RESOLUÇÃO PARALELA DE SUDOKU COM MULTITHREADING

JASMIM SABINI E VINICIUS MARTINS

Classic Sudoku

				5				
	7	2				4	6	
3			6		1			9
9			2		6			4
	6	7				1	3	
				8				
6								7
	8						5	
		5	9	6	2	3		

O que é Sudoku?

- Jogo de lógica numérica.
- Consiste em preencher uma grade 9x9 com números de 1 a 9.
- Regras: Não repetir números na mesma linha, coluna e bloco 3x3.

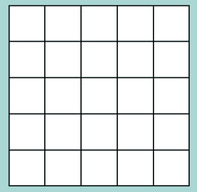
Objetivo do Projeto

Desenvolver uma aplicação que utiliza multithreading para acelerar o processo de resolução ou verificação de um tabuleiro de Sudoku.

- Identificar erros em linha, coluna ou bloco.
- Demonstrar o uso de threads para executar tarefas em paralelo, acelerando a verificação.

9 linhas

9 colunas



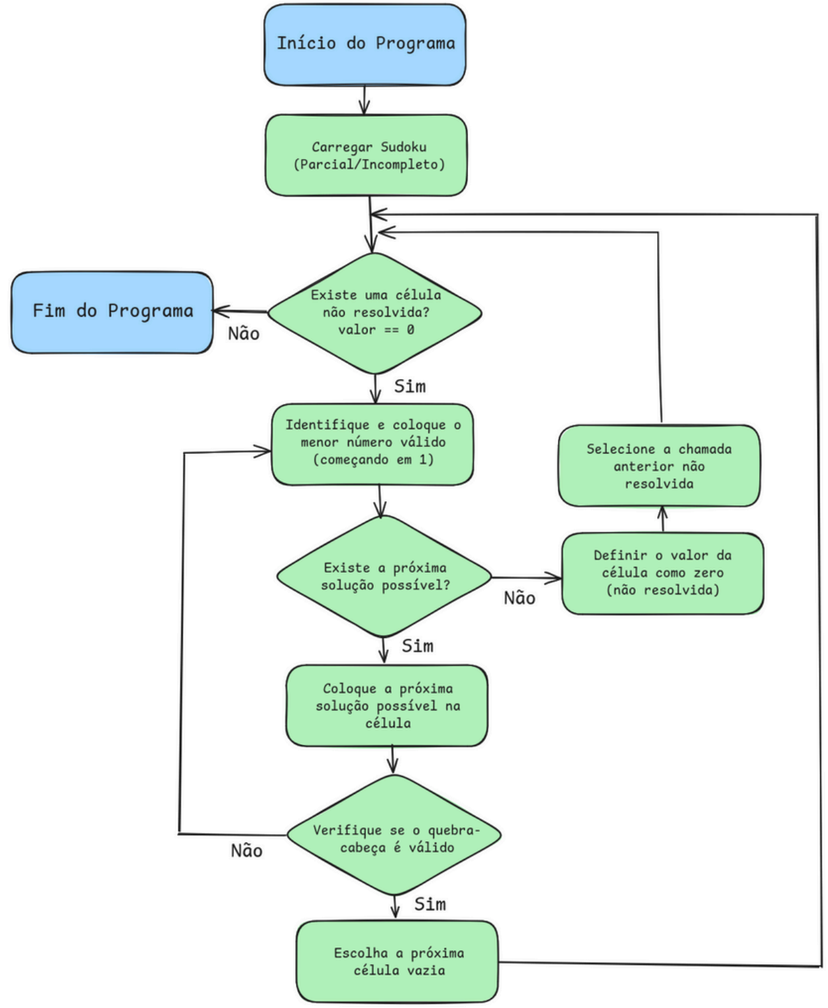
9 blocos 3 x 3

Resolução do tabuleiro Sudoku

Utiliza o algoritmo de Backtracking Recursivo.

Processo:

- Busca a primeira célula vazia.
- Tenta preencher com números de 1 a 9.
- Verifica se o número é válido (linha, coluna, bloco).
- Se não for possível, faz backtracking e tenta outra combinação.
- Repete até encontrar a solução.



