Perbandingan Algoritma A-Star dan Dijkstra pada Pencarian Jalur Evakuasi Tsunami Terpendek Menuju Shelter di Kabupaten Bantul Berbasis Aplikasi Android

Dwiyan Yogaswara [*1)], Suhartono[*2)]
*Departemen Ilmu Komputer/ Informatika, Fakultas Sains Dan Matematika, Universitas Diponegoro
[1)]dwiyanyogaswara@gmail.com, [2)]suhartono@if.undip.ac.id

Abstrak

Gempa merupakan penyebab utama tsunami di Indonesia. Salah satu daerah yang merupakan daerah rawan gempa bumi dan tsunami adalah Kabupaten Bantul karena berada di zona subduksi lempeng Australia dan Eurasia. Untuk itu manajemen mitigasi pra bencana tsunami sangat diperlukan. Akan tetapi jalur evakuasi sering dilupakan sehingga justru dapat menambah korban. Salah satu sarana penyampaian informasi adalah internet. Dari data Asosiasi Penyedia Jasa Internet Indonesia (APJII) 2017, pengguna internet di Indonesia semakin tinggi, mencapai angka 143,26 juta. Metode pencairan rute yang digunakan untuk membuat aplikasi rute shelter tsunami Bantul adalah algoritma A-Star dan Dijkstra. Penelitian ini bertujuan untuk menghasilkan suatu aplikasi android yang dapat memberikan informasi prosedur evakuasi tsunami, informasi shelter, dan jalur evakuasi tsunami rute terdekat menuju shelter dan menganalisis hasil perbandingan algoritma A-Star dan Dijkstra. Data shelter diperoleh dari Badan Penanggulangan Bencana Daerah (BPBD) Bantul. Penelitian ini berhasil mengembangkan Perbandingan Algoritma A-Star dan Dijkstra pada Pencarian Jalur Evakuasi Tsunami Terpendek menuju Shelter di Kabupaten Bantul berbasis Aplikasi Android. Aplikasi berhasil dikembangkan dengan model Waterfall. Aplikasi dapat menampilkan rute dari posisi pengguna menuju shelter terdekat dengan algoritma A-Star dan Dijkstra. Algoritma A-Star menghasilkan rata-rata lama eksekusi 0.14 detik lebih cepat dan rata-rata jumlah node yang dicek 224 lebih sedikit dibandingkan Algoritma Dijkstra.

Kata kunci *: aplikasi, evakuasi, tsunami, shelter, Bantul, android, Algoritma A-Star, Algoritma Dijkstra.*

Abstract

Earthquake is the main cause of tsunami in Indonesia. One of the areas that are prone to earthquakes and tsunamis is Bantul Regency because it is located in the Australian and Eurasian plate subduction zones. Therefore, tsunami disaster mitigation pra management is very necessary. However, evacuation routes are often forgotten so that they may add to the casualties. One means of delivering information is the internet. From Indonesian Internet Service Provider Association (APJII) 2017 data, internet users in Indonesia are getting higher, reaching 143,26 million. The finding route method used to make the Bantul tsunami shelter route application is A-Star algorithm and Dijkstra algorithm. This study aims to produce an android application that can provide information on tsunami evacuation procedures, shelter information, and tsunami evacuation routes the closest route to the shelter and to analyse the comparasion result between A-Star and Dijkstra algorithms. Shelter data was obtained from the Bantul Regional Disaster Management Agency (BPBD). This research succeeded in developing a Comparison of A-Star and Dijkstra Algorithms on Searching for the Shortest Tsunami Evacuation Path to Shelter in Bantul Regency in Android Application-based. The application successfully developed with the

Waterfall model. The application can display the route from the user's position to the nearest shelter with the A-Star and Dijkstra algorithms. The A-Star algorithm produces an average execution time 0.14 seconds quicker and the average number of nodes 224 is less than the Dijkstra Algorithm.

