Penerapan Algoritma *Greedy Best First Search* untuk Menyelesaikan Permainan *Chroma Test : Brain Challenge*

Ikhwanul Muslimin/13514020
Program Studi Teknik Informatika, Fakultas Teknik Elektro dan Informatika
Institut Teknologi Bandung (ITB)
Bandung, Indonesia
13514020@std.stei.itb.ac.id

***Abstraksi*—Algoritma *greedy best first search* adalah salah satu algoritma untuk menyelesaikan persoalan dengan waktu yang lebih cepat daripada algoritma pencarian pertama mendalam (*depth first search*/DFS) dan algoritma pencarian pertama melebar (*breadht first search*/BFS). Permainan *Chroma test: brain challenge* adalah jenis permainan puzzle yang memiliki misi untuk menyamakan seluruh warna yang ada dalam permainan tersebut dengan batasan langkah yang ditentukan. Permainan ini memerlukan pertimbangan terhadap batasan langkah yang boleh dilakukan. Algoritma Greedy *best first search* dapat digunakan untuk menyelesaikan permainan ini dengan mempertimbangkan nilai heuristik yang dibuat sedemikian sehingga optimal.**

Keywords—greedy best first search, heuristik, pohon, optimal, pembangkitan, simpul.

I. PENDAHULUAN

Permainan *puzzle* adalah jenis permainan yang biasanya sederhana, tidak membutuhkan sumber daya yang besar, dan mengutamakan logika dalam menyelesaikannya. Permainan jenis *puzzle* biasanya dimainkan dalam batasan langkah, waktu, atau hal lainnya yang sederhana. Karena kesederhanaannya ini, terkadang menjadi tren.

Tujuan permainan jenis puzzle biasanya hanya untuk hiburan ringan di waktu luang. Permainan puzzle dapat dimainkan kapan saja dengan waktu yang relatif singkat untuk satu level, berbeda dengan video game yang sering memakan waktu banyak untuk satu misi. Permainan puzzle dapat dikatakan ramah sumber daya, karena tidak berjalan di latar belakang seperti video game.

Salah satu permainan puzzle yang cukup menarik bernama "*Chroma Test : Brain Challenge*" yang dibuat oleh Playtouch. Permainan ini tersedia gratis untuk Android dan iOS. Dalam permainan ini, ditampilkan kotak berisi warna-warna yang disusun secara acak. Misi dalam permainan ini hanyalah menjadikan warna-warna tersebut menjadi satu warna dengan cara memilih warna yang ingin dihilangkan dengan tepat, karena dibatasi oleh jumlah langkah yang diperbolehkan.

Permasalahan dalam permainan ini adalah bagaimana caranya memilih warna yang tepat untuk dihilangkan (disatukan) dalam langkah selanjutnya. Karena jika salah langkah, jumlah langkah yang diperlukan bisa jadi melebihi yang dibolehkan. Algoritma Greedy *best first search* merupakan salah satu algoritma yang dapat digunakan untuk menyelesaikan ini, dengan mempertimbangkan nilai heuristik yang dibuat sedemikian sehingga optimal.

II. DASAR TEORI

A. Algoritma Greedy Best First Search

Algoritma *greedy best first search* adalah salah satu cabang dari algoritma *best first search*. Algoritma *best first search* merupakan metode untuk membangkitkan simpul dari simpul sebelumnya (yang saat ini adalah simpul terbaik menurutnya). Beberapa terminologi dalam algoritma *best first search* adalah sebagai berikut[5].

