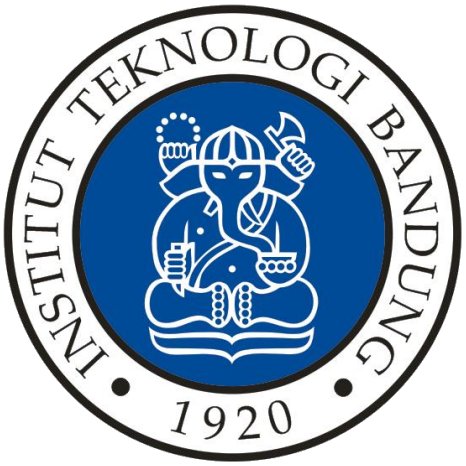
LAPORAN TUGAS KECIL 3
IF2211 - STRATEGI ALGORITMA

Pencarian Jarak Terpendek menggunakan Algoritma A*



Nama	: Joel Triwira
	Josep Marcello
NIM	: 13519073
	13519164
Kelas	: K02
	: K03
Nama Dosen	: Dr. Nur Ulfa Maulidevi
	Prof. Dwi Hendratmo Widyantoro
Dalam Bahasa	: Go
Deadline	: Rabu, 7 April 2021

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

2021

A. Source Code Program

a. graph.go

```go
// Package graph ADT graf dengan representasi adjacency matrix
package graph

import (
   "chizu-ru/node"
   "chizu-ru/prioqueue"
   "errors"
   "fmt"
   //"time"
)

type vertex = node.Node

// Graph ADT graf dengan adjacency matrix
type Graph struct {
   adjMatrix map[*vertex]map[*vertex]bool
}

// New fungsi untuk membuat sebuah graf baru
func New() *Graph {
   g := new(Graph)
   g.adjMatrix = make(map[*vertex]map[*vertex]bool)

   return g
}

// AddVertex fungsi untuk menambahkan sebuah sudut ke graf
// Jika sudut sudah ada, tidak akan dilakukan apa-apa
func (g *Graph) AddVertex(v *vertex) {
   if _, exists := g.adjMatrix[v]; exists {
       return
   }

   g.adjMatrix[v] = make(map[*vertex]bool)
   for k := range g.adjMatrix {
       g.adjMatrix[v][k] = false
       g.adjMatrix[k][v] = false
   }
   g.adjMatrix[v][v] = false
}

// AddEdge fungsi untuk menambahkan sisi dari sudut src ke sudut dst
```

```go
// dengan bobot weight
func (g *Graph) AddEdge(src *vertex, dst *vertex) {
   g.AddVertex(src)
   g.AddVertex(dst)

   g.adjMatrix[src][dst] = true
   g.adjMatrix[dst][src] = true
}

func (g *Graph) Print() {
   for k, v := range g.adjMatrix {
       fmt.Print(*k)
       fmt.Println(":")
       for k2, v2 := range v {
           fmt.Printf("\t%s\t%t\n", k2.GetInfo(), v2)
       }
   }
}

func (g *Graph) AStar(src string, dest string) ([]*vertex, float64, error) {
   // graf baru yg nyimpen jarak spherical antar-sudut
   var srcVert, destVert *vertex
   srcVert = nil
   destVert = nil
   ret := make([]*vertex, 0)

   for k := range g.adjMatrix {
       if k.GetInfo() == src {
           srcVert = k
       } else if k.GetInfo() == dest {
           destVert = k
       }
   }

   if srcVert == nil || destVert == nil {
       return nil, 0, errors.New("EdgeNotFound")
   }

   prevVerts := make(map[*vertex]*vertex)
   vertCostFromSrc := make(map[*vertex]float64)
   toVisit := prioqueue.New()

   vertCostFromSrc[srcVert] = 0
   prevVerts[srcVert] = nil
   toVisit.Enqueue(srcVert, 0)
```

```
// nyari tetangga
var fn, gn, hn float64
fn = 0
gn = 0
hn = 0
var curVert *vertex = nil

for !toVisit.IsEmpty() {
    var err error
    curVert, err = toVisit.Dequeue()
    if err != nil {
        return nil, 0, errors.New("NoConnectionToDest")
    }

    for adj, isAdj := range g.adjMatrix[curVert] {
        if isAdj {
            // hitung gn
            gn = vertCostFromSrc[curVert] + node.Distance(curVert, adj)
            // jadiin gn sbg cost dari source vertex ke k
            if cost, exists := vertCostFromSrc[adj]; !exists {
                vertCostFromSrc[adj] = gn
                prevVerts[adj] = curVert
            } else {
                if gn < cost {
                    // update jarak dari source, kalo ternyata bisa lebih
                    // kecil. Update juga parent vertex-nya
                    vertCostFromSrc[adj] = gn
                    prevVerts[adj] = curVert
                }
            }
            // kalo ketemu deestiation vertex, stop
            if adj == destVert {
                toVisit.Clear()
                break
            }
            // hitung hn
            hn = node.Distance(adj, destVert)
            // hitung fn
            fn = gn + hn

            // ga usah dimasukin kalo udah ada di queue dengan cost lebih
            // kecil atau jarak dari src lebih kecil dari gn yang dihitung
            if toVisit.ContainsLEQ(adj, fn) || gn > vertCostFromSrc[adj] {
                continue
```

```
                }

                toVisit.RemoveEl(adj)
                toVisit.Enqueue(adj, fn)
            }
        }
    }

    curVert = destVert
    for curVert != nil {
        prevVert := prevVerts[curVert]
        ret = append(ret, curVert)
        curVert = prevVert
    }

    // reverse result graph
    for i, j := 0, len(ret)-1; i < j; i, j = i+1, j-1 {
        ret[i], ret[j] = ret[j], ret[i]
    }

    return ret, vertCostFromSrc[destVert], nil
}
```

