SAINTEKS
Volume 16 No 1, April 2019
ISSN: 0852-1468 (55 – 69)

Rancang Bangun Aplikasi Penyelesaian *Puzzle* 8 Angka Menggunakan Metode *Hill Climbing*

Application Development of the 8 Figures Puzzle Completion Using Hill Climbing

Hindayati Mustafidah[1], Bangkit Nurdiyansah[2]

[1,2]*Teknik Informatika – Unievrsitas Muhammadiyah Purwokerto*

[1]h.mustafidah@ump.ac.id

ABSTRAK

Puzzle 8 adalah sebuah permainan dimana terdapat sembilan kotak angka yang diacak, yang harus disusun kembali ke posisi yang benar dan terurut. *Hill Climbing* merupakan salah satu metode yang dapat digunakan untuk menyelesaikan permainan *puzzle* 8, dimana dalam metode *Hill Climbing* terdapat sebuah fungsi yang dinamakan fungsi *heuristic*. *Heuristic* adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian. Fungsi *heuristic* yang digunakan adalah dengan melihat banyaknya kotak pada posisi yang benar dan total keseluruhan jarak dari kotak yang berada ditempat yang benar untuk mencapai posisi yg benar atau disebut dengan *manhattan distance*. Metode *Hill Climbing* berfungsi sebagai pemberi solusi pergerakan dalam permainan *puzzle* 8, yang diharapkan solusi yang diberikan mampu mempercepat proses penyelesaian permainan *puzzle* 8.

Kata-kata Kunci : *puzzle,* metode *Hill Climbing, heuristic, manhattan distance*

ABSTRACT

Puzzle 8 is a game where there are nine randomized box number, which must be arranged correct position and sorted. Hill Climbing is one method that can be used to complete the puzzle 8 game , wherein the method of Hill Climbing there is a function called heuristic function . Heuristics is a technique that develops efficiency in the search process. Heuristic function that is used is to look at the number of boxes in the correct position and the total distance of the box is the correct place to achieve the correct position or called manhattan distance . Hill Climbing method serves as a giving solution puzzle game 8 , which is expected given the solution that is able to accelerate the process of resolving this puzzle game 8.

Keywords: *puzzle , Hill Climbing method , heuristic , manhattan distance*

PENDAHULUAN

Puzzle 8 adalah sebuah permainan dimana terdapat sembilan kotak angka, huruf maupun gambar yang diacak, yang harus disusun kembali ke posisi yang benar dan terurut. Banyak metode yang dapat digunakan untuk menyelesaikan *puzzle* 8, salah satunya menggunakan metode *Best First Search*. Di dalam *Best First Search* terdapat sebuah fungsi yang dinamakan fungsi *heuristic*.*Heuristic* adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian. Salah satu algoritma yang memakai fungsi *heuristic* adalah algoritma *Hill Climbin*g. Fungsi *heuristic* yang digunakan adalah dengan melihat banyaknya kotak pada posisi yang benar dan total keseluruhan jarak dari

kotak yang berada di tempat yang benar untuk mencapai posisi yang benar. Fungsi ini sering juga disebut dengan *manhattan distance*.

Metode *Hill Climbing* merupakan salah satu variasi metode *(Generate and Test*) dimana umpan balik yang berasal dari prosedur uji digunakan untuk memutuskan arah gerak dalam ruang pencarian *(search)*. Dalam prosedur *Hill Climbing*, fungsi uji dikombinasikan dengan fungsi *heuristik* yang menyediakan pengukuran kedekatan suatu keadaan yang di berikan dengan tujuan *(goal)*.

Hill Climbing pada permainan *puzzle* 8 angka sangat berguna, satu kali proses evaluasi menggunakan *Hill Climbing* hanya akan melibatkan maksimal 4 *state* untuk kondisi *initial state*, dan maksimal 3 *state* untuk kondisi *state* selain *initial state*. Sehingga *state space* untuk algoritma ini dapat dikatakan relatif sangat kecil. Maka dari itu digunakan metode ini untuk menyelesaikan permainan *puzzle* 8 angka tersebut.

Adapun ruang lingkup pada penelitian ini adalah ukuran dari *puzzle* yang digunakan statis dan digunakan ukuran standar yang biasa digunakan yakni 3x3 sel, dengan keadaan akhir *(goal state)* yang sudah ditentukan. Sedangkan untuk algoritma yang digunakan dalam penelitian ini adalah *Steepest-Ascent Hill Climbing*.

Tujuan penelitian ini adalah membangun aplikasi yang dapat digunakan untuk mengoptimalkan waktu dan langkah pergerakan dalam menyelesaikan permainan *puzzle* 8 angka menggunakan metode *Hill Climbing*.

Manfaat dari penelitian ini memberikan wawasan baru bagi semua orang bahwa metode *Hill Climbing* dapat digunakan untuk menyelesaikan permainan *puzzle* 8 angka.

Beberapa acuan yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut.

1.*Puzzle 8* Angka

Puzzle 8 angka adalah representasi permainan teka-teki yang dapat diselesaikan dengan mengurutkan atau menyusun komponen-komponen pembentuknya sesuai dengan kondisi yang diinginkan. Komponen pada *puzzle* adalah berupa kotak-kotak bernomor atau bergambar (sesuai kebutuhan) yang dapat diacak sedemikian hingga menjadi suatu pola *random* ditunjuk dengan Gambar1, yang dapat dicari jalan penyelesaiannya ditunjuk dengan Gambar2.

1	**2**	**3**
	8	**4**
7	**6**	**5**

Gambar1. Pola random *(initial state)*

1	**2**	**3**
8		**4**
7	**6**	**5**

Gambar2. Tujuan *(goal state)*

Sesuai namanya, *puzzle*8 terdiri atas 8 kotak dan 1 tempat kosong yang dapat digerakkan dengan aturan tertentu. Aturan pergerakannya hanya berupa empat (4) arah pergerakan, yaitu: atas, bawah, kanan, dan kiri, sertaterlimitasi oleh ukuran dimensi papan yang ditempatinya. Pada *puzzle* 8, batasannya adalah ukuran 3×3. Sehingga, 8 kotak yang dimiliki hanya dapat bergerak dalam lingkup ukuran tersebut (Jokodo, 2011).

