Gymnázium, Praha 6, Arabská 14

Programování

Ročníková práce

2023

Ivan Merkulov

Čestné prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 28. dubna 2023 _____

Anotace

Cílem projektu je vytvořit program, který bude schopen vyřešit jakékoli řešitelné sudoku. To znamená, že program musí zaplnit uživatelem zadané sudoku čísly, tak že v každém řádku, sloupci a čtverci 3x3 musí být každé číslo od 1 do 9 právě jednou. Následně program graficky zobrazí správně vyřešené sudoku.

Abstract

The aim of the project is to make a program which could solve every solvable sudoku. This means that we have to fill the sudoku grid with numbers, so that each row, column and square 3x3 must have each number from 1 to 9 exactly once. Then the program will graphically display the right solved sudoku grid.

Obsah

1 Uvod **4**

2 Řešení problému **5**

2.1 Backtracking 5

2.2 Struktura 6

2.2.1 Sudoku Solver 6

2.2.2 Cell Coordinates 9

2.2.3 Hello Application 11

3 Závěr **13**

1 Uvod

Sudoku je logická hra s číslicemi. Cílem hry v základní podobě je doplnit chybějící cifry 1-9 v zadané, zčásti vyplněné čtvercové mřížce s 9×9 poli. Sudoku se skládá z devíti čtverců o rozměrech 3x3. K předem vyplněným číslicím je třeba doplnit další číslice tak, aby platilo, že v každém řádku, v každém sloupci a v každém z devíti čtverců 3x3 jsou použity vždy všechny číslice 1-9 a každá právě jednou. Tato práce se vlastně tímto bude zabývat. Popisuje program, který dokáže vyřešit jakékoli řešitelné sudoku, které mu uživatel zadá. Pokud uživatel zadá neřešitelné sudoku program upozorní uživatele o tom, že sudoku, které zadal není možné vyřešit.

Zadání

Zadání mé ročníkové práce bylo:
Napsat program, který dokáže vyřešit jakékoli řešitelné sudoku a když sudoku je špatně zadané (nemá řešení) program o tom dá uživateli vědět.

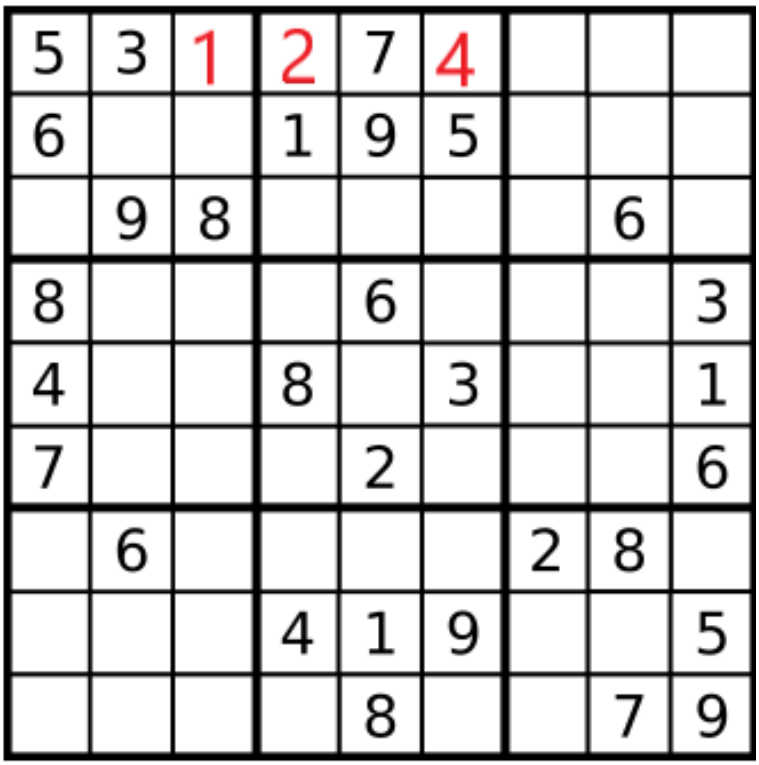
2 Řešení problému

2.1 Backtracking

Pro řešení sudoku program využívá rekurzivní algoritmus Backtracking. Tento algoritmus se snaží postupně vyřešit problém. Zkouší na dané místo vyplnit jednu z možností a když se ta možnost hodí k vyplnění postupuje na další místo, které je potřeba vyplnit, ale když se možnost na dané místo nehodí algoritmus vyzkouší jinou možnost. Pokud vyzkouší všechny možnosti, které mohl vyzkoušet posune se na předchozí pozici a vyzkouší všechny možnosti tam. Tento proces se opakuje do té dob dokud problém není vyřešen.

