UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THIAGO ALMEIDA DE OLIVEIRA

Sudoku: Um estudo sobre algoritmos de geração, classificação e resolução para aplicação educacional

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Renato Perez Ribas

Porto Alegre
2024

AGRADECIMENTOS

Agradeço aos meus pais, Lucinda e Sérgio Oliveira, pela dedicação em me proporcionar a melhor educação possível, e pelo apoio constantemente presente.

Agradeço a minha vó, Lourdes Rachinhas, que até o presente momento, nunca pôde ver um neto concluir a faculdade, pelo apoio e amor incondicional aos netos.

Agradeço aos professores e professoras do Instituto de Informática e da UFRGS, pela contribuição na formação deste jovem empenhado em ser a melhor versão de si mesmo.

RESUMO

Jogos são cada vez mais utilizados na educação de jovens, possuindo grande potencial para o desenvolvimento de muitas habilidades importantes. O Sudoku é um quebra-cabeça de raciocínio lógico com diversos tipos e variações. Nesse contexto, este trabalho tem como objetivo analisar, implementar e validar algoritmos capazes de gerar quebra-cabeças Sudoku regulares válidos, bem como mensurar o nível de dificuldade dele, e analisar algoritmos capazes de resolver esses quebra-cabeças Sudoku gerados. Foram implementados algoritmos capazes de gerar, resolver e classificar um nível de dificuldade, e cada um foi validado com experimentos utilizando datasets disponíveis na internet. Essas análises possibilitam a utilização do Sudoku em uma aplicação para estimular o pensamento computacional na educação.

Palavras-chave: Computação. Sudoku. Educação. Algoritmo.

ABSTRACT

Games are increasingly used in the education of young people, possessing great potential for the development of many important skills. Sudoku is a logic puzzle with various types and variations. In this context, this work aims to analyze, implement, and validate algorithms capable of generating valid regular Sudoku puzzles, as well as measuring their level of difficulty, and analyzing algorithms capable of solving these generated Sudoku puzzles. Algorithms were implemented to generate, solve, and classify a level of difficulty, and each one was validated with experiments using datasets available on the internet. These analyses enable the use of Sudoku in an application to stimulate computational thinking in education.

Keywords: Computation, Sudoku, Education, Algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Diante da influência da tecnologia na sociedade atual, e com a inclusão da computação na BNCC(Base Nacional Comum Curricular), em que um dos pilares é o pensamento computacional, o Sudoku se apresenta como um quebra-cabeça adequado para desenvolver essa habilidade, podendo desempenhar um papel importante na formação de crianças e adolescentes.

Diversos quebra-cabeças e jogos de lógica têm se popularizado ao longo dos anos, de muitas maneiras diferentes. Para algumas pessoas, esses puzzles são vistos como uma forma de entretenimento, de forma a testar diferentes capacidades humanas. Para outros, podem ser vistos como um problema matemático ou lógico a ser resolvido em diversas áreas do conhecimento, inclusive na própria ciência da computação, como, por exemplo, o problema das 8 damas(ou N-damas), (BALL, 1905) advindo do jogo de tabuleiro chamado xadrez. Esse é um problema NP-completo clássico, que consiste em distribuir N damas em um tabuleiro de xadrez NxN, de forma que não haja mais de uma dama na mesma linha, coluna ou diagonal. Existem vários jogos de entretenimento popular associados a problemas da computação. Esses puzzles exploram diferentes habilidades cognitivas, estimulam o raciocínio lógico e são capazes de promover uma forma algorítmica de pensar em possíveis soluções para problemas propostos.

Nesse contexto, o LogicaMente é um projeto coordenado pelo professor Renato Perez Ribas, que tem como objetivo utilizar diversos jogos como um instrumento pedagógico para a formação de indivíduos, promovendo a capacidade de resolução de problemas em exercícios colaborativos, além dos benefícios cognitivos.

Existem vários jogos que possuem grande potencial para esse objetivo, como, por exemplo, o Tangram, os Enigmas de Einstein, os Desafios de Tabuleiro e o Sudoku. Esses quatro jogos são apresentados e abordados na proposta de uma metodologia de integração na rotina escolar, no trabalho de conclusão de curso (GONZAGA, 2023).

Esses e outros jogos têm grandes benefícios cognitivos que podem ser explorados, em união com o estímulo ao pensamento computacional do indivíduo, gradualmente aumentando o padrão de complexidade do jogo proposto. Essa metodologia, aliada a uma didática cuidadosa e a um ambiente colaborativo, atrai o público e o motiva a desenvolver o raciocínio lógico e a capacidade de resolver problemas, além de promover diversas interações sociais.

1.1 Sudoku

O Sudoku é um jogo de quebra-cabeça inventado por Howard Garns no século XX, com suas primeiras aparições em revistas publicadas nos Estados Unidos. Altamente popular no mundo, o jogo emergiu como um passatempo envolvente que desafia os jogadores a exercitarem suas habilidades lógicas, dedutivas e de resolução de problemas. Apesar de ser um jogo de regras de simples compreensão, ele esconde uma grande gama de possibilidades, atraindo dois tipos principais de entusiastas: jogadores buscando o entretenimento, e estudiosos buscando soluções para problemas relativos ao jogo. O Sudoku é comumente jogado com números, mas quaisquer outros símbolos podem ser usados, como letras e figuras, a depender da aplicação. As figuras como símbolos do Sudoku poderiam, por exemplo, ser uma melhor opção quando o público-alvo da aplicação do jogo é composto por crianças. Essa liberdade na escolha de símbolos, aliada a outras ideias, propicia uma enorme variedade do Sudoku que transcende os quebra-cabeças mais populares. Existem diversos níveis de dificuldade no próprio Sudoku tradicional, e existem variações que permitem a criação de mais níveis de dificuldade e até mesmo junções com outros desafios, a exemplo de Sudokus aritméticos que misturam matemática ao jogo, ou mesmo Sudokus de palavras que unem elementos de caça-palavras ao jogo.

1.2 Motivação

Com a grande adesão social à tecnologia somada a sua evolução exponencial, e diante da inclusão da computação na Base Nacional Comum Curricular(BNCC), onde o pensamento computacional é uma das habilidades relacionadas, abre-se a necessidade de encontrar os melhores caminhos para conferir um processo educacional eficiente e inclusivo na área da tecnologia, levando em consideração o contexto sociocultural de cada aluno. O pensamento computacional se apresenta como uma importante competência do aluno para a formulação e resolução de problemas em um caráter algorítmico, de forma eficiente. Existem diversos jogos que envolvem o uso do raciocínio lógico, criatividade e reconhecimento de padrões, permitindo o uso deles como um instrumento pedagógico poderoso para a aplicação do pensamento computacional. O ensino de diversos tópicos em várias áreas do conhecimento tem sido cada vez mais gamificado, ou seja, há uma aplicação de mecanismos presentes em jogos para se consolidar a transferência de conhecimento. Por exemplo, a aplicação de testes de conhecimento multidisciplinar via quiz-

zes, explorada por plataformas como Kahoot (KAHOOT, 2013). A gamificação traz uma maior aproximação entre o aluno e o seu desenvolvimento, possibilitando formas lúdicas de aprendizado que contribuem para o desenvolvimento pedagógico. O Sudoku é um jogo desplugado que pode ser utilizado como uma ferramenta para o devido preparo do aluno na prática do pensamento computacional. Nesse contexto, o projeto LogicaMente, criado pelo professor Renato Perez Ribas, propõe o uso de jogos para o ensinamento do pensamento computacional, por meio de um método de aplicação de alguns jogos de forma que o nível de dificuldade oferecido seja gradual.

1.3 Objetivo

Este trabalho é uma parte do projeto LogicaMente, mencionado na seção anterior, para construir uma plataforma computacional que busca proporcionar o uso de alguns jogos como aplicação educativa no âmbito da computação. Seguindo a proposta apresentada no trabalho de conclusão de curso de Bruna Gonzaga (GONZAGA, 2023), o Sudoku é um dos jogos sugeridos para a prática do pensamento computacional para crianças do primeiro ao quinto ano do ensino fundamental. Essa aplicação computacional necessitará de um algoritmo para gerar Sudokus válidos de diferentes tamanhos que possam ser usados em sala de aula, bem como um algoritmo para classificar os Sudokus gerados em diferentes níveis de dificuldade. Por fim, também precisará de um algoritmo capaz de resolver os Sudokus gerados. Nesse contexto, este trabalho propõe a utilização de alguns métodos para a geração, classificação de nível de dificuldade e resolução de quebra-cabeças Sudoku, para ser utilizado na plataforma computacional mencionada.

1.4 Estrutura do texto

O Capítulo 2 apresenta alguns fundamentos e revisão técnica para melhor informar o leitor do que será abordado nos capítulos seguintes. O Capítulo 3 traz alguns trabalhos e referências de diferentes autores relacionados ao tema. O Capítulo 4 apresenta a proposta do trabalho e sua implementação. O Capítulo 5 discute resultados da proposta apresentada, com experimentos e validação da proposta. Finalmente, o Capítulo 6 elabora a conclusão e comenta possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão abordados alguns tópicos necessários relacionados para melhor entendimento do trabalho. As regras principais do Sudoku serão explicadas, junto a detalhes adicionais do funcionamento do jogo, técnicas de resolução do jogo e variações. Ademais, serão apresentados alguns conceitos relevantes para a proposta, como as tabelas Hash, a linguagem de programação Python e o algoritmo de busca em profundidade.

