

**STRATEGI ALGORITMA IF2211 *Implementasi Algoritma A* untuk
Menentukan Lintasan Terpendek***

LAPORAN

Sebagai Bagian dari Tugas Kecil 3 mata kuliah Strategi Algoritma Kelas 02 pada semester 2
Tahun Akademik 2020/2021



Oleh

Kadek Dwi Bagus Ananta Udayana 13519057

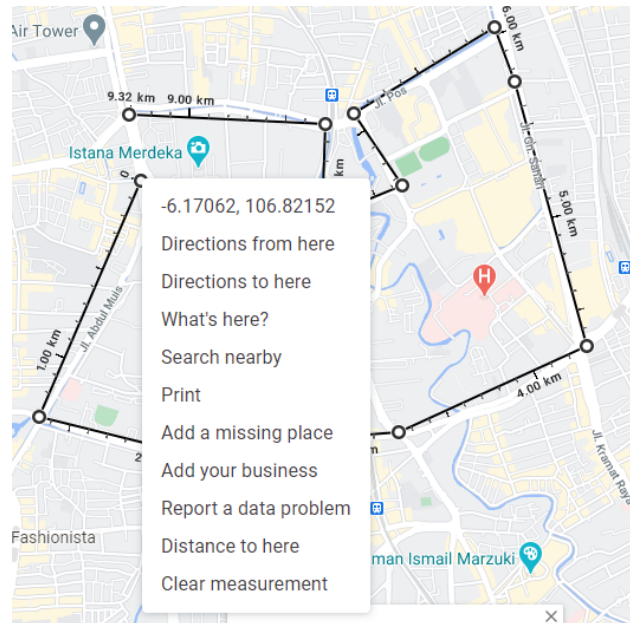
Faris Aziz 13519065

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2021

A. Deskripsi Program

Program Map Iki? adalah sebuah program yang berfungsi untuk menghasilkan lintasan terpendek dari suatu nodes dengan menggunakan algoritma A*. Program ini mengharuskan pengguna menginput file yang berisi jumlah titik yang ada di peta yang berisi nama beserta latitude dan longitudenya.



Gambar A.1. Google Maps yang menunjukkan latitude dan longitude

Lalu pengguna dapat memilih 2 titik yang akan menjadi simpul awal dan juga akhir dari perjalanan. Setelah dipilih 2 simpul tersebut maka program akan memanggil fungsi A* yang menghitung total jarak rute terpendek lalu memvisualisasikannya.

Algoritma A* yang digunakan ini adalah algoritma pencarian rute yang menghindari pemilihan suatu rute yang sudah terlalu besar nilainya. Dalam menentukan nilai besar dan kecilnya suatu rute algoritma ini menghitung dengan menyimpan total jarak yg telah dilalui lalu nilainya ditambah dengan jarak euclidean dari simpul sekarang menuju dengan simpul tujuan.

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

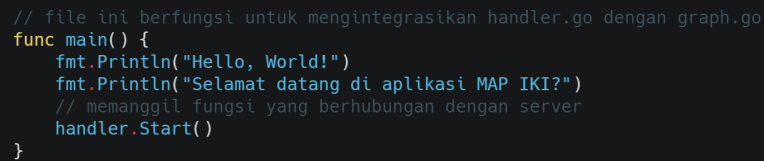
Fungsi h menyimpan perkiraan nilai dari simpul sekarang sampai simpul tujuan dengan menggunakan jarak euclidean, sedangkan fungsi g menyimpan nilai rute yang sudah dilalui. Lalu nilai yang dihasilkan dari penjumlahan fungsi h dan fungsi g disimpan sebagai perkiraan nilai apabila melalui rute tersebut. Prosedur ini dilakukan dengan memeriksa simpul tetangga dari simpul sekarang lalu tentukan nilai fungsi f nya dan dimasukkan ke dalam suatu variabel penampung data. Selanjutnya jika sudah selesai pemrosesan simpul tetangga maka akan dilanjutkan dengan pemrosesan simpul yang sudah disimpan dengan nilai fungsi f terbesar.

B. Algoritma

Folder utama untuk menyimpan algoritma pemrograman ini adalah folder graph, main, dan handler. Masing-masing folder memiliki peran tersendiri, yaitu untuk folder main merupakan tempat untuk menyimpan inti dari program ini seperti

memanggil dan mengintegrasikan 2 folder lainnya juga terdapat template html,css dan juga javascript untuk menunjang proses visualisasi hasil. kemudian pada folder graph disimpan algoritma-algoritma dan struktur data yang menunjang pencarian sebuah rute dengan menggunakan representasi graf sebagai matriks ketetanggaan, lalu pada folder handler menyimpan algoritma algoritma yang berkaitan dengan visualisasi data.

