Assignment – Week 1

Azhar Harisandi

22320011

1. Soal

- Construct an algorithm (flowchart) to solve sudoku!
- Bonus : Solve the following riddle and submit the code

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

2. Solusi

Pendekatan yang saya ajukan adalah :

1. Cari indeks baris dan kolom untuk kotak kosong, jika tidak terdapat kotak kosong maka puzzle sudah selesai.
2. Jika terdapat kotak kosong maka tebak dengan angka dari 1 hingga 9, dan cek validitasnya dengan melakukan *horizontal scanning, vertical scanning*, dan *local-grid scanning* (grid 3x3).
3. Jika memenuhi kriteria, maka tebakan akan mengisi kotak kosong tersebut, jika tidak posisi di indeks tersebut akan dikembalikan menjadi kotak kosong.
4. Panggil fungsi utama secara rekursif untuk menyelesaikan kotak kotak berikutnya. Disini terlihat bahwa jika di panggilan rekursif ke i suatu tebakan tidak memenuhi syarat atau tidak dapat menyelesaikan puzzle, panggilan rekursi ke i tersebut akan mengalami terminasi dan kembali ke

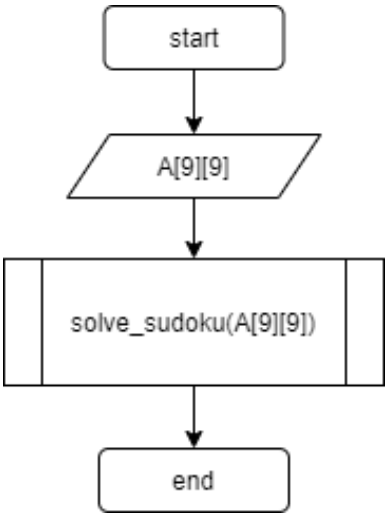
panggilan rekursi ke i-1 dan seterusnya hingga ke panggilan rekursi ke n (dimana n < i) dan di saat itu tebakan akan diubah sehingga panggilan rekursi berikutnya dapat dijalankan.

5. Jika semua tebakan untuk semua sel tidak ada yang memenuhi maka puzzle tidak dapat diselesaikan.

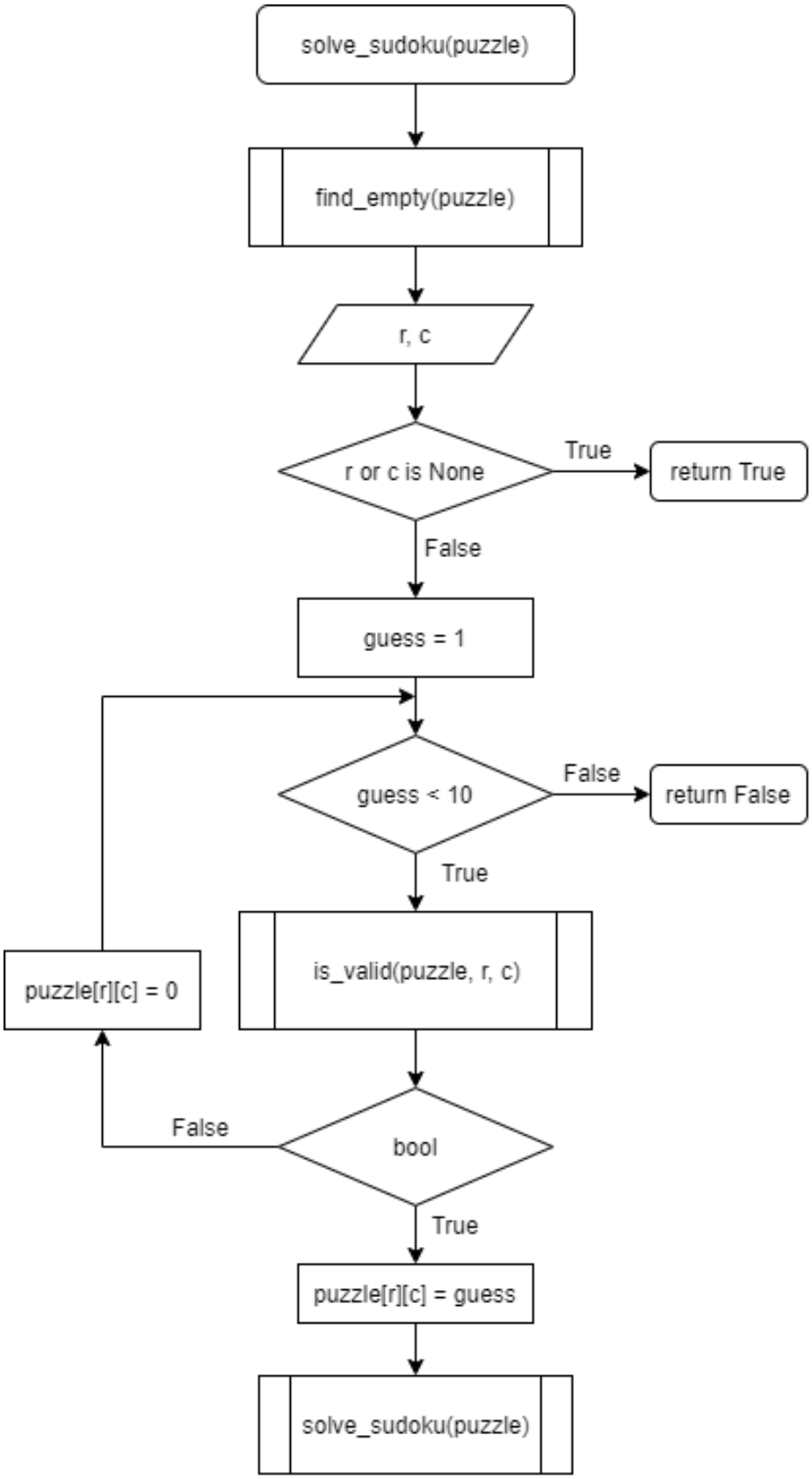
Implementasinya dalam program adalah sebagai berikut. Saya menggunakan struktur data berupa array dua dimensi, dimana dalam python diimplementasikan dalam tipe list. Untuk merepresentasikan tabel 9x9 dibutuhkan list dengan sembilan elemen yang kesembilan elemen tersebut sendiri adalah list, dan setiap elemen dari list tersebut memiliki 9 elemen lagi. Sel kosong ditandai dengan angka 0 karena nilai yang mungkin untuk mengisi puzzle hanya berada di rentang 1 sampai 9. Angka-angka di dalam list akan dimutasi sesuai dengan solusi yang dihasilkan oleh algoritma tanpa harus membuat list baru.

