PENERAPAN ALGORITMA BFS PADA CHINESE SLIDE BLOCK PUZZLE (KLOTSKI)

Ibnu Sina Wardy - 13505035

Teknik Informatika Institut Teknologi Bandung
Jl. Ganeca 10 Bandung
Email : if15035@students.if.itb.ac.id

ABSTRAK

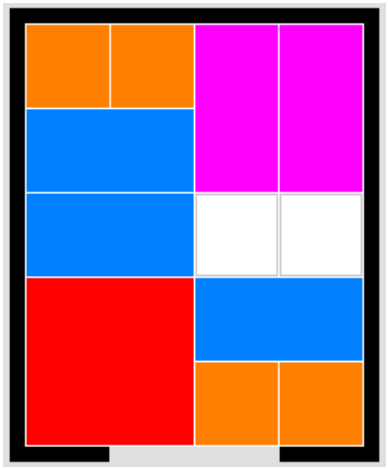
Dalam makalah ini, saya akan membahas algoritma BFS untuk mencari solusi dalam "Chinese Slide Block Puzzle". "Chinese Slide Block Puzzle" atau yang bernama lain Klotski adalah suatu *puzzle* yang berasal dari Cina. *Puzzle* ini bertujuan untuk mengeluarkan blok bujursangkar besar dari sebuah kotak besar yang mempunyai ruang kosong terbatas. Cara mengeluarkannya adalah dengan menggerakkan blok-blok lain pada ruang kosong sehingga blok bujursangkar yang terbesar dapat menuju pintu keluar.
Banyak sekali algoritma yang digunakan unuk menyelesaikan *puzzle* ini. Salah satu cara pencarian solusi persoalan ini adalah dengan mengguanakan algoritma traversal yang dapat direpresentasikan dalam graf, contohnya dengan algoritma pencarian melebar atau *Breadth First Search* (BFS).
Algoritma BFS akan mengunjungi kandidat solusi ke semua simpul yang bertetangga terlebih dahulu, tanpa melakukan pencarian ke dalam pada simpul anak pertama.
Untuk menyelesaikan *Chinese Slide Block Puzzle*, algoritma BFS merupakan salah satu cara untuk menyelesaikan *puzzle* tersebut dengan cepat dan efisien.
Algoritma BFS yang digunakan memanfaatkan *queue* dalam menyimpan kandidat solusi dari semua anak terdekat. Penggunaan *queue* ini dapat mengefisiensikan memori dalam menyimpan kandidat solusi.

Kata kunci: *Breadth first search*, BFS, klotski, *Chinese Slide Block Puzzle*, graf.

1. PENDAHULUAN

Chinese Slide Block Puzzle adalah suatu puzzle yang berasal dari sejarah Cina. Pada suatu masa, dimana tiga dinasti kerajaan di Cina sedang berkuasa, CaoCao adalah raja yang paling kuat. Dia menginvasi dua kerajaan lainnya dengan sebuah pasukan yang sangat hebat. Tetapi dalam perang ChiBi, CaoCao dapat dikalahkan oleh kedua kerajaan lainnya karena merke bersatu. Pada saat CaoCao melarikan diri, ia bertemu dengan Jenderal Guan Yu. Karena CaoCao pernah berbuat baik padanya, maka Jenderal Guan Yu mengijinkan CaoCao untuk pergi, tak peduli atas hukuman yang akan diberikan pada Guan Yu nanti. Tujuan utama kita yaitu untuk membantu Guan Yu memerintah pasukannya untuk mencari jalan keluar bagi CaoCao.[1]

Tujuan utama dalam permainan Klotski adalah mengeluarkan sebuah blok bujursangkar besar (CaoCao) dari sebuah kotak. Kotak ini mempunyai ruang kosong terbatas yang akan dipakai untuk menggeser blok-blok agar blok bujursangkar besar dapat keluar. Blok yang dipindahkan harus cukup pada ruang kosong terdekat yang tesedia.



