KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Al-Imam Mohammed University
College of Computer & Information Sciences

Sudoku Game

CS361-Artificial Intelligance

Instructor

Ms.Amjad alamrmo

may 18 2022

Alanoud almousa. 440019237

Rania alshodokhi. 440020768

Raghad alghanim. 440022172

CONTENTS:

المحتويات:

Introduction

Recently, artificial intelligence has received great attention because it simply refers to systems and devices that simulate the human mind.

1.SUDOKU

sudoku, also known as **Su Doku**, The popular Japanese puzzle game Sudoku is based on the **logical** placement of numbers. Sudoku doesn't require any calculation nor special math skills; all that is needed are brains and **concentration**.

sudoku consists of a 9 × 9 grid with numbers appearing in some of the squares. The object of the puzzle is to fill the remaining squares, using all the numbers 1–9 exactly once in each row, column, and the nine 3 × 3 subgrids.

2.SUDOKU ORIGINATE

Maki Kaji, the creator of the numbers puzzle sudoku that's drawn fans around the world, has died. He was 69. TOKYO-

Although sudoku-type patterns had been used earlier in agricultural design, their first appearance in puzzle form was in 1979 in a New York-based puzzle magazine, which called them Number Place puzzles. They next appeared in 1984 in a magazine in Japan, where they acquire the name sudoku (abbreviated from *suuji wa dokushin ni kagiru*, meaning “the numbers must remain single”). In spite of the puzzle’s

popularity in Japan, the worldwide sudoku explosion had to wait another 20 years.

3. WHY WE CHOSE SUDOKU

The puzzle,however, raised interesting **combinatorial** problems for mathematicians, two of whom proved in 2005 that there are 6,670,903,752,021,072,936,960 possible sudoku grids.

an **optimization** problem **Sudoku** belongs to the group of **combinatorial** problems, but it is also a constraint **satisfaction** problem. We will test if genetic algorithm **optimization** is an efficient method for solving **Sudoku** puzzles and to generate new puzzles. Another goal is to find out if the puzzles, that are difficult for human solver, are also difficult for the **genetic algorithms**.

Sudoku today is a popular game throughout the world and it appears in multiple medias, including websites, newspapers and books. As a result, it is of interest to find effective Sudoku solving and generating algorithms. For most purposes there already exists satisfactory algorithms, and it might be hard to see the benefit of studying Sudoku solving algorithms. There is, however, still some value in studying Sudoku solving algorithms as it might reveal how to deal with difficult variations of **Sudoku**, such as puzzles with 16x16 grids. **Sudoku** is also, a **NP-Complete** problem which means that it is one of a set of **computational** difficult problems.One goal of this study is therefore to

contribute to the discussion about how such puzzles can be dealt with.

المقدمة

نال الذكاء الاصطناعي في الاونه الاخيره اهتمام كبير لأنه ببساطه يشير الى الانظمة والاجهزة التي تحاكي العقل البشري، سنتطرق في هذه المادة على انشاء لعبة تعتمد على خوارزمية التراجع التي هي نوع من انواع csp .

1- سودوكو

Sudoku ، المعروف أيضًا باسم Su Doku ، تعتمد لعبة الألغاز اليابانية الشهيرة Sudoku على الوضع المنطقي للأرقام. لا تتطلب لعبة Sudoku أي حسابات ولا مهارات رياضية خاصة ؛ كل ما هو مطلوب هو العقول والتركيز.

يتكون سودوكو من شبكة 9 × 9 مع ظهور أرقام في بعض المربعات. الهدف من اللغز هو ملء المربعات المتبقية ، باستخدام جميع الأرقام من 1 إلى 9 مرة واحدة بالضبط في كل صف وعمود والشبكات الفرعية التسع 3 × 3.

2-أصل سودوكو

مات ماكي كاجي ، مبتكر لعبة الألغاز سودوكو للأرقام التي جذبت المعجبين حول العالم. كان عمره 69 عامًا. طوكيو-

على الرغم من استخدام أنماط سودوكو في وقت سابق في التصميم الزراعي ، إلا أن ظهورها الأول في شكل ألغاز كان في عام 1979 في مجلة ألغاز مقرها نيويورك ، والتي أطلق عليها اسم Number Place Puzzles. ظهروا بعد ذلك في عام 1984 في مجلة في اليابان ، حيث حصلوا على اسم سودوكو (مختصر من suuji wa dokushin ni kagiru ، مما يعني "يجب أن تظل الأرقام مفردة").

على الرغم من شعبية اللغز في اليابان ، كان على انفجار سودوكو العالمي أن ينتظر 20 عامًا أخرى.

3. لماذا نختار سودوكو

ومع ذلك ، أثار هذا اللغز مشاكل اندماجية مثيرة للاهتمام لعلماء الرياضيات ، أثبت اثنان منهم في عام 2005 أن هناك 6،670،903،752،021،072،936،960 شبكة سودوكو محتملة.

تنتمي مشكلة التحسين Sudoku إلى مجموعة المشكلات الاندماجية ، ولكنها أيضًا مشكلة إرضاء القيد. سنختبر ما إذا كان تحسين الخوارزمية الجينية طريقة فعالة لحل ألغاز Sudoku ولإنشاء ألغاز جديدة. الهدف الآخر هو معرفة ما إذا كانت الألغاز ، التي يصعب على الإنسان حلها ، صعبة أيضًا على الخوارزميات الجينية.

تعد Sudoku اليوم لعبة شائعة في جميع أنحاء العالم وتظهر في وسائط متعددة ، بما في ذلك مواقع الويب والصحف والكتب. نتيجة لذلك ، من المهم إيجاد خوارزميات فعالة لحل سودوكو وتوليدها. بالنسبة لمعظم الأغراض ، توجد بالفعل خوارزميات مرضية ، وقد يكون من الصعب رؤية فائدة دراسة خوارزميات حل Sudoku. ومع ذلك ، لا تزال هناك بعض القيمة في دراسة خوارزميات حل Sudoku لأنها قد تكشف عن كيفية التعامل مع الاختلافات الصعبة في Sudoku ، مثل الألغاز ذات الشبكات x1616. سودوكو هي أيضًا مشكلة NP-Complete مما يعني أنها واحدة من مجموعة من المشاكل الحسابية الصعبة ، لذلك فإن أحد أهداف هذه الدراسة هو المساهمة في المناقشة حول كيفية التعامل مع مثل هذه الألغاز.

What is sudoku

It is a game in which the player is given a grid of numbers, which is partially filled, and the player completes filling in the numbers according to the restrictions of the game, which is no .repetition in the row, column or cell

It's interesting that Sudoku is a number game, but no math is required for anyone trying to solve it

We must know that there is no known method or rule for finding the result quickly, it is considered an "np-problem"

1.DEFINITIONS

Box: A 3x3 grid inside the Sudoku puzzle.