V případě našeho problému v sudoku postupuje algoritmus následně: Funkce v programu dostane nějaké zadání sudoku a snaží se ho vyřešit. První krok co musí udělat je zjistit, která políčka se mají vyplnit čísly, a která ne. Funkce projíždí mřížku dokud nenarazí na prázdné políčko bez čísla. Když na takové prázdné políčko narazí začne tam zkoušet dosadit čísla od 1 do 9. Program dosadí číslo 1 do prázdného pole a potom zkontroluje zda-li dodržuje všechna pravidla sudoku. To znamená, že zkontroluje jestli se doplněné číslo neopakuje v daném řádku, sloupci a čtverci 3x3. Pokud se číslo hodí do políčka funkce zapíše dané číslo do daného políčka a rekurzivně se znovu zavolá. Po rekurzivním zavolání začne znovu hledat prázdné nevyplněné políčko a bude se snažit doplnit do něj číslo od 1 do 9. Proč po každém doplnění čísla do sudoku voláme rekurzivně funkci znovu, protože když bychom doplnili do sudoku pokaždé první možnost, která se nám bude hodit nemusíme vždy dokázat vyřešit celé sudoku.

Uveďme si to na příkladu: Podívejme se na obrázek 2.1 a zkusme doplnit do tohoto sudoku číslice na prázdná políčka. Budeme postupovat pořádcích zleva doprava. První prázdné políčko,které najdeme je na prvním řádku vedle čísla 3. Zkusíme sem doplnit číslo 1 a zkontrolujeme zda-li toto číslo splňuje podmínky a zjistíme, že ano. Jednička se neopakuje v řádku, sloupci ani čtverci 3x3. Doplníme sem tedy číslo 1. Další prázdné políčko je vedle námi doplněnému číslu 1. Zjistíme, že sem můžeme jako první variantu doplnit číslo 2. Postupujeme dál do dalšího políčka můžeme jako první variantu doplnit



Obr. 2.1: Sudoku

číslo 4 a teďkdyž se posuneme na další prázdné políčko vidíme, že potřebujeme do prvního řádku doplnit čísla 6,8 a 9, ale také vidíme, že políčka do kterých máme doplnit číslo 6 leží ve čtverci, který již číslo 6 obsahuje. To znamená, že tam toto číslo nemůžeme doplnit. Proto voláme rekurzivně funkci po každém doplnění nového čísla do mřížky sudoku, protože se pak funkce může vrátit na předchozí pozici a zkusit tam jinou možnost. V našem případě by to znamenalo, že by se funkce vrátila na políčko kam jsme doplnili číslo 4 a zkusila by tam jiné číslo, které splňuje všechny požadavky proto aby bylo na tomto políčku a zrovna by to vyšlo na číslo 6. Potom bychom mohli dokončit řádek čísly v tomto pořadí 4, 9, 8. [6] [5]

2.2 Struktura

Program se skládá ze tří tříd: Sudoku Solver, Hello Application a CellCoordinates.

2.2.1 Sudoku Solver

Třída Sudoku Solver je takovým „mozkem" celého programu. Vytváří řešeni sudoku. Má parametr matrix = 2D pole. Třída má jeden konstruktor s parametrem 2D pole. [7] [4]

