Jurnal Ilmiah Ilmu Komputer, Vol 15 No. 2, Desember 2010 : 17 - 24

Penentuan Rute Terpendek Menggunakan Variasi Fungsi Heuristik Algoritme A* Pada Mobile Devices

Pandu Satria Nur Ananda, Sri Wahjuni, Endang Purnama Giri

Departemen Ilmu Komputer, Institut Pertanian Bogor, Jl. Meranti Wing 20 Lv.V, Bogor, Jawa Barat, 16680

Abstract---GPS (Global Positioning System) is a technology that can help people in reaching the destination place in a fastest time and shortest route. Many mobile devices, some of them are cellular phones, have GPS facility. But this type of cellular phone is still expensive. This research intended to develop a shortest path searching application for mobile devices using variation of heuristic function A* algorithm, such as Manhattan, Euclidian, and Square of Euclidian. In addition, we intended to compare the execution time of these heuristic functions in finding shortest route.

The data used in the experiment was Bogor Agricultural University map that obtained from Bogor Agricultural University Facility and Property Directorate. The map ought to modified to grid form to be appeared in cellular phone screen. The matrix that used to grid had a size of 15 x 12 and values of 0, 1, 5, 6, 7, 8, 9, and 10. Value of 0 expressed walkable route, and value of 1 expressed unwalkable route. The rest values expressed the map facilities, like faculties and rectorate buildings.

The experiments indicated that Square of Euclidian had faster execution time and less amount of checked nodes than other heuristic functions. The lack was that this algorithm did not always provide shortest route. Euclidian heuristic function had slower execution time and more amount of checked nodes than others, whereas Manhattan have faster execution time and less amount of checked node than Euclidian. Euclidian and Manhattan heuristic function always provide shortest route in finding destination.

Keyword: GPS, mobile devices, shortest path searching, heuristic function, A* Algorithm

PENDAHULUAN

Sebagian besar orang ingin mencapai tempat yang dituju dengan waktu yang cepat dan rute yang pendek. Pada umumnya, orang masih menggunakan cara yang konvensional dalam mencari rute agar mencapai tempat yang dituju dengan cepat, yaitu dengan membuka peta lalu mencari jalur yang dapat dilalui dari tempat asal ke tempat tujuan yang terdapat di peta atau bertanya ke orang lain. Mencari jalur yang terdapat di peta memerlukan waktu yang cukup lama sedangkan bertanya ke orang lain belum tentu ada yang tahu jalurnya. Dengan adanya kendala tersebut maka alangkah baiknya jika ada sebuah aplikasi yang menyediakan informasi mengenai lokasi tujuan dan rute terpendek untuk mencapainya.

Kebanyakan aplikasi pencarian rute terpendek masih terdapat pada komputer *desktop* (PC) dan *notebook* (Adipranata *et. al* 2007, Riftadi 2007). Sehingga apabila ingin mencari rute terpendek ke suatu lokasi, kita harus menyalakan komputer lalu mencari lokasi yang kita inginkan yang terdapat pada aplikasi pencarian yang telah terinstal di komputer. Hal ini juga membutuhkan waktu yang cukup lama.

Perkembangan teknologi pada *mobile phone* sekarang ini sudah sangat pesat. Dengan fitur yang ada, telepon seluler tidak hanya digunakan sebagai alat komunikasi saja, tetapi juga sebagai alat untuk memperoleh informasi. Fitur SMS (*Short Message Service*), GPRS (*General Packet Radio Service*), e-mail dan GPS (*Global Positioning System*) merupakan beberapa fitur yang memberikan kemudahan bagi pengguna telepon seluler untuk memperoleh informasi dengan cepat dan tepat. Fitur GPS pada telepon seluler dapat digunakan untuk memperoleh informasi lokasi yang diinginkan oleh pengguna. Informasi tersebut dapat diperoleh dengan cepat, namun hanya terdapat pada beberapa tipe telepon seluler tertentu yang sudah didukung oleh fasilitas GPS tersebut. Harga telepon seluler dengan dukungan GPS masih relatif mahal. Oleh karena itu diperlukan sebuah aplikasi alternatif untuk pengguna telepon seluler yang tidak dilengkapi dengan fasilitas GPS untuk memudahkan dalam menuju lokasi yang sudah ditentukan.

Penelitian ini menggunakan algoritme A* sebagai algoritme untuk pencarian rute terpendek. Algoritme A* memiliki nilai-nilai heuristik antara lain jarak Manhattan (*Manhattan Distance*), jarak Euclidian (*Euclidian Distance*), dan jarak Euclidian kuadrat.

Penelitian ini juga bertujuan untuk membangun suatu aplikasi pada perangkat *mobile* khususnya telepon seluler yang dapat menyajikan rute terpendek dari sebuah lokasi awal ke sebuah lokasi tujuan dan membandingkan waktu eksekusi algoritme dan jumlah node yang diperiksa untuk masing-masing fungsi heuristik dalam pencarian rute terpendek dengan menggunakan algoritme A*.

Batasan-batasan yang terdapat dalam penelitian ini antara lain peta yang digunakan adalah peta IPB yang ditampilkan dalam bentuk *grid*, menggunakan algoritme A* sebagai algoritme untuk mencari rute terpendek dengan menggunakan tiga fungsi heuristik yaitu Manhattan, Euclidian dan Euclidian kuadrat, informasi yang disampaikan meliputi waktu eksekusi program, rute yang diperiksa untuk menemukan rute terpendek, dan rute yang dijadikan rute terpendek untuk mencapai tujuan, dan pembobotan jarak antar *node* atau lokasi belum menggunakan bobot yang sebenarnya karena ketidaktersediaan data bobot jarak yang sebenarnya.