b. handler.go

```
package handler

import (
    "chizu-ru/graph"
    "chizu-ru/parser"
    "encoding/json"
    "html/template"
    "io"
    "log"
    "net/http"
    "net/url"
    "path"
)

type aStarRes struct {
    Distance float64
    Nodes    []resNode
}

type resNode struct {
    Name      string
```

```
    Latitude  float64
    Longitude float64
}

func HomeHandler(w http.ResponseWriter, r *http.Request) {
    log.Println("/ hit")
    if r.URL.Path != "/" {
        http.NotFound(w, r)
        return
    }
    tmpl, err := template.ParseFiles(path.Join("views", "Home.html"))
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    err = tmpl.Execute(w, nil)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
}

// AStar format query: AStar?filePath=[filePath]&src=[src]&dest=[dest]
func AStar(w http.ResponseWriter, r *http.Request) {
    log.Println("/AStar endpoint hit")
    u := r.URL
    m, _ := url.ParseQuery(u.RawQuery)

    if len(m["src"]) == 0 || len(m["dest"]) == 0 {
        http.Error(w, "src and dest not set. hint: /Astar?dest=Arad&src=Bucharest",
http.StatusBadRequest)
        return
    }

    var g *graph.Graph
    var err error
    if len(m["filePath"]) != 0 || r.Method != "POST" {
        if len(m["filePath"]) == 0 {
            http.Error(w, "filePath query not set", http.StatusBadRequest)
        }
        g, err = parser.ParseFile(m["filePath"][0])
        if err != nil {
            http.Error(w, err.Error(), http.StatusBadRequest)
            return
        }
```

```
    } else { // parse request body. Request body is the file's content
        var content []byte
        content, err := io.ReadAll(r.Body)
        if err != nil {
            http.Error(w, err.Error(), http.StatusBadRequest)
            return
        }
        g, err = parser.Parse(string(content))
        if err != nil {
            http.Error(w, err.Error(), http.StatusBadRequest)
            return
        }
    }

    nodes, jarak, _ := g.AStar(m["src"][0], m["dest"][0])
    res := new(aStarRes)
    for i := len(nodes) - 1; i > -1; i-- {
        res.Nodes = append(res.Nodes,
            resNode{Name: nodes[i].GetInfo(),
                Latitude:  nodes[i].GetLat(),
                Longitude: nodes[i].GetLong()},
        )
    }
    res.Distance = jarak
    jsRes, err := json.Marshal(res)
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }
    w.Header().Set("Content-Type", "application/json")
    w.Write(jsRes)

    log.Println("Finish counting distance.")
    log.Println(res)
}
```