It works the same as rows and columns,

meaning it must contain the digits 1-9.

Region: This refers to a row, column or box.

Candidate: An empty square in a Sudoku puzzle

have a certain set of numbers that does not

conflict with the row, column and box it is in.

Those numbers are called candidates or candidate numbers.

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

Figure 1. sample

Clue: A clue is defined as a number in the original Sudoku puzzle. Meaning that a Sudoku puzzle have a certain number of clues which is then used to fill in new squares. The numbers filled in by the solver is, however, not regarded as clues.

2.SUDOKU FUNDAMENTALS

A **Sudoku** game consists of a 9x9 grid of numbers, which is divided into nine smaller 3x3 grids (known as minigrids). where each number belong to the range 1-9 . Initially a subset of the grid is revealed and the goal is to fill the remaining grid with valid numbers. The grid is divided into 9 **boxes** of size 3x3. **Sudoku** has only one rule and that is that all **regions**, that is rows, columns, and **boxes**, contains the numbers 1-9 exactly once. In order to be regarded as a proper **Sudoku** puzzle it is also required that a unique solution exists, a property which can be determined by solving for all

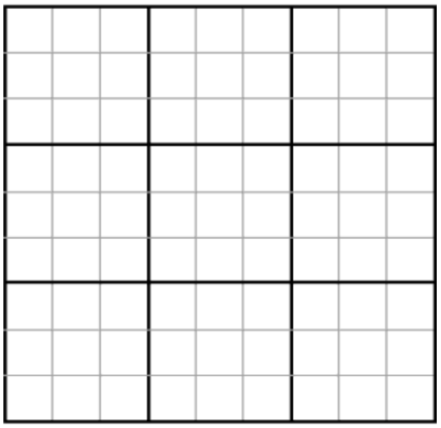


Figure 2. Sudoku grid

possible solutions. Players may use a wide range of strategies which deals the solutions either from zones or from the whole grid.

The aim of the game is to place a number from 1 to 9 into each of the cells, such that each number must appear exactly once in each row and in each column in the grid. Additionally, each minigrid must contain all the numbers 1 through 9. It's also possible to use any other set of symbols. However, using numbers is the obvious choice.

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

Figure 3 partially completed puzzle

A Sudoku puzzle usually comes with a partially filled grid. The aim is to complete the grid in the shortest amount of time.At first glance, a Sudoku puzzle looks simple enough, but upon further examination,it is not as trivial as you might have initially imagined. The placement of the various numbers in a partially filled Sudoku puzzle determines the level of difficulty of the game.

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

Figure 4 A Sudoku puzzle

3. THE DIFFICULTY LEVEL OF SUDOKU

Sudoku puzzles are widely accepted to have varying difficulty levels. The level of difficulty is not always easy to classify as there is no easy way of determining hardness by simply inspecting a grid. Instead the typical approach is trying to solve the puzzle in order to determine how difficult it is. A common misconception about Sudoku is that the number of clues describes how difficult it is. While this is true for the bigger picture it is far from true that specific 17-clue puzzles are more difficult than for instance 30-clue puzzles.

Extremely Easy

- Sudoku have more than 50 givens

Easy

- Sudoku generally have 36 - 49 givens (out of the total of 81 numbers in the answer).
- It has more than one given in every box.
- This level each digit from 1–9 appears as a given at least 3 times.
- It is very quick to solve, less than 10 minutes on average, and no guesses are required.

Medium

- Sudoku have around 32–35 givens.
- It has a couple of boxes with only one given.

- This level has some digits may only appear twice as a given, the rest will appear at least 3 times each.
- It take a bit longer to solve, somewhere between 10–20 minutes. No guesses are required.

Hard

- Sudoku have around 28–31 givens.
- It may have a couple of boxes with only one given
- This level has three or four digits may only appear 2 times as givens, and one digit may only appear once.
- It may take up to 45 minutes to solve, and some trial and error may be needed for example, one box may have two candidates, and no way of determining which .

Very Hard

- Sudoku may have several boxes with less than 22 - 27 givens.
- It may have several boxes with no givens at all
- This level has most digits appear only 2 or 3 times, as well as several single occurrences.
- It can take over an hour to solve, and require trial and error

ماهي لعبة سودوكو ؟

وهي لعبة يمنح فيها الاعب شبكة ارقام ، مملوءه جزئيا ويكمل الاعب تعبئة الارقام حسب قيود اللعبة وهي عدم التكرار في الصف او العمود او الخلية .

من المثير ان لعة سودوكو لعبة ارقام ولكن لايحتاج من يحاول حلها الى اي قدر من الرياضيات

يجب ان نعرف ان ليس هناك طريقة معروفة او قاعدة معينه للعثور على النتيجة بسرعة فهي تعتبر " np-problem"

1. تعاريف

Box: شبكة x33 داخل أحجية Sudoku.

يعمل مثل الصفوف والأعمدة ،

بمعنى أنه يجب أن يحتوي على الأرقام من 1 إلى 9.

المنطقة: يشير هذا إلى صف أو عمود أو مربع.

المرشح: مربع فارغ في أحجية سودوكو

لديك مجموعة معينة من الأرقام التي لا

تتعارض مع الصف والعمود والمربع الموجود فيه.

تسمى هذه الأرقام أرقام المرشحين أو المرشحين.

فكرة: يتم تعريف الدليل على أنه رقم في لغز Sudoku الأصلي. بمعنى أن أحجية سودوكو تحتوي على عدد معين من القرائن والتي تُستخدم بعد ذلك لملء مربعات جديدة. ومع ذلك ، لا تعتبر الأرقام التي تم ملؤها من قبل الحلول بمثابة أدلة.

2-أساسيات سودوكو

تتكون لعبة سودوكو من شبكة x99

من الأعداد مقسمة إلى تسعة

شبكات أصغر حجمًا x33 (تعرف بالشبكات الصغيرة).

حيث ينتمي كل رقم إلى النطاق 1-9

. في البداية يتم الكشف عن مجموعة فرعية من الشبكة و

الهدف هو ملء الشبكة المتبقية مع صالح

أعداد. الشبكة مقسمة إلى 9 مربعات من

الحجم x33. سودوكو لديه قاعدة واحدة فقط وهي تلك

هو أن جميع المناطق ، أي الصفوف والأعمدة والمربعات ، تحتوي على الأرقام من 1 إلى 9 مرة واحدة تمامًا. لكي يتم اعتباره لغزًا مناسبًا لـ Sudoku ، يلزم أيضًا وجود حل فريد ، وهي خاصية يمكن تحديدها من خلال حل جميع الحلول الممكنة. يمكن للاعبين استخدام مجموعة واسعة من الاستراتيجيات التي تتعامل مع الحلول إما من المناطق أو من الشبكة بأكملها.