2. Metode*Hill Climbing*

Metode *Hill Climbing* adalah metode yang dikenal untuk pencarian lokal. Gagasan untuk metode *Hill Climbing* ini adalah mulai secara acak dari *state* yang sudah ada bergerak ketetangga dengan nilai evaluasi yang terbaik dan jika suatu minimum lokal telah dicapai lalu memulai lagi secara acak pada *state* yang berbeda. Pengulangan prosedur ini dilakukan hingga solusi ditemukan. Metode *Hill Climbing* sering digunakan jika terdapat fungsi *heuristic* yang baik untuk mengevaluasi *state* (Rich dalam Kusumadewi, 2003). Metode ini mirip dengan metode *Depth-First Search*. Sebagai contoh, seandainya seseorang berada di sebuah kota yang tidak dikenal, tanpa peta dan ingin menuju kepusat kota, cara sederhana yang harus dilakukan adalah menuju gedung yang tinggi. Fungsi *heuristic*-nya adalah jarak antara lokasi sekarang dengan gedung yang tinggi dan *state* yang diinginkan adalah *state* yang mana jarak tersebut merupakan yang terpendek (Kusumadewi, 2003).

Pada *puzzle* 8 angka, satu kali proses evaluasi menggunakan metode *Hill Climbing* hanya akan melibatkan maksimal 4 *state* untuk kondisi *initial state*, dan maksimal 3 *state* untuk kondisi *state* selain *intial state*. Sehingga *state space* pada algoritma ini dapat dikatakan relatif sangat kecil. Terdapat dua jenis *Hill Climbing* yang sedikit berbeda, yakni *Simple Hill Climbing (Hill Climbing* sederhana) dan *Steepest-Ascent Hill Climbing (Hill Climbing* dengan memilih kemiringan yang paling tajam/curam).*Simple Hill Climbing*, secara sederhana langsung memilih *new state* yang memiliki jalur yang lebih baik (curam) dari pada jalur-jalur sebelumnya tanpa memperhitungkan jalur-jalur lain yang lebih curam. Sedangkan *Steepest-Ascent Hill Climbing*, sesuai dengan namanya, akan mengevaluasi semua *state* yang berada dibawah *currentstate* dan memilih *state* dengan jalur paling curam (Kusumadewi, 2003). Algoritma *Hill Climbing* ada 2 yaitu **Algoritma *Simple Hill Climbing*** dan ***Steepest-Ascent Hill Climbing*** (Kusumadewi dan Purnomo, 2005). Pencarian *Simple Hill Climbing* dimulai dari arah kiri. Apabila nilai *heuristic* dari arah kiri lebih baik maka dibuka untuk pencarian selanjutnya. Jika tidak maka akan dilihat tetangga dari arah kiri tersebut, dan berlaku seterusnya. Hal tersebut menjadikan pengetahuan yang dapat digunakan oleh metode ini dan metode *heuristic* lainnya untuk memecahkan permasalahan yang mungkin sangat sukar. Sementara *Steepest–Ascent Hill Climbing* hampir sama dengan *Simple Hill Climbing*, hanya saja gerakan pencarian tidak dimulai dari kiri, tetapi berdasarkan nilai *heuristic* terbaik.

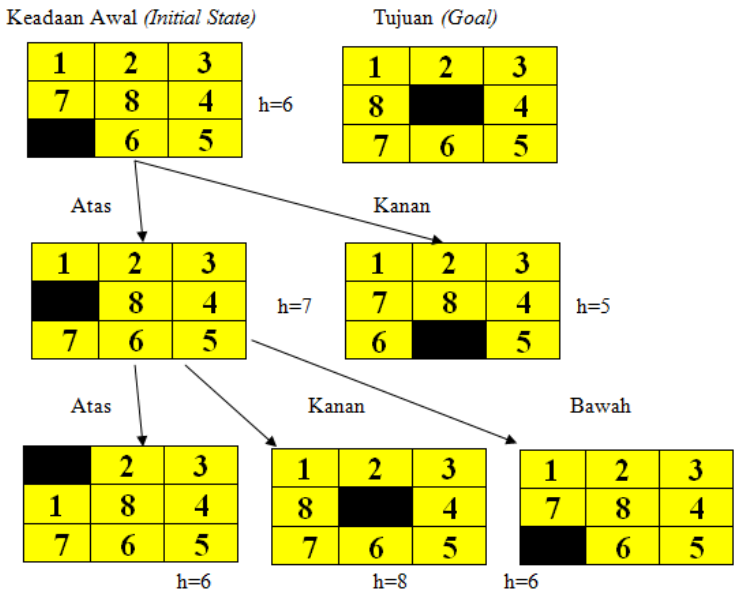
3. Fungsi *Heuristic*

Fungsi *heuristic* dapat didefinisikan sebagai suatu fungsi yang menghitung biaya perkiraan *(estimasi)* dari suatu simpul menuju kesimpul yang dianggap sebagai tujuan. Pada permainan *puzzle* 8 angka fungsi *heuristic* ditentukan berdasar kedekatan antara kondisi awal *(initial state)* dengan kondisi tujuan *(goal state)* (Kusumadewi, 2003).

Fungsi *heuristic* yang digunakan dalam permainan *puzzle* 8 angka ini adalah; Jumlah kotak angka yang menempati posisi yang benar dan mendekati posisi tujuan *(goal state)* yang diharapkan menjadi yang terbaik.

4. Representasi Ruang Keadaan

Ruang keadaan *(state space)* merupakan suatu ruang yang berisi semua keadaan yang mungkin dan untuk menyelesaikan permainan *puzzle* 8 angka. Dengan memulai permainan yang dimulai dari keadaan awal, kemudian menggerakan kotak kosong sesuai dengan operator-operator yang ada, dan mengakhiri permainan apabila sudah mencapai tujuan. Gambar 10 merupakan sedikit representasi ruang keadaan dari permainan *puzzle* 8 angka yang dimulai dari keadaan awal *(initial state)* menuju tujuan *(goal)* (Gambar 3).



