Trabalho Prático 1: Ordenação de Arquivos Grandes de Registros

[1]Alexandre Aparecido Scrocaro Junior R.A.: 2135485, [2]Pedro Acácio Rodrigues R.A.: 2135655
Universidade Tecnológica Federal do Paraná – UTFPR
COCIC – Coordenação do Curso de Bacharelado em Ciência da Computação
Campo Mourão, Paraná, Brasil
[1]alexandre.2001@alunos.utfpr.edu.br
[2]pedrorodrigues.2019@alunos.utfpr.edu.br

Resumo

Esse projeto tem como objetivo explorar uma solução para ordenar arquivos grandes com registros do tipo ITEM_VENDA. Porém, há um empecilho para a resolver o problema: a memória RAM é limitada a alguns valores que o professor definiu, e devemos testá-los para preencher as tabelas contidas no tópico de análise(2).

1. Modularização

Para a implementação do projeto, utilizamos os TADs recomendados (BufferEntrada e BufferSaida), além de outro TAD auxiliar para manuseamento do arquivo grande, o TAD_Arquivo. Ademais, criamos structs de vetor de buffers e de buffers, sendo a struct buffer utilizada para entrada e saída, as utilidades de suas variáveis estão comentadas em código.

- TAD_Arquivo:
  - Dividir:
    Abre o arquivo grande e o divide em K pedaços já ordenados;
  - Ordenação:
    Foi utilizado o quickSort para a ordenação dos arquivos;

- TAD_BufferEntrada:
  - Criar:
    Aloca um buffer que recebe os registros do arquivo (já dividido em arquivos menores) e inicializa suas variáveis;
  - Proximo:
    Apenas retorna o próximo registro do buffer;
  - Consumir:
    Verifica se o buffer de entrada está vazio, se estiver o reenche. Consome o primeiro registro do buffer(que vai ser sempre o menor, já que o TAD_Arquivo o organiza desse modo);
  - Vazio:
    Verifica se o arquivo está vazio;
  - Destruir:
    desaloca os buffers;

- TAD_BufferSaida:
    - Criar:
        Aloca um buffer (de saída) e inicializa suas variáveis;
    - Inserir:
        Insere um registro no buffer de saída e, se o buffer de saída estiver cheio, escreve os registros no arquivo de saída;
    - Despejar:
        Escreve os registros no arquivo de saída quando o buffer está cheio;
    - Destruir:
        Desaloca os buffers;

2. Análise

Fazendo uma análise das tabelas abaixo (Tabela 1, Tabela 2, Tabela 3 e Tabela 4), sabe-se que, ao aumentar o Buffer de Saída e a memória RAM, o tempo de execução do problema também diminui. (Nossos testes foram executados em um HDD).
Para fazer a análise dos testes feitos, iremos verificar as tabelas individualmente. Todos os testes foram executados em um HDD.

- **Tabela 1:**
    Nessa tabela, pode-se perceber que ao aumentar a capacidade do buffer de saída, o tempo diminuiu; isso também ocorreu ao aumentar a quantidade de memória RAM.

		S		
		B/8	**B/4**	**B/2**
B	**8388608(8MB)**	19s	17s	14s
	16777216(16MB)	13s	11s	9s
	33554432(32MB)	12s	13s	12s

TABELA 1: Tempo De Execução Para Ordenar Arquivo com 256000 Registros

- **Tabela 2:**
    Na segunda tabela, vimos que os resultados foram parecidos, independente dos diferentes fatores que os buffers foram submetidos.

		S		
		B/8	**B/4**	**B/2**
B	**16777216(16MB)**	25s	25s	23s
	33554432(32MB)	25s	26s	27s
	67108864(64MB)	25s	25s	30s

TABELA 2: Tempo De Execução Para Ordenar Arquivo com 512000 Registros

- **Tabela 3:**

  Ao analisar essa tabela, percebe-se que, ao aumentar o tamanho máximo do buffer de saída, o tempo também aumenta. Tal acontece independente da RAM utilizada. Também, ao aumentar a RAM, aumenta o tempo de execução.

		S		
		B/8	B/4	B/2
B	67108864(64MB)	74s	78s	88s
	134217728(128MB)	97s	104s	117s
	268435456(256MB)	173s	189s	213s

TABELA 3: Tempo De Execução Para Ordenar Arquivo com 921600 Registros

- **Tabela 4:**

  A última tabela é similar, em questão de análise, à Tabela 3, mas com valores maiores

		S		
		B/8	B/4	B/2
B	67108864(64MB)	142s	155s	176s
	134217728(128MB)	195s	217s	284s
	268435456(256MB)	355s	376s	407s

TABELA 4: Tempo De Execução Para Ordenar Arquivo com 1572864 Registros

*observação: após as tabelas serem preenchidas, descobrimos alguns bugs no programa, tentamos consertar, não conseguimos compilar mais o programa por falhas de segmentação.

3. Distribuição do projeto

O trabalho não foi dividido em partes para os dois alunos, ambos participaram de todas as etapas de produção do trabalho, sendo realizado da seguinte forma:

- Código: para a interpretação, elaboração da lógica e desenvolvimento dos arquivos(.c e .h), nos reunimos pelo aplicativo Discord usando chamadas de voz e compartilhamento de tela;

- Relatório: para escrever o presente documento, também fizemos uso do discord para chamadas de voz e da ferramenta online Docs, do google; desse modo conseguimos redigir o relatório de forma simultânea.