Analisis Perbandingan Efisiensi Algoritma BFS dan DFS dalam Penyelesaian Permainan Knight’s Tour

Oleh :
Kelompok 6

Jasmine Aziza
5053231003
Andina Pasha Rahmania
5053231021
Berliana Panca Dewi Nurhidayah
5053231028

Abstrak

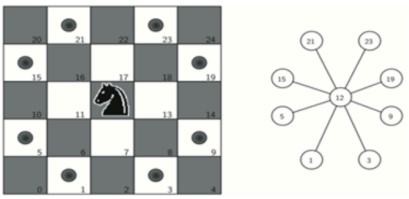
Knight's Tour adalah masalah klasik dalam yang melibatkan pergerakan ksatria pada papan catur 8x8. Tujuan dari masalah ini adalah menemukan lintasan dimana ksatria dapat mengunjungi setiap kotak pada papan hanya dalam satu kali. Penelitian ini menggunakan algoritma *Depth-First Search (DFS) dan Breadth-First Search (BFS)* untuk menyelesaikan masalah *Knight's Tour* dan menganalisis efektivitas masing-masing pendekatan. Melalui simulasi dan implementasi, makalah ini membandingkan kinerja kedua algoritma dalam menemukan solusi lintasan yang valid. Hasil menunjukkan perbedaan dalam efisiensi dan kecepatan antara algoritma DFS dan BFS dalam konteks masalah ini. Kesimpulan dari penelitian ini memberikan pemahaman lebih lanjut tentang aplikasi algoritma pencarian dalam pemecahan masalah berbasis graf.

Kata kunci : *Knight’s Tour, Breadth First Search, Depth First Search, Artificial Intelligent*

1. Pendahuluan

Knight's Tour adalah masalah dalam teori graf yang melibatkan pergerakan ksatria pada papan catur 8x8, di mana ksatria harus mengunjungi setiap kotak. Masalah ini menarik perhatian karena tantangan yang dihadirkan dalam pencarian lintasan yang valid dan aplikasinya dalam berbagai bidang komputasi. Dalam permasalahan dalam Knight Tour ini menggunakan algoritma *Depth-First Search* (DFS) dan *Breadth-First Search* (BFS) untuk memecahkan masalah *Knight's Tour* dan membandingkan efektivitas keduanya. Dengan pendekatan ini, diharapkan dapat memberikan wawasan tentang bagaimana algoritma pencarian dapat digunakan untuk menyelesaikan masalah berbasis graf.

`



Gambar 1. Knight’s Move

2. Implementasi Algoritma

2.1. Definisi Breadth-First Search (BFS)

Breadth-First Search (BFS) adalah metode pencarian yang memperluas dan memeriksa semua node pada graf secara sistematis tanpa menggunakan algoritma heuristik. BFS melakukan pencarian mendalam di seluruh graf, menemukan solusi jika ada, dan menjamin jalur terpendek. Kelebihan lainnya adalah BFS tidak terjebak dalam looping dan meskipun memerlukan ruang penyimpanan untuk antrian node, jumlah ruang tersebut bisa cukup besar. Karakteristik BFS mencari semua node pada level yang sama terlebih dahulu sebelum ke level berikutnya, lalu pencarian dilakukan dari node awal hingga mencapai kedalaman maksimum.

Algoritma BFS bekerja dengan cara:

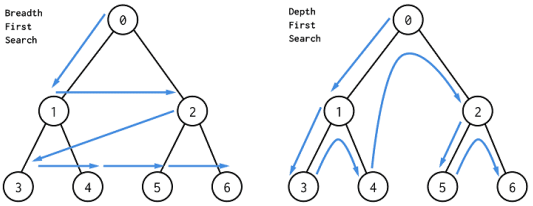
1. Mengunjungi node awal (v) saat pertama kali.
2. Mengunjungi semua node tetangga dari node v.
3. Mengunjungi node yang belum dikunjungi dan bertetangga dengan node-node yang telah dikunjungi sebelumnya.

2.2 Definisi Depth-First Search (DFS)

Depth-First Search (DFS) adalah metode pencarian sistematis yang mengembangkan jalur tunggal hingga menemukan solusi atau dead end, sebelum mengeksplorasi jalur lain. Proses ini dimulai dari node akar (v) dan mengikuti urutan yang telah ditentukan (dari kiri ke kanan atau sebaliknya). Jika mencapai dead end, DFS kembali ke node terakhir untuk memeriksa cabang yang belum dieksplorasi.

Algoritma DFS bekerja dengan langkah-langkah berikut:

1. Kunjungi node awal (v).
2. Kunjungi node tetangga (w) dari node v.
3. Ulangi proses DFS dari node w.
4. Jika semua node tetangga dari node u telah dikunjungi, kembali ke node sebelumnya untuk mengeksplorasi cabang lainnya.



