Penerapan Algoritma Backtracking untuk Menemukan Solusi Dalam Permainan Sudoku

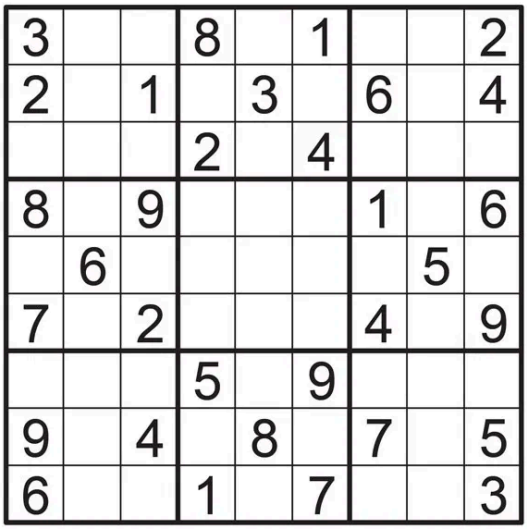
Fabian Radenta Bangun - 13522105
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): radentafabian@gmail.com

***Abstract*—Sudoku adalah permainan angka yang menuntut pemainnya untuk mengisi setiap grid pada grid 9x9 dengan suatu angka dengan aturan tidak boleh ada angka yang sama pada setiap baris, kolom, dan subgrid 3x3 yang ada. Permainan sudoku dapat ditemukan solusinya dengan pendekatan sistematis yang tepat. Salah satu metode yang dapat diterapkan untuk menemukan solusi dari permainan sudoku adalah algoritma *backtracking* atau runut-balik. *Backtracking* adalah metode pemecahan yang mangkus, ia bekerja dengan mengevaluasi setiap pilihan yang mengarah pada solusi dan memangkas pilihan yang tidak mengarah ke solusi. *Backtracking* dapat dipandang juga sebagai sebuah fase di dalam algoritma traversal *Deepening Search First (DFS).* Pada penerapan algoritma *backtracking* di program, akan digunakan prinsip rekursif.**

Keywords—Sudoku, Backtracking, Runut-balik, DFS, Rekursif, Mangkus.

I. PENDAHULUAN

Sudoku adalah permainan teka-teki yang menantang logika dan keterampilan pemainnya. Permainan ini menuntut pemain untuk mengisi grid berukuran 9x9 dengan suatu angka dari 1-9 dengan aturan tidak boleh ada angka yang muncul lebih dari sekali pada baris, kolom, dan subgrid 3x3. Permainan sudoku pertama kali diciptakan oleh Leonhard Euler, seorang matematikawan asal Swiss pada tahun 1783. Pada awal penciptaannya, permainan sudoku bukan ditujukan sebagai sebuah permainan. Sudoku lebih terlihat seperti proyek matematika daripada permainan. Leonhard Euler menemukan ide sudoku karena beliau sedang mengembangkan gagasan tentang cara mengatur susunan angka sehingga tidak ada angka yang muncul berulang pada setiap baris atau kolom. Pencetus permainan sudoku modern seperti yang dapat dilihat pada saat ini adalah Howard Grans, seorang arsitek asal Amerika . Howard Grans menerbitkan permainan sudoku pada majalah *Dell Pencil Puzzles and Word Games* pada Mei 1979. Perkembangan permainan sudoku juga tidak terlepas dari jasa Wayne Gould, seorang pensiunan hakim asal Selandia Baru. Wayne Gould yang tertarik pada permainan sudoku mengembangkan program komputer yang dapat memproduksi teka-teki sudoku secara massal. Kemudian, popularitas permainan sudoku terus berkembang hingga diterbitkan di koran-koran terkemuka seperti *The Times* dan *The Conway Daily Sun.*



Gambar 1.1 Permainan Sudoku
(Sumber: Gambar Permainan Sudoku)

Pencarian solusi permainan sudoku dengan menggunakan pendekatan yang sistematis menjadi subjek penelitian yang menarik di bidang komputasi. Penemuan cara menemukan solusi yang teratur tentu sangat memudahkan komputer untuk melakukan pencarian solusi tanpa perlu melakukan pencarian solusi secara manual oleh pemain. Salah satu metode yang efektif digunakan untuk menemukan solusi dari permainan sudoku adalah algoritma *backtracking* atau dikenal juga sebagai algoritma runut-balik. *Backtracking* adalah teknik pemrograman yang bekerja dengan mencoba mencari solusi dari permasalahan dengan membangun ‘jalur’ solusi secara bertahap dan berjalan mundur (*backtrack*) apabila ‘jalur’ yang dibangun terdeteksi tidak menuju ke arah solusi atau ‘jalur’ buntu.