c. node.go

```
// Package node ADT vertex untuk graf
package node

import (
    "math"
)

type gcs struct {
```

```
    latitude  float64 // N, satuan derajat
    longitude float64 // W, satuan derajat
}

// Node ADT untuk sudut graf
// gcs adalah global coordinate system yang menandakan longitude dan latitude
// sudut dengan satuan derajat
type Node struct {
    info string
    pos  gcs
}

// New fungsi untuk membuat sebuah vertex baru dengan
func New(info string, latitude float64, longitude float64) *Node {
    node := new(Node)
    node.info = info
    node.pos.longitude = longitude
    node.pos.latitude = latitude

    return node
}

func toRadian(deg float64) float64 {
    return deg * math.Pi / 180
}

func haversine(rad float64) float64 {
    return math.Pow(math.Sin(rad/2), 2)
}

// Distance fungsi untuk menghitung jarak antara dua buah sudut dengan
// menggunakan persamaan great-sphecial distance. Hasilnya dalam kilometer
func Distance(v1 *Node, v2 *Node) float64 {
    r := 6371.0 // jari-jari bumi
    v1lat := toRadian(v1.pos.latitude)
    v1lon := toRadian(v1.pos.longitude)
    v2lat := toRadian(v2.pos.latitude)
    v2lon := toRadian(v2.pos.longitude)

    h := haversine(v2lat-v1lat) +
        math.Cos(v1lat)*math.Cos(v2lat)*haversine(v2lon-v1lon)

    return 2 * r * math.Asin(math.Sqrt(h))
}
```

```
func (v *Node) GetInfo() string {
   return v.info
}

func (v *Node) GetLat() float64 {
   return v.pos.latitude
}

func (v *Node) GetLong() float64 {
   return v.pos.longitude
}
```

d. osm.go

```
package osm

import (
   "bufio"
   "bytes"
   "encoding/json"
   "fmt"
   "net/http"
   "net/url"
)

type OverpassResult struct {
   tipe string
   id   int64
   lat  float64
   lon  float64
}

func Search(searchType string, searchQuery string, searchArea string, timeout uint)
{
   link := "https://overpass.kumi.systems/api/interpreter?data="
   queryBuf := new(bytes.Buffer)
   _, err := fmt.Fprintf(queryBuf,
       `[out:json][timeout:%d];
              area[name="%s"]->.searchArea;
              (
                  node[%s="%s"](area.searchArea);
                  way[%s="%s"](area.searchArea);
                  relation[%s="%s"](area.searchArea);
              );
              out;
              >;
```

```
                out skel qt;`,
        timeout, searchArea, searchType, searchQuery, searchType,
        searchQuery, searchType, searchQuery)
    query := url.QueryEscape(queryBuf.String())
    if err != nil {
        panic(err)
    }
    resp, err := http.Get(link + query)
    if err != nil {
        panic(err)
    }
    defer resp.Body.Close()

    fmt.Println(resp.Status)
    scanner := bufio.NewScanner(resp.Body)
    for scanner.Scan() {
        fmt.Println(scanner.Text())
    }

    if err := scanner.Err(); err != nil {
        panic(err)
    }
    fmt.Println(link + query)
    fmt.Println(queryBuf.String())
}

func SearchIntersectionInArea(areaName string, timeout uint, target interface{})
error {
    link := "https://overpass.kumi.systems/api/interpreter?data="
    queryBuf := new(bytes.Buffer)
    _, err := fmt.Fprintf(queryBuf, `[out:json][timeout:%d];
area[name~".*%s.*",i]->.searchArea;
way[highway~"^(motorway|trunk|primary|secondary|tertiary|(motorway|trunk|primary|se
condary)_link)$"](area.searchArea)->.major;
way[highway~"^(motorway|trunk|primary|secondary|tertiary|(motorway|trunk|primary|se
condary)_link)$"](area.searchArea)->.minor;
node(w.major)(w.minor);
out;`,
        timeout, areaName)
    query := url.QueryEscape(queryBuf.String())
    if err != nil {
        return err
    }
    resp, err := http.Get(link + query)
    if err != nil {
```

```
        return err
    }
    defer resp.Body.Close()

    fmt.Println(resp.Status)
    //fmt.Println(link + query)
    //fmt.Println(queryBuf.String())

    return json.NewDecoder(resp.Body).Decode(target)
}
```

