Universidade Federal de Minas Gerais
DCC- Departamento de Ciência da Computação
Engenharia de Sistemas

PEDRO CAROLINO MALAQUIAS - 2016070115

TRABALHO PRÁTICO 0
ALGORITMOS E ESTRUTUDAS DE DADOS 3

BELO HORIZONTE

2018

1 INTRODUÇÃO

O objetivo desse trabalho é a resolução de um problema de ordenação, onde não se há espaço suficiente na memória RAM do computador para armazenar todas as linhas necessárias, se tornando assim, impossível de se resolver somente ordenando as linhas internamente. Nesse problema os únicos dados que sabemos são que todas as linhas têm um tamanho fixo, previamente dado, e que sua memória interna sempre irá armazenar no mínimo 2 linhas.

Desta forma, cheguei à conclusão que para a resolução do problema, tendo em vista meus conhecimentos prévios, era necessário realizar essa ordenação em um componente fora da memória RAM, ou seja, realizar uma ordenação externa. Essa ordenação externa foi feita pelo método de Intercalação Balanceada de Vários Caminhos, como citado acima a memória interna sempre irá armazenar no mínimo 2 linhas do arquivo, portanto essa foi a premissa utilizada para a implementação do método, comparar sempre de duas a duas linhas e ir alocando a menor dentre essa comparação na fita, até que todo o arquivo esteja ordenado.

2 MODELAGEM DO PROBLEMA

O problema se tratava de ordenar vários arquivos com uma dada quantidade de memória, portanto para a resolução do problema foi aplicado o método de intercalação balanceada de vários caminhos.

Intercalar significa combinar dois ou mais blocos ordenados em um único bloco ordenado. E fazendo isso uma certa quantidade determinada de vezes é possível ordenar qualquer arquivo. Fazendo assim a intercalação o método mais importante de ordenação externa.

A estratégia utilizada é bastante simples, no entanto:

- Quebrar o arquivo em blocos do tamanho de memória desejado.
- Ordenar cada bloco na memória interna, podendo ser usado uma ordenação interna nesse auxilio.
- Intercalar os blocos ordenados, fazendo várias passadas no arquivo
- A cada passada são criados blocos ordenados cada vez maiores, até que todo o arquivo esteja ordenado.

Podemos visualizar melhor essa estratégia verificando as figuras abaixo.

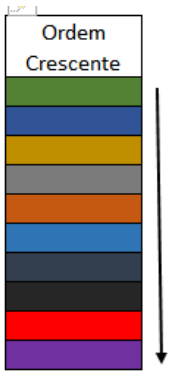

Figura 2-1: Ordem de Cores

Dado um arquivo de entrada qualquer, temos que ordena-lo de acordo com a ordem crescente das cores dado pela figura 2.1. Para isso, realizaremos o passo a passo do algoritmo de intercalação balanceada de vários caminhos.

O primeiro passo é quebrar o arquivo em bloco de memória desejado, como no problema foi quebrado em blocos de memória de tamanho 2, iremos fazer o mesmo aqui.

Figura 2-2: Criação de Blocos Ordenados de tamanho 2

Pelo Algoritmo você deve ter o dobro de fita em relação ao tamanho de memoria, como aqui temos tamanho de memoria 2, precisaremos de 4 fitas.

Nessa primeira etapa, lemos os arquivos de entrada 2 a 2 e vamos ordenando eles e intercalando uma vez a fita 1 outra vez a fita 2 até lermos o arquivo inteiro. Nessa etapa não utilizamos as fitas 3 e 4.

Após finalizar a construção dos blocos ordenados, fazemos a primeira intercalação.

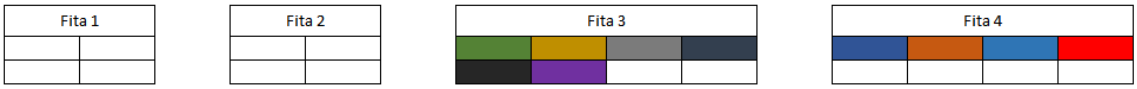

Figura 2-3: Primeira Intercalação dos Blocos Ordenados

Onde intercalamos os blocos das fitas 1 e 2, e escrevemos nas fitas 3 e 4.

Figura 2-4: Segunda Intercalação dos Blocos Ordenados

Refazemos essa operação de intercalação dos blocos, até que sobre somente um único bloco ordenado, que será a resposta correta.

Figura 2-5: Terceira Intercalação dos Blocos Ordenados

Podemos perceber que nesse algoritmo a cada passada da intercalação o tamanho do bloco ordenado vai dobrando até que se chegue na resposta correta.

3 ANÁLISE TEORICA DO CUSTO ASSINTÓTICO

Nesta seção analisaremos os custos teóricos de tempo e espaço do algoritmo.

3.1 ANÁLISE TEORICA DO CUSTO ASSINTOTICO DE TEMPO

Uma boa medida de complexidade de um algoritmo de ordenação por intercalação é o número de vezes que um item é lido ou escrito na memória auxiliar. Tendo em vista isso, temos que calcular quantas passadas o algoritmo realiza.