2.1 Sudoku

O Sudoku é um jogo que possui similaridade com os Quadrados Latinos, construção matemática de Leonhard Euler (ENSINO, 2024), trazendo inclusive a hipótese de que o Sudoku seria inspirado nesses quadrados. Segundo sua definição, um Quadrado Latino de ordem N é uma matriz NxN em que um símbolo não pode se repetir em linhas ou colunas. O Sudoku é um Quadrado Latino de ordem N, contendo N símbolos para serem preenchidos, com a restrição adicional de que a matriz(comumente chamada de grade) é dividida em N subgrades(ou regiões), cada uma delas com um tamanho menor do que a grade, de forma que as regiões juntas ocupem toda a grade, e os símbolos utilizados também devem aparecer exatamente uma vez em cada uma dessas regiões. Dessa forma, sucintamente, o Sudoku tem 3 regras principais:

1. Não pode haver símbolos repetidos em uma linha;
2. Não pode haver símbolos repetidos em uma coluna;
3. Não pode haver símbolos repetidos em uma região.

O objetivo do jogador é encontrar uma solução que preencha toda a grade seguindo essas 3 regras principais descritas, a partir de uma grade inicial parcialmente preenchida com alguns valores iniciais, chamados de pistas. Para ser considerado um Sudoku, a grade inicial deve garantir que exista uma solução e que ela seja única. A Figura 2.1 ilustra um exemplo de um Sudoku não resolvido contendo 9 linhas, 9 colunas e divididos em regiões de 3 linhas por 3 colunas.

Normalmente, os símbolos usados em Sudokus são números inteiros, sendo o Sudoku de ordem 9 o mais comum, utilizando números de 1 a 9, frequentemente chamado de Sudoku clássico(Figura 2.1). Contudo, existem outras variantes do Sudoku que não seguem completamente esse formato. Algumas delas são apresentadas a seguir:

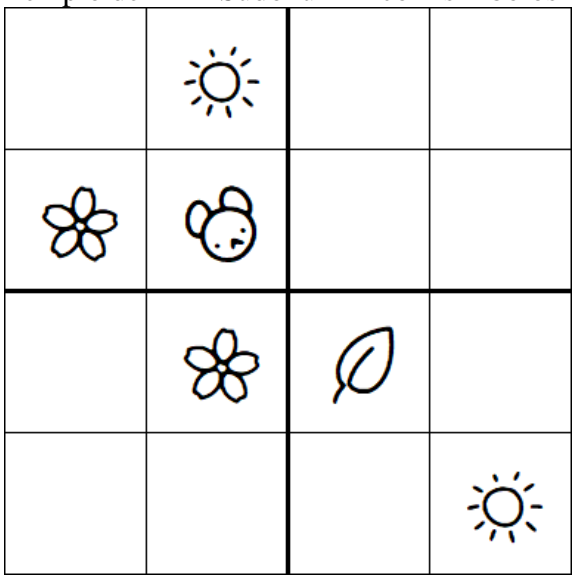
Figura 2.1: Exemplo de Sudoku

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

Fonte: (WIKIPEDIA-SUDOKU, 2024)

- Sudokus regulares: Um Sudoku regular NxN possui qualquer número de linhas e colunas, e suas regiões precisam ter um formato de paralelogramo. É nessa classificação que o Sudoku clássico se insere. Mas além dele, existem, por exemplo, Sudokus 4x4 com regiões quadradas 2x2, contendo 4 símbolos para serem preenchidos, Sudokus 6x6 com regiões retangulares 2x3, com 6 símbolos para preencher, dentre outros. As versões de Sudokus regulares menores que a clássica também podem ser chamadas de “Mini Sudokus”. As Figuras 2.2 e 2.3 mostram exemplos de Mini Sudokus, sendo dois exemplos de Sudokus regulares e não clássicos. A Figura 2.3 ilustra também o uso de desenhos como símbolos, no lugar dos números.

Figura 2.3: Exemplo de Mini Sudoku 4x4 com símbolos não numéricos



Fonte: (SUDOKU-PUZZLES, 2024)

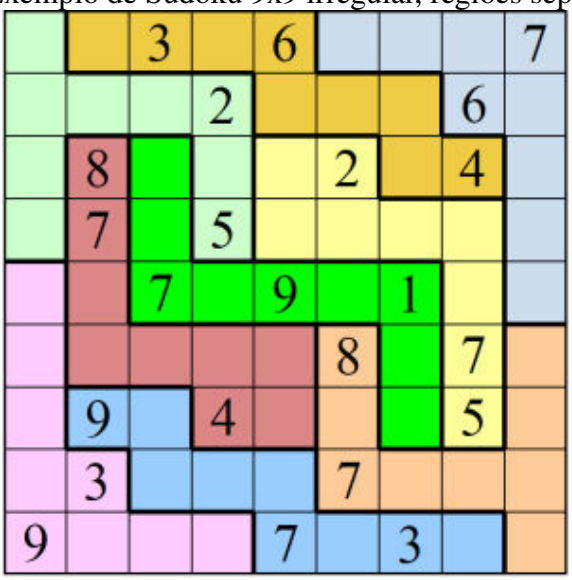
Figura 2.2: Exemplo de Mini Sudoku 6x6

	3		4		
		5	6		3
			1		
	1		3		5
	6	4		3	1
		1		4	6

Fonte: (SUDOKU-PUZZLES, 2024)

- Sudokus irregulares: Um Sudoku irregular segue as mesmas regras principais do Sudoku regular, e também pode ter NxN linhas e colunas, mas suas regiões têm formatos variados e desconexos. São também chamados de "Jigsaw Sudokus". A Figura 2.4 ilustra um exemplo.

Figura 2.4: Exemplo de Sudoku 9x9 irregular, regiões separadas por cor



Fonte: (MEDIA, 2024)

- Sudoku diagonal: O Sudoku diagonal possui as mesmas regras de Sudokus regulares, mas inclui uma restrição adicional em que os símbolos não podem se repetir nas diagonais principais da grade, adicionando um grau extra de complexidade. A Figura 2.5 ilustra um exemplo, com os valores das diagonais principais marcados em vermelho.

Figura 2.5: Exemplo de Sudoku Diagonal

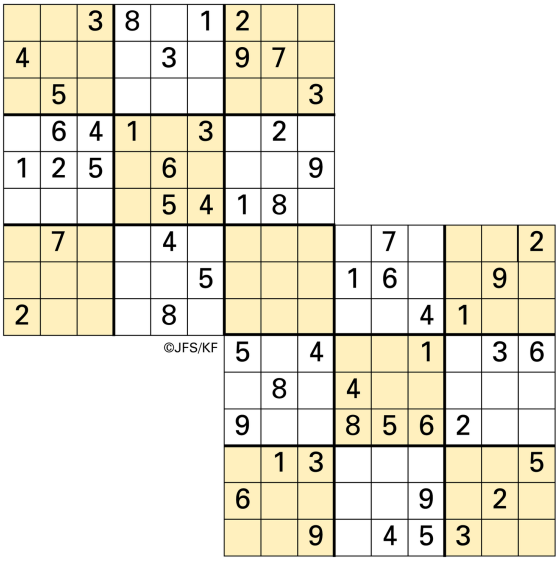
4	1	5	6	3	8	9	7	2
3	6	2	4	7	9	1	6	5
7	8	9	2	1	5	3	6	4
9	2	6	3	4	1	7	5	8
1	3	8	7	5	6	4	2	9
5	7	4	9	8	2	6	3	1
2	5	7	1	6	4	8	9	3
8	4	3	5	9	7	2	1	6
6	9	1	8	2	3	5	4	7

Fonte: (COMMONS, 2024)

- Sudokus gêmeos: Este é um quebra-cabeça em que existem duas grades de Sudokus regulares compartilhadas, normalmente duas clássicas. Nessa variante, cada símbolo em uma grade mapeia outro símbolo na outra grade. Isto é, se existe um símbolo X inserido em uma posição (A, B), sendo A a linha, e B a coluna, em uma grade G1, e um símbolo Y inserido na mesma posição (A, B) em outra grade G2, toda ocorrência de X em G1 implica em uma ocorrência de Y em G2 na mesma posição (A,B). Normalmente é preciso resolver ambos os Sudokus para obter a solução final.

  Existem também Sudokus gêmeos em que somente uma ou mais regiões das grades é compartilhada, ao invés da grade inteira. Nesses casos, não há nenhum tipo de mapeamento. A região compartilhada deve ter exatamente os mesmos números para ambos os Sudokus. A Figura 2.6 mostra um exemplo de Sudokus gêmeos compartilhando a região superior esquerda de um Sudoku clássico.