e. parser.go

```
// Package parser untuk parsing file menjadi graf
// format file:
// baris pertama adalah banyak sudut, misalnya N sudut
// baris kedua sampai N + 1 akan berisi informasi sudut (nama, latitude, longitude)
// baris N + 2 sampai akhir akan berisi matriks berukuran N x N yang berisi
// matriks ketetanggaan antarsudut
// misalnya, jika ada 2 sudut Dago dan ITB dengan jarak dan koordinat arbitrary:
// ```
// 2
// Dago 0 0
// ITB 69 420
// 0 10
// 10 10
// ```
package parser

import (
    "chizu-ru/graph"
    "chizu-ru/node"
    "io"
    "os"
    "strconv"
    "strings"
)

type vertex = node.Node

// Parse fungsi untuk parsing file menjadi graf kemudian pointer graf akan
// dikembalikan
func Parse(fileContent string) (*graph.Graph, error) {
    f := strings.Split(fileContent, "\n")

    // buat nyimpen sudut yang udah dibuat, jadi tinggal tambah ke graf
```

```
    nVertex, _ := strconv.Atoi(f[0])
    vertices := make([]*vertex, nVertex)

    ctr := 1
    for ctr <= nVertex {
        splitted := strings.Split(f[ctr], ",")
        latitude, _ := strconv.ParseFloat(splitted[1], 64)
        longitude, _ := strconv.ParseFloat(splitted[2], 64)

        vertices[ctr-1] = node.New(splitted[0], latitude, longitude)
        ctr++
    }

    g := graph.New()
    for i := 0; i < nVertex; i++ {
        splitted := strings.Split(f[ctr], " ")
        for j := i; j < nVertex; j++ {
            if splitted[j] == "0" {
                continue
            } else {
                g.AddEdge(vertices[i], vertices[j])
            }
        }
        ctr++
    }

    return g, nil
}

func ParseFile(pathToFile string) (*graph.Graph, error) {
    file, ferr := os.Open(pathToFile)
    if ferr != nil {
        return nil, ferr
    }

    content, err := io.ReadAll(file)
    if err != nil {
        return nil, err
    }

    return Parse(string(content))
}
```

f. prioqueue.go

```
// Package prioqueue lirary priority queue dengan elementnya adalah pointer
```

```go
// sudut pada graf. Bisa ada beberapa elemen yang sama dengan priority berbeda.
package prioqueue

import (
   "chizu-ru/node"
   "errors"
)

type queueElement struct {
   info     *node.Node
   priority float64
}

type PrioQueue struct {
   elements []*queueElement
}

func New() *PrioQueue {
   pq := new(PrioQueue)
   pq.elements = make([]*queueElement, 0)

   return pq
}

func (pq *PrioQueue) Enqueue(n *node.Node, prio float64) {
   a := new(queueElement)
   a.info = n
   a.priority = prio

   if pq.IsEmpty() {
       pq.elements = append(pq.elements, a)
       return
   }

   for i, el := range pq.elements {
       if el.priority > prio {
           temp := make([]*queueElement, 0)
           temp = append(temp, pq.elements[:i]...)
           temp = append(temp, a)
           pq.elements = append(temp, pq.elements[i:]...)
           return
       }
   }

   pq.elements = append(pq.elements, a)
```

```
}

func (pq *PrioQueue) Dequeue() (*node.Node, error) {
   if pq.IsEmpty() {
       return nil, errors.New("queue kosong")
   }

   temp := pq.elements
   pq.elements = pq.elements[1:]
   return temp[0].info, nil
}

func (pq *PrioQueue) IsEmpty() bool {
   return len(pq.elements) == 0
}

// ContainsLEQ memeriksa apakah di queue sudah ada elemen yang sama dengan
// priority lebih rendah atau sama
func (pq *PrioQueue) ContainsLEQ(n *node.Node, prio float64) bool {
   for _, e := range pq.elements {
       if e.info == n && e.priority <= prio {
           return true
       }
   }

   return false
}

// RemoveEl menghapus elemen n dari queue
func (pq *PrioQueue) RemoveEl(n *node.Node) {
   tmp := make([]*queueElement, 0)
   for i, e := range pq.elements {
       if e.info == n {
           tmp = append(tmp, pq.elements[:i]...)
           tmp = append(tmp, pq.elements[i+1:]...)
           pq.elements = tmp
       }
   }
}

// Clear mengosongkan queue
func (pq *PrioQueue) Clear() {
   for !pq.IsEmpty() {
       pq.Dequeue()
   }
```

```
}
```

