Optimalisasi Penyelesaian Permainan pada Game Puzzle 8 dengan Perbandingan Algoritma A* dan Greedy

Optimization of Game Solutions in Puzzle 8 with Comparison of A and Greedy Algorithms*

Rini Wijayanti*[1], Wahyu Nugraha[2], Kusrini[3]
[1,2,3]Magister Teknik Informatika Universitas AMIKOM Yogyakarta
E-mail: *[1]**riniwijayanti12@gmail.com**, [2]**wahyu.1104@students.amikom.ac.id,**
[3]**kusrini@amikom.ac.id**

Abstrak

Perkembangan game yang pesat serta beragam menjadikan game sebagai salah satu hiburan penghilang stress untuk manusia. Ada beberapa game yang diciptakan untuk mengasah kemampuan penggunanya namun ada pula game yang diciptakan hanya sebatas hiburan ringan. Game yang terlalu sulit untuk diselesaikan, bukan tidak mungkin akan membuat penggunanya malah menjadi stres dan berhenti memaikan permainan. Game yang cukup menguras pikiran salah satunya adalah game Puzzle 8, yang menuntut pemain untuk mampu menyelesaikan permainan dengan menyusun setiap kotak permainan sehingga dapat tersusun secara berurut. Banyak metode yang dapat digunakan untuk membantu menyelesaikan permainan puzzle 8, salah satunya adalah algoritma A-star. A-star digunakan untuk menyelesaikan permainan secara optimal dengan menggunakan fungsi heuristik sebagai salah satu teknik yang mengembangkan efisiensi dalam pencarian. Penelitian ini mengimplementasikan algoritma A-star dalam permainan puzzle 8 dengan menggunakan bahasa pemrograman Java. Dalam pengujian aplikasi algoritma A-star memiliki langkah yang lebih sedikit dibandingkan dengan algoritma Greedy, namun waktu yang dibutuhkan algoritma A-star lebih lama. Hal ini disebabkan A-star melakukan perhitungan biaya sebanyak dua kali, tetapi algoritma Greedy hanya satu kali. Setelah dilakukan pengujian dapat disimpulkan bahwa algoritma A-star memberikan langkah optimal dalam penyelesaian puzzle 8 dilihat dari faktor jumlah langkah untuk menyelesaikan permainan.

Kata Kunci*—Game, A-star, Penyelesaian, Optimal.*

Abstract

The development of a fast and diverse game makes the game as one of the stress relievers for humans. There are some games that are created to hone the ability of users, but there are also games that are created only limited to mild entertainment. Games that are too difficult to solve, it is not impossible that it will make users event stress and stop playing the game. One of the games that is enough to drain the mind is puzzle 8, where players are required to be able to cpmplete the game by arranging each game box so that they can be arranged in order. Many methods can be used to help solve puzzle 8 games. One of them is the A-star algorithm. A-star is used to finish the game optimally by using heuristic function. Which heuristics is one technique that develops efficiency in search. The making of this project is to implement the A-star algorithm in puzzle 8 using the Java programming language. In testing the A-star algorithm application has fewer steps compared to the Greedy algorithm, but the time required for the A-star algorithm is longer. This is because A-star calculates costs twice, but the Greedy algorithm is only one time. After testing it can be concluded that the S-star algorithm provides optimal steps in solving puzzle 8 viewed from the number of steps to complete the game.

Keywords*—Game, A-star, Completion, Optimal.*

1. PENDAHULUAN

Permainan atau *game* merupakan salah satu kegiatan yang biasa dilakukan untuk kesenangan atau juga terkadang dilakukan untuk sarana pendidikan [1]. Permainan merupakan sekumpulan aturan untuk membangun situasi bersaing antara dua atau lebih orang maupun kelompok yang akan menentukan strategi untuk dapat memenangkan permainan [2].

Sebagian permainan puzzle memiliki pola berupa teka – teki yang hanya memiliki satu solusi, apabila *user* tidak dapat menyelesaiakan permainan maka permainan akan berakhir [3]. Permainan dengan tingkat kesulitan tinggi akan membuat para *user* frustasi dan enggan memainkan lagi karena *user* gagal menemukan solusi atau tidak menguasai permainan [4]. Kunci dari sebuah permainan adalah tujuan, peraturan, tantangan dan interaksi yang umumnya menstimulasi fisik dan kejiwaan para pemainnya. Beberapa jenis permainan diciptakan untuk meningkatkan kemampuan sensorik dan motorik, serta sebagai latihan fisik dan mental [3].

