Jogo das Somas 2.0

Luis Felipe Pereira de Carvalho

Departamento de Ciências Exatas - Engenharia de Computação

Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana – BA – Brazil

luisoftbr@outlook.com.br

Abstract. *seeking educational alternatives for their amusement during the harsh winters, the brothers Lara and Liam proposed to the Students of the Computer Engineering Course of the State University of Feira de Santana the creation of an educational game called Soma-sudoku or Jogo das Somas 2.0. The proposed game provokes the creation of a source code for the development of its interface and levels. For the development, the python language was used. In conclusion, the system does not meet all the requirements that were requested.*

Resumo. *Buscando alternativas educativas para sua diversão durante os invernos rigorosos, os irmãos Lara e Liam propuseram aos Alunos do Curso de Engenharia de Computação da Universidade Estadual de Feira de Santana a criação de um jogo educativo chamado de Soma-Sudoku ou Jogo das Somas 2.0. O jogo proposto provoca a criação de um código fonte para o desenvolvimento de sua interface e níveis. Para o desenvolvimento, foi utilizada a linguagem python. Em conclusão, o sistema não cumpre com todos os requisitos que foram solicitados.*

1. Introdução

Segundo Silva (2019), atualmente as crianças têm sofrido muita influência das mídias. Definitivamente é praticamente impossível imaginar uma infância livre da influência dos equipamentos eletrônicos, e não podemos encarar a tecnologia como vilã na vida das crianças. Logo, nesse meio tecnológico também estão incluídos os jogos digitais que trazem a todos a possibilidade de se entreter, como os jogos brasileiros, Dandara, A lenda do Herói, ou Nyanroo: The SuperCat e de aprender e desenvolver habilidades como os jogos da Escola Games.

Analisando o papel que jogos digitais podem assumir e, inserido no contexto da necessidade de entretenimento e de aprendizado para crianças que vivem em locais com invernos rigorosos, como o caso de Ottawa no Canadá. Os irmãos Lara e Liam, que enfrentam esses invernos e que já jogaram outro jogo de somas, fizeram uma proposta aos alunos do curso de Engenharia de Computação da UEFS para a criação de um jogo educativo chamado de Soma-Sudoku ou Jogo das Somas 2.0, que funciona semelhante ao sudoku e possui pontuação atrelada ao preenchimento das linhas, colunas e diagonal principal do tabuleiro.

Para atender aos requisitos do jogo, o código foi elaborado principalmente com modularização e listas. A construção do jogo é apresentada na seguinte seção que trata da metodologia utilizada.

2. Metodologia

De maneira a atender e discutir o desenvolvimento do software, foram realizadas sessões Problem-Based-Learning (PBL) para a discussão do jogo e de seus requisitos que foram solicitados e apresentados pelo tutor. Foram realizadas sessões para discutir fatos, ideias, questões e metas para a resolução do jogo.

Para informações técnicas, é necessário informar que o sistema foi desenvolvido integralmente em linguagem Python na versão 3.10. A IDE utilizada foi a Visual Studio Code na versão 1.71.0 no sistema operacional Linux Mint 20.3 Una.

A seguir são apresentados e detalhados os requisitos solicitados para a conclusão do jogo.

2.1 Definição de requisitos

Como citado anteriormente, para a construção do jogo se fez necessário atender alguns requisitos, os quais estão exibidos na tabela 1:

Tabela 1. Requisitos do processamento do jogo

Requisito	Detalhamento
Tipo do jogo	O jogo é de tabuleiro e deve ser disputado entre dois jogadores: Jogador 1 e Jogador 2, utilizando um único tabuleiro, com ambos jogando no mesmo.
Níveis do jogo	O jogo é processado em dois níveis, um fácil e um difícil. No nível fácil o tabuleiro tem as dimensões 4x4, dividido em 4 seções, e no nível difícil o tabuleiro apresenta as dimensões 9x9, dividido em 9 seções.
Preenchimento das seções	Cada uma das seções do tabuleiro é preenchida aleatoriamente com números de 1 a 4 para o nível fácil, e números de 1 a 9 para o nível difícil, de forma que os números não se repitam em uma mesma seção.
Exibição do tabuleiro	O tabuleiro com os números randomizados deve permanecer oculto e exibir apenas as somas correspondentes das linhas ao seu lado e das colunas, abaixo delas, a soma da diagonal principal também deve permanecer oculta.
Número revelado	• Se o número revelado completa uma linha ou coluna do tabuleiro, o jogador acumula a soma daquela linha ou coluna à sua pontuação; • Se o número revelado completa uma linha e uma coluna ao mesmo tempo, ambas as somas são acrescidas à pontuação do jogador; • Se o número revelado completa a diagonal principal do tabuleiro, o jogador acumula o dobro do valor da soma da diagonal.

