LAPORAN TUGAS BESAR II

IF2211 Strategi Algoritma

Pentomino



disiapkan oleh

1. Baharudin Afif Suryanugraha (13511021)
2. Willy Fitra Hendria (13511086)
3. Dinah Kamia Ulfa (13511087)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | NomorDokumen | Total Halaman |
| *IF2211-TugasBesar2* | *20* |
| *4 November 2013* |

Daftar Isi

[1 Deskripsi Masalah 3](#_Toc371341085)

[2 Dasar Teori 4](#_Toc371341086)

[2.1 BFS 4](#_Toc371341087)

[2.2 DFS 5](#_Toc371341088)

[2.3 Backtracking 6](#_Toc371341089)

[3 Analisis Pemecahan Masalah 7](#_Toc371341090)

[3.1 BFS 7](#_Toc371341091)

[3.2 DFS 8](#_Toc371341092)

[4 Implementasi dan Pengujian 10](#_Toc371341093)

[4.1 Spesifikasi Teknis Program 10](#_Toc371341094)

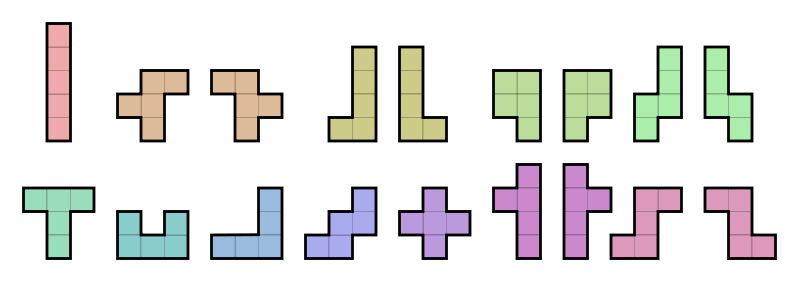
[4.2 Pengujian 19](#_Toc371341095)

[4.3 Analisis Hasil Pengujian 22](#_Toc371341096)

[5 Kesimpulan dan Saran 22](#_Toc371341097)

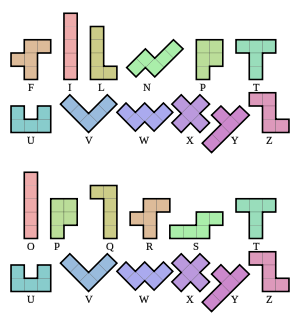
# Deskripsi Masalah

Pentomino adalah polyomino yang terdiri dari lima buah persegi berukuran sama yang terhubung sepanjang tepi mereka (koneksi orthogonal) menciptakan 12 bentuk berbeda yang digunakan sebagai potongan *puzzle* dalam matematika rekreasi (Sumber kutipan: Wikipedia).



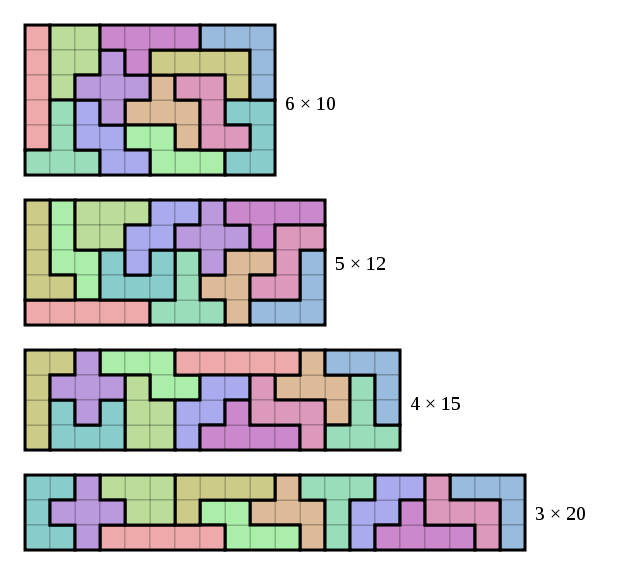
Biasanya, pentomino yang diperoleh dari refleksi atau rotasi pentomino tidak dihitung sebagai pentomino yang berbeda. F, L, N, P, Y, dan Z adalah pentomino *chiral* (tidak identik dengan *mirror image-*nya) dalam dua dimensi; dengan menambahkan refleksi mereka (F ', J, N', Q, Y ', S) menghasilkan jumlah pentomino menjadi 18. Huruf lain, I, T, U, V, W, dan X, setara dengan beberapa rotasi gambar cermin mereka. (Sumber kutipan: Wikipedia).

Masing-masing dari dua belas pentomino dapat diubinkan untuk mengisi sebuah bidang datar. Selain itu, setiap pentomino *chiral* dapat diubinkan tanpa menggunakan refleksi (Sumber kutipan: Wikipedia).



Gambar 1. Perbandingan skema pelabelan pentomino. Yang pertama adalah konvensi penamaan yang digunakan dalam artikel ini. Metode kedua adalah Conway (Sumber: Wikipedia)

Sebuah *puzzle* pentomino standar mengubinkan kotak persegi panjang dengan pentamino, yaitu menutupinya tanpa tumpang tindih dan tanpa celah. Masing-masing dari 12 pentamino memiliki luas 5 unit kuadrat, sehingga kotak harus memiliki area seluas 60 unit. Kemungkinan ukuran yang lain adalah 6 × 10, 5 × 12, 4 × 15 dan 3 × 20. (Sumber kutipan: Wikipedia).



