LAPORAN HASIL PRAKTIKUM

Program Penyelesaian Masalah Pada Menara Tower Of Hanoi Menggunakan Algoritma A*

Disusun Oleh :

Dasep Rizalalludin | 20230040042

Fakultas Teknik Dan Komputer Program Studi Teknik Informatika
Universitas Nusa Putra
2024

1. PENDAHULUAN

Menara Hanoi adalah sebuah permainan matematis atau teka-teki. Teka-teki ini ditemukan Eduard Lucas, ahli matematika Perancis di tahun 1883, metode penyelesaian menara hanoi ini sangat banyak beberapa diantaranya adalah algoritma BFS, DFS dan bidirectional A+ dan yang terakhir adalah algoritma A* yang sedang saya teliti.

Menara Hanoi yang juga disebut Menara Brahma merupakan sebuah permainan matematis atau teka-teki. Teka-teki ini terdiri dari tiga tiang dan sejumlah cakram dengan ukuran yang berbeda yang bisa dimasukkan ke tiang mana saja. Permainan dimulai dengan cakram-cakram yang tertumpuk rapi berurutan berdasarkan ukurannya dalam salah satu tiang, cakram terkecil diletakkan teratas, sehingga membentuk kerucut. Tujuan dari permainan matematis ini adalah memindahkan seluruh cakram dari satu tiang ke tiang yang lain dengan beberapa aturan. Dalam permasalahan menara hanoi ini, solusi berusaha didapatkan dengan algoritma A*.

A* mendeskripsikan proses pemecahan masalah dimana seseorang harus mengambil setiap keputusan terbaik pada setiap langkahnya. Optimisasi dengan dynamic programming yang diterapkan di A* dicapai dengan memilih setiap keputusan terbaik (optimal) dalam setiap tahap. Dalam penyelesaian Menara Hanoi terdapat beberapa algoritma yang bisa digunakan, termasuk salah satunya adalah Pemrograman Dinamis (Dynamic Programming).

Langkah yang digunakan untuk menyelesaikan permasalahan di atas, saya ingin merancang suatu program yang mampu untuk melakukan penyelesaian terhadap Puzzle Tower Hanoi. Oleh karena itu, saya mengambil penelitian dengan judul **"Penyelesaian Masalah Pada Menara Tower Of Hanoi Menggunakan Algoritma A*.**

Pada penelitian ini saya mengambil rumusan masalah diantaranya adalah :

a. Bagaimana penerapan algoritma A* dalam menyelesaikan permasalahan tower of hanoi?
b. Setelah menentukan rumusan masalah penulis juga menentukan batasan masalah pada penelitian ini adalah sebagai berikut :
    a. Jumlah cakram dibatasi maksimal 5 buah dan minimal 3 buah.
    b. Setiap cakram mempunyai besar yang berbeda, semakin kebawah cakramnya maka akan semakin besar.
    c. Jumlah tiang(tower) sebanyak 3 buah.
    d. Input dari perangkat lunak berupa keadaan awal (initial state) berupa posisi awal setiap cakram dalam tumpukan tertentu.
    e. Keadaan akhir (goal state) berupa tumpukan cakram yang sama dengan ketentuan cakram awal.

2. METODELOGI

Metode merupakan suatu cara atau teknik yang sistematik untuk memecahkan suatu kasus sehingga memberikan hasil sesuai dengan yang diharapkan. Saya menggunakan studi kepustakaan (library research), yaitu menggunakan sumber-sumber melalui buku, jurnal, browsing melalui internet, serta sumber - sumber lain yang relevan untuk digunakan dalam penelitian ini. Studi kepustakaan dalam penelitian ini adalah hal-hal yang berkaitan dengan permasalahan problema keadaan dan ruang dengan penyelesaian algoritma A*.

