Jogo das somas 2.0

Luis Felipe Cunha

Departamento de ciências exatas - Universidade Estadual de Feira de Santana (UEFS)

CEP 44036-900 – Feira de Santana – BA – Brasil

luis.lipecunha@gmail.com

Abstract. *The sum game 2.0 is an adaptation between sudoku and the sum game. It supports up to two players at the same time and has two difficulty levels, an easy 4x4 board and a hard 9x9. Players will complete rows and columns of the boards with numbers to be revealed by them, structurally organized into sections represented by matrices. When completing a row, column or the main diagonal, the player takes the sum of that item in points, whoever gets the most points wins.*

Resumo. *O jogo das somas 2.0 é uma adaptação entre o sudoku e o jogo das somas. Suporta até dois jogadores ao mesmo tempo e possui dois níveis de dificuldade, um fácil com tabuleiro 4x4 e um difícil 9x9. Os jogadores completarão linhas e colunas dos tabuleiros com números a serem revelados por eles, estruturalmente organizados em seções representadas por matrizes. Ao completar uma linha, coluna ou a diagonal principal o jogador leva a soma daquele item em pontos, ganha aquele que obter mais pontos.*

1. Introdução

O sudoku foi criado pelo matemático suíço Leonard Euler no século 18. O jogo tem como objetivo treinar o raciocínio do jogador que deve completar um tabuleiro com números, sem repeti-los dentro das linhas, colunas e blocos. [FOLHA UOL,2021].

Uma adaptação entre o sudoku e o jogo das somas foi solicitada aos alunos do curso de Engenharia de Computação da Universidade Estadual de Feira de Santana (UEFS). O novo jogo, nomeado como jogo das somas 2.0 é desenvolvido com a linguagem de programação Python. Elementos dos dois jogos estarão presentes nesta aplicação digital, além disso, o jogo deve suportar dois jogadores de forma simultânea.

O jogo das somas 2.0 adapta o tabuleiro do sudoku, sendo utilizado um modelo 4x4 no nível fácil e um 9x9 no difícil. O jogo estará dividido em seções com números que não se repetirão. As linhas e colunas do tabuleiro serão somadas e como incremento extra, também haverá a soma da diagonal principal do tabuleiro, uma adaptação do jogo das somas.

Os jogadores realizarão suas jogadas de forma intercalada. Ao completar uma linha ou coluna o jogador ganha a soma destes elementos como pontos. No momento em que a diagonal se completa a sua soma é multiplicada por dois, garantindo um bônus para o usuário. Ganha o jogador que acumular mais pontos durante a partida.

2. Metodologia

Para um melhor desenvolvimento do programa sessões tutoriais do componente MI algoritmos foram realizadas para compartilhamento de ideias e fatos ao longo do

desenvolvimento da aplicação. A utilização de funções para criação do tabuleiro com os números e tabuleiro resposta foi compartilhada ao grupo. Funções que realizam tarefas especificas no Python que seriam úteis a aplicação também foram expostas, como o sum para somatório de listas e o .sample para que os números não se repetissem na seção.

Os jogadores devem escolher apenas números validos ao longo das rodadas, ou seja, seções que ainda estão disponíveis ou existam, além de números que ainda não foram escolhidos para exibição no tabuleiro. Listas auxiliares para controle dessas opções foram utilizadas, cada nível de jogo possui sua lista representando as seções e cada seção possui uma lista contendo os números disponíveis para uso.

2.1. Requisitos e funcionalidades

O jogo precisa suportar até dois jogadores ao mesmo tempo realizando suas ações a cada rodada, assim como, exibir o somatório de pontos de cada um e o tabuleiro resposta. A cada rodada caso um numero seja revelado no tabuleiro resposta, este deve ter sua posição exibida de forma correta, assim que um jogador completa uma linha ou coluna totalmente, a soma de todos os números daquela linha ou coluna é transformada em pontos para o jogador que a revela.