Game puzzle merupakan salah satu permainan yang menantang daya kreativitas serta daya ingat lebih dalam karenan munculnya dorongan motivasi untuk menyelesaikan permainan, namun tetap menyenangkan apabila dilakukan berulang – ulang. Tantangan dalam setiap permainan dapat memberikan efek ketagihan bagi para *user* untuk selalu terus mencoba.

Untuk membantu mencari penyelesaian permainan pada *game* puzzle 8 dapat menggunakan algoritma A*. Algoritma ini digunakan untuk membantu menyelesaikan permainan apabila *user* sudah tidak mampu untuk mencari solusi penyelesaian permainan. Algoritma ini ditanamkan dalam aplikasi yang nantinya akan dijalankan secara otomatis oleh sistem apabila diperlukan oleh user. Dengan adanya bantuan ini diharapkan para pemain atau *user* tidak kesulitan lagi dalam menyelesaikan permainan dan tetap akan menyenangkan bila dilakukan kembali.

Penelitian sebelumnya mengenai game puzzle 8 juga pernah dilakukan oleh beberapa peneliti sebelumnya. Perancangan *game* puzzle menggunakan algoritma Decision Tree dilakukan oleh Muhaimin pada tahun 2017 [3]. Penelitian ini menggunakan algoritma Decision Tree untuk mengukur tingkat akurasi *game* puzle yang dibentuk. *Game* pada penelitian ini bertujuan untuk meningkatkan kemampuan konsentrasi dan mengurangi stress *user*.

Membuat *game* puzzle dengan metode Ascent Hill Climbing pernah dilakukan sebelumnya [5]. Penggunaan metode Ascent Hill untuk mencari metode optimasi penyelesaian *game*. Penelitian ini melakukan pengujian dengan melakukan uji fungsi heuristik, yaitu pembangkitan keadaan berikutnya sangat tergantung pada *feedback* dari prosedur pengujian. Peelitian ini memiliki kelemahan yaitu terdapatnya local optimum, yang mana terdapat keadaan *stack* sehingga puzzle tidak dapat bergerak.

Penelitian oleh Bobby tahun 2017 membuat sebuah *game* puzzle bertemakan pemadam kebakaran yang bertujuan untuk mengenalkan berbagai macam kendaraan pemadam kebaran serta kegunaannya. Penggunaan algortima pada penelitian ini yaitu Linier Congruential Generator (LCG) yang digunakan untuk membantu pengacakan gambar pada permainan [6].

Untuk menguji optimal penggunaan algoritma A* akan dilakukan perbandingan antara penggunaan algoritma A* dan Greedy. Dipilih algoritma Greedy karena algoritma ini memiliki langkah yang serupa dengan algoritma A* dan kedua algoritma ini termasuk ke dalam kategori *informed search*. Dengan adanya perbandingan ini, nantinya akan diketahui mana yang lebih optimal dalam menyelesaikan puzzle yang tersusun secara acak.

2. METODE PENELITIAN

2.1. Rancangan Game

3.4.1. Gambaran Umum

Prototype dalam *game* ini adalah permainan *puzzle* yang memiliki ukuran strandart, yaitu 3x3. Permainan ini menuntut seorang *user* untuk dapat menyelesaikan permainan dengan

menyusun kotak dengan urutan yang benar. Dalam penyelesaian kasus ini digunakan algoritma A* sebagai fungsi heuristik untuk bantuan kepada *user* dalam membantu menyelesaikan permainan *puzzle*.

*3.4.2. Algoritma A**

Algoritma A-star merupakan algoritma yang dikemukakan oleh Hart, Nilson, dan Raphael pada tahun 1968. Algoritma A* termasuk dalam algoritma Branch & Bound karena dalam melakukan pencarian solusi menggunakan tambahan informasi (heuristik) dalam menghasilkan solusi [7].