A tabela 2 corresponde aos requisitos de saídas que são exibidas no jogo, durante e ao final das jogadas:

Tabela 2. Requisitos do jogo relacionados a saída dos dados

Requisito	Detalhamento
Tabuleiro	Corresponde ao tabuleiro que é exibido aos jogadores, suas posições são ocultas com o zeros que são substituídos pelos números revelados. O tabuleiro apresenta, ainda, as somas das linhas e das colunas.
Posição revelada	Corresponde a exibição do número escolhido pelo jogador no tabuleiro de saída.
Pontuação momentânea	Exibe a pontuação dos jogadores a cada jogada realizada.
Pontuação final	Exibe qual jogador foi vitorioso e sua pontuação.

Nas proximas subseções, é apresentado como se decorreu o desenvolvimento do código fonte para a atender a resolução do problema proposto.

2.2 Desenvolvimento

Para a manipulação dos dados do jogo foram utilizadas variáveis, bibliotecas, condicionais do tipo *if* (se) e laços de repetição do tipo *while*(enquanto) e *for* (para) e, principalmente, funções e listas.

```
import random
import time
secao1 = []; secao2 = []; secao3 = []; secao4 = []
secao5 = []; secao6 = []; secao7 = []; secao8 = []; secao9 = []
```

Figura 1.Importação das bibliotecas de randomização e tempo e inicialização das principais seções a serem manipuladas durante a execução do código.

Com a finalidade de melhor compreensão do código, é importante dividi-lo em subseções para melhor elucidação do que foi realizado em cada parte. Na seguintes subseções são apresentados e explicitados: as funções, menu principal, os níveis do jogo e a saída do jogo.

2.2.1 Funções

De modo a fazer a modularização do código, foram utilizadas funções que, segundo a documentação do Python, são subprogramas responsáveis por realizar processamentos menores no código e evitar repetições desnecessárias, desse modo seus resultados são recombinados em um programa principal, que no caso desse jogo, são o menu principal e os níveis.

No código do jogo ao total foram utilizadas 12 funções, onde são processados uma ou mais dados.

```
def gerar_secao(base): ···

def verificar_menu(): ···

def num_e_secoes(): ···

def somas_col_facil(): ···

def somas_col_dif(): ···

def somas_lin_facil(): ···

def somas_lin_dif(): ···

def somas_diag_facil(): ···

def somas_diag_dif(): ···

def pontuacao_parcial(pont_j1 , jogador1, pont_j2, jogador2): ···

def exibir_matriz(base): ···

def pontuacao_final(pont_j1 , jogador1, pont_j2, jogador2): ···
```

Figura 2. As funções que utilizadas durante a execução do código

A função *gerar_secao* é responsável por randomizar os números no intervalo de (1-4) no nível fácil ou (1-9) no nível difícil. Através da biblioteca *random* com o uso da extensão *sample* os números são inseridos nos índices das seções enquanto elas são geradas. as seções de ambos os níveis são geradas, conforme a alteração de uma variável chamada *base* que recebe ou o valor de 2 referente ao tamanho do lado de um seção do nível fácil (4x4) ou o valor de 3 que se refere ao tamanho do lado de uma seção do nível difícil (9x9). A função pode ser observada na figura 3.

```
def gerar_secao(base):
    nivel = base**2
    matriz_interna= []
    num_random = random.sample(range(1, nivel + 1), nivel)
    if base == 2:
        l1= [(num_random[0]),(num_random[1])]
        l2= [(num_random[2]),(num_random[3])]
        matriz_interna.append(l1)
        matriz_interna.append(l2)
        return matriz_interna
    elif base == 3:
        l1= [(num_random[0]),(num_random[1]),(num_random[2])]
        l2= [(num_random[3]),(num_random[4]),(num_random[5])]
        l3= [(num_random[6]),(num_random[7]),(num_random[8])]
        matriz_interna.append(l1)
        matriz_interna.append(l2)
        matriz_interna.append(l3)
        return matriz_interna
```

Figura 3. A função *gerar_secao* com o uso da variável base para gerar seções do nivel facil (base = 2) ou difícil (base = 3)

A função *verificar_menu* é responsável por validar as escolhas para o menu através de um laço de repetição *while* com o condicional *try/except*, que verifica se a entrada do usuário é um número inteiro e se está no intervalo das escolhas possíveis (0-2), isto se repete até que seja inserida uma entrada válida. ela retorna o valor

escolhido pelo usuário para que prossiga para a execução ou do nível fácil ou do nível difícil ou para encerrar o jogo, A função pode ser observada na figura 4.

```
def verificar_menu():
    print('''\nO que desejam fazer?
[1] - Selecionar nível fácil \n[2] - Selecionar nível difícil \n[0] - Sair do jogo
\n================================================\n''')
    while True:
        n = input('Por favor, escolha uma das opções válidas: ')
        try:
            n = int(n)
            if 0 <= n <= 2:
                break
            else:
                print('\nOpção inválida, tente novamente!')
        except ValueError:
            print("\nSomente números são permitidos.")
    return n
```

Figura 4. A função *verificar_menu* mostra o uso do try/except para validar a entrada

A função *num_e_secoes* é dividida em duas partes e também recebe o valor de base, para executar as instruções conforme o nível escolhido.

