Jurnal Matematika Vol. 14, No.1, Juni 2024, pp. 52-60 ISSN: 1693-1394
Article DOI: 10.24843/JMAT.2024.v14.i01.p171

Algoritma Dijkstra : Rute Pengungsian Terpendek Daerah Rawan Bencana di Desa Canggu

Ida Bagus Kade Puja Arimbawa K.
Universitas Bali Dwipa
e-mail: kemenuh.puja@gmail.com

***Abstract*:** *Canggu Village, located in North Kuta, Badung Regency, Bali, Indonesia, is a popular tourist area with a population density of 1,305 people per square kilometer. This region offers an attractive combination of natural beauty, rich Balinese culture, and modern lifestyle, yet it also faces high risks of natural disasters such as floods, tsunamis, and coastal erosion due to its coastal location. The coastal geography of Canggu necessitates awareness and preparedness for disasters. This study focuses on formulating and implementing evacuation routes using the Dijkstra algorithm to find the shortest and safest evacuation path from disaster-prone areas to safe zones. The results indicate that the Dijkstra algorithm is effective in establishing evacuation routes from the most vulnerable points, such as Batu Bolong Beach, to safe zones with varying distances, allowing residents to leave high-risk areas quickly and safely. These recommendations are expected to be used by local governments in disaster planning and management.*

Keywords: *dijkstra algorithm, shortest route, canggu village*

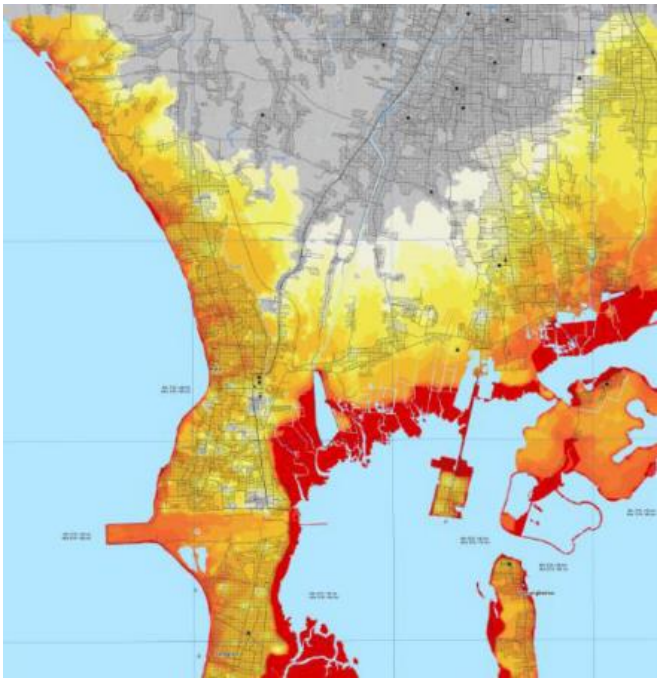
***Abstrak*:** *Desa Canggu, terletak di Kuta Utara, Kabupaten Badung, Bali, Indonesia, merupakan sebuah kawasan wisata populer dengan kepadatan penduduk sebesar 1.305 orang per kilometer persegi. Wilayah ini menawarkan kombinasi menarik antara keindahan alam, kekayaan budaya Bali, dan gaya hidup modern, namun juga menghadapi risiko bencana alam yang tinggi seperti banjir, tsunami, dan abrasi pantai karena lokasinya yang berada di pesisir. Geografi pesisir Canggu menuntut kesadaran dan kesiapsiagaan terhadap bencana. Penelitian ini fokus pada perumusan dan implementasi rute pengungsian menggunakan algoritma Dijkstra untuk menemukan jalur pengungsian terpendek dan teraman dari daerah rawan bencana ke zona aman. Hasil penelitian menunjukkan bahwa algoritma Dijkstra efektif dalam menetapkan rute pengungsian dari titik-titik paling rawan seperti Pantai Batu Bolong menuju zona aman dengan jarak yang variatif, memungkinkan warga meninggalkan area berisiko dengan cepat dan aman. Rekomendasi ini diharapkan dapat digunakan oleh pemerintah setempat dalam perencanaan dan manajemen bencana.*

Kata Kunci: *algoritma dijkstra, rute terpendek, desa canggu*

1. Pendahuluan

Desa Canggu merupakan sebuah area yang terletak di Kuta Utara, Kabupaten Badung, Bali, Indonesia (BPS Kabupaten Badung, 2023). Kawasan ini dikenal sebagai salah satu destinasi wisata populer, terutama di kalangan wisatawan asing. Canggu menawarkan kombinasi unik antara keindahan alam, budaya Bali yang kental, serta gaya hidup modern.