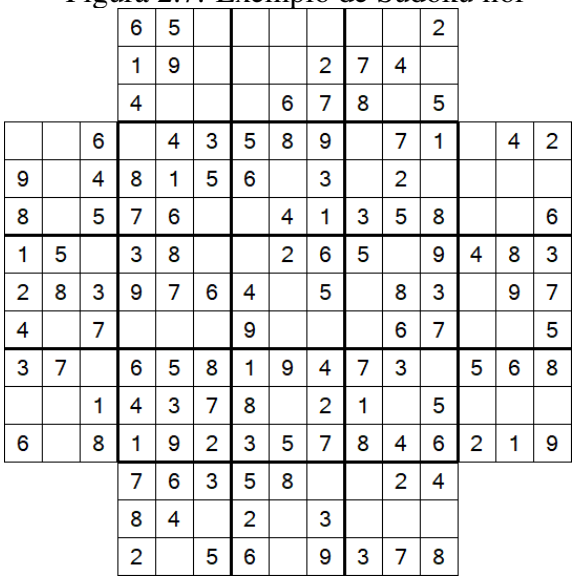
Figura 2.6: Exemplo de Sudokus Gêmeos compartilhando uma região



Fonte: (FEATURES, 2024)

- Outras variações específicas: Dentre a ampla gama de variações do Sudoku, existem algumas com características mais específicas. Um exemplo disso são Sudokus na forma de diversas figuras, como o Sudoku Circular, onde todos os símbolos estão presentes em um círculo concêntrico com regiões divididas por fatias, ou como o Sudoku Flor, que possui uma organização da grade e das regiões no formato de uma flor. Ambos ainda seguem as mesmas regras principais de outros Sudokus. Outro possível exemplo são Sudokus que criam restrições aritméticas, tendo números como símbolos, e selecionando conjuntos de células cujos símbolos precisem resultar em uma soma ou produto previamente especificado.

Figura 2.7: Exemplo de Sudoku flor



Fonte: (SUDOKU-PUZZLES, 2024)

2.2 Técnicas de resolução

Neste trabalho, a proposta será apresentada utilizando Python 3.8 para a implementação dos algoritmos relevantes, que se destinam a fazer parte do Sudoku na aplicação computacional do projeto LogicaMente.

Python é uma linguagem de programação de alto nível extremamente popular, principalmente devido à facilidade de aprendizado com sua sintaxe clara e simples de entender, e à sua versatilidade. Utilizado em desenvolvimento web, ciência de dados, inteligência artificial e em muitas outras aplicações do campo da computação, possui uma comunidade bastante ativa que contribui para o desenvolvimento e documentação da linguagem. A clareza da sintaxe dessa linguagem, somada à legibilidade de código, facilita o entendimento de algoritmos escritos nela.

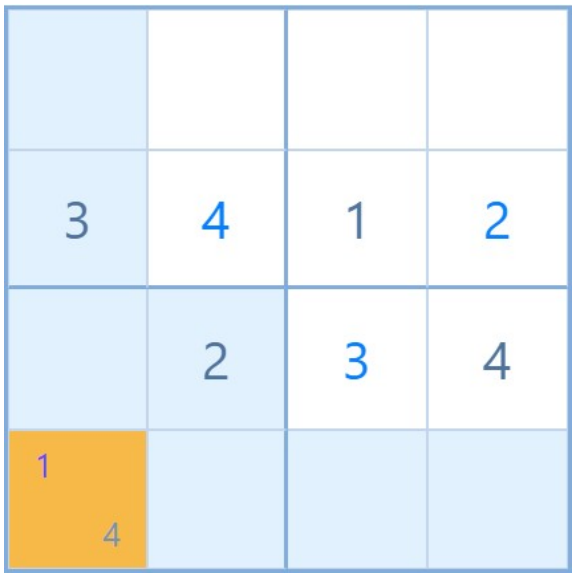
Tabelas Hash e busca em profundidade são conceitos relacionados aos algoritmos implementados, que serão explicados mais adiante.

No Sudoku, existem diversas técnicas utilizadas pelos jogadores para conseguir desvendar uma solução para um dado quebra-cabeça. Algumas delas são mais simples, outras mais complexas. Neste tópico será apresentado algumas dessas técnicas e seu funcionamento.

Antes de prosseguir, é importante definir que um "candidato"de uma célula, no contexto do Sudoku, se refere a um símbolo que poderia ser inserido numa célula, sem diretamente quebrar alguma das três regras principais do jogo, com base nos símbolos já presentes nas outras células. A Figura 2.8 ilustra um exemplo, onde a célula marcada em laranja pode, a princípio, receber os números 1 e 4, pois esses dois números não aparecem na mesma linha, coluna ou região da célula marcada.

- Candidato Simples: A primeira técnica, também chamada de “Simples Óbvios” (SUDOKU.COM, 2024), se baseia nas regras principais do Sudoku, de que não podem haver símbolos repetidos em uma mesma linha, coluna ou região da grade. Uma forma de enxergar o funcionamento da técnica é pela escrita de todos os candidatos possíveis para cada célula livre, analisando cada linha, coluna e região da célula em questão. Se, ao fazer isso, existe alguma célula que só possua um candidato possível, então insere-se aquele único candidato. Ou seja, a técnica consiste na observação direta de todos os números já inseridos em uma determinada linha, coluna e região de uma célula e, por eliminação, a inserção da única possibilidade, se houver, na célula escolhida. Essa técnica básica é uma das mais utilizadas

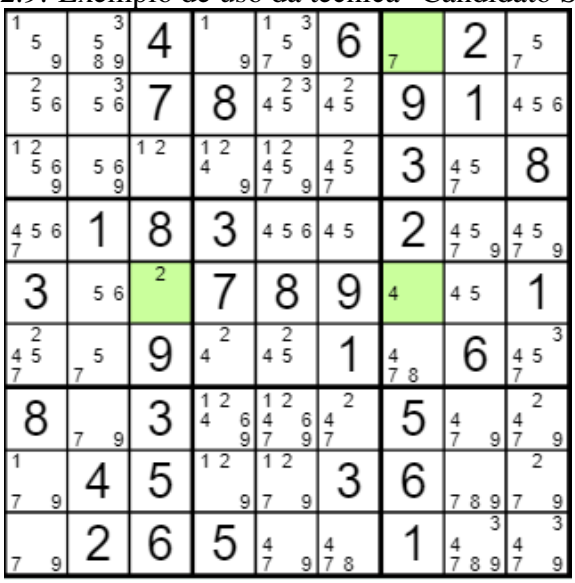
Figura 2.8: Candidatos em uma célula de Sudoku 4x4



Fonte: (ONLINE, 2024)

por jogadores, pois tem sua base na aplicação direta das regras principais do jogo, não necessitando de nenhum reconhecimento de algum padrão específico para seu avanço. A Figura 2.8 ilustra o uso dessa técnica, sendo as células com números menores representando todos os candidatos daquela célula.

Figura 2.9: Exemplo de uso da técnica "Candidato Simples".

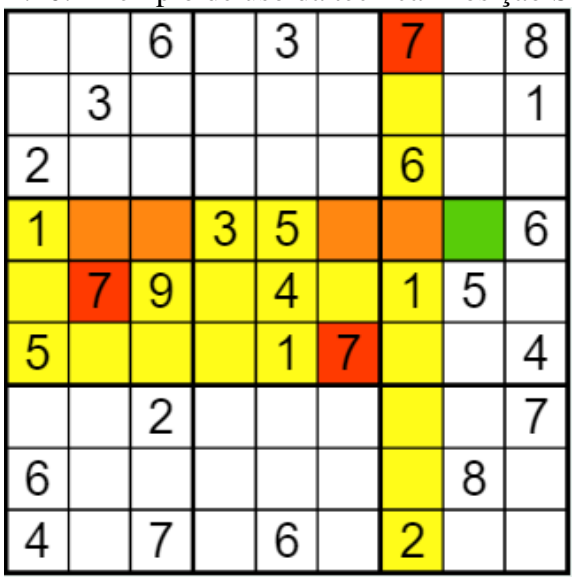


Fonte: (SUDOKUOFTHEDAY, 2024)

- Posição Simples: esta técnica também se baseia nas regras principais do Sudoku, e é análoga à técnica anterior. Para aplicação da “Posição Simples”, deve-se escolher uma linha, coluna ou região e observar quais símbolos já foram colocados nela. E então, é preciso analisar quais símbolos estão faltando, e identificar onde um símbolo poderia se encaixar nas células livres da linha, coluna ou região escolhida, sem

infringir as regras principais do Sudoku. Portanto, em outras palavras, deve-se testar cada símbolo faltante em cada célula livre da linha, coluna ou região escolhida, para saber se, ao inserir aquele símbolo, alguma das regras principais seria infringida. Dessa maneira, insere-se símbolos que encontrem essa condição. A técnica explicada também é bastante utilizadas pelos jogadores, sendo de fácil aplicação e exigindo somente uma aplicação direta das regras principais do jogo. A Figura 2.9 mostra possibilidades de se colocar o símbolo "7"na quarta linha da grade. Em laranja, as células livres da quarta linha onde não se poderia colocar um "7", devido à sua presença nas células em vermelho. A célula verde ilustra a posição correta de inserção do "7"após o uso da técnica, partindo do princípio que qualquer outra célula não preenchida naquela linha viola as regras principais.

Figura 2.10: Exemplo de uso da técnica "Posição Simples".