A primeira parte é responsável por receber a entrada dos valores da seção e do número escolhido pelo jogador. Esses valores são manipulados dentro da função que remove o número escolhido na seção escolhida de modo a não ser mais disponível a sua escolha, semelhante, a seção é removida caso ela seja totalmente preenchida. Esta parte é disposta na figura 5a.

Na segunda parte, a função procura, em índice por índice, a posição do número escolhido pelo usuário na seção escolhida que foi randomiza na função *gerar_secao* e troca, em igual posição[i][j], o valor de '0' que é exibido na seção resposta, pelo número escolhido. Essa parte é disposta na figura 5b.

```
def num_e_secoes():
    if base == 2:
        num_secao = int(input('Escolha a seção de 1 a 4: '))
        while num_secao not in secoes:
            print('\nSeção inválida ou já preenchida!')
            num_secao = int(input('Escolha outra seção: '))
        if num_secao == 1:
            num = int(input('\nEscolha um número entre 1 e 4: '))
            while num not in sc1:
                num = int(input('Escolha outro número:'))
            sc1.remove(num)
            if sc1 == []:
                secoes.remove(1)
```

Figura 5a. A função *num_e_secoes* recebe os valores da seção e do número escolhido pelo usuário

```
    if num_secao == 1:
        for i in range(2):
            for j in range(2):
                if secao1[i][j] == num: sc1_resp[i][j] = num
```

Figura 5b. A função *num_e_secoes* procura o número escolhido na seção escolhida e exibe o número na mesma posição na seção correspondente que é exibida aos jogadores.

As funções *somas_col* (fácil e difícil) são responsáveis por comparar as somas da matriz interna e dos itens da matriz externa de modo a anexar o valor da soma dos números de uma coluna ao jogador que completá-la. A função sempre retorna o valor da variável *pontos* que recebe o valor dessas somas. A função pode ser observada na figura 6.

```
def somas_col_facil():
    pontos = 0
    if soma_c1 == m_resp[0][0]+ m_resp[1][0]+m_resp[2][0]+m_resp[3][0]:
        pontos += soma_c1
```

Figura 6. As funções *somas_col* retornam pontos para um jogador caso ele complete uma coluna.

As funções *somas_lin* (fácil e difícil) são responsáveis por comparar as somas da matriz interna e dos itens da matriz externa, de modo a anexar o valor da soma dos números de um linha ao jogador que completá-la. A função pode ser observada na figura 7.

```
def somas_lin_facil():
    pontos = 0
    if soma_l1 == m_resp[0][0]+m_resp[0][1]+m_resp[0][2]+m_resp[0][3]:
        pontos += soma_l1
```

Figura 7. As funções *somas_lin* retornam pontos para um jogador caso ele complete uma linha.

A função *somas_diag* (fácil e difícil) é responsável por comparar as somas da matriz interna e dos itens da matriz externa, de modo a anexar o valor da soma da diagonal principal ao usuário que completá-la, o jogador recebe a pontuação de diagonal de dobrada. A função pode ser observada na figura 8.

```
def somas_diag_facil():
    pontos = 0
    if soma_diag == m_resp[0][0]+m_resp[1][1]+m_resp[2][2]+m_resp[3][3]:
        pontos += (soma_diag*2)
    return pontos
```

Figura 8. As funções *somas_diag* retornam pontos dobrados para o jogador que completar a diagonal principal.

A função *exibir_matriz* é responsável por exibir aos jogadores as seções respostas e as somas de cada linha e coluna das seções correspondentes que foram randomizadas internamente. A função recebe o valor da variável *base* para exibir a matriz resposta conforme o nível escolhido. A função pode ser observada na figura 9.

```
def exibir_matriz(base):
    if base == 2:
        print('\n','==='*8,'\n')
        for i in range(2): print('',sc1_resp[i],'| |',sc2_resp[i],'|', sc_soma_pt1[i])
        print('-'*20)
        for i in range(2): print('',sc3_resp[i],'| |',sc4_resp[i],'|', sc_soma_pt2[i])
        print('-'*20)
        print(' ',soma_c1, soma_c2,' | | ', soma_c3, soma_c4,' |')
        print('\n','==='*8,'\n')
```

Figura 9. A função *exibir_matriz* que exibe a matriz para os jogadores após cada jogada.