Manfaat yang dapat diperoleh dari penelitian ini antara lain memberikan kemudahan dalam pencarian rute terpendek

untuk mencapai lokasi tujuan dan memberikan perbandingan hasil rute terpendek yang diperoleh dengan menggunakan tiga fungsi heuristik dalam algoritme A*.

METODOLOGI

Salah satu masalah yang ditemukan dalam mengembangkan aplikasi untuk peralatan *mobile* adalah batasan-batasan yang ditentukan oleh setiap *platform* pengembangan yang digunakan terutama adalah batasan mengenai spesifikasi perangkat keras yang digunakan (Leggett *et. al* 2006). Berikut ini adalah batasan dan fisik dari perangkat keras, serta aspek sosial dari *mobile devices* :

- Kecepatan CPU dan jumlah RAM yang digunakan

  Kecepatan CPU yang digunakan pada telepon seluler saat ini berkisar antara 300 MHz – 500 MHz. PDA menawarkan kecepatan yang lebih cepat yaitu 624 MHz, namun hanya pada tipe tertentu saja. Kecepatan CPU ini masih kalah jauh bila dibandingka dengan komputer (PC), tetapi masih cukup cepat untuk digunakan dalam membuka aplikasi *office* maupun bermain *game*. Ketika menggunakan *platform* Flash dalam mengembangkan aplikasi *mobile*, maka masalah utama terdapat pada jumlah RAM. Flash Lite 1.1 untuk sistem operasi Symbian membutuhkan RAM sebesar 750 KB pada saat proses *startup*. Aplikasi yang dihasilkan juga membuthkan RAM yang cukup besar untuk dapat dijalankan yaitu berkisar 300 KB untuk aplikasi yang berisi *movie clip*, suara, gambar dan kode program Actionscript.

- Ukuran dan resolusi layar

  Ukuran layar mungkin adalah batasan yang paling nyata atau terlihat yang ditemukan pada pada peralatan mobile, khususnya telepon seluler. Resolusi layar juga mempunyai peranan penting dalam aspek *usability*. Resolusi yang tinggi akan memberikan ruang yang lebih untuk menampilkan isi atau informasi yang dibutuhkan. Tabel 1 menunjukkan beberapa ukuran layar yang umum digunakan pada telepon seluler berdasarkan sistem operasi.

Tabel 1. Ukuran layar telepon seluler berdasarkan sistem operasi

Sistem Operasi	Ukuran layar (pixel)
Symbian Series 40	128 x 128
Symbian Series 60	176 x 208
Symbian Series 80	640 x 200
Symbian Series 90	320 x 240, 640 x 320
Symbian UIQ	320 x 208
Windows Mobile	320 x 240 – 640 x 480
PalmOS	160 x 160, 240 x 240, 320 x 320
LinuxOS	240 x 320

- Kedalaman warna layar

  Flash Lite 1.1 yang merupakan versi pertama dari Flash Lite hanya dapat digunakan pada layar telepon seluler yang memiliki kedalaman warna minimal sebesar 65536 warna. Untuk Flash Lite versi 2.0 sampai 3.0 sudah kompatibel dengan layar yang memiliki kedalaman 16 juta warna. Dengan semakin meningkatnya kedalaman warna yang digunakan pada peralatan *mobile*, khususnya telepon seluler, maka akan semakin banyak informasi yang dapat disampaikan dari hanya berupa huruf hingga animasi.

Algoritme A*

A* (dibaca "A bintang" atau "A star") adalah algoritme pencarian graf/pohon yang mencari jalur dari satu titik awal ke sebuah titik akhir yang telah ditentukan. Algoritme A* menggunakan pendekatan heuristik *h(x)* yang memberikan peringkat ke tiap-tiap titik *x* dengan cara memperkirakan rute terbaik yang dapat dilalui dari titik tersebut. Setelah itu tiap-tiap titik *x* tersebut diperiksa satu-persatu berdasarkan urutan yang dibuat dengan pendekatan heuristik tersebut. Maka dari itulah algoritme A* adalah contoh dari *best-first search*.

Algoritme ini pertama kali ditemukan pada tahun 1968 oleh Peter Hart, Nils Nilsson dan Bertram Raphael. Dalam tulisan mereka, algoritme ini dinamakan algoritme A. Penggunaan algoritme ini dengan fungsi heuristik yang tepat dapat memberikan hasil yang optimal, maka algoritme inipun disebut A*.

Sebenarnya, *depth-first search* (DFS) dan *breadth-first search* (BFS) adalah dua kasus khusus dari algoritme A*. Algoritme Dijkstra yang merupakan salah satu *best-first search*, adalah kasus khusus dari A* dimana $h(x) = 0$ untuk semua nilai *x*.

Perbedaan cara kerja A* dengan *best-first search* adalah selain memperhitungkan *cost* dari *current state* ke tujuan dengan fungsi heuristik (seperti *best-first search*), algoritme ini juga mempertimbangkan *cost* yang telah ditempuh selama ini dari *initial state* ke *current state*. Jadi bila jalan yang telah ditempuh sudah terlalu panjang dan ada jalan lain yang lebih kecil *cost*-nya namun memberikan posisi yang sama dilihat dari *goal*, jalan baru yang lebih pendek itulah yang akan dipilih. Notasi yang dipakai oleh algoritme A* ini:

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

Dalam notasi standar yang dipakai untuk algoritme A* di atas, digunakan $g(n)$ untuk mewakili *cost* rute dari *node* awal ke *node* n. Lalu $h(n)$ mewakili perkiraan *cost* dari *node* n ke *node* tujuan yang dihitung dengan fungsi heuristik. A* 'menyeimbangkan' kedua nilai ini dalam mencari jalan dari *node* awal ke *node* tujuan. Dalam menentukan *node* yang akan dikembangkan, algoritme ini akan memilih *node* dengan nilai $f(n) = g(n) + h(n)$ yang paling kecil (Adipranata *et. al* 2007).