Fonte: (SUDOKUOFTHEDAY, 2024)

- Linhas Candidatas: esta técnica, diferentemente das anteriores, requer reconhecimento de padrões por parte do jogador. Outro diferencial dela é o fato de que sua aplicação não implica necessariamente em incluir diretamente um símbolo em uma célula livre. A aplicação de “Linhas Candidatas” permite a retirada de candidatos anteriormente possíveis, ajudando a designar células em que determinados símbolos não podem ser inseridos.

  A técnica é mais fácil de visualizar ao inserir todos os símbolos candidatos possíveis em cada célula livre. Ela consiste em identificar regiões da grade que possuam células livres adjacentes com símbolos candidatos em comum, e que esses candidatos estejam presentes somente nessas células adjacentes na região selecionada.

Ao identificar esse padrão, é possível deduzir que, obrigatoriamente, o símbolo candidato em comum encontrado precisa estar nessa região, em alguma das células adjacentes. Consequentemente, é possível retirar esse símbolo dos candidatos possíveis de outras células livres que estejam na mesma linha ou coluna das células adjacentes identificadas. Dessa maneira, a quantidade de candidatos possíveis em diversas células pode ser reduzida, possivelmente para um candidato possível, podendo ser inserido na célula livre.

A Figura 2.10 ilustra o uso dessa técnica. No exemplo, existem quatro células com o símbolo "4"como candidatos, todas na mesma coluna, e duas delas dentro de uma mesma região. Essa condição garante que o símbolo "4"deve estar na região inferior direita. Portanto, pode-se remover o símbolo "4"dos candidatos das células fora da região demarcada, nas células marcadas em verde forte. É importante notar que a aplicação da técnica reduz o número de candidatos de uma célula a 1, permitindo a inserção direta do candidato restante, enquanto na outra célula ainda sobram três candidatos.

Figura 2.11: Exemplo de uso da técnica "Linhas Candidatas".

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

Fonte: (SUDOKUOFTHEDAY, 2024)

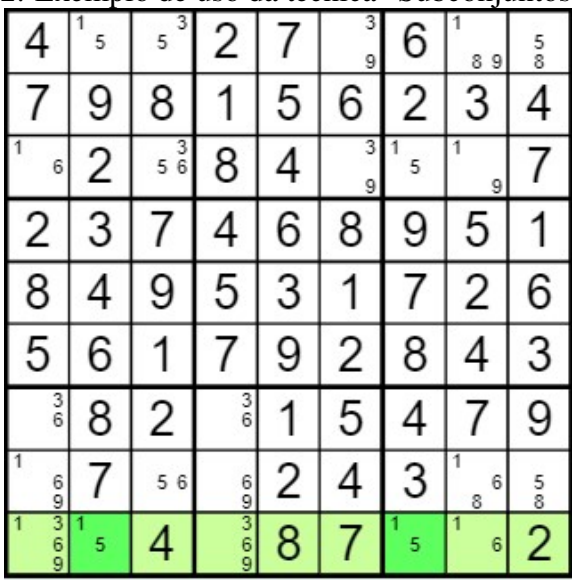
- Subconjuntos Disjuntos: esta técnica possui semelhanças com as Linhas Candidatas explicada anteriormente. Ela também envolve o reconhecimento de padrões para a retirada de alguns candidatos de algumas células.

  A técnica consiste em identificar um par de células que esteja em uma mesma linha, coluna ou região, que tenha os mesmos candidatos, e que tenha exatamente dois candidatos. Por essa identificação, é possível garantir que esses dois candidatos

devem estar nessas duas células, mas não é possível saber qual candidato seria o correto para cada célula do par. Consequentemente, é possível garantir que esses dois candidatos não estão em outras células da mesma linha, coluna ou região do par identificado, possibilitando a remoção dos dois candidatos de outras células. A Figura 2.12 ilustra o uso dessa técnica. No exemplo, as duas células marcadas em verde forte possuem o mesmo par de candidatos, garantindo que os símbolos "1"e "5"estão nessas duas células. Portanto, pode-se remover ambos os símbolos das outras células, marcadas em verde fraco.

Essa técnica também pode ser executada sobre triplas, e até quádruplas, e não somente pares. No caso de triplas, deve-se identificar três células, cada uma contendo exatamente os mesmos três candidatos. A lógica é análoga para quádruplas.

Figura 2.12: Exemplo de uso da técnica "Subconjuntos Disjuntos".



Fonte: (SUDOKUOFTHEDAY, 2024)

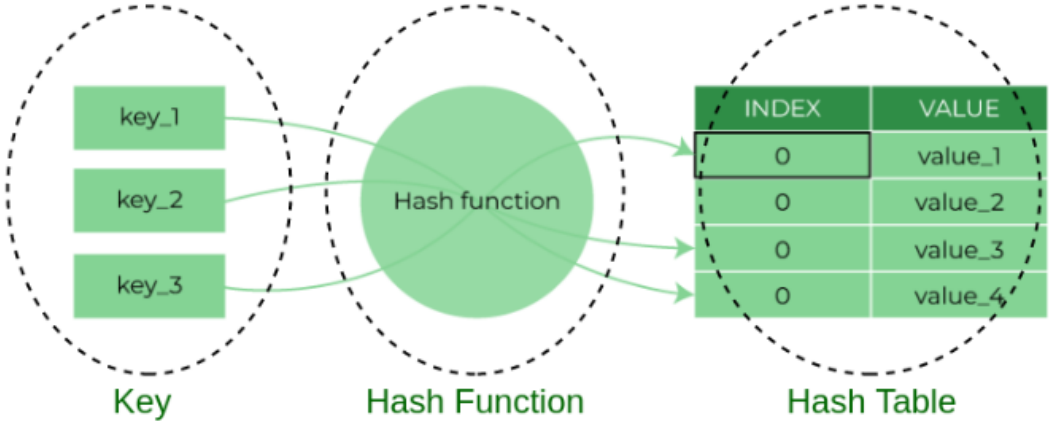
Existem muitas outras técnicas de diferentes complexidades e diversos usos, mas este trabalho irá se ater às explicadas anteriormente. Mais técnicas podem ser conferidas em (SUDOKUOFTHEDAY, 2024)

2.3 Tabelas Hash

As tabelas hash possuem um papel fundamental em muitas aplicações computacionais, a exemplo de bancos de dados, proporcionando eficiência e velocidade na busca, inserção e remoção de dados em estruturas de dados. Ela é uma escolha conveniente e muito popular devido à sua eficiência na manipulação de dados.

Uma tabela hash é uma estrutura de dados que utiliza uma função hash para mapear chaves para índices em uma tabela. A função hash transforma a chave em um valor hash, que é um valor numérico, o qual é utilizado como índice para armazenar ou recuperar informações, assim permitindo localizar diretamente o índice associado à chave, eliminando a necessidade de percorrer um grande volume de dados. A tabela hash permite a inserção de novos elementos de forma rápida, atribuindo-lhes um índice com base na função hash. Na remoção, o processo é similar à busca, bastando acessar diretamente o índice correspondente e retirar o elemento encontrado. A Figura 2.11 ilustra o processo de hashing realizado para formar essa estrutura.

Figura 2.13: Componentes do Hashing



Fonte: (GEEKS, 2024)

Dentro de tabelas hash, pode acontecer o que se chama de colisão. Uma colisão ocorre quando dados distintos têm um mapeamento para um mesmo valor hash. As colisões normalmente podem ser evitadas utilizando boas funções hash, ou seja, que possuam uma grande chance de gerarem valores únicos.

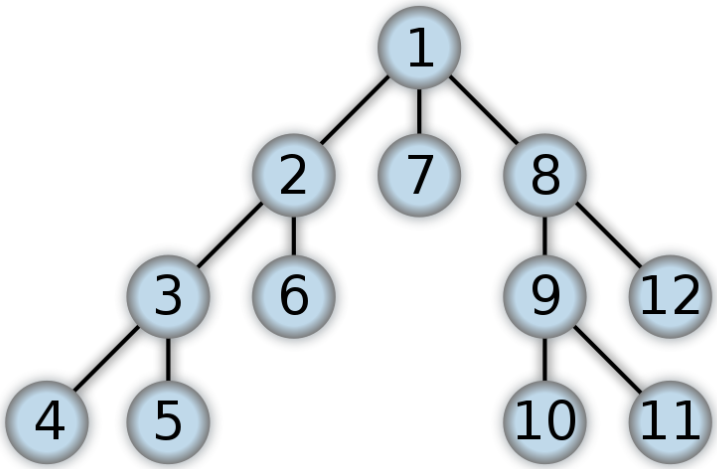
Mais detalhes sobre essa estrutura podem ser encontrados em (DCMUSP, 2024)

2.4 Busca em Profundidade

A busca em profundidade, também conhecido como "Depth-first Search"em inglês, é um algoritmo muito utilizado no campo da ciência da computação para percorrer estruturas de dados, a exemplo de árvores ou grafos. O funcionamento da busca em profundidade se baseia em explorar o máximo possível em uma ramificação, para posteriormente realizar o que se chama de “backtracking”: retroceder em busca de novas

opções, caso não se encontre a solução desejada. Esse controle pode ser realizado de várias maneiras, sendo frequentemente utilizada alguma estrutura de dados auxiliar para essa finalidade, como uma pilha, somada ao uso da recursão. A Figura 2.14 ilustra a aplicação desse algoritmo em uma estrutura de árvore.