A função *pontuacao_parcial* é responsável por receber e exibir para os jogadores as pontuações parciais após cada jogada. Ela pode ser observada na figura 10.

```
def pontuacao_parcial(pont_j1 , jogador1, pont_j2, jogador2):
    print('\nPONTOS PARCIAIS:\n')
    print('*'*40)
    print('O JOGADOR %s, ESTÁ COM %d PONTOS.'% (jogador1.upper(), pont_j1))
    print('-'*40)
    print('O JOGADOR %s, ESTÁ COM %d PONTOS.'% (jogador2.upper(), pont_j2))
    print('*'*40)
```

Figura 10. A função *pontuacao_parcial* que exibe a pontuação para os jogadores após cada jogada.

A função *pontuacao_final* é responsável por mostrar o final do jogo e o resultado junto com a pontuação que cada um dos jogadores adquiriu. Ela pode ser observada na figura 11.

```
def pontuacao_final(pont_j1 , jogador1, pont_j2, jogador2):
    time.sleep(1)
    print('\n','==='*26)
    print('        FIM DE JOGO')
    print('\n','==='*26)
    if pont_j1 > pont_j2:
        print('\n            PARABÉNS %s, VOCÊ VENCEU O JOGO COM %d PONTOS!'% (jogador1.upper(), pont_j1))
    elif pont_j2 > pont_j1:
        print('\n            PARABÉNS %s, VOCÊ VENCEU O JOGO COM %d PONTOS!'% (jogador2.upper(), pont_j2))
    elif pont_j2 == pont_j1:
        print('\n            %s E %s VOCÊS EMPATARAM O JOGO!'% (jogador1.upper() ,jogador2.upper()))
    print('\n','==='*26)
```

Figura 11. A função *pontuacao_final* exibe o fim do jogo e o resultado.

2.2.2 Menu inicial

No menu inicial é solicitada a primeira entrada ao usuário através da função *verificar_menu*(figura 4). Essa entrada permite ao jogador escolher o que ele fará em seguida. O menu principal é mostrado na figura 12.

```
print('==='*12, '\n     BEM-VINDOS AO JOGO DAS SOMAS 2.0\n', '==='*12, '\nESCOLHAM SEUS NOMES!')
jogador1 = str(input('\nJOGADOR 1: '))
jogador2 = str(input('\nJOGADOR 2: '))
num_menu = verificar_menu()
print('\nCarregando jogo...')
time.sleep(1)
```

Figura 12. Menu Inicial.

A variável *num_menu* recebe o resultado da função *verificar_menu*(figura 4) e é utilizada na verificação de um laço de repetição *while (num_menu != 0)* onde o valor digitado é processado por condicionais *if*(se) onde no primeiro com (*num_menu == 1*)

são dispostas as instruções para a execução do nível fácil do jogo, no segundo com (*num_menu* == *2*) são dispostas as instruções para a execução do nível difícil do jogo.

2.2.3 Níveis

Nos niveis são processadas as instruções para o início, execução e término de uma partida ou no nível fácil ou no difícil. Ambos os níveis processam e recebem valores de todas as funções explicitadas anteriormente, exceto aquelas que são divididos para fins de melhor manipulação, como é o caso das funções de somas. É importante ressaltar que todo o processo que é descrito nesta subseção acontece de maneira semelhante em ambos os níveis, desde a inicialização de variáveis, listas, até os processamentos das jogadas e encerramento do jogo e, de modo a facilitar o entendimento, são explicados e apresentados os processos do nível fácil do jogo.

Inicialmente os niveis inicializa as variáveis, *base*, *pont_j1* e *pont_j2* que são as pontuações dos jogadores, a variável *cont* que conta o número de jogadas, as variáveis *pont_ant* e *pont_dps*, que servem para verificar se o jogador completou uma linha, uma coluna ou a diagonal e assim, permite que o valor das somas seja indexado ao valor da pontuação do jogador (*pont_j1* ou *pont_j2*) e a variável *fim* que recebe o valor de *True* pois é utilizada como parâmetro no laço de repetição *while* (enquanto) das jogadas.

