Pencarian Jalur Optimum pada Peta Menggunakan Algoritma A*

Steve Andreas Immanuel - 13517039
Program Studi Teknik Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132, Indonesia
13517039@std.stei.itb.ac.id

Abstrak—Pencarian jalur terpendek pada suatu dari suatu tempat ke tempat lain jika terdapat banyak alternatif jalur adalah suatu persoalan klasik di dunia pemrograman. Pada peta, biasanya selain dicari jalur terpendek, biasanya terdapat batasan-batasan lain seperti waktu tersingkat. Terdapat banyak algoritma yang dapat memecahkan masalah ini dengan kompleksitas yang berbeda-beda. Pada makalah ini, dibahas salah satu algoritma yaitu A* dalam memecahkan masalah ini.

Kata kunci—Jalur terpendek, batasan, A*.

I. PENDAHULUAN

Penggunaan peta digital pada masa sekarang ini adalah hal yang sangatlah biasa. Peta digital dapat menunjukkan jalur yang paling optimum dari suatu tempat ke tempat yang lain. Banyak faktor yang menjadi dasar pemilihan jalur optimum. Contohnya adalah jarak tempuh total dari jalur, waktu total perjalanan jika melewati jalur tersebut, dan lain-lain.

Permasalahan seperti ini serupa dengan persoalan klasik pencarian *shortest path*. *Shortest path* dapat diselesaikan dengan berbagai algoritma. Contoh algoritma paling sederhana untuk memecahkannya adalah *brute force*. Dengan *brute force*, dapat dicoba semua kemungkinan jalur yang mungkin ditempuh dan dipilih jalur yang memiliki ongkos paling kecil. Namun apabila jumlah variasi jalur sudah sangat banyak, maka algoritma *brute force* jelas akan membutuhkan waktu komputasi yang lama dan tidak mangkus. Maka dari itu, digunakanlah algoritma lain untuk menyelesaikan masalah ini. Salah satu algoritma mangkus untuk menyelesaikan masalah ini adalah algoritma A*.

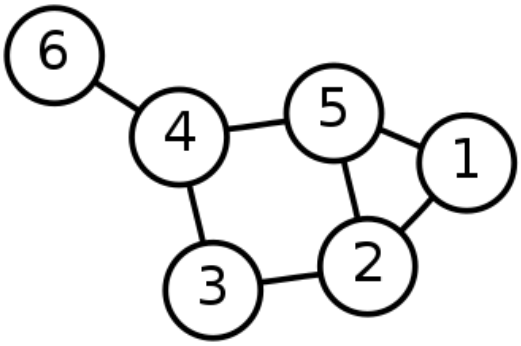
Pada makalah ini, akan dibahas bagaimana algoritma ini dapat menyelesaikan masalah pencarian jalur pada peta yang optimum. Optimum di sini berarti selain merupakan jalur terpendek yang mungkin, juga harus memenuhi berbagai batasan-batasan yang diberikan.

II. DASAR TEORI

Pada permasalahan pencarian jalur optimum ini digunakan struktur data yaitu graf. Untuk perhitungan solusinya digunakan algoritma A* yang merupakan salah satu tipe algoritma *branch and bound*.

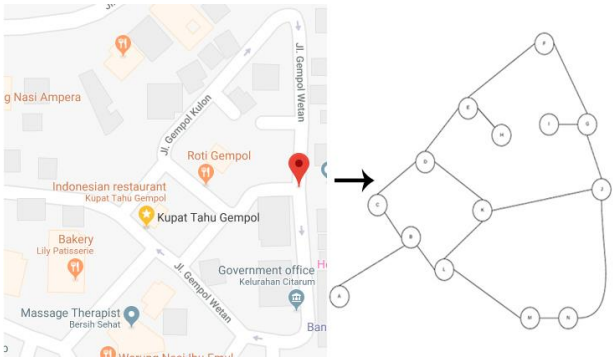
A. Graf

Graf adalah suatu struktur data yang digunakan untuk merepresentasikan objek-objek dan hubungan yang ada antara objek-objek tersebut. Suatu graf terdiri dari simpul-simpul dan sisi-sisi yang menghubungkan antara simpul yang satu dan simpul yang lain.



