RANCANG BANGUN APLIKASI GAME EDUKASI SLIDING PUZZLE MENGGUNAKAN ALGORITMA ITEERATIVE DEEPENING SEARCH

Nur Rahmadani, Marla Sheilamita Shalin Pieter

Program Studi Teknik informatika,
Fakultas Ilmu Komputer dan manajemen
Universitas sains dan teknologi jayapura

ABSTRAK - *Game komputer adalah aplikasi perangkat lunak yang digunakan oleh pengguna. Berbagai macam variasi dan penampilan menarik, termasuk perangkat lunak permainan komputer yang diminati oleh masyarakat. Puzzle adalah game yang menggunakan teknik pencarian (searching). Teknik pencarian dapat dilakukan dengan menggunakan algoritma untuk menemukan solusi atau masalah. Banyak teknik pencarian dapat dilakukan dan teknik ini harus dipilih berdasarkan kriteria masalah yang dihadapi dan tingkat kebutuhan yang harus dipenuhi. Sebagai salah satu teknik pencarian yang banyak digunakan adalah pencarian mendalam berulang (IDS). Pada implementasi, teknik pencarian akan kasus penyelesaian IDS diambil pada masalah Puzzleberukuran 3x3, 4x4 dan 5x5. Penelitian ini memanfaakan Kelebihan Algoritma Iteerative Deepening search (IDS) yang diterapkan dalam merancang sebuah* Game Sliding Puzzle*Sebagai Media*Pembelajaran atau edukasi*. Hasil penelitian iniberupa sebuah aplikasi game edukasi sliding puzzle yang menggunakan Algoritma Iteerative DeepeningSearch sebagai solusi pencarian terhadap Game Sliding Puzzle*.

Kata Kunci* : *Sliding Puzzle, IDS, BFS, DFS, Algoritma

1. Pendahuluan

1.1 Latar Belakang

Game edukasi merupakan sebuah permainan yang telah dirancang untuk mengajarkan pemainnya tentang topik tertentu, memperluas konsep, serta membantu dalam belajar keterampilan karena dapat memacu otak untuk menemukan logika penyelesaian *game*. Munculnya berbagai macam game, termasuk game edukasi juga dipengaruhi oleh semakin berkembangnya teknologi. Salah satu game edukasi yang berkembang yaitu *game sliding puzzle*. *Game sliding puzzle* merupakan bentuk permainan yang menantang daya kreatifitas dan memacu logika pemain lebih mendalam untuk mencoba memecahkan masalah. Pada dasarnya dalam menyelesaikan *game sliding puzzle,* apabila semakin sulit tingkatan *puzzle* dan acakan susunan *puzzle*, tentu akan semakin lama waktu yang dibutuhkan. Proses memecahkan susunan *puzzle* tersebut bahkan dalam kondisi tertentu bisa menemui jalan buntu dalam permainannya sehingga membuat *game sliding puzzle* tidak dapat terselesaikan.

Game sliding puzzle merupakan salah satu implementasi dari kecerdasan buatan. Banyak algoritma pencarian yang dapat diterapkan dalam penyelesaiannya salah satunya adalah dengan menggunakan algoritma Algoritma *Iteerative Deepening Search*. Algoritma *Iteerative Deepening Search*merupakan bagian dari teknik *Breadth First Search* (BFS) dan *Depth First Search* (DFS), dimana algoritma ini menggabungkan manfaat BFS dalam hal teknik pencarian yang lengkap serta *optimal* dan keunggulan DFS dalam hal kompleksitas ruang atau tidak membutuhkan banyak memori.

Kelebihan Algoritma *Iteerative Deepening search* ini akan diterapkan dalam merancang sebuah *Game Puzzle*Sebagai Media *Pembelajaran atau edukasi dimana algoritma ini diharapkan* dapat menemukan langkah yang *optimal* sehingga permainan dapat terselesaikan.

1.2 Rumusan Masalah

Masalah yang dibahas dalam penelitian akan dilakukan berdasarkan latar belakang yaitu: Bagaimana menerapkan algoritma pencarian dengan teknik*Iteerative Deepening Search* pada *game sliding puzzle ?*

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini yaitu merancang bangun sebuah aplikasi*game sliding puzzle* dengan menggunakan Algoritma *Iteerative Deepening Search* sebagai solusi pencarian terhadap *game puzzle*.