Dalam konteks permainan sudoku, algoritma *backtracking* bekerja dengan cara mencoba menempatkan suatu angka pada suatu grid kosong. Kemudian dilakukan pemeriksaan apakah penempatan tersebut melanggar aturan permainan sudoku. Jika tidak maka penempatan angka dilanjutkan pada grid kosong selanjutnya. Namun, jika salah maka akan dilakukan *backtrack* dengan mundur 1 langkah ke grid kosong sebelumnya yang sudah terisi oleh suatu angka, kemudian mengganti angka tersebut dengan angka lain dan melakukan pemeriksaan kembali pada grid itu. Langkah ini terus dilakukan hingga solusi ditemukan atau diketahui teka-teki permainan sudoku

yang diperiksa tidak memiliki solusi. Keunggulan dari penerapan algoritma *backtracking* pada pemecahan solusi permainan sudoku adalah pada kesederhanaannya. Algoritma ini sangat mudah dipahami oleh penggunanya. Selain itu, pencarian solusi yang dilakukan dengan algoritma *backtracking* bersifat komplit, artinya selama solusi dari permainan sudoku itu ada maka solusi pasti ditemukan. Namun, efektivitas dari algoritma ini sangat bergantung pada cara pengguna mengimplementasikan algoritmanya. Sangat penting bagi pengguna untuk mempertimbangkan optimasi sebelum mengimplementasikan algoritma ini agar pencarian dapat dilakukan dengan efisien.

Makalah ini bertujuan untuk menjelaskan dengan rinci mengenai penerapan algoritma *backtracking* atau algoritma runut-balik dalam menemukan solusi dari permainan sudoku mulai dari inisiasi awal hingga hasil pencarian ditemukan. Dengan adanya makalah ini, penulis berharap pembaca mendapatkan wawasan lebih mengenai keunggulan dan penerapan algoritma *backtracking* pada kasus pencarian solusi permainan sudoku.

II. Teori Dasar

A. Konsep Permainan Sudoku

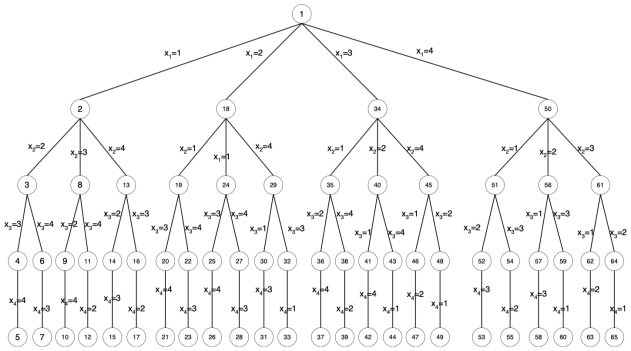
Sudoku adalah permainan tentang bagaimana cara menyusun angka-angka pada sebuah grid berukuran 9x9. Grid berukuran 9x9 tersebut dibagi lagi menjadi 9 subgrid yang masing-masing subgridnya berukuran 3x3. Setiap subgrid tersebut disebut sebagai blok. Tujuan dari permainan ini adalah mengisi setiap sel pada grid dengan angka dari 1 hingga 9. Namun, untuk mengisi sel-sel pada grid, ada aturan permainan yang harus diikuti. Aturan pengisian angka pada permainan sudoku adalah sebagai berikut :

1. Setiap angka hanya boleh muncul sekali dalam setiap baris.
2. Setiap angka hanya boleh muncul sekali dalam setiap kolom.
3. Setiap angka hanya boleh muncul sekali dalam setiap blok.