الهدف من اللعبة هو وضع رقم من

من 1 إلى 9 في كل خلية ، بحيث يكون كل منها

يجب أن يظهر الرقم مرة واحدة بالضبط في كل صف

وفي كل عمود في الشبكة. بالإضافة إلى ذلك ، كل

يجب أن يحتوي minigrid على جميع الأرقام من 1 إلى

9. من الممكن أيضًا استخدام أي مجموعة أخرى من الرموز.
ومع ذلك ، فإن استخدام الأرقام هو الخيار الواضح.

عادة ما يأتي لغز سودوكو جزئيًا
شبكة مملوءة. الهدف هو استكمال الشبكة في
أقصر وقت ممكن للوهلة الأولى ، لعبة سودوكو
يبدو اللغز بسيطًا بما فيه الكفاية ، ولكن بعد ذلك
الفحص ، فهو ليس تافهًا كما قد يكون لديك
تخيلت في البداية. التنسيب من مختلف
الأرقام في لغز سودوكو معبأ جزئيًا
يحدد مستوى صعوبة اللعبة.

3. مستوى الصعوبة في سودوكو

يتم قبول ألغاز Sudoku على نطاق واسع بمستويات صعوبة متفاوتة. ليس من السهل دائمًا تصنيف مستوى الصعوبة حيث لا توجد طريقة سهلة لتحديد الصلابة بمجرد فحص الشبكة. بدلاً من ذلك ، يحاول النهج النموذجي حل اللغز لتحديد مدى صعوبة ذلك. من المفاهيم الخاطئة الشائعة حول لعبة Sudoku أن عدد الأدلة يصف مدى صعوبة ذلك. في حين أن هذا صحيح بالنسبة للصورة الأكبر ، إلا أنه ليس صحيحًا أن الألغاز المحددة المكونة من 17 فكرة أكثر صعوبة من الألغاز المكونة من 30 فكرة على سبيل المثال.

سهل جدا

- **سودوكو لديها أكثر من 50 معطيات**

سهل

- **تحتوي لعبة Sudoku بشكل عام على 36-49 معطيات (من إجمالي 81 رقمًا في الإجابة).**
- **يحتوي على أكثر من واحد في كل صندوق.**
- **هذا المستوى يظهر كل رقم من 1–9 كمعطى 3 مرات على الأقل.**
- **حلها سريع جدًا ، أقل من 10 دقائق في المتوسط ، ولا يتطلب أي تخمينات.**

متوسط

- **تحتوي لعبة Sudoku على حوالي 32-35 معطيات.**
- **يحتوي على صندوقين مع إعطاء واحد فقط.**
- **يحتوي هذا المستوى على بعض الأرقام قد تظهر مرتين فقط كمعطى ، والباقي سيظهر على الأقل 3 مرات لكل منهما.**
- **يستغرق حلها وقتًا أطول قليلاً ، ما بين 10-20 دقيقة. لا توجد تخمينات مطلوبة.**

صعب

- **تحتوي لعبة Sudoku على حوالي 28-31 معطيات.**
- **قد تحتوي على صندوقين مع إعطاء واحد فقط**
- **يحتوي هذا المستوى على ثلاثة أو أربعة أرقام قد تظهر مرتين فقط كمعطيات ، وقد يظهر رقم واحد مرة واحدة فقط.**

- **قد يستغرق الأمر ما يصل إلى 45 دقيقة لحلها ، وقد تكون هناك حاجة إلى بعض التجارب والخطأ ، على سبيل المثال ، قد يحتوي أحد الصناديق على اثنين من المرشحين ، ولا توجد طريقة لتحديد أيهما.**

صعب جدا

- **قد تحتوي لعبة Sudoku على عدة صناديق بها أقل من 22 - 27 معطيات.**
- **قد تحتوي على عدة صناديق بدون معطيات على الإطلاق**
- **يحتوي هذا المستوى على معظم الأرقام تظهر مرتين أو ثلاث مرات فقط ، بالإضافة إلى العديد من التكرارات الفردية.**
- **يمكن أن يستغرق حلها أكثر من ساعة ، وتتطلب التجربة والخطأ**

GAME FORMULATION

1.GENERATING A PUZZLE

When generating a **Sudoku** puzzle, you must satisfy two criteria: it must be solvable, and the solution must be unique. To generate a puzzle, the Generator starts by filling up a 9 x 9 **Sudoku** grid with random numbers in such a way that no number appears twice in any row, column, or box. It then proceeds to remove clues according to your preferences and checks that the puzzle, using the remaining clues, admits a unique solution. When executing the Generator, you can

decide how many clues you want to remove (or eliminate as many as possible). You can also choose to remove clues individually or, to generate pleasingly symmetrical puzzles, two or four at a time. to fill in the initial grid, which represents the solved puzzle, it just inserts one random number (1 to 9) at a time and keeps trying until it finds a valid combination. After completing the removal of the clues, the Generator checks the uniqueness of the solution. What the Generator doesn't do is determine how difficult the new puzzle is.

2. MODELING THE PUZZLE

A **Sudoku** puzzle is a partially filled grid of 9 x 9 cells. Therefore, the obvious way of representing it is to define a matrix of nine rows by nine columns, identify each cell with a pair of numbers indicating row and column starting by zero. the simple grid with one character per cell will not do. The reason is that a Sudoku cell contains a single number only when you have solved it. In general, while you are in

	0	1	2	3	4	5	6	7	8
0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8
1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8
3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8
4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8
5	5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8
6	6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7	6,8
7	7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7	7,8
8	8,0	8,1	8,2	8,3	8,4	8,5	8,6	8,7	8,8

Figure 5. Cell numbering

the middle of solving the puzzle, each cell contains a number of candidates. This means that a Sudoku program must be able to store more than one number in each cell.

تشكيل اللعبة

1- توليد اللغز

عند إنشاء لغز Sudoku ، يجب أن تفي بمعيارين: يجب أن يكون قابلاً للحل ، ويجب أن يكون الحل فريدًا. لإنشاء لغز ، يبدأ المولد بملء شبكة سودوكو 9 × 9 بأرقام عشوائية بحيث لا يظهر أي رقم مرتين في أي صف أو عمود أو مربع. ثم تشرع في إزالة القرائن وفقًا لتفضيلاتك وتتحقق من أن اللغز ، باستخدام القرائن المتبقية ، يعترف بحل فريد. عند تنفيذ المولد ، يمكنك تحديد عدد الأدلة التي تريد إزالتها (أو حذف أكبر عدد ممكن). يمكنك أيضًا اختيار إزالة القرائن بشكل فردي أو لإنشاء ألغاز متناظرة بشكل مبهج ، اثنان أو أربعة في كل مرة. لملء الشبكة الأولية ، التي تمثل اللغز الذي تم حله ، تقوم فقط بإدراج رقم عشوائي واحد (من 1 إلى 9) في كل مرة وتواصل المحاولة حتى تجد مجموعة صالحة. بعد الانتهاء من إزالة القرائن ، يتحقق المولد من تفرد الحل. ما لا يفعله المنشئ هو تحديد مدى صعوبة اللغز الجديد.