Seguidamente, são iniciadas as seções *scx* que são utilizadas na função num_e_secoes que remove os números para que não sejam mais escolhidos e as seções preenchidas para que não sejam mais escolhidas. A figura 13 apresenta as variáveis e as seções *scx* sendo iniciadas no nível fácil, onde a *base* recebe o valor de 2. A figura 13 apresenta a inicialização dessas variáveis e listas.

```
base = 2; pont_j1 = 0; pont_j2 = 0; fim = True; cont = 0; pont_ant = 0; pont_dps = 0
sc1 = [1,2,3,4]; sc2 = [1,2,3,4]; sc3 = [1,2,3,4]; sc4 = [1,2,3,4]; secoes = [1,2,3,4]
```

Figura 13. inicialização das variáveis e das seções auxiliares

Seguidamente, as seções iniciais do tabuleiro são inicializadas com cada uma recebendo a seção que é gerada na função *gerar_secao* , também são criadas as seções *sc_resp* preenchidas com zeros que são exibidas aos jogadores e, conforme os jogadores vão revelando os números, os zeros são substituídos pelos números encontrados pelos jogadores. A figura 14 apresenta a inicialização dessas seções.

```
secao1 = gerar_secao(base); secao2 = gerar_secao(base)
secao3 = gerar_secao(base); secao4 = gerar_secao(base)
sc1_resp = [[0,0],[0,0]]; sc2_resp = [[0,0],[0,0]]
sc3_resp = [[0,0],[0,0]]; sc4_resp = [[0,0],[0,0]]
```

Figura 14. inicialização das seções

De forma subsequente, as seções *secaox* geradas (figura 14), são alocadas em listas de linhas e colunas e da coluna que servem para facilitar a soma dos números e são somadas através da função embutida *sum.* por fim, são criadas listas simples que recebem as somas das linhas para a exibição das mesmas no laço *for* de exibição (figura 9). A figura 15 apresenta essas listas e suas somas.

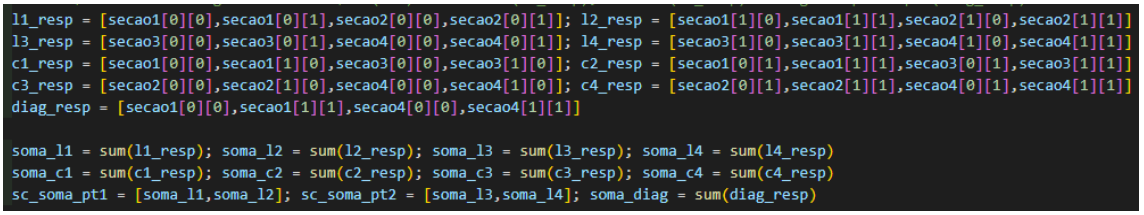

Figura 15. listas construídas de modo a facilitar a soma dos elementos e suas somas com a função *sum*

Por fim, antes da parte principal do nível, as seções do tabuleiro que é exibido ao usuário são indexadas em listas linhas e colunas para que posteriormente suas somas sejam anexadas a variável *pont_dps.* As lista*s lx_resp* são anexadas em uma segunda lista *m_resp* que tem a soma de seus elementos comparados com a somas das linhas, colunas e diagonal principal do tabuleiro intermediário (figuras 6,7 e 8). Essas listas e a matriz *m_resp* podem ser observadas na figura 16.

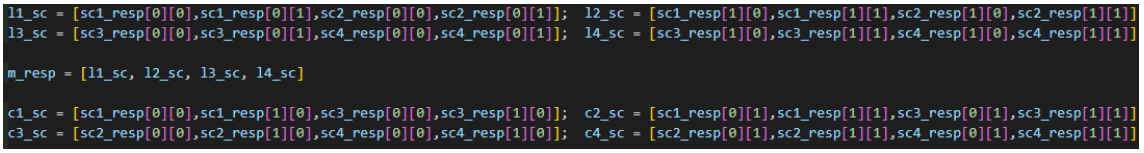

Figura 16. listas construídas de modo a facilitar a soma dos elementos do tabuleiro que é exibido, elas são anexadas na matriz resposta (*m_resp*)

Por fim, as jogadas são realizadas dentro de um laço *while*(enquanto) que recebe como parâmetro, o valor de fim (figura 13) para que as instruções internas sejam continuamente processadas.

Para as jogadas de cada um dos jogadores são processadas as seguintes instruções:

- A frase é imprimida informando ao jogador (jogador 1 ou 2) a sua vez;
- A chamada da função *num_e_secoes* (figuras 5a e 5b);
- A variável *pont_dps* recebe o valor das somas das linhas e colunas (figura 16);
- Em seguida é feita uma verificação com o condicional if(se) pois caso o valor da variável *pont_dps* seja maior que o da variável *pont_ant*, ele permite que a variável *pont_jx* recebe a pontuação para o jogador e, fora do condicional, *pont_ant* recebe o valor de *pont_dps;*
- A variável *cont* recebe o valor de +1 para poder encerrar o jogo quando atingir o limite;
- A chamada da função *pontuacao_parcial* exibe a pontuação parcial dos jogadores;
- A matriz resposta(*m_resp*) e as linhas e colunas são atualizadas depois de cada jogada de modo a permitir as somas dos pontos;

