



Универзитет у Бањој Луци
Природно-математички факултет

ВАЛЕНТИНА СУБОТИЋ

Студијски програм: Математика и информатика, број индекса: 78/21

Рјешавач судоку проблема

Семинарски рад

из наставног предмета Основи програмирања 1

Ментор

Милан Предојевић, ма, асистент

Бања Лука, 2022.

САДРЖАЈ

САДРЖАЈ	2
УВОД	3
О ПРОБЛЕМУ	4
ОПИС АЛГОРИТМА	5
РЕЗУЛТАТИ АЛГОРИТМА	8
ЗАКЉУЧАК.....	9
ЛИТЕРАТУРА.....	10

УВОД

Судоку представља математичку загонетку, представљену квадратном мрежом. Изграђена је од различитих величина, али најпопуларнија судоку загонетка је димензије 9x9, која садржи 81 квадрат. У великом квадратном пољу означено је 9 потквadrата димензије 3x3, који су подијељени на 9 поља. Неопходно је попунити сва поља цифрама од 1 до 9, уз поштовање одређених правила.

Приликом покретања загонетке судоку, одређени број поља ће бити попуњен. Узевши у обзир да број попуњених поља, као и њихов распоред нису увијек исти, судоку слагалице се дијеле по тежини. Написани алгоритам нуди рјешење свим степенима тежине.

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

Слика 1. Примјер неријешеног и ријешеног судоку-а

Циљ семинарског рада је направити судоку рјешавач, алгоритам који ће за одређени временски период ријешити судоку, тако што ће попунити празна поља (0) цифрама од 1 до 9.

О ПРОБЛЕМУ

При попуњавању празних поља, неопходно се осврнути на правила која треба да се поштују:

- Сваки број од 1 до 9 се смије појавити тачно 9 пута
- Један број се смије појавити само једном у реду, колони и потквдрату димензије 3x3

Постоји велики број стратегија за рјешавање судоку загонетке.

Неке од технике које воде до рјешења судоку-а су:

- Елиминација - по реду
- Елиминација - по колони
- Претрага усамљених бројева
- Елиминација – по редовима и колонама
- Скривени парови

Детаљније на линку:

<https://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques> .

Такође, постоји неколико алгоритама, који своју примјену огледају у програмским језицима:

- Бектрекинг (енгл. Backtracking)
- Метода оптимизације (енгл. Optimization methods)
- Ограничено програмирање (енгл. Constraint programming)

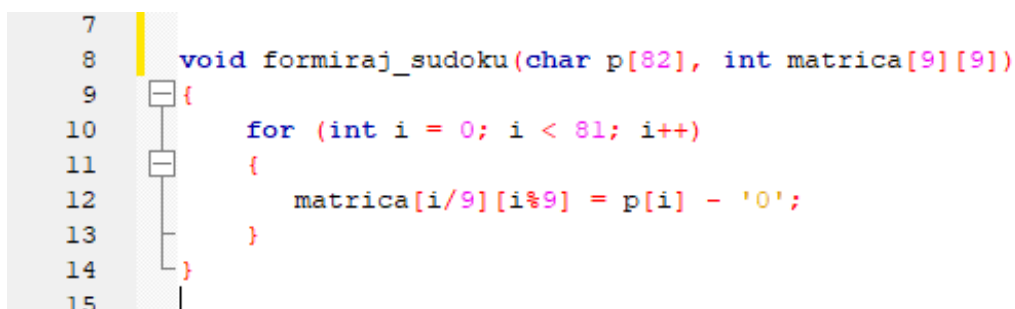
Детаљније на линку:

https://en.wikipedia.org/wiki/Sudoku_solving_algorithms#Exact_cover

ОПИС АЛГОРИТМА

Судоку рјешавач, који се налази у прилогу написан је у програмском језику C, користи бектрекинг алгоритам.

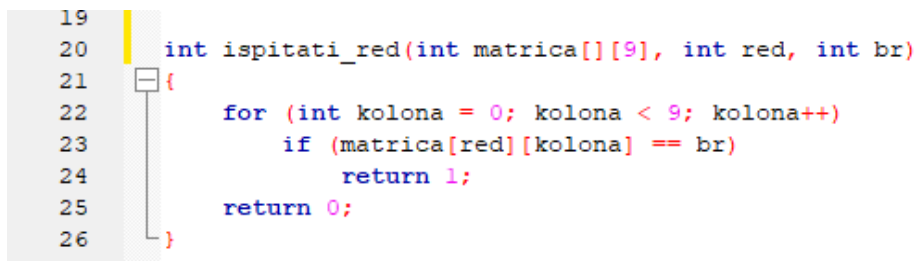
Бектрекинг алгоритам или **метод обрнуте претраге** подразумејева тражење рјешења, гдје се испробавају све могуће комбинације. Алгоритам испробава сва могућа рјешења за свако поље, те ако се пронађе грешку у одређеном пољу, алгоритам се враћа назад и исправља грешку. Његова мана је споро извршавање, а предност је што вријеме извршавања не зависи од тежине судоку-а, те ће сигурно доћи до коначног рјешења.



```
7
8 void formiraj_sudoku(char p[81], int matrica[9][9])
9 {
10     for (int i = 0; i < 81; i++)
11     {
12         matrica[i/9][i%9] = p[i] - '0';
13     }
14 }
15
```

Слика 2. функција 'formiraj_sudoku'

Функција 'formiraj_sudoku' претвара низ карактера p у матрицу цијелих бројева дужине 9x9. Функција као параметар узима низ карактера, који представља бројеве који треба да попуне судоку и од њега направи матрицу 9x9..



```
19
20 int ispitati_red(int matrica[][9], int red, int br)
21 {
22     for (int kolona = 0; kolona < 9; kolona++)
23         if (matrica[red][kolona] == br)
24             return 1;
25     return 0;
26 }
27
```