2. نمذجة اللغز

لغز سودوكو عبارة عن شبكة ممتلئة جزئيًا مكونة من 9 × 9 خلايا. لذلك ، فإن الطريقة الواضحة لتمثيلها هي تحديد مصفوفة من تسعة صفوف بتسعة أعمدة ، وتحديد كل خلية بزوج من الأرقام التي تشير إلى الصف والعمود بدءًا من الصفر.

الشبكة البسيطة بحرف واحد لكل

لن تفعل الخلية. والسبب هو أن لعبة Sudoku (سودوكو)

تحتوي الخلية على رقم واحد فقط عندما

لقد قمت بحلها. بشكل عام ، أثناء وجودك في

منتصف حل اللغز ، كل خلية

يحتوي على عدد من المرشحين. هذا يعنى

أن برنامج سودوكو يجب أن يكون قادرًا على تخزينه

أكثر من رقم واحد في كل خلية.

ماهي خوارزمية التراجع

خوارزمية التراجع قد تكون صعبة على الشخص اذا لم يكن يعرف الريكيرجن. التراجع هو أسلوب خوارزمي لحل المشكلات بشكل متكرر عن طريق محاولة بناء حل بشكل تدريجي ، قطعة واحدة في كل مرة ، وإزالة تلك الحلول التي تفشل في تلبية قيود المشكلة في أي وقت (يشار إليها هنا بمرور الوقت) الوقت المنقضي حتى الوصول إلى أي مستوى من مستويات شجرة البحث.

WHAT IS BACKTRACK ALGORITHM

The backtracking algorithm may be difficult for a person if he does not know the Recursion.

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to

the time elapsed till reaching any level of the search tree).

خصائص خوارزمية التراجع

مزايا:

من السهل جدا تنفيذها. معظم أكواد التتبع الخلفي هي بشكل عام بضعة أسطر من كود الوظيفة العودية.

سلبيات: الوقت غير فعال للغاية في الكثير من الحالات عندما يكون عامل التفرع كبيرًا تعقيد مساحة كبيرة لأننا نستخدم العودية لذلك يتم تخزين معلومات الوظيفة على المكدس.

PROPERTIES OF BACKTRACK ALGORITHM

Advantages:

It is very easy to implement . Most of the backtracking codes are generally few lines of recursive function code.

Disadvantages: Very time inefficient in lot of cases when branching factor is large Large space complexity because we are using recursion so function information is stored on stack.

Procedure (تطبيق)

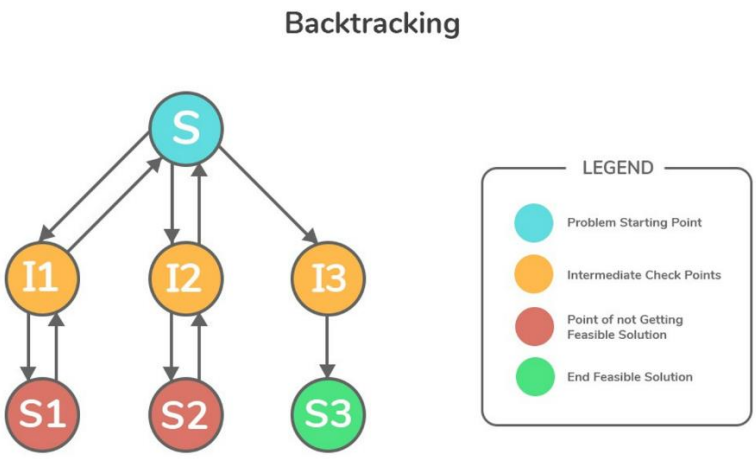
```
procedure AC3-BT(cv)
Q <- {(Vi,Vcv) in arcs(G),i<cv};
consistent <- true;
while not Q empty & consistent
```

```
select and delete any arc (Vk,Vm) from Q;

consistent <- REVISE(Vk,Vm)
```

endwhile
return consistent
end AC3-BT

Example (مثال)



تطبيق خوارزمية التراجع على لعبة سودوكو

وضحنا في الرسم حل خليتين من لعبة سودوكو بأستخدام خوارزمية التراجع.

خوارزمية التراجع هي خوارزمية القوة الفعالة للذاكرة ، مما يجعلها تنفيذ مناسب للأجهزة المدمجة ذات الذاكرة المحدودة. المفهوم من الخوارزمية ما يلي: ابدأ من أي خلية ونفذ الخطوات التالية بشكل متكرر:

(1) إذا كانت الخلية فارغة ، فقم بإضافة رقم غير مقيد.

أ) إذا كان من المستحيل إضافة رقم بسبب قيود ، فقم بالإبلاغ بالفشل.

ب) عدا ذلك ، ابدأ خيطًا جديدًا في خلية جديدة ، بدءًا من الخطوة 2. أنا. إذا أبلغ مؤشر الترابط هذا عن فشل ، فكرر الخطوة 2 بملف رقم (واستنفاد الرقم القديم)

ثانيا. إذا أبلغ هذا الموضوع عن نجاح ، أبلغ عن النجاح ، حيث يمكننا ذلك افترض أنه تم ملء الخلية الأخيرة بنجاح.

إذا تم ملء الخلية ، فتخط هذه الخلية وابدأ سلسلة محادثات جديدة في (2) ملف خلية جديدة ، بدءًا من الخطوة 2.

إذا تمكنت الخوارزمية من تجاوز حدود المجلس ، تقرير النجاح (3).

IMPLEMENTATION OF BACKTRACK ALGORITHM

The backtracking algorithm is a memory efficient brute force algorithm, making it a suitable implementation for embedded devices with limited memory. The concept of the algorithm is the following: Start from any cell and do the following steps recursively:

1. **If the cell is empty, add a number that is not constrained. a) If it is impossible to add a number due to constraints, report failure. b) Else start a new thread on a new cell, starting from step 2. i. If this thread reports a failure, repeat step 2 with a new number (and exhaust the old number) ii. If this thread reports success, report success, since we can assume that the last cell has been successfully filled.**
2. **If the cell is filled, then skip this cell and start a new thread on a new cell, starting from step 2.**

3. **If the algorithm has managed to move beyond the bounds of the board, report success**

In the figure, we showed solving two Sudoku cells using the backtracking algorithm.