Kompleksitas waktu dari A* sangat bergantung dari heuristik yang digunakannya. Pada kasus terburuk, jumlah titik yang diperiksa berjumlah eksponensial terhadap panjang solusi (jalur terpendek), tetapi A* akan memiliki kompleksitas waktu polinomial apabila fungsi memenuhi kondisi berikut:

$$|h(x) - h^*(x)| \leq O(\log h^*(x)$$

Dimana *h** adalah heuristik optimal, yaitu ongkos sebenarnya dari *node* awal ke tujuan. Dengan kata lain, galat dari *h* tidak akan melaju lebih cepat dari logaritma dari heuristik optimal *h** yang memberikan jarak sebenarnya dari *x* ke tujuan (Russel & Norvig 2003).

Penggunaan memori dari A* bahkan lebih bermasalah dari kompleksitas waktunya. Beberapa varian dari A* telah dikembangkan untuk mengatasi masalah ini, antara lain *Iterative-Deepening-A*, Memory-Bounded A** (MA*), *Simplified-Memory-Bounded-A** (SMA*) dan *Recursive-Best-First-Search* (RBFS). RBFS juga akan memberikan hasil yang optimal apabila heuristiknya dapat diterima.

FUNGSI HEURISTIK

Algoritme A* adalah algoritme pencarian yang menggunakan fungsi heuristik untuk 'menuntun' pencarian rute, khususnya dalam hal pengembangan dan pemeriksaan *node-node* pada peta (Adipranata *et. al* 2007). Fungsi-fungsi heuristik yang biasa digunakan dalam algoritme A*, antara lain :

- Manhattan

  Fungsi ini merupakan fungsi heuristik yang paling umum digunakan. Fungsi ini hanya akan menjumlahkan selisih nilai *x* dan y dari dua buah titik. Fungsi heuristik ini dinamakan Manhattan karena di kota Manhattan di Amerika, jarak dari dua lokasi umumnya dihitung dari blok-blok yang harus dilalui saja dan tentunya tidak bisa dilintasi secara diagonal. Perhitungannya dapat ditulis sebagai berikut :

  $h(n)= abs(tujuan.x-n.x)+abs(tujuan.y -n.y)$

  Dimana *h(n)* merupakan perkiraan *cost* dari *node* n ke *node* tujuan yang dihitung dengan fungsi heuristik. Variabel *n.x* merupakan koordinat *x* dari *node n*, sedangkan *n.y* merupakan koordinat y dari *node n*. Variabel *tujuan.x* merupakan koordinat *x* dari *node* tujuan dan *tujuan.y* merupakan koordinat *y* dari *node* tujuan. Nilai dari *h(n)* akan selalu bernilai positif.

- Euclidian

  Heuristik ini akan menghitung jarak berdasarkan panjang garis yang dapat ditarik dari dua buah titik. Perhitungannya adalah sebagai berikut :

  $h(n)= \sqrt{((tujuan.x -n.x)^2+(tujuan.y -n.y)^2)}$

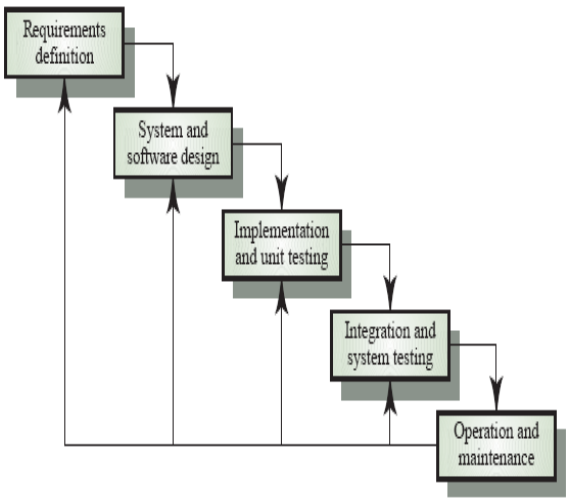
  Dalam kasus ini, skala relatif nilai g mungkin akan tidak sesuai lagi dengan nilai fungsi heuristik *h*. Karena jarak Euclidian selalu lebih pendek dari jarak Manhattan, maka dapat dipastikan selalu akan didapatkan jalur terpendek, walaupun secara komputasi lebih berat.

- Euclidian kuadrat

  Dalam beberapa literatur juga disebutkan jika nilai g adalah 0, maka lebih baik jika ongkos komputasi operasi pengakaran pada heuristik jarak Euclidian dihilangkan saja, menghasilkan rumus sebagai berikut:

  $h(n) = (tujuan.x -n.x)^2 + (tujuan.y -n.y)^2$

Perancangan dalam pembuatan sistem ini dikembangkan dengan *Waterfall Model* (Sommerville 2001) dengan tahap seperti pada Gambar 1.



Gambar 1. *Waterfall Model* (Sommerville 2001).

5.1. Analisis dan definisi kebutuhan

Tahap analisis dan definisi kebutuhan terdiri atas identifikasi kebutuhan data dan kebutuhan fungsional.