Třída obsahuje tyto metody:

checkRow - zkontroluje, jestli může dosadit do řádku číslo

checkColumn - zkontroluje, jestli může dosadit do sloupce číslo

checkSquare - zkontroluje, jestli může dosadit do čtverce 3x3 číslo

checkValidRow - zkontroluje, jestli se v řádku neopakují čísla

checkValidColumn - zkontroluje, jestli se ve sloupci neopakují čísla

checkValidSquare - zkontroluje, jestli se ve čtverci 3x3 neopakují čísla

checkAllRows - zkontroluje, jestli se neopakují čísla ve všech řádcích v sudoku

checkAllColumns - zkontroluje, jestli se neopakují čísla ve všech sloupcích v sudoku

checkAllSquares - zkontroluje, jestli se neopakují čísla ve všech čtvercích 3x3 v sudoku

checkSudoku - zkontroluje správnost zadaného sudoku

fillCells - vyplní všechna neobsazená políčka správnými čísly dle pravidel sudoku

generate - vygeneruje cele sudoku

print - vytiskne hotové sudoku

Třída dostane 2D pole, ve kterém je zadání sudoku. Pomocí funkce checkSudoku, která využívá funkce checkAllRows, checkAllColumns a checkAllSquares, třída zjistí zda-li je uživatelem zadané sudoku validní nebo ne pokud sudoku zadané uživatelem je validní funkce vrátí true pokud ne false. Metody checkAllRows, checkAllColumns a checkAllSquares kontrolují postupně pomocí metod checkValidRow, checkValidColumn a checkValidSquare.

```
1  public boolean checkValidColumn(int Column) {
2      for (int i = 0; i < this.matrix.length; i++) {
3          for (int j = i + 1; j < this.matrix.length; j++) {
4              if (this.matrix[i][Column] != 0) {
5                  if (this.matrix[i][Column] == this.matrix[j][Column]) {
6                      return false;
7                  }
8              }
9          }
10     }
11     return true;
12 }
```

Listing 2.1: Metoda checkValidColumn

Když budeme mít validní zadání sudoku zavoláme rekurzivní funkci fillCells, která pomocí metod checkRow, checkColumn a checkSquare vyřeší sudoku podle všech jeho pravidel.

```
public boolean fillCells() {
// Porochazim cele pole 9x9 od leveho horniho rohu
for(int cellRow = 0; cellRow < this.matrix.length; cellRow++) {
    for(int cellColumn = 0; cellColumn < this.matrix[cellRow].length;
    cellColumn++) {
        if (this.matrix[cellRow][cellColumn] == 0) {
            for(int newNumber = 1; newNumber <= this.matrix.length;
            newNumber++) {
                if((checkRow(newNumber, cellRow)) &&
                (checkColumn(newNumber, cellColumn)) &&
                (checkSquare(newNumber, cellRow, cellColumn))) {
                    this.matrix[cellRow][cellColumn] = newNumber;
                    //rekurzivne volam funkci
                    if(fillCells()) {
                        return true;
                    }
                    else {
                        this.matrix[cellRow][cellColumn] = 0;
                    }
                }
            }
            return false;
        }
    }
}
return true;
}
```

Listing 2.2: Metoda fillCells

Funkce vyplní všechna neobsazená políčka správnými čísly dle pravidel sudoku. Funkce najde souřadnice prázdného políčka a potom vyplní prázdné políčko vhodným číslem dle pravidel sudoku. Následně provedeme kontrolu řádku, sloupce, čtverce 3x3 abychom se ujistili v tom, že se tam zkoušené číslo neopakuje. Pokud funkce najde číslo, které odpovídá všem parametrům sudoku, zapíše ho do pole a rekurzivně se zavolá. Takovým způsobem postupuje až do té doby než nevyřeší cele sudoku nebo nenarazí na políčko kam

nejde doplnit ani jedno číslo. V takovém případě funkce vrátí false a na současné políčko nastaví 0 a vrátí se na minulé políčko, které bylo prázdné a tam zkusí jiné číslo. To dělá do té doby dokud kompletně nevyřeší sudoku nebo nezkusí všechna prázdná políčka a zjisti ze sudoku nemá řešení.

```
private boolean checkSquare(int Number, int X, int Y) {
    // Vytvorim instanci policka se souradnicemi X, Y
    CellCoordinates cell = new CellCoordinates(X, Y);
    // Ziskam souradnice leveho horniho rohu ctverce 3x3
    CellCoordinates squareLeftCorner = cell.mySquare();

    // Od levehho horniho rohu prochazim ctverec 3x3 po radcich
    for (int i = squareLeftCorner.Row;
    i < squareLeftCorner.Row + 3; i++) {
        for (int j = squareLeftCorner.Column;
        j < squareLeftCorner.Column + 3; j++) {
            if (Number == this.matrix[i][j])
                return false;
        }
    }
    return true;
}
```

