Simulador de Reality Show: Uma Abordagem Baseada em Estruturas de Dados para Modelagem de Interações Sociais

Pedro David Rocha Saldanha[1], **Pedro Vinícius Medeiros Crescencio**[1]

[1]Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal Rural do Semi-Árido (UFERSA) – Mossoró, RN, Brasil
Universidade do Estado do Rio Grande do Norte (UERN) – Mossoró, RN, Brasil

pedrodavirochasaldanha@gmail.com,
pedro.mcrescencio@gmail.com

Abstract. *This paper presents a reality show simulator developed as a final project for a Data Structures course. Built with Python and Tkinter, the simulator models the dynamics of participants in a house, including daily events, social interactions, and eliminations. It showcases the application of data structures such as graphs for modeling participant characteristics, matrices for managing relationships, and lists/dictionaries for efficient data handling. The system provides a detailed event log and real-time updates of participant status.*

Resumo. *Este artigo apresenta um simulador de reality show desenvolvido como projeto final da disciplina de Estruturas de Dados. Construído com Python e Tkinter, o simulador modela a dinâmica de participantes em uma casa, incluindo eventos diários, interações sociais e eliminações. Demonstra a aplicação de estruturas de dados como grafos para modelagem de características de participantes, matrizes para gerenciamento de relações, e listas/dicionários para manipulação eficiente de dados. O sistema oferece um log detalhado de eventos e atualizações em tempo real do status dos participantes.*

1. Introdução

1.1. Contextualização do Problema

Simulações computacionais têm se mostrado ferramentas poderosas para modelar sistemas complexos e compreender suas dinâmicas internas. No contexto de programas de reality show, a interação entre os participantes, suas características individuais e os eventos diários geram um ambiente de alta complexidade e imprevisibilidade. A modelagem dessas interações pode ser um desafio, exigindo o uso eficiente de estruturas de dados para representar e manipular o grande volume de informações envolvidas.

1.2. Justificativa da Escolha do Tema

A escolha do tema ”Simulador de Reality Show”para o projeto final da disciplina de Estruturas de Dados justifica-se pela sua capacidade de integrar diversos conceitos abordados ao longo do curso. A dinâmica de um reality show permite a aplicação prática de grafos para modelar relações, matrizes para gerenciar afinidades, e estruturas de dados como listas e dicionários para organizar participantes e eventos. Além disso, o caráter lúdico e intuitivo do tema facilita a compreensão das funcionalidades e a visualização dos resultados da aplicação das estruturas. O projeto propõe um desafio interessante na representação e manipulação de dados que refletem o comportamento social e as regras de um jogo complexo.

1.3. Objetivo Geral do Projeto

O objetivo geral deste projeto é desenvolver um simulador de reality show utilizando Python e Tkinter, que seja capaz de modelar e simular as interações entre participantes, a ocorrência de eventos diários e o processo de eliminação, demonstrando a aplicação eficiente de conceitos de estruturas de dados na representação e manipulação dos elementos do jogo.

2. Fundamentação Teórica / Referencial Teórico

2.1. Conceitos e Tecnologias Utilizadas no Projeto

O desenvolvimento do simulador se baseou em Python como linguagem de programação principal, devido à sua versatilidade, facilidade de uso e vasta coleção de bibliotecas. Para a construção da interface gráfica do usuário (GUI), optou-se pela biblioteca **Tkinter**, que é a interface padrão do Python para o toolkit Tk e oferece uma maneira simples de criar aplicações desktop.
No que tange às estruturas de dados, o projeto faz uso extensivo de:

- **Grafos**: Utilizados para modelar as relações de semelhança entre as características dos participantes (e.g., uma pessoa ”Extrovertida”pode ter maior afinidade com ”Comunicativa”).
- **Matrizes**: Empregadas para representar a matriz de relações entre todos os pares de participantes, permitindo calcular dinamicamente a afinidade e desafinidade entre eles.
- **Listas e Dicionários**: Estruturas fundamentais para o armazenamento e acesso eficiente de participantes, suas propriedades (aptidões, humor, gostos, desgostos) e o registro dos eventos diários.