Desa Canggu dengan kepadatan hanya 1.305 orang per kilometer persegi (Badan Pusat Statistik Kabupaten Badung, 2022) merupakan daerah yang berpotensi rawan bencana (DLR/GTZ, 2010). Sebagai sebuah desa pesisir di pulau Bali, Canggu menghadapi risiko bencana alam seperti banjir, terutama selama musim hujan(Badan Meteorologi, Klimatologi, 2012). Canggu memiliki potensi risiko banjir yang tinggi ditinjau dari berbagai aspek seperti curah hujan, kemiringan lereng, kerapatan sungai, jenis tanah, penggunaan lahan, dan ketinggian tempat dari permukaan laut. Bencana yang timbul dari aktivitas laut seperti tsunami serta abrasi pantai, canggu juga termasuk daerah yang rawan tingkat tinggi untuk bencana tersebut. Keadaan geografis sebagai daerah pesisir menunjukkan pentingnya kesadaran dan kesiapsiagaan terhadap potensi bencana.



Gambar 1. Peta bahaya bencana (tsunami) Bali Selatan (Anwar et al., 2014)

Perencanaan terkait bencana yang telah dijelaskan sebelumnya sangat penting karena menyangkut nyawa serta upaya meminimalkan kerugian materi ketika bencana terjadi. Perencanaan yang baik harus mencakup pendidikan dan latihan reguler bagi masyarakat tentang prosedur pengungsian, termasuk identifikasi lokasi penampungan sementara. Perencanaan yang tidak kalah pentingnya adalah membuat rute pengungsian yang jelas

dan mudah diakses memungkinkan warga untuk cepat dan aman meninggalkan area berisiko tinggi.

Pembentukan rute pengungsian dalam manajemen bencana, seperti banjir atau tsunami, adalah proses komprehensif yang melibatkan identifikasi area berisiko, analisis geografis untuk menentukan rute pengungsian paling aman serta algoritma pencarian rute untuk menemukan rute terpendek dan teraman ke zona aman. Perumusan masalah yang telah dijelaskan diatas memfokuskan tujuan penelitian ini untuk mencari rute pengungsian terpendek dengan menggunakan algoritma Dijkstra daerah rawan bencana di Desa Canggu. Penentuan rute melibatkan panjang jalan sebagai sisi serta persimpangan dan ujung jalan yang digunakan sebagai simpul.

Beberapa penelitian sebelumnya yang serupa telah diperoleh, salah satunya (Giyai & Pamungkas, 2022) tentang penentuan titik dan rute evakuasi di Kecamatan Mimika Baru, Kabupaten Mimika. Hasil penelitian menunjukan bahwa studi kasus di Desa Koperapoka titik dan rute evakuasi yang diperoleh dengan *content analysis* untuk skoring dan pembobotan penilaian daerah evakuasi serta *network analysis* dalam penentuan rute evakuasi. Diperoleh 22 daerah dan rute evakuasi yang layak, namun dalam penelitian ini masih dapat dicari rute paling optimal dengan menggunakan algoritma tertentu, salah satunya algoritma Dijkstra.

Penelitian lain oleh (Irawan et al., 2023) menentukan jalur evakuasi bencana gunung tangkuban perahu di Kecamatan Lembang, Kabupaten Bandung Barat. Jalur evakuasi terpendek diperoleh dengan menggunakan algoritma Dijkstra dengan jarak sebesar 1,237 Km. Hasil ini memiliki jarak yang lebih pendek dibandingkan perhitungan manual yaitu sebesar 1,877 Km. Komparasi antara 2 algoritma dipaparkan dalam penelitian (Rudiyanto et al., 2020) untuk kasus penentuan rute rumah sakit terdekat jalur evakuasi kecelakaan lalu lintas berbasis web. Dalam penelitian ini dirancang sebuah system web untuk perbandingan algoritma Floyd-Warshall dan Dijkstra untuk proses pencarian rute evakuasi. Hasilnya, algortima Floyd-Warshall menghasilkan rute lebih pendek dibandingkan algoritma Dijkstra, namun dalam penggunaan memori dan waktu, algoritma Floyd-Warshall lebih unggul. Penelitian lain tentang algoritma Dijkstra oleh (K et al., 2023) dalam mencari rute evakuasi terpendek tsunami di Keluarahan Benoa. Hasil penelitian menunjukan bahwa algoritma Dijkstra dapat menghasilkan jarak terpendek dari 8 daerah rawan tsunami menuju 2 daerah zona aman.