Gambar 2. Dua belas pentamino sudah dubinkan pada kotak berukuran 6 × 10, 5 × 12, 4 × 15 dan 3 × 20 (Sumber: Wikipedia)

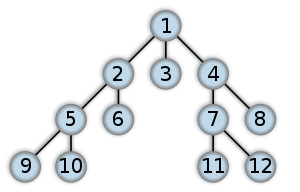
Bila diselesaikan dengan tangan mungkin membutuhkan waktu beberapa jam, namun jika diselesaikan dengan komputer akan membutuhkan pencarian dengan *backtracking*.

Masalah yang diselesaikan dalam tugas besar ini adalah membuat program yang akan menemukan solusi untuk puzzle pentomino menggunakan algoritma pemecahan BFS dan DFS, serta memanfaatkan *backtracking* dalam pencarian solusi masalahnya.

# Dasar Teori

## BFS

BFS (*Breadth First Search*) adalah algoritma yang melakukan pencarian melebar dengan mengunjungi simpul secara *preorder*, yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.



Gambar 1. Ilustrasi graf BFS

Adapun algoritma BFS adalah sebagai berikut:

1. Kunjungi simpul v
2. Kunjungi semua simpul yang bertetangga dengan simpul v terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

## DFS

DFS (*Depth First Search*) adalah algoritma yang melakukan pencarian mendalam, yaitu mulai dari mengunjungi satu simpul, kemudian mengunjungi anak dari simpul tersebut, demikian selanjutya sampai solusi ditemukan atau sampai anak simpul terakhir, dimana kemudian dilakukan *backtracking* ke simpul yang sebelumnya dikunjung, lalu mengunjungi anak simpul tersebut yang belum dikunjungi.



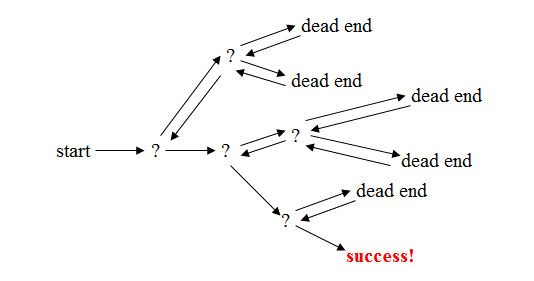
Gambar 2. Ilustrasi graf DFS

Adapun algoritma DFS adalah sebagai berikut:

1. Kunjungi simpul v
2. Kunjungi simpul w yang merupakan anak dari simpul v.
3. Ulangi DFS mulai dari simpul w.
4. Ketika mencapai simpul u sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul w yang belum dikunjungi.
5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

## Backtracking

*Backtracking*/runut balik adalah algoritma pencarian solusi yang secara bertahap membangun kandidat untuk solusi, kemudian meninggalkan kandidat parsial tersebut (backtrack) segera seteleh ditemukan bahwa kandidat tidak mungkin diselesaikan menjadi solusi yang tepat. (Sumber kutipan: Wikipedia).



Gambar 3. Ilustrasi Backtracking

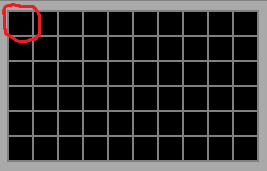
Adapun algoritma *Backtracking* adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti aturan depht-first order (DFS).
2. Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (live node).
3. Simpul hidup yang sedang diperluas dinamakan simpul-E (Expand-node).
4. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang.
5. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut “dibunuh” sehingga menjadi **simpul mati** (*dead node*).
6. Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan **fungsi pembatas** (*bounding function*).
7. Simpul yang sudah mati tidak akan pernah diperluas lagi.
8. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian *backtrack* ke simpul aras diatasnya
9. Lalu, teruskan dengan membangkitkan simpul anak yang lainnya.
10. Selanjutnya simpul ini menjadi simpul-E yang baru.
11. Pencarian dihentikan bila kita telah sampai pada *goal node*.

# Analisis Pemecahan Masalah

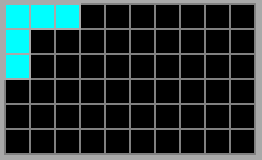
## BFS

Akar dari pohonnya adalah board awal pada kondisi yang masih kosong. Langkah pertama yang dilakukan adalah dengan mencari tempat atau sel yang masih kosong pada Board untuk menempatkan suatu pentomino. Jika tidak ada tempat yang kosong lagi, berarti solusi tercapai. Pencarian tempat kosong ini dilakukan dengan melakukan iterasi dimulai dari sel sudut kiri atas, dan dilanjutkan pada baris selanjutnya hingga baris terakhir. Jika telah mencapai baris akhir, maka pencarian berpindah ke kolom selanjutnya. Pencarian berhenti ketika sel kosong yang pertama kali ditemukan, dan itu artinya pentomino akan ditempatkan pada sel tersebut.