Em determinadas situações em que uma linha ou coluna se completam ao mesmo tempo, o jogador deve levar a soma dos dois itens simultaneamente. A diagonal principal funciona como o bônus do jogo, caso um jogador complete-a ele ganhará a sua soma multiplicada por dois.

O somatório de pontos de cada jogador e o tabuleiro resposta com as posições reveladas devem ser exibidos em todas as rodadas, a soma de cala linha e coluna ficará visível ao seu lado, entretanto a soma da diagonal não ficará disponível para visualização. Os jogadores não poderão escolher seções que já foram totalmente reveladas assim como não podem escolher números que já foram escolhidos ou que até mesmo não estejam disponíveis no jogo.

A opção de jogar ou sair do game, deverá ser apresentada ao final de cada jogo logo após a revelação do ganhador, os jogadores poderão também escolher seus apelidos e o nível que querem jogar. Dois níveis de jogo devem estar disponíveis, um com o tabuleiro 4x4, apresentando quatro seções preenchidas com números de um a quatro e outro 9x9 com nove seções preenchidas com números de um a nove, como requisito os números não poderiam se repetir dentro de uma mesma seção.

2.2 Algoritmo

Inicialmente para gerar os números que preencherão as seções é importada a biblioteca Random do Python, utilizada para gerar números aleatórios, logo em seguida todas as funções utilizadas no programa são criadas.

As seções do jogo são representadas como matrizes (listas de listas no Python), a função criar tabuleiro difícil gera uma lista de nove números de um a nove sem repeti-los, para que essa repetição seja evitada a função sample da biblioteca Random é utilizada. A lista contendo os nove números é repartida em outras três listas, que representam cada linha da seção, ao final a função retorna uma lista contendo as três listas formadas, representando ao final uma seção do tabuleiro difícil.

Outra função realiza o processo descrito acima de forma semelhante, entretanto,

ela é utilizada para criação das seções do tabuleiro fácil. Uma lista preenchida com quatro números de um a quatro é gerada, a função sample é novamente utilizada para evitar repetição de valores. A lista é repartida em outras duas listas e a função retorna uma lista formada pelas duas listas geradas, formando uma seção do modo fácil.

Para criação das seções que fazem parte do tabuleiro resposta apenas uma função é utilizada, ela recebe como parâmetro o nível de jogo escolhido e partir disso dimensiona o for para gerar a matriz seção. Um range de 2 é utilizado para o modo fácil e um de 3 para o modo difícil, respectivamente atendendo aos requisitos de tamanho de cada seção, sendo 2x2 no tabuleiro 4x4 e 3x3 no 9x9. O processo de criação está representado na figura 1 abaixo, assim como o retorno da matriz resposta gerada.

```
def criar_tab_resposta(escolha):
    if escolha == '2':
        n = 3
    elif escolha == '1':
        n = 2
    matrizj = [ [ 0 for i in range(n)] for j in range(n)]
    return matrizj
```

Figura 1. Criação da seção para o tabuleiro resposta

A exibição da soma de cada uma das linhas do tabuleiro é um dos requisitos do jogo, para realizar esta tarefa uma função somar linhas é criada recebendo como parâmetro a seção gerada com os números aleatórios, a mesma função realiza esta ação tanto para o tabuleiro difícil como para o fácil. Neste momento é realizada a soma das linhas de cada seção, uma por vez, ao final é retornada uma lista contendo os valores obtidos, posteriormente estas somas serão unificadas para exibição no tabuleiro.

A soma das colunas também deve ser apresentada ao longo das rodadas, para isso uma função somar colunas é criada, sendo utilizada para os dois níveis de jogo e recebendo como parâmetros a seção gerada anteriormente e o nível escolhido para dimensionamento do for. Semelhante ao que ocorre em soma linhas, neste momento ocorre unicamente a soma de cada coluna por seção, como mostra a figura 2 abaixo.

```
def soma_colunas(secao,escolha):
    if escolha == '2':
        n = 3
    elif escolha == '1':
        n = 2
    lista_somac = []
    for j in range(n):
        somac = 0
        for i in range(n):
            somac += secao[i][j]
        lista_somac.append(somac)
    return lista_somac
```

Figura 2. Soma das colunas de cada seção

Após o somatório das linhas e colunas por seção, é necessário unificar estas informações para exibição no tabuleiro, para realizar esta ação foi criada a função soma ps (soma por seção), recebendo como parâmetro as listas geradas nas somas de três seções, ou seja, as somas são organizadas de três em três.

