



Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Trabalho Prático V

(Data de Entrega: 24/10)

A indústria de jogos digitais movimenta bilhões de dólares anualmente, sendo considerada um dos setores mais dinâmicos e lucrativos do entretenimento moderno. Plataformas como a **Steam** concentram milhares de títulos que atendem a públicos diversos, indo de clássicos consagrados, como *Max Payne* (lançado em 2011 pela Rockstar Games, com nota 89 no Metacritic), a sucessos contemporâneos, como *Oxygen Not Included* (2019, Klei Entertainment, nota 86 no Metacritic, com mais de 3,5 milhões de jogadores estimados).

Além disso, a variedade é ampla: jogos independentes como *Reigns: Game of Thrones* (2018), títulos de grande apelo competitivo como *Call of Duty® 4: Modern Warfare®* (2007, nota 92 no Metacritic) e experiências imersivas em realidade virtual como *Ragnarock* (2021). Esses dados mostram não apenas a diversidade de gêneros — **Ação, RPG, Simulação, Indie, Casual, Esportes**, entre outros — mas também a pluralidade de estúdios envolvidos, desde pequenos desenvolvedores independentes até grandes publishers como **Activision** e **Devolver Digital**.

Neste trabalho, propomos a **análise e manipulação de dados extraídos da Steam**, explorando dimensões como preço, popularidade (número estimado de jogadores), avaliações críticas (Metacritic score), conquistas desbloqueáveis e suporte multilíngue. A partir desse conjunto de dados, buscamos identificar padrões de consumo, tendências do mercado e fatores que podem influenciar o sucesso de um jogo digital.

O arquivo `GAMES.CSV` contém um conjunto de dados extraídos do site Kaggle. Este conjunto de dados contém dados de mais de jogos publicados na plataforma Steam, com detalhes como AppID, nome, data de lançamento, número estimado de jogadores, preço, idiomas suportados, notas da crítica e dos usuários, número de conquistas, publicadoras, desenvolvedoras, categorias, gêneros e tags associadas a cada jogo. Tal arquivo deve ser copiado para a pasta `/tmp/`. Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta `/tmp/`.

A Tabela 1 apresenta os campos disponíveis e seus respectivos tipos de dados.






Campo	Tipo	Descrição
AppID	Numérico	Identificador único do jogo na Steam.
Name	Texto	Nome do jogo.
Release date	Data (texto)	Data de lançamento do jogo.
Estimated owners	Numérico	Estimativa do número de jogadores que possuem o jogo.
Price	Numérico (float)	Preço do jogo em dólares americanos.
Supported languages	Texto (lista)	Idiomas suportados pelo jogo.
Metacritic score	Numérico	Nota atribuída pela crítica especializada (Metacritic).
User score	Numérico	Nota atribuída pelos usuários da Steam.
Achievements	Numérico	Quantidade de conquistas disponíveis no jogo.
Publishers	Texto	Empresa(s) responsável(is) pela publicação do jogo.
Developers	Texto	Empresa(s) ou estúdio(s) responsável(is) pelo desenvolvimento.
Categories	Texto (lista)	Categorias associadas ao jogo (ex.: single-player, multi-player).
Genres	Texto (lista)	Gêneros do jogo (ex.: Ação, RPG, Simulação).
Tags	Texto (lista)	Conjunto de palavras-chave atribuídas pelos usuários.

Tabela 1: Campos do conjunto de dados da Steam.

Regras Gerais

- A partir deste trabalho, você deve utilizar as soluções em **Java** ou **C**, implementados no TP-04.
- Nos exercícios de ordenação ou estruturas de dados, se dois registros tiverem a mesma chave de pesquisa, eles serão ordenados pelo atributo APPID.
- para cada questão, devemos submeter apenas um arquivo (.java ou .c). Essa regra será necessária para a submissão de trabalhos no Verde e no identificador de plágio utilizado na disciplina.
- Para cada exercício: faça (entende-se análise, implementação e comentários), teste (várias vezes) e submeta no Verde. Os exercícios não testados/comentados serão penalizados.
- A correção será realizada automaticamente pelo sistema Verde. Entretanto, você poderá ser entrevistado durante as aulas de laboratório e deverá demonstrar pleno domínio sobre o código desenvolvido. A utilização de ferramentas de IA generativa na elaboração do trabalho implicará em nota zero.

Questões

1.  **Pesquisa Binária em Java:** Faça a inserção de alguns registros no final de um vetor e, em seguida, faça algumas pesquisas. A chave primária de pesquisa será o atributo **Name**. A entrada padrão é composta por duas partes onde a primeira é igual a entrada do TP-04. A segunda parte é composta por várias linhas. Cada uma possui um elemento que deve ser pesquisado no vetor. A última linha terá a palavra FIM. A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se existe cada um dos elementos pesquisados. Lembre-se de ordenar os elementos para execução da pesquisa. Além disso, crie um arquivo de log na pasta corrente com o nome `suamatricula_binaria.txt` com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação `'\t'`.
2.  **Ordenação por Seleção em C:** Usando vetores, implemente o algoritmo de ordenação por seleção considerando que a chave de pesquisa é o atributo **Name**. A entrada e a saída padrão são iguais as da primeira questão, contudo, a saída corresponde aos registros ordenados. Além disso, crie um arquivo de log na pasta corrente com o nome `suamatricula_selecao.txt` com uma única linha contendo sua matrícula, número de comparações (entre elementos do *array*), número de movimentações (entre elementos do *array*) e o tempo de execução do algoritmo de ordenação. Todas as informações do arquivo de log devem ser separadas por uma tabulação `'\t'`.
3.  **Heapsort em Java:** Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Heapsort, fazendo com que a chave de pesquisa seja o atributo **Estimated_owners**. O nome do arquivo de log será `suamatricula_heapsort.txt`. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o APPID do GAME.)
4.  **Quicksort em C:** Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Quicksort, fazendo com que a chave de pesquisa seja o atributo **Release_date**. O nome do arquivo de log será `suamatricula_quicksort.txt`. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o APPID do GAME.)
5.  **Mergesort em Java:** Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Mergesort, fazendo com que a chave de pesquisa seja o atributo **Price**. Ao final imprima os 5 preços mais caros e 5 mais baratos. Lembre-se de criar o do arquivo de log com o nome `suamatricula_mergesort.txt`. (Lembre-se: em caso de empate, o critério de ordenação deverá ser o APPID do GAME.)