Relatório do componente EXA-854 MI – Algoritmos
Problema Final - Jogo da Disputa

Rita Kassiane Santos dos Santos[1]

[1]Engenharia de Computação – Universidade Estadual de Feira de Santana (UEFS)
Caixa Postal 252 e 294 – 44.036-900 – Feira de Santana – BA – Brasil

ritakassiane.t.i@gmail.com

Abstract. *This report is for Problem 3 (final) of component EXA-854 - MI, Algorithms. To solve this problem, basic programming concepts (repetition loops and conditional structures) and introductory subjects to Object Oriented Programming textit (POO) in the textit Python programming language were used. The software consists of a card game, which is a dispute between two players initially receiving five cards which have the attributes textit strength, value, energy and textit Jokenpo (rock, paper, scissors). Each round a player chooses a type of contest based on attributes and wins the highest value. With this, both players discard the contest card and the losing player receives a new card. The game who runs out of cards first wins.*

Resumo. *Esse relatório é referente ao Problema 3 (final) do componente EXA-854 - MI, Algorítmos. Para a resolução deste problema, foi utilizado conceitos básicos de programação (laços de repetição e estruturas condicionais) e assuntos introdutórios à Programação Orientada a Objetos (POO) na linguagem de programação Python. O software consiste num jogo de cartas, o qual é uma disputa entre dois jogadores que recebem inicialmente cinco cartas as quais têm os atributos força, valor, energia e Jokenpo (pedra, papel, tesoura). A cada rodada um jogador escolhe um tipo de disputa com base nos atributos e vencerá o maior valor. Com isso, ambos os jogadores descartam a carta da disputa e o jogador perdedor desta, recebe uma nova carta. Ganha o jogo quem fica sem cartas primeiro.*

1. Introdução

Desde o inicio da linha do tempo da humanidade, é vis´ıvel que a capacidade humana em moldar o mundo para atender seus desejos, é reflexo de um dos legados mais fantásticos da evolução: A criatividade. No mundo contemporâneo, a combinação do caráter criativo com o caráter racional cria ferramentas poderosas que indiretamente alimentam os desejos da sociedade através do entretenimento. Hoje, com o aumento exponencial da tecnologia, é possível adaptar velhos jogos a versões mais modernas de maneira a preservar sua essência e aumentar a praticidade.

Pensando nisso, e a fim de ajudar alguns jovens com a automatização de um jogo, os tutores de MI Algorítmos propôs a turma de tal componente curricular, a criação de um jogo de cartas, o qual consiste numa disputa entre 2 jogadores: Jogador 1 e Jogador 2, os quais inicialmente recebem 5 cartas. Cada carta possui as opções: Valor, Força, Energia e Jokenpo(pedra, papel e tesoura), os quais em cada partida serão compardos, o maior

vence e para casos de Jokenpo valem as regras normais de pedra, papel e tesoura. O jogo acaba quando as cartas de um dos jogadores acabarem ou após 10 partidas.

Para o desenvolvimento do software, foi utilizado Programação Orientada a Objetos e conhecimentos de base como laços de repetição e condicionais. No script, foram criadas duas classes, a quais eram, respectivamente, Cadastro (equivalente aos dados dos usuários) e Carta (cada objeto é uma carta).

2. Metodologia

Inicialmente foi estabelecido que, para o desenvolvimento do problema, seria necessário conhecimentos baiscos de Programação Orientada a Objetos. Posteriormente a isso, o racioc´ınio seguido foi o de criar duas classes, sendo essas *Cartas*, a qual criaria um objeto Carta, com os atributos *força*, *energia e valor*. A outra classe é *Cadastro*, a qual seria encarregada de tratar cada usuário como objeto os quais teriam como atributos *nick, partidas jogadas e partidas ganhas*.

As cartas do jogo são armazenadas num arquivo txt, portanto, para manipula-las foi necessário abrir tal arquivo como *readlines*, transformar cada linha em uma lista (se cada carta estava uma linha, então, cada carta agora será uma lista), e adicionar numa lista vazia, logo, teremos uma matriz. Posteriormente, um *for* andara´ nessa matr´ız, pegando cada elemento(carta) e transformando em objeto, de maneira que o primeiro elemento de cada lista, seja o atributo**.***nome*. Finalmente, teremos cada carta como um objeto.

No entanto, havia o desafio no armazenamento dos dados de cadastro, uma vez que, no problema é solicitado que os dados de winrate (quantidade de partidas ganhas, quantidade de partidas jogadas e taxa de sucesso) sejam armazenados num arquivo binário (*o qual posteriormente fora alterado para arquivo .txt pelo tutor*). Todavia, a cada vez que um usuário ja cadastrado joga, os dados alteram, então como alterar apenas o dado deste usuário específico no arquivo txt?
A estratégia utilizada consiste em criar um dicionário geral, o qual armazenará os dados de todos os usuários jogadores, utilizando o nick como chave e uma lista como valor. Tal lista terá na primeira posição a quantidade de partidas jogadas e na segunda posição, a quantidade de partidas ganhas. Quando o usuário é cadastrado, a chave é zerada e conforme joga, esta é preenchida. No final do jogo, todos os dados que estão no arquivo **Cadastro.txt** e não estão no dicionário, serão adicionados no dicionario. Posteriormente, apaga-se todos os dados desse arquivo e adiciona-se novamente, todos os dados do dicionário nele.