Keywords *: application, evacuation, tsunami, shelter, Bantul, android, A-Star Algorithm, Dijktra Algorithm.*

1 PENDAHULUAN

Gempa merupakan penyebab utama tsunami di Indonesia (Puspito dan Triyoso, 1994). Salah satu daerah yang merupakan daerah rawan gempa bumi dan tsunami adalah Kabupaten Bantul, yakni salah satu kabupaten di Daerah Istimewa Yogyakarta. Kabupaten Bantul terletak di daerah pesisir pantai selatan Jawa yang berbatasan langsung dengan Samudra Hindia . Samudra Hindia merupakan daerah yang di dalamnya terdapat zona subduksi atau zona pertemuan dua lempeng antara lempeng Australia dan Eurasia sehingga menjadikan Kabupaten Bantul termasuk daerah yang rawan terjadi gempa bumi yang menimbulkan tsunami [1].

Menanggapi risiko bencana seperti di atas, maka manajemen mitigasi pra bencana tsunami sangat diperlukan. Salah satu bentuk upayanya adalah memberikan pengetahuan kepada masyarakat tentang jalur evakuasi tsunami terpendek menuju shelter. Upaya tersebut penting agar masyarakat sudah mengetahui shelter mana yang akan mereka tuju dan jalur evakuasi mana yang akan mereka lewati jika sewaktu-waktu tsunami melanda daerah mereka.

Pencarian jalur evakuasi tsunami terpendek menuju shelter dapat dicari menggunakan algoritma A-Star maupun Dijkstra. Beberapa penelitian tentang pencarian jalur terpendek menggunakan algoritma A-Star telah dilakukan sebelumnya. Salah satu penelitian terkait penggunaan algoritma A-Star antara lain pernah dikaji dalam pencarian jalur evakuasi tsunami terpendek di Kota Padang berbasis hybrid application [2]. Penelitian terkait lainnya yang menggunakan algoritma A-Star adalah penelitian yang mengembangkan suatu aplikasi untuk menentukan rute terpendek menuju kantor pemerintah Kota Bontang berbasis web [3]. Sementara itu, pencarian jalur terpendek juga dapat digunakan alternatif metode lain yakni algoritma Dijkstra antara lain dalam penentuan jalur terpendek untuk pemadam kebakaran [4]. Untuk itu perlu diperbandingkan algoritma A-Star dan Dijkstra pada pencarian jalur evakuasi tsunami terpendek menuju shelter di Kabupaten Bantul berbasis aplikasi android.

2 TINJAUAN PUSTAKA

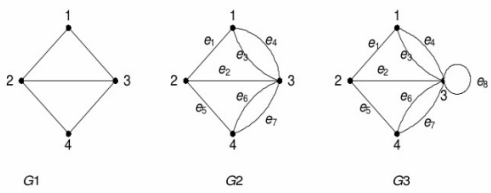
2.1 TEORI DASAR GRAF

Graf pada dasarnya terdiri dari dua komponen, yaitu titik/node (*vertex* atau V) dan busur (*edge* atau E) [5]. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis [6], yaitu graf tak berarah dan graf berarah.

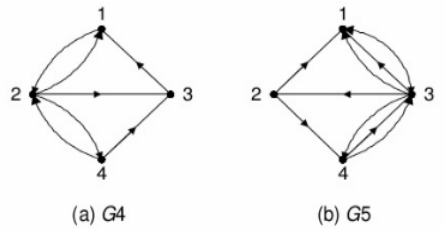
Graf tak berarah adalah graf yang sisinya tidak mempunyai orientasi arah [6]. Pada graf tak berarah, dua buah sisi (u, v) dan (v, u) adalah sisi yang sama, atau dengan kata lain (u, v) = (v, u). Contoh graf tak berarah disajikan pada Gambar 1. Pada gambar tersebut terlihat bahwa G1, G2 dan G3

merupakan graf tak berarah karena setiap sisinya tidak mempunyai orientasi arahikan ulang.

Graf berarah adalah graf yang setiap sisinya diberikan orientasi arah [6]. Pada graf berarah, dia buah sisi (u, v) dan (v, u) adalah sisi yang berbeda, dengan kata lain (u, v) ≠ (v, u). Pada penelitian ini, graf yang digunakan untuk merepresentasikan jaringan jalan yang terdapat di Kabupaten Bantul adalah graf berarah. Contoh graf berarah disajikan pada Gambar 2. Pada gambar tersebut terlihat bahwa G4 dan G5 merupakan graf berarah karena setiap sisinya mempunyai orientasi arah



Gambar 1 Contoh Graf Tak-Berarah [6]