1. File main.go



```
// file ini berfungsi untuk mengintegrasikan handler.go dengan graph.go
func main() {
    fmt.Println("Hello, World!")
    fmt.Println("Selamat datang di aplikasi MAP IKI?")
    // memanggil fungsi yang berhubungan dengan server
    handler.Start()
}
```

Gambar B.1.1 Fungsi main yang mengintegrasikan file

2. File handler.go



```
func Start() {
    // definisi rute web
    http.NewServeMux()
    http.HandleFunc("/", homeHandler)
    http.HandleFunc("/polyline", polylineHandler)
    http.HandleFunc("/map", indexHandler)
    http.HandleFunc("/Hello", helloHandler)
    http.HandleFunc("/input", inputHandler)

    // memfiksasi direktori sehingga dapat dipanggil file yang ada di dalamnya dengan /static/
    http.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir("assets"))))

    fmt.Println("server started at localhost:8080")
    // mengaktifkan server
    http.ListenAndServe(":8080", nil)
}
```

Gambar B.2.1 Fungsi start yang mendefinisikan rute web dan mengaktifkan server lokal

```

func polylineHandler(w http.ResponseWriter, r *http.Request) {
    // mendapatkan input node awal dan tujuan
    if r.Method == "POST" {
        var filepath = path.Join("views", "polyline.html")
        var tmpl = template.Must(template.New("result").ParseFiles(filepath))

        if err := r.ParseForm(); err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
            return
        }
        // graf = graph.ReadFile("tes1.txt")
        var simpulAwal = r.FormValue("simpulAwal")
        var simpulAkhir = r.FormValue("simpulAkhir")
        fmt.Println(simpulAwal)

        // mendefinisikan tipe graf
        graf := graph.ReadFile(fileName)

        // mendapatkan jarak dan rute
        distance, rute := graf.Astar(simpulAwal, simpulAkhir)

        fmt.Print("Jarak dari " + simpulAwal + " ke " + simpulAkhir + " :")
        fmt.Println(distance)
        ruteInfo := graf.GetNodeswithIndex(rute, distance)
        fmt.Println(ruteInfo)

        // mengoper data ke dalam server
        if err := tmpl.Execute(w, ruteInfo); err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
        }
        return
    }
}

```

Gambar B.2.2 Fungsi polyline handler yang mengolah data rute dan mengirimkan ke server lokal

```

func indexHandler(w http.ResponseWriter, r *http.Request) {
    // mengoper data ke method get
    if r.Method == "POST" {
        var filepath = path.Join("views", "index.html")
        var tmpl = template.Must(template.New("result").ParseFiles(filepath))

        if err := r.ParseForm(); err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
            return
        }

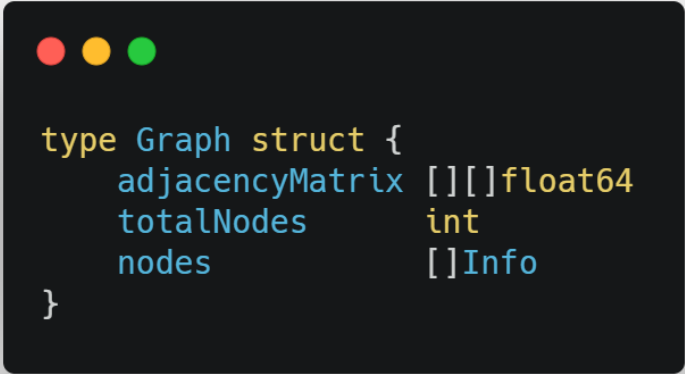
        fileName = r.FormValue("fileName")
        fmt.Println(fileName)
        graf := graph.ReadFile(fileName)
        nodes := graf.GetNodes()

        if err := tmpl.Execute(w, nodes); err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
        }
        return
    }
}

```


Gambar B.2.3 Fungsi index handler yang mengolah data graf awal ketika readfile

3. File graph.go



```
type Graph struct {  
    adjacencyMatrix [][]float64  
    totalNodes      int  
    nodes           []Info  
}
```

Gambar B.3.1 Tipe data graf



```
type Info struct {  
    Latitude float64  
    Longitude float64  
    Name      string  
}
```

Gambar B.3.2 Tipe data untuk menyimpan nodes

```

type Item struct {
    current string
    goal     string
    gn       float64
    fn       float64
    visited  string
    index    int
}

```

Gambar B.3.3 Tipe data bentukan priority queue

```

// menggunakan haversine formula untuk mendapatkan euclidean distance
func GetEuclideanDistance(A Info, B Info) float64 {
    // Coordinate A in radians
    latitude1 := degreesToRadians(A.Latitude)
    longitude1 := degreesToRadians(A.Longitude)

    // Coordinate B in radians
    latitude2 := degreesToRadians(B.Latitude)
    longitude2 := degreesToRadians(B.Longitude)

    // Haversine Formula
    differenceLongitude := longitude2 - longitude1
    differenceLatitude := latitude2 - latitude1

    a := math.Pow(math.Sin(differenceLatitude/2), 2) +
        math.Cos(latitude1)*math.Cos(latitude2)*math.Pow(math.Sin(differenceLongitude/2), 2)
    c := 2 * math.Atan2(math.Sqrt(a), math.Sqrt(1-a))

    // earth radius in KM
    km := c * 6371
    return km
}

