Relatório — Projeto merge_sort

Nome: Bruno Videira Pinho
DRE: 119.161.539

Objetivo

Este projeto implementa um procedimento de **Ordenação Externa** para arquivos CSV muito grandes (superiores a memória RAM do computador), utilizando o algoritmo Merge Sort adaptado.

Estrutura de Diretórios

```
merge_sort/
├── csv_utils.py      # Arquivo
├── full.csv          # Arquivo csv 20 Gb (não vai estar o repo)
├── merge_sort.py     # Arquivo principal contendo a função ext_merge_sort
├── mini.csv          # Arquivo csv para fazer testes rápidos
├── test_sorted.csv   # Arquivo teste csv ordernado pelo algoritmo
└── test.csv          # Arquivo de teste com 0.05 Gb
```

Principais Módulos e Funções

- **merge_sort.py**: Implementa toda a lógica do Merge Sort Externo, incluindo divisão em runs, ordenação interna, merge externo e limpeza de arquivos temporários.
- **csv_utils.py**: Utilitários para manipulação de arquivos CSV, geração de arquivos de teste, contagem de linhas, etc.

Funções do módulo

- `ext_merge_sort(file_path, key_column, order)`: Função principal. Recebe o caminho do arquivo CSV, a coluna-chave (nome ou índice) e a ordem ('asc' ou 'desc'). Realiza toda a ordenação externa.
- `split_csv_run_files(file_path, key_index)`: Divide o arquivo CSV em "runs" (partes menores), cada uma ordenada em memória e salva como arquivo temporário.
- `merge_sort_run_files(run_paths, key_index, order)`: Ordena cada run individualmente em memória usando Merge Sort na ordenação definida.
- `merge_sorted_runs(file_path, run_paths, key_index, order)`: Realiza o merge externo dos runs ordenadas, produzindo o arquivo final ordenado.
- `make_sorted_file(file_path, files, key_index, order)`: Implementa o merge dos arquivos temporários, escrevendo o resultado final e apagando os arquivos temporários.

Fluxo do Algoritmo

1. **Divisão em Runs**
   O arquivo CSV é lido em blocos (chunks) que cabem na memória RAM. Cada bloco é ordenado em memória e salvo como arquivo temporário ("run").

2. **Ordenação Interna**
   Cada run é ordenado usando Merge Sort (implementação recursiva em memória).

3. **Merge Externo**
   Todos os runs ordenados são abertos simultaneamente. O menor (ou maior, dependendo da ordem) registro entre os arquivos é selecionado e escrito no arquivo final ordenado. O processo continua até esvaziar todos os runs.

4. **Limpeza**
   Todos os arquivos temporários e diretórios auxiliares são removidos ao final do processo.

Parâmetros da função principal

- **file_path**: Caminho do arquivo CSV a ser ordenado (ou o nome do arquivo no diretório da função).
- **key_column**: Nome ou índice da coluna usada como chave de ordenação.
- **order**: 'asc' para ordem crescente, 'desc' para ordem decrescente.

Complexidade de Tempo e Espaço

- **Tempo**:
  - Divisão em runs: O(n), onde n é o número de linhas.
  - Ordenação interna: O(m log m) por run, onde m é o tamanho do chunk.
  - Merge externo: O(n log r), onde r é o número de runs.
- **Espaço**:
  - Utiliza espaço proporcional ao tamanho do maior chunk em memória e espaço extra em disco para os arquivos temporários.

Observações e Limitações

- O algoritmo depende do espaço em disco para criar arquivos temporários.
- Suporta qualquer coluna como chave, desde que todos os registros possuam valor numérico.
- Remove automaticamente os arquivos temporários ao final do processo.
- O arquivo final é salvo com o sufixo `_sorted.csv`.
- Funções secundárias foram geradas para geração de arquivos csv grandes.
- Não foi possível testar com arquívos superiores a memória RAM da minha máquina (tenho 32 Gb).
  - Para gerar um arquivo de 20 Gb temorei 5 horas (usando as funções segundárias).

Conclusão

O projeto cumpre o objetivo de ordenar grandes arquivos CSV utilizando Merge Sort Externo, sendo capaz de lidar com arquivos maiores que a memória RAM disponível, com limpeza automática dos arquivos temporários e flexibilidade na escolha da chave e ordem de ordenação.