Gambar 2 Contoh Graf Berarah [6]

2.2 HAVERSINE FORMULA

Metode Haversine Formula digunakan untuk menghitung jarak antara titik di permukaan bumi menggunakan garis longitude/lintang dan garis latitude/bujur sebagai variabel inputan [7]. Haversine formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bumi berdasarkan latitude dan longitude. Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R = 6.367, 45 km, dan lokasi dari 2 titik di koordinat bumi (latitude dan longitude) masing-masing adalah longitude1, latitude1, dan longitude2, latitude2 Adapun rumus Haversine Formula dapat dilihat pada persamaan 1.

$$d = \sqrt{x^2 + y^2} * R \text{..... (1)}$$

$$x = (longitude2 - longitude1) * \cos(\frac{(latitude1+latitude2)}{2})$$

$$y = (latitude2 - latitude1)$$

Keterangan:
x : Longitude (Lintang)
y : Lattitude (Bujur)
d : Jarak
R : Radius Bumi = 6371 km
1 derajat : 0.0174532925 radian

2.3 ALGORITMA A-STAR

Algoritma A-Star atau yang ditulis juga A* diciptakan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968. Algoritma A-Star adalah algoritma pencarian graf/pohon yang mencari jalur dari satu node awal ke sebuah node goal yang telah ditentukan [8]. Algoritma A-Star memberikan biaya estimasi terendah f(n) pada setiap node dan mengevaluasi node-node tersebut berdasarkan nilai f(n) dari node. Semakin kecil nilai f(n) dari node maka semakin besar prioritas node untuk dipilih menjadi jalur terpendek menuju node goal. Biaya estimasi terendah f(n) dari node tersebut diperoleh dengan menggabungkan g(n), yaitu cost untuk mencapai node, dan h(n), yaitu cost yang diperlukan dari node untuk mencapai node goal, sehingga rumus untuk mencari nilai f(n) dapat dilihat pada persamaan 2.

$$f(n) = g(n) + h(n) \text{..... (2)}$$

Keterangan:
f(n) : biaya estimasi terendah
g(n) : biaya dari *node* awal ke *node* n
h(n) : biaya dari *node* n ke *node goal*

Nilai g(n) didapatkan dengan menghitung jarak jalan dari *node start* ke *node* n. Nilai h(n) didapatkan dengan menghitung jarak udara dari *node* n ke *node goal*. Penghitungan jarak antar *node* menggunakan rumus "Haversine Formula". *Pseudocode* dari Algoritma A-Star ditunjukkan pada Gambar 3.

```
1. Put node_start in the OPEN list with f(node_start) =
   h(node_start)
2. While the OPEN list is not empty{
3.    Take from the open list the node node_current with the
   lowest
4.          f(node_current)= g(node_current) + h(node_current)
5.    If node_current is node_goal we have found the solution;
   break
6.    Generate each state node_successor that come after
   node_current
7.    For each node_successor of node_current{
8.        Set successor_current_cost= g(node_current) +
          w(node_current, node_successor)
9.        If node_successor is in the OPEN list{
10.             If g(node_successor) ≤ successor_current_cost
            (to line 20)
11.       } else if node_successor is in the CLOSED list{
12.             If g(node_successor) ≤ successor_current_cost
            (to line 20)
13.         Move node_successor from the CLOSED list to the
            OPEN list
14.       } else {
15.   Add node_successor to the OPEN list
16.             Set h(node_successor) to be the heuristic
            distance to node_goal
17.       }
18.       Set g(node_successor) = successor_current_cost
19.       Set the parent of node_successor to node_current
20.}
21.Add node_current to the CLOSED list
22. }
23. If (node_current != node_goal) exit with error
```

Gambar 3 Kode Sumber /*Pseudocode* Algoritma A-Star [9]

