LAPORAN 8-PUZZLE PROBLEM

Dosen Pengampu:

Mardhiyyah Rafrin, S.T., M.Sc.

Disusun Oleh:

Damar Kandi (231011018)

PROGRAM STUDI ILMU KOMPUTER
JURUSAN TEKNOLOGI PRODUKSI INDUSTRI
INSTITUT TEKNOLOGI BACHARUDDIN JUSUF HABIBIE

Breadth First Search

A. Deskripsi

Breadth-First Search (BFS) adalah metode pencarian yang menjelajahi semua kemungkinan solusi secara bertahap berdasarkan tingkat kedalaman sebelum melanjutkan ke tingkat berikutnya. Algoritma ini menggunakan queue (FIFO - First In, First Out) untuk menyimpan keadaan yang akan dieksplorasi. BFS dimulai dengan memasukkan keadaan awal ke dalam antrian, lalu memprosesnya dengan memeriksa apakah sudah mencapai solusi. Jika belum, algoritma menghasilkan semua kemungkinan keadaan baru, menambahkannya ke antrian jika valid dan belum pernah dikunjungi.

Proses ini berulang hingga solusi ditemukan atau semua kemungkinan telah diperiksa. BFS menjamin solusi yang ditemukan adalah solusi optimal jika setiap langkah memiliki bobot yang sama. Namun, karena harus menyimpan banyak keadaan dalam memori, algoritma ini bisa menjadi tidak efisien untuk masalah dengan ruang pencarian yang sangat besar.

B. Kelebihan

1. **Menemukan Solusi Optimal (Jalur Terpendek)**
   BFS menjamin bahwa solusi yang ditemukan adalah solusi dengan jumlah langkah minimal karena menjelajahi semua kemungkinan konfigurasi secara bertahap berdasarkan kedalaman. Ini sangat berguna untuk masalah seperti 8-Puzzle, di mana kita ingin menemukan solusi dengan jumlah gerakan sekecil mungkin.
2. **Tidak Terjebak dalam Infinite Loops**
   BFS selalu menjelajahi semua kemungkinan sebelum melanjutkan ke tingkat kedalaman berikutnya, sehingga tidak akan terjebak dalam pencarian yang berulang tanpa henti seperti yang bisa terjadi pada algoritma yang tidak mempertimbangkan eksplorasi sistematis.
3. **Sederhana dan Mudah Diimplementasikan**
   Dengan menggunakan queue dan HashSet untuk pencatatan state yang telah dikunjungi, implementasi BFS cukup sederhana dan lebih mudah dipahami dibandingkan algoritma pencarian lain seperti A* atau IDDFS (Iterative Deepening Depth-First Search).
4. **Cocok untuk Masalah dengan Ruang Keadaan Terbatas**
   BFS efektif jika jumlah kemungkinan konfigurasi (state space) relatif kecil. Dalam kasus 8-Puzzle, ruang keadaannya masih cukup terbatas dibandingkan dengan puzzle yang lebih besar, sehingga BFS masih dapat diterapkan dengan efisien.

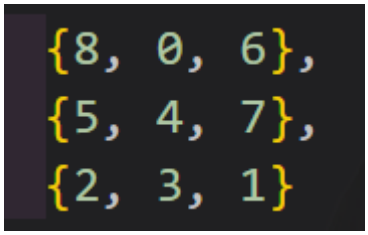
C. Kekurangan

1. **Menggunakan Memori yang Besar**
   BFS menyimpan semua node yang telah dikunjungi dalam queue dan HashSet, yang dapat menghabiskan banyak memori terutama jika ruang keadaan (state space) sangat besar. Dalam kasus terburuk, jumlah konfigurasi yang harus disimpan bisa sangat banyak sehingga dapat menyebabkan program kehabisan memori.
2. **Kurang Efisien untuk Puzzle yang Lebih Besar**
   Jika diterapkan pada versi yang lebih besar seperti 15-Puzzle atau 24-Puzzle, BFS akan mengalami kesulitan karena ledakan kombinatorial yang menyebabkan jumlah state yang dieksplorasi meningkat secara eksponensial.
3. **Tidak Memanfaatkan Heuristik**
   BFS tidak mempertimbangkan seberapa dekat suatu state dengan solusi akhir. Ini berarti BFS bisa saja menjelajahi banyak state yang kurang relevan sebelum menemukan solusi, berbeda dengan algoritma seperti A* atau Greedy Best-First Search yang menggunakan fungsi heuristik untuk mempercepat pencarian.

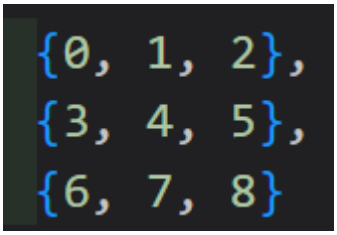
4. **Eksekusi Bisa Lambat**
   Meskipun BFS menjamin solusi optimal, waktu eksekusinya bisa lebih lama dibandingkan algoritma pencarian yang menggunakan pendekatan heuristik. Untuk kasus yang lebih kompleks, BFS bisa membutuhkan waktu yang sangat lama untuk menemukan solusi karena memeriksa setiap kemungkinan tanpa strategi pemangkasan (pruning).

8 Puzzle Problem

Initial State



Goal State



Dengan menggunakan BFS. Program mendapatkan jalur terpendek dengan 31 langkah dari Initial State ke Goal State sesuai dengan aturan yang diberikan dimana kotak kosong (0) hanya bisa bergerak keatas, kekanan, kebawah, dan kekiri didalam kotak.