Exemplificando, ela recebe as listas preenchidas com as somas das linhas das seções um, dois e três e transforma esta informação numa lista contendo as somas das linhas um, dois e três do tabuleiro, a mesma função é utilizada para soma das colunas, recebendo como parâmetro a soma das colunas das seções um, quatro e sete e retornando uma lista com a soma das colunas um, dois e três, a função sum é utilizada para somar os elementos das listas recebidas. O nível escolhido pelo usuário é utilizado como parâmetro para organizar a mesma ação em uma única função. O processo está descrito na imagem 3 abaixo.

```
def soma_ps(item1,item2,item3,escolha):
    if escolha == '2':
        s1 = sum([item1[0],item2[0],item3[0]])
        s2 = sum([item1[1],item2[1],item3[1]])
        s3 = sum([item1[2],item2[2],item3[2]])
        soma = [s1,s2,s3]
        return soma
    elif escolha == '1':
        s1=sum([item1[0],item2[0]])
        s2=sum([item1[1],item2[1]])
        soma=[s1,s2]
        return soma
```

Figura 3. Unificação das somas por seção

Após a criação de todos os dados do jogo, as funções utilizadas durante as rodadas são criadas. Estas especificamente serão abordadas ao longo da descrição do jogo por estarem diretamente inseridas na partida.

O primeiro menu do programa (jogar ou sair) é apresentado, nele foi utilizado um while que evita a escolha de uma opção diferente das apresentadas inicialmente, além disso o strip é usado para apagar espaços em branco e evitar erros. Esta estrutura coordena todo o jogo reaparecendo ao final de cada partida.

Ao iniciar a partida uma variável coringa é inicializada em zero, ela é utilizada para a função que realiza a soma das linhas e colunas por seção, completando um terceiro parâmetro que não é utilizado para o nível um que só envia duas listas ao invés de três como ocorre no modo difícil. Variáveis que representam os pontos dos jogadores também iniciam em zero neste ponto do código.

Os jogadores devem escolher seus apelidos que serão utilizados ao longo do jogo após o iniciarem, duas verificações são feitas para evitar incoerências no programa. Primeiramente para evitar que o nome seja igual para os dois, as variáveis que armazenam seus nomes são iniciadas como strings vazias e logo em seguida um while realiza este controle enquanto os nomes forem iguais. Internamente outros dois whiles evitam que o jogador deixe seu nome em branco, a função strip é paralelamente utilizada para apagar

espaços em branco antes e depois do nome.

A escolha do nível é o próximo processo pelo qual os jogadores passarão, os dois níveis são apresentados e um while é utilizado para evitar que respostas incorretas sejam aceitas, além disso, consequentemente pelo fato do menu ser feito com strings, o strip é usado para apagar espaços em branco.

As variáveis que acumulam os pontos de cada jogador são iniciadas e uma condicional é utilizada para definir em qual caminho o programa deve seguir após escolha do nível. Escolhendo o nível um que possui tabuleiro 4x4 contendo quatro seções, o programa processará os dados do jogo que serão armazenados em dicionários para manipulação posterior.