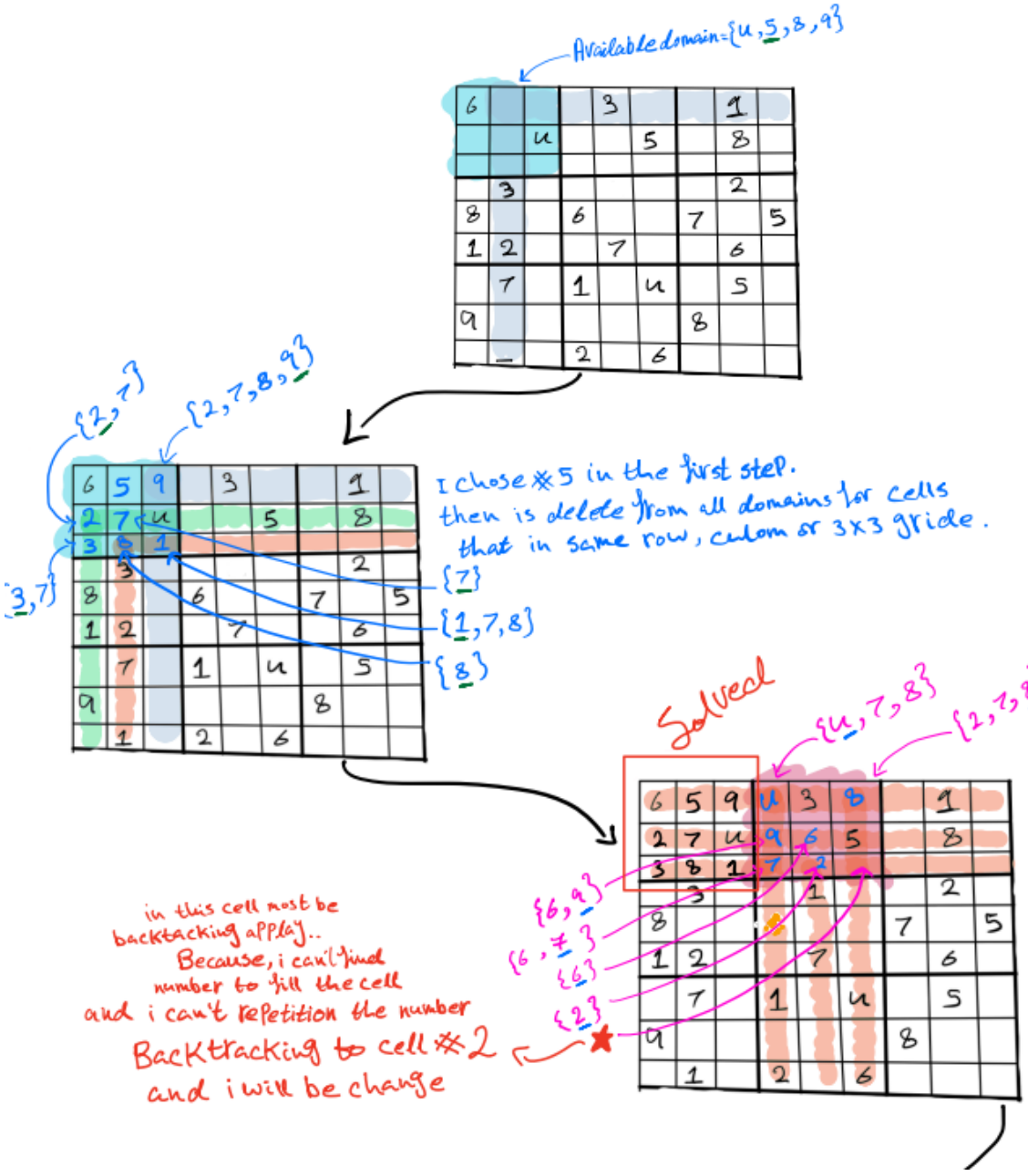
constrains 1: rows should not have duplicate.

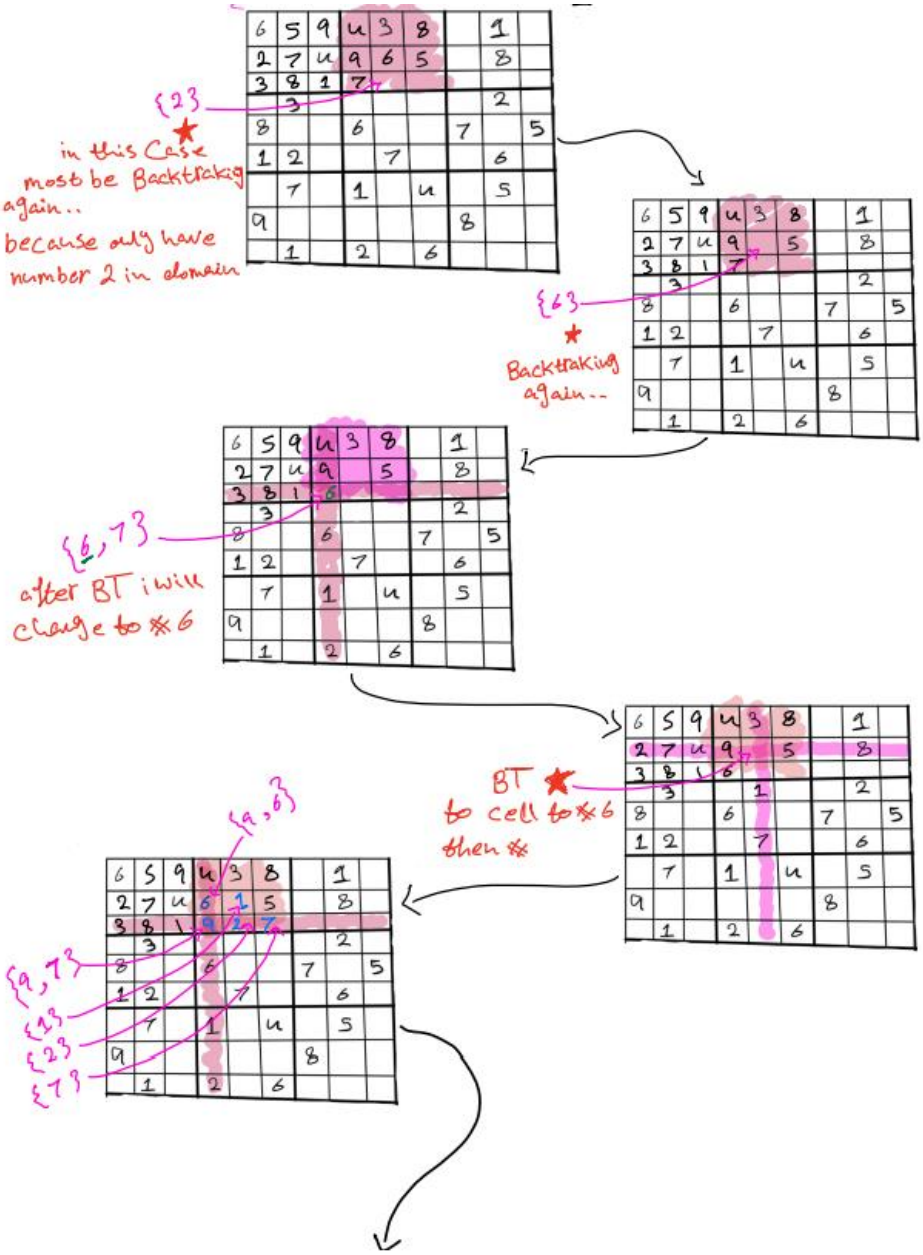
constrains 2:colum should not have duplicate.