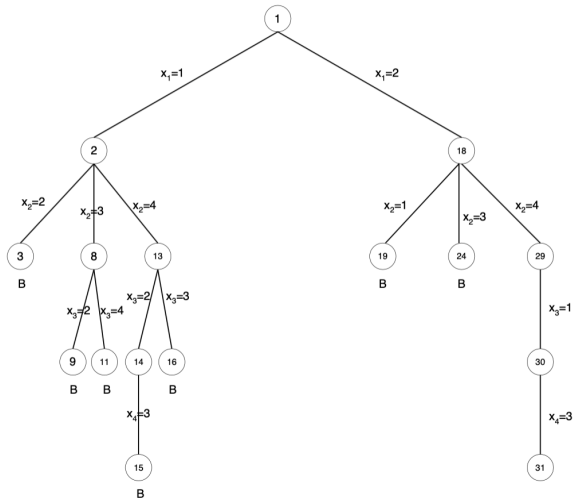
Di awal permainan, keadaan grid akan sudah terisi sebagian. Hal ini menjadi tantangan bagi pemain. Angka yang diisi pada sel-sel yang kosong harus dapat menyesuaikan kondisi angka yang sudah ada tanpa melanggar aturan. Aturan yang sedemikian rupa menuntut pemain untuk menggunakan strategi dan pemikiran logis untuk menemukan solusi. Tingkat kesulitan dari permainan sudoku bervariasi tergantung pada distribusi angka di awal permainan.

B. Algoritma Backtracking

Algoritma *backtracking* atau yang biasa juga disebut algoritma runut-balik diperkenalkan untuk pertama kalinya oleh D. H. Lemer pada tahun 1950 dan disajikan uraian umumnya oleh R. J. Walker, Golomb, dan Baumert. Algoritma ini merupakan salah satu teknik penyelesaian masalah pencarian solusi sistematis yang umum digunakan dalam pemrograman. Algoritma ini adalah bentuk perbaikan dari metode pencarian *exhaustive search*. *Exhaustive search* melakukan pencarian kepada semua kemungkinan solusi tanpa mempertimbangkan apakah kemungkinan yang sedang diperiksa sudah melanggar syarat sebagai solusi. Semua kemungkinan akan dieksplorasi dan dievaluasi satu per satu. Sementara itu, pada algoritma *backtracking*, kemungkinan yang dievaluasi hanyalah kemungkinan yang mengarah menuju solusi. Kemungkinan yang sudah melanggar syarat sebagai solusi akan diabaikan dan tidak dilanjutkan pemeriksaannya. Algoritma ini melakukan *pruning* (pemangkasan) terhadap pilihan-pilihan yang tidak mengarah ke solusi. Untuk keterangan lebih jelasnya dapat dilihat pada gambar 2.1 dan 2.2.



Gambar 2.1 Ilustrasi Ruang Pencarian *N Queen Problem* dengan nilai n adalah 4



Gambar 2.2 Ilustrasi perjalanan pencarian dengan menggunakan algoritma *backtracking*.

(Sumber: Gambar Perjalanan Pencarian Backtracking)

Gambar 2.1 adalah ilustrasi ruang pencarian *N Queen Problem* dengan nilai n adalah 4. Simpul pada gambar menunjukkan kemungkinan yang dapat dipilih dan sisi nya menunjukkan pilihan yang diambil. Jika melakukan pencarian solusi dengan menggunakan metode *exhaustive search*, pencarian akan dilakukan hingga sampai di simpul paling bawah (simpul dengan derajat paling tinggi) kemudian baru menentukan apakah jalur tersebut adalah solusi dari pencarian. Pencarian dengan cara seperti ini kurang efisien karena bisa jadi sebenarnya ada simpul yang tidak perlu diperiksa karena sudah jelas tidak akan menghasilkan solusi.

Pada gambar 2.2 dapat dilihat bahwa pencarian kedalam dihentikan pada simpul nomor 3 dan diberi tanda B. Tanda B

pada gambar menunjukkan bahwa simpul tersebut dibunuh oleh *bounding function* dan dihentikan pencarian dari simpul tersebut. Hal ini dapat terjadi karena semua simpul yang ada dibawah simpul 3 tidak mengarah ke solusi, sehingga dilakukan *backtracking*, yaitu pencarian mundur 1 langkah dan pencarian dilakukan kearah simpul hasil ekspansi yang lain. Metode pencarian seperti ini dapat meningkatkan tingkat efektivitas pencarian. Karena seperti yang sudah disebutkan sebelumnya, simpul-simpul yang tidak mengarah ke solusi dan sudah dipastikan tidak mungkin menjadi solusi tidak perlu diperiksa lagi.

C. *Properti Umum Algoritma Backtracking*

Metode pencarian dengan algoritma backtracking memiliki beberapa properti umum, yaitu :

1. Solusi Persoalan
   Solusi persoalan dinyatakan dalam bentuk vektor n-tuple yang merepresentasikan solusi dari permasalahan.
2. Fungsi Pembangkit
   Fungsi pembangkit adalah fungsi yang membangkitkan nilai selanjutnya yang akan diperiksa.
3. Fungsi Pembatas
   Fungsi pembatas adalah fungsi yang akan berfungsi untuk menentukan apakah simpul yang sedang dievaluasi mengarah ke solusi atau tidak. Fungsi pembatas biasanya mengembalikan nilai boolean yang akan bernilai true apabila simpul yang sedang dievaluasi mengarah ke solusi dan mengembalikan nilai false jika tidak.