Um dicionário contendo as listas de auxilio é criado, dentre estas listas uma é responsável pelo gerenciamento das seções disponíveis e outras quatro gerenciam os números disponíveis em cada seção. Outro dicionário armazena as quatro seções geradas na função criar tabuleiro fácil, assim como as seções do tabuleiro resposta, totalmente zeradas são armazenadas em um outro dicionário. Os dois dicionários utilizam como chaves as seções e como valores os itens armazenados dentro de cada seção, informação detalhada na figura 4 logo abaixo.

```
listas_aux_facil = {'secao_auxiliar_facil': [1,2,3,4],'auxiliar_facil1': [1,2,3,4],'auxiliar_f
secoes_faceis = {'secao1_facil':criar_tabuleiro_facil(),'secao2_facil':criar_tabuleiro_facil()
jogo_facil = {'jogo1_facil':criar_tab_resposta(nivel),'jogo2_facil':criar_tab_resposta(nivel),
```

Figura 4. Dicionários contendo listas e seções

Dicionários também são utilizados para armazenar todas as somas do jogo, em um primeiro momento um dicionário recebe a soma das linhas de cada seção, utilizando como chave uma variável intitulada ‘s1_facil’, representando a soma das linhas de cada seção, nesse caso foi utilizada a um como exemplo e como valor a soma das linhas daquela seção utilizando a função soma linhas. O mesmo modelo de organização é feito para a soma das colunas adicionando um ‘c’ ao final da variável na chave e recebendo como valor a função soma colunas. O processo pode ser visualizado na figura 5 abaixo, a seção um do tabuleiro fácil foi usada como exemplo.

```
soma_linhaf = {'s1_facil':soma_linhas(secoes_faceis['secao1_facil']),'
soma_colunaf = {'s1_facilc':soma_colunas(secoes_faceis['secao1_facil']
```

Figura 5. Armazenamento da soma das linhas e colunas das seções do modo fácil

Para exibição no tabuleiro resposta, a soma de cada linha e coluna deve ser enviada para a função soma ps, para soma das linhas um e dois são enviadas as somas das linhas das seções um e dois do tabuleiro fácil e para as linhas três e quatro, as informações contendo a soma das linhas das seções três e quatro são enviadas.

A mesma função realiza a soma das colunas, para isso são enviadas as somas das

colunas das seções um e três, que retornam a soma das colunas um e dois e a soma das colunas das seções dois e quatro enviam a soma das colunas três e quatro.

Esta função é utilizada tanto para o modo fácil como para o difícil, ao ser utilizada no modo 4x4 ela envia apenas dois parâmetros, para completar os três necessários para o bom funcionamento da função, a variável coringa é enviada assim como nível escolhido para separação dentro da função.

As informações são processadas e armazenadas em um dicionário como mostra a figura 6 abaixo.

```
soma_linha12 = soma_ps(soma_linhaf['s1_facil'],soma_linhaf['s2_facil'],coringa,nivel)
soma_linha34 = soma_ps(soma_linhaf['s3_facil'],soma_linhaf['s4_facil'],coringa,nivel)
soma_coluna12 = soma_ps(soma_colunaf['s1_facilc'],soma_colunaf['s3_facilc'],coringa,nivel)
soma_coluna34 = soma_ps(soma_colunaf['s2_facilc'],soma_colunaf['s4_facilc'],coringa,nivel)
```

Figura 6. Armazenamento da soma de cada linha e coluna do tabuleiro 4x4 como um todo

Antes do começo das rodadas a soma da diagonal é processada, a variável que a armazena é iniciada em zero, no tabuleiro 4x4 a diagonal principal passa pelas seções um e quatro. A estrutura de repetição for é utilizada para realizar a soma como mostra a figura 7 abaixo.

```
soma_diagonalf = 0
for i in range(2):
    for j in range(2):
        if i == j:
            soma_diagonalf += secoes_faceis['secao1_facil'][i][j]
            soma_diagonalf += secoes_faceis['secao4_facil'][i][j]
```

Figura 7. Verificação dos estados

Para controlar as jogadas um while e uma variável acumuladora soma trabalham em conjunto, contando +1 a cada rodada. O modo fácil possui ao todo oito rodadas, assim que a variável atinge esse valor o while é encerrado e o vencedor é revelado.

Na primeira rodada inicialmente o tabuleiro resposta é exibido totalmente zerado, e ao lado de cada linha e coluna as somas calculadas a partir dos valores gerados no tabuleiro com os números randomizados, este por sua vez não será exibido em nenhum momento.