g. chizu-ru.go

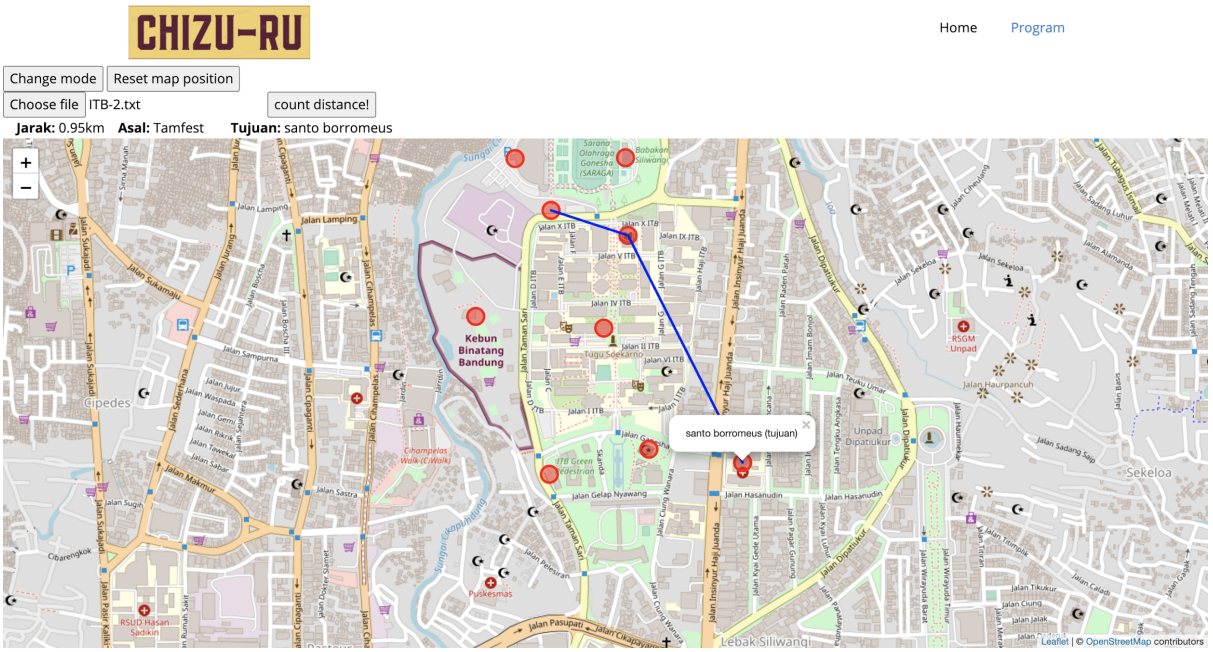
```go
package main

import (
    "chizu-ru/handler"
    "log"
    "net/http"
    "github.com/rs/cors"
)

func main() {
    mux := http.NewServeMux()
    fs := http.FileServer(http.Dir("./views"))
    mux.Handle("/", http.StripPrefix("/", fs))
    mux.HandleFunc("/AStar", handler.AStar)

    handler := cors.Default().Handler(mux)
    log.Println("Starting web port on 8080")
    werr := http.ListenAndServe(":8080", handler)
    log.Fatal(werr)
}
```

C. Peta/Graf Input

ITB-2.txt

```
9
sabuga,-6.88625797794717,107.6078338176012
masjid salman,-6.893662002104935,107.61126905679703
santo borromeus,-6.893994854278154,107.6136776804924
labtek V,-6.890567797116448,107.6101103425026
Tamfest,-6.887580086045596,107.60875850915909
saraga swimming pool,-6.88623268077744,107.61067360639572
BNI Tamansari,-6.894309066515138,107.60873302817345
Bonbin,-6.890285536447924,107.60683804750443
Perpustakaan ITB,-6.888205855965436,107.61073529720306
0 0 0 1 0 0 1 0 1 1
0 0 0 1 0 0 1 1 0 1
0 0 0 1 0 1 0 1 1 1
1 1 1 0 0 1 1 0 1 0
0 0 0 0 0 1 0 1 1 1
0 0 1 1 1 0 1 0 1 0
1 1 0 1 0 1 0 0 1 0
0 1 1 0 1 0 0 0 0 1
1 0 1 1 1 1 1 0 0 0
1 1 1 0 1 0 0 1 0 0
```