2. Metode Penelitian

2.1. Tahapan penelitian

Penelitian ini terbagi menjadi 4 tahap, yaitu identifikasi masalah, pengumpulan data, pengolahan data dan kesimpulan. Adapun penjelasan dari tahap tersebut adalah sebagai berikut.

Identifikasi masalah, melakukan pendekatan dengan petugas yang berwenang di Desa Canggu untuk mengetahui daerah rawan dengan kepadatan penduduk. Seluruh jalan di Desa Canggu ditelusuri untuk mendapat gambaran awal tentang peta pengungsian yang akan dirancang.

Pengumpulan data, tahap ini dilakukan dengan mengukur panjang seluruh jalan yang ada di Desa Canggu kemudian dipetakan sesuai dengan kondisi geografis yang ada. Pada tahap ini disusun graf berbobot dengan simpul (*vertex*) mewakili titik persimpangan jalan, titik pengungsian awal serta zona aman bencana, sisi (*edge*) mewakili jalan dengan jarak yang mewakili bobot graf.

Pengolahan data, implementasi algoritma Dijkstra dari matriks ketetanggaan yang diperoleh dari graf berbobot. Algoritma Djiktra dijalankan dengan memanfaatkan C++ untuk memperoleh rute pengungsian serta jaraknya.

Kesimpulan, disusun setelah memperoleh seluruh rute pengungsian terpendek dari setiap daerah menuju zona aman.

2.2. Matriks *Adjecency*

Dalam sebuah graf $G = (V, E)$ yang memiliki n simpul, matriks ketetanggaan *(adjacency matrix*) dan ditulis sebagai $A = [a_{ij}]$, untuk hubungan antar simpul (Munir, 2010). Setiap a_{ij} dalam matriks memiliki nilai spesifik yang menandakan apakah terdapat hubungan langsung antara simpul i dan simpul j (Khaleel & Al-Shumam, 2020). Berikut adalah penjelasan umum tentang matriks A (Sporns, 2022):

$$a_{ij} = \begin{cases} 1 & \text{jika simpul } i \text{ terhubung dengan simpul } j \\ 0 & \text{jika simpul } i \; tidak \; terhubung \text{ dengan simpul } j \end{cases} \quad (1)$$

Matriks ini berbentuk persegi ($n \times n$) karena memiliki sebuah baris dan kolom untuk setiap simpul dalam graf. Diagonal utama $a_{ii} = 0$, khususnya pada graf yang sederhana di mana tidak diizinkan adanya *loop* (hubungan dari simpul ke dirinya sendiri).

2.3. Algoritma Dijkstra

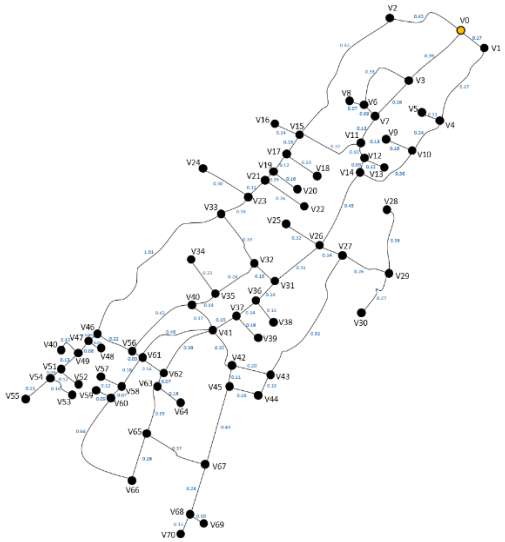
Algoritma ini dikenal dengan nama penciptanya, Edsger Dijkstra, seorang ilmuwan dari Belanda dikembangkan tahun 1956 dan diperkenalkan pertama kali pada tahun 1959. Algoritma Dijkstra berfungsi untuk mengatasi masalah pencarian jalur atau lintasan terpendek, yang pada dasarnya adalah pencarian jalur dengan jumlah bobot terkecil seperti jarak antar dua simpul pada graf berbobot (Dijkstra, 1959). Algoritma ini bertujuan menemukan jarak minimum dari simpul awal ke simpul-simpul lain, sehingga jalur yang dihasilkan dari simpul awal menuju simpul tujuan memiliki bobot total yang paling rendah.

