Implementasi Algoritma Backtracking Pada Game Sudoku Menggunakan Bahasa Pemprograman Python

Tugas Besar Algoritma Strategi

Kelompok 3 - RB

Dosen Pengampu :

Amalya Citra Perdana, S.Kom., M.Si., M.Sc.

Anggota Kelompok :

MOCHAMMAD ADITYA PUTRA SUHENDAR	120450058
MASAYU FRANSTIKA OKTARIA	120450016
GREGORIUS GAMA ABI SURYA	120450018
NAOMI NATASYA GULTOM	120450098
MUHAMAD SYAHID BURHANUDIEN	120450092

INSTITUT TEKNOLOGI SUMATERA

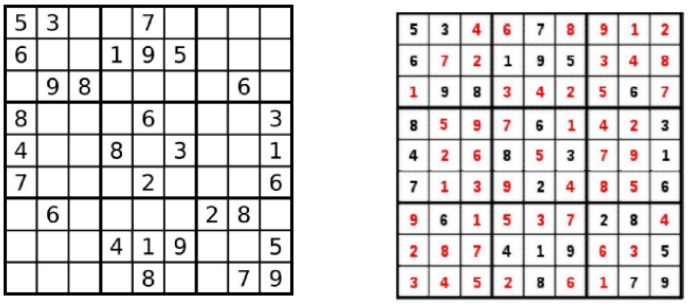
LAMPUNG SELATAN

2022

INTRODUCTION

Di era modern ini, teknologi sangat dibutuhkan untuk mempermudah dan memecahkan masalah untuk meningkatkan efisiensi. Salah satu bidang ilmu yang sering digunakan untuk memecahkan masalah dengan meniru perilaku manusia adalah kecerdasan buatan (AI). Artificial intelligence atau biasa disingkat AI adalah cabang ilmu komputer yang menggunakan kemampuan mesin untuk menyelesaikan tugas dan aktivitas yang biasa dilakukan oleh manusia. AI mengumpulkan dan mengolah berbagai data yang diterimanya menjadi informasi berguna agar dapat menyelesaikan tugas yang diberikan. Kini penggunaan AI adalah hal yang umum dalam segala aspek kehidupan. Dalam kehidupan sehari-hari pun kita akan bertemu dengan banyak pemanfaatan AI yang membantu kita lebih mudah menyelesaikan berbagai tugas.

Perkembangan teknologi informasi terjadi dalam berbagai bidang, salah satunya dalam bidang permainan. Selain sebagai hiburan, permainan juga dapat menjadi suatu hal yang menantang ataupun untuk mengasah kemampuan otak pemain, seperti permainan teka-teki yang saat ini sedang populer di kalangan masyarakat, banyak macam permainan yang di dalamnya diimplementasikan AI, sehingga membuat permainan lebih menarik salah satunya adalah Sudoku. Sudoku adalah sebuah permainan teka-teki angka yang berbasis logika. Pada umumnya, sebuah *puzzle* Sudoku terdiri dari 81 kotak yang disusun menjadi 9 baris, 9 kolom, dan 9 subbagian. Tujuan utama dari permainan ini adalah mengisi seluruh kotak tersebut dengan angka 1 sampai 9.



Sudoku mengharuskan pemainnya untuk berpikir secara logis sesuai dengan aturan-aturan permainan yang ada, permainan memiliki prinsip keunikan, yaitu menciptakan banyak kombinasi angka, yang membuat permainan Sudoku memiliki banyak kemungkinan. Mengisi angka dalam permainan Sudoku ini membutuhkan kesabaran dan ketelitian yang tinggi. Sudoku termasuk dalam masalah NP-complete, sehingga sulit atau bahkan tidak mungkin untuk diselesaikan secara bersamaan. Kami menerapkan algoritma backtracking untuk menghasilkan angka awal di Sudoku untuk mencapai tujuan permainan. Algoritma backtracking dipilih karena dapat dengan cepat menentukan angka yang dihasilkan.

