APLIKASI PENCARIAN JALUR TERPENDEK PADA RUMAH SAKIT UMUM BAHTERAMAS MENGGUNAKAN ALGORITMA A* (*A-STAR*)

[#1]Muh. Yamin, [#2]Moh. Bandrigo Talai
[#1,2]Jurusan Teknik Informatika, FTEKNIK UHO, Kendari

Abstrak

Pencarian jalur terpendek merupakan suatu permasalahan yang sering terjadi pada pengunjung rumah sakit untuk menemukan gedung atau ruangan yang dicari. Salah satu contohnya adalah pada Rumah Sakit Umum Bahteramas. Karena banyaknya gedung dan ruangan yang ada pada rumah sakit tersebut, mengakibatkan pengunjung kesulitan menemukan gedung dan ruangan yang dicari. Oleh karena itu dibutuhkan sistem yang dapat menunjukkan lokasi gedung dan ruangan beserta jalur terpendeknya, agar waktu pencarian lebih efisien. Terdapat beberapa algoritma pencarian jalur terpendek, salah satunya adalah algoritma A (A-Star). Algoritma A* menggunakan estimasi jarak terdekat untuk mencapai tujuan (goal) dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan. Heuristik adalah kriteria, metoda, atau prinsip-prinsip untuk menentukan pilihan sejumlah alternatif untuk mencapai sasaran dengan efektif. Hasil pada penelitian ini adalah aplikasi yang dapat menentukan jalur terpendek antara gedung dan antara ruangan yang diimplementasikan pada Operating System Android dan dibangun dengan menggunakan bahasa pemrograman Actionscript 3.*

Kata kunci : *algoritma A* (A-Star), android, actionscript 3, jalur terpendek.*

1. PENDAHULUAN

Rumah sakit sebagai salah satu organisasi sektor publik yang bergerak dalam bidang pelayanan jasa kesehatan merupakan organisasi yang sangat penting bagi masyarakat. Sebagai organisasi yang sangat dibutuhkan masyarakat, pelayanan yang diberikan rumah sakit tentunya sangat banyak mulai dari pengobatan, perawatan, dan lain sebagainya sehingga rumah sakit harus memiliki banyak ruangan yang akan digunakan untuk pelayanan kesehatan tersebut.

Sulawesi Tenggara khususnya Kota Kendari, rumah sakit telah banyak dibangun untuk memberikan pelayanan kesehatan kepada masyarakat Kota Kendari. Salah satu rumah sakit yang berada di Kota Kendari yaitu Rumah Sakit Umum Bahteramas. Rumah sakit ini memiliki beberapa gedung yang digunakan untuk kegiatan pelayanan rumah sakit seperti gedung administrasi, gedung IGD (Instalasi Gawat Darurat), gedung rawat inap, gedung bersalin, dan gedung-gedung lainnya. Masing-masing gedung juga memiliki beberapa ruangan dengan fungsi berbeda-beda. Karena banyaknya gedung dan ruangan, belum adanya sistem yang memberi petunjuk lokasi serta jalur tependek menuju gedung dan ruangan membuat pengunjung membutuhkan waktu yang cukup lama untuk menemukan gedung dan ruangan yang dicari. Bertanya kepada orang lain atau petugas rumah sakit menjadi cara untuk memperoleh informasi tentang lokasi ruangan yang dicari.

Untuk mengatasi permasalahan pencarian ruangan yang memerlukan waktu cukup lama, diperlukan sistem yang dapat menunjukkan lokasi serta menentukan jalur terpendek menuju ruangan yang dicari. Teknik pencarian yang sering digunakan untuk menentukan jalur terpendek yaitu pencarian buta (*blind search*) dan pencarian heuristik (*heuristic search*). Pencarian buta cenderung lebih mudah dipahami dibandingkan pencarian heuristik, tetapi hasil pencarian yang diperoleh pencarian *heuristic* lebih variatif dan waktu pencarian solusi lebih cepat. Salah satu metode pencarian jalur terpendek yang termasuk dalam pencarian *heurisitc* adalah Algoritma A*. Untuk mencapai tujuan dengan jarak tempuh terdekat, Algoritma A* memiliki suatu nilai heuristik yang digunakan sebagai dasar pertimbangan dimana estimasi nilai/biaya terkecil yang akan menentuan jarak tempuh terdekat. Berdasarkan hal-hal di atas,

maka dalam hal ini penulis ingin membuat Aplikasi Pencarian Jalur Terpendek Menuju Ruangan pada Rumah Sakit Umum Bahteramas Menggunakan Algoritma A*.