A lógica de programação orientada a objetos foi aplicada para encapsular as responsabilidades das entidades do sistema, como ‘Participante‘, ‘Evento‘ e ‘MatrizRelacoes‘.

2.2. Trabalhos Relacionados e Breve Revisão da Literatura

A área de simulações sociais e modelagem de comportamento humano tem sido objeto de estudo em diversas áreas, como inteligência artificial, ciência da computação e sociologia computacional. Trabalhos como os de Epstein e Axtell em ”Growing Artificial Societies”[Epstein and Axtell 1996] exploram o uso de agentes baseados em regras para simular fenômenos sociais complexos. Embora este projeto não se aprofunde em modelos complexos de inteligência artificial ou aprendizado de máquina, ele se alinha com a ideia de usar a computação para explorar dinâmicas sociais simplificadas. A aplicação de estruturas de dados para modelar relações em redes sociais também é um campo consolidado, como visto em trabalhos que utilizam grafos para análise de comunidades e influências [Newman 2006]. Nosso projeto foca em uma aplicação didática desses conceitos, especificamente no contexto de um jogo de realidade.

3. Metodologia / Desenvolvimento

3.1. Estrutura do Sistema

O sistema é modularizado em diferentes arquivos Python, cada um com responsabilidades específicas:

- ‘main_app.py‘: Controla a interface gráfica, a inicialização da simulação e a orquestração dos eventos diários.
- ‘participante.py‘: Define a classe ‘Participante‘, seus atributos (nome, características, aptidões, humor, gostos, desgostos) e métodos para geração e manipulação.
- ‘evento.py‘: Contém a classe ‘Evento‘, que gerencia os diferentes tipos de eventos (provas, interações, paredão) e a lógica de como afetam os participantes.
- ‘relacoes.py‘: Implementa a classe ‘MatrizRelacoes‘ para calcular e armazenar os níveis de relacionamento entre os participantes.
- ‘caracteristica.py‘: Define as características dos participantes e as relações de similaridade entre elas.
- ‘humor.py‘: Define constantes para os estados de humor dos participantes.

3.2. Estruturas de Dados Utilizadas

Com o objetivo de simular as características que moldam a personalidade de uma pessoa, foi construída uma lista enumerada preenchida com variáveis que representam as características. Na **Figura 1** é apresentada a lista e as constantes que representam cada característica abordada no projeto.

```
class Caracteristicas(Enum):

    INTELIGENTE = 0
    PROATIVO = 1
    AMIGAVEL = 2
    PREGUICOSO = 3
    TAGARELA = 4
    TIMIDO = 5
    COMPETITIVO = 6
    PACIENTE = 7
    PAVIOCURTO = 8
    ARROGANTE = 9
```

Figura 1. Lista enumerada com as características

Além disso, para representar a relação entre as características que possuam alguma semelhança foi implementada uma matriz de adjacência. Inicialmente, todos os valores foram zerados. Em seguida, sinais “1” foram atribuídos aos pares de índices correspondentes a nós conectados no grafo, indicando similaridade entre os traços. Na **Figura 2** é possível visualizar a construção da matriz, enquanto na **Figura 3** é apresentado o processo para alteração dos índices que representam pares conectados no grafo.

```
semelhantes = {}

for c1 in Caracteristicas:
    semelhantes[c1] = {}
    for c2 in Caracteristicas:
        semelhantes[c1][c2] = 0
```

