IF2211 - Strategi Algoritma
Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek
Laporan Tugas Kecil 3

Nama	NIM
Moch. Sofyan Firdaus	13521083
Nicholas Liem	13521135

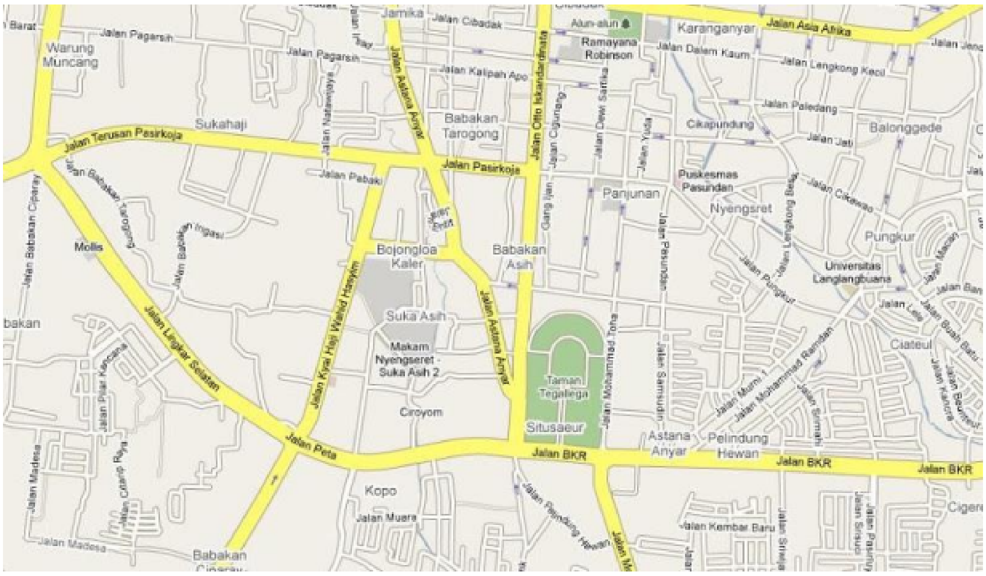
Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Tahun Ajaran 2022/2023

Daftar Isi

1 Deskripsi Masalah

Algoritma UCS (Uniform cost search) dan A* (atau A star) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Pada tugas kecil 3 ini, anda diminta menentukan lintasan terpendek berdasarkan peta Google Map jalan-jalan di kota Bandung. Dari ruas-ruas jalan di peta dibentuk graf. Simpul menyatakan persilangan jalan (simpang 3, 4 atau 5) atau ujung jalan. Asumsikan jalan dapat dilalui dari dua arah. Bobot graf menyatakan jarak (m atau km) antar simpul. Jarak antar dua simpul dapat dihitung dari koordinat kedua simpul menggunakan rumus jarak Euclidean (berdasarkan koordinat) atau dapat menggunakan ruler di Google Map, atau cara lainnya yang disediakan oleh Google Map.

Langkah pertama di dalam program ini adalah membuat graf yang merepresentasikan peta (di area tertentu, misalnya di sekitar Bandung Utara/Dago). Berdasarkan graf yang dibentuk, lalu program menerima input simpul asal dan simpul tujuan, lalu menentukan lintasan terpendek antara keduanya menggunakan algoritma UCS dan A*. Lintasan terpendek dapat ditampilkan pada peta/graf (misalnya jalan-jalan yang menyatakan lintasan terpendek diberi warna merah). Nilai heuristik yang dipakai adalah jarak garis lurus dari suatu titik ke tujuan.



Spesifikasi program:

1. Program menerima input file graf (direpresentasikan sebagai matriks ketetanggaan berbobot), jumlah simpul minimal 8 buah
2. Program dapat menampilkan peta/graf
3. Program menerima input simpul asal dan simpul tujuan
4. Program dapat menampilkan lintasan terpendek besertta jaraknya antara simpul asal dan simpul tujuan.
5. Antarmuka program bebas, apakah pakai GUI atau command line saja.

2 Kode Program

2.1 Struktur File dan Folder

```
Tucil3_13521083_13521135/
|-- bin
|-- doc
‘-- src/
    |-- handlers/
    |   |-- parse.go
    |   ‘-- search.go
    |-- models/
    |   |-- a_star.go
    |   |-- adjacency_matrix.go
    |   |-- graph.go
    |   |-- priority_queue.go
    |   ‘-- ucs.go
    |-- static/
    |   |-- js/
    |   |   ‘-- script.js
    |   |-- index.html
    |   ‘-- style.css
    |-- templates/
    |   |-- calculate.html
    |   ‘-- home.html
    |-- test/
    |   |-- a_star_test.go
    |   |-- graph_test.go
    |   |-- prioqueue_test.go
    |   |-- ucs_test.go
    |   |-- AlunAlunBandung.txt
    |   |-- BuahBatu.txt
    |   |-- Jakarta.txt
    |   |-- SekitarLabtek.txt
    |   |-- tc1.txt
    |   ‘-- tc2.txt
    |-- utils/
    |   ‘-- utils.go
    |-- .gitignore
    |-- go.mod
    |-- main.go
    |-- LICENSE
    ‘-- README.md
```

