Implementasi Algoritma Backtracking Pada Game Sudoku Menggunakan Bahasa Pemprograman Python

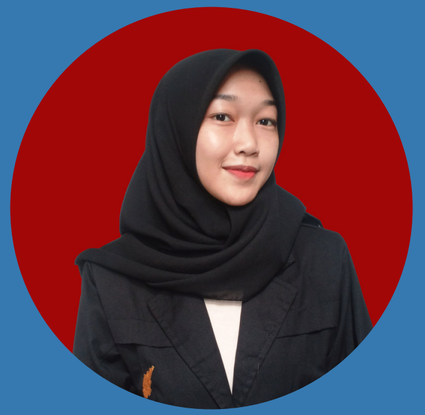
Kelompok 3 - RB

Anggota Kelompok 3

Moch Aditya Suhendar
120450058

Naomi Natasya
120450098

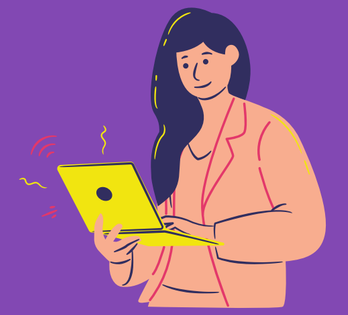
Gregorius Gamma
120450018



Masayu Franstika
120450016

Muhammad Syahid
120450093

INTRODUCTION



Latar Belakang

Sudoku

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

Depth First-Search (DFS)

Algoritma Backtracking

PROBLEM STATEMENT

Backtracking

Masalah penyelesaian sudoku

Data Description

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

METHOD

Deskripsi Umum

METHOD

Algoritma dari Metode Algoritma Strategi

```
procedure RunutBalik(input k: integer)
{ Mencari semua solusi dengan metode
backtracking.
  Masukan: k, yaitu indeks komponen vektor
solusi x[k].
  Keluaran: solusi x = (x[1], x[2], ...,
x[n]).    }
Algoritma:
  for tiap x[k] yang belum dicoba
sedemikian sehingga (x[k] ← T(k)) and
B(x[1], x[2], ..., x[k]) = true do
    if (x[1], x[2], ..., x[k]) adalah
lintasan dari akar ke daun then
      CetakSolusi(x)
    endif
    RunutBalikR(k + 1)
  Endforeach
```

Result and Discussion

Data Awal Sudoku

```
0 3 0 |0 5 5 |5 5 5
5 4 0 |0 1 5 |5 5 0
5 0 0 |1 2 5 |0 5 0
------------------
0 7 2 |0 0 0 |7 6 5
5 0 7 |7 7 5 |5 7 8
5 0 5 |2 0 2 |5 3 5
------------------
6 0 0 |0 7 7 |7 5 5
0 7 7 |2 7 5 |7 5 5
0 0 5 |0 6 5 |3 7 2
```

```
1 3 2 |4 5 5 |5 5 5
5 4 6 |3 1 5 |5 5 7
5 7 8 |1 2 5 |4 5 3
------------------
1 7 2 |3 4 8 |7 6 5
5 3 7 |7 7 5 |5 7 8
5 4 5 |2 1 2 |5 3 5
------------------
6 1 2 |3 7 7 |7 5 5
3 7 7 |2 7 5 |7 5 5
4 8 5 |1 6 5 |3 7 2
Waktu Eksekusi :  0.014168299996526912
```

Hasil Sudoku

Result and Discussion

Implementasi pada fungsi untuk mencari tempat:

```
def solve(bo):
    #print(bo)
    find = find_empty(bo)
    if not find:
        return True
    else:
        row, col = find    ## base case of recursion

    for i in range(1,10):
        if valid(bo, i, (row, col)):
            bo[row][col] = i

            if solve(bo):
                return True

            bo[row][col] = 0
    return False
```

Result and Discussion

Implementasi pada fungsi pembatas

```python
# function for checking valid values

def valid(bo, num, pos):

    # checking the row
    for i in range(len(bo[0])):
        if bo[pos[0]][i] == num and pos[1] != 0:
            return False

    # checking the column
    for j in range(len(bo)):
        if bo[i][pos[1]] == num and pos[1] != 0:
            return False

    # check box
    box_x = pos[1] // 3
    box_y = pos[0] // 3

    for i in range(box_y*3, box_y*3 + 3):
        for j in range(box_x*3, box_x*3 + 3):
            if bo[i][j] == num and (i,j) != pos:
                return False
    return True
```

Result and Discussion

Implementasi fungsi utama backtracking

```python
def print_board(bo) :

    for i in range(len(bo)):
        if i % 3 == 0 and i != 0:
            print("---------------------")
        for j in range(len(bo[0])):                    ## len(bo[0]) is length of each row
            if j % 3 == 0 and j != 0:
                print("|",end="")

            if j == 8 :
                print(bo[i][j])
            else:
                print(str(bo[i][j]) + " ", end="")
#print(print_board(board))

# function for finding empty spots on board

def find_empty(bo): implementasi dngsi utama

    for i in range(len(bo)):
        for j in range(len(bo[0])):
            if bo[i][j] == 0 :
                return (i, j)          ## row,column
    return None

print_board(board)
```

Conclusion

Algorima runut-balik (backtracking) terus berulang hingga seluruh elemen kosong telah diperiksa dan diuji.

Pencarian solusi sudoku dilakukan dari sel sisi kiri atas menuju sel sisi kanan bawah matriks, dengan rincian pencarian pada baris dimulai dari kiri ke kanan dan pada kolom dimulai dari atas kebawah.

Algoritma akan melakukan backtracking/runut balik ke sel sebelumnya apabila terdapat angka yang sama dan angka yang sama tersebut diganti dengan angka yang baru. Pengecekan terus dilakukan sehingga semua sel pada matriks Sudoku telah terisi semua dengan tidak ada kesamaan angka pada satu baris, kolom, dan grid.

THANK YOU