Penggunaan Strategi Algoritma Backtracking pada Pencarian Solusi Puzzle Pentomino

Muhammad Rian Fakhrusy / 13511008
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10, Bandung 40132, Indonesia
staroushi38@gmail.com

***Abstract*— Pada makalah ini akan dijelaskan penggunaan algoritma *backtracking* pada pencarian solusi *puzzle* pentomino. Makalah ini bertujuan untuk menentukan apakah algoritma *backtracking* cukup baik untuk menyelesaikan *puzzle* pentomino. Algoritma *backtracking* dapat menyelesaikan *puzzle* pentomino. Akan tetapi, waktu yang dibutuhkan untuk penyelesaiannya cukup lama sehingga dianggap kurang baik jika digunakan dalam permasalahan pentomino.**

Keywords—backtrack; pentomino; algoritma; puzzle

I. PENDAHULUAN

Puzzle pentomino adalah sebuah permainan logika yang menggunakan 12 jenis pentomino untuk membentuk sebuah pola tertentu. Di dalam permainan ini, pemain harus menempatkan 12 jenis pentomino yang berbeda ke dalam sebuah pola yang memiliki ukuran total 60 petak. Setiap pentomino memiliki ukuran 5 petak, sehingga 12 buah pentomino dapat dimuat ke dalam pola berukuran 60 petak. Permasalahan utama dalam peletakan pentomino ke pola berukuran 60 petak ini, terletak pada bentuk dari 12 buah pentomino yang harus berbeda satu dengan lainnya. Jika saja pemain boleh menggunakan bentuk yang sama secara berulang-ulang, tentulah *puzzle* ini akan dengan mudah dapat diselesaikan. Perbedaan bentuk tiap pentomino lah yang menjadi tantangan utama *puzzle* ini.

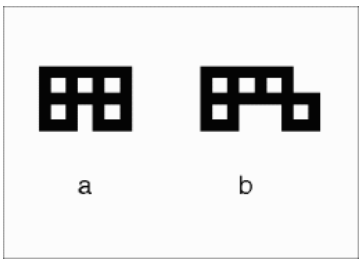
Jika *puzzle* pentomino ini diselesaikan secara manual, tentu akan membutuhkan waktu yang cukup lama. Terdapat 12 bentuk pentomino. Jika pembalikan bentuk pentomino membentuk pentomino baru maka akan terdapat 18 jenis pentomino. Dan jika rotasi dari tiap 18 jenis pentomino tersebut membentuk sebuah pentomino yang baru, maka akan terdapat 63 buah pentomino[1]. Memilih 12 pentomino yang berbeda bentuknya dan sesuai dengan pola yang ditentukan tentu saja membutuhkan waktu yang lama. Untuk puzzle pentomino dengan ukuran pola 6 x 10 terdapat 2339 solusi[2]. Jumlah solusi ini sangatlah sedikit. Terdapat kurang lebih $3x10^{16}$ cara untuk meletakkan 12 pentomino ke dalam sebuah segi empat dengan berbagai rotasi dan pembalikannya[3] dan hanya 2339 cara saja yang merupakan solusi untuk pola dengan ukuran 6 x 10.

Puzzle pentomino ini mungkin saja dapat diselesaikan lebih cepat dengan menggunakan komputer. Salah satu strategi algoritma yang dapat digunakan adalah *backtracking*. Makalah ini akan menjelaskan implementasi strategi algoritma *bracktracking* dalam permasalahan *puzzle* pentomino. Kemudian akan disimpulkan apakah strategi algoritma *backtracking* tersebut cukup baik untuk penyelesaian permasalahan *puzzle* pentomino.

II. DASAR TEORI

A. Pentomino

Pentomino adalah salah satu bentuk dari polyomino dengan ukuran 5 petak. Kata polyomino sendiri diambil dari kata domino (polyomino dengan ukuran 2 petak), sebagai sebuah bentuk generalisasi. Secara istilah, polyomino dapat diartikan sebagai sebuah kumpulan n petak yang berukuran sama yang salah satu sisinya saling berhimpitan. Sebagai ilustrasi, pada gambar berikut ini (a) adalah sebuah pentomino dan (b) bukan sebuah pentomino karena sisinya tidak berhimpitan.

