

# Permainan Sudoku menggunakan Algoritma Backtracking serta pengimplementasian program menggunakan Bahasa pemrograman Python

Ibnu Muzakky M. Noor - 1301184329

Program Studi S1 Informatika – Fakultas Informatika  
Universitas Telkom  
Jalan Terusan Buah Batu, Bandung, Indonesia  
e-mail: ibnumuzakky@student.telkomuniversity.ac.id

## ABSTRAK

Permainan Sudoku ini merupakan sebuah permainan teka-teki logika yang mengharuskan untuk mengisi angka angka kedalam kotak/jaring-jaring tanpa ada angka yang berulang di satu baris, kolom dan kotak area/grid. Sudoku biasanya terdiri dari beberapa ukuran kotak. Ukuran kotak terbesar adalah 9x9. Artinya sudoku itu terdiri dari 9 kolom, 9 baris, 81 kotak kecil, dan 9 kotak area/grid. Setiap kotak area/grid berukuran 3 x 3 (3 baris dan 3 kolom). Pada pengerjaan nya permainan sudoku bisa dikerjakan menggunakan strategi algoritma backtracking. Algoritma backtracking sendiri merupakan algoritma yang bertujuan untuk mencari solusi persoalan lebih mangkus. Algoritma backtracking akan mencari solusi persoalan tanpa harus memeriksa seluruh kemungkinan solusi yang ada, dan hanya pencarian yang mengarah ke solusi saja yang dipertimbangkan oleh algoritma ini. Sedangkan pencarian yang tidak mengarah ke sebuah solusi tidak dilanjutkan. Implementasi dari algoritma backtraing dibuat dalam Bahasa python. Dengan tujuan membuat sebuah aplikasi yang nantinya akan membuktikan teori ketepatan algoritma bacetraking dalam menyelesaikan solusi pada permainan sudoku pada pola 9x9. Terdapat 2 program yang di buat, yaitu program yang hanya menampilkan solusi keseluruhan dan program pertama yag digabungkan dengan GUI sebagai tampilan permainan sudoku. Dan juga user bisa menginputkan angka secara manual.

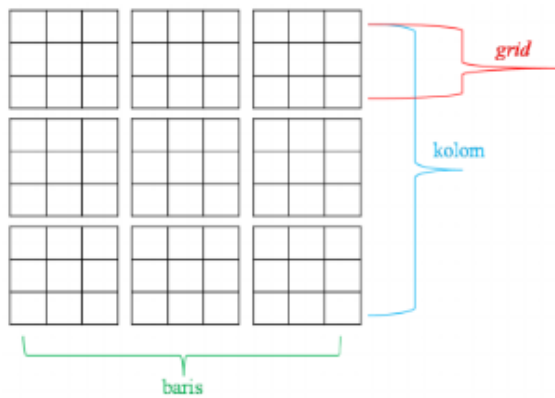
**Kata kunci:** Sudoku, *algoritma backtracking*, kotak, solusi

## 1. PENDAHULUAN

### 1.1 permainan Sudoku

Permainan sudoku pertama kali dipopulerkan oleh beberapa surat kabar Prancis pada akhir abad ke-19 yang mengembangkan konsep permainan sebelumnya yang diciptakan oleh seorang matematikawan Swiss pada abad ke-18 yang mengembangkan konsep “*Latin squares*” dengan konsep permainan bahwa setiap angka atau simbol dalam kotak hanya akan muncul satu kali di setiap baris atau kolom. Sedangkan konsep permainan pada surat kabar Prancis, permainan ini menggunakan dua digit angka dan membutuhkan kemampuan aritmetika dibandingkan kemampuan logika. Kemudian pada tahun 1979, Howard Games mengembangkan sebuah teka teki angka yang dimuat di majalah Dell Magazines Amerika Serikat, permainan ini dikenal dengan nama “*Number Place*”. Dan kemudian permainan ini menjadi populer Kembali di Jepang pada tahun 1986 dengan nama Sudoku [4].

Permainan Sudoku ini merupakan sebuah permainan teka-teki logika yang mengharuskan untuk mengisi angka angka kedalam kotak/jaring-jaring tanpa ada angka yang berulang di satu baris, kolom atau kotak area/grid. Sudoku biasanya terdiri dari beberapa ukuran kotak. Ukuran kotak terbesar adalah 9x9. Artinya sudoku itu terdiri dari 9 kolom, 9 baris, 81 kotak kecil, dan 9 kotak area/grid. Setiap kotak area/grid berukuran 3 x 3 (3 baris dan 3 kolom).