Contoh gambar input graf



Contoh gambar peta dalam program

D. Alamat kode

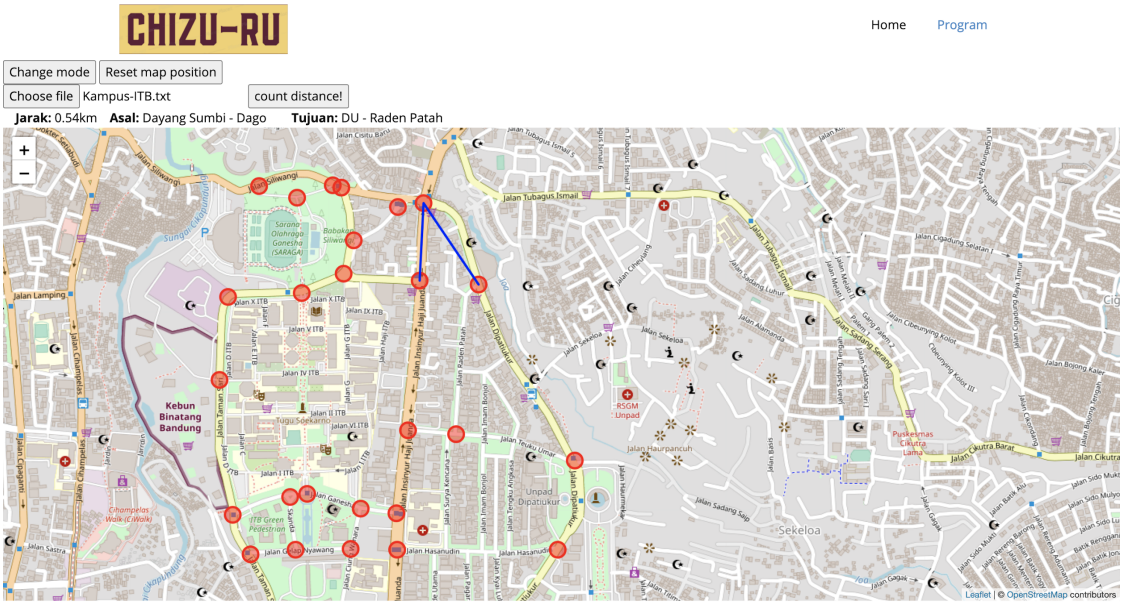
https://github.com/jspmarc/chizu-ru

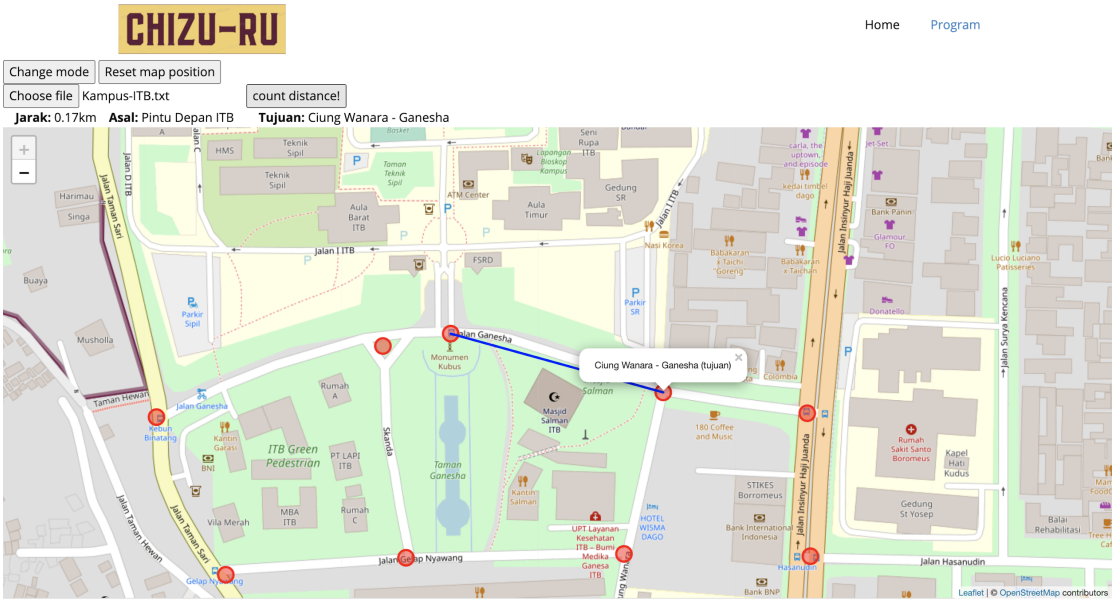
E. Test Case

1. Kampus-ITB.txt

   28
   Skywalk,-6.885112680505222,107.61019953300554
   Siliwangi,-6.884804853041526,107.60914368104865
   Siliwangi - Skywalk,-6.884783550096599,107.61119378599209
   Taman Sari - Siliwangi,-6.884836807457129,107.6114190210842
   Sumur Bandung - Siliwangi,-6.885369380733616,107.61299685054328
   Taman Sari - Sumur Bandung,-6.886285405370851,107.61177322769747
   Dayang Sumbi - Taman Sari,-6.887212079657689,107.61149415582035
   Dayang Sumbi - Dago,-6.887393153732213,107.61359792843244
   Dago - DU,-6.8852628661261415,107.6137052637698
   Pintu Belakang ITB,-6.8877467805165,107.61033651729362
   Tamfes,-6.887851164318019,107.60829556276724
   Pintu Samping ITB,-6.890130559259928,107.60805942502509
   Ganesha - Taman Sari,-6.893826564216093,107.60842402693942
   Taman Sari - Gelap Nyawang,-6.894912992468297,107.60890754800417
   Pintu Depan ITB,-6.893250330197495,107.61047343867449