Algoritma A* menetapkan teknik heuristik dalam membantu menyelesaikan masalah. Heuristik memberikan nilai dan harga pada setiap node yang menentukan A* untuk mendapatkan solusi. Heuristik merupajan fungsi optimasi pada algoritma A* yang menjadikan algoritma a* lebih baik dari lainnya. Tetapi heuristik bersifat perkiraan atau estimasi yang tidak memiliki rumus khusus.

Nilai dalam setiap sumpul dapat dihitung dengan menggunakan persamaan 1.

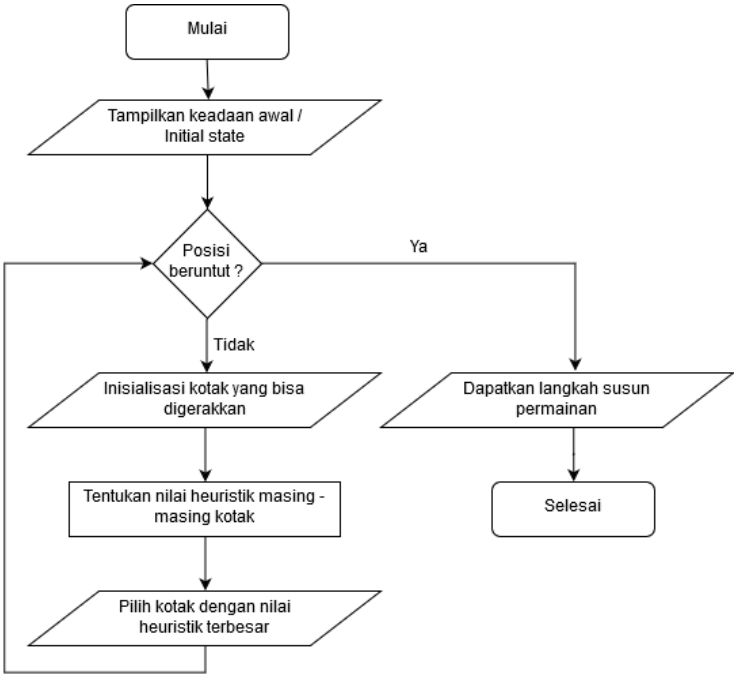
$$f(n) = g(n) + h(n) \quad (1)$$

Posisi koordinat smpul dilambangkan dengan n, untuk nilai fungsi evaluasi dilambangkan denga f(n). Biaya atau cost yang sudah dikeluarkan dari keadaan awal sampai keadaan ke-n digunakan g(n). Sedangkan estimasi biaya untuk sampai pada suatu tujuan yang dimulai dari n menggunakan ambang h(n). Dimana untuk menentukan h(n) menggunakan persamaan 2. Dengan posisi initial state dinamakan sebagai awal dan posisi goal state dinamakan sebagai variable akhir.

$$h(n) = |(awal.x - akhir.x) + (awal.y - akhir.y)| \quad (2)$$

awal = posisi initial state
akhir = posisi goal state

Untuk penerapan algoritma yang digunakan adalah algoritma A* (A-star) pada aplikasi *game* puzzle 8 seperti terlihat pada Gambar 1.



Gambar 1. Flowchart algoritma A*

3.4.3. Algoritma Greedy

Salah satu algoritma yang digunakan untuk memecahkan permasalahan optimasi adalah algoritma *Greedy*, namun algoritma ini memiliki kekurangan yaitu tidak selalu memecahkan solusi dari suatu permasalahan secara optimum. Prinsip kerja dari algoritma *Greedy* ini adalah mengambil sebanyak mungkin apa yang telah didapat saat ini [8].