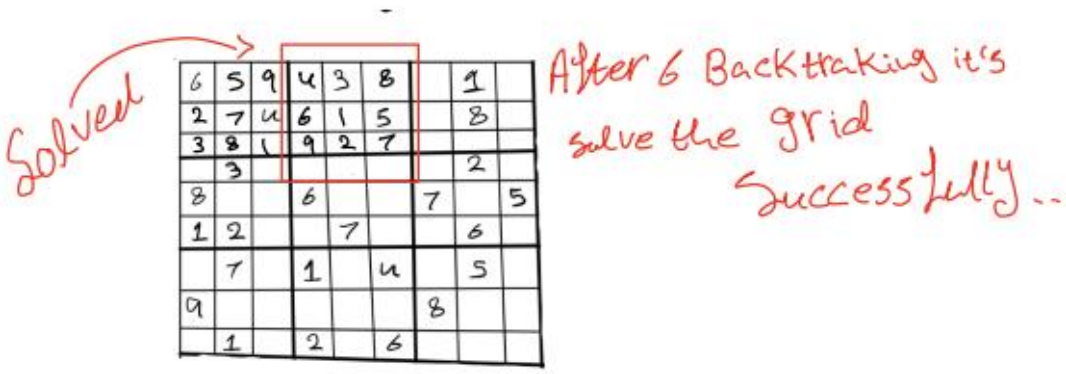
constrains 3: 3*3 grid should not have duplicate.

Domain for every empty cell is {1,2,3,4,5,6,7,8,9}.

Variables: sudoku have 9*9 =81 cell


Available domain={4,5,8,9}
{2,7}
{2,7,8,9}
{3,7}
I chose *5 in the first step.
then is delete from all domains for cells
that in same row, colom or 3x3 gride.
{7}
{1,7,8}
{8}
Solved
{4,9}
{6,≠3}
{6}
{2}
{4,7,8}
{2,7,8}
in this cell most be
backtracking apply..
Because, i can't find
number to fill the cell
and i can't repetition the number
Backtracking to cell *2
and i will be change


{2 3}
in this Case must be Backtraking again.. because only have number 2 in domain
{6 3}
Backtraking again..
{6, 7}
after BT i will change to ✱6
BT
to cell to ✱6
then ✱
{a, 6}
{9, 7}
{1 3}
{2 3}
{7 3}



After explaining the application of the backrtack algorithm in the drawing, we will apply it to the game using the Python programming language.

بعد توضيح تطبيق خوارزمية التراجع في الرسم سنقوم بتطبيقها على اللعبة بأستخدام لغة البرمجة بايثون.

Function "Solve_sudoku"

This function can return fault or True with sulotion

True : indicates that there is a solution to the game of Sudoku according to the conditions.

fault : indicates that there is no solution.

Within the "solve_sudoku" , we will find several functions that you call for the solution, such as the first function, which is "find_next_empty", which is used to find an available place to put the number and give it a value.

This function returns the empty space in terms of a row and a column, This function will work by giving it a complete "puzzle" as an input.

دالة "solve_sudoku"

هذه الدالة من الممكن ان ترجع اما "fault " او "true"

"True" تشير الى وجود حل للعبة السودوكو حسب الشروط

" fault " تشير الى عدم وجود حل للعبة

داخل دالة الحل سنجد عدة دوال تستدعيها للحل مثل اول دالة وهي "find_next_empty " وهي تستخدم لأيجاد مكان متاح لوضع الرقم واعطائه قيمة .

ترجع هذه الدالة ترجع المكان الفارغ بدلالة صف و عمود ، ستعمل هذه الدالة عن طريق اعطائها "puzzle " كاملة كمدخل.

```
def solve_sudoku(puzzle):


    row, col = find_next_empty(puzzle)


    if row is None:
        return True


    for guess in range(1, 10):

        if is_valid(puzzle, guess, row, col):

            puzzle[row][col] = guess

            if solve_sudoku(puzzle):
                return True

        puzzle[row][col] = -1


    return False

if __name__ == '__main__':
    example_board = [
        [3, 9, -1,   -1, 5, -1,   -1, -1, -1],
        [-1, -1, -1,   2, -1, -1,   -1, -1, 5],
        [-1, -1, -1,   7, 1, 9,   -1, 8, -1],

        [-1, 5, -1,   -1, 6, 8,   -1, -1, -1],
        [2, -1, 6,   -1, -1, 3,   -1, -1, -1],
        [-1, -1, -1,   -1, -1, -1,   -1, -1, 4],

        [5, -1, -1,   -1, -1, -1,   -1, -1, -1]
```

How "find_next_empty" works:

First you must know that "1-" means an empty space in the game, it will take "puzzle" as an input in the function and you "will enter into two "for loop

The first will follow the row to find an empty space in field 9.

The second will follow the column to find an empty spot in field 9.

r" and "c" will be compared to the value of 1- if it is equal " then this place will be returned from the function, if not equal it will return "none,none" This means that there is no empty space in the game and this means that the game has been solved and "true" will be returned From the solution function.

طريقة عمل "find_next_empty " :

بداية يجب معرفة ان "1-" تعني مكان فارغ في اللعبة , سيأخذ "puzzle" كمدخل في الدالة وستدخل في ثنين من "for loop"

الأولى ستتبع الصف لأيجاد مكان فارغ في المجال 9 .

الثانية ستتبع العمود لأيجاد مكان فارغ في المجال 9.

سيتم مقارنة "r" و "c" بقيمة 1- اذا كانت تساويه فسيتم ارجاع هذا المكان من الدالة, واذا لم تساوية سيتم ارجاع "none,none" وهذا يعني انه لايجود مكان فارغ في اللعبة وهذا يعني ان اللعبة تم حلها وسيتم ارجاع "true" من دالة الحل

```python
def find_next_empty(puzzle):

    for r in range(9):
        for c in range(9):
            if puzzle[r][c] == -1:
                return r, c

    return None, None
```

بعد رجوع "row,col" فارغ ومناسب لأعطائه قيمة سيتم فحص هذه القيمة.

