Relatório: Ordenação Externa por MergeSort em Python

Equipe

6 de julho de 2025

1 Introdução

Este relatório descreve a implementação de um algoritmo de ordenação externa baseado no MergeSort para arquivos CSV que excedem a capacidade da memória RAM. O código foi desenvolvido em Python e utiliza técnicas de divisão e conquista combinadas com manipulação eficiente de arquivos temporários.

2 Prompts ultilizados

-no contexto de ciência da computação e Programação eu quero que você divida esta tarefa em tarefas mais simples como um passo a passo sem mudar o resultado final e com atenção a detalhes, não invente informações, e me diga se é melhor fazer este trabalho em python ou c # : "Comando da questão-

-siga o passo a passo que voce fez e faça o codigo completo em python , faça um teste para garantir que ele está correto, se tiver algum erro evidencie.-

e a partir disto fiz modificações para corrigir os erros e tentar rodar da melhor forma possível

3 Estrutura do Código

3.1 Divisão Modular

O código está organizado em quatro funções principais:

- `external_sort`: Função principal que coordena o processo
- `split_and_sort`: Divide o arquivo em chunks ordenados
- `save_sorted_chunk`: Ordena e salva chunks temporários
- `merge_runs`: Mescla os arquivos temporários ordenados

4 Descrição das Rotinas

4.1 Função Principal (`external_sort`)

Coordena todo o processo:

- Chama `split_and_sort` para criar runs ordenados
- Chama `merge_runs` para combinar os runs
- Remove arquivos temporários

4.2 Divisão e Ordenação (`split_and_sort`)

- Lê o arquivo CSV em chunks de tamanho `buffer_size`
- Para cada chunk:
  - Ordena em memória usando `list.sort()`
  - Salva em arquivo temporário

4.3 Mesclagem (`merge_runs`)

- Implementa um merge k-way usando heap implícito
- Mantém um registro por arquivo temporário na memória
- Sempre pega o menor/maior elemento disponível

5 Estruturas de Dados e Complexidades

5.1 Estruturas Utilizadas

- **Listas Python**: Para armazenar chunks em memória
- **Heap implícito**: Para o merge k-way (implementado como lista ordenada)
- **Arquivos temporários**: Para armazenar runs ordenados

5.2 Complexidade de Tempo

- **Fase de divisão**: $O(n \log m)$ onde m é o tamanho do buffer
- **Fase de merge**: $O(n \log k)$ onde k é o número de runs
- **Total**: $O(n \log n)$

5.3 Complexidade de Espaço

- **Memória principal**: $O(m)$ onde m é `buffer_size`
- **Disco**: $O(n)$ para arquivos temporários

6 Problemas e Observações

6.1 Desafios Encontrados

- Gerenciamento preciso de arquivos temporários
- Balanceamento entre tamanho do buffer e número de runs
- Tratamento de cabeçalhos nos arquivos CSV

6.2 Soluções Implementadas

- Uso de `tempfile` para criação segura de temporários
- Verificação explícita de ordem durante o merge
- Suporte a chaves de ordenação por nome ou índice

7 Testes e Validação

7.1 Método de Teste

- Geração automática de arquivos CSV de teste
- Verificação de ordenação pós-processamento
- Teste com diferentes tamanhos de buffer

7.2 Resultados

- Ordenação correta em todos os casos de teste
- Performance adequada para arquivos grandes
- Uso de memória conforme o buffer especificado

8 Conclusão

A implementação demonstrou ser eficaz para ordenação de grandes arquivos CSV que não cabem na memória principal. O algoritmo segue a abordagem clássica de ordenação externa com as seguintes características:

- Escalabilidade para arquivos muito grandes

- Controle preciso do uso de memória
- Flexibilidade na escolha da chave de ordenação