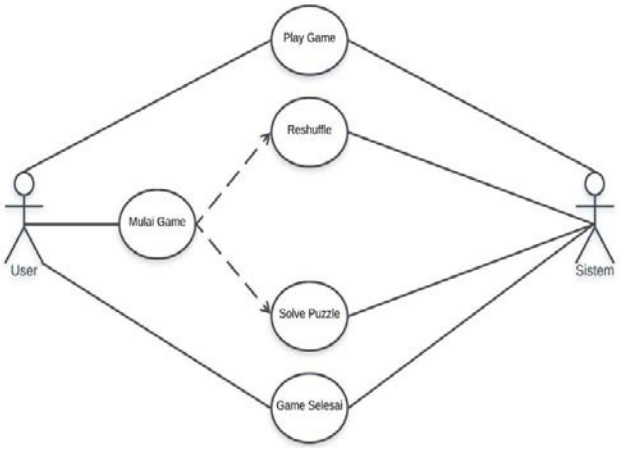
Algoritma *Greedy* memiliki sebuah fungi yang digunakan sebagai acuan dalam menentukan kelayakan sebuah node. Pada algoritma *greedy* tidak bergantung kepada biaya yang dibutuhkan, melainkan dari fungsi heuristik.

Ada perbedaan antara algoritma A* dan *Greedy*, yaitu jika pada algoritma A* pencarian solusi bergantung pada *cost* atau biaya yang dibutuhkan dari sebuah node, sedangkan pada G*greedy* pencarian solusi bergantung pada fungsi evaluasi pada fungsi heuristik yang mengestimasikan arah yang benar untuk menemukan solusi.

2.2. Rancangan Pemodelan Sistem

3.4.1. Use Case Diagram

Use case diagram adalah fungsionalitas yang disediakan oleh sistem sebagai unit – unit yang saling bertukar pesan antar unit atau aktor [9]. Selain itu use case merupakan hubungan interaksi antar aktor yang bertujuan untuk menentukan bagaimana aktor berinteraksi dengan sistem [10]. Gambar 2 merupakan use case pada aplikasi game puzzle 8.



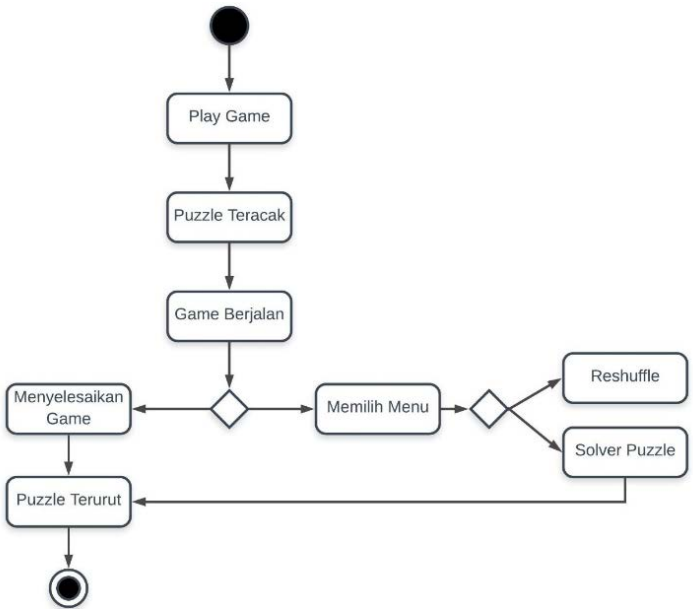
Gambar 2. Use Case Diagram

Dalam *game* puzzle 8 ini terdapat satu aktor yaitu *user*. *User* akan memulai permainan dan terdapat beberapa menu didalam permainan ini, antara lain :

1. Play Game – merupakan menu untuk memulai permainan.
2. Resuffle – untuk melakukan pengacakan ulang kotak – kotak permainan.
3. Solve Puzzle – untuk menyelesaikan permainan secara otomatis oleh aplikasi.