Listing 2.3: Metoda checkSquare

Jedna ze tří podobných funkcí checkSquare. Funkce na vstupu dostane souřadnice políčka a číslo, které chceme do něj dosadit. Funkce zkontroluje, jestli může dosadit do políčka číslo, tak ze porovná číslo na vstupu s čísly, které již ve čtverci 3x3 jsou. Pokud najdu nějaké, kterému se číslo rovna, vrátím false, jinak vrátím true. Funkce využívá třídu 2.2.1, proto aby zjistila v jakém čtverci 3x3 se nachází políčko, jehož souřadnice dostane na vstupu.

2.2.2 Cell Coordinates

Třída dokáže zjistit v jakém čtverci 3x3 se nachází naše políčko a zjistí souřadnice levého horního rohu čtverce 3x3, ve kterém je naše políčko. Třída má jeden parametrický konstruktor, který dostane souřadnice našeho políčka a obsahuje jednu metodu mySquare. Metoda

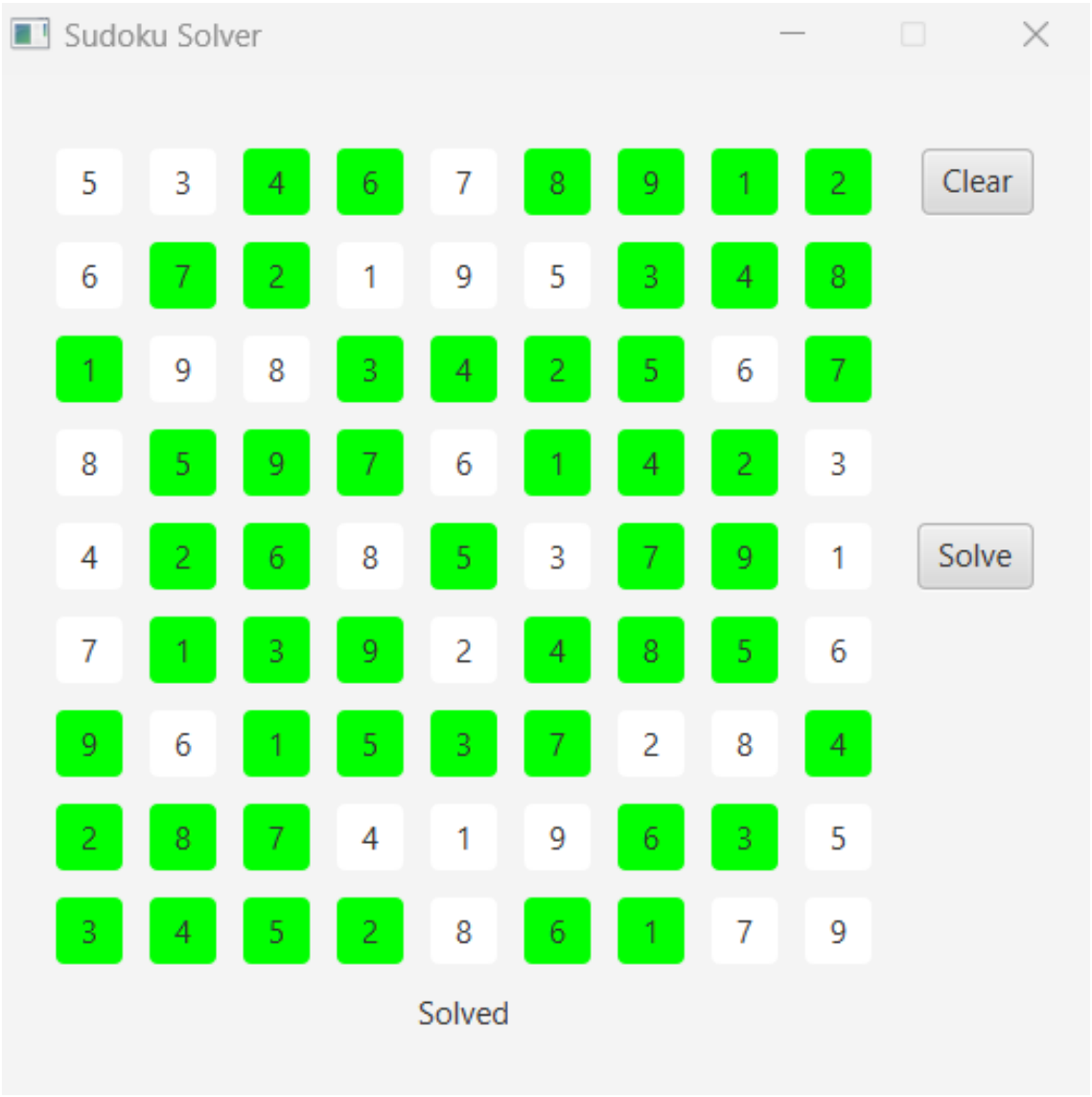
vydělí číslem 3 souřadnice políčka, aby zjistila, ve které ze 3 částí pole 9x9 se políčko nachází. Pokud se podíl bude rovnat 0 políčko se nachází v 1. části atd.

```
1    public CellCoordinates mySquare() {
2        int cornerColumn = 0;
3        int cornerRow = 0;
4        // Najdeme sloupec, ve kterem se nachazi
5        // levy horni roh ctverce
6        if (this.Column / 3 == 0) {
7            cornerColumn = 0;
8        }
9        else if (this.Column / 3 == 1) {
10           cornerColumn = 3;
11       }
12       else if (this.Column / 3 == 2) {
13           cornerColumn = 6;
14       }
15       //Najdeme radek, ve kterem se nachazi
16       //levy horni roh ctverce
17       if (this.Row / 3 == 0) {
18           cornerRow = 0;
19       }
20       else if (this.Row / 3 == 1) {
21           cornerRow = 3;
22       }
23       else if (this.Row / 3 == 2) {
24           cornerRow = 6;
25       }
26
27       return new CellCoordinates(cornerRow, cornerColumn);
28   }
```