Para exibição do tabuleiro foi criada uma função print seção que recebe como parâmetro o nível de jogo escolhido, as seções respostas estão armazenadas em um dicionário e a exibição foi dividida em duas partes, seções um e dois acima e três e quatro logo abaixo.

Em questões relacionadas a interface do jogo, o tabuleiro resposta é exibido a uma distancia de vinte e três espaços da borda esquerda da tela, os sinais ‘-‘ e ‘=’ foram utilizados no topo, na divisão entre as seções horizontalmente e na base do tabuleiro, nas

bordas o símbolo ‘|’ foi utilizado entre as seções e entre a seção da direita e os valores das somas das linhas.

O for foi utilizado para printar todas as informações, elas foram exibidas seguindo o índice ‘i’ do range do for, o resultado final é exibido na figura 8 abaixo.

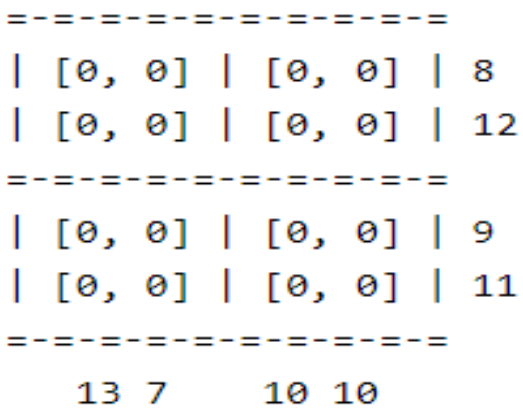

Figura 8. Print do tabuleiro 4x4

Uma única função reúne todas a funções responsáveis pelo funcionamento das rodadas, chamada de jogo, ela é chamada logo após o print do tabuleiro quando se inicia a rodada com o jogador um. A função escolha seção é a primeira a integra-la, ela recebe como parâmetro o nível de jogo escolhido e a partir disso define qual lista auxiliar contendo o número das seções disponíveis será utilizada.

Uma estrutura While True é criada de forma conjunta com o uso do try; except para evitar que o usuário digite letras ou números do tipo flutuante neste menu e quebre o programa, além disso outro while não permite que ele escolha uma seção que não esteja presente na lista auxiliar. Ao final do processo a função retorna à seção escolhida pelo usuário.

Uma nova função integrada a jogo é responsável por traduzir a seção escolhida pelo usuário para as seções e listas auxiliares do jogo, recebendo como parâmetro o nível de jogo escolhido e a seção escolhida na função anterior.

```
if secao == 1:
    se = secoes_faceis['secao1_facil']
    jogo = jogo_facil['jogo1_facil']
    lista_auxiliar = listas_aux_facil['auxiliar_facil1']
```

Figura 9. Tradução de termos

Utilizando a seção um como exemplo, a figura 9 acima relata o processo, caso o usuário tenha escolhido a seção um, a seção gerada como os números randomizados a ser utilizadas será a seção1 fácil, a seção do jogo resposta será a um e a lista auxiliar será aquela contendo os números disponíveis na seção 1.

A função retorna uma tupla preenchida com estas três informações além da seção escolhida pelo usuário anteriormente.

O menu de escolha do numero é apresentado logo em seguida, a presente função que realiza esta ação recebe como parâmetro à tupla gerada anteriormente e o nível escolhido pelo usuário, inicialmente é realizado o desempacotamento da tupla para utilização das informações ao longo do processo.

Uma estrutura while True trabalha em conjunto com um try; except de forma semelhante ao que ocorre no menu de escolha das seções evitando que o usuário digite strings na entrada, além disso um outro while não permite que ele escolha números que não estão disponíveis na lista contendo os valores disponíveis na seção escolhida.

O remove é utilizado logo após a escolha do número, portanto, ele nunca irá escolher um número que já foi revelado, além disso caso a lista auxiliar que representa a seção fique vazia, ou seja, todos os seus valores foram revelados, o numero que representa esta seção na lista auxiliar de seções é removido evitando que o usuário escolha seções que estejam totalmente preenchidas.