2. METODE PENELITIAN

2.1 Lintasan Terpendek (Shortest Path)

Lintasan Terpendek (*Shortest Path*) merupakan lintasan minimum yang diperlukan untuk mencapai suatu titik dari titik tertentu. Dalam pencarian lintasan terpendek masalah yang dihadapi adalah mancari lintasan mana yang akan dilalui sehingga didapat lintasan yang paling pendek dari satu *verteks* ke *verteks* yang lain.

Ada beberapa macam persoalan lintasan terpendek, antara lain : (Pawitri, dkk., 2007)

- Lintasan terpendek antara dua buah *verteks.*
- Lintasan terpendek antara semua pasangan *verteks.*
- Lintasan terpendek dari *verteks* tertentu ke semua *verteks* yang lain
- Lintasan terpendek antara dua buah *verteks* yang melalui beberapa *verteks* tertentu.

2.2 Graph

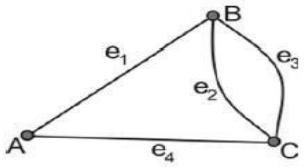
2. 2.1 Definisi Graph

Graph adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lainmelalui sisi/busur (*edges*) (Zakaria, 2006). Suatu Graf G terdiri dari dua himpunan

yaitu himpunan V dan himpunan E.

a. *Verteks* (simpul) :V = himpunan simpul yang terbatas dan tidak kosong.
b. *Edge* (sisi/busur):E = himpunan busur yang menghubungkan sepasangsimpul.

Dapat dikatakan graf adalah kumpulan dari simpul-simpul yang dihubungkan oleh sisi-sisi.



Gambar 2.1 Graf G

Gambar 2.1 menunjukkan graf terdiri dari himpunan V dan E dimana:

V = {A, B, C}

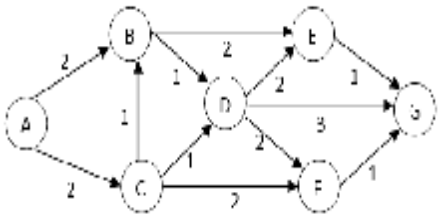
E = {e1, e2, e3, e4}

= {(A,B),(B,C),(B,C),(A,C)}

2.2.2 Macam –macam *graph*

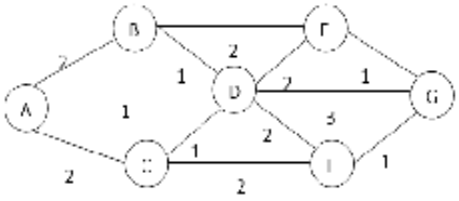
Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu :

1. Graf berarah dan berbobot : tiap busur mempunyai anak panah dan bobot. Gambar 2.2 menunjukkan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menujukkan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan titik C, dan seterusnya. Bobot antar titik A dan titik B pun telah di ketahui.



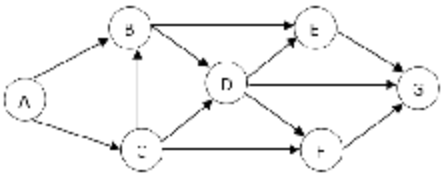
Gambar 2.2 ***Graf berarah dan berbobot***

2. Graf tidak berarah dan berbobot : tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.3 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



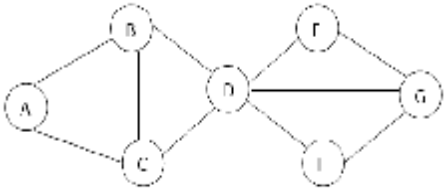
Gambar 2.3 ***Graf tidak berarah dan berbobot***

3. Graf berarah dan tidak berbobot: tiap busur mempunyai anak panah yang tidak berbobot. Gambar 2.4 menunjukkan graf berarah dan tidak berbobot.