3.4.2. Activity Diagram

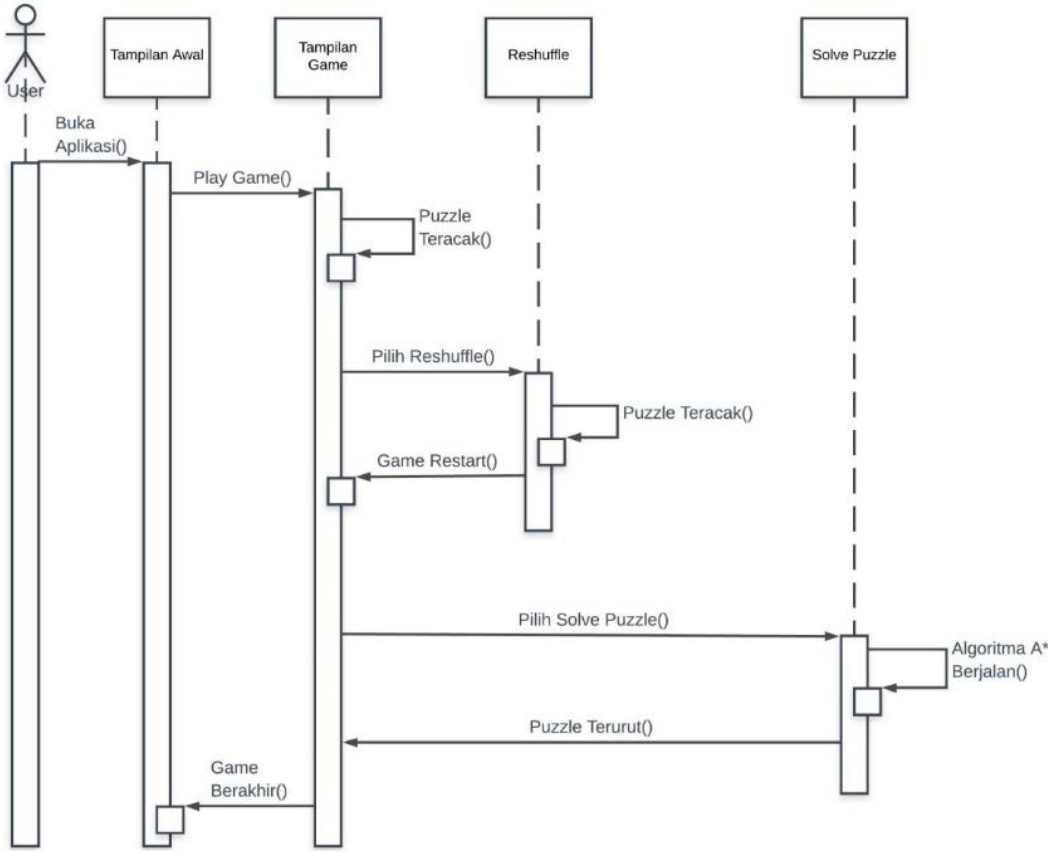
Activity diagram adalah model alur kerja yang menggambarkan sebuah proses bisnis serta urutan aktivitas dalam suatu proses [11]. Gambar 3 merupakan *activity diagram* pada *game* puzzle 8 yang dibangun.



Gambar 3. Activity Diagram

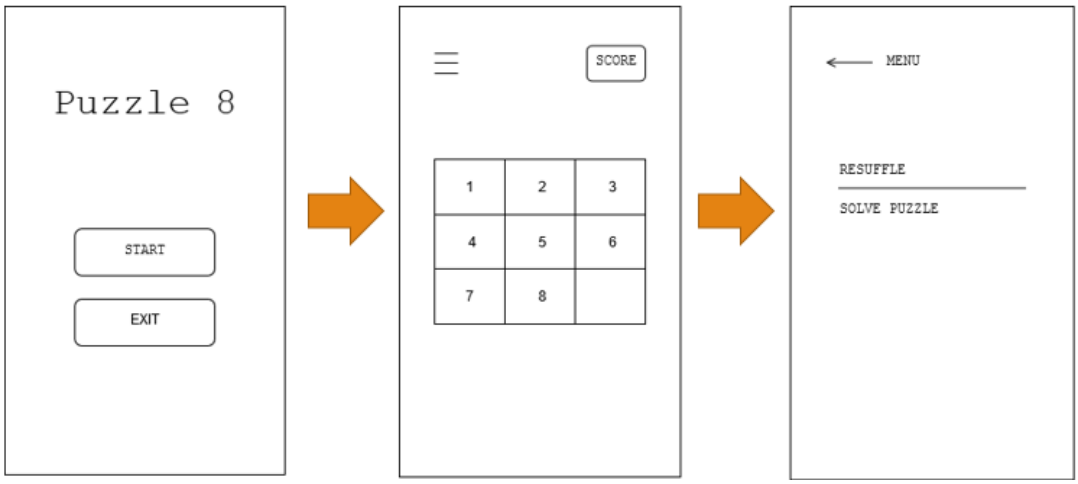
3.4.3. Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan interaksi yang menekankan pada pengaturan waktu dari pesan yang muncul [12]. Untuk mengetahu *sequence diagram* seperti terlihat pada Gambar 4.