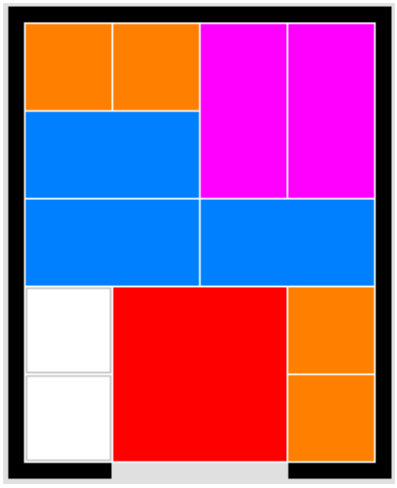
Gambar 1. Contoh soal Klotski

Pada gambar contoh soal di atas, blok bujursangkar paling besar (merah) adalah blok yang harus dikeluarkan dari kotak. Pintu keluar adalah bagian bawah tengah kotak. Blok-blok dapat digeser ke ruang kosong yang berwarna putih. Dalam soal ini, terdapat dua ruang kosong, yaitu di bagian kanan yang dapat digunakan untuk ruang geser blok-blok lain.

Solusi didapatkan jika blok bujursangkar besar sudah berada di pintu. Jumlah solusi dapat lebih dari satu selama

blok bujursangkar besar dapat menuju pintu. Jika struktur blok berbeda, maka solusi dianggap berbeda.

Contoh solusi pada Klotski dapat dilihat pada Gambar 2. Pada gambar tersebut, terlihat jelas bahwa blok bujursangkar besar berada di pintu dan siap untuk dikeluarkan dari kotak.

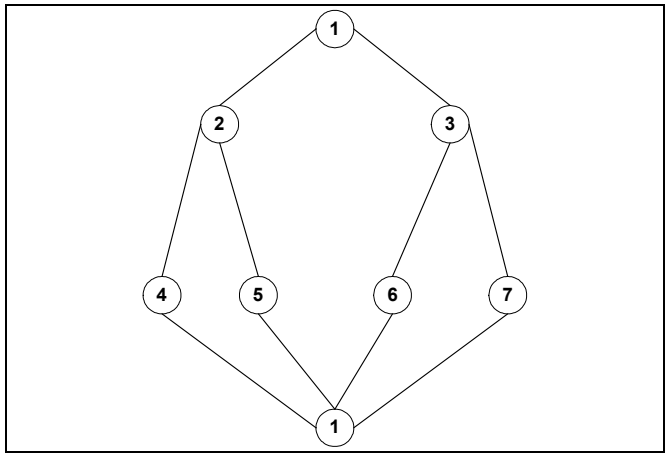


Gambar 2. Contoh solusi Klotski

Chinese Slide Block Puzzle atau Klotski dapat diselesaikan dengan banyak cara. Salah satunya adalah menggunakan **algoritma pencarian melebar** atau *Breadth First Search* (BFS). BFS merupakan salah satu algoritma traversal selain DFS. Algoritma ini menggunakan representasi graf atau pohon dalam proses pencarian solusi.

Misalkan kita mempunyai graf G yang mempunyai *n* buah simpul. Kita akan melakukan traversal di dalam graf, dan misalkan traversal dimulai dari simpul *v*. Algoritma BFS adalah sebagai berikut: Kunjungi simpul *v*, kemudian semua simpul yang bertetangga dan belum dikunjungi dengan simpul-simpul tadi dikunjungi, demikian seterusya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum simpul-simpul pada aras $d + 1$.[2]

Tinjau graf pada Gambar 3. Bila graf dikunjungi mulai dari simpul 1, maka urutan simpul yang dikunjungi adalah 1, 2, 3, 4, 5, 6, 7, 8.



Gambar 3. Contoh graf yang dikunjungi

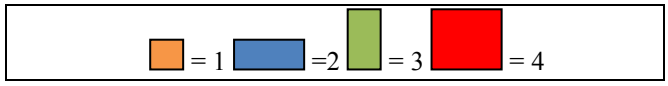
2. PENERAPAN ALGORITMA BFS PADA KLOTSKI

Penyelesaian Klotski dengan BFS menggunakan sebuah *queue* atau antrian untuk menyimpan kandidat-kandidat solusi yag terbentuk. Antrian ini dapat menyimpan kandidat solusi dan juga menghapus kandidat solusi. Kandidat solusi yang disimpan lebih dulu, akan dikeluarkan lebih dulu pula daripada kandidat solusi yang disimpan lebih akhir. Dengan digunakannya antrian, penggunaan memori akan menjadi lebih efisien.

