Penerapan Algoritma Backtracking dalam Strategi Pemecahan Masalah Permainan Catur

Satriadhikara Panji Yudhistira - 13522125
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522125@std.stei.itb.ac.id

***Abstract*—Makalah ini membahas penerapan algoritma backtracking dalam strategi pemecahan masalah permainan catur, khususnya dalam konteks endgame, mate in N, dan best move. Algoritma backtracking memungkinkan eksplorasi sistematis dari berbagai kemungkinan langkah untuk menemukan solusi optimal. Hasil implementasi menunjukkan bahwa algoritma ini efektif dalam menyelesaikan berbagai masalah catur, dengan analisis kinerja yang memadai dalam hal waktu eksekusi dan efisiensi.**

Keywords—Catur, Backtracking, Sekakmat

I. PENDAHULUAN

Catur adalah permainan strategi klasik yang telah menjadi subjek penelitian dalam kecerdasan buatan dan ilmu komputer selama bertahun-tahun. Masalah dari permainan catur ini bervariasi tetapi dalam makalah ini masalah yang difokuskan yaitu dalam permainan *endgame* seperti saat sudah mencapai langkah menuju sekakmat. Algoritma *backtracking* adalah metode yang sistematis dan terstruktur untuk menemukan solusi dengan mengeksplorasi semua kemungkinan langkah secara rekursif dan mundur jika diperlukan.

II. TEORI DASAR

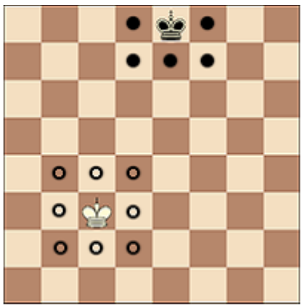
A. *Catur*

Catur adalah permainan yang dimainkan oleh dua orang, masing-masing dilengkapi dengan 16 buah catur. Satu set berwarna hitam dan set lainnya berwarna putih. Setiap pemain memiliki 8 pion, 2 benteng, 2 gajah, 2 kuda, 1 ratu, dan 1 raja [1]. Permainan ini dimainkan pada papan persegi yang disebut 'papan catur', yang terdiri dari kotak berukuran 8 x 8 dengan total 64 kotak [2]. Kotak-kotak ini bergantian antara warna terang (kotak 'putih') dan gelap (kotak 'hitam') [2].

Pada awal permainan, pion-pion ditempatkan di paling depan yang sudah ditentukan. Raja ditempatkan di tengah, untuk pemain putih di sebelah kiri gajah dan untuk pemain hitam di sebelah kanan gajah. Ratu ditempatkan di sebelah raja. Benteng di tempatkan di ujung kiri dan kanan, diikuti dengan kuda dan gajah.

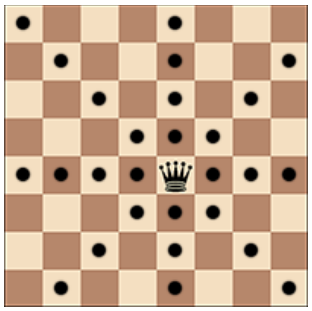
Aturan dasar gerakan catur sebagai berikut

- Tidak diperbolehkan memindahkan sebuah buah catur ke kotak yang ditempati oleh buah catur dengan warna yang sama.
- Buah catur dipindahkan ke petak kosong atau petak yang sudah ditempati oleh buah catur lawan, yang berarti menyingkirkan buah catur lawan dari papan permainan.
- Raja hanya bisa bergerak satu petak ke arah horizontal, vertikal, maupun diagonal. Raja tidak bisa melompati buah catur lain.



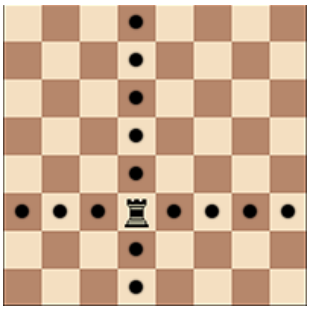
Gambar 2.1 Gerakan Raja

- Ratu (menteri) dapat bergerak ke arah horizontal, vertikal, maupun diagonal sebanyak petak. Ratu tidak bisa melompati buah catur lain.



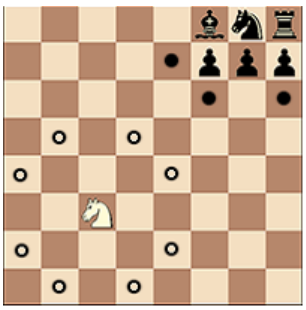
Gambar 2.2 Gerakan Ratu

- Benteng dapat bergerak ke arah horizontal dan vertikal sebanyak petak. Benteng tidak bisa melompati buah catur lain.