Gambar 2. Pohon BFS dan DFS

Tabel 1. Perbandingan *DFS* dan *BFS* :

Aspek	**DFS**	**BFS**
Kecepatan dan Efisiensi	Lebih cepat dalam beberapa kasus; efisien memori	Umumnya lebih lambat; memerlukan lebih banyak memori
Optimalitas Solusi	Tidak menjamin solusi optimal	Menjamin solusi dengan jalur terpendek dalam graf tidak berbobot
Struktur Data yang Digunakan	Menggunakan stack (atau rekursi)	Menggunakan antrian
Penghindaran Looping	Dapat terjebak dalam looping	Menghindari looping dengan melacak node yang sudah dikunjungi
Kelebihan	Lebih efisien dalam memori, dapat menemukan solusi cepat	Menemukan solusi optimal, tidak terjebak dalam looping
Kekurangan	Tidak optimal, bisa terjebak dalam jalur tanpa solusi	Memerlukan banyak memori dan waktu untuk menjelajahi semua node pada satu level

3. Metrik yang Dibandingkan

Metrik	**BFS (Breadth-First Search)**	**DFS (Depth-First Search)**
Ukuran Papan	8x8	8x8
Timelimit (detik)	10	10
Timeout	Ya	Ya
Jumlah Node Diperluas	3.556	23.315

Ukuran Papan: Kedua algoritma diuji pada papan berukuran 8x8, yang merupakan ukuran standar untuk masalah Knight's Tour.

Timelimit (detik): Batas waktu yang diberikan untuk setiap algoritma adalah 10 detik. Ini menunjukkan bahwa algoritma harus menyelesaikan pencarian dalam waktu yang ditentukan.

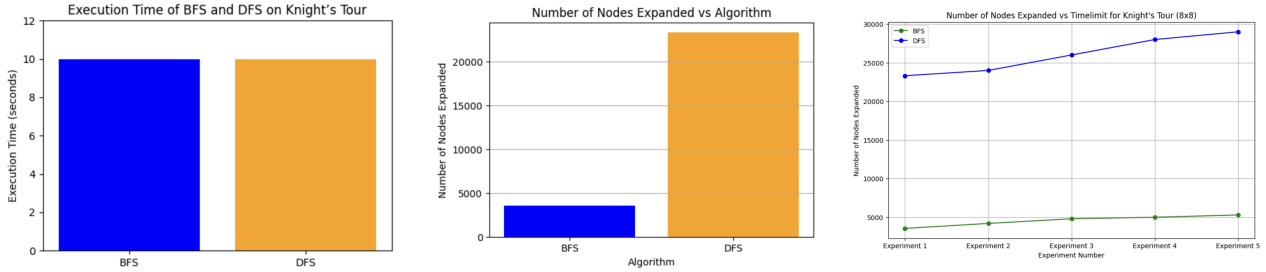
Timeout: Untuk BFS, hasilnya menunjukkan "Ya" yang berarti algoritma mungkin tidak menemukan solusi dalam batas waktu yang diberikan. Sebaliknya, untuk DFS juga menunjukkan "Ya" menandakan bahwa DFS juga dapat mengalami kondisi timeout.

Jumlah Node Diperluas: BFS: 3.556 node diperluas selama pencarian. DFS: 23.315 node diperluas. Ini menunjukkan bahwa DFS perlu memperluas lebih banyak node dibandingkan BFS untuk mencapai solusinya, yang mencerminkan perbedaan dalam cara kedua algoritma melakukan eksplorasi.

Dari tabel ini, dapat dilihat bahwa meskipun kedua algoritma memiliki ukuran papan dan batas waktu yang sama, BFS tampaknya lebih efisien dalam jumlah node yang diperluas dibandingkan DFS, meskipun keduanya menghadapi kemungkinan timeout

4. Hasil dan Pembahasan

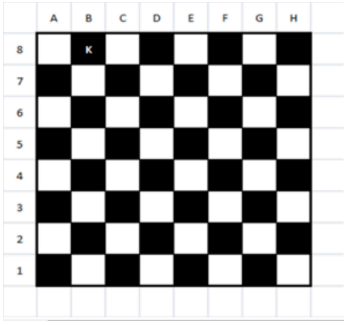
- **Jumlah Node Diperluas:**
  - **BFS** memperluas **3.556 node** dalam batas waktu 10 detik.
  - **DFS** memperluas **23.315 node**, jauh lebih banyak dibandingkan BFS.
- **Timeout:**
  - Kedua algoritma mengalami **timeout**, artinya tidak ada yang berhasil menemukan solusi dalam waktu 10 detik.