Algoritma ini juga menggunakan sebuah *array* yang berisi matriks 5 x 4. *Array* ini berfungsi untuk memeriksa apakah matriks yang dihasilkan setelah pergeseran blok pernah muncul pada hasil pergeseran sebelumnya.

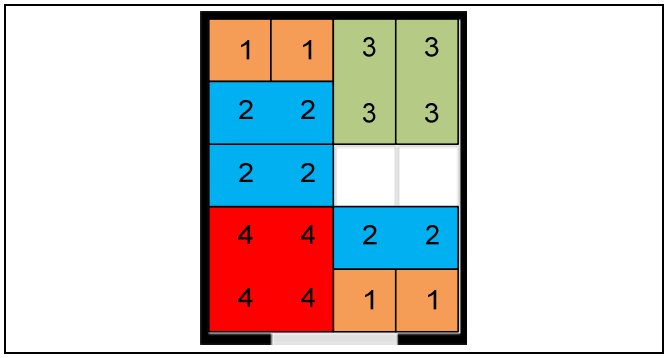
2.1 Pemodelan Blok Klotski

Model yang digunakan untuk blok-blok pada klotski adalah matriks integer berukuran 5 x 4. Matriks ini hanya berisi angka hasil enumerasi blok-blok pada soal kloski. Masing-masing bentuk blok diberi nomor sebagai berikut:



Gambar 4. Enumerasi blok klotski

Pemodelan blok klotski dilakukan agar susunan blok yang sudah pernah muncul dapat diperiksa, sehingga susunan blok yang sama tidak akan diperiksa lagi. Pada Gambar 5, dapat dilihat hasil pemberian nomor blok pada contoh soal klotski pada Gambar 4.



Gambar 5. Hasil pemberian nomor blok

Ruang kosong pada klotski diberi angka 0. Berikut ini adalah matriks 5 x 4 yang dihasilkan setelah memberikan nomor-nomor yang bersesuaian pada contoh soal klotski di atas:

1	**1**	**3**	**3**
2	**2**	**3**	**3**
2	**2**	**0**	**0**
4	**4**	**2**	**2**
4	**4**	**1**	**1**

Gambar 6. Matriks model klotski

Bila salah satu blok digeser ke ruang kosong, maka isi matriks juga akan berubah mengikuti susunan blok tersebut.

2.2 Algoritma BFS

Pada saat pencarian solusi akan dilakukan, matriks *v* yang terbetuk pertama kali atau soal akan dimasukkan ke dalam antrian *q* dan juga akan dimasukkan ke dalam *array d*. Selanjutnya, matriks *v* akan diambil dari antrian. Matriks yang telah diambil akan diperiksa. Semua kemungkinan pergeseran blok dari struktur matriks yang kini sedang diperiksa akan disimpan sementara ke dalam *array d*. Jika masing-masing elemen *array* belum muncul pada *array*, elemen array yang berupa matriks tersebut akan disimpan ke dalam antrian *q* yang nantinya akan dikeluarkan dan diperiksa. Proses tersebut akan terus berjalan hingga ditemukan solusi atau tidak ada solusi.

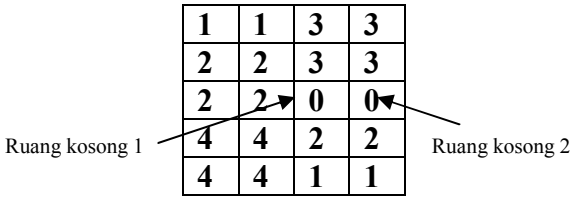
Jika seluruh antrian sudah kosong, maka solusi tidak ditemukan. Hal itu dapat terjadi jika terdapat kesalahan pada soal sehingga pergeseran blok tidak dilakukan lagi.