Gambar 3. Representasi Ruang Keadaan

Gambar 3 merupakan representasi dari ruang keadaan permainan *puzzle* 8 angka yang terdiri dari keadaan awal menuju tujuan. h *(heuristic)* merupakan jumlah kotak angka yang menempati posisi yang benar dan jumlah yang lebih tinggi adalah yang lebih diharapkan sebagai keadaan yang lebih baik dari keadaan sebelumnya. h=8 merupakan tujuan *(goal state)* dari langkah permainan *puzzle* 8 angka, h=8 pada representasi diatas di hasilkan dari pergerakan (operator) kotak kosong ke atas (angka yang berada di atas kotak kosong bergeser ke bawah) untuk langkah pertama, kemudian dilanjutkan kotak kosong bergerak ke kanan (angka yang berada di sebelah kanan kotak kosong bergeser ke kiri) pada langkah kedua, sehingga ditemukan h=8 (tujuan/*goal*).

5. XAMPP dan Apache

XAMPP adalah sebuah paket perangkat lunak yang di dalamnya terdiri dari *Apache*, *MySQL*, dan PHP (Utomo, 2008).*Apache* merupakan salah satu perangkat lunak yang dipergunakan secara luas pada sistem operasi *Linux*. Pengembangannya yang dimulai dari tahun 1995 oleh sekelompok kecil pemrogram yaitu *Apache Software Foundation Incorporated*, pada tahun 1999 mulai berkonsentrasi untuk mendukung projek *Aphace* HTTP *server*. Dengan berbasis jumlah pengguna lebih dari 25 juta *server* di seluruh dunia, membuat *Apache* HTTP *server* mempunyai keunggulan dari sisi fleksibilitas dan perfomansi.

Apache sendiri sebenarnya merupakan suatu *web server* yang dapat dikatakan sederhana dalam implementasinya, dan ini sesuai dengan tujuan awalnya sebagai penyedia layanan untuk halaman internet. Beberapa *web server* komersial menyediakan berbagai macam fasilitas dalam lingkup *web server*, tetapi apabila ditelaah lebih lanjut malah akan menimbulkan celah keamanan yang cukup serius. Kesederhanaan dan desain yang bersifat modular dari *server* HTTPD *Apache* membawa sejumlah aspek sekuritas yang lebih baik (Emanuel, dkk, 2008).

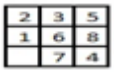
6. PHP (*Programming Hypertext Preprocessor*)

Bahasa pemrograman PHP merupakan bahasa pemrograman untuk membuat web yang bersifat *server-side scripting*. PHP memungkinkan seseorang untuk membuat halaman web yang bersifat dinamis. PHP dapat dijalankan pada berbagai macam OS

(*Operating System)*, misalnya Windows, Linux, dan Mac OS.Selain Apache, PHP juga mendukung beberapa *webserver* lain, misalnya MicrosoftIIS, Caudium, PWS dan lain-lain. PHP dapat memanfaatkan *database* untuk menghasilkan halaman web yang dinamis. Sistem manajemen *database* yang sering digunakan bersama PHP adalah MySQL, namun PHP juga mendukung sistem manajemen *databaseOracle*, *Microsoft Acces*, *Interbase*, *d-Base*, *Postgre* SQL dan sebagainya. Sama dengan *web server* lainnya PHP juga bersifat *open source* sehingga setiap orang dapat menggunakannya dengan gratis (Kadir, 2001).

Beberapa penelitian terkait dengan bahasan tulisan ini telah dilakukan oleh Yuliana, dkk. (2012) dan Zakiah (2012). Yuliana, dkk (2012) telah melakukan penelitian mengenai Implementasi Algoritma *A Star* pada Pemecahan *Puzzle* 8. Tujuan dari penelitian tersebut adalah mengimplementasikan algoritma *AStar* pada pemecahan *puzzle* 8 agar dapat membuktikan bahwa Algoritma *A Star* dapat digunakan dalam penyelesaian *puzzle* 8 dan dapat melihat perbandingan antara Algoritma *A Star* dengan Algoritma *Greedy*. Pengujian yang akan dilakukan terhadap aplikasi adalah dengan membandingkan kedua buah algoritma secara waktu, sistem dan penghitungan keakuratan Algoritma *A Star* yang ditemukan oleh sistem dengan perhitungan manual. Untuk tujuan *goal* posisi angka *puzzle* yang benar dibuat berurut 1, 2, 3, 4, 5, 6, 7, 8, 0. Untuk mempermudah melihat perbandingan tersebut, dapat dilihat pada Tabel1.

Tabel 1. Perbandingan *A Star* dan *Greedy*

Contoh Kasus	Jumlah Langkah *A Star*	Jumlah Langkah *Greedy*
	Berhasildikerjakandengan 6 langkah	Tidak menemukan solusi hingga 250 langkah
	Berhasil diselesaikan dengan 9 langkah	Tidak menemukan solusi hingga 293 langkah
	Berhasil diselesaiakan dengan 14 langkah	Tidak menemukan solusi hingga 144 langkah

Tabel 1 merupakan perbandingan antara Algoritma *A Star* dan Algoritma *Greedy* di dalam menyelesaikan *puzzle* 8. Dari tabel di atas dapat dilihat bahwa, Algoritma *A Star* mampu menyelesaikan contoh kasus *puzzle* 8 dengan baik dan pada Algoritma *Greedy* terdapat beberapa kasus yang tidak dapat terselesaikan. Semakin optimal fungsi *heuristic* yang digunakan maka akan semakin optimal pula solusi yang didapatkan. Dapat dilihat bahwa fungsi *heuristic* pada Algoritma *Greedy* kurang optimal untuk memecahkan *puzzle* 8 dibandingkan dengan fungsi *heuristic* yang digunakan oleh algoritma *A Star*. Fungsi *heuristic* Algoritma *Greedy* menggunakan perhitungan biaya estimasi sedangkan fungsi *heuristic* algoritma *A Star* menggunakan perhitungan biaya estimasi dan fungsi *city block distance* atau sering disebut *manhattan distance*, dimana fungsi ini menghitung biaya sebenarnya untuk mengembalikan kotak pada *puzzle* ke posisi sebenarnya.