Gambar 2.4 ***Graf berarah dan tidak berbobot***

4. Graf tidak berarah dan tidak berbobot: tiap busur tidak mempunyai anak panah dan tidak berbobot.



Gambar 2.5 ***Graf tidak berarah dan tidak berbobot***

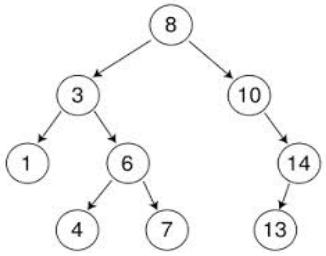
2.2.3 Tree

Tree merupakan struktur data yang mempunyai hubungan *one to many*. Hubungan *one to many* ini meliputi juga hubungan *one to one* atau *one to zero*, dapat dijelaskan bahwa satu *parent* bisa memiliki satu atau nol atau lebih dari satu *child*. Elemen dalam *tree* disebut dengan *node.*

Karakteristik dari *tree*adalah :

- Terdapat satu *node* yang unik, yang tidak memiliki predecessor. *Node* ini disebut *root*.
- Terdapat satu atau beberapa *node* yang tidak mempunyai *successor*. *Node* ini disebut *leaf*.
- Setiap *node* kecuali *root*, pasti memiliki satu *predecessor* yang unik.

- Setiap *node* kecuali *leaf*, pasti memiliki satu atau lebih *successor*.



Gambar 2.6 *Tree*

2.3. Algoritma A*

Algoritma A* merupakan perbaikan dari metode *best first search* dengan memodifikasi fungsi heuristiknya, Algoritma A* akan meminimumkan total biaya lintasan. Pada kondisi yang tepat, Algoritma A* akan memberikan solusi yang terbaik dalam waktu yang optimal (Kusumadewi , 2003).

Fungsi *f(n)* sebagai estimasi fungsi evaluasi terhadap *noden*, ditunjukkan oleh persamaan 2.1:

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

dengan :

$f(n)$ = fungsi evaluasi

$g(n)$ = biaya yang sudah dikeluarkan dari keadaan awal sampai keadaan *n*

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari *n*

Pada Algoritma A*, dibutuhkan 2 antrian, yaitu :

- *OPEN*, yang berisi *node-node* yang sudah dibangkitkan, sudah memiliki fungsi heuristic namun belum diuji.
- *CLOSED* berisi *node-node* yang sudah di uji

3. HASIL DAN PEMBAHASAN

Adapun *Software/Tools* yang digunakan dalam pembuatan aplikasi pencarian jalur terpendek ini adalah :

1. *AutoCAD 2010* : Menentukan *cost*/jarak sebenarnya antara dua *node*/titik yang berhubungan.
2. *Google SketchUP 8 Pro* : Desain denah dan gedung agar lebih menarik.
3. *Adobe Flash Professional CS6* : Desain aplikasi dan koding.