2.2 Struktur Data

2.2.1 Graph

```
package models

type Node struct {
  Index     int
```

```go
  Name      string
  Latitude  float64
  Longitude float64
}

type Graph struct {
  Nodes map[int]*Node
  Edges map[int]map[int]float64
}

func (g *Graph) AddNode(index int, name string, latitude, longitude float64) {
  if g.Nodes == nil {
    g.Nodes = make(map[int]*Node)
  }
  newNode := Node{Index: index, Name: name, Latitude: latitude, Longitude: longitude}
  g.Nodes[index] = &newNode
}

func (g *Graph) AddEdge(fromIndex, toIndex int, weight float64) {
  if g.Edges == nil {
    g.Edges = make(map[int]map[int]float64)
  }
  if _, ok := g.Edges[toIndex]; !ok {
    g.Edges[toIndex] = make(map[int]float64)
  }
  g.Edges[toIndex][fromIndex] = weight
}

func NewGraphFromAdjacencyMatrix(am AdjacencyMatrix) *Graph {
  g := &Graph{
    Nodes: make(map[int]*Node),
    Edges: make(map[int]map[int]float64),
  }
  for i := 0; i < am.NodesCount; i++ {
    g.AddNode(i, am.ColumnLabels[i], am.Latitudes[i], am.Longitudes[i])
  }

  for i := 0; i < am.NodesCount; i++ {
    for j := 0; j < i; j++ {
      if am.Matrix[i][j] == 0 {
        continue
      } else {
        g.AddEdge(i, j, am.Matrix[i][j])
        g.AddEdge(j, i, am.Matrix[i][j])
      }
    }
  }
  return g
}

func (g *Graph) GetEdgeWeight(indexSource, indexDestination int) float64 {
  if _, ok := g.Edges[indexSource]; !ok {
    return -1
  }
  if weight, ok := g.Edges[indexSource][indexDestination]; ok {
    return weight
  }
  return -1
}

func (g *Graph) GetNeighbors(index int) []int {
  if _, ok := g.Nodes[index]; !ok {
    return []int{}
  }
  var neighbors []int
  if edges, ok := g.Edges[index]; ok {
    for nodeIndex := range edges {
      neighbors = append(neighbors, nodeIndex)
    }
  }
  return neighbors
}
```

2.2.2 Adjacency Matrix

```
package models

import (
  "errors"
  "fmt"
  "github.com/NicholasLiem/Tucil3_13521083_13521135/utils"
  "os"
  "strconv"
  "strings"
)

type AdjacencyMatrix struct {
  Matrix       [][]float64
  ColumnLabels []string
  Latitudes    []float64
  Longitudes   []float64
  NodesCount   int
}

func (am AdjacencyMatrix) GetNodesCount() int {
  return am.NodesCount
}

func NewAdjacencyMatrix(nodeCount int, columnLabels []string,
    latitudes, longitudes []float64) AdjacencyMatrix {
  matrix := make([][]float64, nodeCount)
  for i := range matrix {
    matrix[i] = make([]float64, nodeCount)
  }

```

```go
  return AdjacencyMatrix{
    Matrix:       matrix,
    ColumnLabels: columnLabels,
    NodesCount:   nodeCount,
    Latitudes:    latitudes,
    Longitudes:   longitudes,
  }
}

func ParseToAdjacencyMatrix(buf string) (AdjacencyMatrix, error) {
  var res AdjacencyMatrix
  lines := strings.Split(buf, "\n")
  count, err := strconv.ParseInt(strings.TrimSpace(lines[0]), 10, 32)
  if err != nil {
    return res, err
  }
  columnLabels := make([]string, count)
  latitudes := make([]float64, count)
  longitudes := make([]float64, count)
  for i := 0; i < len(columnLabels); i++ {
    label, latitude, longitude, err := utils.ParseNode(lines[i+1])
    if err != nil {
      return res, err
    }
    columnLabels[i] = label
    latitudes[i] = latitude
    longitudes[i] = longitude
  }
  matrix := make([][]float64, count)
  for i := 0; i < len(matrix); i++ {
    matrix[i], err = utils.ParseRow(lines[i+int(count)+1], int(count))
    if err != nil {
      return res, err
    }
  }
  res = NewAdjacencyMatrix(int(count), columnLabels, latitudes, longitudes)
  res.Matrix = matrix
  return res, nil
}

func AdjacencyMatrixFromFile(filepath string) (AdjacencyMatrix, error) {
  var res AdjacencyMatrix
  buf, err := os.ReadFile(filepath)
  if err != nil {
    msg := fmt.Sprintf("[ERROR] cannot read file %s (%s)", filepath, err.Error())
    return res, errors.New(msg)
  }
  lines := strings.Split(string(buf), "\n")
  count, err := strconv.ParseInt(strings.TrimSpace(lines[0]), 10, 32)
  if err != nil {
    msg := fmt.Sprintf("[ERROR] cannot parse node count (%s) at %s:1", err.Error(), filepath)
    return res, errors.New(msg)
  }
  columnLabels := make([]string, count)
  latitudes := make([]float64, count)
  longitudes := make([]float64, count)
  for i := 0; i < len(columnLabels); i++ {
    label, latitude, longitude, ok := utils.ParseNode(lines[i+1])
    if ok != nil {
      msg := fmt.Sprintf("[ERROR] %s at %s:%d", ok.Error(), filepath, i+1)
      return res, errors.New(msg)
    }
    columnLabels[i] = label
    latitudes[i] = latitude
    longitudes[i] = longitude
  }
  matrix := make([][]float64, count)
  for i := 0; i < len(matrix); i++ {
    matrix[i], err = utils.ParseRow(lines[i+int(count)+1], int(count))
    if err != nil {
      msg := fmt.Sprintf("[ERROR] %s at %s:%d", err.Error(), filepath, i+int(count)+1)
      return res, errors.New(msg)
    }
  }
  res = NewAdjacencyMatrix(int(count), columnLabels, latitudes, longitudes)
  res.Matrix = matrix
  return res, nil
}
```