Zakiah (2012) dalam penelitiannya mengatakan 8 puzzle merupakan salah satu implementasi dari *Artificial Intelegence*. Dalam proses penyelesaiannya banyak terdapat algoritma-algoritma pencarian yang dapat diterapkan. Dalam hal ini dipilih algoritma *Steepest-Ascent Hill Climbing* (*Hill Climbing* dengan memilih kemiringan yang paling tajam/curam) yang divariasikan dengan fungsi *heuristic* jarak dan posisi serta *loglist* sebagai tempat penyimpanan sebagai pembanding terhadap kondisi-kondisi yang pernah dilalui untuk menghindari perulangan *(looping)* atau proses pergeseran ke kondisi yang telah ada sebelumnya. Tujuan dari penelitian ini adalah menentukan tujuan *(goal state)* apa yang harus digunakan untuk pembanding untuk dilakukan pergeseran *state*-nya, yaitu:

a. *Goal state* A adalah *goal state* yang dipilih apabila jumlah dari perbandingan semua ubin *(tile)* dengan ubin setelahnya (sebelah kanannya) berjumlah ganjil maka *goal state*-nya melingkar adalah [1,2,3,8,0,4,7,6,5]
b. *Goal state* B adalah *goal state* yang dipilih apabila jumlah dari perbandingan semua ubin *(tile)* dengan ubin setelahnya (sebelah kanannya) berjumlah genap maka *goal state*-nya berurut adalah [0,1,2,3,4,5,6,7,8]

METODE

A. Analisis Sistem

Pada tahap ini akan dilakukan analisis terhadap kebutuhan spesifikasi aplikasi (*Software Requirement Specification*) yang meliputi komponen masukan untuk penyelesaian dalam algoritma, adalah berupa atribut *initial state* dan *goal state.* Analisis sistem menjelaskan gambaran umum dari sistem yang akan di bangun meliputi; kebutuhan *hardware*, kebutuhan *software*, kebutuhan masukan, kebutuhan proses, kebutuhan keluaran, kebutuhan antarmuka dari sistem untuk menyelesaikan permasalahan *puzzle* 8 angka.

1. *Kebutuhan hardware*

Kebutuhan *hardware* yang diperlukan adalah satu unit komputer dengan sistem operasi *Windows* sebagai alat untuk menjalankan sistem.Dengan minimal spesifikasi sebagai berikut :

- Processor Intel Pentium 2 Ghz
- RAM 1 Gb
- Space Minimum Hardisk 40 Gb
- Monitor LCD

2. *Kebutuhan software*

Kebutuhan *software* yang diperlukan antara lain;

- Sistem operasi Windows 7(seven) 32 bit
- XAMPP
- Java
- Google Chrome/Mozilla firefox/internet Explorer

3. *Kebutuhan masukan*

Input atau masukan dari aplikasi penyelesaian *puzzle* 8 angka ini adalah memasukan angka secara acak pada bagian *puzzle initial state.*

4. *Kebutuhan proses*

Berdasarkan analisis yang telah dilakukan maka dapat diketahui kebutuhan yang akan digunakan dalam pengolahan data dari *input* data yang diberikan kepada sistem sehingga menghasilkan *output* yang sesuai dengan yang diharapkan. Proses yang dilakukan dalam aplikasi ini hanya akan mengambil solusi terbaik dengan *heuristic* yang memiliki poin tertinggi untuk menuju *goal state*.

5. *Kebutuhan keluaran*

Keluaran yang diperoleh dari aplikasi penyelesaian *puzzle* 8 angka ini adalah hasil dari pergerakan *puzzle* 8 angka dengan kondisi *goal state* atau dengan kondisi yang semestinya.

6. *Kebutuhan antarmuka*

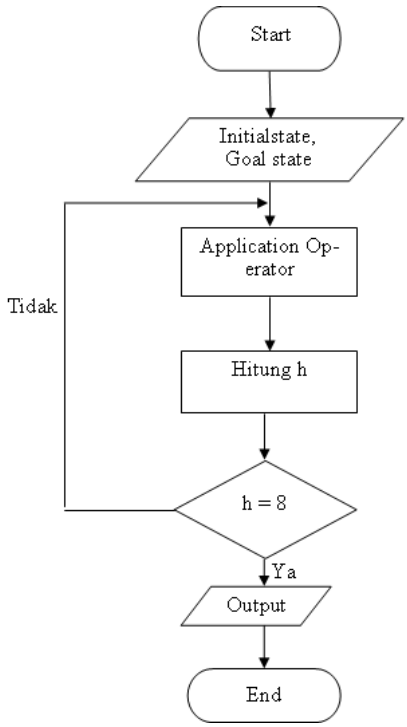
Kebutuhan antarmuka dalam sistem ini berguna untuk menyelesaikan permasalahan *puzzle* 8 angka yang meliputi:

- Kebutuhan antarmuka keadaan awal *(initial state) puzzle* 8 angka.
- Kebutuhan antarmuka tujuan *(goal state) puzzle* 8 angka.
- Kebutuhan antarmuka langkah-langkah.
- Kebutuhan antarmuka proses telah selesai.

B. Perancangan Sistem

1. *Flowchart*

Pada tahap ini akan dilakukan pemodelan desain aplikasi. Alur aplikasi penyelesaian *puzzle* 8 angka dimulai dengan inisialisasi data awal atau *initial state* dengan menginputkan secara manual angka-angka yang di definisikan dimulai dari angka 0 (nol) hingga angka 8 secara acak serta *goal state* sebagai tujuan *puzzle* yang akan diselesaikan. Kemudian proses berlanjut dengan menghitung nilai h untuk mencari nilai terbesar, jika belum ditemukan nilai terbesar maka proses pencarian dilakukan secara berulang hingga ditemukan nilai h yang paling besar. Penjelasan proses tersebut dapat dilihat pada Gambar 4.