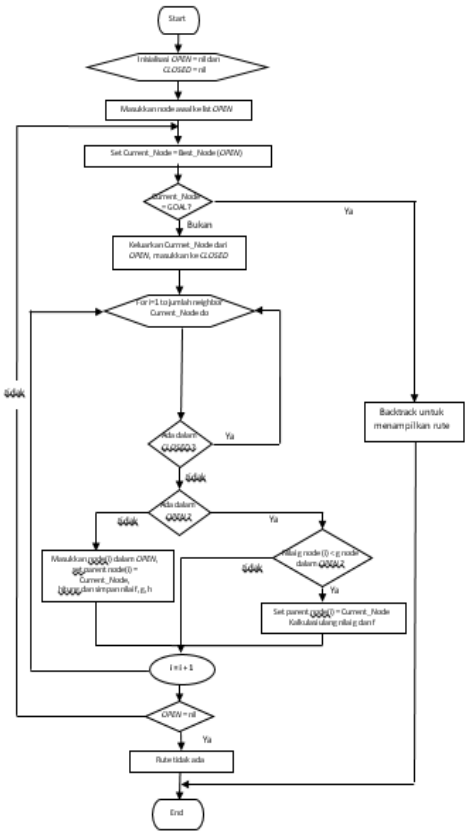
3.1 Analisis Masalah

Algoritma A* (*A-Star*) merupakan suatu algoritma yang termasuk pada kategori metode pencarian yang memiliki informasi (*informed search method*). Algoritma A* menggunakan estimasi jarak terdekat (*cost*/jarak sebenarnya) untuk mencapai tujuan (*goal*) dan memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan pemilihan jalur.

Adapun analisis algoritma A* yang akan dilakukan pada penelitian ini yaitu pada gambar 3.1 akan dicari jalur terpendek antara titik awal (titik S / gedung Delima) menuju ke titik tujuan (titik T / Gedung Asoka).

Gambar 3.1 Analisis Algoritma A*

Untuk menyelesaikan permasalahan pencarian jalur terpendek dengan menggunakan Algoritma A*,maka langkah-langkah untuk *flowchart* Algoritma A* yang ditunjukkan oleh gambar 3.2.



Gambar 3.2*Flowchart* Algoritma A*

Adapun persamaan dari algoritma A* ditunjukkan oleh persamaan :

$$f(n) = g(n) + h(n)$$

Dimana:

$f(n)$= fungsi evaluasi *node*/titik *n*

$g(n)$=*cost* yang sudah dikeluarkan dari *node*/titik awal sampai ke *noden*

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari *n*

Untuk menentukan nilai *h(n)* ditunjukkan oleh persamaan :

$$h(n) = \sqrt{(X_n - X_{goal})^2 + (Y_n - Y_{goal})^2}$$

dimana :

$h(n)$= nilai *heuristic* untuk *node/*titik *n*

X_n= nilai koordinat xdari *node/*titik *n*

Y_n= nilai koordinat ydari *node/*titik *n*

X_{goal}= nilai koordinat xdari *node/*titik tujuan

Y_{goal}= nilai koordinat ydari *node/*titik tujuan

3.2 Penentuan *cost*/biaya antara dua *node*/titik yang berhubungan

Untuk menentukan *cost* antara dua *node* yang berhubungan, akan dicari menggunakan fungsi *Measurement* pada *autocad 2010*. Hasil dari pengukuran nilai *cost* antara dua *node* yang saling berhubungan

a. *cost* node T ke *node* A dan sebaliknya adalah 14.10 meter
b. *cost* node A ke *node* B dan sebaliknya adalah 25.63 meter
c. *costnode* B ke *node* C dan sebaliknya adalah 45.48 meter
d. *costnode* B ke *node* F dan sebaliknya adalah 69.16 meter
e. *costnode* C ke *node* D dan sebaliknya adalah 67.12 meter
f. *costnode* D ke *node* E dan sebaliknyaadalah 4.17 meter
g. *costnode* E ke *node* F dan sebaliknya adalah 41.96 meter
h. *costnode* E ke *node* G dan sebaliknya adalah 23.08 meter
i. *costnode* G ke *node* T dan sebaliknya adalah 7.47 meter

3.3 Penentuan koordinat setiap *node*/titik

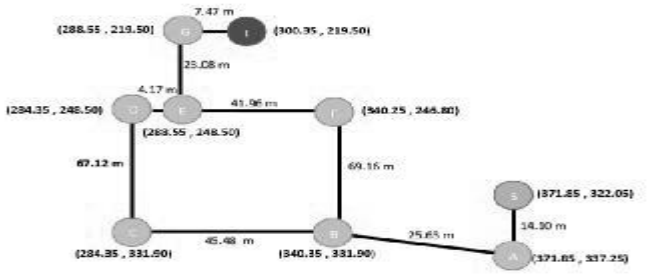
Untuk menentukan titik koordinat suatu *node,* denah pada *autocad* 2010 di *export* pada *Google SketchUp 2014* kemudian dibuat ulang dan dimasukkan ke dalam aplikasi *Adobe Flash CS6* dalam format *jpg*.