Gambar 4. Sequence Diagram

2.2.4 Rancangan User Interface



Gambar 5. Rancangan User Interface

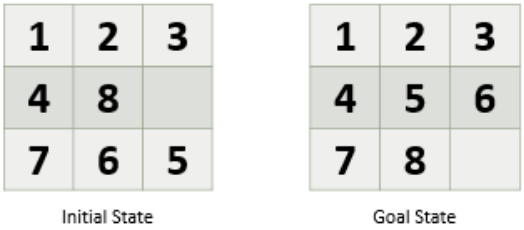
Dalam rancangan user interface terdapat 3 tampilan dalam aplikasi yaitu:

1. Halaman utama – (kiri) merupakan halaman awal aplikasi, terdapat dua pilihan yang dapat dipilih oleh user yaitu mulai permainan atau keluar aplikasi.
2. Halaman permainan – (tengah) merupakan halaman permainan, terdapat kotak permainan tengah layar, penunjuk score yang diperoleh sebelah kanan atas, dan terdapat hidden menu disamping score.
3. Halaman menu – (kiri) merupakan menu yang menampilkan Resuffle (untuk mengacak kembali kotak permainan) dan Solve Puzzle (untuk menyelesaikan permainan secara otomatis).

3. HASIL DAN PEMBAHASAN

*3.1. Penerapan Algoritma A**

Terdapat *initial state* dan *goal state* dalam pernyelesaian masalah algoritma A* pada *game* puzzle 8 ini. *Initial state* merupakan kondisi awal permainan yang mana kotak – kotak akan tersusun secara acak, sedangkan *goal state* merpakan hasil yang harus dicapai untuk menyelesaikan permainan. Gambar 6 merupakan contoh *initial state* dan *goal state.*



Gambar 6. Intial dan Goal State

Untuk mennyelesaikan kasus diatas dilakukan beberapa tahapan. Permainan ini memiliki beberapa peraturan yang harus diperhatikan, antara lain :

1. Kotak yang dapat dipindahkan adalah kota yang bersinggungan langsung dengan kotak *space*.
2. Kotak yang akan dipihdahkan hanya akan dapat bergerak ke kanan, kiri, atas, atau bawah.

Gambar 7 merupakan salah satu contoh penyelesaian dari *initial state* sehingga menjadi *goal state* secara perhitungan manual dengan menggunakan algoritma A*.

Gambar 7. Penyelesaian Algoritma A*

Perhitungan heuristik dimulai pada saat kondisi *initial state*, dan akan dilakukan berulang disetiap kondisi setelah perubahan hingga akhirnya berhenti ketika sudah mencapai *goal state.* Dipilih nilai heuristik yang terbesar. Apabila ada nilai heuristik yang sama, maka dapat dipilih salah satu sesuai dengan urutan.

3.2. Implementasi

Setelah ditentukan cara perhitungan manual, langkah selanjutnya adalah mengubah ke bentuk bahasa pemrograman. Bahasa pemrograman yang digunakan dalam penelitian ini yaitu Java. Pada aplikasi terdapat halaman untuk *user* dapat memilih *menu. Menu* sendiri terdiri dari *resuffle* dan *solve puzzle*, seperti terlihat pada Gambar 8.

Gambar 8. Halaman Menu

Untuk menu *resuffle* akan membuat permainan teracak lagi ketika dipilih oleh *user*. Pilihan menu ini akan memberikan kesempatan kepada *user* untuk pengacakan papan permainan sebanyak semau yang diinginkan oleh *user*.

Pada halaman *menu*, *user* juga dapat memilih bantuan apabila tidak dapat menyelesaikan permainan yaitu dengan memilih *solve.* Pilihan tersebut akan memberikan aplikasi untuk dapat menyelesaikan permainan secara otomatis. Aplikasi akan menyelesaikan permainan sampai *goal state* ditemukan, disinilah peran algoritma A* bekerja. Algoritma A* akan membantu menyelesaikan permainan dan akan memberikan langkah optimal untuk menyelesaikan permainan.

