UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE COMPUTAÇÃO

JULIO AUGUSTO DE CASTILHOS BORGES
VICENTE PIANEZZOLA

Steaming Sort Report

Relatório apresentado como requisito parcial para a obtenção de conceito na Disciplina de Classificação e Pesquisa de Dados.

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2025

SUMÁRIO

1 INTRODUÇÃO

O Steam é uma loja, rede social e plataforma de jogos em geral (mas não é limitada a jogos, também). Por exemplo, contém 394.733 programas disponíveis. Este número é bem variável, pois jogos são removidos e adicionados ao longo do tempo (somente neste ano, 7.099 programas foram adicionados à loja).

Já que nem todos os dados estão disponíveis pela API do Steam, procuramos por fontes externas e encontramos um dataset de Outubro de 2024 com uma mistura dos dados acessíveis pela API e também com dados coletados por scrapers: steam-insights.

Como mencionado anteriormente, nosso problema envolve coletar e manipular os dados de forma a possibilitar a ordenação alfanumérica de lançamento dos jogos, pela quantidade ou qualidade de avaliações, etc. Além disso, é preciso mostrar essas informações de forma organizada e paginada para o usuário, que não deve precisar esperar para ter um resultado. Para tanto, é preciso manter arquivos de índices atualizados que permitam a busca, deleção e inserção eficazes para os novos dados.

Por meio do nosso programa, esperamos possibilitar que o usuário acesse um enorme catálogo de jogos e programas facilmente. Ao usar nossa solução, comparar preços, tags, e até mesmo avaliações da comunidade é simples. Em outras palavras, uma maneira rápida e intuitiva de acessar e avaliar informações sobre jogos.

2 DADOS

De início, nosso dataset foi obtido a partir de steam-insights, no entanto, rapidamente descobrimos que esta fonte continha muitos dados ruins, repetidos, em línguas diferentes, em moedas diferentes ou até mesmo faltando ou não aderindo ao formato `csv`. Por isso, utilizamos um scraper, além de diversos scrpits feitos sob medida e ferramentas como `rbql` (Rainbow Query Language) para auxiliar na arrecadação, coleta e manipulação inicial dos dados. Desta maneira, conseguimos integrar o preço em Reais para todos os jogos, além de ter todas as tags e categorias sem repetição e padronizadas. Optamos por não normalizar e nem omitir dados com caracteres especiais a menos que a entrada no banco de dados fosse composta apenas por tais símbolos. Esta decisão foi tomada, pois, para os fins deste trabalho seria inútil guardar dados que não poderiam ser pesquisados por um teclado convencional.

3 ARQUITETURA E IMPLEMENTAÇÃO

A especificação do trabalho determina que salvemos arquivos binários, portanto, optamos por salvar árvores B+, além de um dicionário em memória. Esta estrutura foi escolhida pela facilidade do uso e velocidade proporcionados …

4 ESTRUTURAS DE DADOS E ORGANIZAÇÃO DOS ARQUIVOS

- **Binary File Structure (Requirement #3):**
  - Detail the exact format of your custom binary files. How is a "game"record stored? Is it a fixed-length or variable-length struct?
  - Provide a table or diagram showing the fields, their data types, and their size in bytes (e.g., `game_id`: int (4 bytes), `name`: char[100] (100 bytes), `price`: float (4 bytes)).
  - Explain why you chose this structure.
- **Indexing Method (Requirement #3):**
  - State which index structure you implemented (e.g., B-Tree, B+ Tree, TRIE/-PATRICIA).
  - Explain why you chose it for your data (e.g., "A B+ Tree was chosen to index game prices because it is efficient for range queries, such as finding all games between $10 and $20").
  - Describe which data field(s) are indexed (e.g., "An index was created for the `game_name` field to allow for fast text-based searches").
  - Explain how the index itself is stored (e.g., in a separate binary file) and how it's loaded/updated.
- **Data Persistence and Incremental Updates (Requirement #4):**
  - Explain how your application saves the binary data and indexes so they can be reused without re-importing the CSVs every time.
  - Describe the functionality that allows a user to add new data, fulfilling the "armazenamento incremental"requirement.

5 FUNCIONALIDADES DO SOFTWARE

O programa oferece ao usuário opções de busca no menu principal.
talvez adicionar imagens de exemplo?
Assim que uma busca é feita, é possível inverter ou até mesmo reordenar o resultado. Caso a busca seja extensa e ocupe mais de uma 'página' no terminal, o usuário pode navegar utilizando $\downarrow$ / $\uparrow$, w/s ou j/k (para quem está acostumado com o `vi`). Também é possível visualizar a quantidade total de jogos com cada tipo de tag e, por fim, é possível adicionar mais dados facilmente, os scripts para a coleta e processamento dos dados estão no repositório referente a este trabalho, ou são instaláveis por *package managers*.

6 CONCLUSÃO

A partir deste trabalho, conseguimos ter uma experiência prática com as diversas estruturas que vimos nas aulas de CPD. Além disso, tivemos a oportunidade de mexer com grandes quantidades de dados reais, que, ao contrário das nossas experiências em laboratório, se revelaram bem mais complicados de lidar do que simplesmente listas ordenadas aleatoriamente. Por fim, foi um trabalho extremamente produtivo e divertido
-por mais que estressante e cansativo nos momentos finais-

Obrigado por tudo, Prof. Dr. Wives!

REFERÊNCIAS

Já tinha usado o LaTeX, com referências .bib, mas não consegui fazer funcionar neste template, de qualquer maneira, todos os links estão funcionando.