Ketiga properti umum algoritma backtracking berbeda-beda pada setiap kasus. Semuanya bentuknya bergantung pada pemodelan dan constraint pada masing-masing kasus.

D. *Pseudocode*

Pada implementasinya pada pemrograman, algoritma backtracking dapat menggunakan prinsip rekursif ataupun iteratif. Pseudocode akan menjelaskan alur algoritma pencarian solusi dengan menggunakan metode backtracking secara umum.

```
procedure RunutBalikR(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah
ditentukan nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  for setiap x[k] ∈ T(x[1], x[2], …, x[k – 1]) do
      if B(x[1], x[2], ... ,x[k]) = true then
         if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
            write(x[1], x[2], …, x[k])     { cetak solusi }
         endif
         if k < n then
            RunutBalikR(k+1)   { tentukan nilai untuk x[k+1]}
         endif
      endif
  endfor
                                                    Pemanggilan pertama kali: RunutBalikR(1)
```

Gambar 2.3 *Pseudocode* implementasi *backtracking* dengan prinsip rekursif
(Sumber: Gambar Perjalanan Pencarian Backtracking)

```
procedure RunutBalikI(input k : integer)
{Mencari semua solusi persoalan dengan metode runut-balik; skema iteratif
Masukan: k, yaitu indeks komponen vektor solusi, x[k]. Diasumsikan x[1], x[2], ..., x[k – 1] sudah ditentukan
nilainya.
Luaran: semua solusi x = (x[1], x[2], ..., x[n])
}
Algoritma:
  while k ≠ 0 do
     if terdapat nilai x[k] yang belum dicoba sedemikian sehingga x[k] ∈ T(x[1], x[2], …, x[k – 1]) and
        B(x[1], x[2], ... ,x[k]) = true then
        if (x[1], x[2], ... , x[k]) adalah lintasan dari akar ke simpul solusi then
           write(x[1], x[2], …, x[k])     { cetak solusi }
        endif
        k ← k + 1      { tentukan nilai x[k] selanjutnya }
     else
        k ← k – 1
     endif
  endwhile
                                                    Pemanggilan pertama kali: RunutBalikI(1)
```

Gambar 2.3 *Pseudocode* implementasi *backtracking* dengan prinsip iteratif
(Sumber: Gambar Perjalanan Pencarian Backtracking)

III. Analisis dan Pembahasan

A. *Penerapan Algoritma Backtracking dalam Menemukan Solusi Permainan Sudoku*

Algoritma *backtracking* pada kasus mencari solusi dari permainan sudoku diterapkan dalam menentukan angka apa yang tepat diletakkan pada sel tertentu. Program akan mencari sel-sel kosong mulai dari sel (0,0). Pencarian dilakukan dari arah kiri ke kanan, lalu atas ke bawah. Jika tidak ditemukan sel yang kosong maka *board* akan dianggap sudah merupakan solusi dari permainan. Jika ditemukan sel yang kosong maka akan diletakkan angka mulai dari 1 hingga 10 dan dilakukan pemeriksaan. Jika angka yang dimasukkan sudah terdapat pada baris, kolom, atau blok tersebut maka angka akan diganti dengan meng-*increment* angka yang dimasukkan. Pemasukan angka ini dimulai dari angka 1 hingga 9. Jika ditemukan angka yang tidak melanggar aturan maka untuk sementara angka tersebut diasumsikan benar dan pencarian sel kosong dilakukan kembali. Jika dalam jangkauan 1 hingga 10 tidak ada angka yang tidak melanggar aturan untuk diletakkan pada suatu sel kosong maka akan dilakukan *backtrack*, yaitu mundur 1 langkah dan meng-*increment* angka yang ada pada sel sebelumnya. Jika pencarian tidak menemukan angka yang tepat untuk diletakkan pada semua sel kosong maka fungsi akan mengembalikan nilai *False* dan program akan menampilkan pesan yang menyatakan bahwa tidak ada solusi yang ditemukan.

B. *Implementasi Program*

Penulis menggunakan bahasa pemrograman *python* pada implementasi di makalah ini. Alasan penggunaan bahasa python adalah karena sintaksis bahasa *python* bersahabat dan mudah dipahami. Sehingga diharapkan pembaca dapat lebih mudah memahami program yang dibuat.

