Analisis Algoritma Iterative Deepening Depth First Search Pada Alpha Beta Perhitungan Poin Algoritma Catur

Kenneth Dave Bahana - 13521145
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail):

***Abstract*— Beradaptasi dengan pesatnya perkembangan teknologi pada jaman sekarang, akan selalu dibutuhkan teknologi yang semakin baik dan efisien. Algoritma IDDFS pada makalah ini menunjukkan efisiensi yang dapat dihasilkan dalam iterasi evaluasi untuk pohon pencarian setiap kemungkinan gerakan yang dapat dilakukan pada suatu kondisi catur beserta daftar poin yang dihasilkan dari setiap gerakan yang dibuat berdasarkan algoritma *Alpha-Beta* yang mencari max-min poin yang bisa dihasilkan pada gerakan selanjutnya.**

Keywords—IDDFS; Alpha-Beta; efisien; catur

I. PENDAHULUAN

Perkembangan teknologi pada jaman sekarang makin pesat, terutama kedatangan seketika Open AI yang kini berbagai permasalahan dapat diselesaikan dengan alat – alat berbasis *Artificial Intelligence*. Berbagai hal yang dapat dilakukan A.I. sekarang berkembang pesat seperti audio membantu pembuatan lagu, pembuatan gambar berdasarkan keinginan pengguna dan berbagai hal lainnya. Namun, salah satu pengembangan terlama berkaitan dengan A.I. adalah permainan catur. *Bot* A.I. seperti Stockfish dan AlphaZero merupakan buatan A.I. yang bisa bermain catur dan memilih Gerakan yang menurut mereka terbaik hingga sekarang belum bisa ditandingi oleh manusia untuk memenangkan satu permainan pun.

Permainan catur telah menjadi tantangan yang menarik bagi para pemain dan peneliti di bidang kecerdasan buatan. Dalam catur, kemampuan untuk menghitung poin dan mencari langkah terbaik menjadi kunci untuk mencapai kemenangan. Dalam upaya untuk mengembangkan sistem yang mampu bermain catur dengan tingkat kecerdasan yang tinggi, algoritma perhitungan poin dan pencarian langkah terbaik yang efisien menjadi sangat penting.

Salah satu pendekatan yang populer adalah menggunakan algoritma DFS untuk mencari langkah terbaik dalam pohon pergerakan. Namun, DFS memiliki keterbatasan dalam pencarian yang tidak efisien ketika kedalaman pencarian menjadi sangat dalam. Untuk mengatasi keterbatasan ini, algoritma Iterative Deepening DFS (IDDFS) digunakan, di mana pencarian dilakukan dengan memperdalam level secara bertahap.

Makalah ini bertujuan untuk menganalisis penggunaan algoritma IDDFS dalam konteks perhitungan poin dan pencarian langkah terbaik dalam catur. Dalam analisis ini, kami akan mengkaji efektivitas dan efisiensi algoritma IDDFS dalam menyelesaikan permasalahan ini, dengan membandingkannya dengan algoritma lain yang umum digunakan dalam domain ini.

Selain itu, makalah ini juga akan mengevaluasi pengaruh parameter-parameter kunci dalam algoritma IDDFS terhadap kinerjanya, seperti tingkat kedalaman pencarian, penggunaan fungsi evaluasi, dan metode penyortiran langkah-langkah yang mungkin.

Diharapkan hasil dari makalah ini dapat memberikan wawasan yang berharga dalam penerapan algoritma IDDFS dalam perhitungan poin dan pencarian langkah terbaik dalam catur, serta membuka jalan untuk pengembangan sistem kecerdasan buatan yang lebih canggih dalam memainkan permainan ini.

II. LANDASAN TEORI

A. Permainan Catur

Permainan catur adalah permainan strategi yang dimainkan antara dua pemain di atas papan catur berukuran 8x8 dengan kotak-kotak berwarna hitam dan putih. Setiap pemain memiliki 16 buah catur yang terdiri dari raja, ratu, dua benteng, dua kuda, dua gajah, dan delapan pion. Tujuan utama dalam permainan catur adalah untuk "mempersiapkan jalan" ke raja lawan dengan cara mengepung dan mengancam raja lawan sampai akhirnya mencapai kondisi "skakmat", di mana raja lawan tak dapat bergerak tanpa terancam oleh lawan.