Adapun hasil penentuan titik koordinat setiap *node*/titik adalah sebagai berikut :

a. Koordinat *nodeS* (371.85 , 322.05)
b. Koordinat*nodeA*(371.85 , 337.25)
c. Koordinat *nodeB*(340.35 , 331.90)
d. Koordinat *nodeC* (284.35 , 331.90)

e. Koordinat *nodeD*(284.35 , 248.50)
f. *Koordinat nodeE* (288.55 , 248.50)
g. Koordinat*NodeF* (340.25 , 246.80)
h. Koordinat*NodeG* (288.55 , 219.50)
i. Koordinat*Node T* (300.35 , 219.50)

Tampilan (dalam bentuk *graph*) analisis algoritma A* setelah *cost*/biaya antara dua *node* yang berhubungan dan koordinat setiap *node* di tentukan ditunjukkan oleh gambar 3.3.



Gambar 3.3 Analisis Algoritma A* dalam bentuk *graph*

3.4 Penentuan koordinat setiap *node*/titik

Setelah mendapatkan nilai *cost*/biaya antara dua *node* yang berhubungan dan titik koordinat setiap *node*/titik, kemudian lakukan proses penghitungan nilai $f(n)$ dengan langkah-langkah sebagai berikut :

Langkah 1

- Set *OPEN* = {}
- Set *CLOSED* = {}
- *Node* Awal = S
- *Node* Tujuan= T

Langkah 2

- Masukkan *node* awal ke *OPEN* List
- *OPEN* = {S}
- set *Current_Node* = {S}

Langkah 3

- Periksa apakah *Current_Node* adalah *node* tujuan
- *Current_Node (node* S) bukan *node* Tujuan
- Karena *node* S bukan *node* tujuan, maka keluarkan *node* S dari *OPEN* dan pindahkan ke *CLOSED*
- *OPEN* = {}
- *CLOSED* = {S}

Langkah 4

- Hitung jumlah jalur yang bisa dilalui dari *Current_Node* (*neighbornode* S)
- Jumlah jalur yang bisa dilalui berjumlah 1 jaluryaitu jalur ke *node* A.
- Lakukan iterasi sebanyak 1 kali (for i=1 to jumlah *neighborCurrent_Node*)
- Iterasi 1 dilakukan pada *node* A
- Periksa apakah *node* A ada di *OPEN* atau di *CLOSED* atau tidak berada di *OPEN* maupun *CLOSED*
- *Node* A tidak berada di *OPEN* maupun *CLOSED*, sehingga *node* A dimasukkan ke *OPEN*
- *OPEN* = {A}
- Hitung nilai $g(A)$, $h(A)$ dan $f(A)$

  $g(A) = costnode$ S ke $node$ A

$g(A) = 14.10\ meter$

$h(A) = \sqrt{(371.85 - 288.55)^2 + (322.05 - 219.50)^2}$
$h(A) = \sqrt{6983.89 + 10516.50}$
$h(A) = 156.47$

$f(A) = g(A) + h(A)$
$f(A) = 14.10 + 156.47$
$f(A) = 170.57$

- Nilai $f(A)$adalah 170.57

Langkah 5

- nilai $f(n)$ terkecil adalah nilai $f(A)$
- *Current_Node* = {A}

Langkah 6

- *Current_Node* (*node* A) bukan *node* tujuan
- pindahkan *node* A dari *OPEN* ke *CLOSED*
- *OPEN* = {}
- *CLOSED* {S,A}