Dalam pengimplementasian algoritma *backtracking* dalam menemukan solusi permainan sudoku ini, penulis membagi penulisan program menjadi 3 *files*. *File* pertama adalah *utils.py* yang berisi 3 prosedur dan 2 fungsi. Fungsi dan Prosedur yang ada pada *file* ini adalah fungsi dan prosedur yang berfungsi untuk mendukung keberjalanan program. Berikut adalah prosedur dan fungsi pada *file utils.py* :

1. Prosedur printBoard

Prosedur ini berguna untuk mencetak board yang berbentuk matrix ke terminal.

2. Prosedur sayHello

   Prosedur ini berguna untuk menampilkan pesan awal ke terminal.

3. Prosedur sayBye

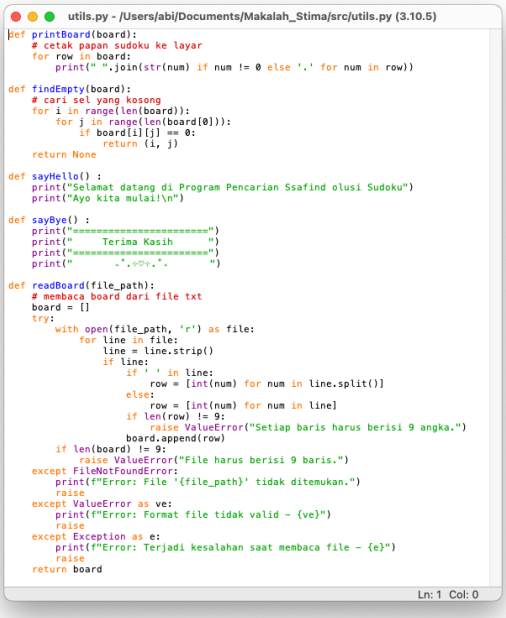
   Prosedur ini berguna untuk menampilkan pesan akhir ke terminal.

4. Fungsi findEmpty

   Fungsi ini berguna untuk mencari sel yang kosong pada *board*. Pencarian dilakukan dari *board* indeks ke (0,0) menuju ke kanan lalu ke bawah.

5. Fungsi readBoard

   Fungsi ini berguna untuk membaca *board* yang berupa matrix dalam file dengan format txt.



Gambar 3.1 *File utils.py*

(Sumber: Dokumentasi Penulis)

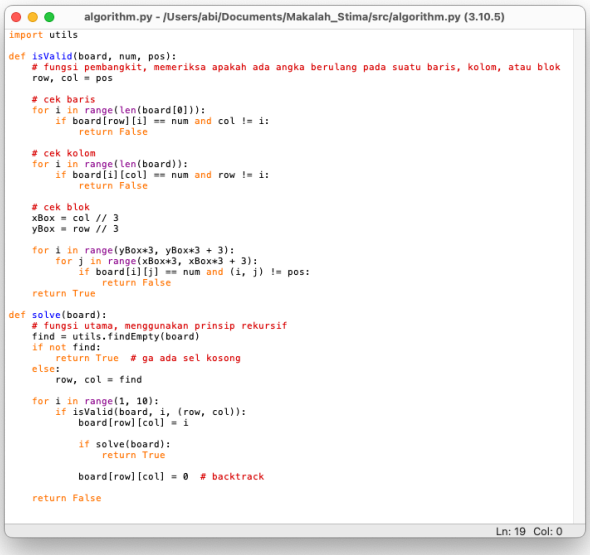
Kemudian file kedua pada implementasi program adalah *file algorithm.py*. Fungsi-fungsi pada *file* ini adalah penerapan utama dari algoritma *backtracking*. Pada *file* ini ada 2 fungsi, yaitu :

1. Fungsi isValid

   Fungsi ini sebagai fungsi pembatas pada permasalahan pencarian solusi dari permainan sudoku. Fungsi ini mengembalikan nilai Boolean yang bernilai *True* apabila angka yang diletakkan pada suatu sel kosong hanya ada 1 di baris, kolom, dan blok tersebut dan mengembalikan nilai *False* jika terjadi perulangan angka pada baris, kolom, atau blok tersebut.