Flowchart penyelesaian

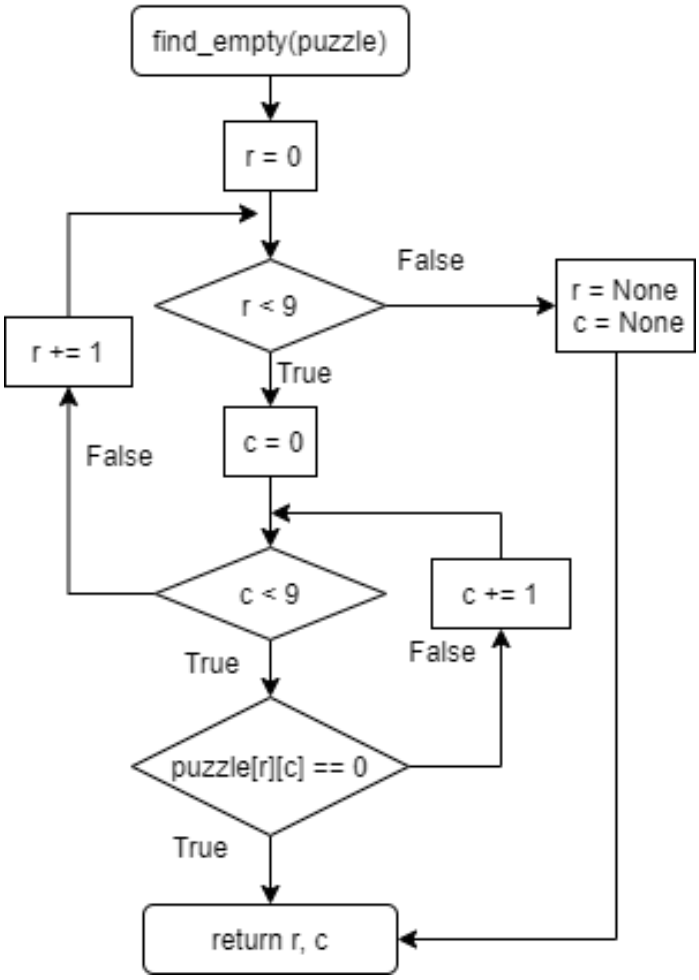
Program utama :



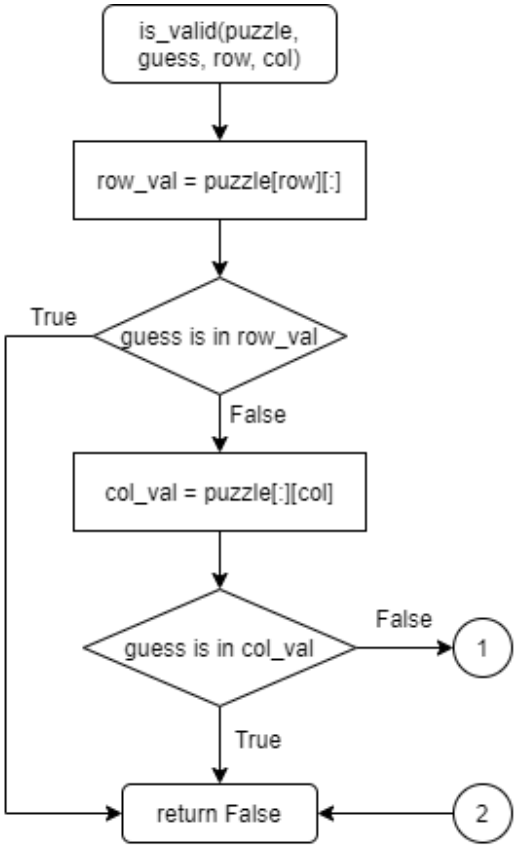
Fungsi solve_sudoku(puzzle):