2.4 ALGORITMA DIJKSTRA

Algoritma Dijkstra adalah sebuah algoritma yang dikembangkan oleh seorang ilmuwan komputer dari Belanda, Edsger Dijkstra. Algoritma Dijkstra adalah sebuah algoritma yang menyelesaikan pencarian jalur terpendek pada graf dengan nilai non negatif untuk bobot setiap simpul, menghasilkan pohon jalur terpendek [10]. Algoritma ini akan mencari jalur dengan cost minimum antara *node* tersebut dengan *node* lainnya. Algoritma Dijkstra memberikan cost untuk mencapai *node* g(n) pada setiap *node* dan mengevaluasi *node-node* tersebut berdasarkan nilai g(n) dari *node*. Semakin kecil nilai g(n) dari *node* maka semakin besar prioritas *node* untuk dipilih menjadi jalur terpendek menuju *node goal*. Algoritma ini dapat digunakan untuk mencari total biaya(*cost*) dari lintasan terpendek yang dibentuk dari sebuah *node* ke sebuah *node* tujuan. *Pseudocode* Algoritma A-Star ditunjukkan oleh Gambar 4.

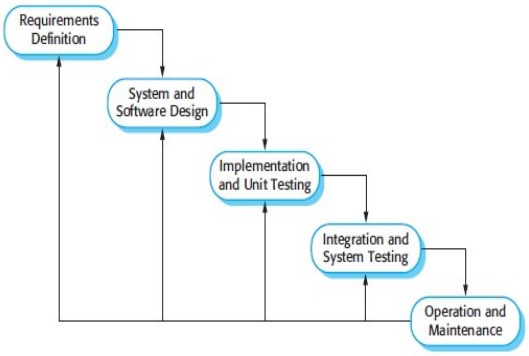
```
1. Put node_start in the OPEN list with f(node_start) =
   h(node_start)
2. While the OPEN list is not empty{
3.    Take from the open list the node node_current with the
   lowest g(node_current)
4.    If node_current is node_goal we have found the solution;
   break
5.    Generate each state node_successor that come after
   node_current
6.    For each node_successor of node_current{
7.        Set successor_current_cost= g(node_current) +
          w(node_current, node_successor)
8.        If node_successor is in the OPEN list{
9.              If g(node_successor) ≤ successor_current_cost
            (to line 20)
10.       } else if node_successor is in the CLOSED list{
11.             If g(node_successor) ≤ successor_current_cost
            (to line 20)
12.         Move node_successor from the CLOSED list to the
            OPEN list
13.       } else {
14.   Add node_successor to the OPEN list
15.       }
16.       Set g(node_successor) = successor_current_cost
17.       Set the parent of node_successor to node_current
18.}
19.Add node_current to the CLOSED list
20. }
21. If (node_current != node_goal) exit with error
```

Gambar 4 Kode Sumber/ Pseudocode Algoritma Dijkstra [11]

2.5 MODEL PENGEMBANGAN PERANGKAT LUNAK

Model pengembangan perangkat lunak yang digunakan pada penelitian ini adalah menggunakan pengembangan model Waterfall. Model ini melakukan pendekatan secara sistematis dan urut mulai dari tahap awal hingga akhir. Adapun model Waterfall menurut [12], model Waterfall memiliki tahapan utama dari model waterfall yang mencerminkan aktifitas pengembangan dasar. Terdapat 5 (lima) tahapan pada model Waterfall, yaitu *requirement analysis and definition, system and software design, implementation and unit testing, integration and system testing, dan operation and maintenance.* Secara umum tahapan pada

Model Waterfall dapat dilihat pada Gambar 5.



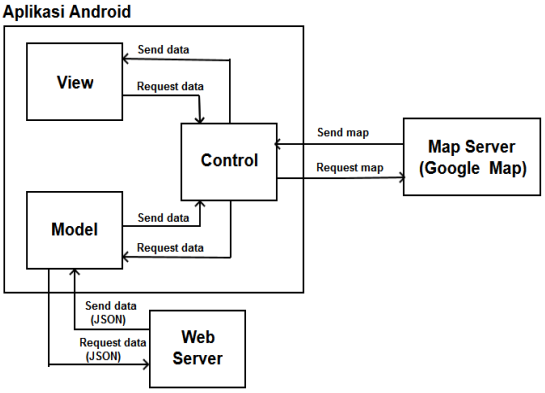
Gambar 5 Model Waterfall [12]

3 METODE PENELITIAN

3.1 GAMBARAN UMUM PERANGKAT LUNAK

Aplikasi Rute *Shelter* Tsunami Bantul adalah aplikasi berbasis android yang dapat memberikan informasi tentang prosedur evakuasi tsunami, informasi *shelter*, dan rute terdekat menuju *shelter*. Aplikasi ini mempunyai cakupan wilayah di Kabupaten Bantul, khususnya daerah yang dekat dengan pesisir pantai yang merupakan daerah yang mempunyai risiko terkena dampak tsunami. Diharapkan dengan adanya aplikasi ini dapat membantu pihak BPBD Kabupaten Bantul dalam rangka mitigasi tsunami kepada masyarakat Kabupaten Bantul sehingga dapat meminimalisir jumlah korban jika sewaktu-waktu bencana tsunami melanda Kabupaten Bantul.