(a) Fungsi evaluasi *f(n)*, adalah fungsi yang digunakan untuk membangkitkan simpul dari simpul sebelumnya.
(b) Nilai heuristik *h(n)*, adalah nilai perkiraan yang menjadi dasar dipilihnya simpul saat ini menjadi simpul terbaik.
(c) Nilai sebenarnya *g(n)*, adalah nilai "jarak" antara simpul akar dengan simpul saat ini.
(d) Simpul awal disebut juga simpul akar, adalah simpul pertama dalam pohon yang akan dibentuk.
(e) Simpul sekarang adalah simpul yang sedang dievaluasi dengan fungsi evaluasi untuk ditentukan apakah layak atau tidak menjadi solusi.
(f) Kandidat (suksesor) adalah simpul selanjutnya yang hendak diperiksa.
(g) *Open list*, adalah daftar simpul yang mungkin diakses dari simpul awal atau simpul yang sedang dijalankan.
(h) *Close list*, adalah daftar simpul yang saat ini menjadi solusi sementara, yaitu solusi terbaik saat ini.
(i) Simpul tujuan, adalah simpul yang hendak menjadi tujuan akhir.

Algoritma Best First Search dibagi menjadi dua, yaitu Greedy *best first search* dan A star. Dalam makalah ini, digunakan algoritma Greedy *best first search*.

Algortima Greedy *best first search*, atau biasa disingkat *Greedy Search* saja, sesuai dengan namanya yang berarti rakus, tamak, prinsipnya adalah mengambil keputusan berdasarkan informasi terbaik saat itu (terbesar untuk kasus maksimasi, atau terkecil untuk kasus minimasi) tanpa mempertimbangkan konsekuensi ke depan[1]. Keputusan yang diambil saat ini diharapkan dapat mengantarkan kepada solusi terbaik di akhir. Keputusan yang telah diambil saat ini tidak dapat dibatalkan.

Dalam *Greedy Search*, keputusan diambil menggunakan fungsi evaluasi *f(n)* tetapi tanpa mempertimbangkan nilai sebenarnya *g(n)* karena informasi nilai *g(n)* tidak berguna sebab *Greedy Search* tidak peduli dengan nilai sebenarnya. *Greedy Search* hanya akan mempertimbangkan nilai heuristik *h(n)* yang ditetapkan dengan aturan tertentu sedemikian rupa sehingga simpul yang dipilih adalah simpul terbaik.

$$f(n) = h(n)$$

Langkah-langkah dalam algoritma *Greedy Search* adalah sebagai berikut[1].

(i) Masukkan simpul awal ke *Open List*.
(ii) Tentukan metode untuk menentukan nilai heuristik *h(n)*.
(iii) Bangkitkan simpul dan hitung nilai heuristik *h(n)* untuk setiap simpul.
(iv) Pilih simpul dengan *h(n)* paling optimal.
(v) Perbarui nilai setiap simpul yang berubah karena keadaan tertentu (misalnya karena digabung dengan simpul lain).
(vi) Jika telah mencapai solusi, berhenti. Jika belum, ulangi langkah (ii) sampai ditemukan solusi.

Penggunaan algoritma *Greedy Search* dapat menyebabkan keadaan tidak ditemukannya solusi, yaitu ketika simpul belum sampai ke simpul daun, tetapi batasan sudah dilanggar, misalnya nilai maksimal atau minimalnya. Dalam permainan *chroma* adalah banyaknya langkah yang dibolehkan.

Kode semu (*pseudo-code*) algoritma *Greedy Search* adalah sebagai berikut.

```
DEKLARASI VARIABEL
type simpul
       parent : simpul
       heuristik : integer
       anak : list of simpul
type solusi: simpul
```

```
function GreedyBFS(S: list of simpul) → list
of solusi
{ fungsi untuk mengembalikan solusi dari
himpunan simpul}

DEKLARASI KAMUS
sol : himpunan_solusi
stemp : simpul

ALGORITMA
stemp= akar S
while stemp bukan daun do
       bangkitkan anak stemp
       pilih anak yang nilainya paling optimal
       sol ← sol ∪ anakstemp
       S ← S-stemp
       stemp ← anakstemp
endwhile

return sol
```

Kompleksitas algoritma *Greedy Search* adalah $O(b^m)$.[2] Penjelasannya adalah sebagai berikut.