2. Fungsi solve

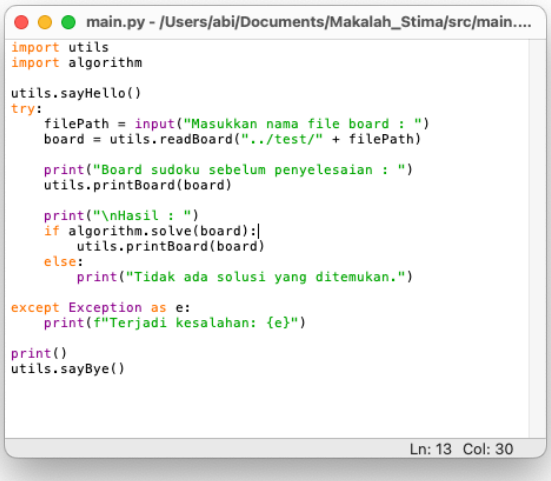
   Fungsi ini adalah algoritma utama pada kasus pencarian dengan algoritma *backtracking* ini.



Gambar 3.2 *File algorithm.py*

(Sumber: Dokumentasi Penulis)

File terakhir yang digunakan adalah *file main.py*. Jika *algorithm.py* tadi berisi fungsi-fungsi yang merupakan ide utama dari metode pencarian *backtracking* pada kasus ini, *file main.py* ini juga merupakan *file* utama. Namun, *main.py* adalah *file* utama dalam pengeksekusian program. Untuk menggunakan program ini pengguna cukup menjalankan *file main.py*.



Gambar 3.3 *File main.py*

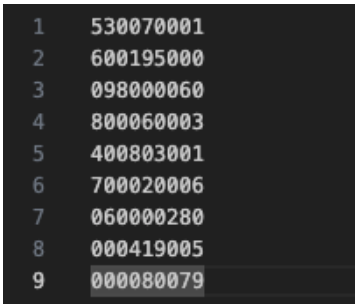
(Sumber: Dokumentasi Penulis)

IV. PENGUJIAN DAN EVALUASI

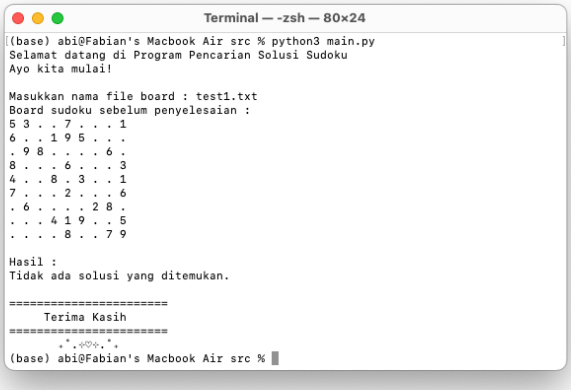
Program akan diuji dengan beberapa *testcase* sederhana. Hasil yang diharapkan dari pengujian adalah program dapat memeriksa apakah solusi dari permasalahan sudoku ada atau tidak, menemukan hasil pencarian solusi permainan sudoku jika solusi ada, memberikan pesan kesalahan kepada pengguna apabila terjadi kesalahan dalam penggunaan program. Kesalahan dalam penggunaan program bisa terjadi karena *file board* tidak ditemukan, *file board* tidak sesuai format yang diharapkan, dan terjadi kesalahan ketika membaca *file board*.

A. Pengujian

1. Pengujian Pertama

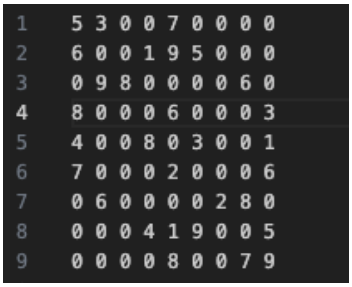


Gambar 4.1 *File test1.txt*
(Sumber: Dokumentasi Penulis)

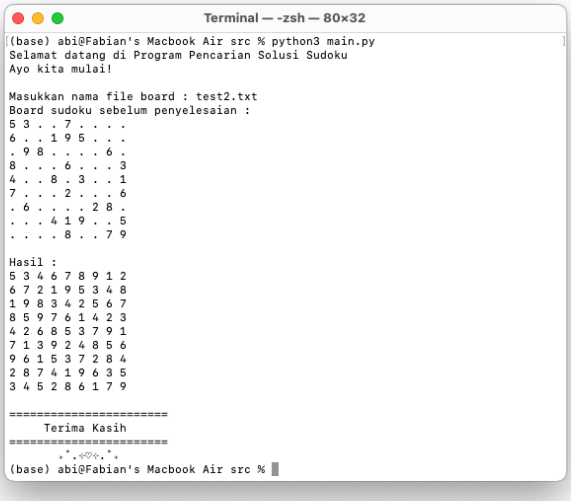


Gambar 4.2 Hasil Pengujian File *test1.txt*
(Sumber: Dokumentasi Penulis)