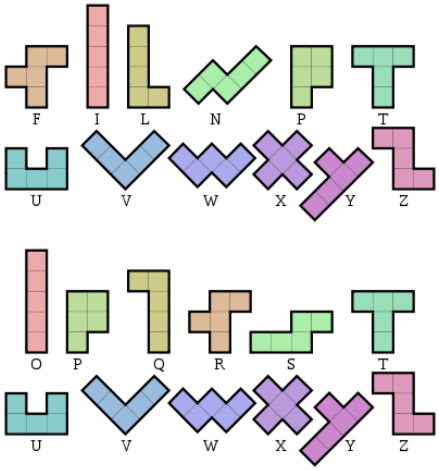


Gambar 1. Pentomino dan bukan pentomino (Sumber : http://www.basic.northwestern.edu/g-buehler/pentominoes/speech.htm)

Polyomino dapat diubah bentuknya dengan cara dirotasikan atau dibalikkan, namun polyomino tersebut masih dikategorikan sebagai polyomino yang sama. Polyomino dikategorikan berbeda jika memiliki kiralitas yang berbeda. Berikut adalah seluruh bentuk polyomino dengan n=1 sampai n=5 yang memiliki kiralitas yang berbeda.

Gambar 2. Seluruh bentuk 1-omino hingga 5-omino (Sumber: http://mathworld.wolfram.com/Polyomino.html)

Pentomino atau bisa juga disebut dengan 5-omino, memiliki 12 bentuk (seperti pada gambar 2). Rotasi dan pembalikan dari ke-12 bentuk tersebut dapat membentuk 63 jenis pentomino. Goloumb (1995) memberi nama ke-12 pentomino tersebut dengan huruf F, I, L, N, P, T, U, V, W, X, Y, and Z (dari kiri ke kenan gambar pentomino). Inilah penamaan yang digunakan secara umum. Namun, untuk penyederhanaan maka huruf F, I, L, dan N diganti dengan huruf R, O, Q, S agar seluruh huruf dari O hingga Z digunakan di dalam pentomino [1].



Gambar 3. Perbedaan penamaan seluruh bentuk pentomino (sumber gambar : wikipedia)

Ke-12 jenis pentomino ini dapat membentuk sebuah pola segi empat yang memiliki ukuran 5 x 12 = 60 unit persegi. Misalnya, pola segi empat dengan ukuran 6 x 10, 5 x 12, 4 x 15, dan 3 x 20. Ke-4 segi empat tersebut memiliki total 60 unit persegi dan dapat dibentuk dengan menggunakan ke-12 jenis pentomino yang telah disebutkan[3] . Pola 6 x 10 memiliki 2339 solusi, pola 5 x 12 memiliki 1010 solusi, pola 4 x 15 memiliki 368 solusi dan pola 3 x 20 hanya memiliki 2 solusi[2] .

B. Backtracking

Algoritma *backtracking* (algoritma runut-balik) adalah salah satu strategi pemecahan masalah yang menggunakan basis pencarian mendalam (*Depth-First Search*) dan memiliki cara untuk memotong solusi persoalan yang tidak mengarah ke solusi. Ketika menemukan jalur yang tidak mengarah ke solusi, algortima *backtracking* akan kembali ke jalur sebelumnya yang memiliki alternatif solusi yang memungkinkan untuk sampai ke solusi[4] .

Skema umum algoritma *backtracking* adalah dengan secara rekursif, walaupun algoritma *backtracking* juga dapat melakukan penjarian solusi dengan cara iteratif. Berikut adalah langkah-langkah penyelesaian permasalahan dengan menggunakan algoritma *backtracking*.

1. Pencarian solusi dimulai dari akar menuju ke daun pohon. Perluasan simpul dilakukan dengna pendekatan depth-first search (secara mendalam). Simpul-simpul yang telah dibangkitkan tersebut menjadi simpul hidup dan simpul yang sedang diperluas disebut dengan *expand node*. Simpul diberi nomor sesuai dengan nomor mulai dibentuknya.
2. Tiap kali *expand node* diperluas, akan diperiksa apakah simpul anak tersebut mengarah ke solusi. Jika tidak, maka simpul tersebut "dibunuh" dan menjadi simpul mati yang tidak akan diperluas kembali. Fungsi untuk membunuh simpul *expand node* disebut dengan fungsi pembatas.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan kembali ke simpul hidup terdekat yaitu orang tua dari simpul yang telah mati tersebut. Kemudian pencarian diteruskan ke simpul hidup terdekat lainnya.
4. Pencarian berhenti ketika simpul solusi telah ditemukan atau tidak ada lagi simpul hidup yang dapat dikembangkan (tidak ada solusi)[5].