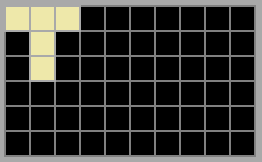


Gambar 1 Sel kosong yang pertama kali ketemu hasil pencarian

Kemudian lakukan penempatan pentomino satu per satu berikut rotate dan juga flipnya. Untuk semua penempatan yang valid dimasukkan kedalam queue. Setelah seluruh child telah di uji dan dimasukan kedalam queue, maka lakukan delete elemen pertama pada queue, dan lakukan proses yang sama pada elemen tersebut, yaitu melakukan pengujian pada seluruh child-nya dan memasukkan child-nya ke dalam queue, kemudian melakukan dequeue.



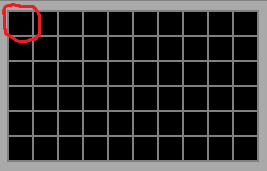
Gambar 2 Dequeue pertama, penempatan pentomino pertama, semua childnya yang valid dimasukkan ke queue



Gambar 3 Dequeue kedua, penempatan pentomino kedua, semua childnya yang valid dimasukkan ke queue

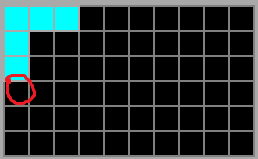
## Algoritma Backtracking

Langkah pertama yang dilakukan adalah dengan mencari tempat atau sel yang masih kosong pada board untuk menempatkan suatu pentomino. Jika tidak ada tempat yang kosong lagi, berarti solusi tercapai. Pencarian tempat kosong ini dilakukan dengan melakukan iterasi dimulai dari sel sudut kiri atas, dan dilanjutkan pada baris selanjutnya hingga baris terakhir. Jika telah mencapai baris akhir, maka pencarian berpindah ke kolom selanjutnya. Pencarian berhenti ketika sel kosong yang pertama kali ditemukan, dan itu artinya pentomino akan ditempatkan pada sel tersebut.



Gambar 4 Sel kosong yang pertama kali ditemukan hasil pencarian

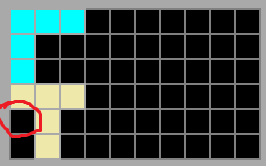
Kemudian letakkan pentomino pada sel yang telah didapatkan pada langkah sebelumnya. Penempatan dilakukan dengan memastikan sel yang masih kosong tersebut dapat terisi oleh pentomino. Disini pentomino diperiksa apakah penempatannya valid atau tidak pada board tersebut. Valid atau tidaknya disini didefinisikan apakah pentomino bertumpukan atau tidak dengan pentomino lainnya, atau pentomino keluar dari board atau tidak.



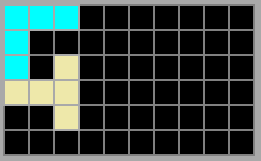
Gambar 5 Pentomino pertama yang diletakkan sudah valid dan temukan sel untuk pentomino selanjutnya

Jika penempatan suatu pentomino valid, maka pilih pentomino selanjutnya dan lakukan proses yang sama seperti pada pentomino sebelumnya, yaitu mencari posisi atau sel yang masih kosong, meletakkan pentomino tersebut, dan memeriksa apakah penempatan pentomino tersebut valid atau tidak.

Jika penempatan suatu pentomino tidak valid, maka lakukan rotate dan juga flip pada pentomino tersebut, dan jika seluruh penempatan dari seluruh kemungkinan rotate dan juga flip pada pentomino tersebut tidak ada yang valid, maka pilih pentomino selanjutnya. Jika seluruh pentomino telah dicobakan dan tidak menemukan yang valid, maka lakukan bactrack.



Gambar 6 Setelah pentomino kedua valid, tidak ada pentomino selanjutnya yang valid yang dapat ditempatkan pada sel diatas



Gambar 7 Setelah semua pentomino telah dicoba dan tidak ada yang valid, maka backtrack dilakukan

# Implementasi dan Pengujian

## Spesifikasi Teknis Program

Pada permainan pentomino ini, kami menggunakan beberapa *Class*. Namun pada penjelasan *Class* dibawah ini, kami kelompokkan terlebih dahulu seluruh *Class*-*Class* yang ada, apakah akan masuk golongan *Class* (struktur data) ataukah masuk *Interface* *Class*.

1. *Class* (struktur data)

* Pentominos

Pentominos merupakan *Class* yang dibuat untuk menangani seluruh persoalan terkait Pentomino. Jadi *Class* inilah yang mengurusi representasi bentuk-bentuk Pentomino. *Class* ini memiliki *Class* *member* sebagai berikut:

* int[,] matrix;

*Class* *member* yang berfungsi untuk merepresentasikan bentuk pentomino. *Class* *member* ini merupakan sebuah matrix 5x5.

* int urutan;

*Class* *member* yang bertugas untuk menyimpan index array pentomino diawal permaianan.

* int index;

*Class* *member* yang bertugas untuk menyimpan urutan pentomino. Misal bentuk pentomino garis lurus diassign dengan index=10. Dan karena jumlah bentuk dasar pentomino berjumlah 12, maka besar maksimum nilai *Class* *member* ini juga 12.