   Ganesha - Skanda,-6.8933366055619265,107.61000236844517
   Skanda - Gelap Nyawang,-6.8947958287570135,107.61016337145116
   Ciung Wanara - Gelap Nyawang,-6.8947681355281825,107.61168076489075
   Ciung Wanara - Ganesha,-6.893655078760995,107.61195442549976
   Ganesha - Dago,-6.893796740627806,107.6129552928273
   Dago - Hasanudin,-6.894785177509091,107.6129755572898
   Taman Sari - Pasupati,-6.898246820474195,107.60954082649454

Pasupati - Dago,-6.898960448340111,107.61283602135124
Hasanudin - DU,-6.894785177509091,107.61740850672243
DU - Teuku Umar,-6.892346035431296,107.6178807822068
Dago - Teuku Umar,-6.891515233080349,107.61326536270067
Teuku Umar - Raden Patah,-6.891621746283635,107.61460487317066
DU - Raden Patah,-6.8874996678613964,107.6152295993742
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0





2. Pademangan.txt

7
Rumah Joel,-6.140365,106.844506
2 - Gg 14 - 1,-6.140126,106.844097
1 - Gg 14 - 2,-6.140727,106.845149
3 - Gg 14 - 2,-6.139000,106.842031
Pademangan 2,-6.143692,106.842134
Pademangan 1,-6.144311,106.843175
Pademangan 3,-6.142492,106.840004
0 1 1 0 0 0 0
1 0 0 1 1 0 0
1 0 0 0 0 1 0
0 1 0 0 0 0 1
0 1 0 0 0 1 1
0 0 1 0 1 0 0