5.1.1. Kebutuhan Data

Pada penelitian ini dibutuhkan data berupa peta. Peta yang digunakan adalah peta kampus IPB Dramaga. Peta tersebut diubah menjadi bentuk *grid*. Bentuk *grid* peta ini diperoleh dengan membuat matrik 15 x 12 yang berisi angka 0 dan 1. Ukuran matrik tersebut dipilih karena ukuran layar yang digunakan yaitu 240 x 320 dan ukuran dari *tile* yang digunakan sebagai *grid* yaitu 20 x 20, sehingga posisi *x* dan *y* dari *grid* adalah 0 dan 10. Peta tersebut dibuat secara manual dengan memasukkan sembarang nilai 0 dan 1 ke dalam matrik tersebut.

5.1.2. Kebutuhan Fungsional

Perancangan aplikasi ini harus dapat memberikan informasi mengenai rute terpendek yang dinyatakan dengan banyaknya *node* yang harus diperiksa dan waktu eksekusi dari masing-masing fungsi heuristik algoritme A* serta banyaknya *node* yang dijadikan rute terpendek.

5.2. Perancangan

Tahap perancangan terdiri atas beberapa bagian yaitu perancangan pemodelan data peta, perancangan algoritme A* dan fungsi-fungsi heuristik, serta perancangan antarmuka.

5.2.1 Perancangan Pemodelan Data Peta

Perancangan pemodelan data peta terdiri atas beberapa tahapan, yaitu :

1 Pembentukan matrik yang digunakan sebagai peta.
2 Pembuatan tampilan peta pada layar telepon seluler.
3 Pengubahan data matrik menjadi *grid* peta.

5.2.2. Perancangan *Pseudo-code* Algoritme A* dan Fungsi-Fungsi Heuristik

Perancangan algoritme A* dan fungsi-fungsi heuristik menerapkan kebutuhan fungsional pada tahap analisis dan definisi kebutuhan. Perancangan ini terdiri atas beberapa tahap, yaitu :

1 Penentuan bobot jarak antar *node*.
2 Penentuan *node* awal dan akhir untuk pencarian rute.
3 Pemeriksaan terhadap *node* awal dan akhir.
4 Perancangan *pseudo-code* algoritme A*.
5 Pencarian rute terpendek dengan menggunakan fungsi heuristik yang berbeda.
6 Pembuatan animasi rute terpendek.
7 Perhitungan waktu eksekusi algoritme, banyaknya jumlah *node* yang diperiksa, dan banyaknya *node* yang dijadikan rute terpendek.

5.3. Implementasi

Tahap implementasi, hasil pemodelan data peta dan perancangan fungsi-fungsi heuristik algoritme A* dari tahap perancangan dibuat dengan kode program menggunakan Adobe Flash CS3 Professional dan bahasa pemrograman actionscript 2.0.

5.3.1 Lingkungan Implementasi

Lingkungan implementasi yang digunakan sebagai berikut :

1 Perangkat lunak : Microsoft® Windows XP SP 2, Adobe® Flash® CS3 Professional, Adobe® Device Central, Nokia PC Suite 6.85.
2 Perangkat keras komputer : *Processor* Intel® Core™ Duo T2300E, RAM 2 GB, VGA Intel® GMA 950, *harddisk* 60 GB, *keyboard*, *mouse*, dan monitor LCD.
3 Perangkat lunak *handphone* : Symbian OS 9.2, Adobe® Flash Lite 2.0.
4 Perangkat keras *handphone* : Nokia 6120 classic, processor ARM 11 369 MHz, 64 MB SDRAM/128 MB ROM.

5.3.2 Pembentukan matrik yang digunakan sebagai peta

Peta kampus IPB Dramaga diperoleh dari direktorat fasilitas dan properti IPB. Dari peta tersebut kemudian dibuat matrik petanya. Matrik yang digunakan sebagai peta adalah matrik dua dimensi dengan ukuran 15 x 12. Matrik tersebut memiliki tinggi sebesar 15 dan lebar sebesar 12. Matrik tersebut terdiri dari dua bilangan, yaitu 0 dan 1. Bilangan 0 menandakan bahwa rute bisa dilewati (*walkable*), sedangkan bilangan 1 menandakan rute tidak bisa dilewati (*unwalkable*). Digunakannya nilai 0 dan 1 adalah untuk memudahkan dalam pembuatan peta, sebenarnya nilai 0 dan 1 tersebut bisa diganti dengan nilai lain. Nilai 5, 6, 7, 8, 9, dan 10 adalah representasi dari properti yang dimiliki oleh peta, misalnya rektorat dilambangkan dengan nilai 5. Bentuk dari matrik tersebut adalah sebagai berikut :

```
myMap =[[1,1,1,0,1,1,0,1,1,1,1,1],
        [1,1,1,0,7,1,0,1,1,1,1,1],
        [0,0,0,0,0,0,0,0,0,0,1,1],
        [0,1,1,1,1,0,1,0,1,0,1,1],
        [0,1,1,1,1,0,1,0,1,0,1,1],
        [0,1,1,1,1,0,0,0,1,0,0,0],
        [0,6,1,1,1,0,1,1,1,0,1,1],
        [0,1,10,10,1,0,1,1,6,0,1,1],
        [0,1,10,10,9,0,1,1,1,0,0,0],
        [0,1,10,10,1,0,1,1,1,0,1,0],
        [0,1,1,1,10,0,0,0,0,0,1,0],
        [0,1,1,5,1,1,1,1,1,0,8,0],
        [0,0,0,0,0,0,0,0,0,0,0,0],
        [1,1,1,1,1,0,1,1,1,1,1,1],
        [1,1,1,1,1,0,1,1,1,1,1,1]];
```