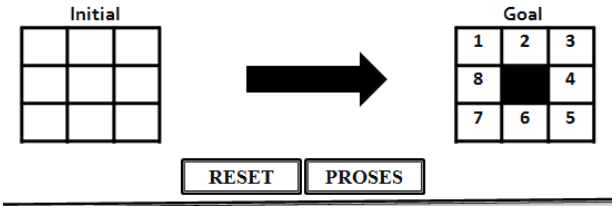


Gambar 4. *Flowchart* Proses Penyelesaian *Puzzle* 8 Angka

2. *Desain interface (antarmuka)*

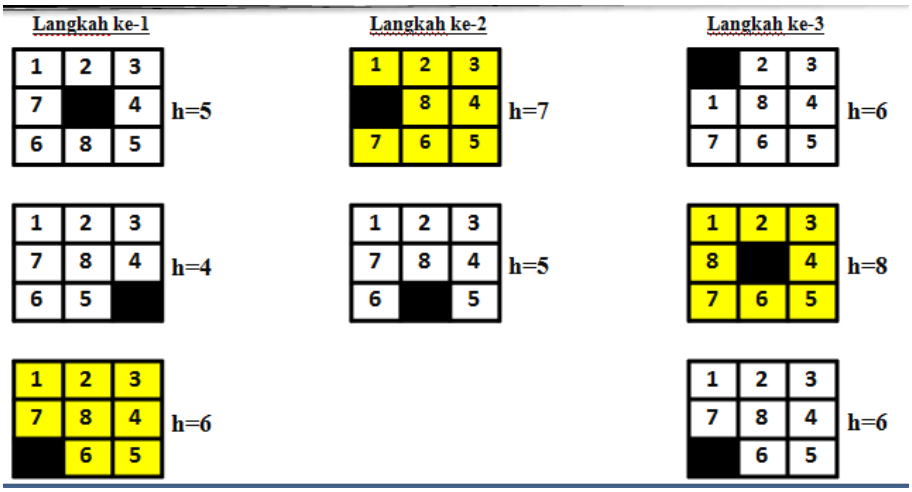
Perancangan antarmuka dimaksudkan untuk menggambarkan desain dari sistem, ilustrasi dan rancangan antarmuka terhadap aplikasi yang akan ditampilkan. Perancangan aplikasi penyelesaian *puzzle* 8 angka ini dibuat dengan memfokuskan pada *easy using*,yang bertujuan untuk memudahkan pengguna dalam memakai aplikasi.

Halaman awal digunakan sebagai halaman inisialisasi aplikasi yang digunakan oleh pemakai untuk menginputkan data inisialisasi atau data awal dan halaman *goal state* digunakan untuk mengetahui posisi *puzzle* yang akan dituju, dapat dilihat pada Gambar 5.



Gambar 5. Halaman *initial state* dan *Goal State*

Selanjutnya halaman proses digunakan sebagai alur proses penyelesaian *puzzle* 8 angka, sehingga langkah-langkah dari proses penyelesaiannya dapat terlihat perpindahannya. Desain antar muka ini dapat dilihat pada Gambar 6.



Gambar 6. Halaman Pemberitahuan Proses Selesai

HASIL DAN PEMBAHASAN

A. Posisi *Heuristic*

Terdapat dua jenis fungsi *heuristic* yang dapat digunakan yaitu:

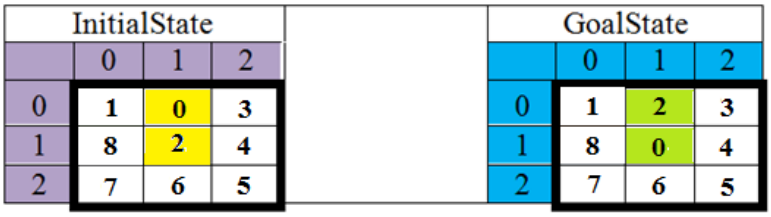
- h1 = jumlah kotak yang menempati posisi yang benar.
- h2 = jarak antara *initial state* dengan *goal state*.

Cara menghitungnya adalah dengan menjumlahkan harga mutlak selisih antara kolom dan baris pada *initial state* dengan *goal state*. Total nilai dari h1 dan h2 adalah maksimal 8 yaitu h1 maksimum apabila posisi kotak sudah sesuai dengan *goal state* adalah 8 dan h2 adalah 0 (nol).

Posisi *Heuristic* adalah nilai dari pencocokan antara *initial state* dengan *goal state*, apabila:

- Jumlah kotak yang menempati posisi yang benar;
  Jumlah *heuristic* yang lebih tinggi adalah yang diharapkan (lebih baik).
- Jumlah kotak yang menempati posisi yang salah;
  Jumlah *heuristic* yang lebih kecil yang diharapkan (lebih baik).

Dalam penelitian ini, fungsi *heuristic* yang digunakan adalah jumlah kotak yang menempati posisi yang benar dimana setiap pergerakan kotak mendapati nilai *heuristic* yang lebih tinggi (lebih baik) dari keadaan sebelumnya.Untuk contoh dapat dilihat pada Gambar 7.



Gambar 7. *Heuristic* Posisi

- Jumlah posisi kotak yang menempati posisi yang benar adalah [1, 3, 4, 5, 6, 7, 8] yaitu h1 = 7.
- Jumlah posisi kotak yang menempati posisi yang salah adalah [0, 2] yaitu h2 = 2.