1. Kalang while-do
   Kompleksitasnya sebesar *O(b)*, dengan *b* adalah kedalaman maksimum pohon.

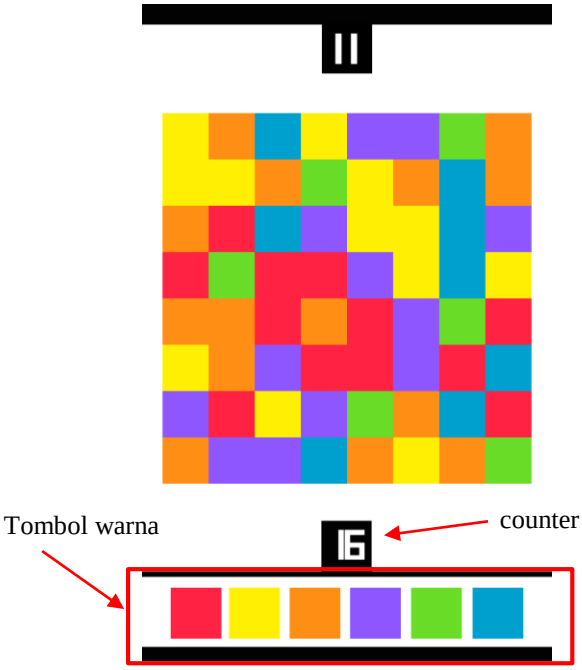
2. Kalang bersarang pemilihan anak paling optimal
   Kompleksitasnya sebesar O(*m*) untuk sekali pemilihan (mencari nilai maksimal atau minimal) dengan *m* adalah banyaknya simpul anak yang dimiliki oleh sebuah simpul.

Dengan demikian, kompleksitas waktunya adalah:
$T(b, m) = O(b) + b^m$
$= O(b^m)$

B. Chroma Test : Brain Challenge

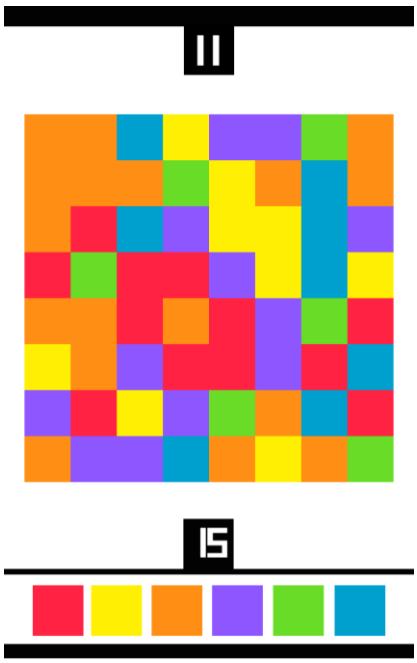
Chroma test : Brain Challenge adalah permainan bergenre puzzle yang dibuat oleh Playtouch. Permainan ini tersedia gratis bagi untuk ponsel pintar Android di Playstore dan iOS di Apple Store. Permainan ini juga dapat diakses melalui peramban web di http://playzool.com/games/chroma. Permainan ini membutuhkan kemampuan memilih langkah yang tepat untuk menyelesaikan misinya.

Dalam permainan ini, disajikan sebuah kotak persegi berukuran n x n (membesar sesuai level) yang berisi enam warna yang berbeda yang disusun secara acak. Terdapat 40 level dalam permainan ini yang masing-masing memiliki tingkat kesulitan sendiri.