* int jumlahrotate;

*Class* *member* yang bertugas untuk menyimpan jumlah putaran dan flip yang dapat dilakukan oleh pentomino

* Solidbrush pentominobrush;

*Class* *member* yang bertugas untuk menyimpan warna pada pentomino.

* int turn;

*Class* *member* yang berfungsi untuk menyimpan jumlah putaran yang telah dilakukan oleh user.

* int poX,posY;

*Class* *member* yang bertugas untuk mencatat posisi pentomino secara realtime dalam arah X dan Y.

* Boolean placed;

*Class* *member* yang berfungsi untuk menyimpan nilai logika yang didapat dari posisi penomino. *Class* *member* ini akan bernilai TRUE jika pentomino telah diletakkan didalam board, dan akan bernilai FALSE jika ternyata pentomino ini belum diletakkan didalam board.

Selain *Class* *member* diatas, sebuah *Class* pastilah memiliki fungsi/prosedur. Berikut fungsi dan prosedur yang dimiliki oleh *Class* pentominos:

* Pentominos();

Merupakan sebuah konstruktor default dari *Class* Pentominos. Pada konstruktor ini, dilakukan inisialisasi niai dari *Class member* matrix dan turn.

* void setPos(int ,int ) ;

prosedur yang digunakan untuk melakukan pengesetan terhadap posisi dari objek Pentominos. Jadi prosedur ini akan mengubah nilai *Class* *member* posX,posY.

* void Createpentominos(int , int);

merupakan sebuah prosedur yang digunakan untuk membuat pentominos berdasarkan index pentomino dan urutan pentomino.

Oleh karena itu prosedur ini membutuhkan masukan 2 buah variable integer. Integer pertama digunakan sebagai index pentomino dan integer digunakan sebagai urutan dari pentomino ketika game pertama kali dijalankan.

* int getJRotate();

merupakan sebuah fungsi yang digunakan untuk mendapatkan nilai dari *Class* *member* “jumlahrotate”.

* int GetX()
* merupakan sebuah fungsi yang digunakan untuk mendapatkan nilai dari *Class* *member* “posX”.
* int GetY()

merupakan sebuah fungsi yang digunakan untuk mendapatkan nilai dari *Class* *member* “posY”.

* bool atPos(int, int)

merupakan sebuah fungsi yang digunakan untuk mengecek apakah pontomino berada pada posisi tertentu. Oleh karena itu fungsi ini membutuhkan dua buah variabel masukan bertipe integer, variabel masukan pertama merepresentasikan posisi X, dan variabel masukan kedua merepresentasikan posisi Y.

* Void Flippentomino()

Merupakan sebuah prosedur yang berguna untuk melakukan flip pada pentomino.

* Void setBackPentaminos(Pentominos)

Merupakan sebuah prosedur yang digunakan untuk mengassign bentuk pentomino ke bentuk dasar.

* Void klikkanan()

Merupakan sebuah prosedur yang mangani perubahan bentuk pada pentomino (flip dan rotate). Perubahan dapat dilakukan dengan melakukan “kilkkanan” pada sebuah objek Pentominos.

* Void Rotatepentomino()

Merupakan sebuah prosedur yang digunakan untuk menangani rotasi pada pentomino.

* Void Draw(Graphics )

Merupakan sebuah prosedur yang menangani antarmuka bentuk pentomino.

* Solver

Merupakan sebuah *Class* yang menangani permasalahan pemecahan solusi dari sebuah persoalan pentomino. Berikut *Class* *member* yang dimiliki oleh *Class* ini :

* Form f;

Merupakan sebuah *Class* *member* yang menyimpan data terkait antarmuka pada permainan ini.

* Int count;

Merupakan sebuah *Class* *member* yang berguna sebagai basis dari penyelesaian game. Permainan akan selesai ketika nilai dari *Class* *member* ini sama dengan 60.

Selain *Class* *member* diatas, sebuah *Class* pastilah memiliki fungsi/prosedur. Berikut fungsi dan prosedur yang dimiliki oleh *Class* Solver:

* Solver(Form )

Merupakan sebuah konstruktor *Class* Solver. Pada konstruktor ini hanya melakukan pengesetan *Class* *member* Form dengan nilai Form yang merupakan masukan dari konstruktor ini.

* int[] searchPosisiPlace(int )

Fungsi ini menghasilkan posisi dimana sebuah pentomino dapat diletakkan ke dalam board.

* Void place\_pentomino(Pentominos, int ,int)

Merupakan sebuauh prosedur untuk meletakkan sebuah bentuk pentomino ke posisi P(X,Y). Oleh karena itu prosedur ini membutuhkan masukan objek dari Pentominos dan dua buah integer.

* Boolean solvedfs()

Merupakan sebuah fungsi yang digunakan untuk menyelesaikan persoalan dengn metode dfs. Jika solusi ditemukan maka fungsi ini akan mengembalikan nilai TRUE, namun jika salah fungsi ini akan mengembalikan nilai FALSE.

* Boolean solvedbfs()

Merupakan sebuah fungsi yang digunkan untuk menyelesaikan persoalan dengan metode bfs. Jika solusi ditemukan maka fungsi ini akan mengembalikan nilai TRUE, namun jika salah fungsi ini akan mengembalikan nilai FALSE.

