

Penyelesaian Permainan Rush Hour Menggunakan Algoritma Pathfinding

Laporan Tugas Kecil 3

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma



oleh

Aramazaya

13523082

DAFTAR ISI

| | |
|--------------------------------|-----------|
| DAFTAR ISI | 2 |
| BAB I | |
| DESKRIPSI MASALAH | 3 |
| BAB II | |
| ANALISIS ALGORITMA | 5 |
| BAB III | |
| PERCOBAAN | 7 |
| III.1 Pengujian 1 | 7 |
| III.2 Pengujian 2 | 9 |
| III.3 Pengujian 3 | 9 |
| III.4 Pengujian 4 | 11 |
| III.5 Pengujian 5 | 12 |
| III.6 Analisis Hasil Percobaan | 13 |
| BAB IV | |
| KESIMPULAN | 14 |
| LAMPIRAN | 14 |

BAB I

DESKRIPSI MASALAH

Rush Hour adalah sebuah permainan puzzle logika berbasis grid yang menantang pemain untuk menggeser kendaraan di dalam sebuah kotak (biasanya berukuran 6x6) agar mobil utama (biasanya berwarna merah) dapat keluar dari kemacetan melalui pintu keluar di sisi papan. Setiap kendaraan hanya bisa bergerak lurus ke depan atau ke belakang sesuai dengan orientasinya (horizontal atau vertikal), dan tidak dapat berputar. Tujuan utama dari permainan ini adalah memindahkan mobil merah ke pintu keluar dengan jumlah langkah seminimal mungkin.

Komponen penting dari permainan Rush Hour terdiri dari:

1. Papan – Papan merupakan tempat permainan dimainkan. Papan terdiri atas cell, yaitu sebuah singular point dari papan. Sebuah piece akan menempati cell-cell pada papan. Ketika permainan dimulai, semua piece telah diletakkan di dalam papan dengan konfigurasi tertentu berupa lokasi piece dan orientasi, antara horizontal atau vertikal. Hanya primary piece yang dapat digerakkan keluar papan melewati pintu keluar. Piece yang bukan primary piece tidak dapat digerakkan keluar papan. Papan memiliki satu pintu keluar yang pasti berada di dinding papan dan sejajar dengan orientasi primary piece.
2. Piece – Piece adalah sebuah kendaraan di dalam papan. Setiap piece memiliki posisi, ukuran, dan orientasi. Orientasi sebuah piece hanya dapat berupa horizontal atau vertikal–tidak mungkin diagonal. Piece dapat memiliki beragam ukuran, yaitu jumlah cell yang ditempati oleh piece. Secara standar, variasi ukuran sebuah piece adalah 2-piece (menempati 2 cell) atau 3-piece (menempati 3 cell). Suatu piece tidak dapat digerakkan melewati/menembus piece yang lain.
3. Primary Piece – Primary piece adalah kendaraan utama yang harus dikeluarkan dari papan (biasanya berwarna merah). Hanya boleh terdapat satu primary piece.
4. Pintu Keluar – Pintu keluar adalah tempat primary piece dapat digerakkan keluar untuk menyelesaikan permainan

5. Gerakan — Gerakan yang dimaksudkan adalah pergeseran piece di dalam permainan. Piece hanya dapat bergerak/bergeser lurus sesuai orientasinya (atas-bawah jika vertikal dan kiri-kanan jika horizontal). Suatu piece tidak dapat digerakkan melewati/menembus piece yang lain.

BAB II

ANALISIS ALGORITMA

Terdapat empat algoritma yang diimplementasikan pada tugas kali ini. Algoritma-algoritma tersebut adalah algoritma A*, Greedy Best First Search, Uniform Cost Search, dan yang terakhir Iterative Deepening A* (IDA*).

Algoritma Uniform Cost Search (UCS) adalah algoritma *blind search* yang memperhatikan *Cost* atau harga dari tiap rute. Cara kerja UCS sangat mirip dengan algoritma *Dijkstra's* dimana perbedaannya berada pada target yang dicari. Cara kerja UCS adalah dengan melihat langkah yang bisa diambil dan menghitung total cost setiap langkah dari simpul awal. Pertama algoritma akan melihat simpul-simpul yang bertetangga dengan simpul awal, menghitung *cost* nya, dan memasukkannya ke sebuah *priority queue*. Kemudian algoritma mengambil elemen terdepan dari queue dan kembali menghitung *cost* dari simpul awal hingga simpul-simpul tetangganya dan memasukan ke *priority queue*. Langkah tersebut diulang hingga mencapai simpul tujuan. Algoritma ini selalu memberikan hasil paling optimal karena simpul yang dikunjungi selalu dari yang paling kecil terlebih dahulu. Tetapi, karena itu UCS cenderung lebih lambat dari algoritma-algoritma lainnya. Terlebih lagi pada kasus seperti permainan Rush Hour dimana semua sisi memiliki *cost* yang sama, algoritma ini menjadi algoritma BFS.