Langkah 7

- Hitung jumlah jalur yang bisa dilalui dari *Current_Node* (*neighbornode* A)
- Jumlah jalur yang bisa dilalui adalah 1 yaitu ke *node* B
- Lakukan iterasi sebanyak 1 kali (for i=1 to jumlah *neighborCurrent_Node*)
- Iterasi 1 dilakukan pada *node* B
- Periksa apakah *node* B ada di *OPEN* atau di *CLOSED* atau tidak berada di *OPEN* maupun *CLOSED*
- *Node* B tidak berada di *OPEN* maupun *CLOSED*, sehingga *node* B dimasukkan ke *OPEN*
- *OPEN* = {B}
- Hitung nilai $g(B)$, $h(B)$ dan $f(B)$

$$g(B) = \text{g(A)} + cost\ node \text{ A ke } node \text{ B}$$

$$g(B) = 14.10 \text{ meter} + 25.63\text{meter}$$

$$g(B) = 39.73\ meter$$

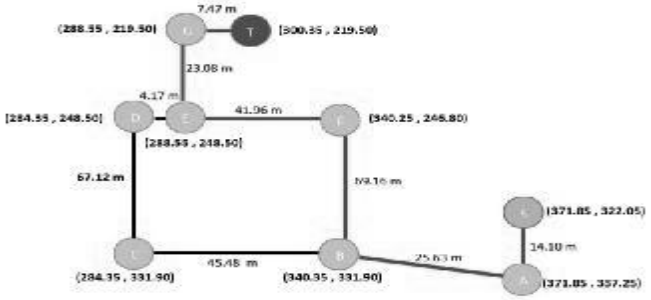
$$h(B) = \sqrt{(340.35 - 300.35)^2 + (331.90 - 219.50)^2}$$

$h(B) = \sqrt{1600 + 12633.76}$
$h(B) = \sqrt{14233.76}$
$h(B) = 119.30$
$f(B) = g(B) + h(B)$
$f(B) = 39.73 + 119.30$
$f(B) = 159.03$
Nilai dari $f(B)$adalah 159.03

Langkah 8

- *OPEN* = {B}
- *Current_Node* = {B}
- Ulangi langkah-langkah diatas pada *node-node* lain dengan menggunakan aturan yang ada pada *flowchart* sampai didapatkan jalur terpendeknya.

Adapun Hasil pencarian jalur terpendek menggunakan algoritma A* berdasarkan perhitungan langkah-langkah di atas ditunjukkan oleh gambar 3.4.



Gambar 3.4 Hasil analisis jalur terpendek menggunakan algoritma A*

Adapun implementasi pada *smartphone* adalah sebagai berikut :

***a. Form* Menu Utama**

Gambar 3.5*Form* menu

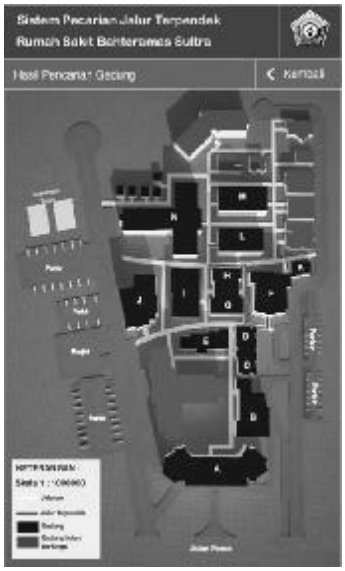
Pada *form* ini *user* memilih menu yang akan dibuka sesuai kebutuhannya, terdapat 3 menu, yaitu: Cari Ruangan, Bantuan, dan Keluar.

***b. Form* Cari Ruangan**

Gambar 3.6*Form* Cari Ruangan

Pada *form* ini *user* menentukan posisi awal berupa posisi gedung beserta ruangan *user* saat ini berada dan menentukan posisi tujuan berupa gedung berserta ruangannya. Setelah menentukan posisi awal dan tujuan, tekan tombol Cari Jalur untuk melihat jalur terpendek antara gedung posisi awal dengan posisi gedung tujuan.

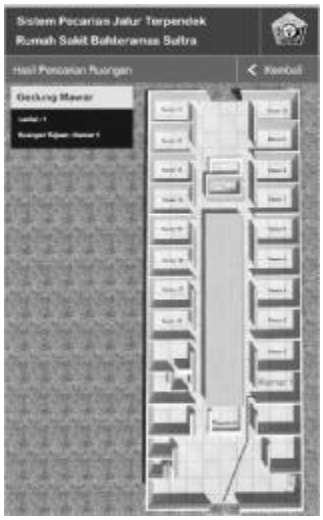
***c. Form* hasil pencarian gedung**



Gambar 3.7*Form* hasil pencarian gedung

Pada *form* ini akan ditampilkan jalur terpendek menuju gedung tujuan dari gedung posisi awal *user*. Untuk melihat jalur terpendek menuju ruangan, user dapat menekan gambar gedung tujuan.