Слика 3. функција 'ispitati_red'

Функција 'ispitati_red' испитује да ли се број, који можда представља потенцијално рјешење налази у реду, тј. да ли је једнак бројевима који се већ налазе у реду. Функција пролази кроз 9 поља која се налазе у истом реду. Уколико је број на позицији [red][kolona] једнак броју за ког смо сматрали да је потенцијално рјешење функција враћа 1, уколико нису једнаки враћа 0.

```

1  int ispitati_kolonu(int matrica[][9], int kolona, int br)
2  {
3      for (int red = 0; red < 9; red++)
4          if (matrica[red][kolona] == br)
5              return 1;
6      return 0;
7  }
8

```

Слика 3. Функција 'ispitati_kolonu'

Функција 'ispitati_kolonu' испитује да ли се број који можда представља потенцијално рјешење налази у колони, тј. да ли је једнак бројевима који се већ налазе у колони. Функција пролази кроз 9 поља која се налазе у истој колони. Уколико је број на позицији [red][kolona] једнак броју за ког смо сматрали да је потенцијално рјешење функција враћа 1, уколико нису једнаки враћа 0.

```

42
43  int ispitati_mini_kvadrat(int matrica[][9], int pocetni_red, int pocetna_kolona, int br)
44  {
45      for (int red = 0; red < 3; red++)
46          for (int kolona = 0; kolona < 3; kolona++)
47              if (matrica[pocetni_red + red][pocetna_kolona + kolona] == br)
48                  return 1;
49      return 0;
50  }
51

```

Слика 4. Функција 'ispitati_mini_kvadrat'

Функција 'ispitati_mini_kvadrat' испитује да ли се број налази у потквadratu тј. квадрату димензије 3x3, ако се налази враћа 1, а ако се не налази враћа 0

```

57  void ispisi_matricu(int matrica[][9], int n, int m)
58  {
59      for (int i = 0; i < n; i++)
60      {
61          for (int j = 0; j < m; j++)
62              printf(" %d ", matrica[i][j]);
63          printf("\n");
64      }
65

```

Слика 5. Функција 'ispisi_matricu'

Функција 'ispisi_matricu' не враћа резултат. Исписује матрицу димензије 9x9.

```

9
10  int da_li_smo_dobili_br(int matrica[][9], int red, int kolona, int broj)
11  {
12      if (!ispitati_red(matrica, red, broj) && !ispitati_kolonu(matrica, kolona, broj) &&
13          !ispitati_mini_kvadrat(matrica, red - red % 3, kolona - kolona % 3, broj) && matrica[red][kolona] == 0)
14          return 1;
15      return 0;
16  }
17

```

Слика 6. Функција 'da_li_smo_dobili_br'

Функција 'da_li_smo_dobili_br' враћа 1, ако функције ispitati_red ispitati_kolonu ispitati_mini_kvadrat враћају 0, а ако враћају 1 онда функција 'da_li_smo_dobili_br' враћа 0

```

1  int rijesi_sudoku(int matrica[][9])
2  {
3      int red, kolona, pronadjen = 0;
4
5      for (int i = 0; i < 9 && pronadjen==0; i++)
6          for (int j = 0; j < 9 && pronadjen==0; j++)
7              if (matrica[i][j] == 0)
8                  {
9                      red = i;
10                     kolona = j;
11                     pronadjen = 1;
12                 }
13     if(pronadjen==0)
14         return 1;
15
16     for (int broj = 1; broj <= 9; broj++)
17     {
18
19         if (da_li_smo_dobili_br(matrica, red, kolona, broj))
20         {
21             matrica[red][kolona] = broj;
22             if (rijesi_sudoku(matrica))
23                 return 1;
24             matrica[red][kolona] = 0;
25         }
26     }
27
28     return 0;

```

Функција rijesi_sudoku изгледа на следећи начин:

Помоћу промјенљиве pronadjen тражимо празно поље, тј. први елемент који је == 0, ако га нађемо промјенљива pronadjen добија вриједност 1. Ако промјенљива pronadjen не промијени вриједност, тј. остане 0, функција враћа 1, судоку је ријешен.

Ако смо пронашли празно поље, попуњавамо га бројевима од 1 до 9, тако што испитујемо да ли је то прави број позивом функције da_li_smo_dobili_br. Уколико смо закључили да то није прави број, онда броју на тој позицију додјељујемо 0 и улазимо опет у петљу, тако што испробавмо неки други број између 1 и 9.

РЕЗУЛТАТИ АЛГОРИТМА

Вријеме потребно за рјешавање свих судуоку са `char easy_opensudoku[100][82]` је 0.000433 секунде, `char medium_opensudoku[100][82]` је 0.000488 секунди , `char hard_opensudoku[100][82]` је 0.000570 секунди.

ЗАКЉУЧАК

Примијетили смо да овај алгоритам ради за све судоку-е, за све нивое тежине, само што неке уради брже, док неке спорије.

Овим алгоритмом смо сигурни да ћемо добити тачан судоку јер смо користили бектрекинг методу, која попуњава сва поља редом.

ЛИТЕРАТУРА

- <https://sr.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%BE%D0%BA%D1%83>
- <https://hr.sudoku-online.net/>
- <https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-program-in-c/>
- <https://www.geeksforgeeks.org/sudoku-backtracking-7/>
- https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
- <https://sr.wikipedia.org/wiki/%D0%91%D0%B5%D0%BA%D1%82%D1%80%D0%B5%D0%BA%D0%B8%D0%BD%D0%B3>
- <https://bestofsudoku.com/sudoku-strategy>
- <https://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques> .