* Void placepentominosfromstate(Queue<int>[])

Merupakan sebuah prosedur untuk menempatkan pentomino dari state BFS ke dalam board.

* Board

Board merupakan sebuah *Class* yang mengurusi seluruh permasalahan board game. Dari pembuatan board, pengecekan board dan lain-lain. *Class* Board ini memiliki beberapa *Class* *member*, yaitu

* Int Cols

Merupakan sebuah *Class* *member* yang merepresentasikan jumlah kolom yang dimiliki oleh board.

* Int Rows

Merupakan sebuah *Class* *member* yang merepresentasikan jumlah baris yang dimiliki oleh board.

* Int[,] matrix;

Merupakan sebuah *Class* *member* yang merepresentasikan board dalam bentuk matrix. *Class* *member* inilah yang diubah-ubah nilainya ketika ada pentomino yang terpasang di board ini.

* Int[,] Tmatrix;

Merupakan sebuah *Class* *member* yang merepresentasikan board dalam bentuk matrix ketika board belum terisi apapun. Jadi *Class* *member* ini akan digunakan ketika *Class* *member* “matix” membutuhkan nilai dimana seluruh cell pada kondis belum terisi.

* Int posBoardX;

Merupakan sebuah *Class* *member* yang digunakan untuk menyimpan posisi cell pojok kiri atas dari board dalam arah sumbu X.

* Int typeBoard;

Merupakan sebuah *Class* *member* yang berfungsi untuk menyimpan tipe board dalam bentuk integer.

* Int posBoardY;

Merupakan sebuah *Class* *member* yang digunkan untuk menyimpan posisi cell pojok kiri atas dari board dalam arah sumbu Y.

Selain *Class* *member* diatas, sebuah *Class* pastilah memiliki fungsi/prosedur. Berikut fungsi dan prosedur yang dimiliki oleh *Class* Board:

* Board()

merupakan sebuah konstruktor default dari *Class* Board.

* Board(Board )

Merupakan sebuah copy konstruktor dari *Class* Board.

* Void createBoard(int)

Merupakan sebuah prosedur yang digunakan untuk membuat sebuah board berdasarkan jenisnya, dimana jenis-jenis board direpresentasikan dalam varibel yang bertipekan integer. Oleh karena itu prosedur ini membutuhkan masukan sebuah variabel bertipe integer.

* Void createBoardFromEditor(Editor ed)

Merupakan sebuah prosedur yang digunakan untuk membuat sebuah board yang tergantung dari masukan yang berupa objek Editor. Jadi prosedur ini akan membuat board yang dapat dipasangkan pentomino diatasnya, sesuai dengan bentuk objek editor.

* Boolean isFull()

Merupakan sebuah fungsi yang digunakan untuk mengecek apakah board telah terisi semua. Jika board telah terisi semua, maka fugsi ini akan menghasilkan nilai TRUE. Namun jika board belum terisi semua maka fungsi ini akan mengembalikan nilai FALSE.

* Boolean isCollisionOnBoard()

Merupakan sebuah fungsi yang dugunakan untuk mengecek apakah terdapat sebuah collision (tumpukan pentomino) pada board.

* Void initinit()

Merupakan sebuah prosedur yang digunakan untuk mengisi nilai *Class* *member* “matrix” sama dengan nilai *Class* *member* “Tmatrix”

* Boolean isValid(int,int. Pentominos)

Merupakan sebuah fungsi yang digunakan untuk mengecek apakah pentomino diletakkan diatas wilayah board. Nilai masukan dua buah integer merupakan posisi pentomino pada board. Jika ternyata pentomino diletakkan pada wilayah board, maka fungsi ini akan menghasilkan nilai TRUE, namun jika tidak, fungsi ini akan menghasilkan nilai FALSE.

* Boolean setMatrixBoard(int, int, Pentominos)

Merupakan sebuah fungsi yang menggabungkan seluruh prosedur validasi pemasangan pentomino ke dalam board. Jika seluruh prosedur validasi bernilai TRUE, maka fungsi ini akan menghasilkan nilai TRUE. Namun jika ada fungsi validasi yang menghasilkan nilai FALSE, maka fungsi ini akan langsung mengembalikan nilai FALSE.

* Void delMatrixBoard(int, int, Pentominos)

Merupakan sebuah prosedur untuk mengurangi nilai-nilai pada cell yang dimiliki oleh *Class* *member* “matrix” ketika pentomino dikeluarkan dari board.

* Boolean isNull()

Merupakan sebuah fungsi yang melakukan validasi apakah boardnya telah diinisialisasi ataukah belum. Jika ternyata telah terinisialisasi maka fungsi ini akan mengembalikan nilai TRUE. Namun jika ternyata belum terinisialisasi maka fungsi ini akan mengembalikan nilai FALSE.

* Void Draw(Graphics)

Merupakan sebuah prosedur yang digunakan untuk menampilkan board kedalam antarmuka permainan (form).

* Editor

Merupakan sebuah *Class* bertugas untuk membuat sebuah bentuk board secara manual. *Class* ini memiliki beberapa *Class* *member*, yaitu

* Int Cols

Merupakan sebuah *Class* *member* yang merepresentasikan jumlah kolom yang dimiliki oleh board manual.