Pergerakan setiap jenis buah catur memiliki aturan yang khas. Raja dapat bergerak ke kotak yang berdekatan, ratu

memiliki pergerakan yang sangat luas di sepanjang baris, kolom, atau diagonal, benteng bergerak maju atau mundur sepanjang baris atau kolom, gajah bergerak diagonal, dan kuda memiliki pergerakan yang unik dalam bentuk "L" dengan dua kotak ke depan dan satu kotak ke samping. Pion merupakan buah catur yang paling sederhana dalam pergerakannya, ia hanya dapat bergerak maju ke kotak kosong di depannya, kecuali pada langkah pertama yang dapat melangkah dua kotak sekaligus.

Gambar Kondisi Awal Permainan Catur
Diambil dari: chess.com

Gambar Permainan Catur berlangsung
Diambil dari: chess.com

B. Iterative Deepening Depth-First Search (IDDFS)

Iterative Deepening Depth-First Search (IDDFS) adalah algoritma penelusuran grafik yang menggabungkan manfaat dari Depth-First Search (DFS) dan Breadth-First Search (BFS). Tujuannya adalah untuk mencari jalur terpendek dalam hal jumlah sisi antara simpul awal dan simpul tujuan dalam sebuah grafik.

IDDFS bekerja dengan melakukan serangkaian penelusuran DFS dengan batasan kedalaman yang meningkat. Algoritma ini dimulai dengan batasan kedalaman 0 dan secara bertahap meningkatkannya hingga simpul tujuan ditemukan atau semua simpul telah dieksplorasi. Proses ini diulang sampai simpul tujuan tercapai atau seluruh grafik telah dilalui.

Berikut adalah cara kerja IDDFS:

- Atur batasan kedalaman awal menjadi 0.
- Lakukan penelusuran DFS yang dimulai dari simpul awal, dengan membatasi kedalaman sesuai dengan batasan kedalaman saat ini.
- Jika simpul tujuan ditemukan, hentikan penelusuran dan kembalikan jalur yang ditemukan.
- Jika simpul tujuan tidak ditemukan dan masih ada simpul yang belum dikunjungi pada batasan kedalaman saat ini, kembali ke langkah 2.
- Jika semua simpul pada batasan kedalaman saat ini telah dikunjungi dan simpul tujuan tidak ditemukan, tingkatkan batasan kedalaman sebesar 1 dan kembali ke langkah 2.
- Ulangi langkah 2-5 sampai simpul tujuan ditemukan atau seluruh grafik telah dilalui.

IDDFS memiliki beberapa keuntungan. Pertama, algoritma ini menjamin penemuan jalur terpendek jika jalur tersebut ada, karena batasan kedalaman secara bertahap ditingkatkan. Kedua, IDDFS memiliki kompleksitas ruang yang serupa dengan DFS, karena hanya perlu menyimpan simpul-simpul pada batasan kedalaman saat ini. Terakhir, IDDFS menghindari kompleksitas ruang eksponensial dari BFS dengan melakukan pendalaman secara iteratif.

Namun, IDDFS dapat menjadi kurang efisien dibandingkan algoritma lain untuk grafik yang besar atau memiliki faktor percabangan yang tinggi. IDDFS mungkin mengunjungi beberapa simpul beberapa kali, yang dapat menyebabkan kerja yang redundan. Meskipun demikian, IDDFS berguna dalam situasi di mana memori terbatas atau ketika grafik terlalu besar untuk muat secara keseluruhan di memori.

C. Algoritma Alpha-Beta

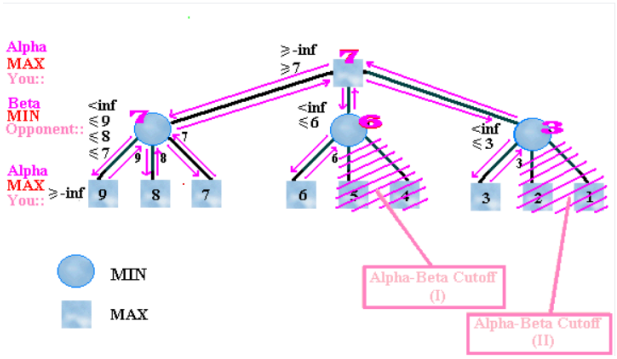
Algoritma Alpha-Beta (Alpha-Beta Pruning, Heuristik Alpha-Beta) adalah peningkatan signifikan dari algoritma pencarian minimax yang menghilangkan kebutuhan untuk mencari sebagian besar pohon permainan dengan menerapkan teknik branch-and-bound. Hal ini dilakukan tanpa risiko mengabaikan langkah yang lebih baik. Jika kita telah menemukan langkah yang cukup baik dan mencari alternatif, satu penolakan sudah cukup untuk menghindarinya. Tidak perlu mencari penolakan yang lebih kuat.