Untuk dapat melakukan proses sesuai dengan fungsi dari masing – masing pilihan *menu*, maka implementasi ke dalam *code* program sebagai berikut.

1. Menu Reshuffle

```
public void randomizeBoard(){
   byte board[];
   while(!isSolvable(board = getRandomBoard()));
   current = board;
}
```

```
private byte[] getRandomBoard(){
   boolean f[]  = new boolean[current.length];
   byte board[] = new byte[current.length];
   Random rand  = new Random();
   for (int i – 0; i < current.length; i++){
      byte t;
      while(f[t = (byte)rand.nextInt(9)]);
      f[t] = true;
      board[i] = t;
   }
   Return board;
}
```

2. Implementasi Algoritma A* untuk Solve Puzzle

```
public static Map<String, byte[]> aStar(byte[] current){
   PriorityQueue<Stage> q = new PriorityQueue<>();
   Map<String, Integer> dist = new HashMap<>();
   Map<String, byte[]> parent = new HashMap<>();
   times = 0;
   dist.put(stringify(current),0);
   q.add(new State(current,0));
   while(!q.isEmpty()){
      State crnt = q.poll();
      times++;
      if(Arrays.equals(crnt.getBoard(),BoardControl.GOAL)) break;
      for(State child : crnt.getNextStates()){
         if(dist.getOrDefult(stringify(child.getBoard()),Integer.MAX_VALUE)>
         child.getCost()){
            parent.put(stringify(child.getBoard()), crnt.getBoard());
            dist.put(stringify(child.getBoard()), crnt.getCost());
            q.add(child);
         }
      }
   }
   Return parent;
}
```

3.3. Pengujian

Setelah proses pembuatan *prototype* selesai dilakukan, langkah selanjutnya adalah melakukan pengujian. Ada beberapa pengujian yang dilakukan dalam penelitian ini.

3.3.1. Pengujian Fungsional Sistem

Pengujian fungsional sistem merupakan pengujian yang dilakukan pada sistem, apakah sistem dapat berjalan dengan skenario yang telah ditentukan sebelumnya atau tidak. Untuk mengetahui pengujian fungsional sistem dapat dilihat pada Tabel 1.

Tabel 1. Pengujian Fungsional Sistem

No	Keterangan	Pengujian	Hasil Uji
1	Tombol Start	Untuk memulai permainan	Berhasil
2	Pindah kotak	Untuk memindahkan kotak agar sesuai urutan puzzle	Berhasil
3	Tombol menu	Untuk menampilkan menu yang ada pada permainan	Berhasil
4	Resuffle	Untuk mengacak kotak permainan	Berhasil
5	Solve puzle	Untuk membantu menyelesaikan puzzle secara otomatis	Berhasil
6	Tombol Exit	Untuk keluar permainan	Berhasil

*3.3.2. Pengujian Optimalisasi Penyelesaian Permainan dengan Algoritma A**

Untuk mengetahui keberhasilan penerapan algoritma dalam membantu menyelesaikan permainan secara optimal, diberikan beberapa *initial state* yang sama kemudian diselesaikan dengan menggunakan algoritma dan dengan tidak menggunakan algoritma, maka hasil perbandingannya dapat dilihat pada Tabel 2. Pengujian ini dilakukan oleh beberapa orang *user*.

Tabel 2. Pengujian Optimalisasi Algoritma

Initial State	Langkah Penyelesaian	
	Dengan Algoritma A*	Tanpa Algoritma
8 7 6 5 4 3 1 1	30	> 30
1 2 3 4 8 7 6 5	5	7
7 6 8 5 1 2 3 4	22	32
5 3 2 1 4 7 6 8	24	28
4 2 7 1 5 6 3 8	26	30

Tabel 2 menunjukkan hasil perbandingan pengujian antara algoritma A* dan algoritma Greedy. Kolom dengan langkah penyelesaian menggunakan algoritma A* memiliki jumlah langkah yang lebih sedikit dibanding dengan menggunakan algoritma *greedy*. Namun, waktu yang diperlukan untuk menyelesaikan relatif lebih lama jika dibandingkan dengan algoritma *greedy* karena algoritma A* melakukan perhitungan dua biaya.