2.1 PENGERJAAN

Python adalah bahasa pemrograman tingkat tinggi yang banyak digunakan dalam berbagai aplikasi, seperti pengembangan web, analisis data, kecerdasan buatan, otomasi, dan pemrograman ilmiah. Python pertama kali dibuat oleh Guido van Rossum dan dirilis pada tahun 1991. Python sangat populer karena sintaksnya yang sederhana, mudah dibaca, dan kemampuannya untuk menangani berbagai tugas dengan efisien. Dalam memecahkan permasalah Menara Tower of Hanoi python sangat membantu, dan untuk penyelsain masalah adalah sebagai berikut :

```python
#Program menara hanoi

piringan = int(input("Masukan jumlah piringan :"))
hanoi(piringan, 'A', 'B', 'C')
def hanoi(piringan, asal, bantuan, tujuan):
    if piringan == 1:
        print("pindahkan piringan 1 dari tiang {} ke tiang {}.".format(asal, tujuan))
        return
    hanoi(piringan -1, asal,tujuan, bantuan)
    print("pindahkan piringan {} dari tiang {} ke tiang {}.".format(piringan,asal,tujuan))
    hanoi(piringan -1, bantuan, asal, tujuan)
```

```
Masukan jumlah piringan :3
pindahkan piringan 1 dari tiang A ke tiang C.
pindahkan piringan 2 dari tiang A ke tiang B.
pindahkan piringan 1 dari tiang C ke tiang B.
pindahkan piringan 3 dari tiang A ke tiang C.
pindahkan piringan 1 dari tiang B ke tiang A.
pindahkan piringan 2 dari tiang B ke tiang C.
pindahkan piringan 1 dari tiang A ke tiang C.
```

2.2 ANALISIS KOMPLEKSITAS RUANG DAN WAKTU PADA PEMROGRAMAN TOWER OF HANOI

Algoritma Tower of Hanoi memiliki kompleksitas ruang dan waktu yang khas karena menggunakan pendekatan rekursif.

- Kompleksitas ruang
  Kompleksitas ruang dari algoritma ini bergantung pada seberapa dalam rekursi berjalan. Dalam implementasi ini, fungsi tower_of_hanoi dipanggil berulang kali hingga mencapai kasus dasar, yang berarti bahwa setiap pemanggilan rekursif harus ditumpuk di memori hingga selesai. Pada kasus terburuk, kita memiliki n tumpukan rekursi yang aktif pada saat yang sama, sehingga kompleksitas ruang adalah O(n). Kompleksitas ruang ini bersifat linear karena hanya terdapat n pemanggilan rekursif aktif pada satu waktu.
- Kompleksitas waktu
  Pada setiap pemanggilan fungsi rekursif tower of hanoi, algoritma memecah masalah menjadi dua sub-masalah dengan mengurangi jumlah piringan sebanyak satu. Setiap sub-masalah memerlukan dua langkah utama :
  1. Memindahkan n-1 piringan dari sumber ke tiang bantu.
  2. Memindahkan satu piringan dari sumber ke tujuan.
  3. Memindahkan n-1 piringan dari tiang bantu ke tujuan.

Ini menghasilkan rumus rekursif :

$T(n) = 2 \cdot T(n-1) + 1$

Keterangan :
T(n) adalah waktu yang dibutuhkan untuk memindahkan n piringan.
T(n-1) adalah waktu yang dibutuhkan untuk memindahkan n-1 piringan.

Jika dihitung secara matematis, kita bisa menemukan bahwa kompleksitas waktu algoritma ini adalah:

T(n) = 2^n - 1

Oleh karena itu, kompleksitas waktu dari algoritma Tower of Hanoi adalah O(2^n), yang merupakan kompleksitas eksponensial. Ini berarti waktu eksekusi bertambah secara eksponensial seiring dengan bertambahnya jumlah piringan.

2.3 OPTIMASI PENERAPAN TEKNIK MEMOIZATION

Memoization adalah teknik optimasi dalam pemrograman yang digunakan untuk mempercepat fungsi atau algoritma dengan cara menyimpan hasil dari pemanggilan fungsi yang telah dilakukan sebelumnya. Jika fungsi tersebut dipanggil kembali dengan parameter yang sama, memoization memungkinkan pengambilan hasil dari penyimpanan tanpa perlu melakukan komputasi ulang.