Código

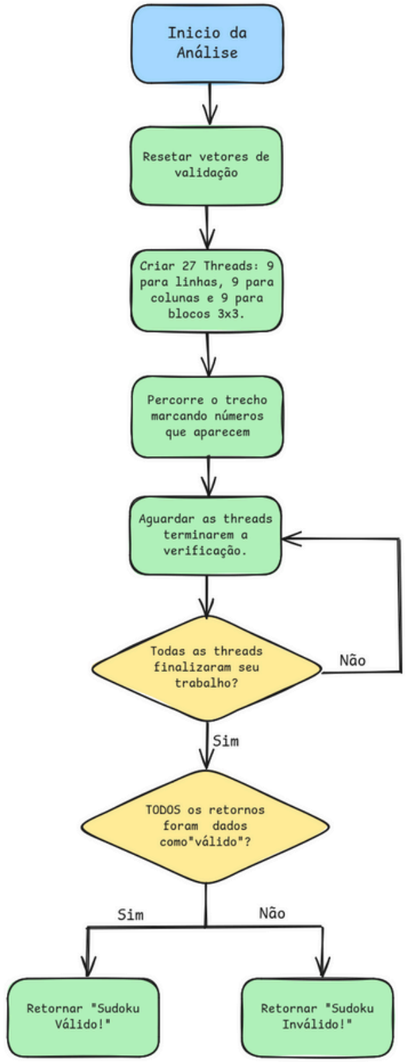
```c
// -----------Resolução com Backtracking---------------
int resolver_sudoku(int sudoku_id) {
    int linha, coluna;
    if (!encontrar_vazio(sudoku_id, &linha, &coluna))
        return 1; // Sudoku resolvido

    for (int num = 1; num <= 9; num++) {
        if (seguro(sudoku_id, linha, coluna, num)) {
            sudokus[sudoku_id][linha][coluna] = num;

            if (resolver_sudoku(sudoku_id))
                return 1;

            sudokus[sudoku_id][linha][coluna] = 0; // Backtrack
        }
    }
    return 0;
}
```

Verificação do tabuleiro de Sudoku



Para tabuleiros já preenchidos, fazemos a validação com threads:

- 9 threads para linhas.
- 9 threads para colunas.
- 9 threads para blocos 3x3.
- Cada thread verifica uma unidade independente e sinaliza se há repetições ou ausência de numeros.
- Em um caso positivo, a thread retorna “1”.

Código

```c
// -----------Funções das threads-----------------
void *verifica_linha(void *param) {
    parametro *p = (parametro *) param;
    int linha = p->index;
    int sudoku_id = p->sudoku_id;
    int check[SIZE] = {0};

    for (int j = 0; j < SIZE; j++) {
        int num = sudokus[sudoku_id][linha][j];
        if (num < 1 || num > 9 || check[num - 1] == 1) {
            printf(" Thread Linha %d do Sudoku %d encontrou erro!\n", linha + 1, sudoku_id + 1);
            pthread_exit(NULL);
        }
        check[num - 1] = 1;
    }

    linha_valida[linha] = 1;
    printf(" Thread Linha %d do Sudoku %d finalizada com sucesso!\n", linha + 1, sudoku_id + 1);
    pthread_exit(NULL);
}
```

Código

```c
// -------------------- Validação --------------------------
void validar_sudoku(int sudoku_id) {
    pthread_t threads[27];
    parametro params[27];
    int t = 0;

    // Resetando vetores de validação
    for (int i = 0; i < SIZE; i++) {
        linha_valida[i] = 0;
        coluna_valida[i] = 0;
        bloco_valida[i] = 0;
    }

    // Criando threads para linhas
    for (int i = 0; i < SIZE; i++) {
        params[t] = (parametro){i, sudoku_id};
        pthread_create(&threads[t], NULL, verifica_linha, &params[t]);
        t++;
    }
```

```c
    // Esperando todas as threads terminarem
    for (int i = 0; i < 27; i++) {
        pthread_join(threads[i], NULL);
    }

    // Verificando resultado final
    int valido = 1;
    for (int i = 0; i < SIZE; i++) {
        if (linha_valida[i] == 0 || coluna_valida[i] == 0 || bloco_valida[i] == 0) {
            valido = 0;
            break;
        }
    }

    if (valido) {
        printf(" Sudoku %d é VÁLIDO!\n\n", sudoku_id + 1);
    } else {
        printf(" Sudoku %d é INVÁLIDO!\n\n", sudoku_id + 1);
    }
}
```

Testes Realizados

Para validar nossa solução fizemos os seguintes testes:

- 3 Sudokus incompletos foram inseridos para resolução e validação.
- 1 Sudoku “base” (resolvido) foi inserido, e um erro foi propositalmente inserido. O algoritmo deve acusar e identificar erro de linha, coluna e bloco.

```
    // Sudoku completo correto (para gerar erros)
    int base[SIZE][SIZE] = {
        {5,3,4,6,7,8,9,1,2},
        {6,7,2,1,9,5,3,4,8},
        {1,9,8,3,4,2,5,6,7},
        {8,5,9,7,6,1,4,2,3},
        {4,2,6,8,5,3,7,9,1},
        {7,1,3,9,2,4,8,5,6},
        {9,6,1,5,3,7,2,8,4},
        {2,8,7,4,1,9,6,3,5},
        {3,4,5,2,8,6,1,7,9}
    };

    // Inserindo 5 na Linha 1 Coluna 2
    for (int i = 0; i < SIZE; i++)
        for (int j = 0; j < SIZE; j++)
            sudokus[3][i][j] = base[i][j];
    sudokus[3][0][1] = 5;

}
```

Saídas do Console

```
===== PARTE 1: RESOLVER + VALIDAR =====

--- Sudoku 1 ---
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

```
Resolvendo Sudoku 1...
Sudoku 1 resolvido com sucesso!

--- Sudoku 1 ---
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Saídas do Console

```
Iniciando validação do Sudoku 1...
Thread Linha 1 do Sudoku 1 finalizada com sucesso!
Thread Linha 2 do Sudoku 1 finalizada com sucesso!
Thread Linha 3 do Sudoku 1 finalizada com sucesso!
Thread Linha 4 do Sudoku 1 finalizada com sucesso!
Thread Linha 5 do Sudoku 1 finalizada com sucesso!
Thread Linha 6 do Sudoku 1 finalizada com sucesso!
Thread Linha 7 do Sudoku 1 finalizada com sucesso!
Thread Linha 8 do Sudoku 1 finalizada com sucesso!
Thread Linha 9 do Sudoku 1 finalizada com sucesso!
Thread Coluna 2 do Sudoku 1 finalizada com sucesso!
Thread Coluna 1 do Sudoku 1 finalizada com sucesso!
Thread Coluna 3 do Sudoku 1 finalizada com sucesso!
Thread Coluna 4 do Sudoku 1 finalizada com sucesso!
Thread Coluna 5 do Sudoku 1 finalizada com sucesso!
```

```
Thread Coluna 5 do Sudoku 1 finalizada com sucesso!
Thread Coluna 6 do Sudoku 1 finalizada com sucesso!
Thread Coluna 7 do Sudoku 1 finalizada com sucesso!
Thread Coluna 8 do Sudoku 1 finalizada com sucesso!
Thread Coluna 9 do Sudoku 1 finalizada com sucesso!
Thread Bloco (1,1) do Sudoku 1 finalizada com sucess
Thread Bloco (1,2) do Sudoku 1 finalizada com sucess
Thread Bloco (1,3) do Sudoku 1 finalizada com sucess
Thread Bloco (2,1) do Sudoku 1 finalizada com sucess
Thread Bloco (2,2) do Sudoku 1 finalizada com sucess
Thread Bloco (2,3) do Sudoku 1 finalizada com sucess
Thread Bloco (3,1) do Sudoku 1 finalizada com sucess
Thread Bloco (3,2) do Sudoku 1 finalizada com sucess
Thread Bloco (3,3) do Sudoku 1 finalizada com sucess
Sudoku 1 é VÁLIDO!
```

Referências

LAMOCCA, Leandro. Sudoku Validation using Multithreading – Final Project Group 7. Oakland University, 2023. Disponível em: https://www.secs.oakland.edu/~llamocca/Courses/ECE4772/F23/FinalProject/Group7_sudoku.pdf. Acesso em: 28 maio 2025.

SANKAR, R.; MILLER, Geoffrey. Sudoku Solver using Parallel Processing. University at Buffalo, 2014. Disponível em: https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Sankar-Spring-2014-CSE633.pdf. Acesso em: 28 maio 2025.

CLASSIC Sudoku Puzzle Image. Blogger, [s.d.]. Disponível em: https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhfQHNo4387gLAhd61H8v1St5Bsqr8OLvzNYBx3WLkSbbah9xctWfqTzhRzvrhrBB7t1YrHFk03LqMvraYX_Cg9ILoifPjiFHsJP295cOSzb0Ry9UQWLXifK_XL8lQf7fjJX9Dd1ar0yrET/s640-rw/classic-sudoku-puzzle-asc2018-theme-smileplease.png. Acesso em: 28 maio 2025.

Obrigado!