Adapun hasil penggunaan algoritma *greedy* membutuhkan waktu yang relatif lebih cepat karena biaya hanya dihitung sekali saja. Namun, dalam hal jumlah langkah penyelesaian algoritma *Greedy* memiliki jumlah langkah yang lebih banyak dibanding dengan algoritma A* karena algoritma *greedy* bergantung pada fungsi heuristik estimasi arah yang benar untuk menemukan solusi.

Hasil ini menunjukkan bahwa dengan penggunaan algoritma A* dapat memberikan langkah optimal dalam menyelesaikan sebuah *game* dari segi jumlah langkah untuk menyelesaikan permainan meskipun waktu yang dibutuhkan relatif lebih lama.

4. KESIMPULAN

Kesimpulan yang dapat diambil dalam penelitian ini adalah algoritma A* menyelesaikan puzzle dengan waktu lebih lama dibandingkan dengan algoritma Greedy karena algoritma A* melakukan perhitungan 2 biaya. Algoritma A* menyelesaikan puzzle dengan jumlah langkah yang lebih sedikit dibandingkan dengan algoritma Greedy.

5. SARAN

Dalam penelitian yang selanjutnya dapat dilakukan pengujian optimalisasi algoritma A* dalam aspek waktu ataupun yang lainnya. Selain itu dapat melakukan penelitian dengan melakukan perbandingan penggunaan algoritma dengan yang lainnya

DAFTAR PUSTAKA

[1] Halimsah, B. H., Margiso, E., 2014, Problem Solving Permainan Puzzle 8 Menggunakan Algoritma a *, Jurnal Ilmiah SISFOTENIKA, No. 1, Vol. 4, Hal. 64–73

[2] Himmanudin, 2011, Rancang Bangun Game Novel Cisual Menggunakan Ren'py v.6.11.2

[3] Hasanudin, M., Salim, T., Robby, A. N., 2017, Rancang Bangun Aplikasi Game Puzzle Berbasis Android Menggunakan Algoritma Decision Tree, Jurnal CERITA, No. 2, vol. 3, Hal. 171–180

[4] U. of Rochester, 2014, Video gamers aggression linked to frustation, not violent content, www.rochester.edu/newscenter/frustation-in-mastering-video-games-linked-to-aggression/.

[5] Uriawan, W., Faroqi, A., Fathonah, R., 2015, Pembuatan Game Slider Puzzle Menggunakan Metode Steepest Ascent Hill Climbing Berbasis Android, Jurnal ISTEK, No. 1, Vol. 9, Hal. 204–221

[6] Prasetyo, B., Agustina, I., Gufroni, M., 2018, Perancangan Game Puzzle Pemadam Kebakaran Menggunakan Metode Linear Congruential Generator (LCG), JOINTECS Journal, No. 2, Vol. 2.

[7] Kurniawan, R., 2016, Penerapan Algoritma A * (A Star) Sebagai Solusi Pencarian Rute Terpendek Pada Maze, Konferensi Nasional Pengembangan Teknologi Informasi dan Komunkasi.

[8] Rachmawati, D., Candra, A., 2013, Implementasi Algoritma Greedy Untuk Menyelesaikan Masalah Knapsack Problem, Jurnal Saintikom, No. 3, Vol. 12, Hal. 185-192

[9] Yuni, S., 2015, Analisis dan Perencanaan UML (Unified Modelling Languange), Graha Ilmu, Yogyakarta.

[10] Windarto, A. P., 2018, Penilaian Prestasi Kerja Karyawan PTPN III Pematangsiantar Dengan Metode Simple Additive Weighting (SAW), Jurasik (Jurnal Riset Sistem Informasi dan Tenologi Informasi, No. 1, Vol. 2, Hal. 84.

[11] Rosa, A., Salahuddin, M., 2011, Modul Pembelajaran Rekayasa Perangkat Lunak (TErstruktur dan Berorientasi Objek), Modula, Bandung.

[12] Sasmito, G. W., 2017, Penerapan Metode Waterfall Pada Desain Sistem Informasi Geografis Industri Kabupaten Tegal, Jurnal Informatika (Jurnal Pengembangan IT), No. 1, Vol. 2, Hal. 6–12.