Gambar 1 Graf dengan 6 Simpul 7 Sisi
Sumber: https://upload.wikimedia.org/wikipedia/commons/5/5b/6n-graf.svg

Berdasarkan orientasi arah pada sisi graf, graf dapat dibedakan menjadi graf berarah dan graf tidak berarah. Graf berarah adalah graf yang sisinya memiliki arah sehingga penelusuran simpul ke simpul yang lain hanya dapat dilakukan jika arahnya sesuai dengan orientasi arah dari sisi graf tersebut. Graf tidak berarah adalah graf yang sisinya tidak memiliki arah sehingga penelusuran simpul ke simpul yang lain dapat dilakukan secara bebas tanpa memperhatikan orientasi arah pada sisi graf.



Gambar 2 Contoh Representasi Peta menjadi Graf
Sumber: *Google Maps*

Graf dapat merepresentasikan tempat-tempat pada peta dengan akurat. Setiap persimpangan pada peta dapat direpresentasikan sebagai simpul, sedangkan setiap jalur yang menghubungkan satu persimpangan ke persimpangan yang lain dapat direpresentasikan sebagai sisi dari graf. Jarak dari suatu persimpangan ke persimpangan yang lain dapat direpresentasikan sebagai bobot dari sisi pada graf. Jalur-jalur searah yang sering ada pada peta dapat direpresentasikan dengan membuat graf memiliki orientasi arah.

Berdasarkan bahasan di atas, struktur data graf sangat cocok untuk merepresentasikan peta dalam membantu memecahkan masalah pencarian jalur optimum. Graf yang dimaksud bertipe graf berarah dan berbobot.

B. *Branch and Bound*

Branch and bound adalah suatu algoritma yang digunakan untuk menyelesaikan masalah-masalah optimasi.

Algoritma *branch and bound* menggunakan teknik yang hampir sama seperti algoritma *backtracking*. Kedua algoritma ini sama-sama membentuk pohon ruang status yang mungkin dari suatu persoalan, kemudian menelusuri simpul yang mengarah ke solusi dan membuang simpul-simpul yang tidak mengarah ke solusi. Perbedaan algoritma branch and bound dengan *backtracking* terdapat pada penentuan simpul mana yang akan ditelusuri berikutnya. Maka pada algoritma *branch and bound* terdapat aturan tertentu untuk menentukan simpul berikutnya yang akan ditelusuri.

Algoritma *branch and bound* memiliki skema umum sebagai berikut:

1. Mulai dari simpul awal pencarian.
2. Bangkitkan semua tetangga dari simpul hidup yang terpilih sekarang dan ubah statusnya menjadi simpul ekspan.
3. Hitung ongkos setiap simpul hidup dengan fungsi pembatas tertentu.
4. Pilih simpul hidup yang memiliki ongkos paling kecil.
5. Jika simpul terpilih adalah simpul hidup, maka solusi sementara ditemukan.
6. Ulangi dari langkah-2 hingga ditemukan solusi. Jika sudah ditemukan solusi, buang simpul-simpul hidup dengan ongkos lebih besar dari simpul solusi sekarang dan ulangi dari langkah-2 hingga tidak ada lagi simpul hidup.

[2]Pada algoritma *branch and bound* pembuangan simpul yang tidak mengarah ke solusi memiliki kriteria sebagai berikut:

- Nilai simpul melebihi nilai terbaik solusi yang sekarang.
- Simpul sudah tidak mungkin ditelusuri karena terdapat batasan yang dilanggar.

Oleh karena itu, algoritma ini sangatlah cocok untuk menyelesaikan masalah pencarian jalur optimum karena terdapat banyak batasan-batasan yang harus dipenuhi.

C. Algoritma Penelusuran Solusi

Masalah pencarian jalur optimum intinya dipecahkan dengan menelusuri peta dalam representasi graf. Dalam penelusuran solusi pada suatu graf, terdapat dua jenis cara penelusuran yaitu *informed search* dan *uninformed search*[2].