Langkah-langkah dalam algoritma Dijkstra, seperti yang dijelaskan oleh (Akram et al., 2021) (Deng et al., 2012) (Luo et al., 2020) sebagai berikut:

a) **Inisialisasi**: Mulai dengan simpul awal, atur jaraknya menjadi nol dan berikan jarak tak terhingga (atau sangat besar) ke semua simpul lain sebagai jarak awal. Tentukan juga simpul awal sebagai simpul aktif saat ini.
b) **Pemilihan Simpul Aktif**: Tentukan simpul dengan jarak terpendek dari simpul awal sebagai simpul aktif. Ini adalah simpul yang pada langkah awal memiliki jarak nol.
c) **Pembaruan Jarak**: Evaluasi semua simpul tetangga dari simpul aktif dan cek apakah jarak melalui simpul aktif ini lebih pendek dibandingkan jarak saat ini ke tetangga tersebut. Jika benar, perbarui jarak tetangga dengan jarak yang lebih pendek.
d) **Penandaan Simpul Aktif**: Setelah memperbarui jarak ke semua tetangga, tandai simpul aktif sebagai "telah dikunjungi" atau "selesai".
e) **Pemilihan Simpul Baru**: Pilih simpul dengan jarak terpendek yang belum dikunjungi sebagai simpul aktif berikutnya. Ini adalah simpul dengan jarak terpendek di antara semua simpul yang belum dikunjungi.
f) **Pengulangan**: Ulangi langkah 3 hingga 5 sampai semua simpul telah dikunjungi atau simpul tujuan tercapai.
g) **Hasil**: Setelah semua simpul telah dikunjungi atau simpul tujuan dicapai, hasil yang diperoleh adalah jarak terpendek dari simpul awal ke simpul tujuan.

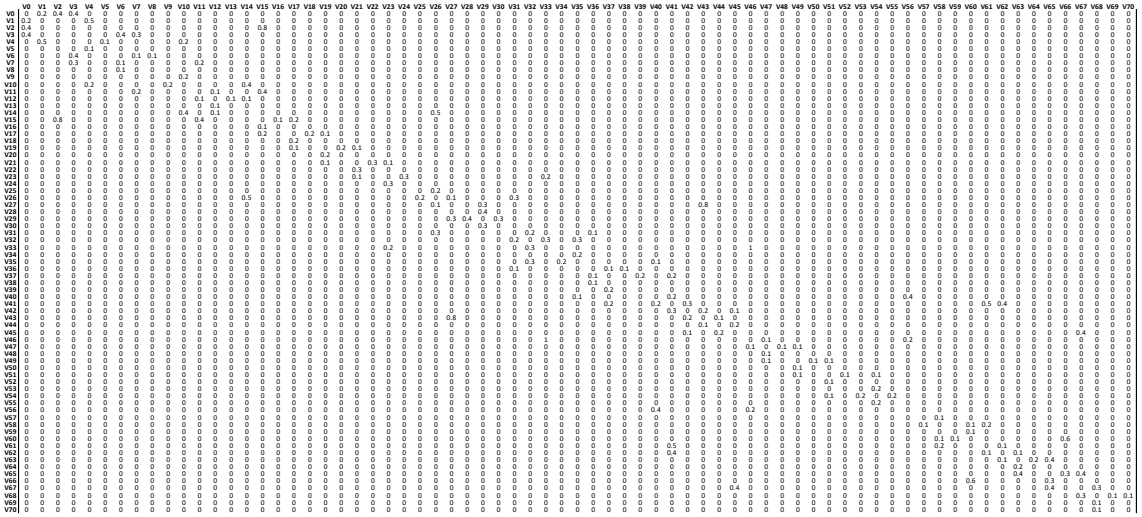
3. Hasil dan Pembahasan

Graf berbobot disusun berdasarkan data yang telah diperoleh dengan mengukur panjang jalan (dalam kilometer) dan data simpul yang telah ditentukan. Simpul mewakili ujung jalan dan persimpangan jalan. Sebanyak 71 simpul diperoleh dan disajikan pada Gambar 1.



