Perbandingan Algoritma *Brute Force* dan *Breadth First Search* dalam Permainan Onet

Dininta Annisa / 13513066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
dinintaannisa@students.itb.ac.id

***Abstrak*—Onet adalah aplikasi permainan logika yang tersedia di PC maupun mobile. Permainan ini bertujuan untuk mencocokkan gambar pada kumpulan kartu yang tersedia di board. Onet secara otomatis akan mengacak susunan kartunya jika sudah tidak ada lagi langkah yang dapat dilakukan pemain. Pengacakan ini melibatkan proses pencarian langkah terhadap seluruh kemungkinan langkah. Makalah ini bertujuan untuk membandingkan algoritma *brute force* dan *breadth first search* jika diterapkan dalam proses pencarian tersebut.**

***Kata Kunci*—Onet, Brute Force, Breadth First Search, Algoritma Pencarian**

I. PENDAHULUAN

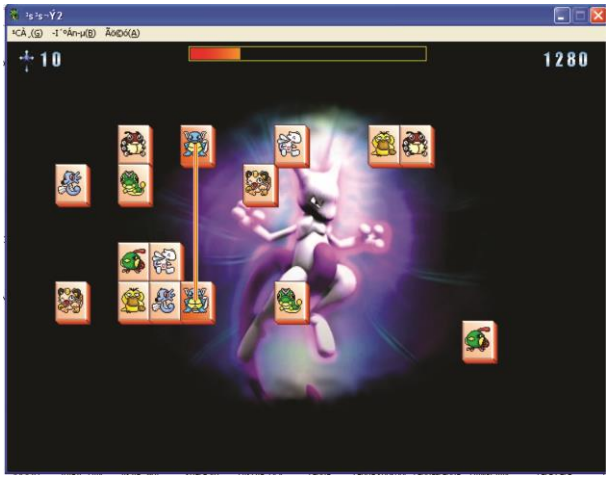
Permainan logika yang mengasah otak semakin marak di kalangan masyarakat. Permainan puzzle seperti kakurasu dan sudoku kini tidak hanya tersedia di koran mingguan saja, tetapi juga dalam bentuk aplikasi yang bebas diunduh siapapun. Game-game yang sudah cukup berumur pun kini muncul lagi dalam kemasan yang baru, tidak lagi tersedia hanya di PC, tapi juga di web dan mobile.

Onet adalah salah satunya. Onet adalah permainan logika yang sempat marak di tahun 2000an. Saat ini, Onet kembali digemari sejak tersedia dalam bentuk aplikasi berplatform desktop dan mobile. Onet bahkan hadir dalam berbagai versi. Jika pada awal dirilis permainan ini menggunakan gambar-gambar pokemon, sekarang sudah banyak versi yang menggunakan gambar-gambar lain seperti buah dan bunga, bahkan dengan tambahan level yang berbeda.

Onet sendiri dibuat pada tahun 2004 oleh Chen Program Study dengan nama program "D4S Pokemon Matching Game". Aplikasi ini terus dikembangkan hingga sekarang dapat dijalankan di mobile.

Gambar 1: Tampilan game Onet

Pada dasarnya, Onet adalah sebuah board game dengan sejumlah kartu bergambar di atas boardnya. Pemain harus memasangkan setiap kartu dengan kartu lain yang bergambar sama dengan syarat kedua kartu tersebut terhubung oleh maksimal tiga garis lurus. Garis penghubung tersebut tidak boleh terputus dan terhalang oleh kartu yang lain. Jika dua buah kartu berhasil dipasangkan, maka kartu tersebut akan hilang dari board. Pemain dinyatakan menang jika seluruh kartu di board habis dalam jangka waktu tertentu.