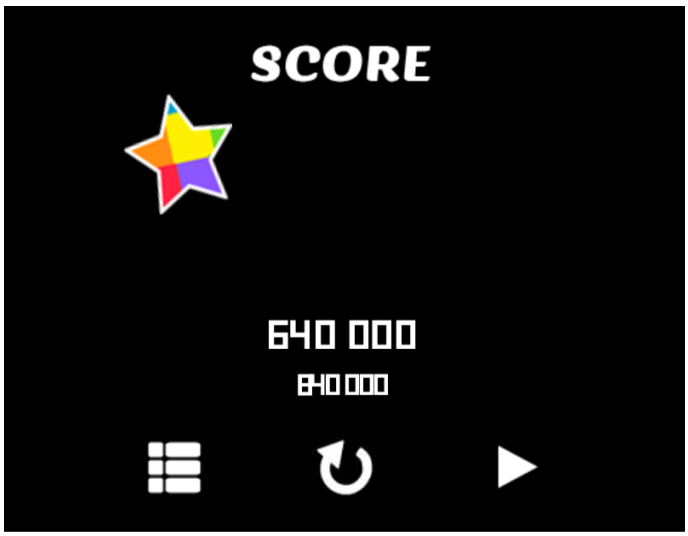
Gambar 1 Layar permainan Chroma dengan bagian-bagiannya

Misi permainan adalah menjadikan wana-warna ini menjadi warna tunggal, dimulai dari warna pojok kiri atas. Caranya adalah dengan menekan salah satu tombol warna. Tombol warna dapat ditekan maksimal sebanyak *counter*.
Contohnya apabila ditekan tombol jingga, maka warna kuning di pojok kiri atas akan menjadi jingga dan *counter* akan berkurang satu, sebagai berikut.



Gambar 2 Layar permainan setelah ditekan tombol warna

Kemenangan diperoleh jika kotak permainan menjadi satu warna dan langkah yang diambil kurang dari batasan yang diperbolehkan. Semakin banyak sisa *counter*, semakin besar skor yang diperoleh.



Gambar 3 Layar skor menang

III. Penerapan Algoritma Greedy Best First Search dalam Permainan Chroma

Seperti yang telah dijelaskan sebelumnya, bahwa dalam permainan ini diperlukan langkah yang tepat untuk menentukan warna apa yang akan ditekan agar tidak melanggar batasan *counter* dan mendapatkan hasil yang lebih baik (karena semakin banyak "nyawa" tersisa, semakin besar skor yang diperoleh).

Persoalan ini akan diselesaikan dengan algoritma *Greedy Search*. Oleh karena itu, perlu ditentukan terlebih dahulu fungsi evaluasinya (*f(n)*) yang hanya bergantung kepada nilai heuristiknya (*h(n)*).

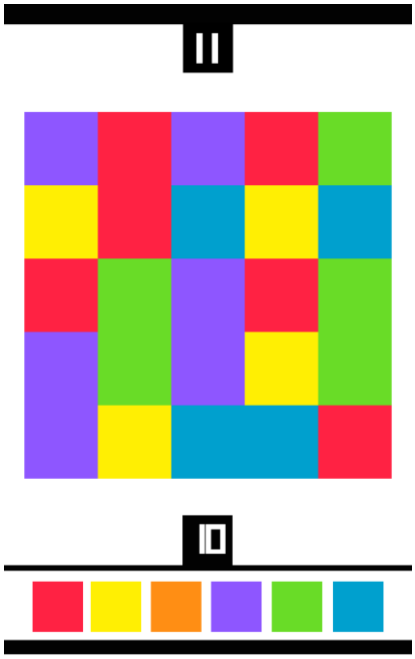
1. Menentukan nilai heuristik *h(n)*

Nilai heuristik harus dibuat seoptimal mungkin sehingga hasil yang diperoleh diharapkan optimal. Pada kasus ini, nilai heurstik yang dapat diambil adalah banyaknya warna yang sama untuk setiap warna yang mungkin dijangkau. Pada gambar 1, warna yang dapat dipilih adalah jingga dan merah. Banyaknya petak berisi warna jingga yang dapat dijangkau adalah 3, sedangkan warna merah hanya 1. Oleh karena itu, yang dipilih adalah warna merah.