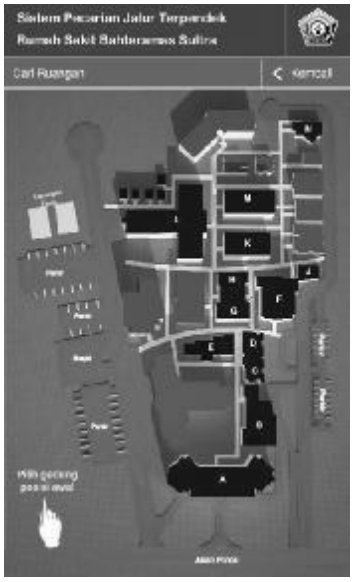
***d. Form* hasil pencarian ruangan**



Gambar 3.8*Form* hasil pencarian ruangan

Pada *form* ini *user* dapat melihat jalur terpendek menuju ruangan tujuan dari pintu masuk gedung.

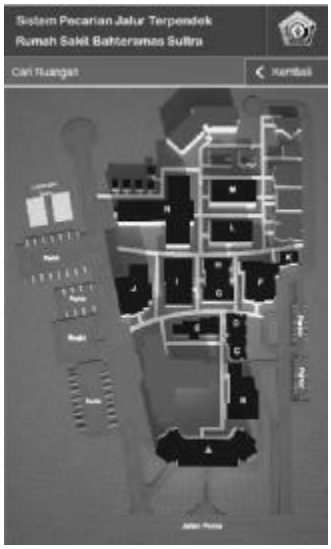
e. **Form Cari Gedung**



Gambar 3.9*Form* Cari Gedung

Pada *form* ini *user* dapat melihat jalur terpendek antara gedung posisi awal dengan gedung tujuan dengan cara menekan gambar gedung posisi awal dan gedung posisi tujuan.

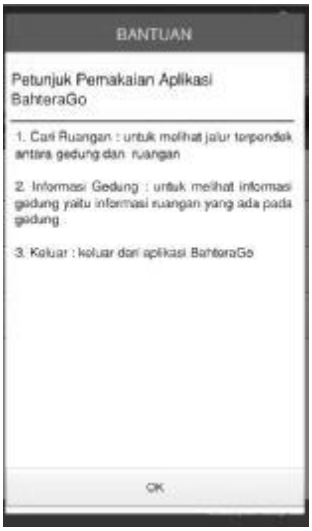
f. ***Form*** **Informasi Gedung**



Gambar 3.10*Form* Informasi Gedung

Pada *form* ini *user* dapat melihat informasi dari gedung yang dipilih.

***g. Form* Bantuan**



Gambar 3.11*Form* Bantuan

Pada *form* ini menampilkan petunjuk pemakaian dari aplikasi, sehingga memudahkan *user* untuk menggunakan aplikasi.

3.4 Analisis Hasil

Pada tahap ini akan dijelaskan analisis hasil kinerja dari aplikasi BahteraGo.

3.4.1 Analisis Pencarian Jalur Terpendek Gedung

Pada aplikasi BahteraGo, user terlebih dahulu menentukan posisi awal dan tujuan berupa gedung. Kemudian sistem akan menghitung jalur terpendek antara gedung menggunakan Algoritma A*.