Gambar 2: Contoh langkah yang valid pada Onet

Meskipun aturannya sederhana, permainan Onet cukup sulit karena jumlah kartu yang banyak serta gambar setiap kartu yang mirip. Untuk mengurangi tingkat kesulitan, aplikasi ini menyediakan fungsi otomatis untuk mengacak ulang board jika sudah tidak ada lagi pencocokkan kartu yang mungkin dilakukan. Jadi, pemain tidak akan kalah karena sudah tidak ada jalan lagi. Satu-satunya penyebab kekalahan pemain hanya karena kehabisan waktu.

Fungsi pengacakan ulang board pada Onet adalah salah satu penerapan dari algoritma pencarian. Aplikasi Onet ini perlu mencari apakah masih ada kartu pada board yang bisa dicocokkan. Ada banyak macam-macam algoritma pencarian yang bisa diterapkan, salah satunya *brute force* dan *breadth first search*. Keduanya memberikan hasil akhir yang sama, tetapi dengan cara kerja dan performansi yang berbeda.

II. DASAR TEORI

A. Algoritma Brute Force

Brute force adalah algoritma yang menyelesaikan masalah dengan lempang (*straightforward*). Cara kerja *brute force* relatif sederhana, jelas, dan didasari pada pernyataan masalah yang bersangkutan.

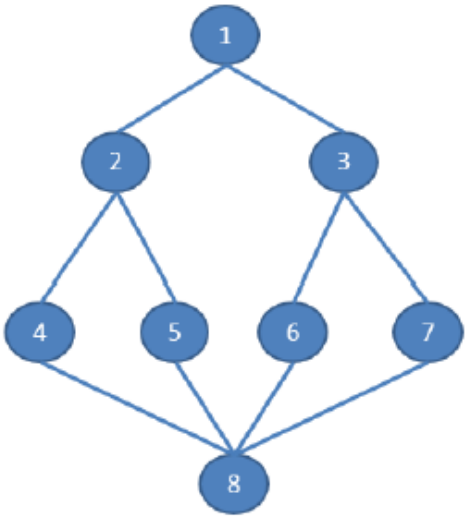
Salah satu persoalan yang dapat diselesaikan *brute force* adalah masalah pencarian. Sebagai contoh, pencarian suatu bilangan dalam sebuah larik secara *brute force* dilakukan dengan membandingkan semua bilangan pada larik dengan bilangan yang dicari sampai bilangan itu ditemukan atau sampai seluruh elemen larik telah diperiksa.

Masalah pencarian lainnya yang dapat diselesaikan secara lempang adalah pencarian jarak terpendek dari sebuah titik ke titik lainnya. Persoalan ini diselesaikan dengan cara mengenumerasi semua kemungkinan, menghitung jarak dari setiap kemungkinan, lalu memilih solusi dengan jarak terkecil. Penyelesaian dengan cara seperti ini lebih tepat disebut sebagai *exhaustive search*. Secara definisi, *exhaustive search* adalah teknik pencarian solusi secara *brute force* pada masalah yang melibatkan pencarian elemen dengan sifat khusus.

B. Algoritma Breadth First Search

Breadth first search (BFS) adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara *preorder* yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi , demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum simpul-simpul pada aras *d*+1.

Algoritma ini memerlukan sebuah antrian *q* untuk menyimpan simpul yang telah dikunjungi. Simpulsimpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungu masuk ke dalam antrian hanya satu kali. Algoritma ini juga membutuhkan table Boolean untuk menyimpan simpul yang telah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali.



Gambar 3: Contoh graf

Dalam algoritma BFS, simpul anak yang telah dikunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian. Untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya, berikut langkah-langkah algoritma BFS:

(a) Kunjungi simpul v
(b) Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu.
(c) Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Algoritma BFS juga dapat menyelesaikan persoalan dengan membentuk pohon dinamis. Pencarian solusi diiringi dengan pembentukan pohon dinamis. Setiap simpul pohon diperiksa apakah solusi telah dicapai atau tidak. Jika telah sampai pada simpul solusi, pencarian dapat selesai (satu solusi) atau dilanjutkan mencari solusi lain (semua solusi).