**Gambar 1. Pembagian pada papan sudoku**

[2] Umumnya ada beberapa aturan-aturan dalam memainkan permainan sudoku (9x9) yaitu:

1. Pada baris harus di isi oleh angka-angka mulai dari 1-9 dan tidak boleh ada angka yang kembar dalam satu baris.
2. Pada kolom juga begitu harus di isi dengan angka-angka mulai dari 1-9 dan tidak boleh ada angka yang kembar dalam satu kolom.
3. Pada setiap bagian kotak area dengan ukuran 3 x 3 yang berisi 9 kotak-kotak kecil. Kotak kecil tersebut harus diisi dengan angka 1-9 dan syaratnya tidak boleh ada angka yang berulang pada setiap kotak area.

## 2.1 Algoritma Backtracking

[1]Algoritma backtracking merupakan sebuah algoritma pencarian yang berdasarkan pada pencarian DFS(Depth First Search) yang bertujuan untuk mencari solusi persoalan lebih mangkus. Algoritma backtracking akan mencari solusi persoalan tanpa harus memeriksa seluruh kemungkinan solusi yang ada, dan hanya pencarian yang mengarah ke solusi saja yang dipertimbangkan oleh algoritma ini. Sedangkan pencarian yang tidak mengarah ke sebuah solusi tidak dilanjutkan.

Pada proses pencarian solusi, algoritma Backtracking membentuk sebuah pohon ruang status sebagai ruang solusi yang diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status (state) persoalan, sedangkan isi (cabang) dilabeli dengan nilai-nilai x ke-i. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi. Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (state space tree).

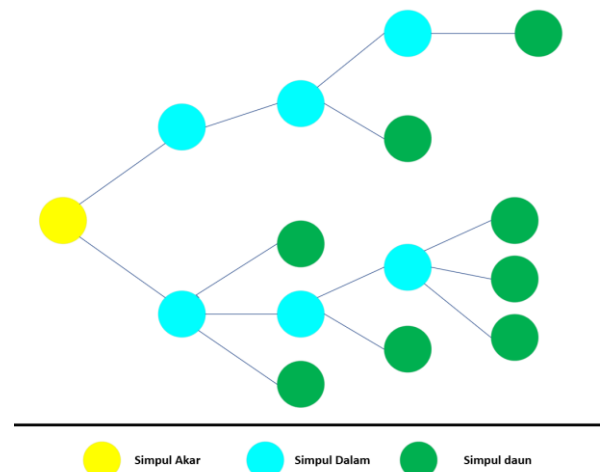
## 2. STRATEGI ALGORITMA

Adapun beberapa strategi terkait penerapan algoritma backtracking pada permainan sudoku Sebagai berikut:

1. Algoritma dimulai memilih kotak kosong pada baris pertama
2. Coba seluruh kemungkinan angka pada kotak kosong tersebut.
3. Temukan angka yag valid dengan aturan permainan sudoku yang berlaku.
4. Lakukan secara berulang hingga mencapai akhir dari solusi
5. Jika angka tidak valid, maka lakukan backtrack ke elemen matriks sebelumnya.

### 2.1 Implementasi Strategi dalam Permainan Sudoku

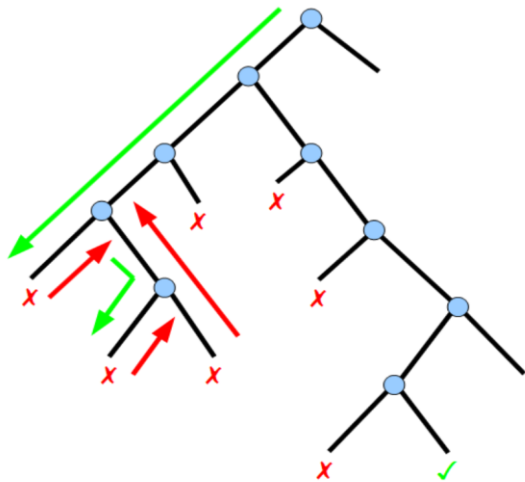
Telah dijelaskan sebelumnya bahwa Algoritma Backtracking membentuk sebuah pohon ruang status sebagai ruang solusi yang diorganisasikan ke dalam struktur pohon. selama prosesnya Struktur pohon inilah pohon ruang status merupakan sebuah graf tak berarah. pada Simpul-simpul pohon ruang status yang tidak mengarah ke solusi maka akan “dimatikan”. Sedangkan simpul-simpul pohon ruang status yang masih mengarah ke solusi maka akan terus berkembang. bisa dilihat pada gambar berikut.