After "row,col" is returned empty and appropriate to give it a value this value will be checked.

"Guess" check is it appropriate?

Using "for loop" will check the value in the range from 1 to 10 meaning that it is the range(1, 2, 3, ... 9).

Using the "is_valid" function which will be explained shortly, it will take "puzzle, guess, row, col" as an input and they will "be checked. This function returns "true" or "fault".

If "true" is returned, this appropriate value will be placed in the appropriate row and column that was previously found by the "find_next_empty" function.

After setting the appropriate value, the "solve_sudoku" function will be called again "recursively" to find another empty space and solve it, and "true" will be returned. But in case the value is not suitable and "recursion" is used for the "solve_sudoku" function and passes all the values in the field (1, 2, 3, ... 9) then "-1" will be assigned to this place, which means to make it empty and go back

to the previous place And repeat all the previously mentioned operations to verify whether there is another suitable value.
Here, the algorithm of the basis of our project has been applied in AI, which is the "backtracking algorithm".
In the end, "FAULT" will be returned if he cannot solve the game.

فحص "guess " هل هو مناسب ؟

يأستخدام "for loop" سيتم فحص القيمة التي في مجال من 1 الى 10 ومعنى ذلك انه المجال (1, 2, 3, ... 9).

عن طريق دالة "is_valid" التي سيتم شرحها بعد قليل, ستأخذ " puzzle, guess, row, col" كمدخل وسيتم فحصهم , ترجع هذه الدالة "true" او " fault " .

في حالة ارجاع "true" سيتم وضع هذه القيمة المناسبة في الصف والعمود المناسب الذي تم ايجاده مسبقا عن طريق دالة "find_next_empty" .

بعد اسناد القيمة المناسبة سيتم مناداة الدالة "solve_sudoku" مره أخرى "recursively" لأيجاد مكان فارغ اخر وحله , وسيتم ارجاع "true" .

ولكن في حالة كانت القيمة غير مناسبة وتم استخدام "recursion" للدالة "solve_sudoku" والمرور على جميع القيم في المجال (1, 2, 3, ... 9) سيتم اسناد "-1" لهذا المكان وهذا يعني جعله فارغ والرجوع الى المكان السابق وإعادة جميع العمليات المذكورة مسبقا ليتم التحقق م اذا كانت هناك فيمة أخرى مناسبة .

وهنا تم تطبيق خوارزمية أساس مشروعنا في الذكاء الاصطناعي وهي "
backtracking algorithm" .

في الأخير سيتم ارجاع "FAULT" اذا لم يستطيع حل اللعبة .

```
def solve_sudoku(puzzle):


    row, col = find_next_empty(puzzle)


    if row is None:
        return True


    for guess in range(1, 10):

        if is_valid(puzzle, guess, row, col):

            puzzle[row][col] = guess

            if solve_sudoku(puzzle):
                return True

        puzzle[row][col] = -1


    return False

if __name__ == '__main__':
    example_board = [
        [3, 9, -1,   -1, 5, -1,   -1, -1, -1],
        [-1, -1, -1,   2, -1, -1,   -1, -1, 5],
        [-1, -1, -1,   7, 1, 9,   -1, 8, -1],

        [-1, 5, -1,   -1, 6, 8,   -1, -1, -1],
        [2, -1, 6,   -1, -1, 3,   -1, -1, -1],
        [-1, -1, -1,   -1, -1, -1,   -1, -1, 4],

        [5, -1, -1,   -1, -1, -1,   -1, -1, -1]
```

function "is_valid"

Check the assigned value for the row, column, and cell 3 * 3.

It will pass the row, column, value and "puzzle" and return true or false.

First you will check the value for the row:

It will take 'puzzle' and find the row from it, check if the value is in that row it will return an error.

Second, the value will be checked for the column:

In the column is not checked directly as it was in the row

It will be checked by the row's significance, it will take "puzzle" on the i row and the given column, it will pass through all the rows by "for loop" finding the column by its semantics and checking the value if it is in this column it will return "fault".

Third, cell examination 3 * 3:

The number 3 will be dealt with because the board of our game is 9 * 9 and one cell is 3 * 3, the area of one cell will be found by dividing the given row and column by 3 and then multiplying it by 3, after that the row and column will be passed through The "for loop" method and comparing it with the value in case it is equal, will return "fault", meaning that

the value is already in the cell, and this violates the terms of the game, and if it does not exist, it will return "true".

دالة "is_valid "

تفحص القيمة المسنده بالنسبة للصف والعمود والخليه 3 *3 .

سيتم تمرير الصف والعمود والقيمة و "puzzle " وارجاع صح او خطأ .

بالبداية ستفحص القيمة بالنسبة للصف :

ستأخذ "puzzle " و اوجد منه الصف ، يأفحص ما اذا كانت القيمة في هذا الصف فسيتم ارجاع خطأ .

ثانيا سيتم فحص القيمة بالنسبة للعمود:

في العمود لايتم الفحص مباشره كما كان في الصف .

سيتم فحصه بدلالة الصف ، سيأخذ "puzzle " عند صف i والعمود المعطى ، ويتم المرور على جميع الصفوف عن طريق "for loop " زايجاد العمود بدلالتها وفحص القيمه في حالة كانت في هذا العمود سيتم ارجاع "fault "

ثالثا فحص الخليه 3 *3 :

سيتم التعامل مع رقم 3 لان لوحة لعبتنا عبارة عن 9 *9 و الخلية الواحده عبارة عن 3 *3 ، سيتم ايجاد مجال الخليه الواحده عن طريق قسمة الصف والعمود المعطى على 3 ثم ضربه في 3 ، بعد ذلك سيتم المرور على كل من الصف والعمود عن طريق "for loop " ومقارنته في القيمة في حالة كانت مساوية سيتم ارجاع "fault " يعني ان القيمه موجودة في الخليه مسبقا وهذا يخالف شروط اللعبة واذا كانت غير موجوده فسيتم ارجاع "true " .

```python
def is_valid(puzzle, guess, row, col):

    row_vals = puzzle[row]
    if guess in row_vals:
        return False

    col_vals = [puzzle[i][col] for i in range(9)]
    if guess in col_vals:
        return False


    row_start = (row // 3) * 3
    col_start = (col // 3) * 3

    for r in range(row_start, row_start + 3):
        for c in range(col_start, col_start + 3):
            if puzzle[r][c] == guess:
                return False

    return True
```

This is an example given for the game board in our project, it is a board that has more than one solution, as far as we know that it is not professional because the professional board has only one solution and in case of changing any value it will give "fault", but in our example this can be changed the value and it will give us another solution as we will explain right Now.