Jika ada dua atau lebih warna yang memiliki jumlah yang sama, maka diambil simpul anak secara acak.

2. Merepresentasikan dalam bentuk list of simpul

Supaya lebih sederhana, diambil level 1.



Gambar 4 Chroma level 1

Representasi simpulnya sebagai berikut.

S = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}

Tabel 1 Representasi simpul

Simpul ke-	Warna	Jumlah	Anak
1	Ungu	1	2, 5
2	Kuning	1	1, 3&5
3	Merah	1	2, 4, 6
4	Ungu	2	3, 6, 7
5	Merah	2	1, 2, 6, 8, 9
6	Hijau	2	3, 4, 7, 5, 10
7	Kuning	1	4, 6, 11
8	Ungu	1	5, 9, 12
9	Biru	1	5, 8, 13, 10
10	Ungu	2	6, 9, 11, 14, 15
11	Biru	2	7, 10, 15, 19
12	Merah	1	8, 13, 16
13	Kuning	1	9, 12, 14, 16
14	Merah	1	10, 13, 15, 18
15	Kuning	1	10, 11, 14, 18
16	Hijau	1	12, 17
17	Biru	1	13, 16, 18
18	Hijau	2	14, 15, 17, 19
19	Merah	1	11, 18

3. Melakukan algoritma *Greedy Search*

Mula-mula, sol kosong.

Sol = {}

Selanjutnya dimulai dari simpul pertama, membangkikan simpul 2 dan 5. Simpul 2 dan 5 dibandingkan, ternyata simpul 5 lebih baik daripada simpul 2 karena warna memiliki jumlah keseragaman lebih banyak, yaitu dua, dibandingkan simpul 2 yang hanya 1.

sol = {5}

Setelah simpul 5 dipilih, simpul 1 dihapus, bergabung dengan simpul 5. Seluruh anak simpul 1 menjadi anak simul 5.

S = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}

Simpul 5 diperbarui sebagai berikut.

5	Merah	3	2, 6, 8, 9

Selanjutnya, simpul 5 membangkitkan simpul 2, 6, 8, dan 9. Dipilih simpul 6, simpul 5 dihapus, dan bergabung dengan simpul 6.

sol = {5, 6}

S = {2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}

Penggabungan simpul 5 dan simpul 6 menyebabkan seluruh anak dari simpul 5 adalah anak dari simpul 6. Sehingga, simpul 6 diperbarui sebagai berikut.

6	Hijau	5	2, 3, 4, 7, 8, 9, 10

Selanjutnya, simpul 6 membangkitkan simpul 2, 3, 4, 7, 8, 9, dan 10. Dipilih simpul 4, simpul 6 dihapus, dan bergabung dengan simpul 4. Seluruh anak simpul 6 menjadi anak simpul 4.

sol = {5, 6, 4}

Karena simpul 8 dan simpul 10 juga berwarna ungu, maka simpul 8 dan 10 bergabung dengan simpul 4. Seluruh anak simpul 8 dan 10 menjadi anak simpul 4. Simpul 8 dan simpul 10 dihapus.

S = {2, 3, 4, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19}

4	Ungu	10	2, 3, 7, 9, 11, 12, 14, dan 15

Akan tetapi, simpul 9 dan 11 memiliki warna yang sama, sehingga digabung menjadi simpul 11. Simpul 3, 12, dan 14 juga memilikiw warna yang sama sehigga digabung menajdi simpul 14. Simpul 3, 9 dan 12 dihapus. Kemudian simpul 4, 11, dan 14 diperbarui.