1.4 Metode Penelitian

Metode penelitian ini terdapat beberapa tahapan yang dilakukan atara lain :

1.4.1 Metode pengumpulan data

1.4.1.1 Observasi

Sumber data yang diperoleh penulis berasal dari hasil observasi permainan *sliding puzzle* manual, aplikasi sliding puzzle berbasis webite, maupun aplikasi *sliding puzzle* yang dimainkan di *Smarthphone*.

1.4.1.2 Studi Pustaka

Melakukan pencari bahan sebagai pertimbangan dengan mencari dan memperoleh data-data yang diperlukan dari materi perkuliahan, internet, jurnal dan berbagai buku-buku yang berhubungan dengan proses pembuatan aplikasi game edukasi *sliding puzzle* menggunakan algoritma *iteerative deepening search* .

1.4.2 Selanjutnya melakukan analisa terhadap beberapa aplikasi permainan untuk mengetahui unsur – unsur yang dapat diterapkan pada permainan yang akan dibuat.

1.4.3 Melakukan proses perhitungan dengan menggunakan Algoritma *Iteerative Deepening Search* agar dapat diterapkan pada aplikasi *Sliding Puzzle*.

1.4.4 Melakukan perancangan dengan menggunakan Data *Flow Diagram* dan Desain Input/Output.

1.4.5 Pembuatan aplikasi dan pengujian terhadap aplikasi yang telah dibuat.

2. Tinjauan Pustaka

2.1 *GamePuzzle*

Riris D. Lomo dalam buku *Game sliding puzzle* (2012; 1) menyatakan bahwa *Game slide puzzle* merupakan permainan menyusun potongan gambar dengan aturan sebuah potongan hanya dapat dipindahkan dengan menggesernya ke ruang kosong (blank tile). *Puzzle* ini merupakan jenis *puzzle* yang memiliki tingkat kesulitan dalam menyelesaikan masalahnya sangat tinggi dibandingkan jenis *puzzle* lain. Umumnya orang yang memainkan *puzzle* ini butuh waktu lama dalam menyelesaikan permainannya. Hal ini disebabkan karena pada *slidepuzzle* tidak ada informasi tambahan yang dimiliki untuk membantu melakukan pencarian solusi, sehingga saat proses penyusunan potongan-potongan *puzzle* terjadi susunan *puzzle* semula.

2.2 Linear Congruent Method

Dirk P. Kroese dalam buku *Monte Carlo Methods* (2011; 12) menyatakan bahwa *Linear Congruent Method* (LCM) adalah generator nomor acak dari bentuk Algoritma, dengan state St = Xt 2 {0, . . . ,m−1} untuk suatu bilangan bulat positif m disebut modulus, dan state transisi.

$$Xt = (aXt-1 + c) \bmod m , \quad t = 1, 2, \ldots n \qquad (2,1)$$

Dimana multiplier dan selisih c adalah bilangan bulat. Menerapkan operator modulom, dan sisanya diambil sebagai nilai Xt. Perhatikan bahwa *multiplier* dan incre-ment dapat dipilih dalam set {0,.....,m−1}. Ketika c = 0, generator kadang-kadang disebut generator congruential perkalian. Kebanyakan implementasi yang ada dari LCM adalah bentuk ini secara umum kenaikan tidak memiliki dampak besar pada kualitas dari LCM.

2.3 Iteerative Deepening Search

Suyanto dalam buku *Artificial Intelligence Searching, Reasoning, Planing dan Learning* (2014; 21) menyatakan bahwa *Interative-Deepening Search* (IDS) merupakan metode yang menggabungkan kelebihan BFS (*Complete* dan *Optimal*) dengan kelebihan DFS (*Space complexity* rendah atau membutuhkan sedikit memori).

IDS melakukan pencarian secara iteratif menggunakan penelusuran *Depth-Limited Search* (DLS) dimulai dengan batasan *level* 0. Jika belum ditemukan solusi, maka dilakukan iterasi ke-2 dengan batasan *level* 1. Demikian seterusnya sampai ditemukan solusi. Untuk mempercepat proses pencarian, kita bisa menggunakan teknik *parallel processing* (menggunakan lebih dari satu processor).

3. Hasil Dan Pembahasan

3.1 Hasil

3.1.1 Pengacakan Menggunakan Linear Congruent Method (LCM)

Contoh Kasus :

pengacakan *Linear Congruent Method* (LCM) pada *puzzle* 3x3 (9 kotak/8angka).
misalnya a= 7 , c = 5, m = 9, xn = 12
penyelesaian :

x(1) = (7 (7) + 5)mod 9 = 8
x(2) = (7 (5) + 5)mod 9 = 7
x(3) = (7 (7) + 5)mod 9 = 0
x(4) = (7 (0) + 5)mod 9 = 5
x(5) = (7 (5) + 5) mod 9 = 4
x(6) = (7 (4) + 5) mod 9 = 6
x(7) = (7 (6) + 5) mod 9 = 2
x(8) = (7 (8) + 5) mod 9 = 1
x(9) = (7 (7) + 5) mod 9 = 3

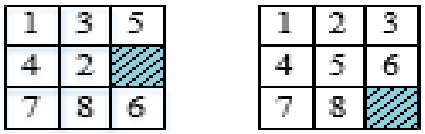
Maka bilangan acak yang dibangkitkan adalah
8 7 0 5 4 6 2 1 3
Sehingga potongan *puzzle* yang teracak akan terbentuk seperti dibawah ini

8	7	0
5	4	6
2	1	3

3.1.2 Penyelesaian Menggunakan Menggunakan Algoritma IDS

Contoh kasus Implementasi *Puzzle* 3x3

Terdapat sebuah *puzzle* 3x3 dengan kondisi yang belum terurut dengan tepat dan akan diselesaikan menggunakan *Algoritma Iterative Deepning Search* (IDS) dengan kondisi awal dan *goal*nya seperti pada gambar di bawah ini :

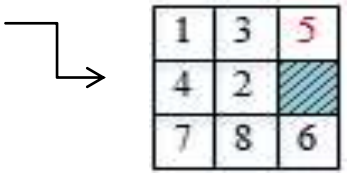


State Awal **Goal**

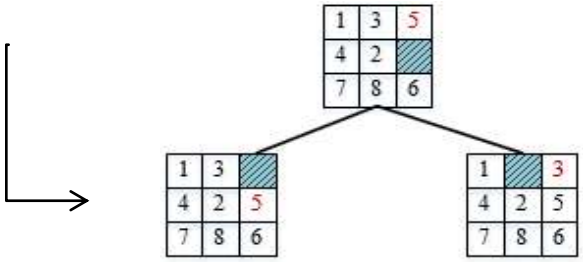
Penyelesaian permainan *slidingpuzzle* 3x3 menggunakan algoritma IDS denganmetode pencarian secara *iterative* terdapat 3 kemungkinan untuk dapat mencapai *goal* atau solusi dari permasalahan.

