**Implementasi Algoritma Greedy Best First Search dan A\***

**pada Pemecahan Puzzle 8**

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 heuristik. Heuristik adalah sebuah teknik yang menggambarkan efisiensi dalam proses pencarian. Algoritma yang memakai fungsi heuristik diantaranya adalah Algoritma Greedy dan A\*. Fungsi heuristik yang digunakan pada algoritma Greedy adalah dengan melihat banyaknya kotak pada posisi yang salah, sedangkan pada algoritma A\* fungsi heuristik yang digunakan adalah dengan melihat banyaknya kotak pada posisi yang salah dan total keseluruhan jarak dari kota yang berada di tempat yang salah untuk mencapai posisi benar. Fungsi ini sering disebut dengan manhattan distance.

**Pemecahan dengan Algoritma Greedy Best First Search**

Algoritma Greedy Best First Search memiliki prinsip yaitu mengambil keputusan yang dianggap terbaik hanya untuk saat itu saja yang diharapkan dapat memberikan solusi terbaik secara keseluruhan. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.

Greedy Best first search seperti halnya algoritma yang menggunakan strategi Best first search lainnya mempunyai sebuah fungsi yang menjadi acuan kelayakan sebuah simpul yaitu fungsi evaluasi . Pada Greedy Best first search fungsi evaluasi tidak bergantung pada cost sebenarnya, tetapi hanya tergantung pada fungsi heuristik itu sendiri. Jika pada algoritma A\* pencarian yang dilakukan bergantung pada cost sebenarnya dari sebuah simpul yaitu , pada Greedy Best first search fungsi evaluasi hanya bergantung pada fungsi heuristik  yang mengestimasikan arah yang benar, sehingga pencarian jalur dapat berlangsung dengan sangat cepat. Secara matematis fungsi evaluasi pada greedy search diberikan oleh:



dengan:

 = total keseluruhan biaya grid yang ada di posisi salah

Algoritma ini akan menghitung nilai  dengan memeriksa jumlah kotak yang berada di posisi yang salah. Semakin banyak jumlah kotak yang berada di posisi yang salah, maka mungkin saja masih banyak langkah yang harus ditempuh. Semakin sedikit jumlah kotak yang berada di posisi yang salah, maka mungkin saja puzzle sudah hampir mendekati penyelesaian.

Rumus heuristic ini diterapkan pada kotak yang mungkin untuk digerakkan. Kemudian dipilih heuristic yang paling optimal diantara semua kemungkinan tadi. Kotak yang terpilih, akan digerakkan ke kotak yang kosong, lalu akan dibangkitkan lagi anak pohon dari status yang sekarang. Dan memulai lagi proses penentuan heuristic untuk kemungkinan kotak yang baru.

**Pemecahan dengan Algoritma A\***

Algoritma A\* merupakan salah satu algoritma yang menggunakan fungsi biaya. Algoritma A\* memeriksa kelayakan biaya yang diperlukan untuk mencapai suatu simpul dari sebuah simpul lain. Dalam kasus puzzle 8 ini, algoritma A\*membandingkan 2 posisi puzzle yaitu posisi puzzle awal (state awal) dengan posisi puzzle yang terurut dengan benar (state akhir).

Rumus yang digunakan oleh algoritma A\* yaitu:



dengan:

 = total keseluran biaya untuk mencapai posisi benar

 = total keseluruhan biaya grid yang ada di posisi salah

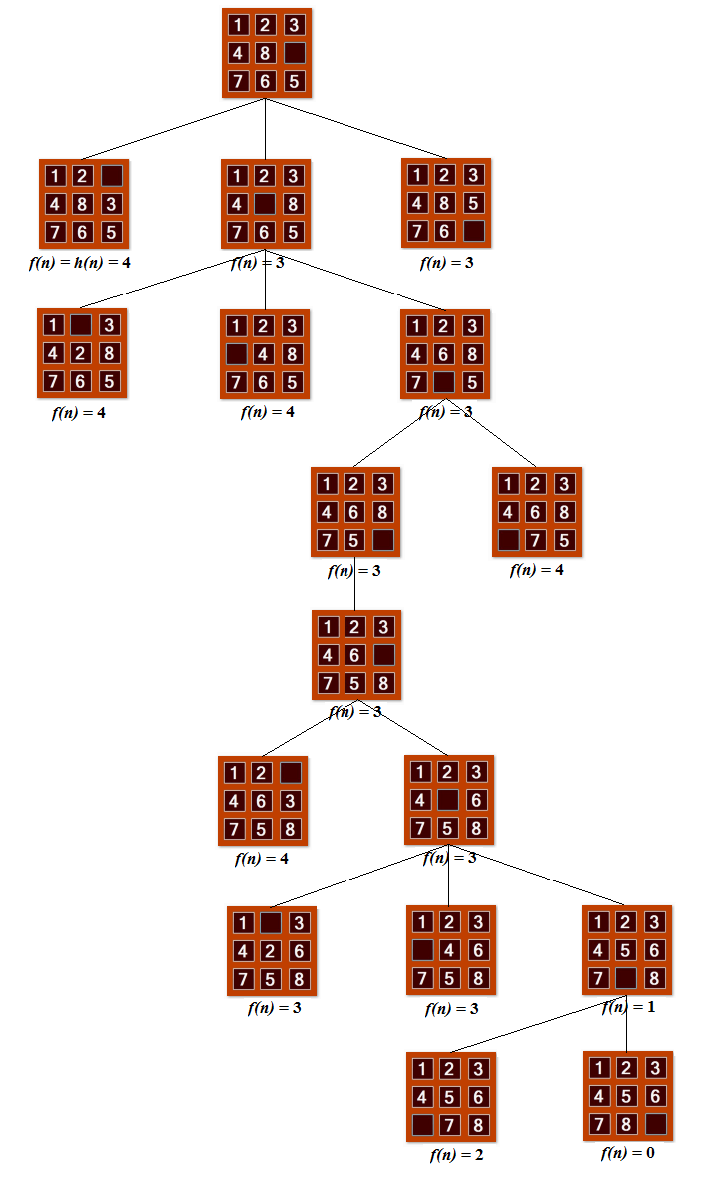
Algoritma ini akan menghitung nilai  yaitu keseluruhan biaya untuk mencapai posisi benar dan  yaitu dengan memeriksa jumlah kotak yang berada di posisi yang salah. Lakukan langkah-langkah diatas sampai ditemukan semua jalur atau langkah untuk mengembalikan Puzzle 8 ke posisi yang berurut.

Perbedaan algoritma A\*dan algoritma Greedy terletak pada rumus yang digunakan oleh kedua algoritma, algoritma greedy hanya menggunakan rumus perkiraan atau estimasi saja tetapi algoritma A\* selain menggunakan rumus perkiraan atau estimasi, juga menghitung cost yang diperlukan untuk mengembalikan puzzle ke posisi berurut. Inilah yang membuat algoritma A\*lebih baik daripada algoritma Greedy. Tetapi dengan lebih banyaknya rumus yang dihitung, hal ini menyebabkan algoritma A\* bekerja dengan lambat sehingga waktu yang diperlukan untuk menemukan solusi akan semakin besar pula karena selain menghitung biaya yang diperlukan untuk berjalan dari simpul satu ke simpul lainnya, Algoritma A\* juga menggunakan fungsi untuk memprioritaskan pemeriksaan simpul-simpul arah yang benar.

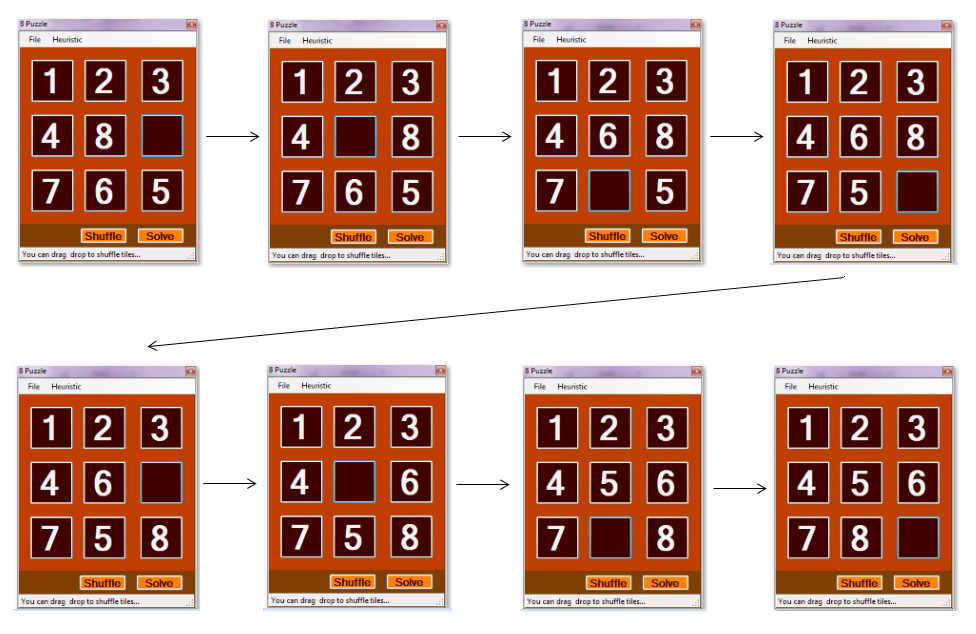
Implementasi algoritma Greedy dan A\* pada puzzle 8 dapat dilihat pada aplikasi di bawah ini:



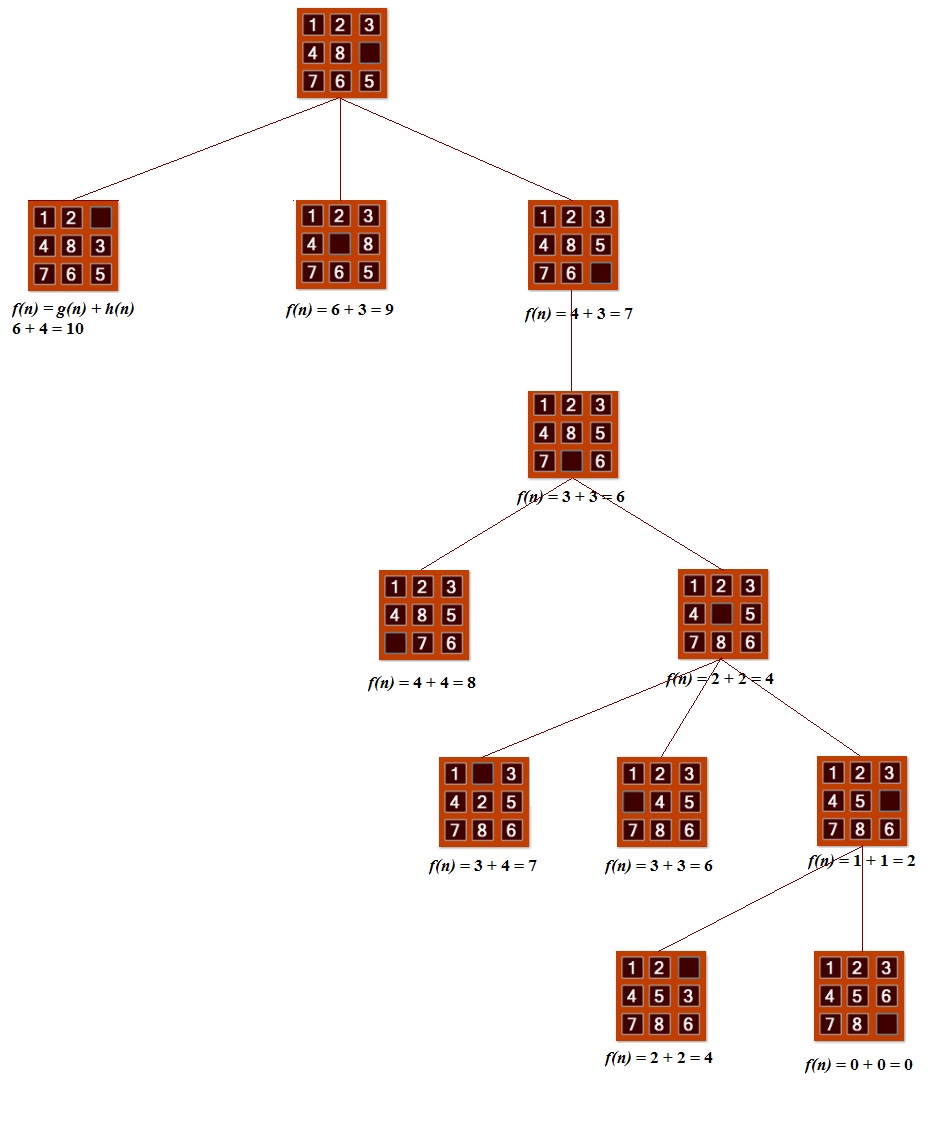
Urutan pemecahan masalah untuk puzzle diatas menggunakan algoritma Greedy adalah sebagai berikut:



Sehingga urutan pemecahannya adalah:



Urutan pemecahan masalah untuk puzzle diatas menggunakan algoritma A\* adalah sebagai berikut:



Sehingga urutan pemecahannya adalah:

