PENERAPAN ALGORITMA *BREADTH FIRSH SEARCH* (BFS) DALAM PEMECAHAN PUZZLE RUSH HOUR

Larissa Rena Mansura

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
E-mail: if15087@students.if.itb.ac.id

ABSTRAK

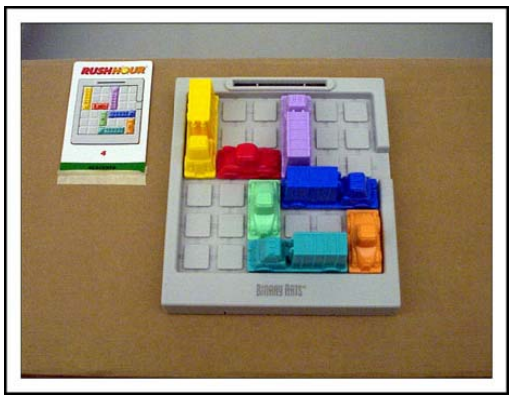
Rush Hour® adalah salah satu jenis puzzle pengasah logika yang terdiri dari balok-balok berbentuk kendaraan yang dapat digeser (*sliding block puzzle*). Objektif dari permainan Rush Hour yaitu mengeluarkan mobil utama dari grid dengan cara menggeser-geser kendaraan. Agak mirip dengan cara memainkan *Klotski* (Finlandia) atau *Hua Rong Dao* (Cina), namun uniknya Rush Hour hanya memperbolehkan kendaraan untuk digerakkan ke depan atau ke belakang sesuai arah peletakan kendaraan. Memainkan Rush Hour membutuhkan pola pikir sekuensial dengan tingkat kesulitan yang tinggi. Bahkan pemecahan permainan ini pada program komputer pun melibatkan komputasi yang kompleks. Dalam makalah ini, penulis mencoba menjelaskan cara memecahkan puzzle Rush Hour yaitu dengan menggunakan algoritma *Breadth First Search*(BFS).

Kata kunci: Breadth First Search, Rush Hour, Puzzle.

1. PENDAHULUAN

Rush Hour® adalah salah satu jenis puzzle pengasah logika yang terdiri dari balok-balok yang dapat digeser (*sliding block puzzle*). Permainan ini pertama kali diciptakan oleh Nob Yoshigahara dengan nama "Tokyo Parking Lot" di akhir tahun 1970-an. Kemudian pada tahun 1996, ThinkFun® (Binary Arts) mulai memproduksi Rush Hour secara massal untuk diperjualbelikan di Amerika Serikat.

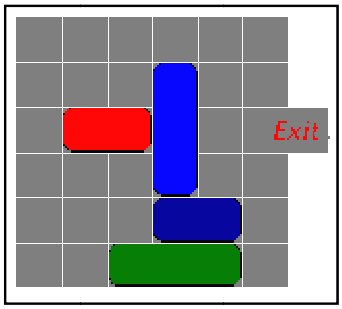
Satu set Rush Hour terdiri dari sebuah grid plastik berukuran 6x6, satu dek kartu permainan, dan miniatur kendaraan dengan rincian sebagai berikut : satu mobil utama berwarna merah, sebelas mobil biasa, dan empat truk. Mobil menempati dua grid, sedangkan truk memerlukan tiga grid. Peletakan kendaraan dilakukan sesuai dengan petunjuk konfigurasi yang terdapat pada masing-masing kartu permainan. Suatu konfigurasi tidak selalu memuat semua kendaraan yang tersedia, namun mobil utama harus selalu diletakkan pada baris tempat pintu keluar berada.



Gambar 1. Konfigurasi Rush Hour® yang siap dimainkan

Objektif dari permainan Rush Hour yaitu mengeluarkan mobil utama dari grid dengan cara menggeser-geser kendaraan. Agak mirip dengan cara memainkan *Klotski* (Finlandia) atau *Hua Rong Dao* (Cina), namun uniknya Rush Hour hanya memperbolehkan kendaraan untuk digerakkan ke depan atau ke belakang sesuai arah peletakan kendaraan.