3.1.2.1 Kemungkinan ke-1 dengan menggeser angka 5 ke posisi bawah dan kotak kosong pada posisi atas.

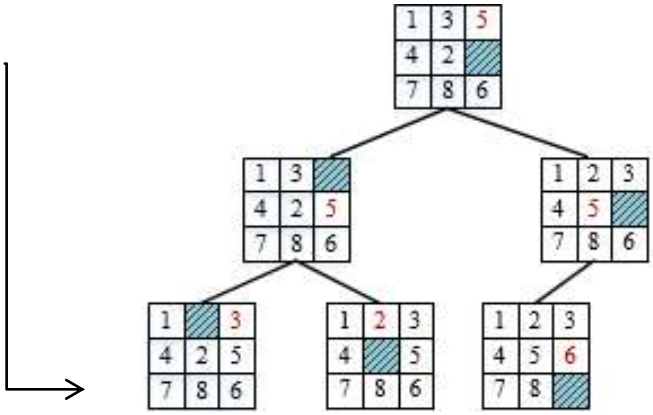
Limit 0



Limit 1



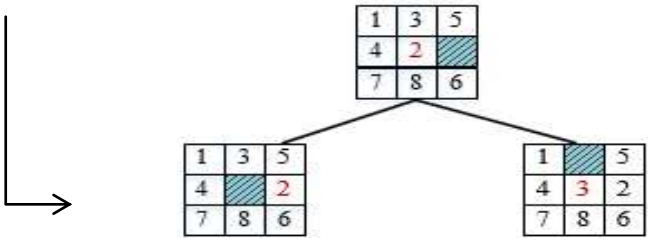
Limit 2



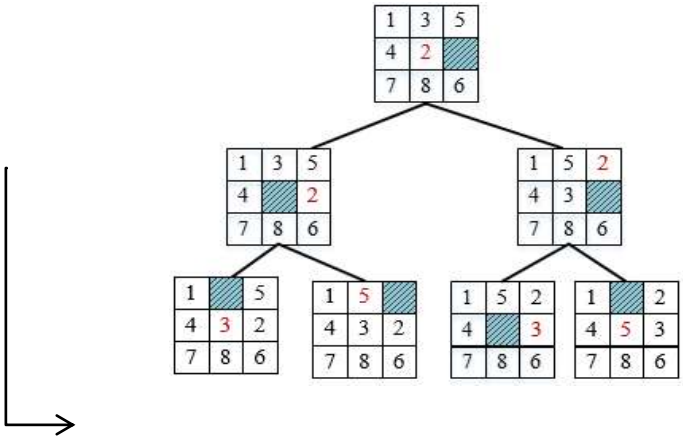
Kemungkinan ke-2 dengan mengeser angka 2 ke posisi kanan dan kotak kosong pada posisi kiri → ***Goal***

Limit 0

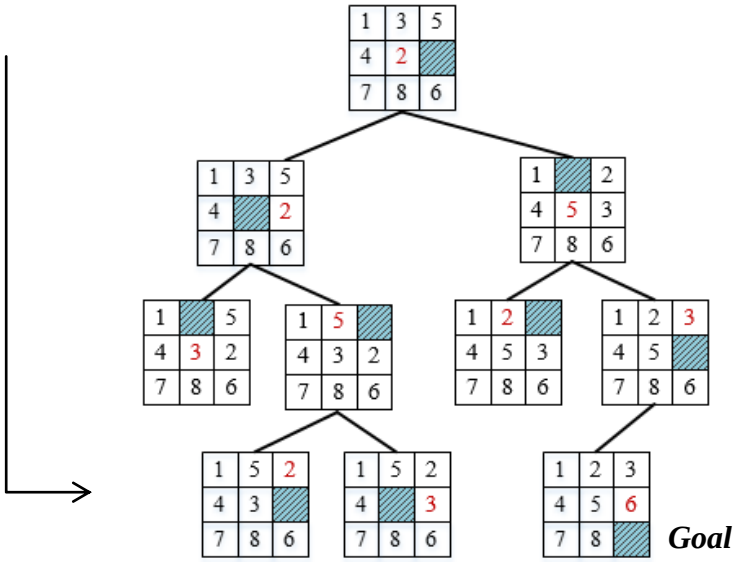
Limit 1



Limit 2



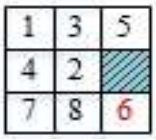
Limit 3



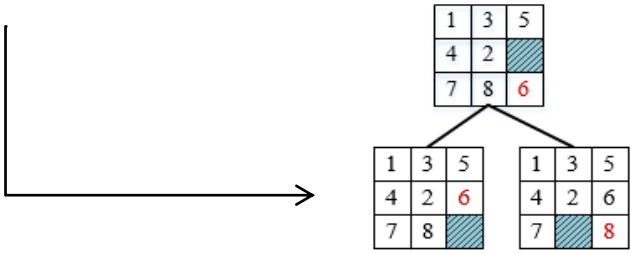
Kemungkinan ke-3 dengan mengeser angka 6 ke posisi atas dan kotak kosong pada posisi bawah