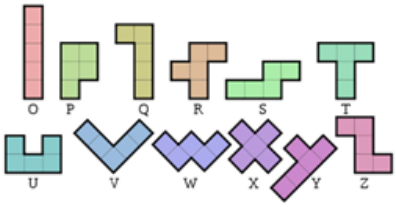
Pseudo-code dari algoritma *backtracking* secara umum adalah sebagai berikut :

```
boolean solve(Node n) {
    if n adalah node daun {
        if n adalah node solusi
            return true
        else
            return false
    } else {
        untuk tiap node anak dari n {
            if solve(c) berhasil
            return true
        }
        return false
    }
}
```

Ketika hasil *true* dikembalikan oleh *pseudo-code* ini, berarti solusi ditemukan dan *node* n saat itulah yang menjadi solusinya. Ketika hasil kembaliannya adalah *false*, maka tidak ada *node* yang menjadi solusi permasalahan tersebut[4].

III. Analisis Pemecahan Masalah

Akan digunakan penamaan pentomino dengan menggunakan format O-Z untuk kemudahan implementasi. Berikut adalah ke-12 bentuk pentomino tersebut.



Gambar 4. Penamaan pentomino dari O-Z

Berikutnya, akan didaftar perubahan bentuk dari tiap pentomino tersebut, baik dengan cara dirotasi ataupun dibalik. Format dari penomoran tersebut adalah dengan menyebutkan 4 lokasi sel pentomino yang bukan merupkan sel yang berada di paling atas kiri, relatif terhadap lokasi persegi yang berada di paling ataas kiri pentomino. Format tersebut adalah {(x1,y1), (x2,y2), (x3,y3), (x4,y4)}

Sel pentomino yang terletak paling kiri atas adalah sel pentomino yang pertama. Lokasinya direpresentasikan dengan format (0,0) yang menjadi acuan terhadap posisi sel pentomino yang lain. Tiap sel pentomino yang lain direpresentasikan dengan format (xi,yi) dimana x adalah baris dari sel pentomino yang lain relatif terhadap pentomino yang pertama secara vertikal ke bawah dan y adalah kolom dari sel pentomino yang lain relatif terhadap pentomino yang pertama secara horizontal ke kanan (jika disebelah kiri berarti bernilai negatif). Misalnya {(1,0), (2,0), (3,0), (4,0)} merepresentasikan pentomino dengan bentuk O yang memanjang secara vertikal. Hal ini dapat dilihat dari sel kedua yang berada pada lokasi (1,0) relatif terhadap pentomino pertama, sel ketiga yang berada pada lokasi (2,0) relative terhadap pentomino pertama dan seterusnya.

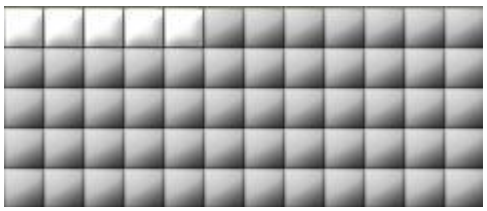
Berikut adalah tabel seluruh bentuk pentomino