5.3.3 Pembuatan tampilan peta pada layar telepon seluler

Telepon seluler yang digunakan dalam pengembangan aplikasi ini adalah nokia 6120 classic. Telepon seluler tersebut memiliki layar dengan ukuran 240 x 320. Proses untuk memposisikan peta pada layar telepon seluler dapat dilihat pada kode program berikut :

```
var mapWidth = 12;
var mapHeight = 15;
var tileW = 20;
var tileH = 20;
wholemap._x =       (240 - mapWidth * tileW)
                    / 2;
wholemap._y =       320 - mapHeight * tileH)
                    / 2;
```

Variabel mapWidth menandakan lebar dari peta yang akan ditampilkan, sedangkan mapHeight menandakan tinggi dari peta. Untuk variabel tileW dan tileH menandakan lebar dan tinggi dari *tile* yang digunakan untuk menandakan *grid*. wholemap._x dan wholemap._y menandakan posisi x dan y dari peta.

5.3.4 Pengubahan data matrik menjadi *grid* peta

Data matrik yang telah dibuat tadi, kemudian akan diubah menjadi *grid* peta. Proses pengubahan data matrik menjadi *grid* peta dapat dilihat pada potongan kode program berikut :

```
wholemap.attachMovie("tile","t_"+i+
"_"+j,i*mapWidth+j);
wholemap["t_"+i+"_"+j]._x=j* tileW;
wholemap["t_"+i+"_"+j]._y=i* tileH;
wholemap["t_"+i+"_"+j].gotoAndStop(myMap[i][
j] + 1);
```

Dimana wholemap merupakan simbol *movie clip* yang akan mengambil tile dan diberi nama lain atau alias t_i_j dengan kedalaman i*mapWidth+j. Posisi x dari setiap t_i_j dalam layar adalah j*tileW dan posisi y dari setiap t_i_j adalah i*tileH. Untuk dapat membedakan antara rute yang bisa dilewati dengan rute yang tidak bisa dilewati maka digunakan fungsi yang akan memerikasa setiap baris dan kolom dalam matrik peta yaitu dengan gotoAndStop(myMap[i][j] + 1).

5.3.5 Penentuan bobot jarak antar *node* dan nilai perpindahan dari satu *node* ke *node* yang lain

Bobot jarak yang digunakan dalam penelitian ini adalah 1. Hal ini dikarenakan jarak sebenarnya dari satu *node* ke *node* yang lain belum pernah diukur secara nyata, misalnya jarak dari rektorat ke graha widya wisuda (GWW).

Penentuan bobot jarak antar *node* dapat dilihat pada kode program berikut ini :

```
var dirA = new Array();
dirA[0] = new Array(1, 0, 1);
dirA[1] = new Array(0, 1, 1);
dirA[2] = new Array(-1, 0, 1);
dirA[3] = new Array(0, -1, 1);
```

5.3.6 Penentuan *node* awal dan akhir untuk pencarian rute

Proses penentuan *node* awal dan akhir dapat dilihat pada potongan kode program berikut :

```
fapetAwal_btn.onPress = function()
{
   xAwal = 0;
   yAwal = 6;
   _parent.awal = "Fapet";
}
rektoratAkhir_btn.onPress = function()
{
   xAkhir = 3;
   yAkhir = 12;
   _parent.tujuan = "Rektorat";
}
```

Misalkan akan dicari rute dari fakultas peternakan menuju rektorat, maka pertama akan di*set* *node* awal untuk fakultas peternakan terletak di koordinaat $x = 0$ dan $y = 6$. Untuk *node* tujuan yaitu rektorat terletak di koordinat $x = 3$ dan $y = 12$.

5.3.7 Pemeriksaan terhadap *node* awal dan akhir

Pemeriksaan terhadap *node* awal dan akhir dilakukan untuk mengetahui apakah *node* awal dan akhir yang sudah dipilih adalah *node* yang bisa dilewati (*walkable*) atau tidak bisa dilewati (*unwalkable*). Proses pemeriksaan *node* awal dapat dilihat pada potongan kode program berikut :

```
if (myMap[i][j] == 0)
{
   findA[j][i] = 0;
   trace("Walk" + j + "_" + i);
}
else
{
   findA[j][i] = "wall";
   trace("Wall" + j + "_" + i);
}
```

Apabila myMap[i][j] == 0 maka rute tersebut bisa dilewati sedangkan apabila myMap[i][j] != 0 maka rute tersebut tidak bisa dilewati. myMap[i][j] akan memeriksa setiap nilai yang terdapat pada matrik peta, apakah bernilai 0 atau 1.

5.3.8 Perancangan *pseudo-code* algoritme A*

Berikut adalah *pseudo-code* dari algoritme A* yang digunakan dalam penenlitian ini :

```
fungsi A*(awal,tujuan)
var himp_tertutup <- himp_kosong
var q <- buat_antrian(awal)
selama (q tidak kosong)
{
   x =  node ada di dalam q dengan nilai f
        terkecil
   if (x = tujuan)
   {
     ditemukan rute terpendek
   }
   hapus x dari q
   tambahkan x ke himp_tetutup
   untuk setiap y adalah node tetangga(x)
   {
     jika y ada dalam himp_tetutup atau y
     adalah unwalkable
     {
        continue
     }
     jika y belum ada pada q
     {
       tambahkan y pada q dan set nilai F,G,H
     }
      jika y terdapat pada q
      {
        periksa nilai G
        jika (Ada y yang memiliki nilai G
        terkecil)
        {
          ganti x dengan y, update nilai
          F,G,H
        }
      }
   }
}
```