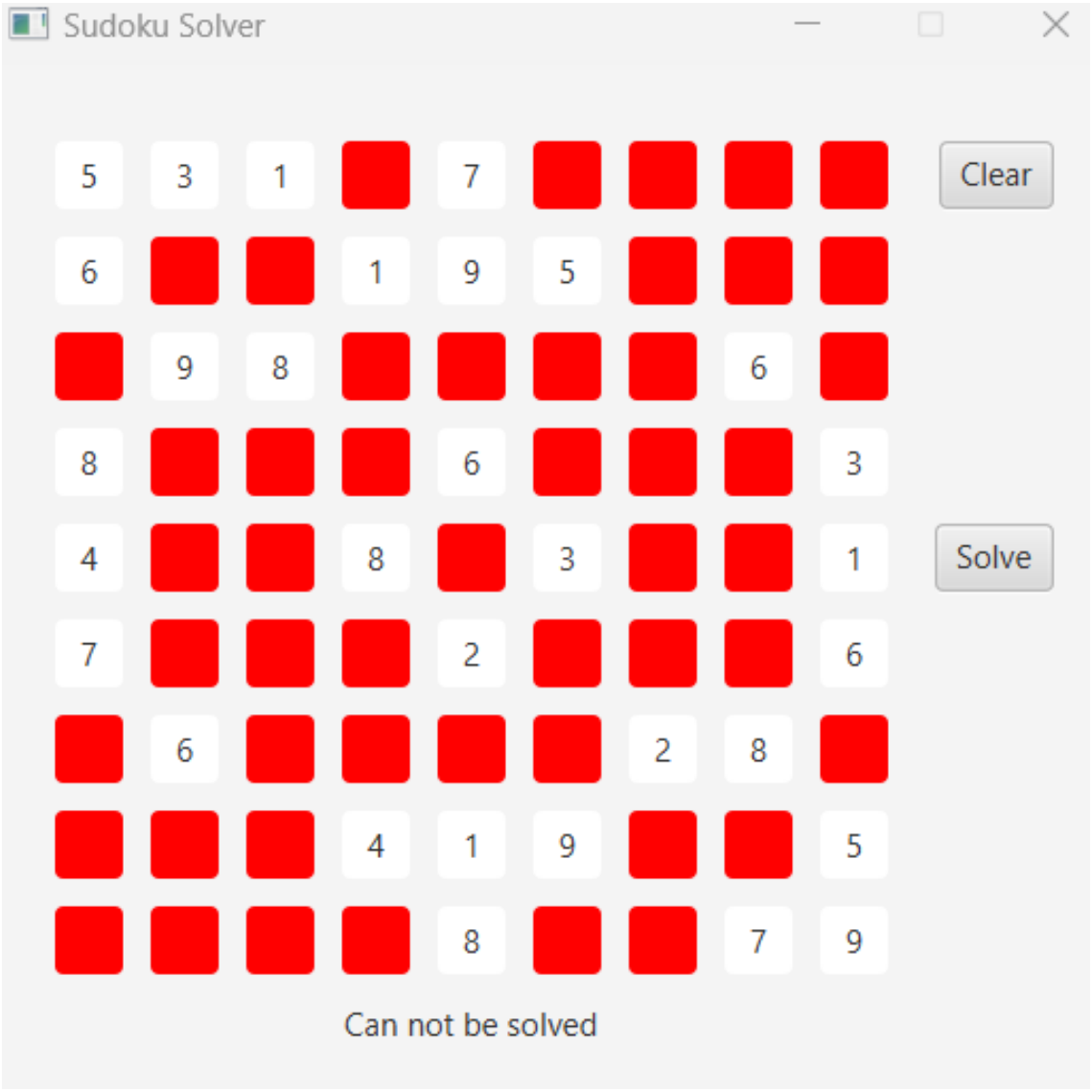
Listing 2.4: Metoda mySquare

2.2.3 Hello Application

Tato třída umožňuje vidět uživateli sudoku v grafickém rozhraní. Také umožňuje uživateli zadat vlastní zadání sudoku. Pro řešení uživatelem zadaného sudoku třída využívá třídu 2.2. Když uživatel zadá řešitelné zadání sudoku program sudoku vyřeší a zobrazí uživateli. Pokud uživatel zadá vadné řešení, tak to program pozná a dá vědět uživateli, že zadání, které zadal je neřešitelné. [3]



Obr. 2.2: Vyřešené sudoku



Obr. 2.3: Neřešitelné sudoku

3 Závěr

Řešení se mi povedlo program dokáže vyřešit jakékoli řešitelné sudoku, které může vlastnoručně zadat uživatel a následně to řešení uživateli i ukáže. Program dokáže rozpoznat sudoku, které nemá validní zadaní a také to zdělí uživateli, který bude moci změnit zadání a bude vědět proč mu program nevyřešil jeho zadané sudoku. Zvládl jsem použít rekurzivní algoritmus Backtracking 2, který umožňuje vyřešit jakékoli řešitelné sudoku. [6] [7] [3] [5] [4] [1] [2]

Zdroje