Figura 2.14: Execução da busca em profundidade. Os valores dos nodos denotam a ordem em que os nodos são visitados



Fonte: (WIKIPEDIA-DFS, 2024)

O algoritmo da busca em profundidade possui inúmeras aplicações, permitindo uma busca eficiente por soluções em diversos problemas. Um exemplo são problemas relacionados a grafos, como o Problema do Menor Caminho (BLACK, 2020), que consiste em encontrar a menor distância entre dois vértices de um grafo. Outro exemplo de aplicação são alguns jogos, como o xadrez, em que programas de computador precisam do algoritmo para explorar as possibilidades de jogadas e determinar a melhor escolha para a situação atual da partida.

Mais detalhes sobre esse algoritmo podem ser encontrados em (IMEUSP, 2024)

3 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns trabalhos relativos ao tópico abordado neste estudo.

3.1 Problemas matemáticos e computacionais relacionados ao Sudoku

O Sudoku possui diversos problemas associados que são estudados por matemáticos da academia científica, desde sua criação e popularização. Alguns problemas matemáticos incluem, por exemplo, o problema da enumeração de todos os Sudokus possíveis, ou seja, contar de quantas maneiras diferentes é possível preencher uma grade com uma solução única válida (FELGENHAUER; JARVIS, 2005). O artigo conclui que existem 6.671×10^{21} possibilidades, número verificado no referido artigo e em diversos outros. Outro problema matemático é o Menor Número de Pistas: encontrar o menor número de pistas que pode existir em uma grade inicial, de forma que ainda garanta solução única para o Sudoku. O artigo (MCGUIRE; TUGEMANN; CIVARIO, 2014) encontra e valida uma resposta que já era apontada frequentemente por diversos trabalhos antes: não existem grades do Sudoku clássico com menos de 17 pistas. Muitos problemas envolvendo o Sudoku são pontuados e discutidos no artigo (DELAHAYE, 2006), inclusive os mencionados anteriormente.

3.2 Algoritmos de resolução do Sudoku

No campo da ciência da computação, também existem problemas vinculados ao Sudoku. O mais clássico é o problema do Sudoku generalizado, um problema conhecidamente NP-completo (HOEXUM, 2020), que consiste em um algoritmo capaz de encontrar a solução de qualquer Sudoku válido, de tamanho N²xN² com regiões NxN. Nesse contexto, o Sudoku pode ser vinculado a outros problemas da computação para ser resolvido, como por exemplo o problema da Coloração de Grafos (JENSEN; TOFT, 2011), cuja definição é pintar vértices de um grafo de forma que nenhum vértice tenha a mesma cor de outro vértice adjacente.

No acervo de trabalhos e pesquisas relacionadas ao Sudoku, é comum encontrar a aplicação de diversos algoritmos para solucionar qualquer instância do quebra-cabeça, normalmente atreladas ao Sudoku clássico, mas também para Sudokus de outros tama-

nhos. Pode-se mencionar (MANTERE; KOLJONEN, 2007), cujo objetivo é usar algoritmos genéticos para gerar instâncias do Sudoku clássico válidas, encontrar sua solução única e classificar a grade gerada em distintos níveis de dificuldade, comparando essa classificação com a percepção humana de dificuldade de resolução. Outro autor explora o uso de um algoritmo de busca harmônica para resolver instâncias do Sudoku clássico (GEEM, 2007). Por fim, também existem trabalhos que buscam resolver o Sudoku usando modelos de inteligência artificial (CAINE; COHEN, 2006).

3.3 Classificação de níveis de dificuldade

Ainda é um desafio nos dias de hoje fazer uma classificação precisa de níveis de dificuldade de resolução de um Sudoku para um usuário. O nível de dificuldade para encontrar a solução única do quebra-cabeça tende a ser uma percepção individual de cada pessoa, o que dificulta uma generalização coerente, dependendo mais do desenvolvedor para determinar as fronteiras entre cada nível de dificuldade. Diversos artigos, como por exemplo (MORAGLIO; TOGELIUS; LUCAS, 2006), apontam que o número de pistas na grade inicial do Sudoku influencia pouco ou nada no nível de dificuldade do quebra-cabeça, necessitando de outras abordagens.

Existem muitos artigos que propõem diferentes modelos e formas de avaliar o grau de dificuldade de uma dada grade do Sudoku. O artigo (PELANEK, 2011) discute sobre esse tópico e apresenta um modelo computacional baseado em ações humanas para efetuar essa classificação, enquanto o artigo (PELANEK, 2014), do mesmo autor, faz uma avaliação de diferentes tipos de modelos já propostos, concluindo, dentre outras ideias, que os pilares da definição do nível de dificuldade de um Sudoku é a complexidade de passos individuais e a dependência entre os passos realizados.

3.4 Geração de Sudokus válidos

Gerar instâncias iniciais de Sudokus que possuam solução única é especialmente relevante para jogadores do Sudoku que querem resolvê-las por conta própria. Existem diversas maneiras de fazer a geração desses puzzles, mas, pelos trabalhos consultados, a geração desses puzzles é constantemente validada por algoritmos de resolução do Sudoku. O artigo (MANTERE; KOLJONEN, 2007) aborda esse tema, e gera instâncias iniciais a

partir de um Sudoku já resolvido com o algoritmo proposto, selecionando um número específico de pistas e tentando repetidamente resolvê-lo novamente para validar se o novo Sudoku gerado possui solução única. É comum a geração de novos Sudokus a partir de outros pré-existentes.

4 PROPOSTA E IMPLEMENTAÇÃO

Este capítulo aborda a implementação de diversos algoritmos relativos ao Sudoku direcionados para a aplicação computacional do LogicaMente. Discutiremos sobre o algoritmo implementado de resolução do Sudoku, um algoritmo para a classificação em diferentes níveis de dificuldade de um dado Sudoku, e a geração de novas instâncias de Sudoku válidas não solucionadas. Esses algoritmos foram implementados na linguagem de programação Python 3.8.

4.1 Pré-processamento e estruturas de dados

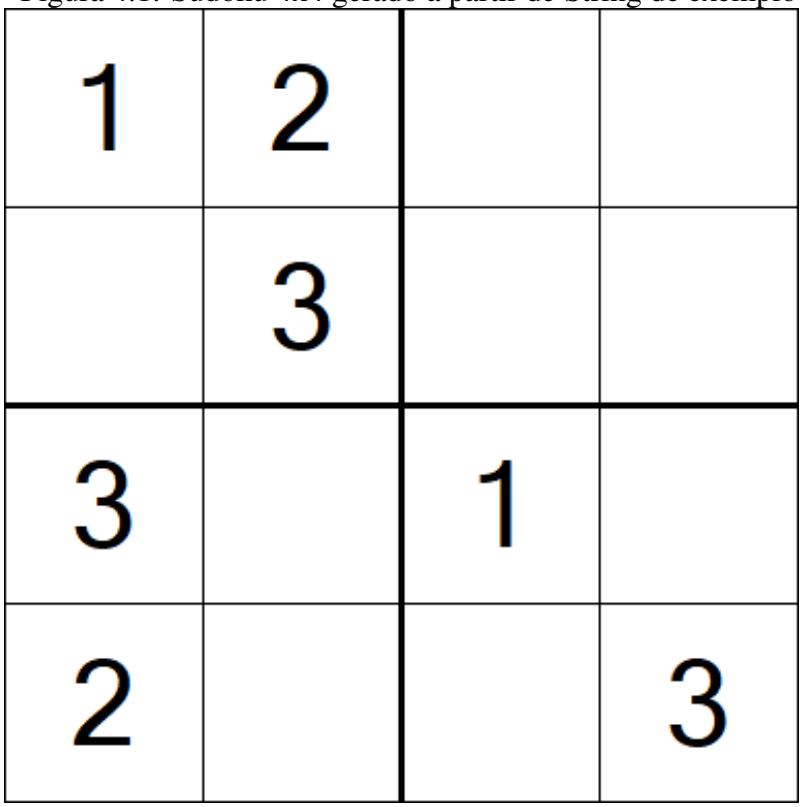
Antes da aplicação dos algoritmos, é importante trazer algumas noções da estrutura da implementação. A estrutura construída foi baseada no trabalho do cientista da computação Peter Norvig (NORVIG, 2006), disponível na web, que detalha a implementação de um algoritmo para solucionar o Sudoku clássico. Uma instância do Sudoku é informada como uma string, que em Python é uma sequência ordenada de caracteres. Cada caractere representa o conteúdo de uma célula, linha a linha, podendo ser um símbolo presente na grade, ou o símbolo “0” caso aquela célula não esteja preenchida. Por exemplo, a sequência de caracteres "1200030030102003"remete ao Sudoku 4x4 ilustrado na Figura 4.1.

Inicialmente, um processamento é feito para converter essa string em uma tabela hash(comumente chamada de “dicionário” na linguagem Python), em que cada chave é uma string que contém dois caracteres, sendo uma letra e um dígito, representando a posição de uma célula na grade, e cada valor é um caractere que representa o conteúdo daquela célula, coletado através da string inicial explicada no parágrafo anterior. A tabela hash construída, portanto, é uma outra representação da grade Sudoku.