Ao final do processo a função escolha número retorna o número escolhido pelo usuário.

Uma função de verificação é criada para revelar a posição do numero aos jogadores, esta revelação ocorre após as jogadas dos dois jogadores, sendo que os mesmos processos apresentados acima ocorrem para o jogador dois.

A função recebe como parâmetro a tupla gerada em traduz termos seção, o número escolhido pelo usuário anteriormente e o nível de jogo informado, em um primeiro momento ocorre o desempacotamento da tupla e logo em seguida dois laços do tipo for revelam o número. A lógica aplicada é simples, um for percorre toda a seção gerada com os números randomizados e encontra o numero escolhido pelo usuário, ao encontra-lo uma condicional o coloca na mesma posição na seção do tabuleiro resposta, a figura 10 abaixo exibe o processo após sua finalização.

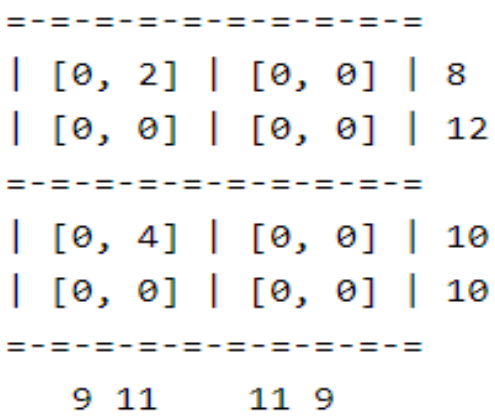

Figura 10. Tabuleiro após revelação do numero

O processo exibido acima exemplifica uma rodada em que um jogador tenha escolhido o número dois da seção um e logo em seguida o jogador dois escolha exibir o número quatro da seção três. A revelação ocorre no início de cada rodada, ou seja, após os dois jogadores realizarem suas jogadas.

A função jogo realiza a soma do tabuleiro resposta para verificar se este possui alguma linha ou coluna igual a do tabuleiro gerado, todos os processos somatórios são semelhantes aos feitos anteriormente para soma do tabuleiro gerado, utilizando as

mesmas funções para realizar a ação. A função jogo é utilizada de forma conjunta para o tabuleiro 4x4 e 9x9, uma tupla contendo a soma de todas as linhas, colunas e diagonal é retornada.

O processo de entrega dos pontos deve ocorrer após cada jogada individual, para isso uma variável acumuladora de pontos recebe a função entregar pontos que utiliza como parâmetro os itens que fazem parte da tupla com as somas geradas no item anterior além de uma variável pontos, que representam os pontos de cada jogador.

A função basicamente verifica se a soma de uma linha ou coluna do tabuleiro resposta é igual ao do tabuleiro original, como este processo ocorre a cada jogada, assim que um jogador realiza sua ação, esta verificação é feita e caso ele complete uma soma, este somatório é entregue a ele como pontos. A verificação responsável pela soma da diagonal é feita e caso está esteja completa o jogador ganha a sua soma multiplicada por dois. Logo após a entrega dos pontos a soma original é zerada evitando que o próximo jogador ganhe aqueles pontos novamente.

O placar parcial é exibido em todas as rodadas, assim que a variável acumuladora de rodadas atinge o valor oito, o jogo termina e uma função placar final exibe o vencedor do jogo realizando uma verificação para descobrir qual possui mais pontos, além disso o tabuleiro com todos os números revelados é exibido, como mostra a figura 11 abaixo.

```
                    >>>PONTUACAO FINAL<<<

PARABENS JOGADOR 2 VOCE VENCEU COM 69 PONTOS

MAIS SORTE NA PROXIMA VEZ JOGADOR 1, VOCE OBTEVE 33 PONTOS
                      =-=-=-=-=-=-=-=-=
                      | [2, 1] | [4, 2] | 0
                      | [4, 3] | [1, 3] | 0
                      =-=-=-=-=-=-=-=-=
                      | [3, 4] | [2, 3] | 0
                      | [1, 2] | [1, 4] | 0
                      =-=-=-=-=-=-=-=-=
                         0 0    0 0
```

Figura 11. Tabuleiro após fim do jogo com placar final

Todos os processos feitos para o nível de jogo fácil são feitos de forma semelhante no modo difícil, ocorre apenas algumas mudanças de parâmetros devido ao maior tamanho do tabuleiro.