Namun jika di tengah pencarian telah ditemukan solusi, pencarian akan berakhir dan langkah yang benar akan disimpan untuk ditampilkan atau diproses.

Pencarian solusi menggunakan proses iteratif, dengan memanfaatkan *loop* sampai solusi ditemukan atau antrian *q* sudah kosong.

```
procedure BFSbacktrack()
Deklarasi
  q: antrian
  kandidat: matriks of integer
  kandidatBaru: array of kandidat
  dikunjungi: array of kandidat
  i,k: integer
  solusi: boolean

  procedure buatAntrian(input/output q:antrian)
  {mebuat antrian kosong, kepala(q) diisi 0}

  procedure masukAntrian(input/output q:antrian,
input v: matriks)
  {memasukkan v ke dalam antrian q pada posisi
belakang}

  procedure hapusAntrian(input/output q:antrian,
output v:matriks of integer)
  {menghapus v dari kepala antrian}

  function antrianKosong(input q:antrian) →
boolean
  {true jika antrian q kosong, false jika
sebaliknya}

  function isSolusi(input v: matriks of integer)
→ boolean
  {true jika matriks v merupakan solusi, false
jika sebaliknya}

  procedure buatKandidatBaru(input v: matriks,
output av: array of matriks, n: integer)
  {membuat array kandidat baru dengan aturan
klotski dan mengisi nilai n dengan jumlah
kandidat baru}

  function pernahDikunjungi(input, v: matriks of
integer, arrv: array of matriks) → boolean
  {true jika matriks v terdapat dalam array}

Algoritma
  solusi←false
  kandidat←kandidatPertama
  dikunjungi[0]←kandidat
  k←1
  buatAntrian(q)
  masukAntrian(kandidat);
  while(antrianKosong=false and solusi=false) do
    hapusAntrian(q, kandidat)
    if (isSolusi(kandidat)) then
      prosesSolusi(kandidat);
      solusi=true
    endif
    else
      buatKandidatBaru(kandidat, kandidatBaru,
jumlah);
      for i←1 to jumlah do
        if(pernahDikunjungi(kandidatBaru[i],
dikunjungi) = false)
          masukAntrian(kandidatBaru[i]);
          dikunjungi[k]←kandidat
          k←k+1
        endif
      endfor
    endelse
  endwhile
  {antrianKosong(q)}
```

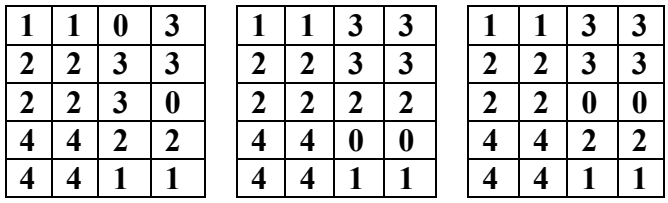
Ruang kosong akan dicari dengan menggunakan pencarian iteratif dari elemen matriks paling kiri atas. Ruang kosong pertama adalah ruang kosong yang ditemukan pertama kali dalam pencarian. Ruang kosong selanjutanya adalah ruang kosong kedua.