Após esse processamento, é realizada a criação de uma grade de candidatos. Ela é uma representação da grade que possui todos os valores possíveis para cada célula. Dessa forma, ela tem a mesma estrutura da tabela hash explicada anteriormente, exceto que, na situação em que o valor da célula ainda não foi preenchido, em vez do símbolo “0”, o valor passa a ser uma string com todos os símbolos possíveis de serem preenchidos naquela célula, ou seja, símbolos que não se repetem na mesma linha, coluna ou região daquela célula.

Uma lista auxiliar é construída contendo todas as linhas, colunas e regiões do

Figura 4.1: Sudoku 4x4 gerado a partir de String de exemplo



Fonte: (SUDOKU-PUZZLES, 2024)

Sudoku informado, de acordo com seu tamanho, que pode ser variável. Uma tabela hash também é construída para mapear cada célula à respectiva linha, coluna e região. Os algoritmos a seguir trabalham somente com Sudokus regulares.

4.2 Algoritmo para resolver o Sudoku

Para a aplicação computacional desejada, um algoritmo capaz de resolver um Sudoku é necessário para, dentre outros objetivos, mostrar a resposta ao usuário, e até mesmo ilustrar um passo-a-passo. Entretanto, o algoritmo de resolução do Sudoku também se mostra como uma ferramenta importante para os outros algoritmos que serão apresentados nas seções 4.3 e 4.4, realizando validações relevantes das instâncias do quebra-cabeça.

Os algoritmos implementados são baseados em diferentes técnicas de resolução manuais, comumente utilizadas por seres humanos, e que foram detalhadas na seção 2.3. A escolha da implementação dessas técnicas facilita uma implementação da classificação de nível de dificuldade dos Sudokus.

O primeiro algoritmo implementado é a técnica “Candidato Simples”. Ele recebe uma célula da grade, e para essa célula, verifica quais símbolos estão presentes na li-

nha, coluna e região da célula informada. Após essa verificação, caso esteja sobrando exatamente um símbolo ainda não inserido, insere esse símbolo nela.

O segundo algoritmo implementado é a técnica “Posição Simples”. O algoritmo recebe uma linha, coluna ou região, e faz uma varredura por ela para selecionar símbolos que ainda não foram inseridos nela. Com os símbolos possíveis armazenados, o algoritmo prossegue para avaliar onde cada um desses símbolos poderia ser inserido nas células livres disponíveis na linha, coluna ou região informada, verificando se seria possível inserir um desses símbolos, ou seja, se ao inserir esse símbolo na célula as 3 regras principais do Sudoku não seriam infringidas.

O terceiro algoritmo implementa a técnica “Linhas Candidatas”. Conforme foi explicado na seção 2.3, esta é uma técnica dedutiva que reduz a quantidade de candidatos possíveis para uma célula, não necessariamente permitindo a inserção garantida de um novo elemento em uma célula. Portanto, o algoritmo atua sobre a estrutura que armazena os candidatos possíveis de cada célula, reduzindo a sua quantidade em algumas células e permitindo a reaplicação das duas técnicas anteriores. O algoritmo procura por regiões que contenham células adjacentes com algum símbolo candidato em comum, e que esse candidato só exista nessas células adjacentes na região selecionada. Ao encontrar essa situação, pode-se deduzir que esse candidato em comum não pode estar presente em outra célula da linha ou coluna dessas células adjacentes, e dessa forma, elimina a possibilidade desse candidato comum em todas essas outras células.

O quarto algoritmo implementa a técnica "Subconjuntos Disjuntos", para pares e triplas. É realizada uma varredura por toda a grade, procurando por pares de células com dois candidatos iguais em uma linha, coluna ou região. O mesmo processo é realizado para triplas. Essa implementação reduz a quantidade de candidatos possíveis para outras células, permitindo a inserção de novos símbolos caso sobre um candidato, e possibilitando a repetição da técnica de "Linhas Candidatas"com diferentes grupos de candidatos.

O algoritmo executa todas essas quatro técnicas em todas as células, linhas, colunas e regiões da grade em uma iteração, de acordo com o parâmetro informado. Entretanto, para evitar que o algoritmo ignore células anteriormente visitadas que possam ter causado algum progresso na resolução, se uma técnica executada causa uma modificação no estado da grade, o algoritmo reexecuta as técnicas novamente, até que o Sudoku esteja resolvido ou até que nenhuma técnica tenha sido capaz de ocasionar progresso.

Além dessas técnicas implementadas, é utilizado também o algoritmo anteriormente mencionado neste trabalho, proposto por Peter Norvig. (NORVIG, 2006) O tra-

balho desse autor implementa uma forma de resolver Sudokus clássicos, com diversas representações de estado do Sudoku, por meio de uma busca em profundidade para testar possibilidades de valores em diferentes células, e propagação de restrições para eliminar candidatos de células ainda não preenchidas da grade. O algoritmo do autor verifica constantemente se uma solução ou uma contradição foi encontrada, e em caso negativo, testa os candidatos possíveis para cada célula livre e continua com a busca a partir da célula que se está testando, resultando em uma busca recursiva por todos os candidatos a uma célula. A busca utiliza uma heurística para priorizar a procura por células com uma menor quantidade de candidatos, aumentando a probabilidade de se acertar a tentativa de inserção de um valor na célula. Esse algoritmo é capaz de resolver qualquer instância do Sudoku clássico, segundo experimentos realizados e ilustrados na própria fonte.

4.3 Algoritmo para classificação de níveis de dificuldade do Sudoku

Os algoritmos explicados na seção anterior possuem um papel essencial para a classificação de níveis de dificuldade proposta neste trabalho. Eles são baseados em técnicas manuais executadas por seres humanos para resolver Sudokus, e pode-se definir um grau de complexidade dessas técnicas com base no que é necessário para executá-las.

A primeira técnica, “Candidato Simples”, exige somente a identificação de um símbolo faltante na linha, coluna ou região, seguindo as três regras principais do Sudoku. Analogamente, a segunda técnica, “Posição Simples”, exige a identificação de uma célula em que um símbolo poderia ser inserido, também seguindo as mesmas três regras. Dessa forma, pode-se dizer que ambas as técnicas são simples de serem executadas, não sendo necessário o uso de alguma outra habilidade, somente o conhecimento das regras do jogo. Considerando essa argumentação, um Sudoku que possa ser resolvido utilizando somente uma dessas técnicas ou ambas é classificado como um Sudoku de nível “Fácil”.

A terceira técnica, “Linhas Candidatas”, é uma técnica que requer o reconhecimento de padrões, um dos pilares do pensamento computacional, para encontrar uma condição específica no quebra-cabeça, e a partir dela deduzir a eliminação de possíveis candidatos. Dessa maneira, caso um Sudoku não possa ser resolvido somente com o uso das duas primeiras técnicas, e necessite do uso da terceira técnica, o Sudoku é classificado como nível “Médio”.

Caso nenhuma dessas três técnicas sejam suficientes para solucionar o Sudoku, será necessário utilizar o algoritmo de busca em profundidade e propagação de restrições

para resolvê-lo, e nesse caso, o Sudoku é classificado como “Difícil”.

O Algoritmo 1 mostra um pseudocódigo do algoritmo de resolução do Sudoku, juntamente ao algoritmo de classificação de nível de dificuldade. Nesse algoritmo, as duas técnicas mais simples para resolver o Sudoku são utilizadas em todas as células do Sudoku, e caso tenha havido algum progresso na resolução dele, repete a execução das mesmas técnicas. No caso de não se ter tido progresso, o algoritmo tenta utilizar as duas técnicas dedutivas para reduzir os candidatos do Sudoku atual, e caso tenha conseguido algum progresso, passa a executar todas as quatro técnicas novamente. Em uma situação em que nenhuma das técnicas consiga causar algum progresso na resolução, usa-se o algoritmo de propagação de restrições. Dessa forma, o algoritmo retorna o Sudoku resolvido e o nível de dificuldade baseado nos algoritmos que foram necessários para resolvê-lo.

Algorithm 1 Algoritmo que resolve e classifica o nível de dificuldade de um Sudoku

```
1:  procedure SolveAndRate(grid)
2:      advanced ← False
3:      while advanced = True do
4:          do CandidatoSimples(grid)
5:          do PosiçãoSimples(grid)
6:          if grid not changed then
7:              usedDedutiveTechs ← True
8:              do LinhasCandidatas(grid)
9:              do SubconjuntosDisjuntos(grid)
10:         end if
11:         if grid changed then
12:             advanced ← True
13:         end if
14:     end while
15:     if notSolved(grid) then
16:         do PropagacaoRestricoes(grid)
17:         return grid, HARD
18:     else if solved(grid) then
19:         if usedDedutiveTechs then
20:             return grid, MEDIUM
21:         else
22:             return grid, EASY
23:         end if
24:     end if
25: end procedure
```

4.4 Geração de novos Sudokus

Existem diversos modos de gerar novos Sudokus não solucionados, mas um método bastante comum, explorado inclusive no artigo (MANTERE; KOLJONEN, 2007), envolve gerar instâncias do Sudoku não solucionadas a partir de uma instância do Sudoku já solucionada. Utilizar um algoritmo que aplique esse método requer a existência de uma base de dados com Sudokus já resolvidos para alimentar o algoritmo.