Dalam menggunakan memoization untuk menyimpan hasil setiap panggilan Hanoi (piringan, asal, bantuan, tujuan) dalam sebuah cache. Namun, karena setiap panggilan menggunakan susunan tiang yang berbeda (asal, bantuan, tujuan), hasil dari satu pemanggilan rekursif tidak dapat digunakan kembali untuk pemanggilan yang lain. Dengan kata lain, tidak ada hasil yang bisa di-*reuse* pada tingkat rekursi yang berbeda, sehingga memoization tidak memberikan manfaat dalam hal mengurangi jumlah operasi.

```
#Menyimpan hasil pemanggilan Memoization
    memo = {}

def hanoi(piringan, asal, bantuan, tujuan):
    if piringan == 1:
        return [(asal, tujuan)]

#Key untuk menyimpan hasil
    key = (piringan, asal, bantuan, tujuan)
    if key in memo:
        return memo[key]

    steps = []
    steps += hanoi(piringan - 1, asal, tujuan, bantuan)
    steps.append((asal, tujuan))
    steps += hanoi(piringan - 1, bantuan, asal, tujuan)

 #Simpan hasil dalam memo
    memo[key] = steps
    return steps
```

2.4 KOMPLEKSITAS WAKTU SEBELUM DAN SESUDAH OPTIMASI

Karena menggunaan memoization tidak akan mengubah kompleksitas waktu dari algoritma Tower of Hanoi, oleh karena itu kompleksitas waktu sebelum dan sesudah optimasi memoization akan tetap menjadi O(2^n)

Mengapa Kompleksitas Tetap O(2^n)?

- Pada setiap tingkat rekursi, kita membagi masalah menjadi dua sub-masalah dengan ukuran n−1 dan hasil dari setiap sub-masalah tidak bisa di-*reuse*.
- Oleh karena itu, setiap panggilan tetap dilakukan untuk tiap langkah baru, menghasilkan 2^n – 1 panggilan rekursif total.

2.5 BAGAIMANA TEKNIK MEMOIZATION TETAP BERFUNGSI ?

Teknik optimasi memorization tetap berfungsi karena dengan cara menyimpan hasil komputasi fungsi yang sudah dihitung sebelumnya, agar tidak perlu dihitung ulang ketika fungsi tersebut dipanggil lagi dengan parameter yang sama. Prinsip utama dari memoization adalah menghindari komputasi berulang pada masalah yang memiliki sub-masalah yang sama berulang kali (overlapping subproblems).

Berikut adalah cara kerja memoization dalam meningkatkan efisiensi:

- Pengecekan Hasil Terdahulu
  Setiap kali fungsi dipanggil, algoritma akan memeriksa apakah hasil dari komputasi untuk parameter tertentu sudah pernah dihitung sebelumnya dan disimpan. Ini biasanya dilakukan dengan memanfaatkan struktur data seperti dictionary atau hash map.
- Mengambil Hasil dari Penyimpanan (Cache)
  Jika hasil tersebut sudah ada, maka nilai yang tersimpan langsung digunakan tanpa harus melakukan komputasi ulang. Hal ini sangat menghemat waktu, terutama pada masalah yang membutuhkan banyak komputasi ulang tanpa memoization.
- Menyimpan Hasil Komputasi Baru
  Jika hasilnya belum ada (belum pernah dihitung), maka fungsi akan menghitungnya, menyimpan hasil tersebut di penyimpanan (cache), dan menggunakannya. Pada pemanggilan berikutnya, hasil ini dapat digunakan kembali.

3. KESIMPULAN

Penyelesaian masalah Tower of Hanoi memberikan wawasan penting tentang cara kerja algoritma rekursif dan karakteristik kompleksitas ruang dan waktu yang menyertainya

Setelah menyelesaikan problome Tower Of Hanoi, saya menarik kesimpulan bahwa penggunaan algoritma A* menjamin solusi pasti ditemukan dengan n Langkah.

Berikut adalah beberapa kesimpulan yang dapat diambil dari penyelesaian masalah ini :

1. Peneraan rekursi
2. Kompleksitas ruang dan waktu
3. Penerapan memorization pada program

Secara keseluruhan, penyelesaian masalah Tower of Hanoi menunjukkan bahwa meskipun beberapa masalah dapat diselesaikan secara elegan dengan rekursi, mereka mungkin tidak selalu optimal dalam hal waktu eksekusi. Hal ini menunjukkan pentingnya mengenali batasan dari metode rekursif dan pemilihan pendekatan yang sesuai berdasarkan struktur dan karakteristik masalah yang dihadapi.

4. DAFTAR PUSTAKA

https://id.scribd.com/document/478615305/Laporan-Hasil-Praktikum-Program-Menara-Hanoi
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2007-2008/Makalah/MakalahIF2153-0708-011.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2007-2008/Makalah/MakalahIF2153-0708-011.pdf