Uninformed search adalah cara penelusuran yang di mana tidak ada informasi yang diberikan mengenai masalah yang ada. Penelusuran semacam ini sering disebut juga sebagai *blind search*. Tanpa mengetahui informasi mengenai persoalan yang dilakukan algoritma ini adalah melakukan ekspansi pada simpul-simpul hidup dalam graf dan membandingkan apakah simpul hasil ekspansi sesuai dengan solusi atau tidak. Algoritma ini memang dapat menyelesaikan berbagai masalah, namun untuk tipe permasalahan tertentu (seperti pada kasus ini pencarian jalur optimum), algoritma ini kurang efisien. Contoh dari algoritma ini adalah sebagai berikut:

- *Breadth First Search* (BFS)

Pencarian dengan algoritma *breadth first search* adalah pencarian melebar di mana ekspansi simpul hidup dilakukan secara total pada satu simpul baru berpindah ke simpul yang lain. Hal ini dilakukan berulang-ulang hingga tidak ada lagi simpul yang belum terekspansi atau solusi ditemukan. Secara umum algoritma ini memiliki kompleksitas waktu dan ruang $O(b^d)$ di mana b adalah jumlah cabang maksimum suatu simpul dan d adalah kedalaman simpul solusi.

- *Depth First Seach* (DFS)

  Pencarian dengan algoritma *depth first search* adalah pencarian mendalam di mana ekspansi simpul hidup dilakukan satu kali dan dilanjutkan langsung dengan ekspansi simpul hasil ekspansi yang sebelumnya. Jika simpul hidup terakhir tidak lagi dapat diekspansi, maka dilakukan backtrack ke simpul ekspan terdekat dan dilakukan ekspansi pada simpul itu lagi. Algoritma ini mempunyai kelebihan pada konsumsi memori dibandingkan BFS, namun pembangkitan simpul dapat mencapai kedalaman tak berhingga jika tidak ada pembatasan jumlah kedalaman. Secara umum algoritma ini memiliki kompleksitas waktu $O(b^m)$ dan kompleksitas ruang O(bm) di mana b adalah jumlah cabang maksimum suatu simpul dan m adalah maksimum kedalaman ruang status.

Informed search adalah cara penelusuran di mana terdapat informasi tambahan terhadap suatu permasalahan. *Informed search* sering disebut juga *heuristic search* karena mengandalkan suatu fungsi heuristik tertentu dalam pencarian solusi. Fungsi heuristik adalah fungsi yang mengkalkulasi ongkos dari suatu aksi.

[2]Kriteria fungsi heuristik yang baik adalah:

- Untuk setiap simpul n, di mana h(n) adalah nilai estimasi ongkos dan h*(n) adalah nilai ongkos sebenarnya, maka berlaku $h(n) \leq h^*(n)$.
- Fungsi heuristik tidak akan kelebihan dalam mengestimasi ongkos (optimistik).

Fungsi ini tidak dapat dibuktikan secara langsung namun secara logika benar. Salah satu contoh dari informed search adalah algoritma A*.

D. Algoritma A*

Algoritma A* adalah salah satu jenis algoritma branch and bound yang idenya adalah menghindari ekspansi simpul-simpul yang ongkosnya sudah terlalu tinggi. Hal ini dilakukan dengan suatu fungsi evaluasi sebagai berikut:

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

Keterangan:

- f(n): fungsi evaluasi untuk mengestimasi ongkos total penelusuran dari simpul n ke simpul tujuan.
- g(n): fungsi yang mengembalikan nilai ongkos yang digunakan untuk mencapai simpul n.
- h(n): fungsi heuristik yang digunakan untuk mengestimasi ongkos dari simpul n ke simpul tujuan