2. Pengujian Kedua

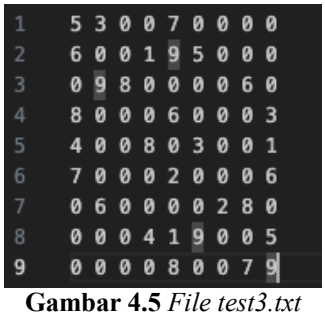


Gambar 4.3 *File test2.txt*

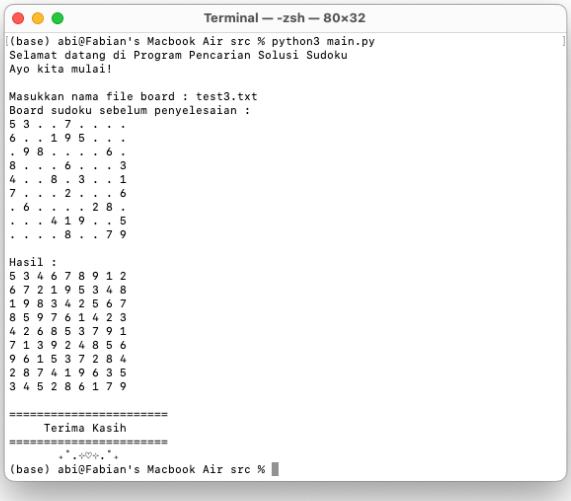


Gambar 4.4 Hasil Pengujian *File test2.txt*
(Sumber: Dokumentasi Penulis)

3. Pengujian Ketiga

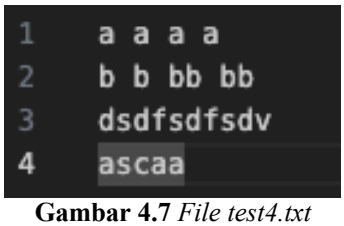


Gambar 4.5 *File test3.txt*
(Sumber: Dokumentasi Penulis)

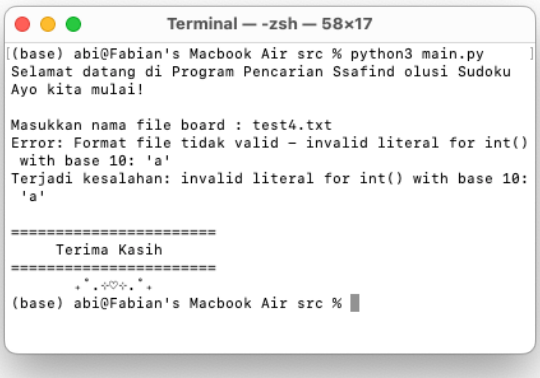


Gambar 4.6 Hasil Pengujian *File test3.txt*
(Sumber: Dokumentasi Penulis)

4. Pengujian Keempat

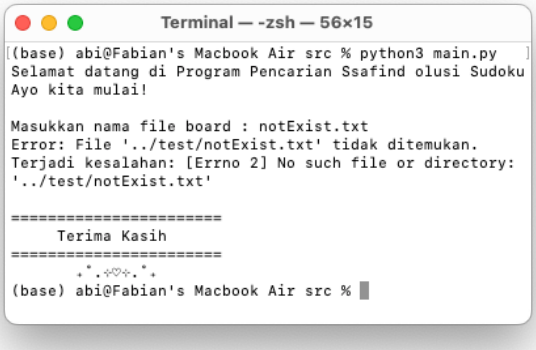


Gambar 4.7 *File test4.txt*
(Sumber: Dokumentasi Penulis)



Gambar 4.8 Hasil Pengujian *File test4.txt*
(Sumber: Dokumentasi Penulis)

5. Pengujian Kelima



Gambar 4.9 Hasil Pengujian dengan input file yang tidak *exist*
(Sumber: Dokumentasi Penulis)