Figura 2. Construção da matriz de adjacência

```
# Lista de pares de caracteristicas semelhantes
pares_semelhantes = [
    (Caracteristicas.INTELIGENTE, Caracteristicas.PROATIVO),
    (Caracteristicas.COMPETITIVO, Caracteristicas.PAVIOCURTO),
    (Caracteristicas.AMIGAVEL, Caracteristicas.TAGARELA),
    (Caracteristicas.PACIENTE, Caracteristicas.INTELIGENTE),
    (Caracteristicas.TIMIDO, Caracteristicas.PREGUICOSO),
    (Caracteristicas.ARROGANTE, Caracteristicas.PAVIOCURTO),
    (Caracteristicas.AMIGAVEL, Caracteristicas.PACIENTE),
    (Caracteristicas.PROATIVO, Caracteristicas.COMPETITIVO),
]

# Preenche a matriz de adjacência de forma bidirecional
for c1, c2 in pares_semelhantes:
    semelhantes[c1][c2] = 1
    semelhantes[c2][c1] = 1
```

Figura 3. Alteração dos valores em pares que possuam conexão

As relações entre os participantes foram armazenadas em uma matriz cuja implementação foi feita usando dicionario de dicionários. Em Python é possível utilizar valores não inteiros como chaves para estruturas de dados, isso é possível com a estrutura de dados Dict (dicionário). As chaves utilizadas na matriz de relações são os nomes dos participantes, isso foi feito para facilitar o acesso aos índices da matriz e a manipulação dos mesmos.

No jogo, as relações entre os participantes são definidas de acordo com a afinidade que um participante tem com as características do outro. Para definir essa relação são utilizados dois laços de repetição e uma função. O primeiro laço percorre toda a lista de participantes e o segundo também percorre essa lista, ignorando apenas o elemento atual no primeiro laço. A função é chamada em cada iteração do segundo laço e nela é definida a relação dos dois participantes. Na **Figura 4** é possível visualizar a construção da matriz de relações, enquanto na **Figura 5** é apresentada a função onde são definidas as relações entre cada participante.

```
@staticmethod
def gerar_matriz(participantes: list[Participante]) -> dict[str, dict[str, int]]:

    matriz = {}
    for p1 in participantes:
        matriz[p1.nome] = {}
        for p2 in participantes:
            matriz[p1.nome][p2.nome] = MatrizRelacoes.nivel_de_relacao(p1, p2)
    return matriz
```

Figura 4. Construção da matriz de relações entre os participantes

```python
@staticmethod
def nivel_de_relacao(p1: Participante, p2: Participante) -> int:
    relacao: int = 0
    if p1.nome != p2.nome:

        for gosto in p1.gosta:
            for char in p2.caracteristicas:
                if gosto == char:
                    relacao += 1

        for desgosto in p1.detesta:
            for char in p2.caracteristicas:
                if desgosto == char:
                    relacao -= 1
    return relacao
```

Figura 5. Função utilizada para definir as relações entre os participantes

3.3. Funcionalidades Principais

As funcionalidades principais do simulador incluem:

- Cadastro inicial de nomes de participantes.
- Geração automática de atributos para os participantes (características, aptidões, humor, gostos/desgostos).
- Progressão da simulação dia a dia com a ocorrência de eventos aleatórios ou programados.
- Realização de provas (física e raciocínio) com eliminação do pior desempenho.
- Interações sociais entre participantes, afetando o humor e as relações.
- Votação do Paredão baseada nas relações interpessoais para eliminação.
- Exibição de um log de eventos detalhado.
- Atualização em tempo real da lista de participantes e seus status.

3.4. Linguagens, Frameworks e Ferramentas

- **Linguagem de Programação**: Python 3.11.6
- **Framework GUI**: Tkinter (biblioteca padrão do Python)
- **Controle de Versão**: Git e GitHub (repositório: `https://github.com/PedroDavid2001/Trabalho_ED_PPgCC`)

3.5. Limitações e Dificuldades Enfrentadas

Durante o desenvolvimento, algumas limitações e dificuldades foram observadas:

- **Complexidade do Comportamento Humano**: A modelagem do humor e das interações sociais é uma simplificação. Comportamentos humanos reais são muito mais complexos e imprevisíveis do que o modelo atual permite.
- **Aleatoriedade e Controlabilidade**: A grande aleatoriedade na geração de características e eventos pode, por vezes, levar a simulações menos ”realistas”ou com resultados muito variados, dificultando a análise de padrões específicos.
- **Interface Gráfica (Tkinter)**: Embora funcional, o Tkinter possui limitações estéticas e de personalização comparado a frameworks GUI mais modernos, o que pode impactar a experiência visual.
- **Escalabilidade**: Para um número muito grande de participantes, a complexidade computacional do cálculo da matriz de relações poderia se tornar um gargalo, exigindo otimizações mais avançadas.