Algoritma ini menggunakan dua nilai, alpha dan beta. Alpha mewakili skor minimum yang dijamin oleh pemain yang memaksimalkan, sedangkan beta mewakili skor maksimum yang dijamin oleh pemain yang meminimalkan. Mari kita lihat contoh berikut.

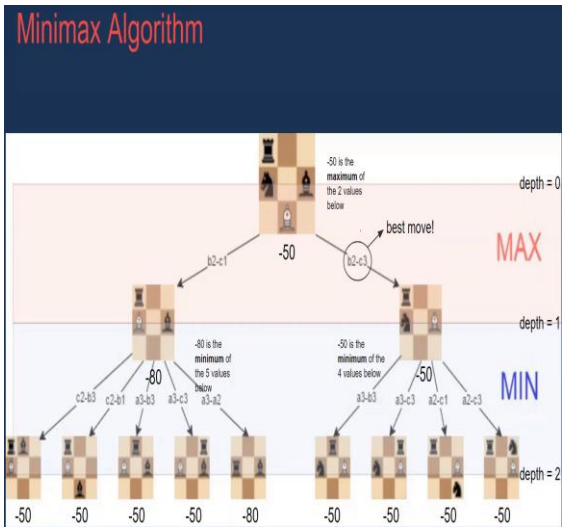
Misalkan saat ini giliran putih untuk bergerak, dan kita mencari hingga kedalaman 2 (artinya kita mempertimbangkan semua langkah putih, dan semua respons hitam terhadap setiap langkah tersebut.) Pertama, kita memilih salah satu langkah yang mungkin untuk putih - mari sebut langkah ini sebagai

Langkah yang Mungkin #1. Kita mempertimbangkan langkah ini dan setiap respons yang mungkin oleh hitam terhadap langkah ini. Setelah analisis ini, kita menentukan bahwa hasil dari melakukan Langkah yang Mungkin #1 adalah posisi genap. Kemudian, kita melanjutkan dan mempertimbangkan langkah lain yang mungkin untuk putih (Langkah yang Mungkin #2). Ketika kita mempertimbangkan langkah balasan pertama oleh hitam, kita menemukan bahwa jika memainkan ini, hitam akan memenangkan sebuah Rook! Dalam situasi ini, kita dapat dengan aman mengabaikan semua respons hitam yang mungkin terhadap Langkah yang Mungkin #2 karena kita sudah tahu bahwa Langkah yang Mungkin #1 lebih baik. Kita tidak terlalu peduli seberapa buruk Langkah yang Mungkin #2 itu. Mungkin ada respons lain yang memenangkan Ratu, tetapi tidak masalah karena kita tahu bahwa setidaknya kita bisa mencapai permainan imbang dengan memainkan Langkah yang Mungkin #1. Analisis penuh dari Langkah yang Mungkin #1 memberikan batas bawah. Kita tahu bahwa setidaknya kita bisa mencapai itu, jadi semua yang jelas lebih buruk bisa diabaikan.

Namun, situasinya menjadi lebih rumit ketika kita mencapai kedalaman pencarian 3 atau lebih, karena sekarang kedua pemain dapat membuat pilihan yang mempengaruhi pohon permainan. Sekarang kita harus mempertahankan batas bawah dan batas atas (yang disebut Alpha dan Beta). Kita mempertahankan batas bawah karena jika sebuah langkah terlalu buruk, kita tidak mempertimbangkannya. Tetapi kita juga harus mempertahankan batas atas karena jika sebuah langkah pada kedalaman 3 atau lebih mengarah pada kelanjutan yang terlalu baik, pemain lain tidak akan mengizinkannya, karena ada langkah yang lebih baik di atas pohon permainan yang bisa dia mainkan untuk menghindari situasi ini. Batas bawah pemain satu adalah batas atas pemain lainnya