B. Evaluasi

Dari hasil pengujian 5 *testcase* pada bagian A, tidak ditemukan kesalahan pada program. Pada kasus pertama, program memberikan pesan bahwa tidak ada solusi yang tepat untuk kasus tersebut. Dapat dilihat pada *board* sel ke (0,8) dan (4,9) terisi oleh angka yang sama. Hal tersebut tentu menyebabkan tidak ada solusi untuk kasus ini. Untuk kasus kedua dan ketiga, program berjalan dengan skenario normal dan menemukan solusi sebagaimana mestinya. Pada kasus keempat, program memberikan pesan kesalahan yang menyatakan bahwa *file* masukan pengguna yang merupakan tempat *board* berada tidak *exist*. Hal ini benar karena tidak ditemukan *file* tersebut memang pada folder *test*. Kemudian untuk kasus yang terakhir, program memberikan pesan kesalahan karena isi file *test5.txt* tidak sesuai format yang diharapkan.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Dari pemodelan dan pengimplementasian permasalahan pencarian solusi permainan sudoku tentu ada kesimpulan-kesimpulan yang dapat diambil. Beberapa poin penting dari hasil implementasi pada kasus ini adalah :

1. Pendekatan algoritma *backtracking* efektif digunakan untuk menemukan solusi dari permasalahan pencarian solusi permainan sudoku karena pencarian dengan algoritma *backtracking* bersifat komplit dan efisien jalur pencariannya.
2. Implementasi algoritma *backtracking* pada pemrograman dapat dilakukan dengan menggunakan prinsip rekursif sehingga memudahkan penulis program.
3. Kecepatan pencarian algoritma *backtracking* untuk menemukan solusi dari permasalahan pencarian solusi permainan sudoku bergantung pada distribusi angka awal saat permainan dimulai.
4. Python sangat cocok digunakan dalam pengimplementasian algoritma *backtracking* karena sintaksisnya bersahabat dan mudah dipahami oleh penulis dan pengguna.

B. Saran

Untuk mengembangkan dan memperbaiki hasil implementasi program, beberapa saran yang bisa dipertimbangkan adalah :

1. Demi kenyamanan pengguna dan tampilan yang lebih layak, penggunaan GUI dapat dipertimbangkan.
2. Kode dapat dioptimasi lebih baik lagi agar proses pencarian dapat diselesaikan dalam waktu yang lebih singkat.
3. Untuk mengetahui algoritma mana yang paling efektif, diperlukan adanya perbandingan antara algoritma *backtracking* den algoritma lainnya dalam pencarian solusi permainan sudoku.

VI. UCAPAN TERIMA KASIH

Penulis mengucap syukur sebesar-besarnya kepada Tuhan yang Maha Esa karena atas nikmat-Nya lah penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M. T. selaku dosen pengajar Mata Kuliah Strategi

Algoritma kelas K-01. Tidak lupa juga, penulis mengucapkan terima kasih kepada seluruh pihak yang memberi dukungan kepada penulis untuk menyelesaikan makalah ini. Semoga Tuhan membalas semua kebaikan kalian.

TAUTAN REPOSITORI

github.com/fabianradenta/Makalah_Stima

REFERENSI

[1] Munir, Rinaldi. 2021. "Algoritma Runut-Balik (Backtracking) Bagian 1". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf

[2] Munir, Rinaldi. 2021. "Algoritma Runut-Balik (Backtracking) Bagian 2". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian2.pdf

[3] Putra, Dhamma Nibbana, Sardjito, Ozzi Oriza, Lawrence, Christopher. "Penereapan dan Implementasi Algoritma *Backtracking*". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/MakalahStmik26.pdf

[4] Simonis, Helmut. 2005. "Sudoku as a Constraint Problem". https://ai.dmi.unibas.ch/_files/teaching/fs21/ai/material/ai26-simonis-cp2005ws.pdf

[5] N. A. Hasanah, L. Atikah, D. Herumurti and A. A. Yunanto, "A Comparative Study: Ant Colony Optimization Algorithm and Backtracking Algorithm for Sudoku Game," 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 2020, pp. 548-553, doi: 10.1109/iSemantic50169.2020.9234267. keywords: {Seminars;Learning systems;Ant colony optimization;Backtracking;Games;Benchmark testing;Search problems;Ant Colony Optimization (ACO);Backtracking Algorithm;Sudoku},

[6] Herimanto, Sitorus P, Zamzami E M. 2019. "An Implementation of Backtracking Algorithm for Solving a Sudoku-Puzzle Based on Android". https://iopscience.iop.org/article/10.1088/1742-6596/1566/1/012038/pdf

[7] Lina, Tirsa Ninia, Rumetna, Matheus Supriyanto. 2021. "Comparison Analysis of Breadth First Search and Depth Limited Search Algorithms in Sudoku Game". http://bcsee.org/index.php/bcsee/article/view/1146/19

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Fabian Radenta Bangun
13522105