Algoritma lain yang mirip dengan BFS adalah DFS atau *Depth First Search*. Berbeda dengan BFS yang melakukan pencarian secara melebar, DFS melakukan pencarian secara mendalam. Pada makalah ini, algoritma BFS yang digunakan sebab algoritma DFS tidak selalu menghasilkan hasil optimal, meskipun kompleksitas ruangnya lebih baik daripada algoritma BFS.

Kompleksitas algoritma BFS sendiri adalah $O(b^d)$, baik utnuk kompleksitas ruang maupun waktu. Nilai b menyatakan maksimum percabangan yang mungkin dalam sistem, sementara nilai d menyatakan kedalaman dari solusi terbaik.

Iterasi	V	Q	dikunjungi							
			1	2	3	4	5	6	7	8
Inisialisasi	1	{1}	T	F	F	F	F	F	F	F
Iterasi 1	1	{2,3}	T	T	T	F	F	F	F	F
Iterasi 2	2	{3,4,5}	T	T	T	T	T	F	F	F
Iterasi 3	3	{4,5,6,7}	T	T	T	T	T	T	T	F
Iterasi 4	4	{5,6,7,8}	T	T	T	T	T	T	T	T
Iterasi 5	5	{6,7,8}	T	T	T	T	T	T	T	T
Iterasi 6	6	{7,8}	T	T	T	T	T	T	T	T
Iterasi 7	7	{8}	T	T	T	T	T	T	T	T
Iterasi 8	8	{}	T	T	T	T	T	T	T	T

Tabel 1: Hasil iterasi algoritma BFS dari graf pada gambar 2

III. Penyelesaian Persoalan

A. Algoritma Brute Force

Untuk mengecek apakah fungsi pengacakan board perlu dipanggil atau tidak, program perlu mencari apakah masih ada kartu pada board yang bisa dicocokkan. Berikut adalah langkah-langkah pencarian tersebut.

1. Pilih salah satu kartu yang akan dicocokkan, lalu cocokkan kartu tersebut ke setiap kartu lain dalam board.
2. Jika ditemukan kartu yang sama, maka cari jalur yang menghubungkan kedua kartu tersebut. Jalur tidak boleh lebih dari tiga garis lurus dan tidak boleh melewati kartu yang lain. Jika tidak ditemukan kartu yang sama, maka pilih kartu lainnya yang akan dicocokkan dan ulangi langkah pertama.
3. Jika ditemukan jalur yang sesuai dengan ketentuan, maka masih ada kartu yang bisa dicocokkan pada board dan fungsi pengacakan board tidak perlu dipanggil. Langkah pencarian berhenti sampai disini.
4. Jika tidak ditemukan, maka ulangi lagi langkah pertama untuk kartu yang lain sampai ditemukan. Jika masih tidak ditemukan, maka tidak ada lagi kartu pada board yang bisa dicocokkan dan fungsi pengacakan board harus dipanggil.

Langkah-langkah di atas melakukan pencarian kartu yang bergambar sama secara *brute force*. Namun, untuk pencarian jalur setelah kartu yang bergambar sama ditemukan, teknik yang digunakan lebih tepat disebut dengan *exhaustive search*. Berikut adalah penjabaran langkah-langkah pencarian jalur tersebut.

1. Enumerasikan setiap jalur yang menghubungkan kedua kartu. Jalur tersebut tidak boleh melewati kartu lain pada board.
2. Hitung berapa kali jalur tersebut berbelok. Jika lebih dari dua, maka jalur tidak ditemukan, tetapi jika kurang dari atau sama dengan dua, maka jalur ditemukan.

Untuk memperjelas langkah-langkah di atas, berikut adalah pseudocode untuk kedua pencarian tersebut.