Gambar Contoh Pohon Algoritma *Alpha-Beta*
Diambil dari: https://www.youtube.com/watch?v=-ivz8yJ4l4E



Gambar Contoh Pohon Algoritma *Alpha-Beta* pada catur
Diambil dari: https://www.youtube.com/watch?v=-ivz8yJ4l4E

Berikut merupakan contoh sederhana yang dimanfaatkan dari algoritma *Alpha-Beta* ini dalam pencarian nilai max dan min. Dari gambar tersebut, *max* dan *min* dihitung untuk setiap kedalaman yang menyatakan nilai terbesar dan terkecilnya. Kedua nilai ini dibutuhkan berdasarkan evaluasi untuk algoritma menyerang dan pertahanan. Nilai terbesar diambil sebagai evaluasi Kejadian gerakan terbaik yang dapat dilakukan sedangkan nilai terkecil digunakan untuk mengevaluasi gerakan dari lawan.

III. METODE

A. Metode Brute force

Algoritma *brute force* dalam bot catur atau program evaluasi adalah pendekatan yang mencoba semua kemungkinan langkah tanpa batasan kedalaman. Berikut ini adalah penjelasan langkah demi langkah dari algoritma *brute force* yang digunakan dalam bot catur atau program evaluasi:

- Mulai dari posisi awal permainan, lakukan pengulangan untuk setiap buah catur yang dimiliki oleh pemain yang sedang bergerak.
- Untuk setiap buah catur yang dipilih, hasilkan semua kemungkinan gerakan yang mungkin dilakukan oleh buah catur tersebut pada posisi saat ini.
- Evaluasi setiap gerakan secara individual dengan menggunakan fungsi evaluasi, yang akan memberikan skor pada posisi hasil setelah gerakan tersebut dilakukan.
- Simpan skor dan gerakan tersebut dalam struktur data (misalnya, dalam daftar atau pohon permainan).
- Lanjutkan ke buah catur berikutnya dan ulangi langkah 2-4.
- Setelah semua kemungkinan gerakan untuk semua buah catur telah dievaluasi, pilih gerakan dengan skor tertinggi sebagai gerakan terbaik.

- Tampilkan gerakan terbaik kepada pengguna atau gunakan dalam strategi permainan yang lebih kompleks.
- Opsional, lakukan pemangkasan (pruning) berdasarkan kriteria tertentu untuk mengurangi jumlah gerakan yang harus dievaluasi. Misalnya, menggunakan alpha-beta pruning untuk memotong cabang yang tidak perlu.

Algoritma *brute force* memiliki beberapa kelemahan dalam konteks permainan catur, terutama ketika kompleksitas permainan meningkat. Karena algoritma ini mencoba semua kemungkinan, jumlah posisi dan gerakan yang harus dievaluasi bisa sangat besar. Hal ini membuat algoritma *brute force* tidak efisien dan membutuhkan sumber daya komputasi yang signifikan.

B. Metode IDDFS

Iterative Deepening Depth First Search (IDDFS) adalah algoritma pencarian yang menggabungkan keuntungan dari Depth First Search (DFS) dan Breadth First Search (BFS). Algoritma ini sering digunakan dalam program catur atau sistem evaluasi untuk mencari gerakan terbaik dan memberikan skor pada posisi permainan. Ketika digabungkan dengan algoritma alpha-beta pruning, IDDFS dapat secara signifikan mengurangi jumlah posisi yang perlu dievaluasi.

Berikut adalah penjelasan langkah demi langkah dari algoritma IDDFS yang digunakan dalam bot catur atau program evaluasi:

- Tetapkan batas kedalaman awal menjadi 1.
- Mulai perulangan iteratif peningkatan kedalaman.
- Lakukan pencarian DFS dengan batasan kedalaman saat ini.
- Pada setiap tingkat kedalaman, hasilkan semua gerakan yang mungkin untuk posisi saat ini.
- Terapkan algoritma alpha-beta pruning untuk mengevaluasi gerakan dan menghilangkan cabang yang tidak perlu. Alpha mewakili skor terbaik yang dapat dijamin oleh pemain yang memaksimumkan, sedangkan beta mewakili skor terbaik yang dapat dijamin oleh pemain yang meminimumkan.
- Berikan skor pada posisi berdasarkan fungsi evaluasi, yang menilai kualitas posisi.
- Jika pencarian mencapai kedalaman maksimum atau posisi terminal (misalnya, skakmat), kembalikan skor.
- Jika pencarian belum mencapai kedalaman maksimum dan semua gerakan telah dievaluasi, tingkatkan batas kedalaman sebesar satu dan kembali ke langkah 3.
- Setelah perulangan iteratif selesai, pilih gerakan dengan skor tertinggi sebagai gerakan terbaik.
- Opsional, simpan informasi tambahan seperti variasi utama (urutan gerakan yang mengarah ke gerakan terbaik) untuk analisis atau tampilan lebih lanjut.