  O processamento da jogada pode ser observado na figura 17. Nela é apresentado o do primeiro jogador pertencente ao nível fácil.

```
print("\nJOGADOR %s, SUA VEZ!\n" % jogador1.upper())
num_e_secoes()
exibir_matriz(base)
pont_dps = sum(l1_sc) + sum(l2_sc) + sum(l3_sc) + sum(l4_sc) + sum(c1_sc) + sum(c2_sc) + sum(c3_sc) + sum(c4_sc)
if pont_dps > pont_ant: pont_j1 += somas_col_facil() + somas_lin_facil() + somas_diag_facil()
pont_ant = pont_dps
cont += 1
pontuacao_parcial(pont_j1 , jogador1, pont_j2, jogador2)
m_resp = [l1_sc, l2_sc, l3_sc, l4_sc]
l1_sc = [sc1_resp[0][0],sc1_resp[0][1],sc2_resp[0][0],sc2_resp[0][1]]
l2_sc = [sc1_resp[1][0],sc1_resp[1][1],sc2_resp[1][0],sc2_resp[1][1]]
l3_sc = [sc3_resp[0][0],sc3_resp[0][1],sc4_resp[0][0],sc4_resp[0][1]]
l4_sc = [sc3_resp[1][0],sc3_resp[1][1],sc4_resp[1][0],sc4_resp[1][1]]
c1_sc = [sc1_resp[0][0],sc1_resp[1][0],sc3_resp[0][0],sc3_resp[1][0]]
c2_sc = [sc1_resp[0][1],sc1_resp[1][1],sc3_resp[0][1],sc3_resp[1][1]]
c3_sc = [sc2_resp[0][0],sc2_resp[1][0],sc4_resp[0][0],sc4_resp[1][0]]
c4_sc = [sc2_resp[0][1],sc2_resp[1][1],sc4_resp[0][1],sc4_resp[1][1]]
```

Figura 17. Processamento que acontece para a jogada de um jogador ser realizada

Por fim o encerramento e apresentação do resultado do jogo é feito com condicional *if*(se) que recebe como parâmetro o valor da variável *cont* e caso *cont* seja igual a 16 no nível fácil ou igual a 81 no nível difícil, ela muda o valor da variável booleana *fim* (figura 13) para *False* e chama a função *pontuacao_final e* após o encerramento do laço *while,* a variável *num_menu* recebe novamente a função *verificar_menu* (figura 4) permitindo uma nova partida. A figura 18 apresenta o encerramento do nível do jogo.

```
    if cont == 16:
        fim = False
        pontuacao_final(pont_j1 , jogador1, pont_j2, jogador2)
#Ao final do jogo, pergunta se os jogadores querem iniciar um novo nivel.
time.sleep(1)
num_menu = verificar_menu()
```

Figura 18. Encerramento do nível do jogo

2.2.4 Saída do jogo

Quando o usuário escolhe a opção zero no menu inicial, as demais partes do código não são mais processadas e é exibida a mensagem disposta abaixo antes da execução do sistema ser interrompida.

```
# Mensagem de encerramento do jogo
time.sleep(1)
print('==='*26)
print('\n  OBRIGADO POR JOGAREM O JOGO DAS SOMAS 2.0, ESPERO QUE TENHAM SE DIVERTIDO!\n')
print('==='*26)
```

Figura 19. Mensagem de encerramento do jogo

Na seguinte seção são apresentados os resultados obtidos com a conclusão do problema, um manual de uso para o usuário e os erros que acontecem durante a execução.

3. Resultados e discussões

O código tem como objetivo principal ser um jogo de somas que funciona de maneira semelhante ao sudoku mas com o diferencial de que linhas e colunas de seções diferentes entre si podem ter números iguais. Ele concede ao jogador, caso ele complete, o resultado da soma dos números da linha, coluna ou da diagonal principal que foi completada, ao final, vence o jogador que obtiver maior pontuação. O sistema foi construído de maneira que sua utilização fosse a mais simples possível, facilitando a experiência do usuário.

A seguir são apresentados os passo-a-passos que o usuário deve seguir para iniciar e jogar o jogo das somas.

3.1 Manual de Uso

Ao executar o jogo, o usuário encontra o menu inicial conforme foi detalhado na figura 12. Na figura 20, ele é mostrado sob a visão do usuário.