Gambar 2. Graf Berbobot

Zona aman ditentukan berdasarkan zona yang direkomendasikan (DLR/GTZ, 2010) yang menjadi simpul tujuan yaitu v_0 (Jalan Raya Canggu Nomor 99 -8.636757, 115.147090) dan simpul $v_{55}, v_{56}, v_{69}, v_{70}$ sebagai simpul paling rawan bencana pesisir Pantai Batu Bolong yang selanjutnya disebut simpul awal. Selanjutnya dibuat matriks ketetanggaan berdasarkan data dari pada gambar 2.



Gambar 3. Matriks ketetanggan

Pencarian rute pengungsian terpendek menggunakan Algoritma Dijkstra dikembangkan dalam bahasa pemrograman C++, dengan menentukan dan menampilkan jarak tempuh dari semua simpul ke simpul zona aman (v_0). Berikut ini adalah sebagian dari sintaks C++ untuk Algoritma Dijkstra.

```
void dijkstra(vector<vector<double>>& graph, int start, int V) {
    double dist[V];
    bool sptSet[V];
    int parent[V];

    for (int i = 0; i < V; i++) {
        dist[i] = numeric_limits<double>::max();
        sptSet[i] = false;
        parent[i] = -1;
    }

    dist[start] = 0.0;

    for (int count = 0; count < V - 1; count++) {
        int u = cariJarakTerpendek(dist, sptSet, V);
        sptSet[u] = true;

        for (int v = 0; v < V; v++) {
            if (!sptSet[v] && graph[u][v] > 0.0 && dist[u] !=
               numeric_limits<double>::max() && dist[u] + graph[u][v] < dist[v]) {
                parent[v] = u;
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
```

Gambar 4. Sintax C++ algoritma Dijkstra

Program ini dijalankan dengan merancang fungsi yang mencari simpul dengan jarak terpendek ke simpul yang belum dikunjungi, dan kemudian membuat fungsi lain untuk mencetak hasil jalur terpendek tersebut. Pada tampilan awal, pengguna akan diminta untuk memasukkan jumlah simpul, matriks ketetanggaan, serta simpul awal.

```
// Fungsi untuk mencari simpul dengan jarak terpendek yang belum dikunjungi
int cariJarakTerpendek(double dist[], bool sptSet[], int V)
{
    double minDist = numeric_limits<double>::max();
    int minIndex;

    for (int v = 0; v < V; v++) {
        if (!sptSet[v] && dist[v] <= minDist) {
            minDist = dist[v];
            minIndex = v;
        }
    }

    return minIndex;
}

// Fungsi untuk mencetak hasil jalur terpendek ke simpul zona aman
void cetakJalur(int parent[], int j) {
    if (parent[j] == -1)
        return;

    cetakJalur(parent, parent[j]);
    cout << " -> " << j;
```

```
int main() {
    int V;
    cout << "Masukkan jumlah simpul: ";
    cin >> V;

    vector<vector<double>> graph(V, vector<double>(V));

    cout << "Masukkan matriks adjacency:" << endl;
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            cin >> graph[i][j];
        }
    }

    int start;
    cout << "Masukkan simpul awal: ";
    cin >> start;

    dijkstra(graph, start, V);
```

Gambar 5. Sintax C++ beberapa fungsi untuk mencari jalur dan matriks ketetanggaan

Rute pengungsian terpendek ke simpul v_0 (zona aman) diperoleh setelah menjalankan program untuk semua simpul. Rute pengungsiannya adalah sebagai berikut :