Kini Rush Hour dapat dimainkan dalam bentuk program komputer. Banyak situs di internet yang menyediakan permainan semacam Rush Hour untuk dimainkan secara *online* maupun berupa *freeware* yang dapat diunduh.



Gambar 2. Aplikasi semacam Rush Hour yang lebih sederhana dalam bentuk program komputer

2. DESKRIPSI MASALAH

Analisis yang dilakukan oleh Gary W. Flake dan Eric B. Baum dari *NEC Research Institute*, Amerika Serikat, membuktikan bahwa permainan Rush Hour memiliki tingkat kesulitan setara dengan permainan Othello. Walaupun cara memainkannya relatif simpel, pemecahan puzzle Rush Hour pada program komputer melibatkan komputasi yang rumit. Salah satu cara yang populer untuk memecahkan puzzle Rush Hour yaitu dengan menggunakan algoritma *Breadth First Search* (BFS).

Untuk mempermudah penyelesaian masalah, grid Rush Hour digambarkan sebagai tabel yang dinomori dari 0 hingga 35, seperti terlihat pada Gambar 3.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

Gambar 3. Ilustrasi grid Rush Hour

Himpunan kendaraan dinyatakan sebagai S dan |S| adalah banyaknya kendaraan. Dengan metode BFS, permasalahan Rush Hour dipandang sebagai sebuah graf G = (V,E) yang memiliki ketentuan sebagai berikut:

- simpul v merepresentasikan status permainan
- sisi e=(v1,v2) merepresentasikan langkah yang diambil untuk mengubah status v1 menjadi v2
- status permainan didefinisikan oleh himpunan {L1,L2,…,L|S|}
- Li, I ϵ {1,2,…,|S|} merupakan tuple dengan tiga elemen: <tipe (mobil biasa atau truk),
  orientasi (horizontal atau vertikal),
  posisi (0..35)>
- Mobil utama dinyatakan dengan L1

Sebagai contoh, status awal untuk level Beginner-4 adalah:

{<mobil,h,13>, <truk,v,0>, <truk,v,15>, <mobil,v,26>, <truk,h,32>, <mobil,v,35>, <truk,h,21>}

0	1	2	3	4	5	
6	7	8	9	10	11	
12	13	14	15	16	17	exit
18	19	20	21	22	23	
24	25	26	27	28	29	
30	31	32	33	34	35	

Gambar 4. Ilustrasi status awal grid Rush Hour level Beginner-4

Sementara status menang yang harus dicapai yaitu: {<mobil,h,13>, <truk,v,0>, <truk,v,15>, <mobil,v,26>, <truk,h,32>, <mobil,v,35>, <truk,h,21>}

0	1	2	3	4	5	
6	7	8	9	10	11	
12	13	14	15	16	17	exit
18	19	20	21	22	23	
24	25	26	27	28	29	
30	31	32	33	34	35	

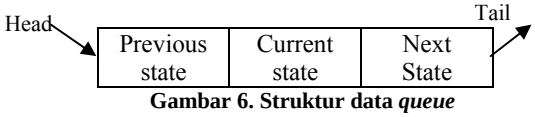
Gambar 5. Ilustrasi status menang grid Rush Hour level Beginner-4

3. PENYELESAIAN MASALAH

Tujuan dari aplikasi BFS pada pemecahan puzzle Rush Hour yaitu mencapai status akhir dengan lintasan paling efektif. Namun penggunaan BFS biasa tentunya akan memberatkan memori komputer, karena semua status yang mungkin harus disimpan. Karena itu banyaknya status yang disimpan dibatasi menjadi tiga saja, yaitu:

- *previous state*, status sebelumnya
- *current state*, status sekarang
- *next state*, status berikutnya

Untuk itu dibuatlah struktur data *queue* (antrian) yang menyimpan ketiga status di atas.