**Gambar 2. Pohon Ruang Status [1]**

**Note:** Backtracking dapat dipandang sebagai pencarian di dalam pohon menuju simpul daun (goal) tertentu

Prinsip pencarian solusi dengan algoritma backtracking bisa kita lihat gambar 3.

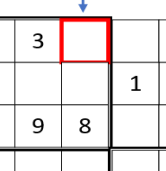


maka prinsip ini akan sama hal nya ketika hal ini di implementasikan kedalam permainan sudoku. Pengimplemenasian ini nantinya bertujuan untuk mengetahui bagaimana cara pada proses pengerjaan permainan sudoku ini menggunakan algortima backtracking sehingga di dapat solusi yang sesuai. Maka untuk membuktikannya dengan melakukan Analisa pengerjaan pada soal sudoku pada gambar 4.

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

**Gambar 4. contoh papan sudoku 9x9**

Angka 1-9



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

pada gambar diatas terdapat kotak kosong dibaris pertama dikolom ketiga, maka dilakukan pencarian kemungkinan angka yang valid pada kotak tersebut dan kemungkinan solusi yang didapat harus bersyaratkan bahwa tidak ada kesamaan angka pada baris, kolom, dan kotak area/grid. misal kita dapatkan angka 4 sebagai solusi pada kotak kosong tersebut.

Angka 1-9

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

**Gambar 6.** kotak kosong yang telah diisi oleh angka 4

setelah ditemukan nya solusi pada kotak kosong tersebut, selanjutnya pengisian kembali pada kotak kosong berikutnya dengan memasukkan kemungkinan angka yang valid sesuai dengan aturan-aturan yang berlaku. misal kita dapatkan angka 6 sebagai solusi pada kotak kosong tersebut. kemudian memeriksa kembali kotak kosong berikutnya dan mengisi kemungkinan angka valid. misal kita dapatkan angka 2 sebagai solusi. selanjutnya lakukan langkah yang sama pada kotak kosong berikutnya. misal kita dapatkan angka 9 sebagai solusi dan hasil sementara bisa kita lihat pada gambar 7.

Angka 1-9

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

**Gambar 7** kotak kosong telah diisi dengan angka 2 dan 9

kemudian pada kotak terakhir di baris pertama, kemungkinan angka yang dapat diisi meyisakan angka 8 untuk dimasukan kedalam kotak. Sedangkan aturan yang berlaku menyebutkan bahwa kemungkinan solusi yang

didapat harus mempunyai ketidaksamaan angka pada baris, kolom, maupun kotak area/grid. dan sedangkan jika angka 8 dimasukan, maka tentunya angka tersebut merupakan kemungkinan solusi yang tidak valid karena di kolom yang sama terdapat angka yang sama. Maka dari itu, dilakukan proses backtrack ke elemen matriks sebelumnya dan merubah angka lama yang digantikan oleh angka dari hasil bactrack.

backtrack

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

Angka sama

**Gambar 8.** proses bactracking ke elemen sebelumnya

Dan dikarenakan pada saat backtrack ke kotak yang berisi angka 9 tidak valid juga dikarenakan terdapat angka 8 pada kotak area yang sama, maka lakukan backtrack kembali ke elemen matriks sebelumnya.

backtrack

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

**Gambar 9** proses bactracking ke elemen sebelumnya

Karena angka pada kotak sudah Kembali valid, maka selanjutnya mengisi kembali kotak yang kosong dengan angka yang valid. dan dikarenakan hanya terdapat 2

kemungkinan solusi yaitu angka 2 dan 9. maka, kita bisa masukan angka 9 terlebih dahulu karena angka tersebut lah yang memiliki solusi yang valid pada kotak tersebut. dan kemudian isikan angka 2 di kotak kosong selanjutnya. Maka didapatkan hasil sementara sebagai berikut.