Aplikasi Rute *Shelter* Tsunami Bantul menggunakan platform android. Ada 3 komponen penting dalam aplikasi android yakni View, Control, dan Model. Aplikasi ini akan berhubungan dengan server peta yakni Google Map untuk meminta data peta dan server web untuk meminta data dari DBMS MySQL. Arsitektur Aplikasi Rute *Shelter* Tsunami Bantul dapat dilihat pada Gambar 6.



Gambar 6 Arsitektur Aplikasi Rute Shelter Tsunami

3.2 KEBUTUHAN PERANGKAT LUNAK

1. Kebutuhan Fungsional

Kebutuhan fungsional merupakan deskripsi kebutuhan atau fungsionalitas dari perangkat lunak yang akan dikembangkan. Kebutuhan fungsional dari perangkat lunak pada penelitian ini dapat dilihat pada Tabel 1.

Tabel 1. Kebutuhan Fungsional Aplikasi

No	SRS-ID	Deskripsi
1.	SRS- ARSTB -F-01	Sistem dapat menampilkan daftar *shelter*
2.	SRS- ARSTB -F-02	*User* dapat memilih *shelter* yang ingin dituju dan mendapatkan informasi dari *shelter* yang dipilih.
3.	SRS- ARSTB -F-03	*User* dapat menemukan *shelter* mana yang paling dekat dengan posisi *user*.
4.	SRS- ARSTB -F-04	*User* dapat mendapatkan rute terpendek menuju *shelter* yang dipilih menggunakan Algoritma A-Star
5.	SRS- ARSTB -F-05	*User* dapat mendapatkan rute terpendek menuju *shelter* yang dipilih menggunakan Algoritma Dijkstra

2. Kebutuhan Non Fungsional

Kebutuhan non-fungsional merupakan deskripsi kebutuhan yang menentukan batasan kriteria untuk menilai pengoperasian perangkat lunak yang akan dikembangkan. Kebutuhan non-fungsional dari perangkat lunak pada penelitian ini dapat dilihat pada Tabel 2.

Tabel 2. Kebutuhan Non-Fungsional Aplikasi Rute *Shelter* Tsunami Bantul

No	SRS-ID	Deskripsi
1.	SRS- ARSTB -NF-01	Aplikasi Rute *Shelter* Tsunami Bantul diakses menggunakan *smartphone* android min versi 4.1 (Jelly Bean).
2.	SRS- ARSTB -NF-02	Aplikasi Rute *Shelter* Tsunami Bantul membutuhkan koneksi internet untuk menggunakannya.

3. Daftar Aktor

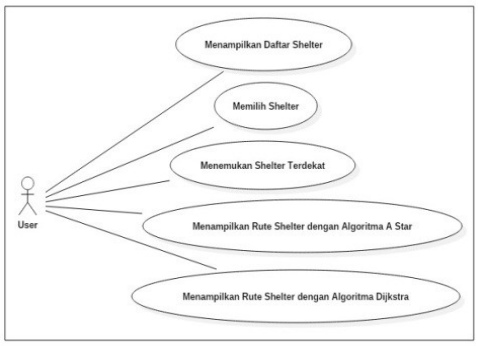
Aktor merupakan orang atau sistem lain yang berinteraksi dengan perangkat lunak. Aplikasi Rute *Shelter* Tsunami Bantul memiliki 1 aktor tunggal, yakni user. Daftar aktor dapat dilihat pada Tabel 3.

Tabel 3. Daftar Aktor Aplikasi Rute *Shelter* Tsunami Bantul

No	Aktor	Deskripsi
1.	*User*	Pengguna yang mengakses Aplikasi Rute *Shelter* Tsunami Bantul.