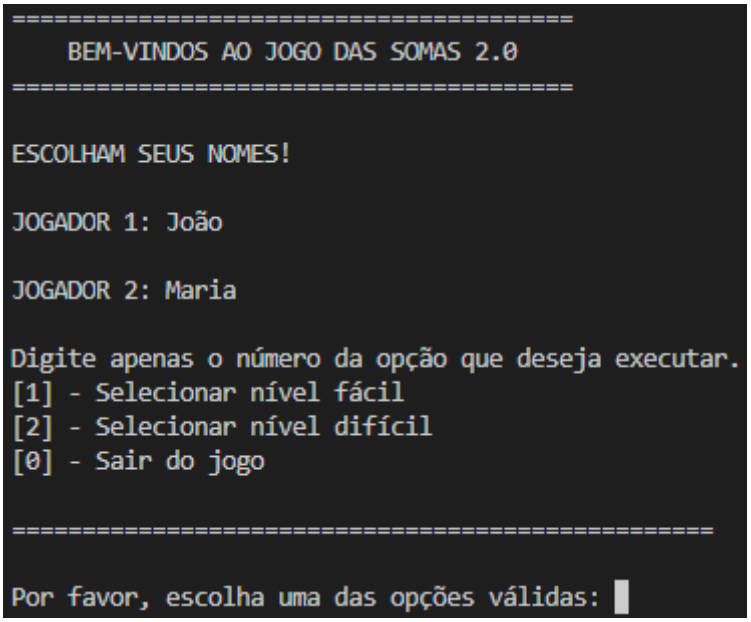

Figura 20. Menu inicial do jogo

De forma a realizar uma execução do jogo são utilizados os jogadores 'João' e 'Maria', e o nível selecionado, o fácil.

3.1.1 iniciando uma nova partida

Conforme explicitado na sub subseção 2.1.2, ao escolher a opção [1] é iniciado o nível fácil do jogo.

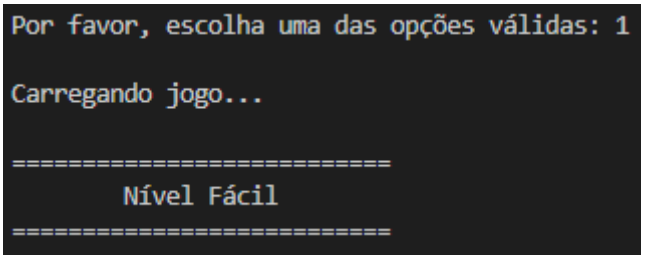

Figura 21. Início do nível fácil do jogo

Subsequentemente, é solicitada a entrada da seção e do número escolhidos pelo jogador 1, que nesse caso é João. Logo após a escolha, é apresentado o tabuleiro com o número escolhido revelado na seção escolhida e também a pontuação parcial de João e Maria. Isto pode ser observado na figura 22.

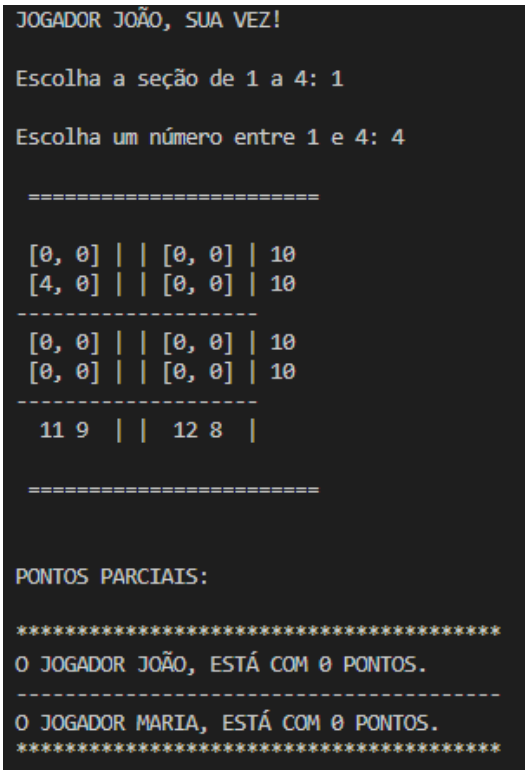

Figura 22. jogada realizada

Ao passo que as jogadas vão sendo realizadas, as pontuações parciais vão sendo atualizadas e recebendo os valores das somas das linhas, colunas e da diagonal principal que o jogador completar.

Ao final, quando todo o tabuleiro é revelado, o jogo é encerrado, apresenta o ganhador da partida e permite o início de uma nova partida. A figura 23 apresenta essa etapa.