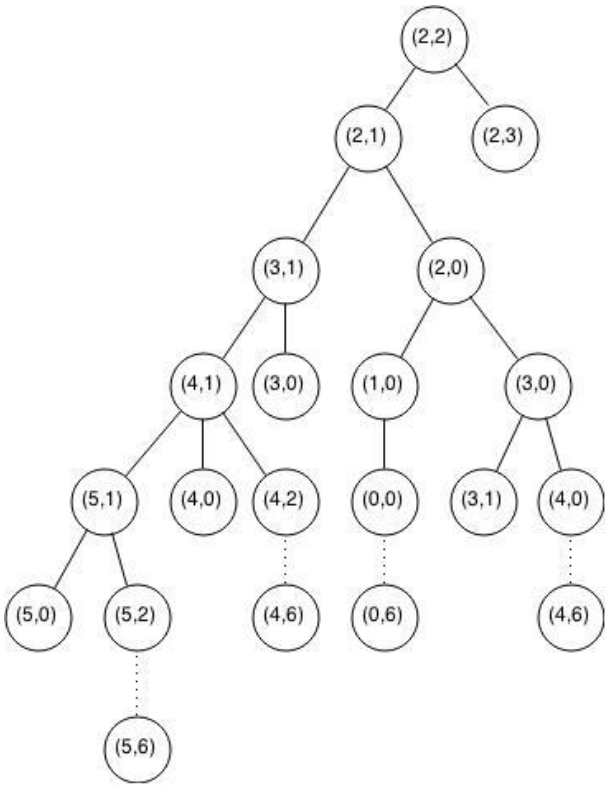
```
procedure CekBoard (input Board M)
{ Mengecek apakah fungsi pemanggilan board
  perlu dipanggil atau tidak }

   int i ← 1;
   int j ← i+1;
   boolean found ← false;
   boolean gambar_sama ← false;
   boolean ada_jalur;

   while not(found) and (i<jml_kartu)
      {cari kartu dengan gambar sama}
      while not(gambar_sama) and
         (j<jml_kartu)
         if (Kartu[i]=Kartu[j])
            ada_jalur ← CariJalur
               (M,Kartu[i],Kartu[j]
            if ada_jalur then
               found ← true

         else
            j ← j+1;
      i ← i+1;

   if not(found) then
      panggil fungsi pengacakan board
```

```
function CariJalur(Board M, Kartu Asal,
Kartu Tujuan) → boolean
{ Menghasilkan true jika terdapat jalur
  dari kartu asal ke tujuan yang tidak
  terhalang kartu lain dan tidak lebih dari
  2 kali berbelok arah }

   Pohon P
   Akar(P) ← Asal
   Simpul E ← Asal
   while E bukan solusi do
      if sel di atas SimpulE kosong
         Anak(E) ← sel di atas E
      else if sel di bawah E kosong
         Anak(E) ← sel di bawah E
      else if sel di kiri E kosong
         Anak(E) ← sel di kiri E
      else if sel di kanan E kosong
         Anak(E) ← sel di kanan E
      E ← semua simpul yang baru dihidupkan

   {Setelah semua simpul pohon terbentuk,
   cari jalur yang tidak berbelok arah
   lebih dari 2 kali}
   for semua jalur di pohon do
      if berbelok ≤ 2
         → true
   → false
```

Perlu diperhatikan bahwa pada fungsi CariJalur dibentuk sebuah pohon pencarian solusi. Meskipun begitu, algoritma tersebut bukan termasuk BFS karena pohon solusi yang dibentuk mengenumerasi seluruh

kemungkinan yang dapat terjadi, kemudian semua jalur pada pohon ditelusuri untuk mengecek mana jalur yang berbelok tidak lebih dari dua kali. Oleh karena itu, algoritma tersebut masih tergolong ke dalam *brute force*, atau lebih tepat lagi jika disebut *exhaustive search*.

Berikut adalah hasil pohon pencarian yang dibentuk.



Gambar 4: Pohon pencarian brute force

B. Algoritma Breadth First Search

Berikut adalah langkah-langkah untuk mencari apakah masih ada kartu pada board yang bisa dicocokkan atau tidak.