B. Pemilihan Pergeseran *State* yang Diperbolehkan

State dapat bergerak ke *state* yang selanjutnya berdasarkan posisi kotak kosong. Operator yang diperbolehkan adalah:

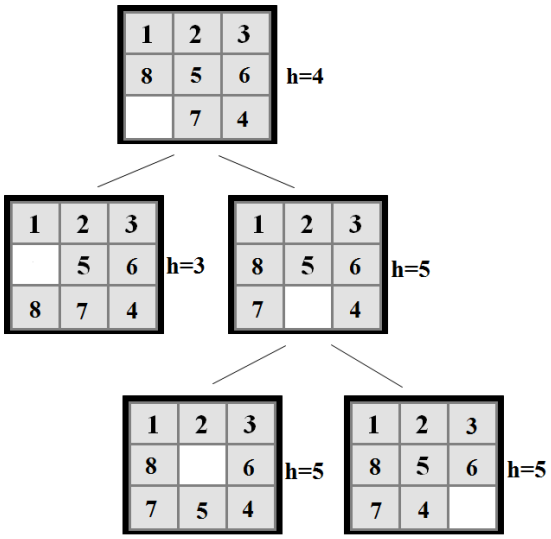
1. Kotak kosong bergeser ke kanan
2. Kotak kosong bergeser ke kiri
3. Kotak kosong bergeser ke atas
4. Kotak kosong bergeser ke bawah

Kemungkinan posisi kotak kosong pada *state* adalah:

1. Posisi kotak kosong pada *index* ke 1 atau pojok kiri atas, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan atau ke bawah.
2. Posisi kotak kosong pada *index* ke 2 atau tengah atas, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan, ke kiri atau ke bawah.
3. Posisi kotak kosong pada *index* ke 3 atau pojok kanan atas, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kiri atau ke bawah.
4. Posisi kotak kosong pada *index* ke 4 atau tengah kiri, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan, ke atas atau ke bawah.
5. Posisi kotak kosong pada *index* ke 5 atau tengah, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan, ke kiri ke atas atau ke bawah.
6. Posisi kotak kosong pada *index* ke 1 atau tengah kanan, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kiri, ke atas atau ke bawah.
7. Posisi kotak kosong pada *index* ke 7 atau pojok kiri bawah, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan atau ke atas.
8. Posisi kotak kosong pada *index* ke 8 atau tengah bawah, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kanan, ke kiri atau ke atas.
9. Posisi kotak kosong pada *index* ke 9 atau pojok kanan bawah, maka pergeseran yang diperbolehkan adalah kotak kosong bergeser ke kiri atau ke atas.

C. *Plateau*

Plateau, kondisi ketika ada dua (2) atau lebih *evaluation state* yang mempunyai nilai *heuristic* sama besar dan juga merupakan nilai terbaik. Dalam kondisi *plateau*, hal itu dapat memicu *local maxima* (Kusumadewi, 2003). *Local maxima*, yaitu solusi lokal yang ditemukan dengan fungsi evaluasi. Tetapi solusi ini bukan kondisi *goal* yang diharapkan, dan jika dilakukan evaluasi secara terus menerus, maka akan kembali lagi ke kondisi solusi lokal itu sendiri. Karena memang fungsi evaluasi yang dilakukan menemui batasan pencarian lokal (Kusumadewi, 2003). Representasi kondisi *plateau* dalam permainan *puzzle* yang menerapkan metode pencarian terbaik dapat dilihat pada gambar 8.



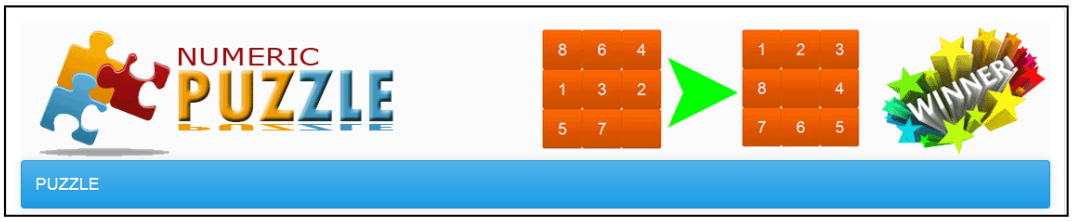
Gambar 8. Representasi Kondisi *Plateau*

D. Implementasi Perangkat Lunak

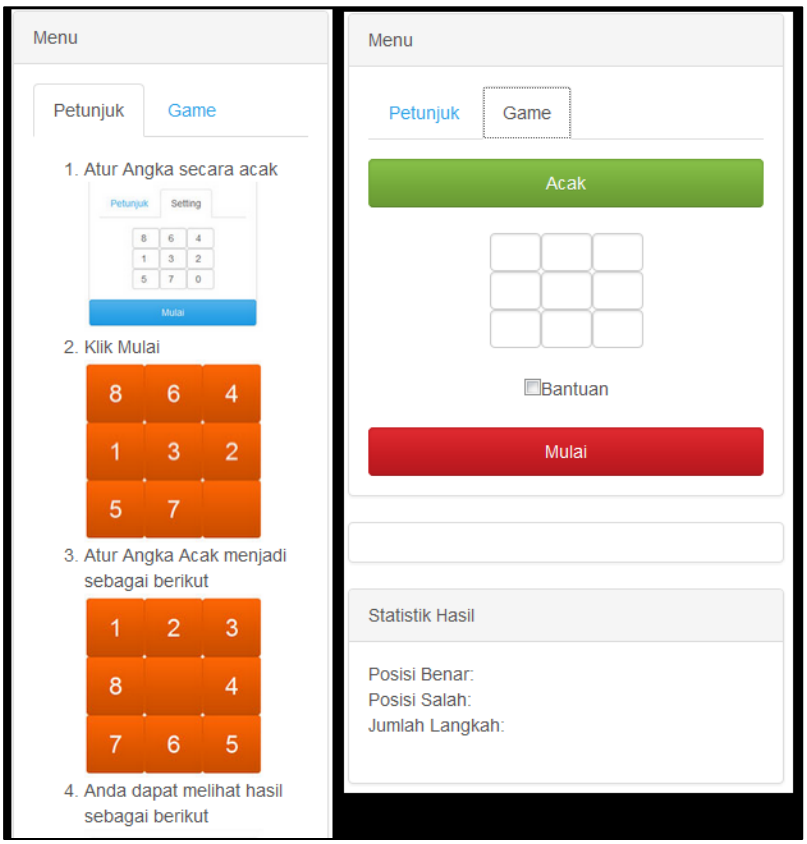
Implemetasi merupakan tahap dimana aplikasi siap untuk dioperasikan pada tahap yang sebenarnya, hal ini dilakukan untuk mengetahui apakah aplikasi yang dibuat telah sesuai seperti yang direncanakan. Pada implementasi perangkat lunak ini akan menjelaskan bagaimana aplikasi tersebut berjalan. Implementasi dari aplikasi *puzzle* 8 angka ini terdiri dari beberapa halaman yang memiliki fungsi sendiri-sendiri, halaman tersebut akan tampil secara terurut sesuai dengan urutan yang telah terprogram, setelah pengguna melakukan proses tertentu.