Gambar 6. Struktur data *queue*

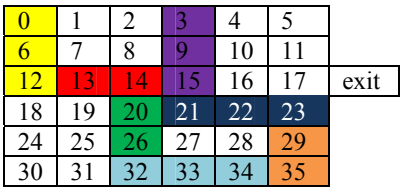
Ketika status baru dibangkitkan, status tersebut ditambahkan ke bagian *tail*. Kemudian status lama di *head* dihapus. Operasi ini ditunjukkan dengan *pseudo code* berikut:

```
procedure updateStates()
     previousStates = currentStates;
     currentStates = nextStates;
     nextStates = null;
```

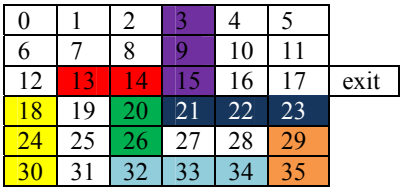
Sementara algoritma BFS secara keseluruhan dapat dilihat pada *pseudo code* berikut:

```
Procedure solve(input Q:queue, output solution)
  Q.push(initialState)
  Q.updateStates()
  while (Q tidak kosong)
    for (tiap status x pada current state)do
       for (tiap status y tetangga x) do
          if (y adalah status menang)then
             return solution
          if (y bukan elemen Q)then
             Q.push(y)
    Q.updateStates()
```

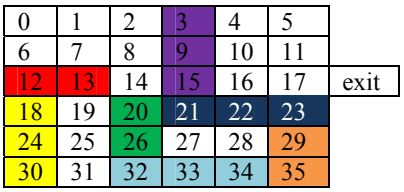
Berikut adalah ilustrasi perpindahan yang dilakukan pada permainan Rush Hour dari status awal seperti pada Gambar 4 menjadi status menang seperti pada Gambar 5



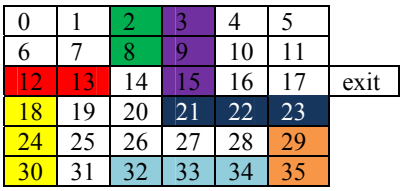
Status awal



Langkah 1



Langkah 2



Langkah 3

Langkah 4

Langkah 5

Langkah 6

Langkah 7

Langkah 7

Status menang

Kompleksitas algoritma ini dapat ditentukan dengan mengasumsikan g(k) sebagai waktu yang dibutuhkan untuk melakukan pengecekan status, dengan k adalah banyaknya langkah yang diambil. Karena algoritma ini menggunakan BFS, maka kompleksitasnya adalah $O(b^d * g(k))$.

4. KESIMPULAN

Permainan Rush Hour merupakan permainan yang menarik sekaligus dapat melatih logika. Tingkat kerumitannya dapat menjadi tantangan tersendiri bagi kita dalam memainkannya, maupun bagi program komputer dalam mencari solusinya. Algoritma Breadth First Search (BFS) mreupakan algoritma yang cukup mangkus dalam memecahkan puzzle Rush Hour.

REFERENSI

[1] Flake, G.W., dan E.B. Baum, "*Rush Hour is PSPACE-complete, or 'Why you should generously tip parking lot attendants'*", *Theoretical Computer Science*, Vol. 270, 2002, 895-911.

[2] Munir, Rinaldi, "Diktat Kuliah IF2251 Strategi Algoritmik", Penerbit ITB, 2007.

[3] Peterson, Ivars, "*Logic in the Blocks : simple puzzles can give computers an unexpectedly strenuous workout*", *Science News*,Vol. 162, No. 7, 2002, 106.

[4] Stamp, Mark, "*Rush Hour and Dijkstra's Algorithm*", *Graph Theory Notes of New York*, Vol. XL, 2001, 23-30.