No	Simpul awal	Rute Evakuasi	Jarak (km)
1	v1	v1 - v0	0,17
2	v2	v2 - v0	0,43
3	v3	v3 - v0	0,39
4	v4	v4 - v1 - v0	0,64
5	v5	v5 - v4 - v1 - v0	0,75
6	v6	v6 - v7 - v3 - v0	0,74
7	v7	v7 - v3 - v0	0,65
8	v8	v8 - v6 - v7 - v3 - v0	0,81
9	v9	v9 - v10 - v4 - v1 - v0	1,04
10	v10	v10 - v4 - v1 - v0	0,88
11	v11	v11 - v7 - v3 - v0	0,83
12	v12	v12 - v11 - v7 - v3 - v0	0,93
13	v13	v13 - v12 - v11 - v7 - v3 - v0	1,05
14	v14	v14 - v12 - v11 - v7 - v3 - v0	1,02
15	v15	v15 - v11 - v7 - v3 - v0	1,20
16	v16	v16 - v15 - v11 - v7 - v3 - v0	1,34
17	v17	v17 - v15 - v11 - v7 - v3 - v0	1,35
18	v18	v18 - v17 - v15 - v11 - v7 - v3 - v0	1,55
19	v19	v19 - v17 - v15 - v11 - v7 - v3 - v0	1,47
20	v20	v20 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,63
21	v21	v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,52
22	v22	v22 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,78
23	v23	v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,64
24	v24	v24 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,94
25	v25	v25 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,69
26	v26	v26 - v14 - v12 - v11 - v7 - v3 - v0	1,47
27	v27	v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,61
28	v28	v28 - v29 - v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,23
29	v29	v29 - v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,87
30	v30	v30 - v9 - v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,14
31	v31	v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,78
32	v32	v32 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,93
33	v33	v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	1,82
34	v34	v34 - v35 - v32 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,44
35	v35	v35 - v32 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,21

No	Simpul awal	Rute Evakuasi	Jarak (km)
36	v36	v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	1,92
37	v37	v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,06
38	v38	v38 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,06
39	v39	v39 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,25
40	v40	v40 - v35 - v32 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,35
41	v41	v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,21
42	v42	v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,46
43	v43	v43 - v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,42
44	v44	v44 - v43 - v27 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,55
45	v45	v45 - v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,57
46	v46	v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	2,83
47	v47	v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	2,89
48	v48	v48 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	2,95
49	v49	v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	2,97
50	v50	v50 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,10
51	v51	v51 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,10
52	v52	v52 - v51 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,22
53	v53	v53 - v54 - v51 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,31
54	v54	v54 - v51 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,15
55	v55	v55 - v54 - v51 - v49 - v47 - v46 - v33 - v23 - v21 - v19 - v17 - v15 - v11 - v7 - v3 - v0	3,36
56	v56	v56 - v61 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,70
57	v57	v57-v58-v61-v41-v37-v36-v31-v26-v14-v12-v11-v7-v3-v0	2,97
58	v58	v58 - v61 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,85
59	v59	v59 - v60 - v58 - v61 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,01
60	v60	v60 - v58 - v61 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,92
61	v61	v61 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,67
62	v62	v62 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,59
63	v63	v63 - v62 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,66
64	v64	v64 - v63 - v62 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	2,82
65	v65	v65 - v63 - v62 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,01
66	v66	v66 - v65 - v63 - v62 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,27
67	v67	v67 - v45 - v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,00
68	v68	v68 - v67 - v45 - v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,28
69	v69	v69 - v68 - v67 - v45 - v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,37
70	v70	v70 - v68 - v67 - v45 - v42 - v41 - v37 - v36 - v31 - v26 - v14 - v12 - v11 - v7 - v3 - v0	3,41

Gambar 6. Rute Pengungsian terpendek (*output* C++)

4. Kesimpulan dan Saran

Dari hasil dan pembahasan, dapat disimpulkan bahwa Algoritma Dijkstra dapat diterapkan untuk mencari rute pengungsian terpendek dari daerah rawan bencana (banjir,

tsunami dan abrasi pantai. Titik paling rawan bencana yaitu simpul $v_{55}, v_{56}, v_{69}, v_{70}$ yang terletak di pesisir Pantai Batu Bolong menuju titik zona aman (simpul v_0) menjadi rute penting yang dapat direkomendasikan ke pemerintah setempat. Jarak simpul berturut-turut adalah sebagai berikut v_{55} menuju v_0 dengan jarak 3,36 km, v_{56} menuju v_0 dengan jarak 2,70 km, v_{69} ke v_0 dengan jarak 3,37 km dan v_{70} ke v_0 dengan jarak 3,41 km. Peran program C++ yang telah dikembangkan sangat membantu penulis dalam menjalankan algoritma Dijkstra dan membantu penulis dalam perhitungan yang lebih cepat dan akurat dibandingkan dengan perhitungan manual.