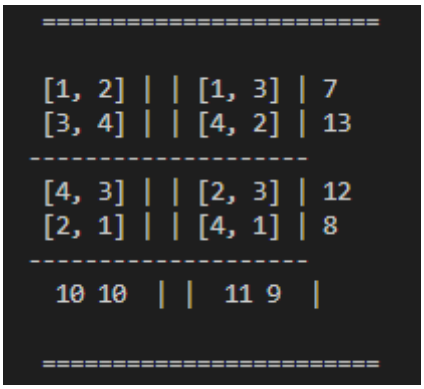

Figura 23a. tabuleiro preenchido

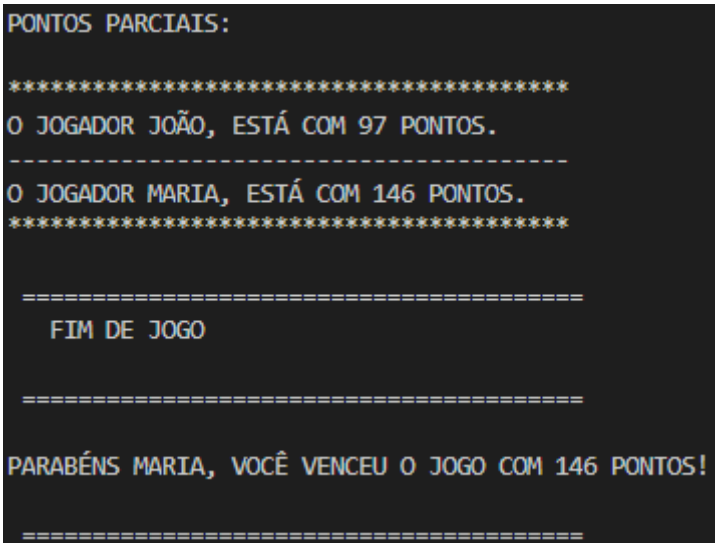

Figura 23b. Pontuação final e mensagem de ganhador do jogo

Ao final da partida, o usuário pode escolher novamente qual nível deseja jogar ou encerrar o jogo, conforme a figura 20.

Na seguinte subseção, são apresentados os problemas encontrados durante a execução do sistema.

3.2 Bugs

Após o desenvolvimento de todo o código fonte foram constatadas algumas falhas. A primeira é: ao escolher a seção e o número durante as jogadas, se a entrada de dados for de maneira vazia ou não utilizar apenas números inteiros nas entradas das opções, o sistema apresenta falha e encerra sua execução; O segundo é que as pontuações apresentam resultados incorretos, de maneira que a somas não são adequadamente indexadas durante as jogadas, fazendo com que apresentem erros no resultado total dos pontos. Demais erros não foram encontrados após a conclusão e em testes do código fonte.

Na seguinte seção é realizada a conclusão do relatório e as considerações finais.

4. Conclusão

Com a conclusão do sistema pode-se colocar em prática o uso de algumas das estruturas básicas do python sendo elas as funções, listas, estruturas condicionais e laços de repetição. Os requisitos de entrada solicitados foram atendidos de modo foi possível a execução do jogo e seus níveis.

Contudo, para os requisitos de saída, quase todos foram atendidos, exceto a indexação correta das pontuações para os jogadores, que apresenta valores diferentes aos referentes às somas que são feitas durante as jogadas.

Além disso, o projeto abre melhorias para o sistema como a possibilidade de criação de novos níveis e novas somas para maior pontuação.

Referências

PYTHON SOFTWARE FOUNDATION. Entrada e saída. 2022. Disponível em: https://docs.python.org/pt-br/3/tutorial/inputoutput.html. Acesso em: 13 set. 2022

PYTHON SOFTWARE FOUNDATION. Funções. 2022. Disponível em: https://docs.python.org/pt-br/3.10/reference/compound_stmts.html. Acesso em: 02 nov. 2022

PYTHON SOFTWARE FOUNDATION. Estrutura de Dados. 2022. Disponível em: https://docs.python.org/pt-br/3/tutorial/datastructures.html. Acesso em: 02 nov. 2022

PYTHON SOFTWARE FOUNDATION. Funções Embutidas. 2022. Disponível em: https://docs.python.org/pt-br/3/library/functions.html. Acesso em: 02 nov. 2022

SILVA, Soraia Aparecida de Oliveira. A importância dos jogos educativos na educação infantil. Disponível em: http://dspace.nead.ufsj.edu.br/trabalhospublicos/handle/123456789/346 Acesso em: 05 nov. 2022

MELO, Diego. 10 jogos brasileiros para baixar no PC, PS4, Xbox e Switch. 2020 Disponível em: https://tecnoblog.net/responde/10-jogos-brasileiros-para-baixar-no-pc-ps4-xbox-e-switch/ Acesso em: 05 nov. 2022

ESCOLA GAMES. Todos os Jogos. 2022. Disponível em: https://www.escolagames.com.br/jogos/ Acesso em: 05 nov. 2022

FELIPE, Murilo Mendes. As 10 cidades mais frias do mundo. 2020. Disponível em: https://diariodoestadogo.com.br/as-10-cidades-mais-frias-do-mundo-76613/ Acesso em: 07 nov. 2022