Goal

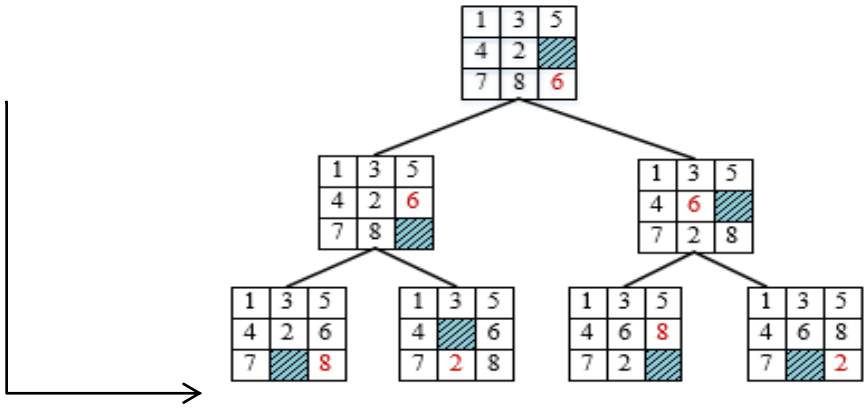
Limit 0



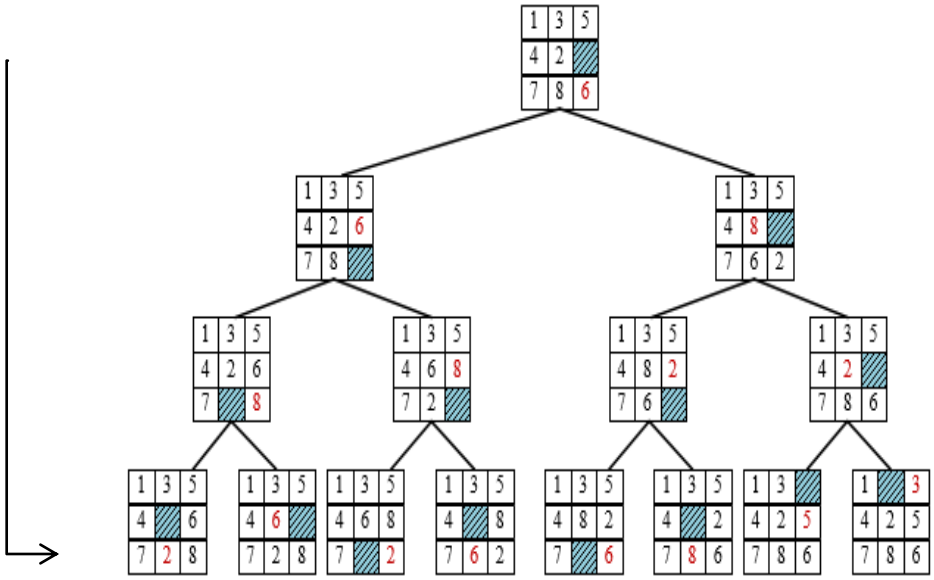
Limit 1



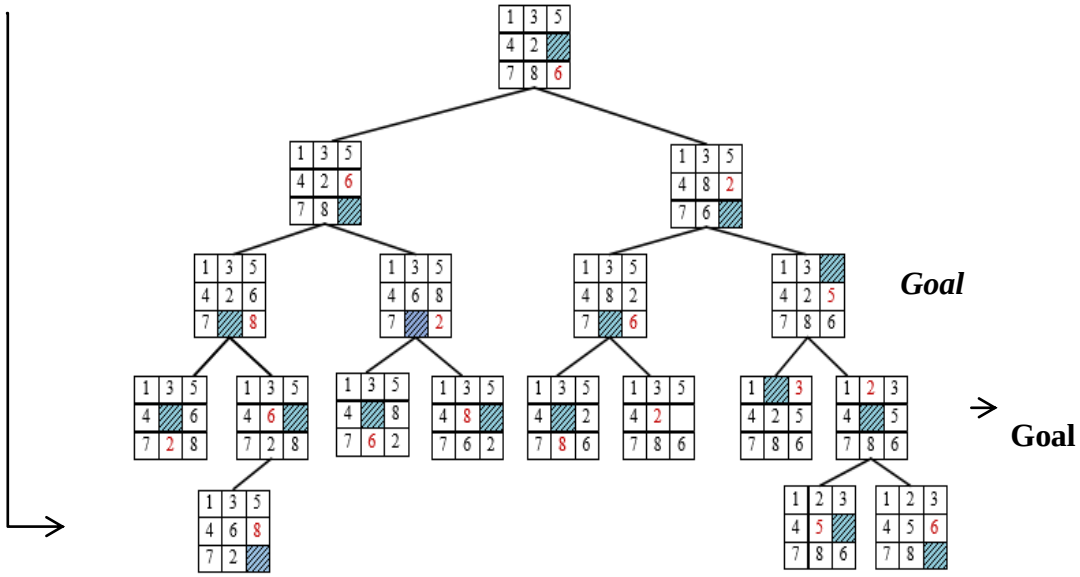
Limit 2



Limit 3



Limit 4



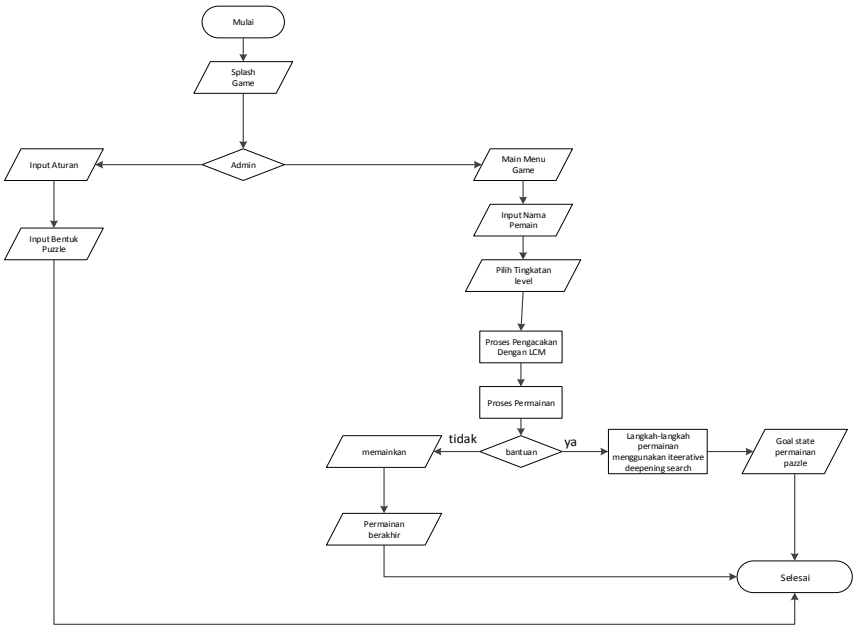
Berdasarkan proses pencarian solusi yang telah dilakukan menggunakan metode secara *iterative* dimana terdapat 3 kemungkinan yang dapat terjadi untuk menuju ke *goal* dan ketiga kemungkinan tersebut sudah mencapai *goal* yang dituju dimana pada kemungkinan ke-1 batasan kedalaman atau limit yang digunakan hingga 2 limit dan perpindahan *puzzle*nya terdapat 5 langkah menuju *goal*, pada kemungkinan ke-2 batasan kedalaman atau limit yang digunakan hingga 3 limit dan perpindahan *puzzle*nya terdapat 9 langkah dan pada kemungkinan ke-3 batasan kedalaman atau limit yang digunakan hingga 4 limit dan perpindahan *puzzle*nya terdapat 17 langkah.

Sehingga dapat disimpulkan bahwa solusi optimal yang diambil adalah kemungkinan ke-1 karena

proses pencapaiannya ke *goal* lebih singkat dengan batasan kedalaman atau limit yang digunakanhingga 2 limit dan perpindahan *puzzle*nya terdapat 5 langkah menuju *goal*.