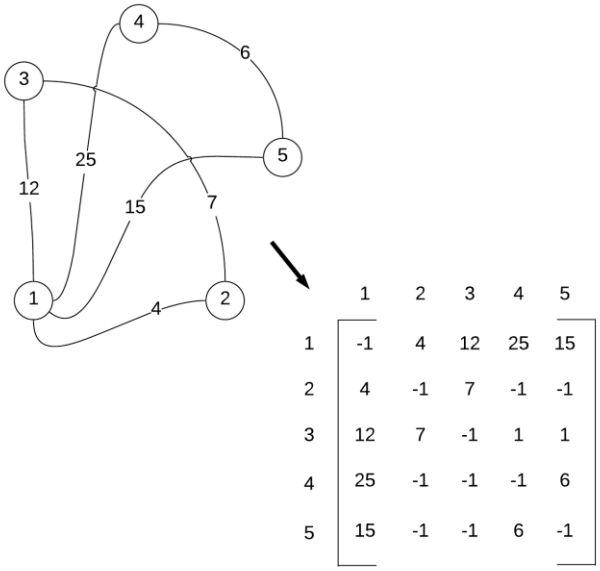
Dengan adanya fungsi evaluasi ini, pembangkitan simpul dapat dilakukan pada simpul hidup yang memiliki ongkos paling kecil sehingga memiliki kemungkinan paling tinggi untuk mendapatkan solusi paling optimum. Dengan demikian pencarian solusi dapat menjadi lebih efisien.

III. PENERAPAN ALGORITMA A*

Seperti yang telah dijelaskan pada bagian sebelumnya, sebuah peta dapat direpresentasikan dengan akurat menggunakan struktur data graf. Maka penerapan algoritma A* ini adalah dengan menelusuri peta dalam representasi graf hingga ditemukan jalur optimum yang memenuhi semua batasan.

A. Translasi Peta ke Graf

Pengubahan peta untuk direpresentasikan menjadi graf dapat dilakukan dengan mudah dengan merepresentasikan graf dalam bentuk matriks ketetanggaan. Setiap elemen ke i,j pada matriks melambangkan bahwa simpul i terhubung ke simpul j dengan nilai bobot sesuai dengan isinya. Jika tidak terdapat sisi dari suatu simpul ke simpul yang lain, pada graf dapat diisikan nilai tertentu yang melambangkan simpul-simpul tersebut tidak terhubung.



	1	2	3	4	5
1	-1	4	12	25	15
2	4	-1	7	-1	-1
3	12	7	-1	1	1
4	25	-1	-1	-1	6
5	15	-1	-1	6	-1

Gambar 3 Representasi Graf dalam Matriks Ketetanggaan
Sumber: Dokumen penulis

B. Cara Kerja Algoritma A*

Yang menjadi garis besar dalam pencarian jalur optimum di makalah ini adalah bagaimana memenuhi batasan-batasan yang ada. Pencarian jalur terpendek pada suatu graf sudahlah biasa, namun dengan ditambahnya banyak batasan tentu saja ini menjadi persoalan yang menarik. Batasan-batasan yang dimaksud contohnya adalah adanya deadline waktu dalam penelusuran, adanya suatu tempat tertentu yang harus dilewati sebelum mencapai tujuan, dan lain-lain.

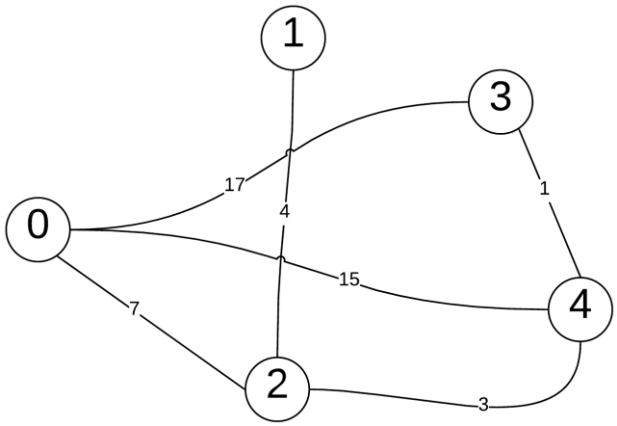
Algoritma A* pada permasalahan ini adalah:

1. Mulai dari simpul awal.
2. Di antara simpul hidup, pilih simpul yang memiliki nilai f(n) yang paling kecil.
3. Jika simpul tersebut adalah simpul solusi, maka solusi sementara ditemukan.
4. Jika simpul tersebut bukan simpul solusi, bangkitkan simpul anak dari simpul hidup tersebut.
5. Untuk semua simpul hidup buang simpul-simpul yang tidak memenuhi batasan-batasan yang ada dan/atau memiliki nilai f(n) yang lebih besar sama dengan nilai f(n) dari simpul solusi sementara jika sudah ditemukan.
6. Ulangi langkah 2-5 hingga tidak ada lagi simpul hidup.