Graf yang dirancang hanya menggunakan 71 simpul yang diperoleh dengan hanya memperhatikan jalan yang dapat dilalui kendaraan. Saran untuk penelitian selanjutnya adalah agar memperhatikan juga jalan kecil serta jalur pejalan kaki sehingga persimpangan jalan yang menjadi simpul pada graf lebih banyak. Akibatnya, rute yang dicari dengan algoritma Djikstra akan semakin bervariasi sehingga mendapat rute yang paling optimal. Penelitian ini juga dapat dibandingkan dengan algoritma optimasi yang lain sehingga memperoleh hasil yang lebih baik.

Daftar Pustaka

Akram, M., Habib, A., & Alcantud, J. R. (2021). An optimization study based on Dijkstra algorithm for a networkwith trapezoidal picture fuzzy numbers. *Neural Computing and Applications*, *33*(4), 1329–1342. https://doi.org/https://doi.org/10.1007/s00521-020-05034-y(0123456

Anwar, H. Z., Latief, H., Meilano, I., Yustiningrum, E., Komarudin, R., & Asvantina, V. (2014). *Pedoman Penyusunan Peta Resiko Tsunami Tingkat Kabupaten/Kota*.

Badan Meteorologi, Klimatologi, dan G. (BMKG) dan G.-I. G. (2012). *Pedoman Pelayanan Peringatan Dini Tsunami InaTEWS*. https://www.gitews.de/tsunami-kit/id/E3/perangkat/Pedoman Pelayanan Peringatan Dini Tsunami InaTEWS (2).pdf

Badan Pusat Statistik Kabupaten Badung. (2022). Kecamatan Kuta Utara dalam Angka. In *Katalog 1102001.5103030*. BPS Kabupaten Badung.

BPS Kabupaten Badung. (2023). *Statistik Daerah Kabupaten Badung 2023*.

Deng, Y., Chen, Y., Zhang, Y., & Mahadevan, S. (2012). Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*, *12*(3), 1231–1237. https://doi.org/10.1016/j.asoc.2011.11.011

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*, 269–271.

DLR/GTZ. (2010). *Peta Bahaya Tsunami Bali*.

Giyai, M. C., & Pamungkas, A. (2022). Penentuan Titik dan Rute Evakuasi dalam Mengurangi Risiko Bencana Banjir (Studi Kasus: Kecamatan Mimika Baru, Kabupaten Mimika). *Jurnal Teknik ITS*, *11*(3), C130–C135.

Irawan, J., Nuryani, S., & Pandin, R. M. (2023). Penentuan Jalur Evakuasi Terbaik Bencana Gunung Tangkuban Perahu Menggunakan Algoritma Dijkstra. *CICES*

(Cyberpreneurship Innovative and Creative Exact and Social Science), *9*(2), 174–184. https://doi.org/10.33050/cices.v9i2.2691

K, I. B. K. P. A., Sukartiasih, W., & Sedayu, A. (2023). Optimization of the Shortest Tsunami Evacuation Route Using Djikstra's Algorithm in Benoa Village. *Brilliance: Research of Artificial Intelligence*, *3*(2), 217–224. https://doi.org/10.47709/brilliance.v3i2.3089

Khaleel, T. A., & Al-Shumam, A. A. (2020). A Study of Graph Theory Applications in IT Security. *Iraqi Journal of Science*, *61*(10), 2705–2714. https://doi.org/https://doi.org/10.24996/ijs.2020.61.10.28

Luo, M., Hou, X., & Yang, J. (2020). Surface Optimal Path Planning Using An Extended Dijkstra Algorithm. *IEEE Access*, *8*, 147827–147838. https://doi.org/10.1109/ACCESS.2019.DOI

Munir, R. (2010). *Matematika Diskrit* (Ketiga). Informatika Bandung.

Rudiyanto, A. D., Wahyuddin, M. I., & Andrianingsih. (2020). Perbandingan Algoritma Floyd-Warshall dan Dijkstra untuk Menentukan Rute Rumah Sakit Terdekat Jalur Evakuasi Kecelakaan Lalu Lintas Berbasis Web. *Journal of Information Technology and Computer Science (INTECOMS)*, *3*(2), 336–345. https://doi.org/https://doi.org/10.31539/intecoms.v3i2.1843

Sporns, O. (2022). Graph theory methods: applications in brain networks. *Dialogues Clin Neurosci*, *20*(2), 111–120. https://doi.org/10.31887/DCNS.2018.20.2/osporns