3.1.3 Flowchart

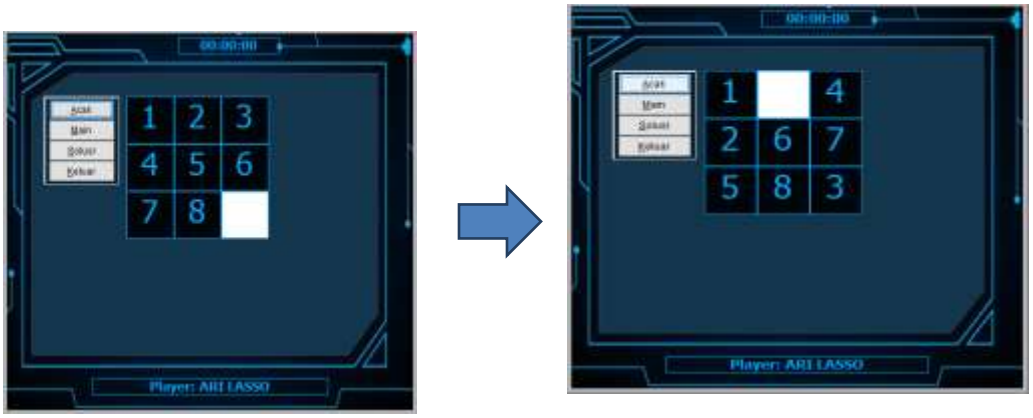
Berikut adalah bagan alir atau flowchart pada *gameslidingpuzzle*.



4. Pembahasan

4.1 *Form* Main

Tampilan *form* main permainan dapat ditunjukkan permainan *puzzle* yang dilakukan oleh pemain bernama “ARI LASSO” dengan level 1 atau bentuk *puzzle* 3x3. Untuk memulai permainan, pemain terlebih dahulu harus mengacak kotak dengan menggunakan tombol acak. *Source code* yang digunakan untuk pengacakan. Proses pengacakan dilakukan menggunakan metode *Linear Congruentmethod*



Setelah proses pengacakan, maka pemain dapat memulai permainan. Permainan dilakukan dengan menggeser kotak tertentu ke kotak kosong yang berdekatan dengan kotak tersebut. Setiap kali dilakukan proses pergeseran, akan dicek apakah susunan kotak yang dihasilkan sudah sesuai dengan *goal* atau *state* akhir. Jika sudah, maka pemain dinyatakan menang. Pada *form* main juga terdapat tombol solusi yang dapat digunakan oleh pemain untuk melihat langkah permainan jika pemain tidak dapat menyelesaikan sebuah permainan.

4.2 *Form* Solusi

Dalam memperoleh solusi, digunakan algoritma *Iteerative Deepning Search* (IDS). Mengingat bahwa terdapat banyak kemungkinan yang dapat muncul serta terdapat keterbatasan memori pada computer yang digunakan, maka dilakukan pembatasan limit atau jumlah iterasi yang dapat dilakukan serta digunakan metode *buffering*, yaitu menampung setiap hasil pemrosesan ke dalam *temporary cursor* sehingga pemrosesan dapat berjalan lebih cepat dan tidak membebani memori.

Langkah terakhir dari proses pencarian solusi adalah menyimpan solusi ke dalam table dan menampilkannya.

5. Penutup

5.1 Kesimpulan

Berdasarkan seluruh rangkaian yang telah dilakukan dalam penelitian ini, maka dapat disimpulkan sebagai berikut:

5.1.1 Algoritma *Itterative depeening search* dapat digunakan pada permainan *Puzzle* namun *time complexity* nya mejadi lebih tinggi sehingga memerlukan waktu yang cukup lama.

5.1.2 Semakin rumit pengacakan yang digunakan semakin lama pula proses pencarian solusi pada algoritma *Iteerative Deepening Search*.

5.1.3 Pada permainan *puzzle* ada kemungkinan tidak ditemukannya solusi karena adanya batasan limit.

5.1.4 Pada pengujian permainan *puzzle* menggunakan spesifikasi laptop yang berbeda menghasilkan waktu permainan *puzzle* yang erbeda yaitu :

5.1.4.1 ukuran *puzzle* 3x3 dan acakan *puzzle* terurut yaitu 1,7,2,5,3,6,4,0,8, menghasilkan 13 langkah dengan waktu penyelesaian pada laptop Acer yaitu 1 menit 20 detik, laptop asus (4 gb) menghasilkan waktu 1 menit 2 detik dan laptop asus (8 gb) menghasilkan waktu 49 detik.

5.1.4.2 untuk ukuran *puzzle* 4x4 dengan acakan *puzzle* terurut yaitu 8,6,0,1,4,14,9,5,2,3,11,12,10,7,15,13, tidak ditemukan solusi sehingga waktu penyelesaian tidak ada.