- **Refatoração e Manutenção**: A transição de funções para métodos de classe e métodos estáticos exigiu uma refatoração significativa, que, embora benéfica para a organização, representou um desafio inicial na adaptação do código existente.

4. Resultados e Discussão

4.1. Apresentação do Sistema Funcionando

O simulador apresenta uma interface gráfica simples e funcional, permitindo ao usuário iniciar a simulação após a inserção de nomes de participantes. A cada clique no botão ”Avançar Dia”, a interface atualiza a lista de participantes restantes e seu humor, aptidões e raciocínio. Um painel de log exibe textualmente os eventos ocorridos, como a geração de participantes, a realização de provas, as interações sociais (com indicação de ganho/perda de humor) e as eliminações.

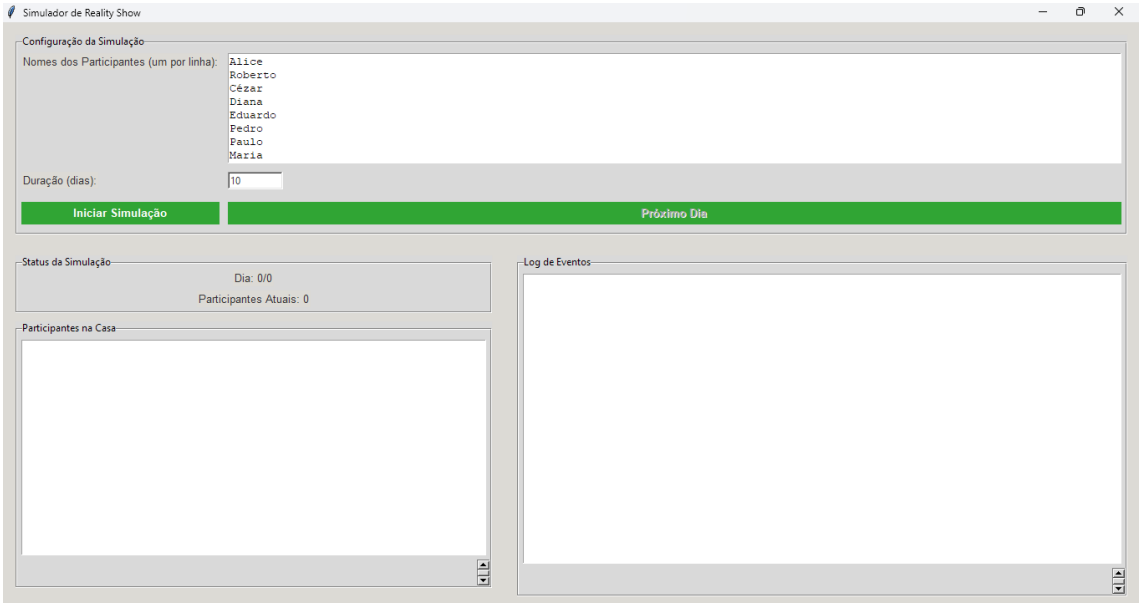

Figura 6. Tela Inicial do Simulador com Entrada de Nomes

O log de eventos detalha, por exemplo, qual prova foi realizada, quem foi eliminado, ou quais participantes interagiram e como seu humor foi afetado. Por exemplo:

```
Dia 1: Chegada dos participantes!
- Leandro
  Aptidão Física: 79,
  Raciocínio Lógico: 86,
  Humor: NEUTRO
  Características: ARROGANTE, PAVIOCURTO
  Gosta: TAGARELA, PAVIOCURTO, ARROGANTE
  Detesta: INTELIGENTE, PROATIVO, AMIGAVEL
- Lucas
  Aptidão Física: 20,
  Raciocínio Lógico: 86,
  Humor: CANSADO
```

```
  Características: PREGUICOSO
  Gosta: PREGUICOSO, TAGARELA, TIMIDO
  Detesta: INTELIGENTE, PROATIVO, AMIGAVEL, COMPETITIVO

Dia 2: Prova de Aptidão Física!
Descrição: Teste de resistência física extremo!
Participante eliminado por menor aptidão física:
- Lucas (Aptidão: 20)

Dia 3: Interações Sociais!
O clima na casa mudou após interações sociais.
```

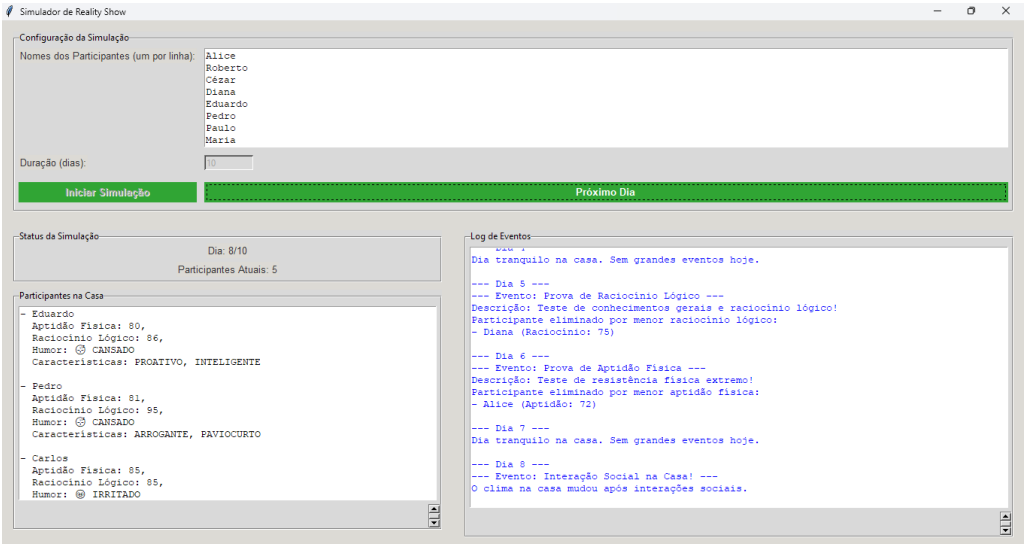

Figura 7. Simulação em Andamento com Log de Eventos e Lista de Participantes

4.2. Benefícios do Sistema

O principal benefício deste simulador reside na sua capacidade de demonstrar de forma prática e interativa a aplicação de estruturas de dados e algoritmos em um cenário complexo. Ele serve como uma ferramenta didática para visualizar como conceitos abstratos como grafos e matrizes podem modelar relações e dinâmicas sociais. Além disso, o sistema é modular e expansível, permitindo a adição de novos tipos de eventos, características ou regras com relativa facilidade.

5. Conclusão

Este projeto demonstrou com sucesso a aplicação de estruturas de dados e algoritmos no desenvolvimento de um simulador de reality show. Através da modelagem de participantes, eventos e relações, foi possível criar um ambiente interativo que ilustra de forma didática conceitos importantes da ciência da computação. As principais estruturas de dados utilizadas, como grafos e matrizes, se mostraram essenciais para a representação das complexas interações sociais. Futuros trabalhos podem explorar a adição de modelos de aprendizado de máquina para tornar o comportamento dos participantes mais adaptativo, a implementação de uma interface gráfica mais robusta, e a introdução de dinâmicas de jogo mais complexas, como formação de grupos ou tarefas em equipe.

Referências

Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press.

Newman, M. (2006). *Networks: An Introduction*. Oxford University Press.