5.3.9 Pencarian rute terpendek menggunakan fungsi heuristik berbeda

Pencarian rute terpendek ini menggunakan tiga fungsi heuristik yang berbeda. Pada dasarnya struktur dari algoritme A* yang digunakan pada ketiga fungsi heuristik ini sama, hanya berbeda pada variabel *h(n)* yaitu variabel untuk menyimpan fungsi heuristik. Untuk lebih jelasnya, variabel *h(n)* yang digunakan pada masing-masing fungsi heuristik dapat dilihat di bawah ini :

- Fungsi heuristik Manhattan

```
var hval = Math.abs(ClickA[0]- adjX) +
Math.abs(ClickA[1]- adjY);
```

- Fungsi heuristik Euclidian

```
var hval = Math.sqrt(((ClickA[0]-
adjX)*(ClickA[0] - adjX)) + ((ClickA[1] -
adjY)*(ClickA[1]- adjY)));
```

- Fungsi heuristik Euclidian kuadrat

```
var hval = ((ClickA[0] - adjX)*(ClickA[0] -
adjX))  + ((ClickA[1] - adjY)*(ClickA[1] -
adjY));
```

Dimana ClickA[0] merupakan koordinat *x node* tujuan dan adjX merupakan koordinat *x node* ke-*n*. Sedangkan ClickA[1] merupakan koordinat *y node* tujuan dan adjY merupakan koordinat *y node* ke-*n*.

5.3.10 Pembuatan animasi rute terpendek

Untuk memudahkan dalam pencarian rute terpendek, maka digunakan animasi objek berjalan yang melewati rute terpendek tersebut, sehingga akan memudahkan pengguna dalam mencari rute terpendek yang diinginkan dan menandakan bahwa rute terpendek telah ditemukan.

5.3.11 Perhitungan waktu eksekusi algoritme, banyaknya jumlah *node* yang diperiksa, dan *node* yang dijadikan rute terpendek

Proses perhitungan waktu eksekusi dilakukan setelah *node* awal dan *node* akhir terpilih, kemudian dilakukan proses pencarian rute terpendek. Begitu juga dengan perhitungan jumlah *node* yang diperiksa untuk mendapatkan rute terpendek, dan jumlah *node* yang dilalui dari *node* awal hingga *node* akhir dalam rute terpendek.

5.4. Pengujian

Pengujian dilakukan dengan metode *black-box testing*. Fokus pengujian adalah membandingkan fungsi heuristik pada algortime A* untuk parameter-parameter sebagai berikut :

1 Waktu eksekusi dari fungsi heuristik dalam mencari rute terpendek.
2 Jumlah *node* yang diperiksa untuk mendapatkan rute terpendek.
3 Jumlah *node* yang dijadikan rute terpendek.

5.5. Pemeliharaan

Pada tahap ini dilakukan perbaikan terhadap hasil pengujian jika terdapat kesalahan pada tahap pengujian serta pendokumentasian aplikasi (*user manual*).

HASIL DAN PEMBAHASAN

Setelah pembuatan sistem selesai, maka akan dilakukan proses pengujian untuk memeriksa apakah tujuan dari penelitian ini tercapai atau tidak. Tahap pengujian untuk mencari rute terpendek dibagi menjadi 12 rute antara lain :

1 Rute rektorat – fakultas pertanian.
2 Rute rektorat – fakultas peternakan.
3 Rute rektorat – perpustakaan pusat IPB.
4 Rute fakultas pertanian – rektorat.
5 Rute fakultas pertanian – perpustakaan pusat IPB.
6 Rute fakultas pertanian – fakultas peternakan.
7 Rute fakultas peternakan – rektorat.
8 Rute fakultas peternakan – perpustakaan pusat IPB.
9 Rute fakultas peternakan – fakultas pertanian.
10 Rute perpustakaan pusat IPB – rektorat.
11 Rute perpustakaan pusat IPB – fakultas pertanian.
12 Rute perpustakaan pusat IPB – fakultas peternakan.

Dari 12 rute di atas akan diuji menggunakan tiga fungsi heuristik algoritme A*. Unsur-unsur yang diujikan yaitu waktu eksekusi masing-masing fungsi heuristik (*Time*), banyaknya *node* yang diperiksa untuk menemukan rute terpendek (*Nodes*), dan rute terpendek dari *node* awal ke *node* akhir *(Shortest_P)*. Pengujian dilakukan dengan membagi 12 (dua belas) rute di atas menjadi empat rute utama yaitu pengujian rute Rektorat, pengujian rute Fakultas Pertanian, pengujian rute Perpustakaan Pusat IPB, dan pengujian rute Fakultas Peternakan. Pengujian masing-masing rute tersebut dilakukan dengan menggunakan percobaan sebanyak tiga kali kemudian hasil dari tiga percobaan tersebut dirata-ratakan.

6.1 Pengujian rute Rektorat

Pada pengujian pertama, pencarian rute dilakukan dari rektorat menuju fakultas pertanian, perpustakaan IPB, dan fakultas peternakan. Hasil dari pengujian ini dapat dilihat pada Tabel 2. Dari hasil yang diperoleh pada Tabel 2, *Shortest_P* yang didapat oleh tiga fungsi heuristik yaitu 11 untuk rute rektorat-fakultas pertanian, 14 untuk rute rektorat-perpustakaan IPB, dan 9 untuk rute rektorat-fakultas peternakan.