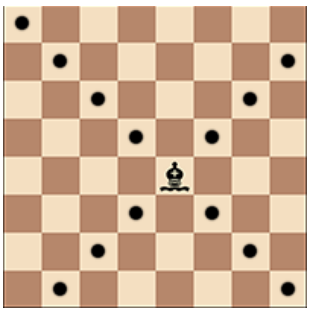
Gambar 2.3 Gerakan Benteng

- Kuda dapat bergerak melompati buah catur lain dengan lompatan berbentuk L.



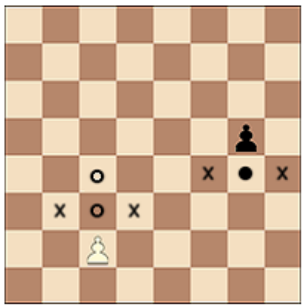
Gambar 2.4 Gerakan Kuda

- Gajah dapat bergerak ke arah diagonal sebanyak petak. Gajah tidak bisa melompati buah catur lain.



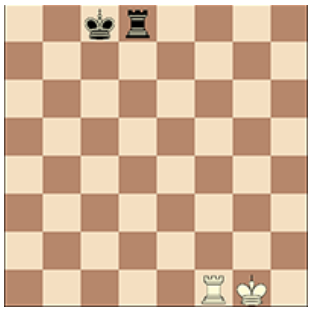
Gambar 2.5 Gerakan Gajah

- Pion dapat bergerak ke arah depan sebanyak dua petak atau satu petak jika masih berada di posisi awal, selain itu pion hanya bisa bergerak satu petak. Pion tidak bisa melompati buah catur lain dan hanya bisa memakan buah catur lain yang di depan diagonal satu petaknya.



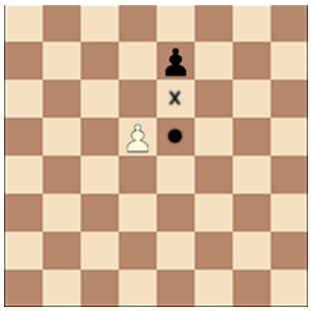
Gambar 2.6 Gerakan Pion

- Barikade adalah di mana raja dan salah satu dari dua benteng (biasanya yang berada di sisi raja atau sisi ratu) bergerak secara bersamaan. Barikade terdapat dua jenis yaitu sisi Raja dan sisi Ratu



Gambar 2.7 Gerakan Barikade

- "En passant" adalah aturan khusus dalam catur yang memungkinkan pion untuk menangkap pion lawan yang bergerak dua langkah dari kotak asalnya dan berada di sebelah pion yang menangkapnya. Aturan ini hanya berlaku jika kondisi-kondisi tertentu terpenuhi dan dilakukan segera setelah pion lawan bergerak dua langkah.



Gambar 2.8 Gerakan En passant

B. Algoritma Backtracking (Runut-Balik)

Backtracking adalah sebuah metode pemecahan masalah yang sistematis dan terstruktur untuk penyelesaian optimal maupun tidak [3]. *Backtracking* mencari solusi melalui pemilihan mundur dan percobaan berulang jika langkahnya tidak menuju ke solusi yang diinginkan.

1) Properti umum algoritma backtracking

a) Solusi Persoalan: Solusi persoalan dinyatakan sebagai vektor terdiri dari n-tuple:

$$X = (x1, x2, \ldots, xn)$$

di mana $xi \in Si$. Biasanya $S1 = S2 = \ldots = Sn$.

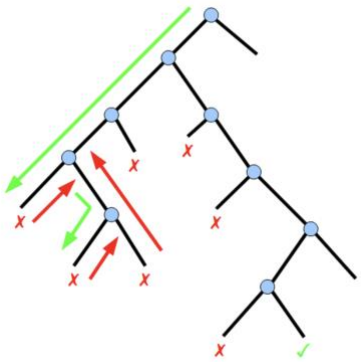
b) Fungsi Pembangkit Nilai xk: Fungsi pembangkit dinyatakan sebagai predikat T(). $T(x[1], x[2], \ldots, x[k-1])$ menghasilkan nilai untuk xk, yang merupakan bagian dari vektor solusi.

c) Fungsi Pembatas: dinyatakan sebagai predikat $B(x1, x2, \ldots, xk)$. B bernilai benar jika $(x1, x2, \ldots, xk)$ mengarah pada solusi. Mengarah pada solusi berati tidak melanggar kendala (constraints). Jika bernilai benar, maka

proses menghasilkan nilai untuk $xk + 1$ dilanjutkan; jika bernilai salah, maka $(x1, x2, \ldots, xk)$ diabaikan.