Para a implementação da geração de Sudokus, foi necessário também o uso de um algoritmo capaz de verificar se um dado Sudoku possui solução única. Esse algoritmo tenta recursivamente preencher todas as células desocupadas com todas as possibilidades de símbolos, sem violar as regras principais do Sudoku, removendo o elemento da célula após a busca para a exploração de novas possibilidades de solução. Caso encontre mais de uma solução dessa forma, o algoritmo interrompe sua execução, pois o objetivo não é encontrar todas as soluções, e sim descobrir se há mais de uma.

O Algoritmo 2 mostra um pseudocódigo do algoritmo de geração de Sudokus. O algoritmo retira uma dada quantidade de valores aleatórios de uma instância já solucionada do Sudoku. Para cada valor retirado do Sudoku dessa maneira, é utilizado o algoritmo de resolução do Sudoku proposto neste trabalho para realizar a validação do novo Sudoku gerado a partir da retirada de um valor. Então, é executado o algoritmo que verifica se o Sudoku informado possui solução única. Se o Sudoku não puder ser resolvido após a retirada desse valor, ou se o Sudoku resultante possui nenhuma ou mais de uma solução, o algoritmo reinsere o valor retirado de volta ao Sudoku, e tenta retirar outro valor aleatório. Esse processo se repete até que se retire a quantidade de valores passada como argumento ao algoritmo, ou até que se tente retirar todos os valores disponíveis sem sucesso, retornando o Sudoku com o máximo de valores que se conseguiu remover.

O algoritmo explicado permite a geração de vários Sudokus não solucionados diferentes a partir de um único Sudoku já resolvido.

Algorithm 2 Algoritmo que gera novos Sudokus

```
1:  procedure GENERATE(grid, cluesRemoved)
2:      advanced ← True
3:      while cluesRemoved > 0 do
4:          randomSquare ← getRandomFilledNotVisitedSquare()
5:          removeElement(grid, randomSquare)
6:          if isUniqueSolution(grid) then
7:              cluesRemoved = cluesRemoved − 1
8:          else if isNotUniqueSolution(grid) then
9:              assignElement(grid, randomSquare)
10:         end if
11:         if notVisitedSquares.size() = 0 then
12:             break
13:         end if
14:     end while
15: end procedure
```

5 RESULTADOS E VALIDAÇÃO DA IMPLEMENTAÇÃO

Neste capítulo, serão realizados alguns experimentos para validar os algoritmos propostos no capítulo anterior. Serão gerados novos Sudokus regulares de ordem 4, 6 e 9, e estes serão resolvidos e classificados conforme seu nível de dificuldade pelo critério proposto.

5.1 Sudokus de ordem 4

Conforme mencionado na seção 4.3, o algoritmo de geração de novos Sudokus depende de um Sudoku previamente solucionado. Existem diversos datasets de Sudokus solucionados espalhados na web, e alguns foram escolhidos para serem utilizados nos experimentos. O dataset (ARCOT; KALLURAYA, 2019) contém 288 Sudokus regulares de ordem 4 solucionados, e foi o primeiro a ser utilizado para testar os algoritmos propostos.

Para esse experimento, foram selecionados aleatoriamente 50 Sudokus solucionados distintos do dataset. A partir de cada um deles, foram gerados 4 novos Sudokus, com números de pistas distintos em cada um. Esses 4 novos Sudokus tiveram os seguintes números de pistas: o primeiro Sudoku é gerado com 12 pistas; o segundo Sudoku é gerado com 10 pistas; o terceiro Sudoku é gerado com 7 pistas; por fim, o último Sudoku foi gerado com o menor número de pistas que o algoritmo conseguisse. É importante lembrar que, conforme explicado anteriormente, caso o algoritmo não consiga mais remover valores da grade sem tornar o Sudoku inválido, será gerado o Sudoku com o máximo de pistas que conseguiu remover. Dessa forma, o algoritmo foi capaz de gerar 200 novos Sudokus válidos diferentes dos originais. A tabela 5.1 mostra o número de pistas em cada Sudoku gerado.

Tabela 5.1: Quantidade de pistas por Sudoku: ordem 4

Pístas	*Quantidade de Sudokus*
12	50
10	50
7	50
6	1
5	18
4	31

Fonte: O Autor

Após a geração desses Sudokus, usou-se o algoritmo de resolução implementado para resolver os 200 Sudokus gerados, onde todos os Sudokus de ordem 4 gerados foram

resolvidos corretamente, em um tempo de execução de, no máximo, 0.001 segundo cada puzzle. Os Sudokus tiveram seu nível de dificuldade classificado como "Fácil". Essa classificação ocorre porque, dentro de meu conhecimento, todo Sudoku regular de ordem 4, sem regras adicionais ou outras restrições, pode ser resolvido com inserção direta de símbolos por eliminação, ou seja, as técnicas de "Posição Simples"e "Candidato Simples"são suficientes. Não é necessário nenhuma técnica dedutiva ou propagação de restrições que testa diversos símbolos para resolver um Sudoku dessa ordem, pois nunca haverão células com múltiplos candidatos a serem inseridos. Portanto, pelo critério utilizado de classificação de nível de dificuldade, Sudokus regulares dessa ordem serão sempre de nível "Fácil".

Da mesma maneira, o algoritmo de resolução do Sudoku foi capaz de resolver todos os 288 Sudokus de ordem 4 originais do dataset, classificando todos também como nível de dificuldade "Fácil".

5.2 Sudokus de ordem 6

Para a experimentação de Sudokus de ordem 6 não foram encontrados datasets disponíveis na web, mas muitos sites oferecem Sudokus para serem resolvidos, com solução disponível. Portanto, foram coletados 10 Sudokus de ordem 6 para este experimento, a partir do site (SUDOKU-PUZZLES, 2024).

O experimento foi realizado de forma similar ao experimento anterior. Entretanto, dessa vez foram gerados cinco Sudokus a partir de cada um dos Sudokus disponíveis, gerando, no total, 50 novos Sudokus. O primeiro dos cinco, contendo 30 pistas; o segundo, contendo 20; o terceiro, 12 pistas; o quarto, tentando inserir 10 pistas; e o quinto, tentando o menor número de pistas possível. A tabela 5.2 ilustra o número de pistas por Sudoku.

Tabela 5.2: Quantidade de pistas por Sudoku: ordem 6

Pístas	*Quantidade de Sudokus*
30	10
20	10
12	10
11	7
10	12
9	1

Fonte: O Autor

Os Sudokus gerados tiveram sua classificação de nível de dificuldade definida conforme ilustrado na tabela 5.3. É possível perceber, por ela, que somente dois Sudokus não foram classificados como "Fácil". Segundo o critério utilizado, puzzles médios ou

difíceis são raros para Sudokus dessa ordem.

Tabela 5.3: Nível de dificuldade por Sudoku gerado: ordem 6

Dificuldade	*Quantidade de Sudokus*
Fácil	48
Médio	1
Difícil	1

Fonte: O Autor

Posteriormente, os 10 Sudokus originais foram resolvidos pelo algoritmo de resolução proposto. Apenas um foi classificado como "Difícil", enquanto todos os outros foram classificados como "Fácil".

5.3 Sudokus de ordem 9

Para Sudokus de ordem 9, foi usado um dataset que contém 3 milhões de Sudokus clássicos resolvidos, encontrado em (RACLIFFE, 2021). Foi realizada uma seleção aleatória de 50 Sudokus dele, e a partir de cada um, gerou-se quatro novos Sudokus. O primeiro Sudoku desses quatro foi gerado com 60 pistas; o segundo Sudoku foi gerado com 30 pistas; o terceiro Sudoku com 23 pistas; e o quarto, novamente, com o menor número possível de pistas. A tabela 5.4 mostra a quantidade de pistas por Sudoku gerado.

Tabela 5.4: Quantidade de pistas por Sudoku: ordem 9

Pístas	*Quantidade de Sudokus*
60	50
30	50
28	1
27	1
26	9
25	36
24	38
23	13
22	2

Fonte: O Autor

Todos os 200 Sudokus clássicos gerados foram resolvidos pelo algoritmo de resolução proposto. Cada um deles foi classificado em um nível de dificuldade, conforme ilustrado na Tabela 5.5.

O dataset utilizado, além de possuir o Sudoku não resolvido e a solução, também possui uma coluna com um valor numérico não inteiro que representa a dificuldade do Sudoku, em que a magnitude do valor e a dificuldade do Sudoku são diretamente proporcionais. Segundo a descrição do dataset presente na fonte, esse valor foi calculado

Tabela 5.5: Nível de dificuldade dos Sudokus gerados: ordem 9

Dificuldade	*Quantidade de Sudokus*
Fácil	133
Médio	14
Difícil	53

Fonte: O Autor

com base na profundidade média da árvore de busca em 10 tentativas. Ainda segundo a descrição, no conjunto de dados, 43% dos Sudokus possuem dificuldade 0.0, enquanto a dificuldade mais alta é dada pelo valor 8.5, e a maior parte da distribuição de valores de dificuldade está entre 0 e 3. Essas informações foram utilizadas para fins de comparação com o nível de dificuldade identificado pelo algoritmo proposto, realizando uma seleção aleatória de 200 Sudokus do dataset original para o algoritmo de resolução, com posterior classificação do seu nível de dificuldade.