```

Gambar B.3.4 Mencari Euclidean Distance dengan Haversine Formula

```

//fungsi Astar
func (graf *Graph) Astar(A string, B string) (float64, string) {
    a, _ := Search(A, B, graf.nodes)
    pq := make(PriorityQueue, 1)
    pq[0] = &Item{
        current: A,
        goal:     B,
        gn:       0,
        fn:       graf.GetDistance(A, B),
        visited: strconv.Itoa(a),
        index:    0,
    }
    heap.Init(&pq)

    // while pq is not empty
    for pq.Len() > 0 {
        now := heap.Pop(&pq).(*Item)
        // apabila node sekarang == tujuan maka return
        if now.current == now.goal {
            return now.fn, now.visited
        }
        charac := string(now.visited[len(now.visited)-1])
        a, _ = strconv.Atoi(charac)
        // mengecek semua tetangga yang belum pernah dikunjungi
        for i := 0; i < graf.totalNodes; i++ {
            if graf.adjacencyMatrix[a][i] > 0 && !isVisited(now.visited, i) {
                updategn := now.gn + graf.adjacencyMatrix[a][i]
                updatehn := graf.GetDistance(graf.nodes[i].Name, now.goal)
                updatefn := updategn + updatehn
                item := &Item{
                    current: graf.nodes[i].Name,
                    goal:     now.goal,
                    gn:       updategn,
                    fn:       updatefn,
                    visited: now.visited + strconv.Itoa(i),
                }
                heap.Push(&pq, item)
            }
        }
    }
    return 0., ""
}

```

Gambar B.3.5 Fungsi A*

Algoritma A* yang kami membuat menggunakan pendekatan bfs dimana setiap tetangga di iterasi dan dimasukkan ke dalam queue. Namun, queue yang digunakan merupakan queue yang dapat menyesuaikan urutan nilai di dalamnya. Disini kami mengurutkan dari terkecil berdasarkan nilai yang dihasilkan $f(n)$ nya.

C. Hasil Pengujian

Test Case 1 Menggunakan File Tes Case 1 (Sekitar ITB)



Simpul awal

"Ciungwanara-Ganesa"

Simpul akhir

"Ganesa-Tamansari"

cari

MAPIKI

Jarak : 0.4004194718350911 KM



Test Case 2 Menggunakan File Tes Case 2 (Sekitar Alun-Alun Bandung)



Simpul awal

"C"

Simpul akhir

"F"

cari

MAPIKI

Jarak : 0.2848235099153762 KM



Test Case 3 Menggunakan File Tes Case 3 (Sekitar Monas)



Simpul awal

"Merdeka Timur"

Simpul akhir

"M.H. Thamrin"

cari

MAPIKI

Jarak : 2.032448055342723 KM



Test Case 4 Menggunakan File Tes Case 4 (Sekitar Denpasar Barat)



Simpul awal

"foursma-rinjani"