Algoritma backtracking merupakan peningkatan dari algoritma brute force yang membangun pohon state-space untuk menemukan solusi. Pembentukan state space tree pada algoritma backtracking didasarkan pada prinsip depth-first search (DFS). Prinsip dari algoritma backtracking adalah jika terjadi kesalahan dalam menyelesaikan suatu node, maka akan backtrack ke node sebelumnya. Algoritma backtracking bekerja seperti mencoba beberapa kemungkinan yang mengarah pada solusi sampai yang paling cocok ditemukan. Tidak perlu memeriksa semua kemungkinan solusi, tetapi cukup mengarah pada solusi, yaitu dengan memangkas kemungkinan yang tidak mengarah pada solusi, waktu pencarian dapat dihemat. Oleh karena itu, algoritma ini sangat efisien dan ideal untuk menyelesaikan masalah yang berkaitan dengan game AI, terutama game angka seperti Sudoku.

PROBLEM STATEMENT

Algoritma runut-balik (*backtracking*) merupakan algoritma yang cukup baik untuk menyelesaikan persoalan, terutama program game dan persoalan *artificial intelligence.* Menggunakan algortima backtracking dalam menyelesaikan persoalan pengisian angka-angka dalam sudoku cukup baik mendapatkan solusi persoalan tersebut. Pada game sudoku terdapat suatu masalah dalam penyelesaiannya dimana eberapa sel sudoku yang sudah terisi dengan angka dan beberapa sel belum terisi maka diisi dengan angka dengan peraturan setiap sel hanya diisi datu angka yang hanya muncul sekali pada baris, kolom dan area. Semakin sedikit angka yang sudah terisi maka pemain semakin sulit untuk memikirkan angka yang harus diisi. Permasalahan dalam permainannya adalah dibutuhkannya suatu pemikiran dan strategi karena banyak terdapat penghalang dan aturan dalam permainan. Sehingga pemain harus benar-benar memikirkan langkah yang jitu untuk menyelesaikan permainan. Berdasarkan persoalan tersebut. didapatkan perumusan masalahnya adalah bagaimana mengimplementasikan atau menerapkan algoritma *backtracking* pada game Sudoku.

DATA DESCRIPTION

Berikut ini adalah kotak-kotak pada setiap baris, kolom dan blok/subgrid yang berisi angka-angka. Setiap element angka-angka maupun huruf yang di isikan harus unik dan tidak boleh ada yang element dalam setiap baris, kolom dan blok/subgrid yang berulang. Oleh karena itu kami akan melakukan uji coba melalui data dibawah ini dimana data yang kami miliki masih banyak sekali angka-angka yang berulang dalam setiap baris, kolom maupun blok/subgrid.

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

]

METHODS

1. Deskripsi umum Tentang Metode Algoritma Strategi yang digunakan :

   Algoritma *backtracking* (runut balik) pada dasarnya mencari segala kemungkinan solusi seperti halnya *brute-force* dan *exhaustive search*. Yang membedakannya adalah pada *backtracking* semua kemungkinan solusi dibuat dalam bentuk pohon terlebih dahulu baru kemudian pohon tersebut dijelajahi (*explore*) secara *DFS* (*Depth Field Search)*. Secara umum algoritma ini berfungsi dengan baik untuk memecahkan masalah-masalah yang berkembang secara dinamik (*dynamic problem solving)* sehingga menjadi dasar algoritma untuk *Artificial Intelligence* (intelejensia buatan). Makalah ini akan membahas kegunaan algoritma *backtracking* dan bagaimana mengimplementasikannya dalam bahasa pemograman secara umum.

   Karena algoritma *backtrack* akan mencoba menelusuri semua kemungkinan solusi yang mungkin, maka hal pertama yang harus dilakukan adalah membuat algoritma dasar yang akan menjelajahi semua kemungkinan solusi. Kemudian algoritma ini diperbaiki sehingga cara pencarian solusinya dibuat lebih sistematis. Lebih tepatnya jika algoritma itu dibuat sehingga akan menelusuri kemungkinan solusi pada suatu pohon solusi abstrak. Algoritma ini memperbaiki teknik *brute-force* dengan cara menghentikan penelusuran cabang jika pada suatu saat sudah dipastikan tidak akan mengarah ke solusi. Dengan demikian jalan-jalan yang harus ditempuh yajg ada dibawah *node* pada pohon tersebut tidak akan ditelusuri lagi sehingga kompleksitas program akan berkurang.