[1] Chewy. Javafx sudoku gui game [online], 2022. Dostupné z: https://www.youtube.com/watch?v=46aPwIF29Lct=294sab$_channel = Chewy$.

[2] Bro Code. Javafx gui full course [online], 2021. Dostupné z: https://www.youtube.com/watch?v=9XJicRt$_FaIt = 1553sab_channel = BroCode$.

[3] djp3. Coding a sodoku solver 03 - creating a user interface [online], 2020. Dostupné z: https://www.youtube.com/watch?v=zQ5GOoyUEOsab$_channel = djp3$.

[4] djp3. Coding a sodoku solver 04 - creating a board representation [online], 2020. Dostupné z: https://www.youtube.com/watch?v=acWj5-2muWkab$_channel = djp3$.

[5] Neuvedeno. Algorithm to solve sudoku [online], 2023. Dostupné z: https://www.geeksforgeeks.org/sudoku-backtracking-7/.

[6] Neuvedeno. Backtracking algorithms [online], 2023. Dostupné z: https://www.geeksforgeeks.org/backtracking-algorithms/.

[7] Coding with John. Create a sudoku solver in java in 20 minutes - full tutorial [online], 2022. Dostupné z: https://www.youtube.com/watch?v=mcXc8Mva2bAt=900sab$_channel = CodingwithJohn$.