Algoritma di atas hanya akan menelusuri simpul-simpul yang mengarah ke solusi sehingga tentu saja akan lebih mangkus dibandingkan algoritma BFS.

IV. Eksperimen dan Analisis

Pada bagian eksperimen dan analisis ini akan digunakan graf yang ditunjukkan pada gambar di bawah ini sebagai acuan untuk kasus-kasus uji yang digunakan.



Gambar 4 Kasus Uji yang Digunakan pada Eksperimen
Sumber: Dokumen penulis

A. Eksperimen

Pencarian jalur optimum dari suatu simpul menuju simpul yang lain akan dicoba pada beberapa tipe kasus yaitu:

1. Mencari mencari jalur optimum dengan waktu terkecil

   Untuk mencari jalur optimum dengan waktu terkecil, algoritma A* dapat diterapkan dengan menambahkan nilai waktu tersebut ke dalam g(n) pada A*. Maka seharusnya algoritma akan memilih jalur yang memiliki kombinasi jarak dan waktu yang paling kecil. Sedangkan untuk fungsi heuristik dapat diisikan nilai *eucledian distance* yaitu jarak simpul jika ditarik garis lurus secara langsung.

   Pada Gambar 5 tampak bahwa ditemukan jalur optimum dari simpul 0 ke simpul 4 yaitu 0→3→4 dengan jarak total 18.

```
Masukan jumlah simpul:5
Masukan representasi matriks ketetanggaan:
-1 -1 12 23 35
-1 -1 7 -1 -1
12 7 -1 -1 18
23 -1 -1 -1 2
35 -1 18 2 -1
Masukan matriks heuristik:
0 2 5 17 15
2 0 4 15 25
7 4 0 20 15
17 15 25 0 1
15 25 3 1 0
Masukan simpul awal:0
Masukan simpul akhir:4
0 ->3 ->4
```

Gambar 5 Hasil Uji Eksperimen pada Kasus Uji 1
Sumber: Dokumen penulis

2. Mencari jalur optimum dengan batasan simpul

   Untuk mencari jalur optimum dari satu simpul ke simpul yang lain namun harus melewati suatu simpul x, maka dapat diselesaikan dengan 2 tahap. Yaitu mencari jalur optimum dari simpul awal ke simpul x, kemudian dari simpul x ke simpul tujuan yang tidak melewati simpul awal. Dengan demikian akan diperoleh jalur optimum yang diharapkan.

```
C:\Users\Steve\Desktop>python astar.py
Masukan jumlah simpul:5
Masukan representasi matriks ketetanggaan:
-1 -1 7 17 15
-1 -1 4 -1 -1
7 4 -1 -1 3
17 -1 -1 -1 1
15 -1 3 1 -1
Masukan matriks heuristik:
0 2 5 17 15
2 0 4 15 25
7 4 0 20 15
17 15 25 0 1
15 25 3 1 0
Masukan simpul awal:0
Masukan simpul akhir:2
0 ->2
```

```
Masukan jumlah simpul:5
Masukan representasi matriks ketetanggaan:
-1 -1 7 17 15
-1 -1 4 -1 -1
7 4 -1 -1 3
17 -1 -1 -1 1
15 -1 3 1 -1
Masukan matriks heuristik:
0 2 5 17 15
2 0 4 15 25
7 4 0 20 15
17 15 25 0 1
15 25 3 1 0
Masukan simpul awal:2
Masukan simpul akhir:3
2 ->4 ->3
```

Gambar 6 Hasil Uji Eksperimen pada Kasus Uji 2
Sumber: Dokumen penulis

Pada gambar di atas tampak bahwa jalur optimum dari simpul 0 menuju simpul 3 dan harus melewati simpul 2 adalah 0→2→4→3 dengan jarak total 11.