Para calcularmos a quantidade de passadas de um arquivo contendo n registros ,uma memória interna de m palavras e f fitas. A cada passada sobre o arquivo é produzido blocos de tamanho n/m ordenados. Tendo que P(n) é uma função de complexidade temos que:

$$P(n) = \log_f \frac{n}{m}.$$

Sendo essa função a complexidade teórica do algoritmo.

Como no problema utilizamos um número pré-estabelecido de 4 fitas e 2 memorias, a complexidade do algoritmo ficaria P(n) = log n/2 na base 4.

3.2 ANÁLISE TEORICA DO CUSTO ASSINTOTICO DO ESPACO

O custo para realização desse algoritmo é relativamente baixo e seu valor varia em relação a sua quantidade de memória utilizada.

Foi gasto para fabricação desse algoritmo 2 vetores (vet1 e vet2) com o tamanho da memória interna, para fazer a manipulação de dados entre as fitas, e as próprias fitas.

O custo dos vetores em todo o algoritmo seria O(registros*memoria)

4 ANÁLISE DOS EXPERIMENTOS

A análise experimental da implementação pode ser visualizada pela Tabela 1. Cada instancia foi gerada 4 vezes e foi retirada a média para obtermos um tempo com maior confiabilidade e precisão. Tendo em vista que os testes foram realizados em um computador com processador Core i5-7200 8GB de RAM e 2.71GHz.

Resultados		
Arquivo	Tempo(ms)	Memoria(kb)
10x10-1	19,297	0,0195
10x10-2	12,418	0,0195
10x10-3	37,546	0,0195
10x40-1	12,502	0,0195
10x40-2	9,562	0,0195
10x40-3	29,144	0,0195
10x80-1	72,73	0,0195
10x80-2	30,567	0,0195
10x80-3	28,5655	0,0195
10x160-1	18,033	0,0195
10x160-2	11,966	0,0195
10x160-3	308,4195	0,0195
10x320-1	48,11	0,0195
10x320-2	119,639	0,0195
10x320-3	17,04	0,0195
10x640-1	26,604	0,0195
10x640-2	17,872	0,0195
10x640-3	22,963	0,0195

Resultados		
Arquivo	Tempo(ms)	Memoria(kb)
20x10-1	9,49	0,0391
20x10-2	36,66	0,0391
20x10-3	14,863	0,0391
20x40-1	10,72	0,0391
20x40-2	12,768	0,0391
20x40-3	12,912	0,0391
20x80-1	14,677	0,0391
20x80-2	45,766	0,0391
20x80-3	16,472	0,0391
20x160-1	25,22	0,0391
20x160-2	12,785	0,0391
20x160-3	19,453	0,0391
20x320-1	28,296	0,0391
20x320-2	35,1213	0,0391
20x320-3	24,564	0,0391
20x640-1	29,485	0,0391
20x640-2	49,45	0,0391
20x640-3	42,355	0,0391

Resultados		
Arquivo	Tempo(ms)	Memoria(kb)
80x10-1	21,455	0,1562
80x10-2	8,878	0,1562
80x10-3	9,4515	0,1562
80x40-1	10,154	0,1562
80x40-2	18,515	0,1562
80x40-3	9,125	0,1562
80x80-1	39,754	0,1562
80x80-2	77,145	0,1562
80x80-3	22,782	0,1562
80x160-1	24,775	0,1562
80x160-2	29,846	0,1562
80x160-3	59,545	0,1562
80x320-1	16,516	0,1562
80x320-2	27,55	0,1562
80x320-3	11,846	0,1562
80x640-1	24,5163	0,1562
80x640-2	22,56	0,1562
80x640-3	18,584	0,1562

Tabela 1: Resultados Obtidos

Podemos constatar que a medida que o tamanho da linha dobra, o tamanho de memória utilizada também dobra de modo praticamente linear. Constatando assim a nossa afirmação da analise teórica de espaço.

Visualizamos também que o aumento da quantidade de linhas não altera a memória utilizada, o que nos faz perceber que mesmo fazendo mais intercalações a memória utilizada é a mesma.

Os arquivos de linha tamanho 40 não conseguiram ser compilados causando erros de segmentation fault e double free or corruption, então não foram considerados na análise dos dados.

5 CONCLUSÃO

Neste trabalho, foi realizado o algoritmo de ordenação externa (Intercalação Balanceada de Vários Caminhos), para a resolução do problema falta de memória na ordenação de linhas escritas por macacos.

Os resultados obtidos foram satisfatórios, a maioria dos testes foram bem-sucedidos exceto os testes para 40 caracteres nas linhas. Acredito eu, que estes estejam com problema na alocação ou no \0 dos vetores auxiliares. Tendo em vista que eles estão sendo ordenados normalmente e com problema na hora de escreve-los no arquivo de saída, onde ocorre segmentation fault, ou double free or corruption.