Nomor urutan pemasangan pentomino	Bentuk pentomino	Sel ke-i relatif terhadap sel pertama dalam format (x,y)			
		2	3	4	5
1	O	(1,0)	(2,0)	(3,0)	(4,0)
2		(0,1)	(0,1)	(0,3)	(0,4)
3	P	(0,1)	(1,0)	(1,1)	(2,0)
4		(0,1)	(0,2)	(1,1)	(1,2)
5		(1,-1)	(1,0)	(2,-1)	(2,0)
6		(0,1)	(1,0)	(1,1)	(1,2)
7		(0,1)	(1,0)	(1,1)	(2,1)
8		(0,1)	(1,-1)	(1,0)	(1,1)
9		(1,0)	(1,1)	(2,0)	(2,1)
10		(0,1)	(0,2)	(1,0)	(1,1)
11	Q	(0,1)	(1,1)	(2,1)	(3,1)
12	Q	(1,-3)	(1,-2)	(1,-1)	(1,0)
13		(1,0)	(2,0)	(3,0)	(3,1)
14		(0,1)	(0,2)	(0,3)	(1,0)
15		(0,1)	(1,0)	(2,0)	(3,0)
16		(0,1)	(0,2)	(0,3)	(1,3)
17		(1,0)	(2,0)	(3,-1)	(3,0)
18		(1,0)	(1,1)	(1,2)	(1,3)
19	R	(0,1)	(1,-1)	(1,0)	(2,0)
20		(1,-1)	(1,0)	(1,1)	(2,1)
21		(1,0)	(1,1)	(2,-1)	(2,0)
22		(1,0)	(1,1)	(1,2)	(2,1)
23		(0,1)	(1,1)	(1,2)	(2,1)
24		(1,-2)	(1,-1)	(1,0)	(2,-1)
25		(1,-1)	(1,0)	(2,0)	(2,1)
26		(1,-1)	(1,0)	(1,1)	(2,-1)
27	S	(0,1)	(1,-2)	(1,-1)	(1,0)
28		(1,0)	(2,0)	(2,1)	(3,1)
29		(0,1)	(0,2)	(1,-1)	(1,0)
30		(1,0)	(1,1)	(2,1)	(3,1)
31		(0,1)	(1,1)	(1,2)	(1,3)
32		(1,-1)	(1,0)	(2,-1)	(3,-1)
33		(0,1)	(0,2)	(1,2)	(1,3)
34		(1,0)	(2,-1)	(2,0)	(3,-1)
35	T	(0,1)	(0,2)	(1,1)	(1,2)
36		(1,-2)	(1,-1)	(1,0)	(2,0)
37		(1,0)	(2,-1)	(2,0)	(2,1)
38		(1,0)	(1,1)	(1,2)	(2,0)
39	U	(0,2)	(1,0)	(1,1)	(1,2)
40		(0,1)	(1,0)	(2,0)	(2,1)
41		(0,1)	(0,2)	(1,0)	(1,2)
42		(0,1)	(1,1)	(2,0)	(2,1)
43	V	(1,0)	(2,0)	(2,1)	(2,2)
44		(0,1)	(0,2)	(1,0)	(2,0)
45		(0,1)	(0,2)	(1,2)	(2,2)
46		(1,0)	(2,-2)	(2,-1)	(2,0)

47	W	(1,0)	(1,1)	(2,1)	(2,2)
48		(0,1)	(1,-1)	(1,0)	(2,-1)
49		(0,1)	(1,1)	(1,2)	(2,2)
50	W	(1,-1)	(1,0)	(2,-2)	(2,-1)
51	X	(1,-1)	(1,0)	(1,1)	(2,0)
52	Y	(1,-1)	(1,0)	(2,0)	(30,)
53		(1,-2)	(1,-1)	(1,0)	(1,1)
54		(1,0)	(2,0)	(2,1)	(3,0)
55		(0,1)	(0,2)	(0,3)	(1,1)
56		(1,0)	(1,1)	(2,0)	(3,0)
57		(0,1)	(0,2)	(0,3)	(1,2)
58		(1,0)	(2,-1)	(2,0)	(3,0)
59		(1,-1)	(1,0)	(1,1)	(1,2)
60	Z	(0,-1)	(1,1)	(2,1)	(2,2)
61		(1,-2)	(1,-1)	(1,0)	(2,-2)
62		(0,1)	(1,0)	(2,-1)	(2,0)
63		(1,0)	(1,1)	(1,2)	(2,2)

Urutan pemasangan pentomino ke dalam segi empat sesuai dengan urutan 1-63 pada tabel diatas. Tiap sel dari segi empat akan diberi penomoran 1-60 dari kiri ke kanan lalu dari atas ke bawah. Kemudian dibuat daftar pentomino yang belum digunakan. Ketika, ada pentomino yang diletakkan ke dalam segi empat, maka pentomino tersebut dibuat dari daftar pentomino yang belum digunkan. Berikut adalah algoritma *backtracking* untuk pencarian solusi pentomino.