5.1.4.3 untuk ukuran *puzzle* 5x5 dengan acakan *puzzle* terurut yaitu 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0,16,17,18,19,21,22,23,24,20, menghasilkan 5 langkah dengan waktu penyelesaian pada laptop Acer yaitu 20 detik, laptop asus (4 gb) menghasilkan waktu 50 detik dan laptop asus (8 gb) menghasilkan waktu 13 detik.

5.1.5 Pada ukuran 4x4 dan 5x5 penggunaan algoritma *Iteerative Deepening Search* kurang optimal dikarenakan tingkat kesulitan acakan yang rumit menyebabkan solusi yang ditemukan juga cukup panjang dan apabila memori yang dimiliki tidak mencukupi maka program akan menjadi *overload* dan tidak ditemukannya solusi.

5.2 Saran

Saran yang dapat diberikan dalam penelitian ini adalah :

5.2.1 Sistem yang dihasilkan dalam penelitian ini dapat dikembangkan lebih lanjut sehingga dapat diakses melalui *platform* yang berbeda, misalnya berbasis android.

5.2.2 Pencarian solusi menggunakan algoritma *Iteerative Deepening Search* pada sistem ini apabila dikembangkan cobalah menggunakan teknik *parallel processing* (menggunakan lebih dari satu processor) untuk menambah kecepatan pencarian solusi *puzzle.*

6. Daftar Pustaka

Fatansyah, 2012, Basis Data Edisi Revisi, Penerbit Informatika, Bandung.

Hermawan Latius, Jawa Bendi Kristoforus, 2013, Penerapan Algoritma A* pada Aplikasi Puzzle, Jurnal, Jurusan Teknik Informatika, Sekolah Tinggi Musi.

Kroese D.P, 2011, *Monte Carlo Methods*, Department of Mathematic School of Mathematic and Physcs, The University of Queensland, diakses tanggal 19 maret 2018 dari E-book.

Lomo, Riris D., 2015, *Game Sliding Puzzle*, Diakses tanggal 1 Juni 2018, dari E-book.

Mauluddin Amras, dkk, (2016), Penelitian Implementasi Algoritma Steepest Ascent Hill Climbing Pada Permainan Slide Puzzle Berbasis Android, Jurnal, Program Studi Teknik Informatika, Universitas Langlang Buana Bandung.

Nasrul Anwar, M., 2010, 8-Puzzle Dengan Menggunakan Algoritma Iterative Deepening Search (IDS), Jurnal, Jurusan Teknik Informatika, Sekolah Tinggi Management Informatika ddan Komputer Amikom Yogyakarta.

Raharjo, Budi , 2011, *Belajar Otodidak Membuat Database Menggunakan MySql*, Informatika, Bandung. Diakses tanggal 1 Juni 2018, dari E-book.

Supardi Yuniar, Asriyanik, 2017, *Mudah dan CepatMembuatSkripsidengan Visual Foxpro9*, Jakarta: Elex Media Komputindo. Diakses tanggal 1 Juni 2018, dari E-book.

Suyanto, 2014, *Artificial Intelligence Searching, Reasoning, Planing dan Learning*, Revisi 2, Cetakan 1, Penerbit Informatika, Bandung .

Uriawan Wisnu, dkk, (2015), Penelitian Pembuatan Game Slider Puzzle Menggunakan Metode Steepest Ascent Hill Climbing Berbasis Android, jurnal, Jurusan Teknik Informatika, Universitas Islam Negri Sunan Gunung Djati Bandung.

Wikcasono, Irfandi, 2017, *PenerapanAlgoritma Steepest Ascent Hill Climbing Dan Linear Congruent Method (LCM) dalam Game Slide Puzzle Pengenalan Sembilan Sunan Berbasis Android*, Skripsi, Prodi Teknik Informatika, Universitas Sains dan Teknologi Jayapura.

Yatini, Indra, 2010, *Flowchart, Algoritma, dan Pemrograman Menggunakan Bahasa C++ Builder*, Edisi 1, Cetakan 2, Graha Ilmu, Yogyakarta.