4. Use Case Diagram

Use Case diagram menggambarkan interaksi antara aktor dengan use case. Use Case diagram dari Aplikasi Rute *Shelter* Tsunami Bantul dapat dilihat pada Gambar 7.



Gambar 7 Use Case Diagram Aplikasi

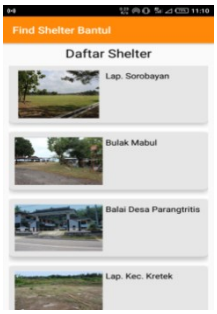
4 IMPLEMENTASI DAN PENGUJIAN

4.1 IMPLEMENTASI INTERFACE

Implementasi interface merupakan hasil implementasi dari interface design pada tahap design. Berikut ini adalah hasil dari implementasi interface.

1. Menampilkan Daftar Shelter

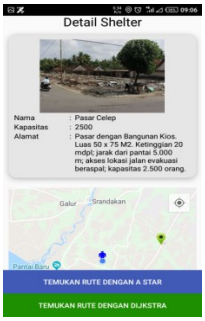
Berikut ini merupakan antarmuka menampilkan daftar *shelter* pada Aplikasi Rute Sehelter Tsunami Bantul yang dapat dilihat pada gambar 8.



Gambar 8 Antarmuka Daftar Shelter

2. Memilih Shelter

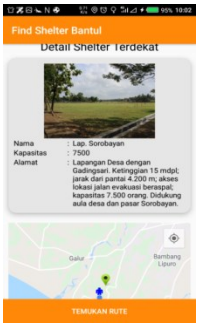
Berikut ini merupakan antarmuka detail *shelter* pada Aplikasi Rute Sehelter Tsunami Bantul yang dapat dilihat pada Gambar 9.



Gambar 9 Antarmuka Detail Shelter

3. Menemukan Shelter Terdekat

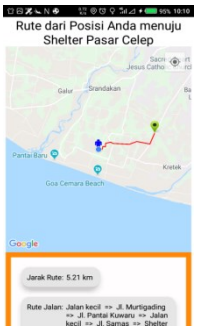
Berikut ini merupakan antarmuka detail *shelter* terdekat pada Aplikasi Rute Sehelter Tsunami Bantul yang dapat dilihat pada Gambar 10.



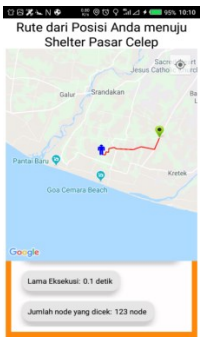
Gambar 10 Antarmuka Detail Shelter Terdekat

4. Menampilkan Rute Selter dengan Algoritma A-Star

Berikut ini merupakan antarmuka rute menuju *shelter* dengan Algoritma A-Star pada Aplikasi Rute Sehelter Tsunami Bantul yang dapat dilihat pada Gambar 11 dan 12.



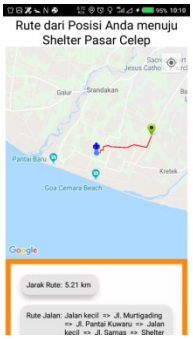
Gambar 11 Antarmuka Rute Shelter dengan A-Star (1)



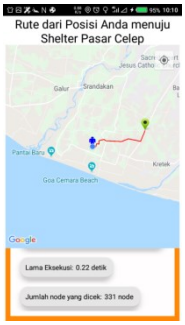
Gambar 12 Antarmuka Rute Shelter dengan A-Star (2)

5. Menampilkan Rute Selter dengan Algoritma Dijkstra

Berikut ini merupakan antarmuka rute menuju *shelter* dengan Algoritma Dijkstra pada Aplikasi Rute Shelter Tsunami Bantul yang dapat dilihat pada Gambar 13 dan 14.