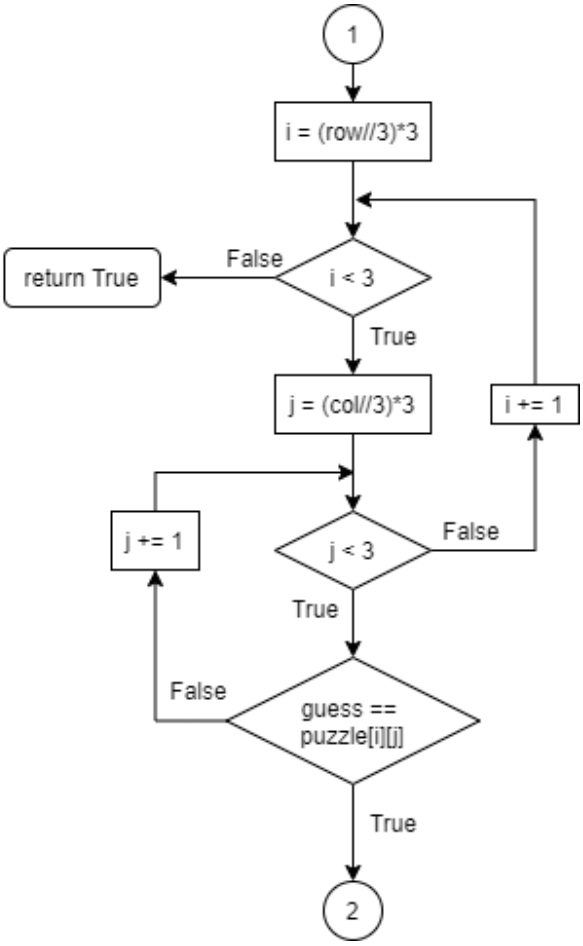


Fungsi find_empty(puzzle):



Fungsi is_valid(puzzle, guess, row, col)





Hasil penyelesaian puzzle

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

Cuplikan code

```
def find_empty(puzzle):
    """
    subroutine untuk mencari indeks baris dan indeks kolom
    yang tidak terisi.

    argumen
    ---------
    puzzle:
        list dengan 9 elemen yang setiap elemennya
        adalah list berisi 9 elemen

    return
    ---------
    row, col:
        index dari baris dan kolom yang kosong, jika puzzle sudah
        penuh maka return None, None
    """
    for row in range(9):
        for col in range(9):
            if puzzle[row][col] == 0: return row, col
    return None, None
```

```python
def is_valid(puzzle, guess, row, col):
    """
    subroutine untuk melihat apakah tebakan untuk mengisi
    sel yang kosong valid. Subroutine ini melakukan horizontal
    scan, kemudian vertical scan dan terakhir local grid scan.

    argumen
    --------
    puzzle:
        puzzle sudoku (list of lists)
    guess:
        tebakan angka untuk mengisi sudoku
    row:
        indeks baris yang ingin diisi
    col:
        indeks kolom yang ingin diisi

    return
    ---------
    boolean:
        jika tidak ditemukan angka yang sama dari hasil
        horizontal scan dan vertical scan dan local grid scan,
        maka return True, jika tidak maka False.
    """

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

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


    # local grid scan (mini 3x3 grid)
    # cari indeks awal grid 3x3 dimana row, col berada
    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
```

```python
def solve_sudoku(puzzle):
    # Fungsi utama

    # 1. cari sel kosong
    row, col = find_empty(puzzle)

    # jika row atau col == None, maka puzzle sudah selesai
    if row is None:
        return True


    # 2. jika ada sel kosong, buat tebakan dengan nilai 1 sampai 9
    for guess in range(1, 10):

        # 3. cek apakah tebakannya valid
        # atau tidak ada elemen yg sama dalam satu baris, kolom, dan grid lokal
        if is_valid(puzzle, guess, row, col):

            # Jika valid, maka ganti nilai di puzzle[row][col]
            # dengan tebakannya
            puzzle[row][col] = guess

            # 4. panggil fungsi secara rekursif
            if solve_sudoku(puzzle):
                return True


        # 5. Jika tidak valid maka ubah kembali nilai sel menjadi 0
        puzzle[row][col] = 0

    # 6. jika tidak ada solusi, maka puzzle tidak bisa diselesaikan
    return False
```

```
if __name__ == "__main__":

    from pprint import pprint

    puzzle =[[5, 3, 0, 0, 7, 0, 0, 0, 0],
             [6, 0, 0, 1, 9, 5, 0, 0, 0],
             [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]]

    solve_sudoku(puzzle)
    pprint(puzzle)
```

```
>>>
= RESTART: C:\Users\PC-USER\Desktop\S2\Semester 2\
Algoritma\Tugas\minggu_01\sudoku.py
[[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]]
```