2. Algoritma / Pseudocode dari Metode Algoritma Strategi yang digunakan :

```
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

Hasil running Program Sudoku solving. Pengujian menggunakan bahasa pemrograman python. Pada gambar 1 menunjukkan Puzzel Sudoku yang belum dikerjakan. Pada gambar 2 merupakan hasil run dari kode program sudoku solving.

```
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
```

Gambar 1.

```
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
```

Gambar 2.

Implementasi pada fungsi untuk mencari tempat

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

    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
```

Pada kasus ini, matriks sudoku berukuran 9x9 sehinggga ukuran sub mattriks yaitu 3x3. Fungsi ini digunakan untuk mencari elemen kosong berikutnya pada matriks valid. Elemen kosong dinyatakan sebagai elemen yang bernilai 0.

Implementasi pada fungsi pembatas

Pada gambar merupakan fungsi untuk memeriksa apakah suatu angka/nilai num memnuhi batasan-batasan dalam sudoku. Akan diperiksa apakah num pada baris dan kolom . Jika angka num ditemukan maka akan mengembalikan False. Jika setelah semua batasan telah diperiksa dan semuanya memenuhi, maka fungsi akan mengembalikan nilai true.

```
def valid(bo, num, pos):

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

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

    #check box (cek kotak)
    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
```

Implementasi fungsi utama *backtracking*.

Fungsi ini menggunakan fungsi sebelumnya yaitu fungsi *find_empty* dan Valid. Pertama menemukan elemen kosong [0], jika tidak ditemukan maka akan mengembalikan fungsi True. Dilanjutkan dengan pengujian angka-angka num {1,2,3,4,5,6,7,8,9} dengan memeriksa batasan untuk num dengan fungsi pembatas Valid. Jika num valid maka isi elemen kosong tersebut dengan nilai num dan teruskan pengujian untuk elemen kosong berikutnya. Jika sukses, maka fungsi akan mengembalikan nilai true. Proses ini akan terus berulang hingga seluruh elemen kosong telah diperiksa dan diuji.

```
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])):
            if j % 3 == 0 and j != 0:
                print("|",end="")

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

KESIMPULAN

Algorima runut-balik (backtracking). Proses ini akan terus berulang hingga seluruh elemen kosong telah diperiksa dan diuji. Oleh karena itu, algoritma runut-balik banyak diterapkan dalam berbagai persoalan, terutama program game dan persoalan artificial intelligence. Telah dihasilkan sebuah aplikasi Sudoku Solver, implementasi dari metode backtracking algorithm untuk menampilkan solusi Sudoku pola 9x9 yang unik dan terbukti bahwa teori metode backtracking algorithm dalam penggunaan pencarian solusi Sudoku pola 9x9 cukup tepat. Pencarian solusi Sudoku pada aplikasi Sudoku Solver dilakukan menggunakan backtracking algorithm/algoritma runut balik, dimana pencarian 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. Pada implementasinya algoritma akanmengumpulkansemua angka kemungkinan yang akan menjadi solusi, kemudian menandai angka tersebut dan meletakkannya di dalam matriks Sudoku. Setelah itu algoritma akan memulai proses pencarian solusi pada sel pertama (sel kiri atas), jika sel pertama merupakan angka awal, maka pencarian dilakukan pada sel setelahnya, begitu seterusnya. Pada proses pencarian algoritma mengecek apakah angka yang ditandai tersebut memiliki kesamaan pada satu baris, kolom, dan grid. Apabila terdapat angka yang sama pada sel kedua, ketiga, dan seterusnya, algoritma akan melakukan backtracking/runut balik ke sel sebelumnya 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.

Refrensi code :

1. https://github.com/Pydare/Sudoku-Game-Solver/blob/master/sudoku_game_solver.ipynb
2. https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking/

Tugas Masing – Masing anggota :

1. Mochammad Aditya Putra Suhendar
   - Menyusun laporan
   - Membuat dan menyusun code python
   - Mencari referensi
   - Mengedit video
2. Masayu Franstika Oktaria
   - Menyusun laporan
   - Membuat dan menyusun code python
   - Mencari referensi
3. Gregorius Gama Abi Surya
   - Menyusun laporan
   - Membuat dan menyusun code python
   - Mencari referensi
4. Naomi Natasya Gultom
   - Menyusun laporan
   - Membuat dan menyusun code python
   - Mencari referensi
5. Muhamad Syahid Burhanudien