Gambar 13 Antarmuka Rute Shelter dengan Dijkstra (1)



Gambar 14 Antarmuka Rute Shelter dengan Dijkstra (2)

4.2 PERBANDINGAN ALGORITMA A-STAR DENGAN DIJKSTRA

Algoritma A-Star dan Dijkstra masing-masing mempunyai cara kerja tersendiri untuk mencari rute terpendek menuju suatu tempat. Oleh karena itu pada penelitian ini dilakukan beberapa percobaan dimana pada setiap percobaan akan digunakan algoritma A-Star dan Dijkstra. Hal tersebut bertujuan untuk mengetahui perbandingan hasil dari penggunaan algoritma A-Star dengan Dijkstra. Ada 2 aspek yang diteliti pada perbandingan hasil dari penggunaan kedua algoritma tersebut yakni perbedaan lama eksekusi dan perbedaan jumlah *node* yang dicek. Detail Percobaan Algoritma dapat dilihat pada Tabel 4 dan hasil dari perbandingan algoritma dapat dilihat pada Tabel 5.

Tabel 4 Detail Percobaan Algoritma

No	*User*		*Shelter* Tujuan			Nama Percobaan
	Latitude	Longitude	Latitude	Longitude	Nama *Shelter*	
1.	-7.984792	110.272569	-7.968779	110.2554	Lap. Sorobayan	Percobaan 1
2.	-7.984792	110.272569	-8.020999	110.334533	Bulak Mabul	Percobaan 2
3.	-7.984792	110.272569	-8.00398	110.317108	Balai Desa Parangtritis	Percobaan 3
4.	-7.984792	110.272569	-7.972455	110.317636	Lap. Kec. Kretek	Percobaan 4
5.	-7.984792	110.272569	-7.970701	110.311313	Pasar Turi	Percobaan 5
6.	-7.984792	110.272569	-7.9696	110.299099	Lap. Tirtomulyo	Percobaan 6
7.	-7.984792	110.272569	-7.9693	110.285929	Pasar Celep	Percobaan 7
8.	-7.984792	110.272569	-7.975599	110.282884	Lap. Srigading	Percobaan 8

Tabel 5 Hasil Perbandingan Algoritma

No	Nama Percoban	Lama Eksekusi			Jumlah *Node* yang Dicek		
		A-Star	Dijkstra	Selisih	A-Star	Dijkstra	Selisih
1.	Percobaan1	0.1	0.29	0.17	89	302	213
2.	Percobaan2	0.31	0.42	0.11	291	530	239
3.	Percobaan3	0.28	0.41	0.13	231	468	237
4.	Percobaan4	0.28	0.4	0.12	232	461	229
5.	Percobaan5	0.18	0.38	0.2	156	443	287
6.	Percobaan6	0.09	0.25	0.16	82	346	264
7.	Percobaan7	0.02	0.19	0.17	23	221	198
8.	Percobaan8	0.02	0.11	0.09	17	143	126
Total				1.15			1793
Rata-rata				0.14			224

Berdasarkan hasil dari beberapa percobaan yang telah dilakukan maka dihasilkan poin-poin sebagai berikut

1. Algoritma A-Star menghasilkan rata-rata lama eksekusi 0.14 detik lebih cepat dibandingkan algoritma Dijkstra.
2. Algoritma A-Star menghasilkan rata-rata jumlah *node* yang dicek 224 *node* lebih sedikit dibandingkan algoritma Dijkstra.

4.3 PENGUJIAN

1. Perangkat Keras Pengujian

Perangkat keras pengujian terhadap Perbandingan Algoritma A-Star dan Dijkstra pada Pencarian Jalur Evakuasi Tsunami Terpendek menuju *Shelter* di Kabupaten Bantul Berbasis Aplikasi Android berupa smartphone. Smartphone tersebut mempunyai spesifikasi sebagai berikut.

a) CPU: MediaTek MT6753 1.30 GHz
b) RAM : 2907 MB
c) Internal Storage : 24.03 GB
d) Network : GSM
e) GPS : A-GPS

2. Perangkat Lunak Pengujian

Perangkat lunak pengujian terhadap Perbandingan Algoritma A-Star dan Dijkstra pada Pencarian Jalur Evakuasi Tsunami Terpendek menuju *Shelter* di Kabupaten Bantul Berbasis Aplikasi Android mempunyai spesifikasi sebagai berikut.

a) Platform : Android OS
b) OS Name : Nougat
c) OS Version : 7.0
d) API Level : 24

3. Pengujian

Pengujian terhadap Aplikasi Pencarian Jalur Evakuasi Tsunami Terpendek menuju *Shelter* di Kabupaten Bantul dengan algoritma A-Star dan Dijkstra berbasis android mengggunakan jenis pengujian black box. Jenis pengujian black box merupakan jenis pengujian yang hanya menguji fungsionalitas dari perangkat lunak. Rencana pengujian dapat dilihat pada Tabel 6.