Para o print do tabuleiro a função print seção utiliza um range de 3 no tabuleiro difícil, nela três seções são exibidas por vez e a soma das linhas e colunas se faz presente respectivamente na lateral e na base do jogo, como mostra a figura 12.

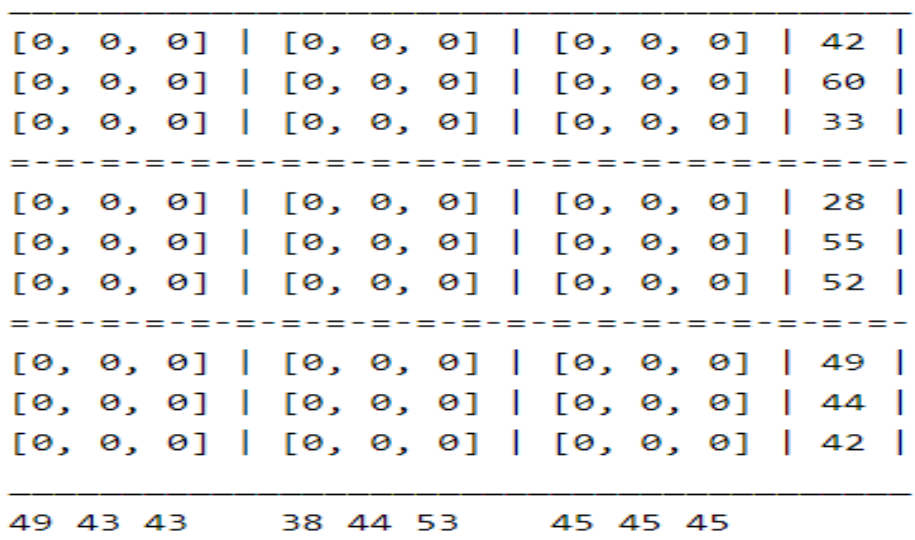

Figura 12. Exibição do tabuleiro no modo difícil (9x9)

Os processos relacionados as jogadas e somas são igualmente feitos, mudanças ocorrem apenas no uso das listas auxiliares que dessa vez são traduzidas em traduz termos seção para aquelas que correspondem ao nível difícil com nove números por seção.

As listas preenchidas com as somas são divididas de três em três partes, ou seja, soma da linha123, soma da coluna 456 e assim sucessivamente. A soma da diagonal ocorre utilizando um for da mesma forma feita no modo fácil, entretanto as seções utilizadas são um, cinco e nove.

Após o fim do jogo o tabuleiro totalmente revelado é exibido, além do placar final utilizado de forma conjunta pelos dois modos.

2.3 **Ordem de codificação**

A linguagem de programação Python exige que todas as funções sejam criadas fora do programa principal, portanto todas as funções são criadas inicialmente, além disso a biblioteca Random é importada para utilização na criação das seções com os números.

O programa principal inicia com os menus de cadastro, ou seja, jogar ou sair, escolha dos nomes e escolha do nível de jogo, após a escolha do nível do jogo o tabuleiro totalmente zerado é exibido e as funções que controlam a escolha das seções, dos números e de verificações são ativadas, todas integradas a uma única função para uma melhor modularidade.

O placar parcial surge somente após a primeira rodada, assim como o tabuleiro, portanto os dois jogadores devem realizar suas ações para que o tabuleiro revele as posições dos números escolhidos e o placar parcial informe se algum deles conseguiu ganhar algum somatório.