Dengan menggunakan IDDFS dalam kombinasi dengan algoritma alpha-beta pruning, bot catur dapat menjelajahi pohon permainan secara lebih efisien, menghindari evaluasi yang tidak perlu pada posisi yang tidak menjanjikan. Pendekatan iteratif peningkatan kedalaman memungkinkan bot untuk secara bertahap meningkatkan kedalaman pencarian, sehingga dapat menemukan gerakan terbaik dan mengevaluasi posisi yang lebih kompleks tanpa membutuhkan sumber daya komputasi yang berlebihan.

C. Implementasi IDDFS

```
def evaluate_position(position):
    # Fungsi evaluasi untuk memberikan
skor pada posisi permainan
    # Menghitung skor berdasarkan
penilaian material, kontrol pusat, dan
faktor lainnya
    return score

def alpha_beta_search(position, depth,
alpha, beta, maximizing_player):
    if depth == 0 or
is_game_over(position):
        return
evaluate_position(position)

    if maximizing_player:
        max_eval = float('-inf')
        for move in
generate_moves(position):
            new_position =
make_move(position, move)
            eval =
alpha_beta_search(new_position, depth -
1, alpha, beta, False)
            max_eval = max(max_eval,
eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                break
        return max_eval
    else:
        min_eval = float('inf')
        for move in
generate_moves(position):
            new_position =
make_move(position, move)
            eval =
alpha_beta_search(new_position, depth -
1, alpha, beta, True)
            min_eval = min(min_eval,
eval)
            beta = min(beta, eval)
            if beta <= alpha:
                break
        return min_eval
```

```
def iterative_deepening_search(position):
    max_depth = 4  # Kedalaman maksimum
yang ingin dicapai
    best_move = None
    for depth in range(1, max_depth + 1):
        alpha = float('-inf')
        beta = float('inf')
        eval =
alpha_beta_search(position, depth, alpha,
beta, True)
        # Simpan gerakan terbaik untuk
kedalaman saat ini
        best_move =
get_best_move(position, depth, eval)
    return best_move
```

Implentasi program ini menggunakan beberapa *library* dalam perhitungan poin yang dimiliki oleh *python*. Program sederhana ini mendefinisikan pencarian alpha beta yang juga dibentuk sebuah contoh pencarian IDDFS pada kedalaman empat, atau untuk kedua pemain pada gerakan ke-4. Hasil dari program alpha-beta ini akan mendapatkan poin evaluasi kondisi permainan sekarang dan IDDFS akan memberikan opsi pilihan gerakan terbaik untuk bot selanjutnya.

IV. ANALISIS

A. Perbandingan Brute force dan IDDFS

Berdasarkan kedua metode tersebut, dapat dibandingkan beberapa keuntungan yang dihasilkan dengan algoritma IDDFS. Untuk Algoritma IDDFS:

- IDDFS melakukan pencarian secara mendalam dengan meningkatkan batasan kedalaman secara bertahap.
- Dalam setiap iterasi, IDDFS hanya mempertimbangkan subset terbatas dari seluruh ruang pencarian, yaitu kedalaman yang ditentukan.
- IDDFS sangat efisien dalam hal penggunaan sumber daya karena membatasi pencarian pada tingkat kedalaman yang sesuai dengan kemampuan komputasi yang tersedia.
- Dengan menggunakan alpha-beta pruning, IDDFS dapat memotong lebih banyak cabang yang tidak perlu dan mengurangi jumlah posisi yang harus dievaluasi.