Tabel 6 Pengujian

No	*Use Case*	Jenis Pengujian	Kode Pengujian
1.	Menampilkan Daftar *Shelter*	*Black Box*	P-1-01
2.	Memilih *Shelter*	*Black Box*	P-2-01
3.	Menampilkan *Shelter* Terdekat	*Black Box*	P-3-01
4.	Menampilkan Rute *Shelter* dengan Algoritma A-Star	*Black Box*	P-4-01
5.	Menampilkan Rute *Shelter* dengan Algoritma Dijkstra	*Black Box*	P-5-01

5 KESIMPULAN DAN SARAN

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Aplikasi android perbandingan algoritma A-Star dan Dijkstra pada pencarian jalur evakuasi tsunami terpendek menuju *shelter* di Kabupaten Bantul berhasil dikembangkan dengan model Waterfall.
2. Algoritma A-Star menghasilkan rata-rata lama eksekusi 0.14 detik lebih cepat dan rata-rata jumlah *node* yang dicek 224 *node* lebih sedikit dibandingkan algoritma Dijkstra.

Saran untuk pengembangan lebih lanjut penelitian ini adalah perbandingan perhitungan waktu dapat dibandingkan berdasarkan perhitungan kompleksitas dari algoritmanya.

DAFTAR PUSTAKA

[1] G. Grehenson, "Rawan Gempa dan Tsunami, Bantul Perlu Perkuat Mitigasi," 2011. [Online]. Available: https://ugm.ac.id/id/berita/3911-rawan.gempa.dan.tsunami.bantul.perlu.perkuat.mitigasi. [Diakses 20 10 2015].

[2] A. M. Zulfa, "a. Aplikasi Penentuan Rute Evakuasi Bencana Tsunami Kota Padang Menggunakan Algoritma A-Star Berbasis Hybrid Application," DigilibFakultas Sains dan Matematika Universitas Diponegoro, Semarang, 2015.

[3] Yuliani dan F. Agus, "Webgis Pencarian Rute Terpendek Menggunakan Algoritm A Star (A*) (Studi Kasus: Kota Bontang)," *Informatika,* vol. VII, no. 2, pp. 50-55, 2013.

[4] B. T. Wibowo, "Aplikasi Penentuan Jalur Terpendek untuk Pemadam Kebakaran dengan Menggunakan Metode Dijkstra," *Informatika,* vol. 7, no. 2, pp. 172-176, 2014.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest dan C. Stein, Introduction to Algorithms, 3rd penyunt., London: Massachusetts Institute of Technology Press, 2009.

[6] R. Munir, Matematika Diskrit, Bandung : Informatika, 2012.

[7] D. Prasetyo, "Penerapan Haversine Formula pada Aplikasi Pencarian Lokasi dan Informasi Gereja Kristen di Semarang Berbasis Mobile," 18 12 2018. [Online]. Available: http:/http://eprints.dinus.ac.id/15004/1/jurnal_14842.pdf.

[8] M. H. Fadhlurrahman, "Implementasi dan Analisis Penggunaan Algoritma A-Star dengan Prioritas pada Pemilihan Rute Lintas Kendaraan Roda Dua," Universitas Telkom, 2015, 2015.

[9] L. Alsedà, "Optimisation. Master's degree in Modelling for Science and Engineering," 18 12 2018. [Online]. Available: http://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf.

[10] W. Setiawan, "Pembahasan Pencarian Lintasan Terpendek Menggunakan Algoritma Dijkstra dan A*," Sekolah Teknik Elektro dan Informatika, ITB, Bandung, 2010.

[11] A. Goyal, P. Mogha, R. Luthra dan M. N. Sangwan, "Path Finding: A* o]r Dijkstra's?," *International Journal in IT and Engineering,* vol. 02, no. 01, pp. 1-15, January 2014.

[12] I. Sommerville, "Software Engineering (Rekayasa Perangkat Lunak)," Jakarta, Erlangga, 2011, p. 30.