Para criação do programa o Windows 11 Home baseado em 64 bits foi utilizado, o ambiente integrado de desenvolvimento utilizado foi o Visual Studio Code utilizando o Python 3.11.

3. Resultados e discussões

De forma geral diversos testes foram feitos para verificação das jogadas e entrega dos

pontos, além disso, todas as entradas foram otimizadas e aprimoradas para evitar a aceitação de dados errôneos e incoerentes ao que se solicitava.

Diversos jogos teste foram feitos nos dois níveis utilizando um teste de mesa com a contagem de pontos para cada jogador. Era observado se os valores eram coerentes a aqueles que se correspondiam entre tabuleiro resposta e somas do tabuleiro gerado.

3.1 Dados de entrada

O usuário deve seguir primeiramente a opção que deseja no menu de entrada, após isso, caso a opção escolhida seja jogar eles terão que escolher seus nomes, não havendo limitações para a criação, portanto, podem ser usados letras, números e símbolos diversos o programa apenas não permitirá que estes nomes sejam iguais.

Os jogadores poderão escolher o nível através de um outro menu, não serão aceitas respostas diferentes a aquelas que foram apresentadas. Ao iniciar o jogo dois dados são disponibilizados para o usuário, a escolha da seção e do numero a ser revelado. Não será permitido a escolha de seções que não existem ou já foram totalmente reveladas ou números que não estão presentes no jogo e já foram escolhidos.

3.2 Dados de saída

O programa realizará o processamento de todas as informações informadas pelo usuário, revelando os números que foram escolhidos por cada jogador e entregando os pontos caso um jogador tenha completado uma linha ou coluna, além disso, caso ele complete os dois itens ao mesmo tempo, a aplicação entrega o somatório da mesma forma.

A soma da diagonal também é verificada e esta é entregue com seu valor multiplicado por dois. Após a entrega dos pontos o placar parcial vai informando a cada rodada os pontos de cada jogador, inclusive caso esta informação esteja zerada. Na saída final o programa exibe o placar final com o vencedor e o tabuleiro resposta totalmente completo.

3.3 Testes e erros

Todos os menus foram testados com diversos tipos de dados de entrada para averiguação e aprimoramento dos dados. Em cada menu um while foi colocado para evitar a entrada de informações incoerentes, letras e números do tipo float foram inseridos nos menus do tipo int, para evitar ValueError uma estrutura while true foi inserida.

Para evitar a escolha de seções ou números inválidos o programa utiliza listas de auxilio contendo estes números, testes foram feitos para verificar se o programa aceitava apenas números pertencentes a estas listas, além disso foi verificado se um numero escolhido ou uma seção vazia estava sendo removida da lista.

Jogos testes foram feitos para observar se as somas das linhas e colunas eram entregues de forma coerente aos jogadores que as completavam. Foi averiguado também se a soma da diagonal estava sendo entregue como bônus ao jogador que a completava, garantindo uma pontuação dobrada.

Nenhum erro foi localizado após todas as otimizações e aprimoramentos feitos para evitar a entrada de dados inválidos e a incorreta entrega dos pontos aos jogadores. O programa funciona em qualquer situação, possuindo como limitação apenas a participação de dois jogadores por vez e unicamente dois níveis de jogo disponíveis.

4. Conclusão

O jogo está apresentado de forma otimizada, erros não foram encontrados em sua versão final, todos os requisitos apresentados foram cumpridos, entretanto, nenhuma funcionalidade foi adicionada a aplicação de forma extra.

Pensando em aprimoramentos e melhorias em futuras versões, a aplicação poderá suportar mais de dois jogadores e níveis de jogo extras ao que estão em funcionamento hoje, aumentando a oferta de dificuldade e consequentemente melhorando a disponibilidade de opções ao jogo.

A oferta de números pode também ser maior que a utilizada atualmente, aumentando a dificuldade de realizar as somar do jogo por parte dos jogadores.

5. Referências

Disponível em: https://www1.folha.uol.com.br/webstories/tec/2021/08/a-historia-do-sudoku/. Acesso em:13/out/2022