هذه هي مثال المعطى للوحة اللعبة في مشروعنا ، انها لوحة لها اكثر من حل ، على حد علمنا انها ليست احترافيه لان اللوحة الاحترافية لها حل واحد فقط وفي حالة تغيير اي قيمة ستعطي "fault " ولكن في مثالنا هذا يمكن تغير القيمة وستعطينا حل اخر كما سنوضح الان .

```
example_board = [
    [3, 9, -1,   -1, 5, -1,   -1, -1, -1],
    [-1, -1, -1,   2, -1, -1,   -1, -1, 5],
    [-1, -1, -1,   7, 1, 9,   -1, 8, -1],

    [-1, 5, -1,   -1, 6, 8,   -1, -1, -1],
    [2, -1, 6,   -1, -1, 3,   -1, -1, -1],
    [-1, -1, -1,   -1, -1, -1,   -1, -1, 4],

    [5, -1, -1,   -1, -1, -1,   -1, -1, -1],
    [6, 7, -1,   1, -1, 5,   -1, 4, -1],
    [1, -1, 9,   -1, -1, -1,   2, -1, -1]
]
print(solve_sudoku(example_board))
pprint(example_board)
```

Now let's play it

It's solved correct and return "true".

تم حلها بشكل صحيح و ارجاع "true" .

```
True
[[3, 9, 1, 8, 5, 6, 4, 2, 7],
 [8, 6, 7, 2, 3, 4, 9, 1, 5],
 [4, 2, 5, 7, 1, 9, 6, 8, 3],
 [7, 5, 4, 9, 6, 8, 1, 3, 2],
 [2, 1, 6, 4, 7, 3, 5, 9, 8],
 [9, 3, 8, 5, 2, 1, 7, 6, 4],
 [5, 4, 3, 6, 9, 2, 8, 7, 1],
 [6, 7, 2, 1, 8, 5, 3, 4, 9],
 [1, 8, 9, 3, 4, 7, 2, 5, 6]]
```

If you change number 2, for example, that points in orange color on the above, will give us another solution.

في حالة تغيير رقم 2 على سبيل المثال المشار اليه باللون البرتقالي في الأعلى , ستعطينا حل اخر.

```
example_board = [
    [3, 9, -1,   -1, 5, -1,   -1, -1, -1],
    [-1, -1, -1,   2, -1, -1,   -1, -1, 5],
    [-1, -1, -1,   7, 1, 9,   -1, 8, -1],

    [-1, 5, -1,   -1, 6, 8,   -1, -1, -1],
    [2, -1, 6,   -1, -1, 3,   -1, -1, -1],
    [-1, -1, -1,   -1, -1, -1,   -1, -1, 4],

    [5, -1, -1,   -1, -1, -1,   -1, -1, -1],
    [6, 7, -1,   1, -1, 5,   -1, 4, -1],
    [1, -1, 9,   -1, -1, -1,   3, -1, -1]
]
print(solve_sudoku(example_board))
pprint(example_board)
```

```
True
[[3, 9, 7, 8, 5, 6, 4, 2, 1],
 [8, 6, 1, 2, 3, 4, 9, 7, 5],
 [4, 2, 5, 7, 1, 9, 6, 8, 3],
 [7, 5, 4, 9, 6, 8, 1, 3, 2],
 [2, 1, 6, 4, 7, 3, 5, 9, 8],
 [9, 3, 8, 5, 2, 1, 7, 6, 4],
 [5, 4, 2, 3, 9, 7, 8, 1, 6],
 [6, 7, 3, 1, 8, 5, 2, 4, 9],
 [1, 8, 9, 6, 4, 2, 3, 5, 7]]
```

.Now we will check if the function "is_valid" works

الان سنتحقق من عمل دالة " is_valid"

```
[[3, 9, -1, -1, 5, -1, -1, -1, -1],
 [-1, -1, -1, 2, -1, -1, -1, -1, 5],
 [-1, -1, -1, 7, 1, 9, -1, 8, -1],
 [-1, 5, -1, -1, 6, 8, -1, -1, -1],
 [2, -1, 6, -1, -1, 3, -1, -1, -1],
 [-1, -1, -1, -1, -1, -1, -1, -1, 4],
 [5, -1, -1, -1, -1, -1, -1, -1, -1],
 [6, 7, -1, 1, -1, 5, -1, 4, -1],
 [1, -1, 9, -1, -1, -1, 2, -1, -1]]
```

```
In [6]: is_valid(example_board, 9, 2, 1)
Out[6]: False

In [7]: is_valid(example_board, 4, 2, 1)
Out[7]: True
```

CONCLUSION

We've created the Utility Function that is used by Backtracking algorithm. we found that Backtracking is the suitable algorithm for Sudoku since it can find all or part of the solutions, and it is especially suitable for constraint satisfaction problems

الخاتمة

لقد أنشأنا وظيفة الأداة المساعدة التي تستخدمها خوارزمية التتبع الخلفي. وجدنا أنها هي الخوارزمية المناسبة للعبة سودوكو بما أنها يمكن أن تجد كل أو جزء من الحلول ، وهي مناسبة بشكل خاص لمشاكل الرضا عن القيود

REFERENCES (المراجع)

1. https://imisu-cs340.github.io/
2. https://github.com/SrinidhiRaghavan/AI-Sudoku
3. https://www.geeksforgeeks.org/backtracking-introduction/#:~:text=Backtracking%20is%20an%20algorithmic%2Dtechnique,reaching%20any%20level%20of%20the
4. Programming Sudoku book
5. Sudoku Programming with C book
6. New Developments in Artificial Intelligence and the Semantic Web Proceedings of the 12 th Finnish Artificial Intelligence Conference STeP 2006, Helsinki University of Technology, Espoo, Finland, October 26-27, 2006.
7. Recursive Backtracking for Solving 9*9 Sudoku Puzzle- Dhanya Job and Varghese Paul -Bonfring International Journal of Data Mining, Vol. 6, No. 1, January 2016 .
8. Study on the Performance Characteristics of Sudoku Solving Algorithms -Akta Agrawal M.E student, Computer Technology & Application -Padma Bonde, Ph.D Professor, Computer Department. International Journal of Computer Applications (0975 – 8887) Volume 122 – No.1, July 2015 10.
9. A study of SudokuSolving Algorithms -PATRIK BERGGRENand DAVID NILSSON KTH computer science.

10. Rating and Generating Sudoku Puzzles Based On
11. Constraint Satisfaction Problems Bahare Fatemi, Seyed Mehran Kazemi, Nazanin Mehrasa World Academy of Science, Engineering and Technology International Journal of Computer, Information, Systems and Control Engineering Vol:8 No:10, 2014