Brute force:

- *Brute force* mencoba semua kemungkinan langkah tanpa batasan kedalaman.
- Dalam konteks catur, *brute force* akan mempertimbangkan semua kemungkinan gerakan pada setiap posisi, tanpa memperhatikan tingkat kedalaman atau kualitas posisi tersebut.
- Pendekatan *brute force* dapat membutuhkan waktu dan sumber daya komputasi yang sangat besar, terutama saat jumlah posisi yang harus dievaluasi semakin bertambah.
- Tidak ada strategi pemotongan yang digunakan dalam *brute force*, sehingga seluruh ruang pencarian harus dijelajahi secara lengkap.

Dalam konteks algoritma untuk bot catur, penggunaan IDDFS dengan alpha-beta pruning memiliki keuntungan dibandingkan dengan pendekatan *brute force*:

- Efisiensi: IDDFS dengan alpha-beta pruning membatasi ruang pencarian pada kedalaman yang relevan, menghemat sumber daya komputasi dan waktu yang dibutuhkan. Sedangkan *brute force* akan mencoba semua kemungkinan tanpa memperhatikan tingkat kedalaman, yang bisa memakan waktu dan sumber daya yang signifikan.
- Penilaian Kualitas: IDDFS dengan alpha-beta pruning menggunakan fungsi evaluasi untuk memberikan skor pada posisi permainan, sehingga mampu membedakan posisi yang lebih baik dari yang buruk. *Brute force* hanya mempertimbangkan semua kemungkinan tanpa adanya penilaian kualitas, sehingga tidak ada cara untuk memilih gerakan terbaik.
- Scalability: Ketika jumlah posisi dan kemungkinan gerakan semakin besar, *brute force* akan menjadi tidak praktis karena kompleksitas waktu dan sumber daya yang terlalu tinggi. IDDFS dengan alpha-beta pruning, meskipun tetap membutuhkan sumber daya yang signifikan, tetapi bisa lebih scalable dan dapat diimplementasikan dalam batasan yang terjangkau.

Dengan demikian, IDDFS dengan alpha-beta pruning memberikan pendekatan yang lebih efisien dan cerdas dalam pencarian gerakan terbaik dalam permainan catur dibandingkan dengan pendekatan *brute force* yang tidak mempertimbangkan tingkat kedalaman atau kualitas posisi.

V. KESIMPULAN

Algoritma IDDFS dengan alpha-beta pruning adalah pendekatan yang lebih efisien dan cerdas dalam mencari gerakan terbaik dalam permainan catur dibandingkan dengan pendekatan brute force. IDDFS memanfaatkan peningkatan kedalaman secara bertahap, membatasi ruang pencarian, dan menggunakan alpha-beta pruning untuk memotong cabang yang tidak perlu. Dalam hal ini, IDDFS memperoleh skor dan gerakan terbaik dengan cara yang lebih efisien daripada brute force yang mencoba semua kemungkinan gerakan tanpa memperhatikan tingkat kedalaman atau kualitas posisi kurang baik dalam segi efisiensi, kualitas penilaian, serta skalabilitas untuk kepraktisan penggunaan algoritma secara keseluruhan..

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan rasa syukur kepada Tuhan yang Maha Esa karena atas anugerah-Nya, penulis dapat menyelesaikan makalah Mata Kuliah IF2211 yang berjudul "Analisis Algoritma Iterative Deepening Depth First Search Pada Alpha Beta Perhitungan Poin Algoritma Catur". Tak lupa

juga, penulis pun mengucapkan terima kasih kepada orang tua yang telah membantu penulis dalam pembuatan makalah ini. Penulis juga ingin berterima kasih kepada Dr. Ir. Rinaldi Munir, M.T.,selaku dosen pengajar Mata Kuliah IF2211, beserta dosen pengampu lainnya yang telah membimbing dalam pembelajaran perkuliahan.

REFERENSI

GeeksforGeeks. (2023). Iterative Deepening Search (IDS) or Iterative Deepening Depth First Search (IDDFS): https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/ Diakses pada 21 Mei 2023, pukul 23:13 WIB

GeeksForGeeks (2023). Minimax Algorithm in Game Theory (Alpha-Beta Pruning): https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/ Diakses pada 21 Mei 2023, pukul 09:25 WIB

365Chess (2007). Chess Database: https://www.365chess.com/opening.php Diakses pada 21 Mei 2023, pukul 14:13 WIB

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, 22 Mei 2023

Kenneth Dave Bahana, 13521145