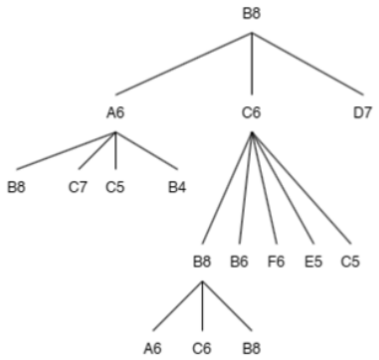
Gambar 3. Grafik Perbandingan Algoritma

4.1 Pemecahan dengan BFS untuk Knight's Tour

1. Inisialisasi: Masukkan posisi awal kuda ke dalam antrian beserta jalur yang telah diambil dan tandai kotak sebagai sudah dikunjungi.
2. Ambil Posisi Terdepan: Ambil posisi dari antrian untuk diproses.
3. Cek Tujuan: Jika jumlah langkah yang diambil sama dengan total kotak (64), kembalikan jalur sebagai solusi.
4. Hitung Langkah Valid: Dari posisi saat ini, hitung semua langkah valid kuda (8 kemungkinan gerakan).
5. Masukkan Langkah ke Antrian: Untuk setiap langkah valid yang belum dikunjungi, tandai posisi baru sebagai sudah dikunjungi dan masukkan ke dalam antrian.



Gambar 4. Posisi Bidak Kuda



Gambar 5. Gambaran Pohon sederhana BFS

4.2 Pemecahan dengan DFS untuk Knight's Tour

1. Inisialisasi: Masukkan posisi awal kuda ke dalam daftar jalur dan tandai sebagai sudah dikunjungi.
2. Cek Tujuan: Jika jumlah langkah yang diambil sama dengan total kotak (64), kembalikan jalur sebagai solusi.
3. Hitung Langkah Valid: Dari posisi saat ini, hitung semua langkah valid kuda (8 kemungkinan gerakan).
4. Rekursi untuk Langkah Valid: Untuk setiap langkah valid yang belum dikunjungi, tandai posisi baru sebagai sudah dikunjungi dan panggil DFS secara rekursif.
5. Backtracking: Jika tidak ditemukan solusi, kembalikan tanda kunjungan pada posisi sebelumnya.

5. Kesimpulan

DFS memperluas lebih banyak node dibandingkan BFS dalam batas waktu yang sama, namun tetap tidak menemukan solusi. BFS lebih lambat dalam memperluas node dibanding DFS untuk kasus ini, tetapi karena kedua algoritma mengalami timeout, ini menunjukkan bahwa untuk papan 8x8, time limit 10 detik mungkin terlalu singkat untuk menyelesaikan masalah menggunakan metode ini. Dari hasil pengujian, algoritma Breadth-First Search (BFS) memperluas 3.556 node dalam waktu 10 detik, menunjukkan efisiensi dalam pengelolaan memori dan penelusuran.

Sementara itu, algoritma Depth-First Search (DFS) memperluas 23.315 node, yang jauh lebih banyak dibandingkan BFS, mencerminkan pendekatan yang lebih dalam tetapi kurang efisien dalam hal node yang diperluas. Meskipun BFS memperluas lebih sedikit node, kedua algoritma mengalami timeout, yang menunjukkan bahwa tidak ada dari keduanya yang mampu menemukan solusi Knight's Tour dalam batas waktu yang ditentukan. Hal ini menegaskan tantangan kompleksitas masalah Knight's Tour, yang memerlukan eksplorasi semua kemungkinan langkah. Kesimpulannya, meskipun BFS lebih efisien dalam hal jumlah node yang diperluas, kedua algoritma tidak efektif dalam menemukan solusi dalam waktu yang singkat.

6. Referensi

1. Aprilia, I. (2016). Analisis dan penyelesaian permainan river crossing ultimate menggunakan algoritma BFS dan DFS. *Jurnal Teknik Elektro*, 6(2), 17-24. https://ejournal.upm.ac.id/index.php/energy/article/view/144/431
2. Fauzi, R. R., Faturrohman, M. T., & Samhari, R. (2023). Penggunaan algoritma backtracking pada permainan Knight's Tour dengan membandingkan algoritma BFS dan DFS. *Jurnal Ilmiah Teknik Informatika dan Komunikasi*, 3(2), 168-173. https://journal.sinov.id/index.php/juitik/article/view/512/458/
3. Wijaya, D. S. (2007). Perbandingan algoritma BFS dan DFS dalam pembuatan rute perjalanan objek permainan 2 dimensi. *Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung*. Tersedia di: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2006-2007/Makalah_2007/MakalahSTMIK2007-058.pdf
4. Gudari, S. P. (Tahun). *A Study on the Performance of the A-Star Algorithm with Various Heuristics in Grids and Graphs*. Department of Computing Technologies, SRM Institute of Science and Technology, Chennai, India.