* Int Rows

Merupakan sebuah *Class* *member* yang merepresentasikan jumlah baris yang dimiliki oleh board manual.

* Int posBoardX;

Merupakan sebuah *Class* *member* yang digunakan untuk menyimpan posisi cell pojok kiri atas dari board manual dalam arah sumbu X.

* Int posBoardY;
* Merupakan sebuah *Class* *member* yang digunakan untuk menyimpan posisi cell pojok kiri atas dari board manual dalam arah sumbu Y.
* Int[,] matrix;

Merupakan sebuah *Class* *member* yang merepresentasikan board manual dalam bentuk matrix. *Class* *member* inilah yang diubah-ubah nilainya ketika ada pentomino yang terpasang di board ini.

Selain *Class* *member* diatas, sebuah *Class* pastilah memiliki fungsi/prosedur. Berikut fungsi dan prosedur yang dimiliki oleh *Class* Editor:

* Editor()

Merupakan sebuah konstruktor default pada kelas ini.

* Void createEditorBoard()

Merupakan sebuah prosedur yang digunkan untuk membuat kotak board kosong, board kosong ini belum terdapat cell yang dapat dipasang pentomino diatasnya.

* Void atPos(int, int. int)

Merupakan sebuah prosedur untuk mengubah nilai cell dengan nilai tertentu (1 merepresentasikan cell tidak dapat digunakan untuk meletakkan pentomino diatasnya, dan 0 dapat untuk meletakkan pentomino). Nilai dua buah masukan integer pada prosedur ini berguna untuk menentukan cell mana yang akan di set nilaiya.

* Int isEditorValid()

Merupakan sebuah fungsi yang digunakan untuk mengembalikan jumlah cell pada board manual dengan kriteria nilai cell nya sama dengan 0 (mampu diletakkan pentomino diatasnya).

* Void Draw(Graphics)

merupakan sebuah prosedur untuk menampilkan editor board pada layar antarmuka permainan.

1. *Interface* *Class*

* Form

Fungsi dari kelas ini ialah untuk menangani permasalahan terkait *interface* *windows form*. Jadi kelas ini dipenuhi dengan *Class member* yang merupakan antarmuka pada permainan ini. Seperti

* DateTime X;

Yang merupakan *Class member* *member* yang menangai masalah waktu (waktu eksekusi).

* Button X;

Merupakan *Class* *member* yang menangai seluruh permasalahan tombol tombol yang ada pada antarmuka. Seperti tombol “PAUSE”, “STOP”, dll.

* Boolean X;

Merupakan *Class* *member* yang menampung semua logika (true/false) yang ada pada game ini. Semisal jika ingin menunda proses pencarian solusi yang sedang dilakukan oleh komputer, user dapat melakukannya dengan menekan tombol “PAUSE”. Setelah tombol tersebut ditekan maka variabel bertipekan boolean yang bernama “PAUSE” akan bernilai TRUE.

* int X;

Merupakan *Class* *member* yang menampung nilai-nilai bertipekan integer yang akan digunakan pada pembentukan antarmuka. Sebagai contoh ‘int delayKecepatan’, *Class member* ini digunakan untuk menampung besarnya delay untuk setiap proses mecoba pada tiap-tiap kemungkinan bentuk pentomino. Karena pada permainan ini user hanya disetting untuk memilih antara delay cepat(0ms), lambat (1000ms) atau sedang(500ms), maka pada antarmuka permainan ini tidak ada opsi untuk memasukkan nilai (integer) dari delay yang user kehendaki.

* Solver X;

Merupakan *Class* *member* yang digunakan untuk menangani penyelesaian terhadap permainan pentomino ini. Sebagai contoh ‘Solver solve’. Jika ternyata user menginginkan dilakukan penyelesaian game oleh computer dengan menggunakan BFS, maka *Class* *member* inilah yang mampu melakukannya, yaitu dengan menggunakan fungsi solve.solvedfs().

* Board board;

Merupakan sebuah *Class* *member* dari *interface* *Class* Form yang bertugas untuk menampung data-data pada objek *Class* board. Sehingga bentuk board yang ditampilkan pada antarmuka permainan ini ditentukan oleh *Class* *member* ini.

* Pentominos draggedpentomino;

Sebuah *Class* *member* yang merupakn objek dari *Class* Pentominos. Nilai dari *Class* *member* ini tergantung pada objek pentomninos yang digerakkan oleh user.

* Pentominos[] pentomino;

Merupakan sebuah *Class* *member* yang berfungsi untuk menampung seluruh objek pentomino yang terdapat pada antarmuka permainan ini bisa saat pentomino dalam keadaan bentuk dasar, atau saat pentomino pada saat bentuk yang telah dirotasi atau di flip.

* Pentominos[] initialpentomino;

Merupakan sebuah *Class* *member* yang berfungsi untuk menampung seluruh objek pentomino yang terdapat pada antarmuka game ketika permainan baru saja dijalankan (bentuk dasar).

* Editor editor;

*Class* *member* ini berfungsi untuk menampung data-data yang terdapat pada objek *Class* Editor. Sehingga ketika ingin mengubah board manual, maka kita bisa melakukannya dengan mengubah nilai-nilai dari *Class* *member* ini.