1. Letakkan pentomino ke-1 (Bentuk O, jenis ke-1) pada sel pertama. Sel pertama segi empat ditempati oleh sel pertama dari lima sel pentomino. Peletakan sel pentomino pada segi empat berikutnya tergantung jarak dalam (x,y) relatif dari lokasi sel pentomino tersebut terhadap sel pertama. Kemudian cek apakah terjadi pelanggaran peletakan sel, yaitu peletakan pentomino yang dilakukan melebihi batas dari segi empat ataupun peletakan yang tidak mungkin karena berhimpit dengan pentomino lain pada sel yang sama



Gambar 5. Contoh peletakan pentomino pertama pada sel

2. Apabila tidak terjadi pelanggaran dalam pemasangan pentomino, maka pentomino tersebut dianggap sudah dipakai dan dibuang dari daftar pentomino yang belum digunakan. Lalu, peletakan pentomino maju ke sel segi empat dengan nomor berikutnya yang masih kosong.
3. Apabila terjadi pelanggaran peletakan pentomino (ada sel pentomino yang berada di luar batas segi empat atau berhimpit dengan sel pentomino lain yang telah terpasang terlebih dahulu), maka nomor pentomino seperti yang tertera pada tabel dimajukan (ditambah satu). Pentomino yang sudah terpasang di sel segi empat tidak diperhitungkan pada langkah ini, jadi bisa saja nomor yang dimajukan lebih dari satu, karena nomor pentomino yang sudah terpasang tidak diperhitungkan.
4. Apabila tidak ada jenis pentomino dari angka 1 sampai 63 (tidak termasuk pentomino yang sudah terpasang) yang cocok untuk diletakkan pada sel segi empat, maka dilakukan *backtrack*. Sel yang ingin ditempati sekarang dibiarkan tetap kosong, lalu pentomino yang telah dipasang sebelumnya dicabut dan diliat nomor urutan pemasangan pentomino tersebut. Kemudian, dipasang pentomino lain yang memiliki nomor pemasangan setelah pentomino yang dicabut tersebut (satu nomor diatas jika pentomino tersebut belum dipasang pada sel segi empat).
5. Ulangi langkah 1 untuk pentomino berikutnya hingga seluruh sel pada segi empat (ada 60 sel) ditempati oleh pentomino yang berbeda bentuknya.

IV. Implementasi dan Pengujian

A. Implementasi

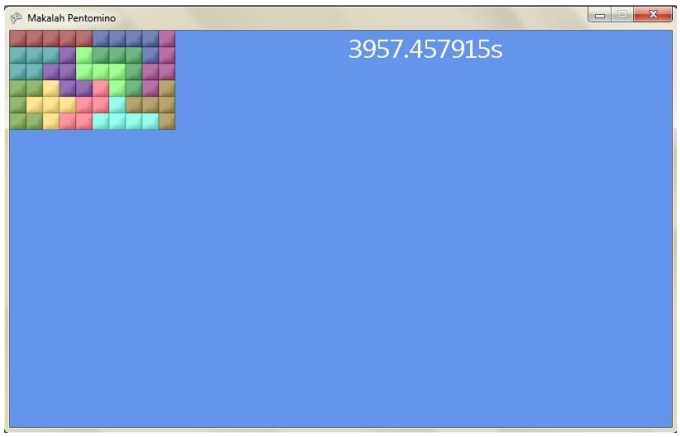
Implementasi algoritma backtrack untuk penyelesaian puzzle pentomino ini dilakukan dengan menggunakan kakas Visual Studio 2010 dengan bahasa C#. Berikut adalah *pseudo-code* algoritma *backtracking* pemasangan pentomino tersebut.

```
while (i<12)
   if (not backtrack)
      if (pentomino sekarang bisa dipasang){
          letakkan pentomino tersebut
          buang pentomino dari daftar yang belum dicabut
          i <- i + 1
          form = 0
          if (belum seluruh sel yang telah terpasang)
             cari sel segi empat kosong berikutnya
       }
       else if(pentomino sekarang telah ada di segiempat)
            cari pentomino yang belum dipasang berikutnya
       else if (belum semua jenis dari pentomino sekarang
digunakan)
           ubah jenis pentomino dengan cara pencerminan
atau rotasi
```