1. Halaman Antarmuka Awal

Halaman antarmuka awal merupakan tampilan yang pertama kali muncul saat aplikasi dijalankan. Pada halaman ini terdapat menu dari tampilan permainan *puzzle* 8 angka, diantaranya adalah; *form* menu yang berisi petunjuk penggunaan aplikasi dan *game*, *form initial state*, *form goal state*, *form* statistik, tombol acak, tombol bantuan dan tombol mulai. Masing-masing tombol tersebut memiliki fungsi nya tersendiri, tampilan halaman utama dari permainan *puzzle* 8 angka dapat dilihat pada Gambar 9 dan Gambar 10.



Gambar 9. Tampilan *Form Initial State* dan *Goal State*



Gambar 10. *Form* Menu

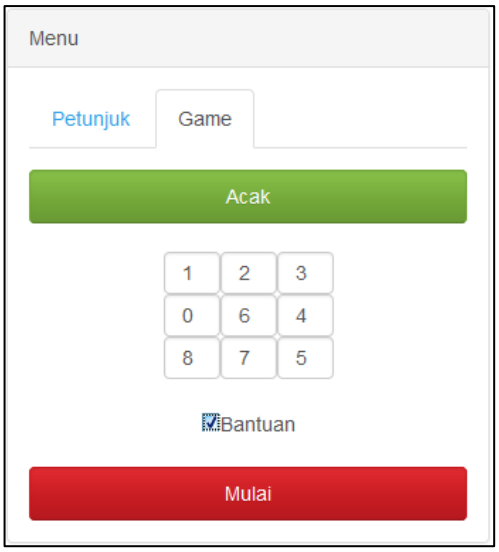
Pada *form* menu ini terdapat beberapa halaman lagi seperti *form* petunjuk dan *form game*, pada *form* petunjuk pengguna aplikasi dapat melihat langkah-langkah cara pemakaian aplikasi yang benar, sedangkan pada *form game* berisi seperti yang dijelaskan pada *form* petunjuk. Untuk *form* statistik hasil berguna untuk melihat *heuristic* yang ditunjukan berdasarkan posisi angka yang benar dan posisi angka yang salah, kemudian disertai pula dengan perhitungan langkah yang bekerja ketika pengguna melakukan pergeseran kotak-kotak angka, dapat dilihat pada Gambar 11.

Gambar 11.*Form* Statistik Hasil

2. Halaman Antarmuka Proses

Halaman antarmuka proses berguna untuk menampilkan langkah-langkah dari proses pergeseran angka dan kotak kosong. Untuk proses pengujian pengguna perlu menginputkan atau dapat juga melakukan klik pada tombol acak, setelah *form initial state* berisi angka pengguna dapat memilih apakah penyelesaian *puzzle* tersebut akan

diselesaikan menggunakan metode pencarian atau tidak, ditunjukan dengan malakukan cek pada tombol bantuan, dapat dilihat pada Gambar 12.



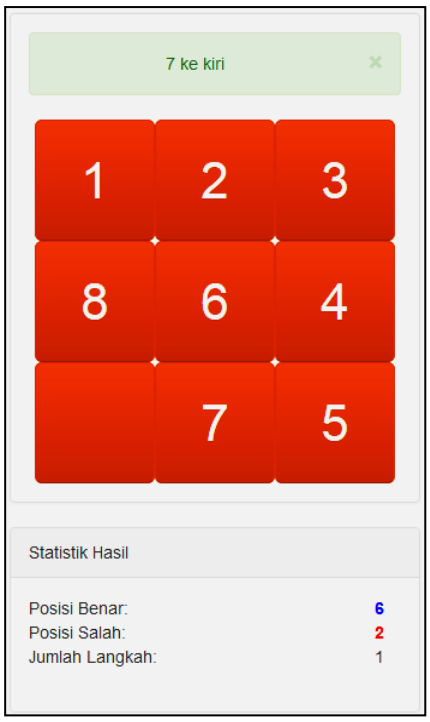
Gambar 12. *Form* Inisialisasi

Proses pengujian akan berjalan ketika pengguna melakukan klik pada tombol mulai, setelah itu sistem akan melakukan pencarian *heuristic* untuk menentukan langkah-langkah pergeseran kotak kosong, dan untuk *heuristic* tertinggi ditandai dengan solusi pegeseran angka yang diberikan oleh sistem, dapat dilihat pada contoh berikut:

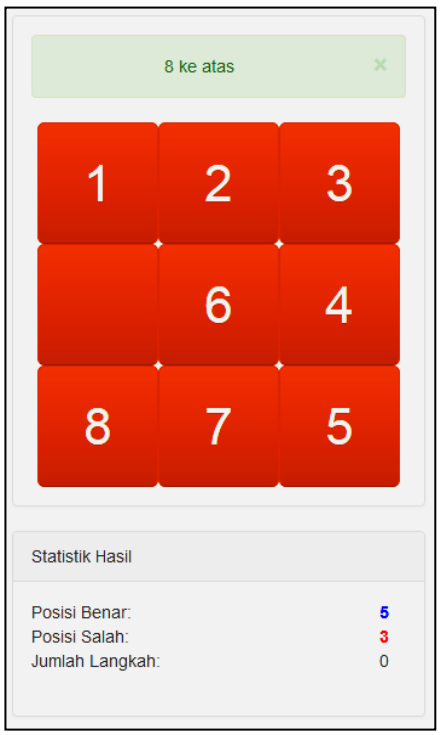
a) Langkah ke 1
Langkah ke 1 mendapati 3 keadaan yang dari masing-masing keadaan memiliki *heuristic* h=4 apabila angka 1 bergeser ke bawah, h=5 apabila angka 6 bergeser ke kiri, h=6 apabila angka 8 bergeser ke atas. Dari keadaan ini maka sistem akan memberikan solusi pergeseran dengan *heuristic* tertinggi yaitu h=6 atau angka 8 bergeser ke atas. Dapat dilihat pada Gambar 13.

b) Langkah ke 2
Langkah ke 2 mendapati 2 keadaan yang dari masing-masing keadaan memiliki *heuristic* h=5 apabila angka 8 bergeser ke bawah, h=7 apabila angka 7 bergeser ke kiri. Dari keadaan ini maka sistem akan memberikan solusi pergeseran dengan *heuristic* tertinggi yaitu h=7 atau angka 7 bergeser ke kiri. Dapat dilihat pada Gambar 14.