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

**Gambar 10** baris pertama selesai diisi

Karena baris pertama sudah berhasil diisi penuh dengan angka valid yang sesuai dengan peraturan permainan yang berlaku. Maka lakukan pengerjaan pada baris selanjutnya dengan angka yang valid sehingga tercapai solusi yang terselesaikan seperti gambar dibawah ini.

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

**Gambar 11.** akhir solusi permainan sudoku

Algoritma akan berhenti Ketika ditemukan solusi atau Ketika tidak terdapat kemungkinan solusi.

Adapun algoritma utama yang mencari solusi pada program pengimplementasian permainan sudoku dengan algoritma backtracking menggunakan bahasa python sebagai berikut.

```
def solusi(kotak):
    ketemu = kotakkosong(kotak)
    if not ketemu:
        return True
    else:
        baris,kolom = ketemu

    for i in range(1,10):
        if cekvalid(kotak, i, (baris, kolom)):
            kotak[baris][kolom] = i

            if solusi(kotak):
                return True

            kotak[baris][kolom] = 0

    return False
```

### 3. FUNGSIONALITAS PROGRAM

Terdapat program pengimplementasian permainan sudoku dengan algoritma backtracking yang dibuat oleh Tech With Tim dengan bahasa pemrograman python. program yang dibuat juga memiliki GUI sebagai tampilan antarmuka permainan sudoku. berikut ini penjelasan singkat program utama sebelum di implementasikan kedalam GUI.

#### 1. Papan Sudoku

pertama, dalam permainan sudoku dibutuhkan papan sudoku dan jenis sudoku yang dipakai merupakan sudoku dengan besar 9x9 maka dibutuhkan perancangan terhadap papan tersebut. seperti program dibawah digunakan nya array 2D sebagai perancangan papan dengan asumsi bahwa angka 0 menunjukan kotak kosong.

```
papansudoku = [
    [5,3,0,0,7,0,0,1,0],
    [6,0,0,1,9,5,0,0,8],
    [0,9,8,0,0,0,0,6,0],
    [8,0,0,0,6,0,0,0,3],
    [4,0,0,8,0,3,0,0,1],
    [7,0,0,0,2,0,0,0,6],
    [0,6,0,0,0,0,2,8,0],
    [0,0,0,4,1,9,0,0,5],
    [0,0,0,0,8,0,0,7,9]
]
```

**Gambar 12.** Array untuk papan sudoku

#### 2. Fungsi menampilkan papan sudoku

Selanjutnya terdapat fungsi untuk membuat tampilan papan sudoku dengan array yang telah dibuat. fungsi ini nantinya akan melakukan pemisahan beberapa bagian array yang telah dibuat sehingga nantinya array tersebut akan membentuk sebuah papan sudoku.

```
def tampilanpapan(kotak):
    for baris in range(len(kotak)):
        if baris % 3 == 0 and baris != 0:
            print("-----")

        for kolom in range(len(kotak[0])):
            if kolom % 3 == 0 and kolom != 0:
                print(" | ", end="")

            if kolom == 8:
                print(kotak[baris][kolom])
            else:
                print(str(kotak[kolom][baris]) + " ", end="")
```

**Gambar 13. fungsi menampilkan papan sudoku**

dibaris 2, 3 dan 4 pada digambar diatas berfungsi untuk mencetak garis pembatas horizontal yang digunakan untuk membuat grid/kotak area dimana pembatas tersebut akan tercetak ketika panjang i mod 3 = 0 dan tidak sama dengan 0. Selanjutnya pada baris 5,6 dan 7 bertujuan untuk mencetak garis pembatas secara vertikal yang digunakan untuk membuat grid/kotak area dimana pembatas tersebut akan tercetak ketika panjang j mod 3 dan tidak sama dengan 0. dan selanjutnya terdapat kondisi dimana ketika j = 8 maka akan mencetak kotak [i][j] yang artinya untuk memastikan ketika panjang baris sudah mencapai 9 (0...8) maka akan dilakukan backslash ke baris berikutnya. dan jika tidak maka akan tetap mencetak array untuk tampil pada baris. Hasil output tampilan papan tanpa GUI bisa dilihat dibawah ini.