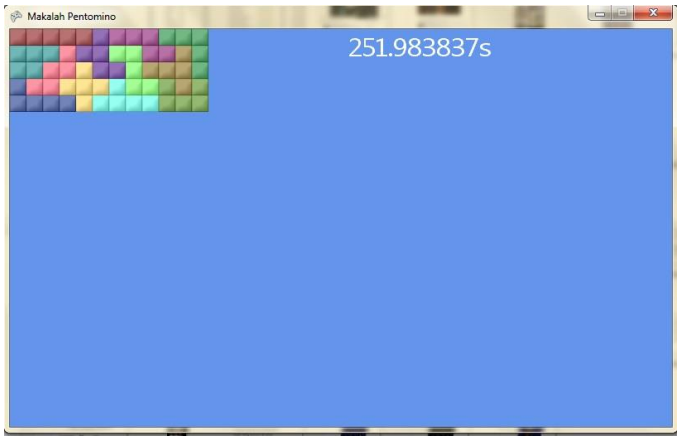
```
          else {
             form = 0
             cari pentomino yang belum dipasang berikutnya
             if (seluruh urutan pentomino telah dicoba){
                backtrack <- true
                cabut pentomino yang terakhir dipasang
                tambahkan kembali pentomino tersebut ke
daftar belum terpasang
                cari pentomino berikutnya untuk dipasang
          }
    }
    else
       backtrack <- false;
```

B. Pengujian

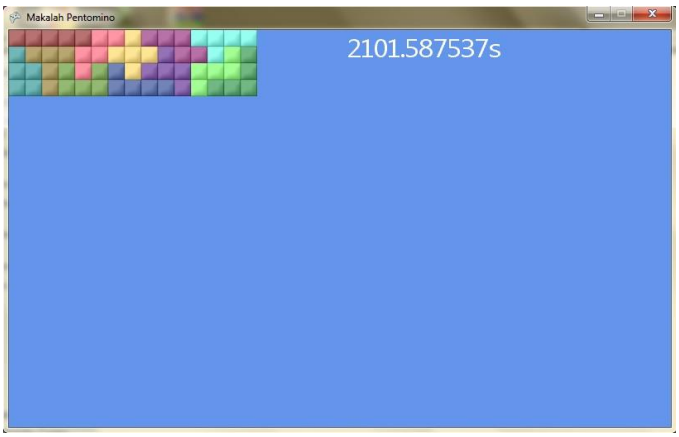
Pengujian dilakukan dengan menggunakan laptop Toshiba Core i7, RAM 8 MB dan sistem operasi windows 7. Ada 4 jenis *puzzle* segi empat berukuran 60 petak yang diujikan : ukuran 6 x 10, ukuran 5 x 12, ukuran 4 x 15 dan ukuran 3 x 20. Berikut adalah screenshot hasil pengujian tersebut.



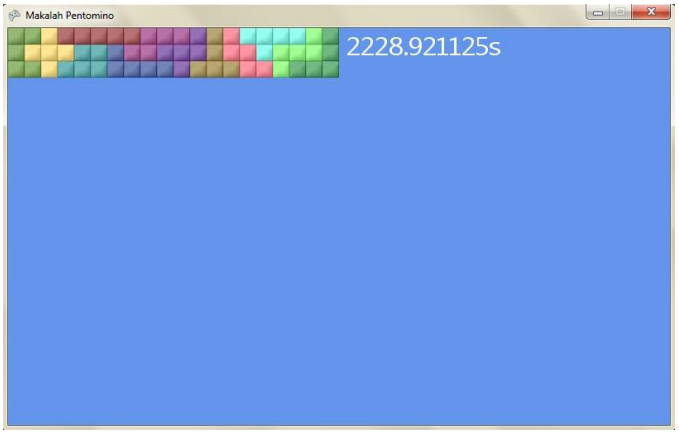
Gambar 6. Hasil pengujian puzzle 6 x 10



Gambar 7. Hasil pengujian puzzle 5 x 12



Gambar 8. Hasil pengujian puzzle 4 x 15



Gambar 9. Hasil pengujian puzzle 3 x 20

Berikut adalah hasil pengujian tersebut dalam bentuk tabel beserta waktu pencarian solusi dalam format jam:menit:detik.

Ukuran *Puzzle*	Jumlah Seluruh Solusi yang Ada	Waktu Penemuan Solusi Pertama
3 x 20	2	00:37:09
4 x 15	368	00:35:02
5 x 12	1010	00:04:12
6 x 10	2339	01:05:57

V. ANALISIS

A. Algortima Backtracking

Algoritma *backtracking* dapat digunakan untuk menyelesaikan *puzzle* pentomino. Namun, waktu yang digunakan untuk mencari satu solusi pertama cukup lama. Dibutuhkan waktu sekitar satu jam dan enam menit untuk menyelesaikan *puzzle* pentomino berukuran 6 x 10. Itu pun hanya 1 dari 2339 solusi yang tersedia saja yang ditemukan. Tentu jika ingin mencari seluruh solusi diperlukan waktu yang lebih lama lagi.