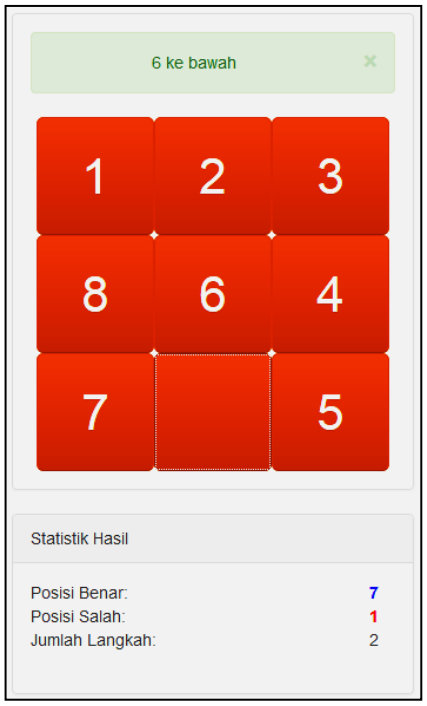
c) Langkah ke 3
Langkah ke 3 mendapati 3 keadaan yang dari masing-masing keadaan memiliki *heuristic* h=6 apabila angka 7 bergeser ke kanan, h=6 apabila angka 5 bergeser ke kiri, h=8 apabila angka 6 bergeser ke bawah. Dari keadaan ini maka sistem akan memberikan solusi pergeseran dengan *heuristic* tertinggi yaitu h=8 atau angka 6 bergeser ke bawah. Dapat dilihat pada Gambar 15.



Gambar 13. Tampilan Langkah ke 1



Gambar 14. Tampilan Langkah ke 2



Gambar 15. Tampilan Langkah ke 3

3. Halaman Antarmuka Proses Selesai

Setelah melakukan klik pada tombol mulai hingga menemukan nilai h yang diharapkan yaitu h=8 maka akan tampil suatu tanda bahwa proses yang dilakukan telah selesai. Itu adalah tujuan akhir dari proses pencarian langkah-langkah *puzzle* 8 angka, dapat dilihat pada Gambar 16.

Gambar 16. Tampilan Proses Selesai

E. Pengujian Perangkat Lunak

Pada tahap ini dilakukan pengujian terhadap fungsionalitas aplikasi untuk memastikan aplikasi dapat berjalan dengan baik dan dapat memenuhi *requirement* aplikasi sebagaimana didefinisikan pada tahap analisis kebutuhan. Hasil pengujian

berdasarkan fungsi dari penentuan kedudukan awal (initial state), proses, dan kedudukan akhir (goal state) menunjukkan bahwa aplikasi berjalan sesuai dengan yang diharapkan. Selain itu, pengujian lain yang dilakukan adalah pengujian manfaat. Hasil pengujian manfaat menunjukkan bahwa aplikasi mampu memberikan informasi mengenai langkah-langkah penyelesaian *puzzle* 8 angka dan memberikan kemudahan dalam memainkannya.

KESIMPULAN

Berdasarkan hasil penelitian, dapat disimpulkan bahwa metode Hill Climbing dapat digunakan sebagai acuan dalam mengembangkan aplikasi *puzzle* 8 angka. Selain itu, aplikasi yang dibangun dapat memberikan informasi tentang langkah-langkah pergeseran dalam menyelesaikan permainan *puzzle* 8 angka. Sebagai saran yang dapat diberikan, yaitu aplikasi dapat dikembangkan lebih lanjut untuk kondisi *plateau* dan *local maximum* dalam menentukan aturan yang tepat seperti misalnya prioritas pergerakan ubin kosong, dan untuk kebutuhan masukan *puzzle* dapat ditambahkan bentuk gambar ataupun huruf.

DAFTAR PUSTAKA

Emanuel, A.W.R., Witono, T., Handaya, W.B.T., 2008, *Cara Praktis Membangun Situs e-Learning dengan Teknologi Open Source*, Graha Ilmu, Yogyakarta.

Jokodo, 2011, *Game Puzzle Number*, Universitas Guna dharma, Jakarta.

Kadir, A., 2001, *Dasar Penggunaan Web Dinamis dengan PHP*, ANDI, Yogyakarta.

Kusumadewi, S., 2003, *Artificial Intelegence*, Graha Ilmu, Yogyakarta.

Kusumadewi, S., dan Purnomo, H., 2005, *Penyelesaian Masalah Optimasi dengan Teknik-Teknik Heuristik*, Graha Ilmu, Yogyakarta.

Utomo, P, A., 2008, *Membangun Aplikasi PHP dan AJAX Tanpa Mengenal Script*, ANDI, Yogyakarta.

Yuliana., Ananda., Surya, I., 2012, Implementasi Algoritma A Star pada Pemecahan Puzzle 8, *Jurnal Teknik Informatika*,Vol. 1, Halaman 1-9.

Zakiah, A., 2012, Penyelesaian Masalah 8 Puzzle dengan Algoritma Hill Climbing Stepest Ascent Loglist Heuristik Berbasis Java, *Seminar Nasional Teknologi Informasi dan Komunikasi*, ISSN: 2089-9815, Yogyakarta, 10 Maret 2012.