3. Mencari jalur optimum dengan batasan *deadline* waktu

   Untuk mencari jalur optimum dengan batasan deadline waktu dapat dilakukan dengan menambah batasan yang ada pada algoritma A* tersebut. Hal ini dilakukan dengan cara membuang semua simpul hidup yang sudah memiliki ongkos waktu melebihi batas waktu.

Pada Gambar 7 tampak bahwa ditemukan jalur optimum dari simpul 4 menuju 2 dengan batasan waktu tidak boleh melebihi 13 adalah 4→3→0→2 dengan jarak total 25.

```
Masukan jumlah simpul:5
Masukan representasi matriks ketetanggaan:
-1 -1 12 23 35
-1 -1 7 -1 -1
12 7 -1 -1 18
23 -1 -1 -1 2
35 -1 18 2 -1
Masukan matriks heuristik:
0 2 5 17 15
2 0 4 15 25
7 4 0 20 15
17 15 25 0 1
15 25 3 1 0
Masukan simpul awal:4
Masukan simpul akhir:2
Masukan waktu maksimal:13
4 ->3 ->0 ->2
```

Gambar 7 Hasil Uji Eksperimen pada Kasus Uji 3

Sumber: Dokumen penulis

B. Analisis

Dari semua kasus uji yang dicoba pada bagian eksperimen sebelumnya, sebenarnya penulis merepresentasikan pencarian jalur yang sering terjadi pada peta baik cetak maupun digital. Orang-orang biasanya selain ingin mencari jalur terpendek dari satu tempat ke tempat yang lain, mereka juga menginginkan hal-hal yang lain seperti transit di suatu tempat, adanya batasan waktu, dan lain-lain.

Di representasi matriks ketetanggaan, nilai -1 melambangkan bahwa simpul-simpul yang bersangkutan tidak terhubung.

Pada kasus uji yang pertama, digambarkan keadaan bahwa pencarian jalur tidak hanya mencari yang terpendek saja, namun harus didapatkan waktu yang paling singkat juga. Terlihat pada Gambar 5 bahwa matriks ketetanggaan memiliki nilai yang tidak sama seperti yang tertera pada Gambar 4. Hal ini dikarenakan adanya penambahan matriks *M*, di mana

$$M = \begin{bmatrix} 0 & 0 & 5 & 6 & 20 \\ 0 & 0 & 3 & 0 & 0 \\ 5 & 3 & 0 & 0 & 15 \\ 6 & 0 & 0 & 0 & 1 \\ 20 & 0 & 15 & 1 & 0 \end{bmatrix}$$

melambangkan batasan waktu dari simpul ke simpul yang lain seperti matriks ketetanggan yang biasa. Hal ini dilakukan agar algoritma A* memperhitungkan adanya batasan waktu sehingga pembangkitan dan pemilihan simpul juga mempertimbangkan batasan waktu tersebut.

Maka dari itu meskipun terdapat jalur yang lebih pendek daripada jalur optimum yang ditentukan (misalnya jalur 0→4 dengan jarak total 15 atau jalur 0→2→4 dengan jarak total 10) jalur 0→3→4 tetap dipilih sebagai jalur optimum karena memiliki total ongkos yang paling kecil.

Pada kasus uji yang kedua, digambarkan keadaan ketika dalam pencarian suatu jalur harus dilakukan transit pada suatu tempat sebelum menuju tujuannya. Hal ini sering terjadi misalkan pada turis-turis yang sedang berwisata ke tempat asing. Misalkan mereka dari hotel akan menuju bandara namun harus melewati tempat oleh-oleh, maka hal tersebut sama seperti apa yang diujikan pada kasus uji kedua.