Após cada um dos 200 Sudokus ser resolvido e seu nível de dificuldade definido pelos algoritmos propostos, o valor numérico da dificuldade presente no dataset foi coletado para todos os Sudokus classificados como nível "Fácil"pelo algoritmo proposto para se calcular a média aritmética de Sudokus desse nível. O mesmo procedimento foi realizado para os outros dois níveis de dificuldade propostos. Os resultados dessa comparação estão na tabela 5.6.

Tabela 5.6: Comparação de níveis de dificuldade

Quantidade de Sudokus	*Dificuldade: algoritmo*	*Dificuldade média: dataset*
77	Fácil	0.0
17	Médio	1.61
106	Difícil	2.25

Fonte: O Autor

No procedimento realizado, todos os Sudokus com dificuldade 0.0 encontrados foram classificados como "Fácil"pelo algoritmo proposto. Sudokus com dificuldade acima de zero foram classificados como "Médio"ou "Difícil", com diversas variações nos valores encontrados. Apesar de valores mais altos(3 ou acima) terem sido classificados como "Difícil", houveram muitas classificações "Médio"e "Difícil"distribuídas entre valores de 1 a 2.5.

6 CONCLUSÃO E TRABALHOS FUTUROS

O Sudoku é um jogo desplugado e com regras de simples entendimento, sendo capaz de explorar o raciocínio lógico, reconhecimento de padrões e resolução de problemas. Isso permite que esse jogo seja utilizado como uma ferramenta pedagógica para o desenvolvimento do pensamento computacional, que é uma habilidade cada vez mais essencial na sociedade moderna.

Sudokus foram amplamente estudados em diversas áreas, com muitas possibilidades de algoritmos para serem usados para gerar e resolver o quebra-cabeça. A geração de Sudokus regulares proposta permite a criação de vários novos quebra-cabeças únicos a partir de um Sudoku já solucionado, de todos os níveis de dificuldade propostos, com número de pistas variável conforme o interesse e as necessidades do usuário, e garantindo que o quebra-cabeça chegue em um ponto onde não é possível remover mais pistas sem retirar a unicidade da solução.

A classificação em níveis de dificuldade de um Sudoku, apesar de ser algo subjetivo e comumente sujeito à definição do desenvolvedor, é um tópico que possui muitas abordagens em diversos trabalhos realizados. Este trabalho propôs o uso de técnicas variadas utilizadas manualmente por jogadores de Sudoku como uma forma não apenas de resolver, mas também de classificar a dificuldade de um Sudoku regular, assim considerando a dificuldade em cada passo de uma sequência de passos realizada para resolver o quebra-cabeça.

É importante pontuar algumas limitações da proposta apresentada. O algoritmo de geração de Sudokus não verifica se os quebra-cabeças gerados são somente diferenciados por alguma transposição, permitindo que dois Sudokus com as mesmas relações entre as células e os símbolos, mas posições diferentes, sejam produzidos. Dessa forma, o algoritmo pode gerar dois Sudokus equivalentes como dois quebra-cabeças diferentes. Além disso, a proposta de classificação de níveis de dificuldade contempla somente três categorias de dificuldade, mas seria interessante criar mais categorias para melhor distribuir a dificuldade, que pode não se resumir a três categorias apenas. Uma outra limitação é que os algoritmos apresentados se restringem a Sudokus regulares, mas existem Sudokus irregulares e muitas outras variantes, algumas mostradas neste trabalho, que não são consideradas nesses algoritmos. Ademais, com os algoritmos implementados, é necessário que eles sejam incorporados em uma aplicação computacional construída para os fins justificados previamente, permitindo a visualização e interação dos Sudokus com o usuário

por intermédio de uma interface gráfica. O código está disponível e público no GitHub: https://github.com/Thiago-inf/SudokuAlgorithms

Em resumo, os trabalhos futuros devem ter enfoque em aprimorar e expandir as abordagens realizadas, levando em consideração as adversidades reconhecidas, com o objetivo de conferir ao usuário um jogo que possa ser utilizado como instrumento pedagógico para o aprendizado do pensamento computacional.

REFERÊNCIAS

ARCOT, V.; KALLURAYA, S. **3 million Sudoku puzzles with ratings**. 2019. Available from Internet: <https://github.com/Black-Phoenix/4x4-Sudoku-Dataset/blob/master/4x4_sudoku_unique_puzzles.csv>.

BALL, W. R. **Mathematical Recreations and Essays**. [S.l.: s.n.], 1905.

BLACK, P. E. **Shortest Path**. 2020. Acessado em 2024. Available from Internet: <https://www.nist.gov/dads/HTML/shortestpath.html>.

CAINE, A.; COHEN, R. Mits: A mixed-initiative intelligent tutoring system for sudoku. **Lecture Notes in Computer Science**, 2006.

COMMONS, W. **Sudoku Diagonal**. 2024. Available from Internet: <https://commons.wikimedia.org/wiki/File:Diagonal-Sudoku-by-Skratt.svg>.

DCMUSP. **Dept. de Matematica e Computacao**. 2024. Available from Internet: <https://dcm.ffclrp.usp.br/~augusto/teaching/icii/Hash-Tables-Apresentacao.pdf>.

DELAHAYE, J.-P. The science behind sudoku. **Scientific American**, 2006.

ENSINO. **Quadrado Latino**. 2024. Available from Internet: <https://www.ensinoeinformacao.com/estatistica-quadrado-latino-introducao#:~:text=Um%20quadrado%20latino%20de%20ordem,utilizou%20caracteres%20latinos%20como%20s%C3%ADmbolos.>

FEATURES, K. **Sudokus Gêmeos**. 2024. Available from Internet: <https://www.knightfeatures.com/syndication-updatessudoku/2018/1/11-ba49f-2lhcp-8le7n>.

FELGENHAUER, B.; JARVIS, F. Enumerating possible sudoku grids. 2005.

GEEKS, G. F. **Componentes Hash**. 2024. Available from Internet: <https://www.geeksforgeeks.org/implementation-of-hash-table-in-python-using-separate-chaining/>.

GEEM, Z. W. Harmony search algorithm for solving sudoku. **Lecture Notes in Computer Science**, 2007.

GONZAGA, B. Promovendo competências com diversão: Puzzles e o desenvolvimento do pensamento computacional no ensino fundamental i. **UFRGS**, 2023.

HOEXUM, E. Revisiting the proof of the complexity of the sudoku puzzle. 2020.

IMEUSP. **DFS IME USP**. 2024. Available from Internet: <https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dfs.html>.

JENSEN, T. R.; TOFT, B. **Graph coloring problems**. [S.l.]: John Wiley & Sons, 2011.

KAHOOT. **Plataforma de aprendizado gamificado**. 2013. Acessado pela última vez 16 de Dezembro de 2023. Available from Internet: <https://kahoot.com/>.

MANTERE, T.; KOLJONEN, J. Solving, rating and generating sudoku puzzles with ga. **2007 IEEE International Conference on Evolutionary Computation**, 2007.

MCGUIRE, G.; TUGEMANN, B.; CIVARIO, G. There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem. **Experimental Mathematics**, 2014.

MEDIA, C. **Sudoku Irregular**. 2024. Available from Internet: <https://www.clarity-media.com/onlinepuzzles/about-jigsaw-sudoku-puzzles.php>.

MORAGLIO, A.; TOGELIUS, J.; LUCAS, S. Product geometric crossover for the sudoku puzzle. In: **2006 IEEE International Conference on Evolutionary Computation**. [S.l.: s.n.], 2006. p. 470–476.

NORVIG, P. **Solving Every Sudoku Puzzle**. 2006. Available from Internet: <https://norvig.com/sudoku.html>.

ONLINE, S. **Candidatos do Sudoku**. 2024. Available from Internet: <https://www.sudokuonline.io/>.

PELANEK, R. Difficulty rating of sudoku puzzles by a computational model. **Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference**, 2011.

PELANEK, R. Difficulty rating of sudoku puzzles: An overview and evaluation. **arXiv preprint arXiv:1403.7373**, 2014.

RACLIFFE, D. **3 million Sudoku puzzles with ratings**. 2021. Available from Internet: <https://www.kaggle.com/datasets/radcliffe/3-million-sudoku-puzzles-with-ratings/data>.

SUDOKU-PUZZLES. **Sudokus disponíveis para resolver**. 2024. Available from Internet: <https://sudoku-puzzles.net/>.

SUDOKU.COM. **Simples óbvios**. 2024. Available from Internet: <https://sudoku.com/br/regras-do-sudoku/simples-obvios/>.

SUDOKUOFTHEDAY. **Técnicas de Sudoku**. 2024. Available from Internet: <https://www.sudokuoftheday.com/>.

WIKIPEDIA-DFS. **Depth-First Search**. 2024. Disponível na Internet. Available from Internet: <https://en.wikipedia.org/wiki/Depth-first_search>.

WIKIPEDIA-SUDOKU. **Sudoku**. 2024. Available from Internet: <https://pt.wikipedia.org/wiki/Sudoku>.