* System.ComponentModel.IContainer components;

Merupakan *class member* yang digunakan untuk berisi dan mengorganisasi komponen-komponen yang ada pada antarmuka permainan.

* System.Windows.Forms.MenuItem X;

Merupakan sebuah *Class* *member* yang digunakan untuk membentuk menu item pada antarmuka permainan.

* System.Windows.Forms.Timer X;

Meruapakan sebuah *Class* *member* untuk membentuk waktu pada antarmuka permainan.

* System.Media.SoundPlayer X;

Merupakan sebuah *Class* *member* yang digunakan sebagai sumber suara (source audio) pada pemainan ini.

* Bitmap image;

Merupakan *Class* *member* yang menangani gambar background dari antarmuka.

Selain *Class* *member* diatas, sebuah *Class* pastilah memiliki fungsi/prosedur. Berikut fungsi dan prosedur yang dimiliki oleh *interface* *Class* Form:

* Form()

Merupakan sebuah konstruktor deault dari kelas ini.

* Void OnPaint(System.Windows.Forms.PaintEventArgs)

Merupakan sebuah prosedur untuk menampilkan seluruh elemen yang ada pada antarmuka, seperti menampilkan board (dengan perintah board.Draw()).

* Void initializeComponent()

Merupakan sebuah prosedur yang digunakan untuk menginisialisasi seluruh *Class* *member*. Seperti menginisialiasi menu item.

* Void Init()

Merupakan sebuah prosedur yang digunakan untuk menginisialisasi form. Semisal meng-*assign* bentuk pentomino ke posisi awal (bentuk dasar).

* Void frmPentominos\_mouseDown(object, System.Windows.Forms. MouseEventArgs )

Merupakan sebuah prosedur yang digunakan untuk mengurusi event ketika user melakukan klik pada mouse. Prosedur ini memiliki dua buah masukan, masukan pertama (objek) digunakan untuk menentukan mana objek yang dikenai kerja, dan kemudian masukan kedua menentukan event (mousDown) yang dikenai ke objek. Semisal objek pentomino di kenai kerja klik kanan, maka bentuk pentomino akan berubah.

* Void frmPentominos\_MouseMove(object, System.Windows.Forms. MouseEventArgs)

Merupakan sebuah prosedur yang digunakan untuk mengurusi event ketika user melakukan klik pada mouse. Prosedur ini memiliki dua buah masukan, masukan pertama (objek) digunakan untuk menentukan mana objek yang dikenai kerja, dan kemudian masukan kedua menentukan event (mouveMove) yang dikenai ke objek. Semisal objek pentomino dipilih dan dikenai kerja mouseMove, posisi pentomino akan berubah sesuai dengan posisi mouse.

* Void isFinish()

Merupakan sebuah prosedur yang digunakan untuk menentukan apakah permainan telah selesai ataukan belum.

* Void TimerElapsed\_Tick(object, System.EventArgs)

Merupakan sebuah prosedur untuk menghandle nilai-nilai waktu yang akan ditampilkan pada antarmuka permainan.

* Void menuItem5\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user meng-klik menu editor

* Void btn2\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada tombol “STOP”

* Void btn3\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada tombol “PAUSE”

* Void btn4\_Click(object, System.Event.Args)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada tombol “HIDE” or “SHOW”

* Void btn\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada tombol “SET THIS!! ”

* Void menucarapakai\_Click(obejct,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “Cara Penggunaan ”

* Void menuabout\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “About”

* Void menuplayer\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “Player”

* Void menulambat\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “lambat”

* Void menusedang\_Click(object, System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “sedang”

* Void menucepat\_Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “Cepat”

* Void menu3x20\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 4 (3 x 20)”

* Void menu6x10\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 1 (6 x 10)”

* Void menu5x12\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 2 (5 x 12)”

* Void menu4x15\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 3 (4 x 15)”

* Void menuTR\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 5 (Flip T)”

* Void menuDiamond\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 6 (Diamond)”

* Void menuBolaBolong\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “level 7 (Holey Oval)”

* Void menuDFS\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “DFS”

* Void menuBFS\_ Click(object,System.EventArgs)

Merupakan sebuah prosedur yang digunakan untuk menghandle event ketika user melakukan klik pada item menu “BFS”

* Void randomPentominos()

Merupakan sebuah prosedur yntuk merandom inisialisasi pentomino.

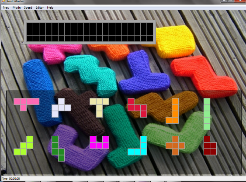
* Boolean pentominoCollision(Pentominos, int)

Merupakan sebuah fungsi yang berguna untuk mengecek apakah objek pentomino masukan bertumpukan dengan pentomino lainnya. Dan nilai masukan integer merupakan index objek pentomino yang akan dipasang. Nilai masukan integer ini berfungsi untuk mencegah objek pentomino masukan dibandingkan dengan objek pentomino masukan.