Jumlah node yang diperiksa dengan menggunakan fungsi heuristik Euclidian kuadrat lebih sedikit dibandingkan dengan fungsi heuristik yang lainnya yaitu 16 utnuk rute rektorat-fakultas pertanian, 21 untuk rute rektorat-perpustakaan IPB, dan 10 untuk rute rektorat-fakultas peternakan. Fungsi heuristik Manhattan menghasilkan rwaktu pencarian rute yang lebih cepat dibandingkan dengan fungsi Euclidian, tetapi fungsi heuristik Euclidian kuadrat menghasilkan waktu pencarian yang lebih cepat dibandingkan fungsi heuristik lainnya yaitu 110.3 ms pada rute rektorat-fakultas pertanian, 160.7 ms pada rute rektorat-perpustakaan pusat IPB, dan 81.7 ms pada rute rektorat-fakultas peternakan.

6.2 Pengujian rute Fakultas Pertanian

Pada pengujian kedua, pencarian rute dilakukan dari fakultas pertanian menuju rektorat, perpustakaan IPB, dan fakultas peternakan. Hasil dari pengujian ini dapat dilihat pada Tabel 3. Dari hasil yang diperoleh pada Tabel 3, *Shortest_P* yang didapat oleh tiga fungsi heuristik memiliki perbedaan yaitu 11 untuk rute fakultas pertanian-rektorat, 9 untuk rute fakultas pertanian-perpustakaan IPB, sedangkan rute fakultas pertanian-fakultas peternakan, besarnya *Shortest_P* pada fungsi heuristik Manhattan dan Euclidian adalah 18, tetapi fungsi heuristik Euclidian kuadrat menghasilkan nilai 24.

Pada rute fakultas pertanian-rektorat, banyaknya node yang diperiksa menggunakan fungsi heuristik Euclidian kuadrat memiliki nilai paling besar yaitu 23, sedangkan pada rute yang lain, fungsi heuristik ini memiliki nilai yang paling kecil yaitu 13 untuk rute fakultas pertanian-perpustakaan IPB dan 36 untuk rute fakultas pertanian-fakultas peternakan.

Tabel 2. Hasil pencarian rute dengan rute awal Rektorat

	Fakultas Pertanian			Perpustakaan Pusat IPB			Fakultas Peternakan		
	Man	Euc	Euc^2	Man	Euc	Euc^2	Man	Euc	Euc^2
Time (ms)	112	131.3	110.3	216.7	235	160.7	83.3	91.7	81.7
Nodes	16	18	16	30	31	21	10	11	10
Shortest_P	11	11	11	14	14	14	9	9	9

Tabel 3. Hasil pencarian rute dengan rute awal Fakultas Pertanian

	Rektorat			Perpustakaan Pusat IPB			Fakultas Peternakan		
	Man	Euc	Euc^2	Man	Euc	Euc^2	Man	Euc	Euc^2
Time (ms)	142.3	164	161.3	120	148.3	95.3	358.7	586.3	290
Nodes	20	22	23	17	19	13	48	59	36
Shortest_P	11	11	11	9	9	9	18	18	24

Fungsi heuristik Euclidian kuadrat memiliki waktu pencarian tercepat pada rute fakultas pertanian-perpustakaan IPB yaitu 95.7 ms dan 290 ms pada rute fakultas pertanian-fakultas peternakan, sedangkan pada rute fakultas pertanian-rektorat, fungsi heuristik Manhattan memiliki waktu pencarian tercepat yaitu 142.3 ms.

6.3 Pengujian rute Perpustakaan Pusat IPB

Pada pengujian ketiga, pencarian rute dilakukan dari perpustakaan pusat IPB menuju rektorat, fakultas pertanian, dan fakultas peternakan. Hasil dari pengujian ini dapat dilihat pada Tabel 4. Dari hasil yang diperoleh pada Tabel 4, *Shortest_P* yang didapat oleh tiga fungsi heuristik yaitu 14 untuk rute perpustakaan pusat IPB-rektorat, 9 untuk rute perpustakaan IPB- fakultas pertanian, dan 15 untuk rute perpustakaan pusat IPB-fakultas peternakan.

Pada rute perpustakaan pusat IPB-fakultas pertanian, banyaknya node yang diperiksa menggunakan fungsi heuristik Manhattan memiliki nilai paling kecil yaitu 18, sedangkan pada rute yang lain, fungsi heuristik Euclidian kuadrat memiliki nilai yang paling kecil yaitu 20 untuk rute perpustakaan IPB-rektorat dan 22 untuk rute perpustakaan pusat IPB-fakultas peternakan.

Fungsi heuristik Euclidian memiliki waktu pencarian terlambat dibandingkan dengan fingsi heuristik lainnya yaitu 199.3 ms, 135.7 ms, dan 236 ms. Untuk rute perpustakaan pusat IPB-rektorat dan perpustakaan pusat IPB-fakultas peternakan, fungsi heuristik Euclidian kuadrat memiliki waktu pencarian tercepat yaitu 149 ms dan 168.3 ms. Sedangkan fungsi heuristik Manhattan memiliki waktu pencarian tercepat pada rute perpustakaan pusat IPB-fakultas pertanian yaitu sebesar 123 ms.