O jogo é dividido em dois modos: Aleatório e Manual. A diferença entre eles é que enquanto no modo aleatório as cartas são ordenadas em ordem crescente e antes de cada disputa, sorteia-se um número no intervalo da quantidade de cartas da mão de cada jogador, o qual éequivalente a carta que sera´ jogada obrigatoriamente. Ja no modo manual, as cartas são ordenadas em ordem alfabética e o jogador escolhe a carta que quer jogar.

Para a ordenação de maneira crescente, o algorítmo implementado foi uma função a qual recebe um *x* (equivale ao atributo, por exemplo ”valor”) e uma lista de cartas como parametro. Posteriormente, dentro desta, cria-se uma lista vazia e um laço *for*.A função *min()* ira´ pegar o menor valor da lista de cartas o qual sera´ *appendado*, na lista vazia e

posteriormente, através do método *remove*, esse valor será removido da lista de cartas. O laço se repete até que o tamanho da lista de cartas seja igual a zero. Para a ordenação alfabeticamente, é relativamente mais simples, uma vez que o próprio Python reconhece a ordem. Por exemplo, é pre-estabelecido por ele que **a** < *w* ja que *a* vem antes de *w* em ordem alfabética. Logo, foi criada uma função que recebe como parâmetro uma lista de cartas. Posteriormente, dentro desta, um *for* percorre cada objeto da lista através do *atributo* ***nome*** fazendo a comparação: se o objeto.*nome* da posição posterior à do objeto analisado for ¡ o objeto analisado, eles trocam de posição. Esse processo é repetido até que a lista esteja ordenada.

Como todo jogo de carta, nesse também é necessário embaralharmos uma pilha para funcionar como *cava* e para que a destribuição seja aleatória. Para isso, foi criada uma função que recebe como parâmetro a lista de cartas e sorteia um número aleatório (através da biblioteca *random*) no intervalo de 0 ao tamanho da lista subtraido 1, (ja que conta-se iniciando de 0). Posteriormente, cria-se em uma lista do tamanho da lista de cartas porém preenchida com zeros. Depois, faz um *for* caminhar pela lista de cartas e colocar o elemento na posição em que o número foi sorteado.

Cada jogador tem uma lista de cinco cartas. Para a destribuição dessas seguindo a lógica de pegar ”cartas de uma pilha”, foi criada uma função que recebe a lista de cartas e a quantidade *x* de cartas a ser destribuidas. Dentro desta, inicialmente é criada uma lista vazia, e posteriormente, um *for* ira rodar num intervalo *x appendando* na lista vazia sempre o ultimo elemento da lista de cartas e posteriormente o excluindo da lista de cartas através do método *.pop*.

3. Resultados e Discussões

Para a utilização correta e eficaz do software, é necessário que o arquivo Cadastro.txt esteja no mesmo diretório do arquivo .py. Para que o usuário tenha uma experiência mais divertida com o jogo, é necessário que ele digite corretamente o que é pedido, pois, embora a maioria dos erros tenham sido tratados através do método *try,except* há casos em que, a implementação destes não são suficientes para que a execução normal do jogo seja prejudicada.

Foram utilizadas as bibliotecas *os* e *random*, as quais foram ferramentas para, respectivamente, ao final do jogo ou possíveis erros, o programa pausar ao invés de fechar e voltar ao início quando o usuário apertar qualquer tecla, e para o sorteio de um número aleatório. Para a optimização de processos, como na criação do algoritmo de ordenação, foram utilizadas a função *min()* e o método *.remove*

4. Conclusão

Com o problema proposto, além da valiosa experiência adquirida no trabalho em equipe, foi possível aprender, embora de maneira superficial, a manipular objetos e como criar classes, além de também, concretizar o aprendizado básico em laços de repetição e estruturas condicionais.

O software cumpre os requisitos solicitados pelo problema, no entanto, poderia ser implementado outras maneiras de armazenamento de dados dos usuário cadastrados, uma vez que, não foi encontrado uma maneira de apagar apenas uma linha de um arquivo *txt*, maneira essa que, viabilizaria a manipulação e armazenamento dos dados dos

usuários num arquivo desta extensão simplificando a quantidade de passos e sem exigir tanto processamento. Outra problemática que poderia ter sido resolvida caso houvesse conhecimento suficiente, é acerca da interface gráfica e da exibição das cartas dos jogadores. No jogo, ambos os jogadores podem ver as cartas entre si. No entanto, poderia ter sido implementado uma biblioteca a qual, a apagaria as cartas do jogador do prompt a cada jogada. Essa ultima idéia não fora cumprida por questão de tempo.

5. Bibliografia

Programação Básica - Introdução em Python. Disponível em: https://www.docdroid.net/SI5rExq/programacao-basica-introducao-em-python.pdf <Acesso em 10 de setembro de 2019.>

Arquivos - Python - Orientação Objeto Disponível em: https://www.caelum.com.br/apostila-python-orientacao-objetos/arquivos-e-modulos/exerccios—leitura-de-arquivos <Acesso em: 6 de setembro de 2019>

Como computadores ordenam dados? Dispon´ıvel em: <https://www.youtube.com/watch?v=DT7TVYHhepY> <Acesso em 3 de setembro de 2019>.