4	Ungu	10	2, 7, 11, 14, dan 15
11	Biru	3	4, 7, 13, 14, 15, 19
14	Merah	3	2, 4, 13, 15, 16, 18

S = {2, 7, 11, 13, 14, 15, 16, 17, 18, 19}

Selanjutnya, simpul 4 membangkitkan simpul 2, 3, 7, 11, 14, dan 15. Dipilih simpul 14, simpul 4 dihapus, dan bergabung dengan simpul 14. Seluruh anak simpul 4 menjadi anak simpul 14.

sol = {5, 6, 4, 14}

S = {2, 7, 11, 13, 14, 15, 16, 17, 18, 19}

14	Merah	13	2, 7, 11, 13, 15, 16, 18

Akan tetapi, karena simpul 2, 7, 13 dan 15 sama-sama berwarna kuning, keempatnya digabungkan menjadi simpul 15. Simpul 2, 7, dan 13 dihapus. Begitu juga dengan simpul 16 dan 18, keduanya sama-sama hijau, sehingga digabungkan menjadi simpul 18. Simpul 16 dihapus. Simpul 15 dan 18 diperbarui.

14	Merah	13	11, 15, 18
15	Kuning	4	11, 14, 17, 18
18	Hijau	3	14, 15, 17, 19

S = {11, 14, 15, 17, 18, 19}

Selanjutnya, dipilih simpul 15 dan simpul 14 digabung menjadi simpul 15.

sol = {5, 6, 4, 14, 15}

S = {11, 15, 17, 18, 19}

15	Kuning	17	11, 17, 18

Karena simpul 17 sama dengan simpul 11, simpul 11 digabung dengan simpul 17. Simpul 11 dihapus.

S = {15, 17, 18, 19}

15	Kuning	17	17, 18
18	Hijau	3	15, 19
17	Biru	4	15, 18,19

Selanjutnya, dipilih simpul 17. Simpul 15 digabung dengan simpul 17.

sol = {5, 6, 4, 14, 15, 17}

S = {18, 17, 19}

17	Biru	21	18, 19

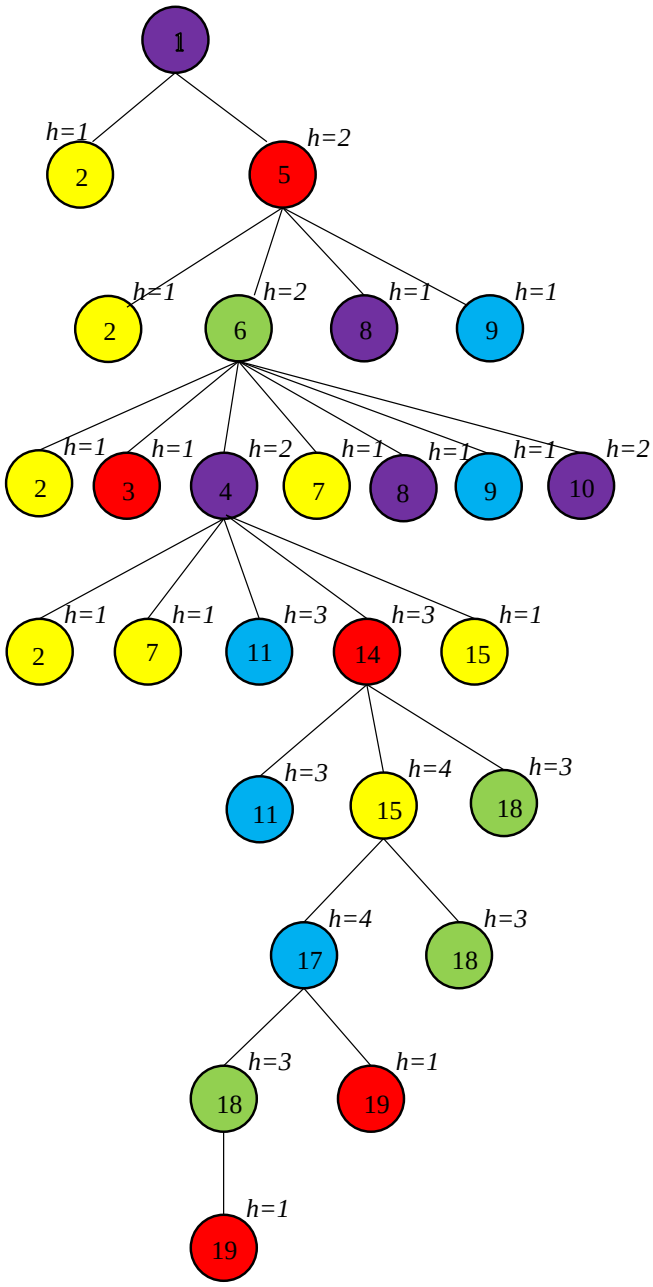
Selanjutnya dipilih simpul 18, dan akhirnya simpul 19.