6.4 Pengujian rute Fakultas Peternakan

Pada pengujian keempat, pencarian rute dilakukan dari fakultas peternakan menuju rektorat, perpustakaan IPB, dan fakultas pertanian. Hasil dari pengujian ini dapat dilihat pada Tabel 5. Dari hasil yang diperoleh pada Tabel 5, *Shortest_P* yang didapat oleh tiga fungsi heuristik memiliki perbedaan yaitu 9 untuk rute fakultas peternakan-rektorat, 18 untuk rute fakultas peternakan-fakultas pertanian dengan menggunakan fungsi heuristik mahattan dan Euclidian, sedangkan untuk fungsi heuristik Euclidian kuadrat pada rute yang sama menghasilkan cost sebesar 24. Untuk rute fakultas peternakan-perpustakaan pusat IPB, *Shortest_P* pada fungsi heuristik Manhattan dan Euclidian adalah 15, tetapi fungsi heuristik Euclidian kuadrat menghasilkan nilai 23.

Jumlah node yang diperiksa dengan menggunakan fungsi heuristik Euclidian lebih besar daripada fungsi heuristik yang lain yaitu 11 untuk rute fakultas peternakan-rektorat, 47 untuk rute fakultas peternakan-fakultas pertanian, dan 31 untuk rute fakultas peternakan-perpustakaan pusat IPB. Fungsi heuristik Manhattan memiliki waktu pencarian tercepat pada rute fakultas peternakan – fakultas pertanian dan rute fakultas peternakan-perpustakaan pusat IPB yaitu 219.3 ms dan 282.7 ms. Pada rute fakultas peternakan-perpustakaan pusat IPB, fungsi heuristik Euclidian kuadrat memiliki waktu pencarian terlambat yaitu 245 ms.

Tabel 4. Hasil pencarian rute dengan rute awal Perpustakaan Pusat IPB

	Rektorat			Fakultas Pertanian			Fakultas Peternakan		
	Man	Euc	Euc^2	Man	Euc	Euc^2	Man	Euc	Euc^2
Time (ms)	174.7	199.3	149	123	135.7	131	191.7	236	168.3
Nodes	24	26	20	18	19	19	26	30	22
Shortest_P	14	14	14	9	9	9	15	15	15

Tabel 5. Hasil pencarian rute dengan rute awal Fakultas Peternakan

	Rektorat			Fakultas Pertanian			Perpustakaan Pusat IPB		
	Man	Euc	Euc^2	Man	Euc	Euc^2	Man	Euc	Euc^2
Time (ms)	82.7	92	82	282.7	373.7	305.3	219.3	234.3	245
Nodes	10	11	10	37	47	39	31	31	30
Shortest_P	9	9	9	18	18	24	15	15	23

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Penelitian ini berhasil membangun aplikasi pencarian rute terpendek pada peralatan *mobile* (telepon seluler, PDA). Selanjutnya aplikasi ini dapat digunakan sebagai alternatif untuk mencari rute terpendek ke lokasi yang telah diketahui sebelumnya.

Penggunaan fungsi heuristik pada algoritme A* sangat mempengaruhi dalam pencarian rute terpendek dan waktu eksekusi dari algoritme A* dalam menemukan rute terpendek. Jumlah *node* yang diperiksa menggunakan fungsi heuristik Euclidian cenderung lebih banyak dan waktu pencarian yang dihasilkan juga cenderung lebih lambat bila dibandingkan dengan fungsi heuristik lainnya, walaupun selalu menemukan rute terpendek dalam mencapai tujuan.

Fungsi heuristik Euclidian kuadrat cenderung memiliki waktu pencarian yang lebih cepat dan jumlah *node* yang diperiksa juga lebih sedikit, tetapi belum tentu menghasilkan rute yang paling pendek. Fungsi heuristik Manhattan cenderung memiliki waktu pencarian yang lebih cepat dan jumlah *node* yang diperiksa lebih kecil daripada fungsi heuristik Euclidian. Fungsi heuristik Manhattan juga selalu menemukan rute terpendek untuk mencapai tujuan.

7.2 Saran

Penelitian ini masih dapat dikembangkan lagi pada penelitian berikutnya. Pada penelitian ini hanya menggunakan algoritme A* untuk pencarian rute terpendek dengan menggunakan tiga fungsi heuristik.

Untuk penelitian selanjutnya dapat digunakan algoritma pencarian rute yang lain untuk menemukan rute terpendek dan membandingkan hasilnya dengan penelitian ini ataumenggunakan fungsi heuristik yang lain yaitu diagonal distance dan breaking ties distance. Pada penelitian selanjutnya juga dapat menggunakan bentuk peta asli tanpa dikonversi ke dalam bentuk grid. Penelitian ini juga masih menggunakan bobot yang sama untuk setiap jarak dua node dalam peta, diharapkan pada penelitian selanjutnya dapat digunakan bobot jarak yang sebenarnya antara dua node.

DAFTAR PUSTAKA

Adipranata R, Handojo A, Setiawan H. 2007. Aplikasi Pencari Rute Optimum Pada Peta Guna Meningkatkan Efisiensi Waktu Tempuh Pengguna Jalan Dengan Metode A* dan Best First Search. [laporan penelitian]. Surabaya : Universitas Kristen Petra.

Leggett R, de Boer W, Janousek S. 2006.*Foundation Flash Application for Mobile Devices*. Friendsoft : New York.

Riftadi M.. 2007. Variasi Penggunaan fungsi heuristik dalam pengaplikasian algoritme A*. *Makalah IF2251 Stategi Algoritmik*. Bandung : Institut Teknologi Bandung.

Russell S. J. , Norvig P. 2003. *Artificial Intelligence: A Modern Approach*. New Jersey : Prentice Hall.

Sommerville I. 2001. *Software Engineering*. Ed ke-6. England : Pearson Education.

Weisstein, Eric W. 2007. Grid Graph. http://mathworld.wolfram.com/GridGraph.html. [23 Agustus 2008]