Relatório:Sudoku em Python

Professor: Alisson Zanetti

Matéria: Programação

Alunos: Gabriela Nietiedt e Daniel Masson

Data: 17/08/25

Lógica Utilizada

Estrutura do Tabuleiro

O tabuleiro foi representado como uma **matriz 9x9** (lista de listas em Python). Cada posição da matriz pode conter um número de **1** a **9** ou o valor **0**, que representa uma célula vazia.

Regras do Sudoku

- 2 números não podem ser repetidos na mesma linha, coluna ou matriz 3x3
- O jogador deve solucionar o tabuleiro com base neste estando incompleto e com x números de casas vazias dependendo da dificuldade

Geração de Tabuleiros Preenchidos

Para gerar um tabuleiro completo válido, foi utilizada a técnica de backtracking (tentativa e erro):

1. A função **achar_vazio()** encontra a primeira célula vazia (com valor 0).
2. Para essa célula, é criada uma lista de números de 1 a 9, embaralhados aleatoriamente.
3. O algoritmo tenta colocar cada número por meio da função **válido().**
4. Caso seja válido, o número é colocado e a função é chamada novamente.
5. Se em algum ponto não for possível continuar, o número é retirado (backtrack) e o processo segue com outro valor.
6. A função executa até todo tabuleiro estar preenchido.

Verificação de Validade

A função **`valido()`** é responsável por verificar se o numero é valido naquela determinada posição do tabuleiro que um número pode ser inserido em uma determinada do tabuleiro Ela recebe como parâmetros: o tabuleiro atual, a linha, a coluna e o número a ser testado.

1. **Verificação da linha e coluna:**

- Percorre todos os elementos da linha escolhida, Se o número já existir nela, a função retorna **`False`**. Faz o mesmo para as colunas.

2. **Verificação do bloco 3x3**

   - Cada tabuleiro de Sudoku é dividido em 9 blocos de 3x3. Para identificar em qual bloco a posição está, o programa usa divisão inteira:

     `linha // 3 * 3` → retorna o início do bloco na vertical.

     `coluna // 3 * 3` → retorna o início do bloco na horizontal.

   - A partir desses índices, a função percorre apenas as 9 células do bloco correspondente. Se o número já estiver nesse bloco, retorna **`False`**

3. **Criação de Puzzles (Dificuldade)**

   Com um tabuleiro completo pronto, o programa remove números de acordo com o nível de dificuldade escolhido pelo jogador:

   Fácil → 35 casas removidas, Médio → 45 casas removidas, Difícil → 55 casas removidas, Teste → 5 casas removidas
   A remoção é feita aleatoriamente, e o resultado é o **jogo**.

   Durante a execução do jogo:

   1. O jogador escolhe uma dificuldade.
   2. O programa apresenta o tabuleiro no console com pontos (.) representando casas vazias.
   3. O jogador insere linha, coluna e número desejado.
   4. O programa verifica:
      - Se a posição existe;
      - Se a casa já estava preenchida;
      - Se o número inserido é o correto comparando com a solução do tabuleiro.
   5. O tabuleiro é atualizado e reimpresso a cada jogada.
   6. O jogo termina quando o puzzle está completamente igual à solução, exibindo a mensagem de vitória.

   Fontes: W3Schools,
   https://github.com/tejasmorkar/sudoku-solver

   https://www.youtube.com/watch?v=G_UYXzGuqvM&t=178s

   (O ChatGPT-5 foi utilizado para explicar melhor o funcionamento do backtracking)