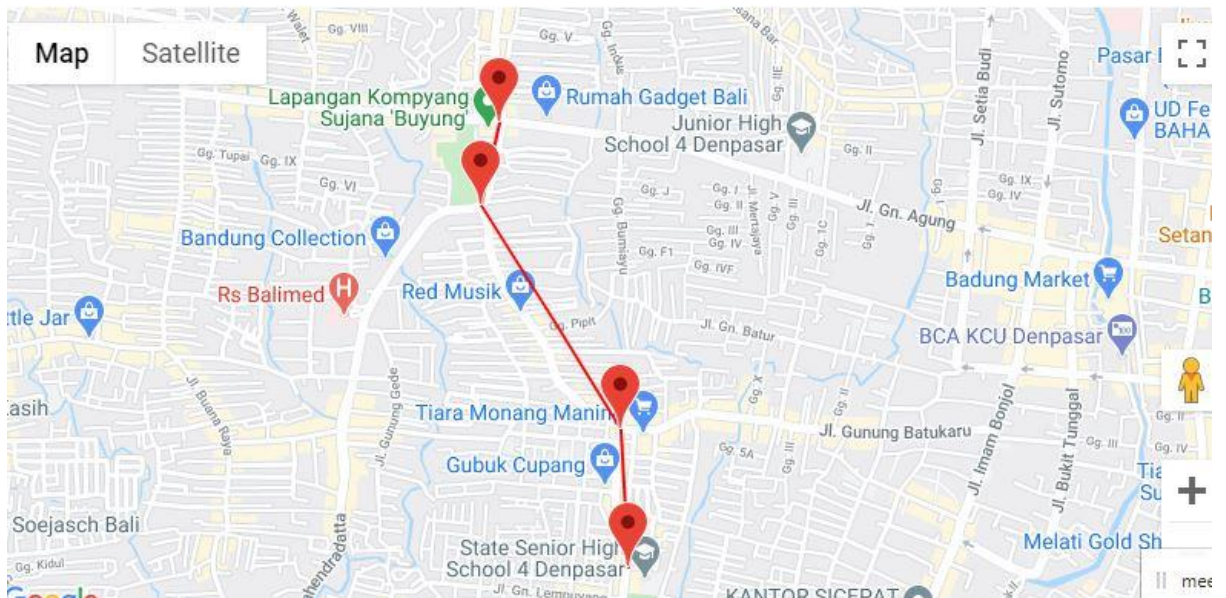
Simpul akhir

"buluh indah-mahendradata-gunung agung"

cari

MAPIKI

Jarak : 1.4958051650776416 KM



Test Case 5 Menggunakan File Tes Case 5 (Sekitar Buah Batu)



Simpul awal

"simpang-klaracondong"



Simpul akhir

"rancabolang"

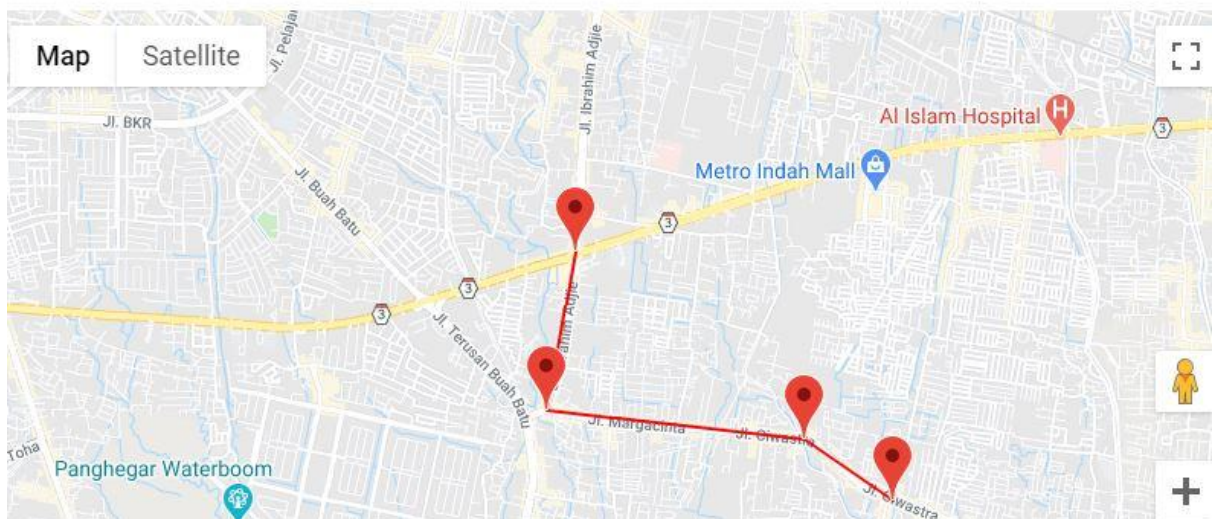


cari

MAPIKI



Jarak : 3.2800885120957277 KM



Test Case 6 Menggunakan File Tes Case 6 (Sekitar Jimbaran)



Simpul awal

"Auditorium"



Simpul akhir

"Uluwatu-II"



cari

MAPIKI



Jarak : 1.5601446863750228 KM



Test Case 7 Menggunakan File Tes Case 3 (Sekitar Monas)



Simpul awal

"Ridwan Rais"



Simpul akhir

"Santa Ursula"



cari

MAPIKI



Tidak Ada Jalur



Test Case 8 Menggunakan File Tes Case 6 (Sekitar Jimbaran)



Simpul awal

"Auditorium"

Simpul akhir

"Auditorium"

cari

MAPIKI

Tidak Ada Jalur



LAMPIRAN

Cara menggunakan aplikasi Ih-Solved

1. Install Golang disini(<https://golang.org/doc/install>)
2. Arahkan direktori kedalam folder proyek ini
3. Buka folder bin
4. Jalankan main.exe
5. Program dapat dijalankan melalui <http://localhost:8080/>

Link Github : https://github.com/DTalone/Tucil3_13519057

Poin	Ya	Tidak
1. Program dapat menerima input graf	√	
2. Program dapat menghitung lintasan terpendek	√	
3. Program dapat menampilkan lintasan terpendek serta jaraknya	√	
4. Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	√	