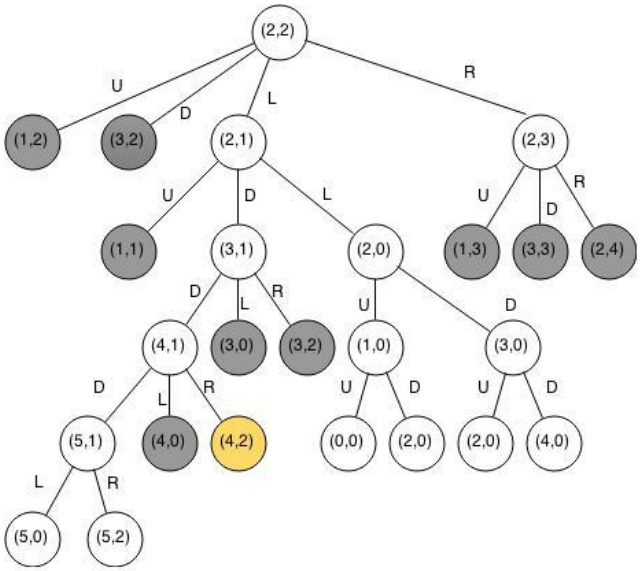
1. Pilih salah satu kartu pada board yang akan diperiksa. Jadikan posisi kartu tersebut sebagai simpul akar.
2. Tambahkan empat buah simpul anak. Masing-masing simpul berisi posisi kartu di simpul parent jika digerakkan ke atas, bawah, kanan, dan kiri. Simpul tidak jadi ditambahkan jika:
   - posisi kartu sudah di luar board
   - lintasan dari simpul akar ke simpul tersebut sudah lebih dari dua kali berubah arah
   - posisi kartu sama dengan posisi parentnya
3. Untuk setiap simpul anak yang baru dibuat, periksa apakah kartu pada posisi tersebut sama dengan kartu yang sedang dicocokkan.
4. Jika sama, maka proses pencarian berhenti. Fungsi pengacakan board tidak perlu dipanggil. Jika tidak sama, ulangi lagi mulai dari langkah dua.
5. Jika tidak ada lagi simpul yang bisa ditambahkan, maka ulangi lagi mulai dari langkah 1 untuk kartu yang lain.
6. Jika seluruh kartu sudah diperiksa, tapi masih belum ditemukan sepasang kartu bergambar sama dengan jarak kurang dari tiga lintasan, maka maka proses pencarian berhenti. Fungsi pengacakan board dipanggil.

Misalkan board permainan yang digunakan berukuran 4 x 4 dan huruf pada petak menyatakan gambar dari kartu sebagaimana berikut ini.

	0	1	2	3	4	5
0						
1		B	C	D		
2			A		C	
3			B	D		
4			A			
5						

Gambar 5: Board game

Jika kita memilih kartu A pada posisi (2,2) sebagai kartu yang akan dicocokkan, maka akan diperoleh pohon hasil pencarian seperti berikut.



Gambar 6: Pohon hasil pencarian

Dari proses pencarian tersebut, diperoleh posisi ke (4,2) yang mengandung kartu bergambar sama dengan kartu di posisi (2,2).

IV. ANALISIS

Baik algoritma *brute force* maupun *breadth first search*, keduanya menghasilkan hasil yang sama dalam persoalan pencarian dalam permainan Onet, yaitu perlu atau tidaknya pemanggilan fungsi pengacakan board. Meskipun begitu, proses pencarian dengan algoritma *brute force* menghabiskan waktu yang lebih lama dibandingkan dengan BFS.

Misalkan ada n buah kartu yang tersisa pada board dengan m petak. Pada algoritma *brute* force, perbandingan gambar antar kartu membutuhkan waktu

dalam O(n). Setelah itu, jika gambar yang sama ditemukan, akan dilakukan pencarian jalur. Karena ada (m-n)! sel yang kosong dan perlu diperiksa, maka pencarian tersebut menghabiskan waktu dalam O(n!). Jadi, kompleksitas algoritma *brute force* dalam persoalan ini adalah O(n + n!) = O(n!).