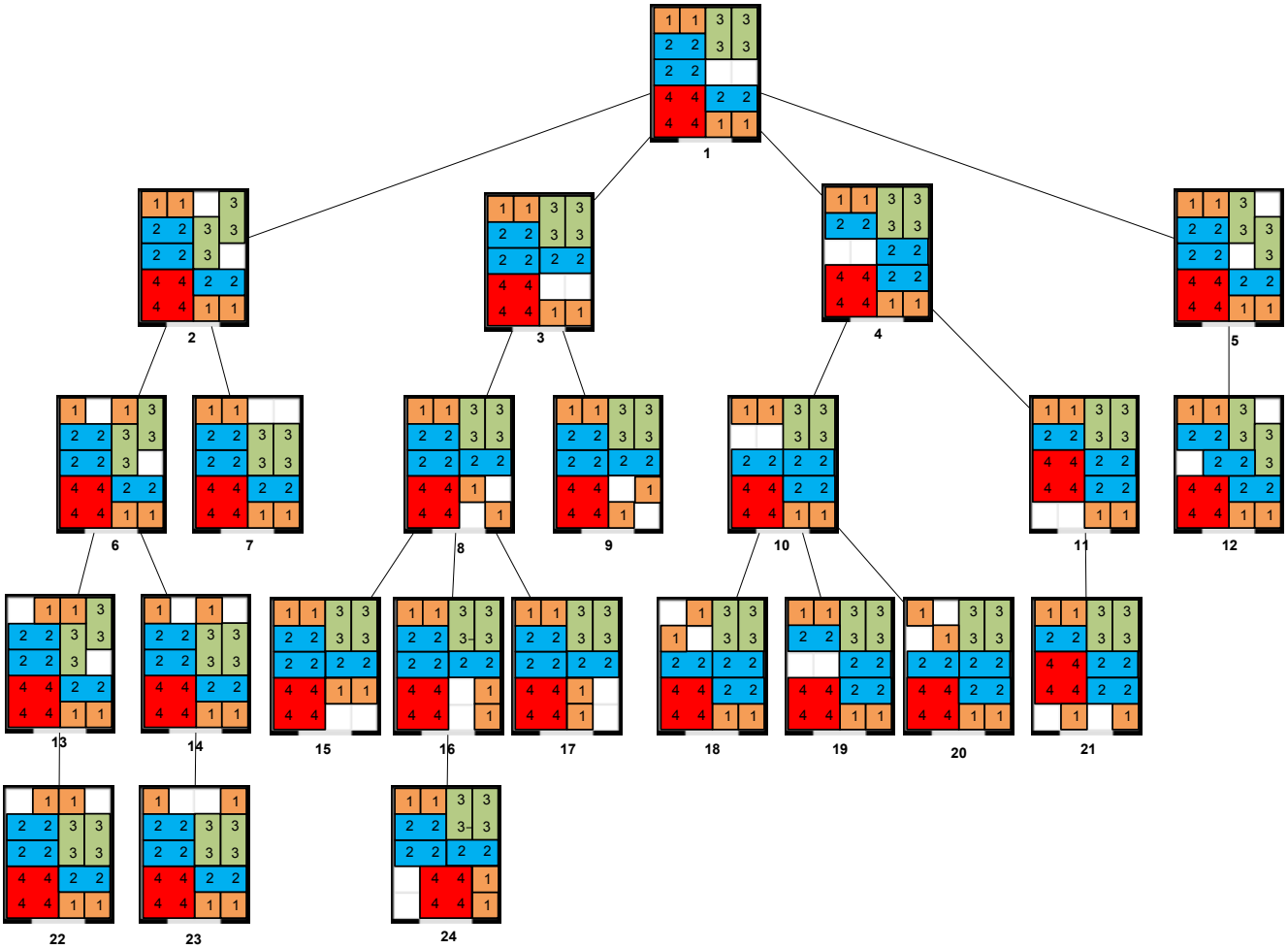
Gambar 7. Index ruang kosong

Jika ruang kosong sudah ditemukan, maka dari posisi index ruang kosong tersebut akan diperiksa dari urutan atas, kanan, bawah, dan kemudian kirinya. Jika ada dari arah tersebut blok yang dapat digeser ke ruang kosong tersebut dan matriks hasil pergeseran belum pernah muncul, maka pergeseran dilakukan.

Jika melihat dari pohon pembangkitan, setiap pergeseran akan membangkitkan satu simpul dan setiap simpul akan mempunyai anak jika masih bisa dilakukan pergeseran dari matriks klotski tersebut.



Gambar 8. Urutan pergeseran pada ruang kosong 1



Gambar 9. Pohon pembangkitan klotski

3. KESIMPULAN

Dalam memecahkan persoalan dalam **Chinese Slide Block Puzzle** atau Klotski, algoritma BFS dapat menjadi salah satu alternatif. Algoritma ini lebih baik daripada harus mengguanakan *brute force*. Tetapi algoritma ini juga tidak dapat dinilai paling bagus, karena masih banyak algoritma lain yang dapat menyelesaikan persoalan klotski.

REFERENSI

[1] http://www.codeproject.com

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