Contoh :

- Posisi gedung awal : Gedung Delima
- Posisi koordinat gedung awal : (371.85 , 322.05)
- Posisi gedung tujuan : Gedung Asoka
- Posisi Koordinat gedung tujuan (300.35 , 219.50)
- Total jarak terpendek : 181.40 meter

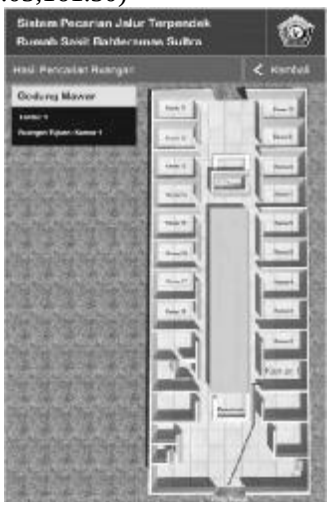
Gambar 3.12 Analisis pencarian jalur terpendek gedung

3.4.2 Analisis Pencarian Jalur Terpendek Ruangan

Pada aplikasi BahteraGo, user terlebih dahulu menentukan posisi awal dan tujuan berupa ruangan. Kemudian sistem akan menghitung jalur terpendek antara pintu masuk gedung dengan ruangan yang dicari.

Contoh:

- Posisi awal dalam gedung Asoka : Pintu masuk gedung mawar
- Koordinat posisi awal : (304.40,640.45)
- Posisi tujuan dalam gedung Asoka : Kamar 1
- Koordinat posisi tujuan : (300.05,161.50)



Gambar 3.13 Analisis hasil pencarian jalur terpendek ruangan

4. KESIMPULAN

Kesimpulan yang dapat diambil dari penulisan tugas akhir ini adalah sebagai berikut:

1. Untuk menentukan jalur terpendek pada pencarian ruangan dengan menggunakan algoritma A* (*A-Star*), dibutuhkan beberapa data yaitu, jarak sebenarnya antara *node* yang berhubungan (*cost* antar node) dan koordinat setiap *node.* Dengan data tersebut, maka pencarian jalur terpendek pada ruangan dapat diimplementasikan.
2. Aplikasi BahteraGo dibangun menggunakan aplikasi *Adobe Flash Profesional CS6* dengan bahasa pemrograman *actionscript 3* agar dapat diimplementasikan pada *smartphone* dengan *operating system* (OS) *Android* memanfaatkan fitur *Adobe Air.*

DAFTAR PUSTAKA

Chambers, M., Dura, D., Georgita, D. dan Hoyt, K., ***2008***, ***Adobe AIR*** *for JavaScript Developers Pocket Guide*, O'Reilly Media Inc., Canada.

Champeaux, D. dan Sint, L., 1977, *An improved bidirectional heuristic search algorithm*, Journal of the ACM 24 (2):177–191,doi: 10.1145 / 322003. 322004.

Doran, J.E. dan Michie, D., 1966, *Experiments with the Graph Traverser Program. Proc. Roy. Soc.*, Vol. 294, pp. 235-259.

Kusumadewi, S., 2003, *Artificial Intelligence (teknik dan Aplikasinya)*, Graha Ilmu, Yogyakarta.

Mario, I. dan Aryo., 2004, *Analisis Fungsi-Fungsi Heuristic Pada Algoritma Pathfinding A**, Skripsi S1 Universitas Bina Nusantara, Jakarta.

Patel, A.D., 2003, *Amit's Thoughts on Pathfinding*, Nature Neuroscience, 6, 674–681.

Pawitri, K., Ayu, Y. dan Joko, P., 2007, *Implementasi Algoritma PHYSICAL-A* (PHA*) untuk menemukan Lintasan Terpendek*, http://journal.amikom.ac.id/index.php/SN/article/view/2075, diakses 12 Juni 2014.

Pressman, R.S., 2001, *Software Engineering : A Practitioner Approach*, McGraw - Hill Companies.

Rossa, A.S., dan Shalahiddun, M. 2011, *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*, Modula, Bandung.

Rosenzweig, G., 2008, *ActionScript 3.0 Game Programming University*, Que Publishing, USA.

Russel, S. dan Peter, N., 1995, *Artificial Intelligence A Modern Approach Second Edition, Pearson Education*, Inc., New Jersey.

Wiitala, S.A., 1987, *Discrete mathematics a unified approach*, McGraw-Hill, New York.

Zakaria., 2006, *Teknologi Informasi dan Komunikasi*, Arya Duta, Jakarta.