Dari kompleksitas tersebut, dapat diyakini bahwa algoritma *brute force* kurang mangkus untuk menyelesaikan persoalan pencarian ini, sebab kompleksitasnya non polinomial.

Untuk algoritma *breadth first search*, setiap simpul dapat memiliki maksimal 4 buah simpul anak. Asumsikan jarak kedua kartu kurang lebih separuh dari jumlah petak yang ada, maka pencarian dengan BFS akan menghabiskan waktu dalam $O(4^m)$.

Kompleksitas algoritma BFS juga dinyatakan dalam fungsi non polinomial, sama seperti algoritma *brute force*. Namun, pencarian dengan BFS masih lebih cepat dibandingkan algoritma *brute force*. Misalkan untuk kasus board seperti pada gambar 5. Pada algoritma *brute force*, perlu dibangkitkan 37 simpul pada pohon, lalu diperiksa ulang 18 simpul diantaranya saat lelaran yang terakhir. Sementara itu, pada algoritma BFS, ada 25 simpul yang perlu dibangkitkan dan tidak perlu diperiksa ulang. Jadi pada kasus tersebut, algoritma BFS dapat menyelesaikan persoalan dua kali lebih cepat dibanding penyelesaian dengan algoritma *brute force*.

Meskipun waktu eksekusi kenyataan kedua alternatif penyelesaian tersebut relatif sangat cepat (kurang dari satu detik), tetapi pemilihan algoritma tersebut sangat berpengaruh terhadap kinerja program. Program harus bisa secepeat dan seefisien mungkin, agar tidak memakan banyak ruang memori. Ruang memori yang berlebih karena penggunaan algoritma *brute force* akan lebih baik lagi jika dialokasikan untuk kebutuhan program yang lain seperti sound atau tampilan grafisnya. Selain itu, program juga harus secepat mungkin karena waktu penyelesaian persoalan ini akan memakan durasi permainan bagi user.

V. KESIMPULAN

Algoritma *brute force* dan *breadth first search* merupakan salah satu algoritma pencarian yang kerap digunakan. Keduanya dapat diimplementasikan dalam salah satu fitur game Onet, yaitu untuk mengecek apakah masih ada kartu yang dapat dicocokkan atau tidak. Hasil yang diperoleh kedua algoritma tersebut sama, tetapi didapatkan dalam selang waktu yang berbeda. Hal ini disebabkan oleh perbedaan kedua kompleksitas algoritma. Dalam persoalan ini, kompleksitas pencarian dengan *brute force* adalah O(n!) Sementara kompleksitas pencarian dengan *breadth first search* adalah $O(4^m)$.

Perbedaan waktu eksekusi dan kemangkusan kedua algoritma ini menjadi sangat penting, sebab jika tidak ada lagi kartu yang dapat dicocokkan pada board, board harus segera di-*shuffle* sesegara mungkin agar pemain tidak kehilangan banyak waktu. Oleh karena kompleksitas algoritma BFS lebih mangkus daripada algoritma *brute force*, algoritma BFS lebih cocok untuk diterapkan dalam persoalan ini.

REFERENCES

[1] Munir, Rinaldi. () Diktat Kuliah IF2211 Strategi Algoritma. (Diktat kuliah). Institut Teknologi Bandung.

[2] http://deddy99.blogspot.com/2013/01/download-game-onet-msvbvm50dll.html, diakses pada tanggal 4Mei 2015 pukul 13.00

[3] https://www.cs.umd.edu/class/summer2014/cmsc132/lectures/Week9/AlgorithmStrategies.pdf, diakses pada tanggal 5 Mei 2015 pukul 04.00

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 04 Mei 2015

Dininta Annisa
13513066