Algoritma Greedy Best First Search (GBFS) adalah sebuah algoritma *informed search* yang lebih mementingkan heuristik. Jika UCS menggunakan jarak dari simpul awal hingga simpul hidup, GBFS menggunakan heuristik untuk mendapatkan nilai yang kemudian akan dipakai untuk mengevaluasi jalur dari simpul hidup hingga ke simpul tujuan. Contoh heuristik yang digunakan pada tugas ini adalah *manhattan distance* dan *blocked car heuristic*. GBFS hanya memperhitungkan nilai heuristik sehingga dapat terjebak pada *local minima*. Cara kerja GBFS mirip dengan UCS hanya saja fungsi evaluasi diganti dengan fungsi heuristik. Oleh karena itu, GBFS tidak menjamin hasil yang paling optimal. Algoritma GBFS mengikuti heuristik tanpa *backtracking* sehingga jika heuristiknya salah, GBFS akan memakai jalur yang kurang optimal.

Algoritma A* adalah algoritma *informed search* yang merupakan pencampuran dari GBFS dan UCS. Dimana GBFS hanya mementingkan heuristik dan UCS hanya mementingkan *path-cost*, A* mencampurkan kedua algoritma tersebut sehingga dapat menciptakan rute yang secara teoritis paling optimal. Algoritma A* memakai fungsi evaluasi $f(n) = g(n) + h(n)$ dimana $g(n)$ adalah *cost* hingga ke simpul n dan $h(n)$ adalah nilai evaluasi heuristik pada simpul n (jarak optimal ke target dari n). Cara kerja A* mirip dengan UCS dan GBFS hanya saja fungsi evaluasi diubah. A* pasti memberikan solusi paling optimal jika heuristiknya *admissible*. Oleh karena itu, pada kasus Rush Hour, A* merupakan algoritma yang paling efisien karena secara teoritis, fungsi heuristik yang tidak pernah 0 menyebabkan $f(n)$ A* selalu lebih besar dari UCS.





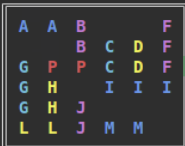
Heuristik dapat dianggap *admissible* jika tiap nilai evaluasinya tidak melebihi jalur asli menuju simpul tujuan. Sebagai contoh pakailah heuristik manhattan distance dan blocked car yang dipakai pada tugas kali ini. Manhattan distance pasti memberikan angka yang lebih kecil dari jumlah langkah aslinya karena heuristik tersebut menghitung nilai dari mobil primer hingga ke tujuan. Dikarenakan sisi yang bernilai sama, tidak mungkin manhattan distance memberikan heuristik yang lebih besar dari nilai aslinya. Begitu juga dengan heuristik blocked car yang mencampurkan manhattan distance dengan banyaknya mobil yang memblokir mobil primer untuk keluar.

Terakhir algoritma IDA* . Algoritma ini mencampurkan algoritma Iterative Deepening Search dengan algoritma A*. Perbedaan IDA* dengan IDS adalah pada *Threshold* nya. Jika IDS menggunakan batas kedalaman, IDA* menggunakan batas $f(n)$. Algoritma pertama melakukan IDS hingga mencapai simpul yang $f(n)$ nya melebihi *threshold* awal. Ketika semua simpul yang $f(n)$ nya dibawah *threshold*, algoritma mengganti *threshold* menjadi $f(n)$ terkecil yang melewati *threshold* sebelumnya dan kembali melakukan search. Hal ini diulang-ulang hingga mencapai simpul tujuan. Kelebihan IDA* adalah pada efisiensi memorinya dimana memori yang dipakai lebih sedikit dari A*. Namun, IDA* cenderung lebih lambat karena algoritma ini kembali menelusuri simpul-simpul yang telah ditelusuri berulang-ulang.

BAB III

PERCOBAAN

III.1 Pengujian 1

| Input | Output |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>A*:</p>  <p>Solution found in 6 Moves. States explored: 7 Time taken: 0.0202147 seconds. Initial state:</p>  |
| | <p>GBFS:</p>  <p>Solution found in 6 Moves. States explored: 7 Time taken: 0.0175697 seconds. Initial state:</p>  |

UCS:




Result

Solution found in 5
Moves.
States explored: 256
Time taken: 0.435329 seconds.
Initial state:

| | | | | | |
|---|---|---|---|---|---|
| A | A | B | | F | |
| | | B | C | D | F |
| G | P | P | C | D | F |
| G | H | | I | I | I |
| G | H | J | | | |
| L | L | J | M | M | |

The UCS search result visualization shows a 6x6 grid of letters. The letters are colored: A (blue), B (purple), C (green), D (yellow), F (pink), G (light blue), H (orange), I (light green), J (light purple), L (light blue), M (light blue), and P (red). The grid is enclosed in a double border. Above the grid, the word "Result" is written in a stylized, outlined font. Below the grid, the search statistics are listed: "Solution found in 5 Moves.", "States explored: 256", "Time taken: 0.435329 seconds.", and "Initial state:".