5	3	0		0	7	0		0	1	0
6	0	0		1	9	5		0	0	8
0	9	8		0	0	0		0	6	0
-	-	-		-	-	-		-	-	-
8	0	0		0	6	0		0	0	3
4	0	0		8	0	3		0	0	1
7	0	0		0	2	0		0	0	6
-	-	-		-	-	-		-	-	-
0	6	0		0	0	0		2	8	0
0	0	0		4	1	9		0	0	5
0	0	0		0	8	0		0	7	9

**Gambar 14 tampilan papan sudoku**

### 3. Fungsi mencari kotak kosong

Selanjutnya terdapat fungsi kotakkosong() yang bertujuan untuk mencari kotak kosong pada papan yang telah dibuat. karena kotak kosong dinotasikan oleh angka 0 pada papan maka program ini akan melakukan pencarian dengan perulangan pada papan sudoku dan

nantinya program akan mereturn posisi kotak kosong tersebut. program bisa dilihat pada gambar dibawah ini

```
def kotakkosong(kotak):
    for i in range(len(kotak)):
        for j in range(len(kotak[0])):
            if kotak[i][j] == 0:
                return (i, j)

    return None
```

**Gambar 15. fungsi pencarian kotak kosong**

### 4. Fungsi mengecek nilai yang valid

fungsi cekvalid(kotak, angka, posisi) yang berfungsi untuk memeriksa apakah angka yang akan di inputkan valid atau tidak. terdapat 3 kondisi pengecekan pada program, yang pertama dengan mengecek pada baris jika terdapat angka di posisi baris yang sama maka bernilai salah atau false, selanjutnya melakukan pengecekan pada kolom apakah ada angka yang sama di kolom yang sama dan jika ada maka akan mereturn false. dan yang terakhir dengan mengecek kotak area/grid jika terdapat angka yang sama pada grid/kotak area maka akan mereturn false. dan jika ada kondisi diluar pada ke 3 kondisi tersebut maka bernilai benar atau true.

```
def cekvalid(kotak, angka, posisi):
    # cek baris
    for i in range(len(kotak[0])):
        if kotak[posisi[0]][i] == angka and posisi[1] != i:
            return False

    # Check column
    for i in range(len(kotak)):
        if kotak[i][posisi[1]] == angka and posisi[0] != i:
            return False

    # Check box
    kotak_x = posisi[1] // 3
    kotak_y = posisi[0] // 3

    for i in range(kotak_y*3, kotak_y*3 + 3):
        for j in range(kotak_x * 3, kotak_x*3 + 3):
            if kotak[i][j] == angka and (i,j) != posisi:
                return False

    return True
```

**Gambar 16 mengecek nilai yang valid**

### 5. Fungsi mencari solusi

fungsi solusi(kotak) yang berfungsi untuk mencari solusi angka yang valid, pada program terdapat 2 kondisi utama.yeng pertama men-set fungsi kotakkosong(kotak) dengan ketemu dan terdapat kondisi if not ketemu maka akan bernilai true dan jika tidak maka akan masuk pada

kondisi ke 2 yang akan melakukan perulangan untuk mengecek nilai (1,10) artinya 1 hingga 9 dan akan memeriksa apakah nilai tersebut valid atau tidak. lalu setelah itu terdapat kondisi di dalamnya yang akan mencoba secara rekursif untuk menyelesaikan solusi dengan memanggil fungsi solusi(kotak). dan jika solusi pemanggilan fungsi solusi(kotak) terus berniali true maka akan dilakukan bactrack ke elemen matriks sebelumnya. dan jika dari kedua kondisi utama tidak ada yang terpenuhi maka akan mengembalikan nilai false.

```
def solusi(kotak):
    ketemu = kotakkosong(kotak)
    if not ketemu:
        return True
    else:
        baris,kolom = ketemu

    for i in range(1,10):
        if cekvalid(kotak, i, (baris, kolom)):
            kotak[baris][kolom] = i

            if solusi(kotak):
                return True

            kotak[baris][kolom] = 0

    return False
```

**Gambar 17. fungsi mencari solusi**

Adapun GUI yang dibangun oleh *Tech With Tim* dengan bahasa pemrograman python. Jika program yang sebelumnya hanya bisa menampilkan solusi saja dan tidak memiliki fitur penginputan angka ada kotak secara manual. Maka dengan adanya GUI ini terdapat fitur penginputan angka pada kotak kosong. Berikut penjelasan singkat dari fungsi GUI.

