Sistema de Adoção de Gatos

Gabriel R. C. Nunes[1], **Samuell C. O. Aguiar**[1]

[1]Instituto de Ciências Exatas e Aplicadas – Universidade Federal de Ouro Preto (UFOP)

gabriel.candido@aluno.ufop.edu.br, samuell.aguiar@aluno.ufop.edu.br

1. Introdução

Este trabalho apresenta o desenvolvimento de um sistema de adoção de gatos, implementado em Java com foco em manipulação de arquivos. O sistema simula o funcionamento de um abrigo, oferecendo funcionalidades de cadastro, busca, ordenação externa, estatísticas e controle de adoções, sem uso de estruturas em memória para armazenamento permanente.

2. Objetivo da Aplicação

A aplicação tem como objetivo possibilitar a prática de conceitos de algoritmos e estruturas de dados II, com ênfase na manipulação direta de arquivos texto. As funcionalidades simulam um sistema real de adoção animal, com foco em persistência, ordenação e busca eficiente de dados.

3. Entidades do Sistema

3.1. Gato

Cada gato possui os seguintes atributos:

- **ID**: Identificador único numérico.
- **Nome, Raça, Idade, Sexo**: Características básicas.
- **Status**: Define se está disponível ou já foi adotado.

3.2. Adoção

Cada adoção está vinculada a um gato e contém:

- **ID da Adoção**.
- **Nome do adotante**.
- **Data da adoção**.
- **ID do gato adotado**.

4. Funcionalidades da Aplicação

4.1. Cadastro e Manipulação de Arquivos

O sistema permite a criação e manipulação de arquivos texto contendo dados dos gatos e das adoções. As opções incluem:

- Cadastro manual de gatos com entrada pelo usuário.
- Geração automática de registros de gatos com dados aleatórios.
- Registro de adoções atualizando o status do animal.
- Armazenamento persistente em `gatos.txt` e `adocoes.txt`.

4.2. Geração e Ordenação Interna

Para fins de teste e organização, a aplicação oferece:

- Geração de base de dados desordenada.
- Ordenação interna dos dados via algoritmo **BubbleSort**, aplicada somente quando os dados estão em memória.

4.3. Busca de Registros

Foram implementados dois tipos de busca:

- **Busca Sequencial**: Varre o arquivo linha a linha até localizar o ID desejado.
- **Busca Binária**: Exige arquivo previamente ordenado, oferecendo maior desempenho.
- Ambos os métodos geram logs com número de comparações e tempo de execução.

4.4. Relatórios Estatísticos

O sistema gera um resumo contendo:

- Número total de gatos cadastrados.
- Quantidade de gatos disponíveis para adoção.
- Quantidade de gatos já adotados.

4.5. Ordenação Externa e Intercalação

Duas abordagens de ordenação externa foram implementadas:

- **Seleção Natural**: Gera partições ordenadas com base em um buffer de memória limitado. Registros fora de ordem são armazenados em um reservatório temporário e processados em ciclos.
- **Árvore Binária de Vencedores**: Intercala partições geradas comparando os menores elementos simultaneamente, utilizando uma árvore para selecionar o menor valor a cada etapa.
- Os tempos de execução e quantidade de partições são registrados em arquivos de log, como `Log_Selecao_Natural.txt`.

5. Avaliação Experimental dos Métodos de Ordenação

5.1. BubbleSort

Tamanho do Arquivo	Tempo de Execução (s)
10 registros	0,047 s
100 registros	0,572 s
1.000 registros	6,154 s
10.000 registros	170,186 s

Table 1. Tempo de execução do BubbleSort para diferentes tamanhos de arquivo

A ordenação utilizando BubbleSort apresentou desempenho satisfatório apenas para bases muito pequenas. Conforme mostrado na Tabela 1, o tempo de execução cresce exponencialmente com o aumento do número de registros. Para 10.000 registros, o tempo ultrapassa 170 segundos, o que torna o método inviável em situações com maior volume de dados. Apesar de simples, o BubbleSort evidencia suas limitações em eficiência, sendo útil apenas em contextos controlados e didáticos.

5.2. Ordenação com Seleção Natural + Árvore de Vencedores

5.2.1. Seleção Natural

Tamanho do Arquivo	Tempo de Execução (s)
10 registros	0,009 s
100 registros	0,128 s
1.000 registros	0,748 s
10.000 registros	6,924 s

Table 2. Tempo de execução da Seleção Natural para diferentes tamanhos de arquivo

A fase de geração de partições via Seleção Natural demonstrou um desempenho significativamente superior ao BubbleSort. Mesmo com 10.000 registros, o tempo de execução foi inferior a 7 segundos, o que mostra a adequação do método para arquivos grandes. Esse resultado comprova a eficiência da Seleção Natural na criação de partições ordenadas utilizando um buffer de memória limitado.

5.2.2. Árvore de Vencedores

Tamanho do Arquivo	Tempo de Execução (s)
10 registros	0,004 s
100 registros	0,027 s
1.000 registros	0,256 s
10.000 registros	1,854 s

Table 3. Tempo de execução da Árvore de Vencedores para diferentes tamanhos de arquivo

A etapa de intercalação utilizando a Árvore Binária de Vencedores também se mostrou eficiente, com tempos de execução baixos em todos os testes. Como pode ser observado na Tabela 3, mesmo para 10.000 registros, o tempo total foi inferior a 2 segundos. A estrutura de árvore permitiu comparar os menores elementos entre várias partições de forma otimizada, concluindo a ordenação de forma eficiente.

Com base nos testes realizados, concluimos que a combinação entre Seleção Natural e Árvore de Vencedores oferece uma solução robusta para ordenação de grandes volumes de dados em arquivos. Enquanto o BubbleSort possui implementação simples, sua baixa escalabilidade compromete seu uso prático. Por outro lado, a ordenação externa demonstrou excelente desempenho e viabilidade, sendo a abordagem mais indicada para aplicações reais que lidam com grandes arquivos.