Permasalahan seperti ini dapat diselesaikan dengan mudah dengan membagi permasalahan menjadi 2 tahap seperti pada penjelasan di bagian eksperimen. Yang perlu diperhatikan adalah pada pencarian jalur minimum dari simpul x ke simpul tujuan di hasilnya tidak boleh mengandung simpul awal karena hal ini akan menimbulkan *looping* tak berhingga dan tidak ditemukannya solusi. Prinsip yang ditekankan penulis pada bagian ini intinya adalah bahwa jika jarak simpul awal ke simpul x adalah jarak minimum dan jarak dari simpul x ke simpul tujuan adalah simpul minimum, maka jarak dari penggabungan kedua hal tersebut adalah jarak dari simpul awal melewati simpul x menuju simpul tujuan dengan optimum. Hal ini tampak pada hasil pengujian pada Gambar 6.

Pada kasus uji yang terakhir, digambarkan keadaan ketika jalur optimum yang didapatkan harus memiliki total waktu perjalanan yang tidak melewati batas tertentu. Pada implementasi algoritmanya dilakukan pembuangan simpul-simpul hidup yang sudah tidak memenuhi batas waktu lagi.

Pada kasus uji ini dicari jalur dari simpul 4 menuju simpul 2 yang memiliki batas total waktu perjalanan tidak lebih dari 13. Pada Gambar 7 tampak bahwa matriks ketetanggaan memiliki nilai yang berbeda dengan bobot sisi graf pada Gambar 4 karena alasan yang sama seperti pada kasus uji yang pertama. Maka didapatkanlah jalur optimum yaitu 4→3→0→2.

Dari total semua kasus uji yang diberikan pada bagian eksperimen, algoritma A* dapat memberikan jalur optimum sesuai yang keadaan yang diinginkan dengan adanya sedikit perubahan pada fungsi kelayakan dan evaluasi. Algoritma ini jelas lebih mangkus dibandingkan algoritma breadth first search karena tidak mencoba semua kemungkinan jalur yang mungkin dengan mengeleminasi jalur-jalur yang sudah tidak memenuhi batasan-batasan yang diberikan.

V. KESIMPULAN

Penelusuran jalur-jalur pada peta menggunakan algoritma A* dipastikan dapat menghasilkan jalur yang paling optimum. Dengan sedikit modifikasi pada implementasi, algoritma A*

dapat memberikan jalur optimum pada berbagai jenis keadaan seperti yang sudah diujikan.

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis sampaikan kepada Tuhan yang Maha Esa karena berkat dan rahmatnya, penulis dapat menyelesaikan tugas makalah Strategi Algoritma ini. Penulis juga ingin mengucapkan banyak terima kasih pada Dr. Ir. Rinaldi Munir, MT. sebagai dosen penulis karena sudah mengajarkan sangat banyak hal mengenai strategi-strategi algoritma selama satu semester ini. Terakhir, penulis juga berterima kasih kepada semua teman dan keluarga yang membantu penulis menyelesaikan makalah ini.

REFERENCES

[1] Biggs, N.; Lloyd, E.; Wilson, R. (1986), *Graph Theory, 1736–1936*

[2] Rinaldi Munir . Diktat Kuliah IF2211 Strategi Algoritma . 2019

[3] Bollobás, B.Graph Theory: An Introductory Course. New York: Springer-Verlag, 1979

[4] Masoud Nosrati, Ronak Karimi, Hojat Allah . Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches . 2012

[5] https://socs.binus.ac.id/2013/04/23/uninformed-search-dan-informed-search/ [Diakses pada 26 April 2019]

[6] https://techdifferences.com/difference-between-bfs-and-dfs.html [Diakses pada 26 April 2019]

[7] https://www.geeksforgeeks.org/bfs-vs-dfs-binary-tree/ [Diakses pada 26 April 2019]

[8] https://ieeexplore.ieee.org/document/8405474 [Diakses pada 26 April 2019]

[9] https://www.cs.ubc.ca/~kevinlb/teaching/cs322%20-%202006-7/Lectures/lect7.pdf [Diakses pada 26 April 2019]

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, 26 April 2019

Steve Andreas Immanuel
13517039