#### 1. Tampilan awal papan sudoku

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

Time: 0:3

**Gambar 18 tampilan papan sudoku menggunakan GUI**

Pada gambar 18 menggambarkan tampilan awal Ketika aplikasi berhasil dijalankan. Terdapat juga tampilan waktu yang berjalan maju sesuai dengan lama pengerjaan program.

#### 2. Penginputan angka pada kotak

Proses penginputan angka dengan mengarahkan kursor pada kotak yang ingin di isi lalu tuliskan angka yang akan diinputkan jika yakin terhadap angka yang akan diinputkan maka tekan enter. Jika angka yang di inputkan salah maka program akan menampilkan tanda X di bawah papan yang menandakan bahwa jawaban salah.

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

Time: 1:13

**Gambar 19 contoh inputan dengan hasil benar**

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

X Time: 1:46

**Gambar 20 contoh inputan dengan hasil salah**

## 4. SCREENSHOT OUTPUT PROGRAM

### 1. Output Program Tanpa GUI

```

=====sudoku=====
5 3 0 | 0 7 0 | 0 1 0
6 0 0 | 1 9 5 | 0 0 8
0 9 8 | 0 0 0 | 0 6 0
- - - - -
8 0 0 | 0 6 0 | 0 0 3
4 0 0 | 8 0 3 | 0 0 1
7 0 0 | 0 2 0 | 0 0 6
- - - - -
0 6 0 | 0 0 0 | 2 8 0
0 0 0 | 4 1 9 | 0 0 5
0 0 0 | 0 8 0 | 0 7 9

```

Gambar 21 Tampilan angka awal pada papan sudoku

```

=====solusi sudoku=====
5 3 4 | 6 7 8 | 9 1 2
6 7 2 | 1 9 5 | 3 4 8
1 9 8 | 3 4 2 | 5 6 7
- - - - -
8 5 9 | 7 6 1 | 4 2 3
4 2 6 | 8 5 3 | 7 9 1
7 1 3 | 9 2 4 | 8 5 6
- - - - -
9 6 1 | 5 3 7 | 2 8 4
2 8 7 | 4 1 9 | 6 3 5
3 4 5 | 2 8 6 | 1 7 9

```

Gambar 22 solusi akhir dari papan sudoku

### 2. Output Program jika Memakai GUI

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

Time: 0:3

Gambar 23 tampilan angka awal

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

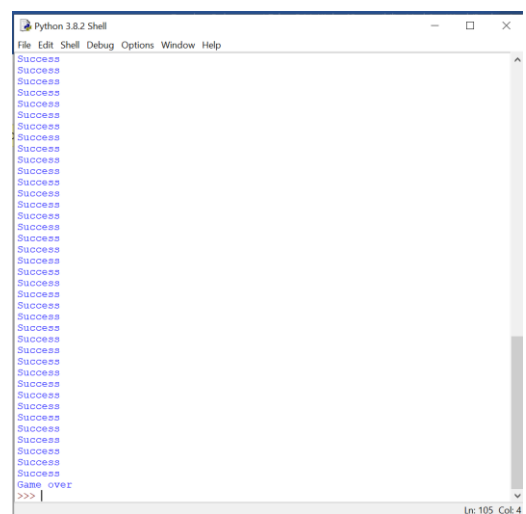
Time: 2:18

Gambar 24 proses pengisian angka

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

Time: 4:40

Gambar 25 solusi akhir dari permainan sudoku



Gambar 26 Riwayat status pengerjaan



## 5. REFERENSI

- [1] Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma, Bandung: Program Studi Teknik Informatika STEI ITB.
- [2] Techwithtim.net. (2019, April). Retrieved from Tech With Tim: <https://techwithtim.net/>
- [3] Zamzani, E. T. (2015, November 6). *Tips Bermain Sudoku dengan Cepat dan Benar*. Retrieved from Kompasiana: <https://www.kompasiana.com/www.etzcoy.blogs.pot.com/563c0ccd6323bd20059498ef/tips-bermain-sudoku-dengan-cepat-dan-benar>
- [4] blj.co.id. (2015, Juni). *Mengenal Asal Permainan Teka Teki Angka Sudoku*. Diambil kembali dari business Lounge Journal: <https://www.blj.co.id/2015/06/06/mengenal-asal-permainan-teka-teki-angka-sudoku/>