IDA*:




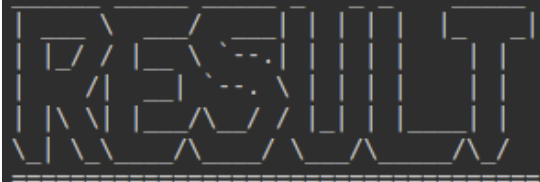

Result

Solution found in 11
Moves.
States explored: 320
Time taken: 0.0700923 seconds.
Initial state:




| | | | | | |
|---|---|---|---|---|---|
| A | A | B | | F | |
| | | B | C | D | F |
| G | P | P | C | D | F |
| G | H | | I | I | I |
| G | H | J | | | |
| L | L | J | M | M | |

The IDA* search result visualization shows a 6x6 grid of letters, identical to the UCS result. The letters are colored: A (blue), B (purple), C (green), D (yellow), F (pink), G (light blue), H (orange), I (light green), J (light purple), L (light blue), M (light blue), and P (red). The grid is enclosed in a double border. Above the grid, the word "Result" is written in a stylized, outlined font. Below the grid, the search statistics are listed: "Solution found in 11 Moves.", "States explored: 320", "Time taken: 0.0700923 seconds.", and "Initial state:".

III.2 Pengujian 2

| Input | Output |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>Semua algoritma:</p>  <p>Solution found in 0 Moves. States explored: 1 Time taken: 1.6588e-05 seconds. Initial state:</p>  |

III.3 Pengujian 3

| Input | Output |
|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>A*:</p>  <p>Solution found in 4 Moves. States explored: 237 Time taken: 0.442816 seconds. Initial state:</p>  |

GBFS:




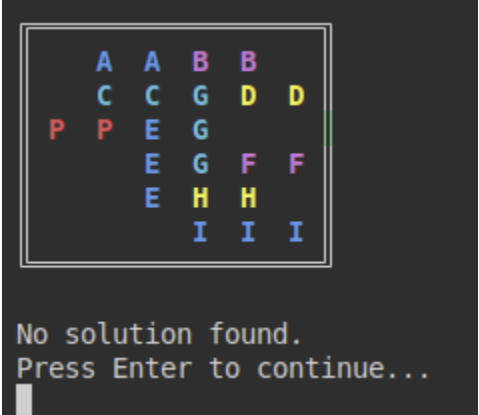
UCS:



IDA*:







III.4 Pengujian 4

| Input | Output |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
|  |  <p>No solution found. Press Enter to continue...</p> |

Untuk semua algoritma

III.5 Pengujian 5

| Input | Output |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>A*:</p>  <p>Solution found in 51 Moves. States explored: 4336 Time taken: 6.44979 seconds. Initial state:</p>  |
| | <p>GBFS:</p>  <p>Solution found in 214 Moves. States explored: 8696 Time taken: 11.3672 seconds. Initial state:</p>  |

UCS:



```
RESULT
=====
Solution found in 50
Moves.
States explored: 3582
Time taken: 5.33213 seconds.
Initial state:
  A B B B
  A C
P P E C D
F F E I I D
G H L L D
G H J J
```

IDA*:
gagal

III.6 Analisis Hasil Percobaan

Didapatkan analisa algoritma sebagai berikut:

- Penggunaan heuristik blocked car lebih efisien dari manhattan distance.
- Penggunaan UCS selalu memberikan hasil optimal dengan A* dekat di belakangnya
- Penggunaan GBFS tidak selalu memberikan hasil yang mendekati optimal dan waktu yang kurang optimal juga
- Penggunaan UCS memiliki waktu yang terbilang stabil dengan A* dan IDA* terkadang sangat cepat dibanding UCS namun terkadang lebih lambat.
- Penggunaan IDA* memiliki state_explored yang paling besar.

BAB IV

KESIMPULAN

Didapatkan bahwa metode A* dan UCS memiliki hasil paling optimal dengan A* memiliki waktu yang sedikit lebih baik dari UCS. Algoritma GBFS dapat bersaing dengan kedua algoritma tersebut namun hanya pada beberapa bagian saja. Algoritma IDA* memiliki penggunaan waktu paling buruk karena mengulangi banyak *state* hingga test terakhir tidak dapat diselesaikan dalam jangka waktu yang normal.

LAMPIRAN

1. Repositori Github

https://github.com/Aramazaya/Tucil3_13523082

| Poin | Ya | Tidak |
|------------------------------------------------------------------------------------------------------------------------|----|-------|
| 1. Program berhasil dikompilasi tanpa kesalahan | ✓ | |
| 2. Program berhasil dijalankan | ✓ | |
| 3. Solusi yang diberikan program benar dan mematuhi aturan permainan | ✓ | |
| 4. Program dapat membaca masukan berkas .txt dan menyimpan solusi berupa print board tahap per tahap dalam berkas .txt | ✓ | |
| 5. [Bonus] Implementasi algoritma pathfinding alternatif | ✓ | |
| 6. [Bonus] Implementasi 2 atau lebih heuristik alternatif | ✓ | |
| 7. [Bonus] Program memiliki GUI | | ✓ |
| 8. Program dan laporan dibuat (kelompok) sendiri | ✓ | |