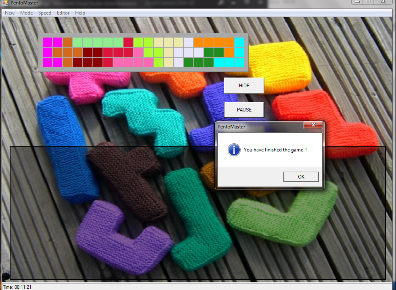
## Pengujian

1. **Menggunakan Algoritma Backtracking**

* Level 4 (3 x 20)

1. (b)

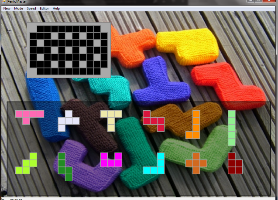
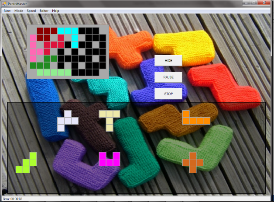


(c)

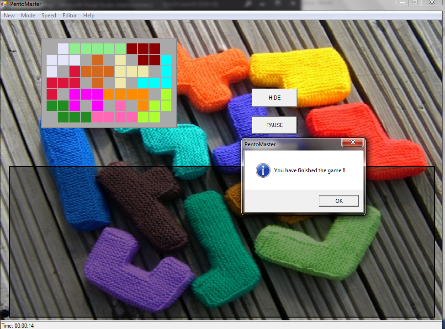
Screenshot eksekusi (waktu eksekusi 00:11:21)

1. Awal
2. Proses
3. Selesai

* Level 7 (Holey Oval)

1. (b)

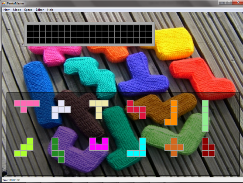
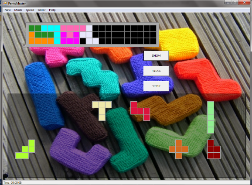


(c)

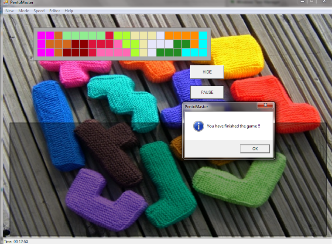
Screenshot eksekusi (waktu eksekusi 00:00:14)

1. Awal
2. Proses
3. Akhir
4. **Menggunakan Algoritma BFS**

* Level 4 (3 x 20)

1. (b)

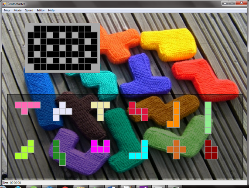
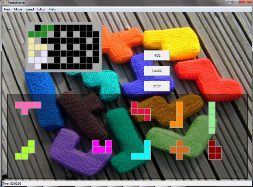


(c)

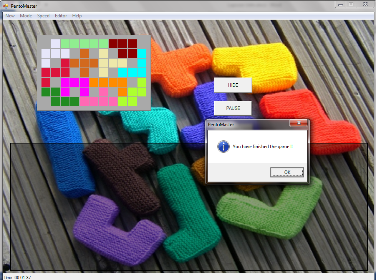
Screenshot eksekusi (waktu eksekusi 00:12:53)

1. Awal
2. Proses
3. Selesai

* Level 7 (Holey Oval )

1. (b)



(c)

Screenshot eksekusi (waktu eksekusi 00:01:37)

1. Awal
2. Proses
3. Selesai.

## Analisis Hasil Pengujian

Pencarian solusi pentomino berhasil ditemukan dengan algoritma backtracking maupaun algoritma BFS. Untuk pentomino level 4, waktu yang dibutuhkan algoritma backtracking adalah 00:11:21, sedangkan untuk algoritma BFS adalah 00:12:53. Untuk pentomino level 7, waktu yang dibutuhkan algoritma backtracking adalah 00:00:14, sedangkan untuk algoritma BFS adalah 00:01:37. Dari dua contoh kasus diatas, terlihat bahwa algoritma backtracking lebih cepat daripada algoritma BFS.

# Kesimpulan dan Saran

Solusi permainan pentomino dapat ditemukan dengan meggunakan algoritma backtracking dan BFS. Pencarian solusi dengan algoritma backtracking relatif lebih cepat dari pada dengan algoritam BFS, karena anak yang direntangkan untuk setiap node pada algoritma BFS berjumlah sangat banyak dan letak solusi terdapat pada node terdalam (ketika level kedalaman sama dengan 12). Selain waktu pencarian dengan algoritma yang cenderung lebih lama, pencarian solusi dengan algoritma ini akan sangat memakan memori dan sering terjadi kasus *out of memory*. Jadi kelompok kami menyarankan agar pencarian solusi untuk permainan ini menggunakan algoritma backtracking.

# Referensi

[1] http://puzzler.sourceforge.net/docs/pentominoes.html. (1/11/2013 23.59)

[2] http://www.mathematische-basteleien.de/pentominos.htm (1/11/2013 20.00)

[3] http://www.codeproject.com/Articles/1971/Pentominos-A-C-implementation-of-the-famous-puzzle (2/11/2013 21.00)

[4] http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch19.html (2/11/2013 19.00)

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014/Algoritma%20Runut-balik%20(2013).ppt (29/10/2013 20.00)