0 0 0 1 1 0 0

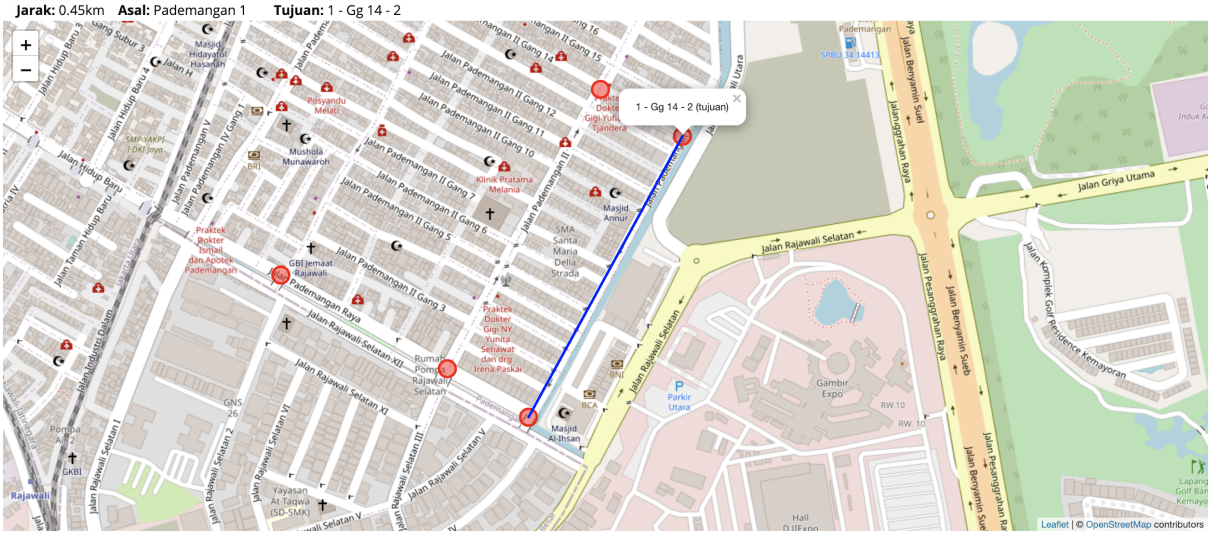
CHIZU-RU

Home Program

Change mode | Reset map position

Choose file | Pademangan.txt | count distance!

Jarak: 0.45km **Asal:** Pademangan 1 **Tujuan:** 1 - Gg 14 - 2



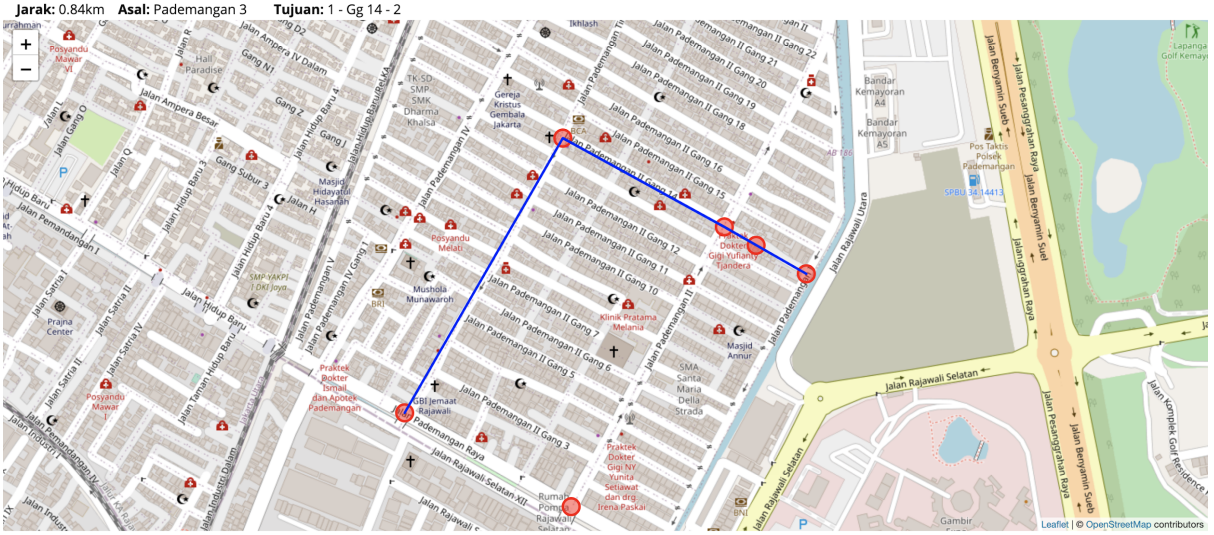
CHIZU-RU

Home Program

Change mode | Reset map position

Choose file | Pademangan.txt | count distance!

Jarak: 0.84km **Asal:** Pademangan 3 **Tujuan:** 1 - Gg 14 - 2



F. Cuplikan Layar Hasil Input dan Output

Centang (v) jika ya

1	Program dapat menerima input graf	v
2	Program dapat menghitung lintasan terpendek	v
3	Program dapat menampilkan lintasan terpendek serta jaraknya	v
4	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	x