambiente, pois testamos em mais de uma máquina.

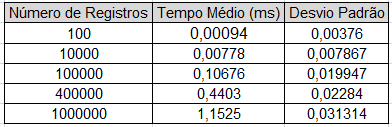
Catalogamos a execusão de teste em uma máquina com as espeficicações abaixo:

Processador: AMD E-300 APU with Ra Leon™ HD Graphics 1,30 GHz

Memória RAM: 2,00 GB (Utilizável 1,73 GB)

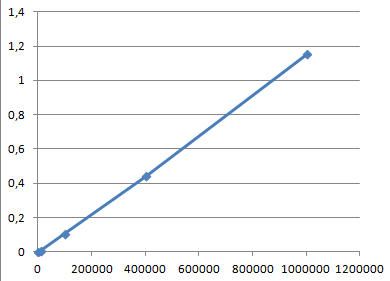
Tipo de Sistema: Sistema Operacional de 64Bits, processador com base de x64.

Sistema Operacional: Windows 10 Pro.



Número de Registro referente ao arquivo final. (com as junção dos subs-txts)

Tempo expresso em milisegundos.



Plotagem: Tempo x Número de dados

Complexidade:

Pior Caso -> Θ(n log 2n)

Médio Caso -> Θ(n log n)

Melhor Caso -> Θ(n)

Demostração:

Limite Superior

log(1) + log(2) + ... + log(n) <= log(n) + log(n) + ... + log(n)

= n\*log(n)

Limite Inferior

log(1) + ... + log(n/2) + ... + log(n) >= log(n/2) + ... + log(n)

>= log(n/2) + ... + log(n/2)

= n/2 \* log(n/2)

Apessar da grande demora quando utilizamos um grande arquivo para ordenação, o merge sort é um ótimo mecanismo de ordenação para o contexto de dividir para conquistar. Pois divide o problema principal em problemas menoras para asolução geral do problema.