sol = {5, 6, 4, 14, 15, 17}

S = {}

Solusi ditemukan.

Dari proses di atas, diperoleh langkah sebanyak enam. Jadi, dari total 10 kesempatan, masih tersisa 4 kesempatan. Dalam bentuk pohon, langkah-langkah tersebut dapat direpresentasikan sebagai berikut.



Gambar 5 Pohon pembangkitan simpul solusi

KESIMPULAN

Algoritma Greedy *Best First Search* dapat diterapkan di berbagai kasus, terutama kasus penentuan rute, langkah, jalan yang mangkus dan "murah". Kemangkusan algoritma ini bergantung pada bagaimana nilai heuristik diperoleh. Oleh karena itu, diperlukan teknik yang tepat dalam memilih nilai heuristik. Kesalahan pengambilan nilai heuristik kadang menjadikan algoritma ini kadang tidak menemukan solusi optimalnya.

Penerapan algoritma *Greedy Best First Search* pada permainan Chroma cukup mangkus dengan nilai heuristiknya adalah banyaknya petak dengan warna yang sama. Langkah yang diambil dapat lebih sedikit daripada kesempatan yang disediakan. Sehingga, skor yang diperoleh dapat lebih tinggi. Dengan kata lain, penerapan algoritma ini menjadikan peluang kemenangan lebih besar daripada hanya memilih secara acak.

Kelemahan dari algoritma ini adalah adanya potensi kesalahan, karena tidak menyimpan simpul yang mungkin dapat dipilih jika tidak ada simpul yang memenuhi solusi. Algoritma lain seperti A* bisa menjadi alternatif, tetapi memori yang dibutuhkan lebih besar daripada algoritma Greedy.

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah, karena dengan karunia-Nya, saya dapat menyelesaikan makalah ini dengan baik. Terima kasih saya ucapkan kepada Ibu Nur Ulfa Maulidevi dan Bapak Rinaldi Munir selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma, terutama pada bab algoritma *Best First Search*.

REFERENSI

[1] Munir, Rinaldi, *Diktat Kuliah IF2211, Strategi Algoritma*, Program Studi Teknik Informatika, STEI, ITB, 2009

[2] Bethopedia, *Best-First-Search*, http://wiki.bethanycrane.com/greedy-best-first-search, diakses 6 Mei 2016

[3] Asuarez, *Informed Search*, http://arantxa.ii.uam.es/~asuarez/docencia/ai/english/slides/T5_informedSearch.pdf, diakses 6 Mei 2016

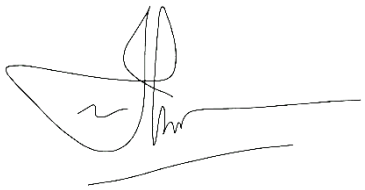
[4] Carnegie Mellon School of Computer Science, Greedy Best-First Search when EHC Fails http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume28/coles07a-html/node11.html, diakses 6 Mei 2016

[5] Repository Universitas Pendidikan Indonesia, Algoritma Greedy dan Algortima A*, http://a-research.upi.edu/operator/upload/s_mat_055961_chapter3.pdf, diakses 6 Meni 2016

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2016



Ikhwanul Muslimin/13514020