2) Prinsip Pencaran Solusi dengan Algoritma Backtracking

Proses pencarian solusi menggunakan algoritma runut-balik beroperasi dengan menghasilkan simpul-simpul status secara berurutan, membentuk lintasan dari akar hingga daun dalam struktur pohon pencarian. Aturan pembangkitan simpul mengikuti urutan pertama-dalam (DFS), yang berarti simpul anak dari simpul saat ini dieksplorasi sebelum melanjutkan ke simpul-simpul saudaranya. Simpul-simpul yang aktif dieksplorasi disebut simpul hidup, dan ketika salah satu dari mereka sedang dalam proses ekspansi, disebut simpul-E (Expand-node). Setiap kali simpul-E diekspansi, lintasan pencarian bertambah panjang. Jika lintasan yang dibangun tidak menuju ke solusi, simpul-E tersebut "dimatikan" dan berubah menjadi simpul mati (dead node), menggunakan fungsi pembatas (bounding function). Ketika sebuah simpul menjadi mati, simpul-simpul anaknya dipangkas (pruning). Jika proses pembentukan lintasan berakhir dengan simpul mati, pencarian mundur (backtrack) dilakukan ke simpul di atasnya dalam pohon pencarian, kemudian lanjutkan dengan mengeksplorasi simpul anak lainnya. Simpul yang baru dieksplorasi kemudian menjadi simpul-E yang baru. Pencarian berhenti saat mencapai simpul tujuan (goal node).



Gambar 2.8 Contoh visualisasi backtracking

III. IMPLEMENTASI

Pada implementasi ini saya menggunakan bahasa pemrograman python dengan versi 3.11.5 yang di gunakan pada komputer dengan spesifikasi Apple Silicon M2 Pro. Lingkungan yang saya pakai yaitu Visual Studio Code sebagai Text Editor dan Pip sebagai *package manager* untuk python.

Untuk mempermudah implementasi, saya menggunakan *library* python-chess untuk membaca *config* Forsyth-Edwards Notation (FEN) menjadi sebuah papan catur yang sudah mempunyai metode-metode seperti *move* yang untuk menggerakan buah catur yang valid dan apakah sudah sekakmat.

```python
import chess

fen = ""
board = chess.Board(fen)
```

Gambar 3.1 Pemakaian *Library* python-chess

A. *Mencari N langkah sekakmat*

Untuk mempermudah implementasi karena kurangnya persediaan komputer yang memenuhi spesifikasi dan mengurangi waktu yang digunakan, saya akan membatasi $N \leq 3$.

```python
def backtrack(board, depth, maximizing_player):
    if depth == 0 or board.is_game_over():
        return board.is_checkmate(), []
    best_moves = []
    if maximizing_player:
        best_score = float("-inf")
        for move in board.legal_moves:
            new_board = make_move(board, move)
            score, moves = backtrack(new_board, depth - 1, False)
            if score > best_score:
                best_score = score
                best_moves = [move] + moves
        return best_score, best_moves
    else:
        best_score = float("inf")
        for move in board.legal_moves:
            new_board = make_move(board, move)
            score, moves = backtrack(new_board, depth - 1, True)
            if score < best_score:
                best_score = score
                best_moves = [move] + moves
        return best_score, best_moves
```

Gambar 3.2 Implementasi *Backtrack*

Algoritma backtrack dalam fungsi tersebut adalah implementasi dari algoritma minimax dengan pemangkasan alpha-beta. Algoritma ini secara rekursif mengevaluasi semua kemungkinan langkah yang mungkin diambil oleh pemain dan lawan, hingga kedalaman tertentu.

1) Pengecekan Kondisi Berhenti

Fungsi dimulai dengan mengecek apakah kita telah mencapai kondisi berhenti. Ini terjadi jika kedalaman pencarian telah mencapai 0 atau permainan telah selesai (misalnya, ada checkmate). Jika salah satu kondisi ini terpenuhi, fungsi akan mengembalikan nilai evaluasi dan daftar langkah kosong.

2) Langkah Pemain Maksimalisasi

Jika pemain yang sedang bergerak ingin memaksimalkan skor, kita mulai dengan menginisialisasi best_score dengan nilai negatif tak hingga dan daftar best_moves kosong. Kita kemudian melakukan loop melalui semua langkah yang sah yang bisa diambil di papan saat ini. Untuk setiap langkah, kita membuat papan baru yang mencerminkan langkah tersebut dan kemudian melakukan pemanggilan rekursif

ke backtrack dengan kedalaman yang dikurangi 1 dan mengganti giliran ke pemain yang ingin meminimalkan. Ini mewakili langkah pemain lawan. Setelah mendapatkan skor dan langkah dari pemanggilan rekursif, kita membandingkan skor tersebut dengan best_score. Jika skor tersebut lebih tinggi dari best_score, kita memperbarui best_score dan best_moves dengan skor dan langkah tersebut. Ini dilakukan hingga semua langkah telah dievaluasi.

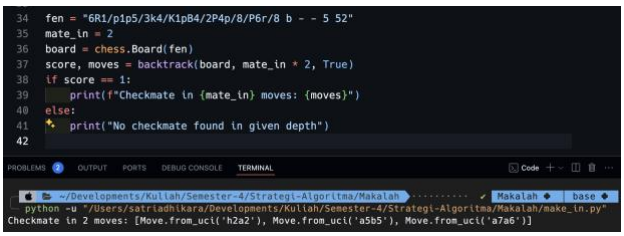
3) Langkah Pemain Minimalisasi

Langkah-langkah yang diambil dalam langkah ini mirip dengan langkah-langkah pada langkah pemain maksimalisasi, namun dengan perbedaan bahwa pemain yang sedang bergerak ingin meminimalkan skor. Kita menginisialisasi best_score dengan nilai positif tak hingga dan melakukan pemanggilan rekursif ke backtrack dengan pemain yang sedang bergerak ingin memaksimalkan skor. Kita membandingkan skor yang dihasilkan dengan best_score dan memperbarui best_score dan best_moves sesuai kebutuhan.

4) Pengebalian Hasil Terbaik

Setelah evaluasi semua kemungkinan langkah, kita mengembalikan best_score yang merupakan skor terbaik yang dapat dicapai dan best_moves yang merupakan daftar langkah yang harus diambil untuk mencapai skor tersebut.

Mari kita uji coba dalam salah satu puzzle yang disediakan oleh chess.com [4]. Dalam hasil algoritma tersebut, program mengembalikan langkah-langkah untuk menuju sekakmat dalam N langkah tersebut.



Gambar 3.3 Uji program

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Penerapan algoritma *backtracking* dapat memberikan langkah atau solusi terbaik dalam situasi tertentu dalam permasalahan dalam permainan catur. Solusi yang dihasilkan oleh algoritma tidak selalu mengembalikan solusi optimum dikarenakan kurangnya kedalaman algoritma.

B. Saran

Dalam penyusunan makalah ini, penulis sadar akan kurangnya kelengkapan data maupun hasil algoritma dan analisis yang sudah dipaparkan. Oleh karena itu, penulis menerima semua masukan yang dapat meningkatkan hasil dari penelitian ini.

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa, Allah SWT yang Maha Esa maha penyayang karena memberikan kekuatan, kesehatan, dan dukungan sepanjang perjalanan akademis ini. Dan terima kasih kepada seluruh Dosen yang saya hormati mata kuliah Strategi Algoritma ini, atas kontribusinya bimbingan, dorongan, dan dukungan yang tak tergoyahkan selama ini seluruh proses. Terima kasih yang tulus saya sampaikan kepada teman-teman saya dan keluarga atas pengertian, kesabaran, dan dorongannya selama periode yang menuntut ini. Kepercayaanmu kepadaku adalah sumber motivasi yang konstan. Dan yang terakhir, saya ingin melakukannya menyampaikan rasa terima kasih yang tulus kepada semua pihak yang telah berkontribusi hingga selesainya makalah ini. Dukungan dan bantuan Anda telah sangat berharga, dan saya benar-benar berterima kasih atas Anda kemurahan hati.

REFERENCES

[1] Kamus Besar Bahasa Indonesia (KBBI) Daring, "Catur," diakses dari https://kbbi.web.id/catur.

[2] Wayback Machine, "FIDE Handbook - FIDE Laws of Chess - January 2023," diakses dari https://web.archive.org/web20240427115216/https://handbook.fide.com/chapter/E012023.

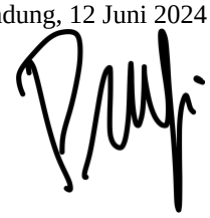
[3] Rinaldi Munir, "Algoritma Backtracking - Bagian 1," diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf.

[4] Chess.com, "Analisis Posisi," diakses dari https://www.chess.com/analysis?fen=r2qk2r%2F2p1bpQp%2Fp2p1n2%2F1p1Np3%2F2BnP3%2FP2P4%2F1P3PPP%2F2R2RK1+b+kq+-+1+14&flip=false&tab=analysis.

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



Satriadhikara Panji Yudhistira
13522125