Algoritma *backtracking* banyak menghabiskan waktu pada pemilihan 1 dari 63 jenis pentomino yang sesuai ditempatkan pada sel yang tersedia. Algoritma *backtraking* tidak memakai *heuristic* untuk menentukan pentomino berbentuk apa yang sebaiknya diletakkan terlebih dahulu dalam situasi tertentu sehingga pemilihan pentomino menjadi penyebab lambatnya pencarian solusi yang dilakukan oleh algoritma *backtracking*.

B. *Kecepatan Pencarian Solusi*

Jumlah solusi yang tersedia untuk *puzzle* pentomino dengan ukuran tertentu tidak selalu mempengaruhi kecepatan pencarian solusi. Hal ini dapat dilihat pada pengujian *puzzle* berukuran 6 x 10 yang memiliki jumlah solusi paling banyak. *Puzzle* ini, malah memakan waktu yang paling banyak jika dibandingkan dengan waktu pengerjaan *puzzle* ukuran lainnya meskipun memiliki kemungkinan solusi yang lebih banyak. Kecepatan pencarian solusi lebih ditentukan oleh pemilihan pentomino awal yang digunakan. Semakin dekat pemilihan awal pentomino tersebut dengan solusi, maka akan semakin cepat solusi pentomino tersebut ditemukan.

Ukuran lebar *puzzle* yang lebih kecil juga dapat menyebabkan pencarian solusi yang lebih cepat. Hal ini dapat diperhatikan pada *puzzle* yang berukuran 3 x 20 yang melakukan pergantian pentomino dengan sangat cepat. Hal ini disebabkan lebih seringnya ditemukan pentomino yang tidak memenuhi *constraint* sehingga teknik *backtrack* pada pencarian lebih jarang digunakan. Ketidakcocokan pentomino lebih sering ditemukan di awal, sehingga waktu pencarian tidak banyak terbuang untuk pentomino yang tidak mengarah ke solusi. Meskipun begitu, karena jumlah solusi pada pentomino dengan lebar yang lebih kecil juga lebih sedikit, maka waktu yang dibutuhkan untuk menemukan solusi tetap cukup lama.

VI. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Strategi algoritma *backtracking* dapat digunakan dalam penyelesaian *puzzle* pentomino, akan tetapi dibutuhkan waktu yang cukup lama dalam pengerjaannya. Oleh karena itu, penggunaan algoritma *backtracking* dalam pencarian solusi *puzzle* pentomino masih dianggap kurang baik.

B. *Saran*

Penelitian lebih lanjut untuk penyelesaian *puzzle* pentomino dapat dilakukan dengan menerapkan Algoritma X dengan teknik Dancing Links yang diperkenalkan oleh Donald Knuth. Algoritma ini adalah sebuah improvisasi dari teknik *backtracking* dengan memanfaatkan *double linked list*.

DAFTAR PUSTAKA

[1] Weisstein, Eric W. "Polyomino." dari MathWorld--A Wolfram Web Resource. Diakses dari http://mathworld.wolfram.com/Polyomino.html pada 8 Mei 2016

[2] Goodger, David. 2015. Pentominoes: Puzzles & Solutions. Diakses dari http://puzzler.sourceforge.net/docs/pentominoes.html pada 8 Mei 2016

[3] Albrecht-Buehler, Guenter. Pentomino Wood Mosaics. Diakses dari. http://www.basic.northwestern.edu/g-buehler/pentominoes/speech.htm pada 8 Mei 2016

[4] Matuszek, David. 2002. BackTracking. Diakses dari https://www.cis.upenn.edu/~matuszek/cit594-2012/Pages/backtracking.html pada 8 Mei 2016.

[5] Munir, Rinaldi. 2015. Backtracking. Institut Teknologi Bandung. Diakses dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Algoritma%20Runut-balik%20(2015).ppt pada 8 Mei 2016.

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, 9 Mei 2016

M. Rian Fakhrusy / 13511008