2.3 Utilities

```
package utils

import (
  "fmt"
  "math"
  "strconv"
  "strings"
)

func ParseNode(line string) (string, float64, float64, error) {
  words := strings.Split(line, ",")
  name := strings.TrimSpace(words[0])

  latitude, err := strconv.ParseFloat(strings.TrimSpace(words[1]), 64)
  if err != nil {
    return "", 0, 0, fmt.Errorf("cannot parse latitude (%w)", err)
  }

  longitude, err := strconv.ParseFloat(strings.TrimSpace(words[2]), 64)
  if err != nil {
    return "", 0, 0, fmt.Errorf("cannot parse longitude (%w)", err)
  }

  return name, latitude, longitude, nil
}

func ParseRow(line string, columns int) ([]float64, error) {
  words := strings.Fields(line)
  if len(words) != columns {
    return nil, fmt.Errorf("invalid number of columns (%d), expected %d", len(words), columns)
  }

  row := make([]float64, columns)
  for i, word := range words {
    val, err := strconv.ParseFloat(strings.Trim(strings.TrimSpace(word), "\x00"), 64)
    if err != nil {
      return nil, fmt.Errorf("cannot parse column %d (%w)", i+1, err)
    }
    row[i] = val
  }

  return row, nil
}

func Distance(lat1, lon1, lat2, lon2, weightRange float64) float64 {
  deltaX := lon2 - lon1
  deltaY := lat2 - lat1
  return math.Sqrt(deltaX*deltaX+deltaY*deltaY) * weightRange
}
```

2.4 UCS Algorithm

```
package models

type ucsnode struct {
  nodeIndex int
  g         float64
  trace     []int
}

func UniformCostSearch(graph Graph, src, dest int) ([]int, float64) {
```

```go
  pq := NewPriorityQueue(func(value ucsnode) float64 {
    return -value.g
  })

  pq.Enqueue(ucsnode{nodeIndex: src, g: 0, trace: []int{src}})

  visited := make(map[int]bool)

  for !pq.IsEmpty() {
    curr := pq.Dequeue()
    if visited[curr.nodeIndex] {
      continue
    }
    visited[curr.nodeIndex] = true

    if curr.nodeIndex == dest {
      return curr.trace, curr.g
    }

    for neighbour, distance := range graph.Edges[curr.nodeIndex] {
      if visited[neighbour] {
        continue
      }
      next := ucsnode{
        nodeIndex: neighbour,
        g:         curr.g + distance,
        trace:     append(append([]int{}, curr.trace...), neighbour),
      }
      pq.Enqueue(next)
    }
  }
  return []int{}, 0
}
```

```go
package models

type ucsnode struct {
  nodeIndex int
  g         float64
  trace     []int
}

func UniformCostSearch(graph Graph, src, dest int) ([]int, float64) {
  pq := NewPriorityQueue(func(value ucsnode) float64 {
    return -value.g
  })

  pq.Enqueue(ucsnode{nodeIndex: src, g: 0, trace: []int{src}})

  visited := make(map[int]bool)

  for !pq.IsEmpty() {
    curr := pq.Dequeue()
    if visited[curr.nodeIndex] {
      continue
    }
    visited[curr.nodeIndex] = true

    if curr.nodeIndex == dest {
      return curr.trace, curr.g
    }

    for neighbour, distance := range graph.Edges[curr.nodeIndex] {
      if visited[neighbour] {
        continue
      }
      next := ucsnode{
        nodeIndex: neighbour,
        g:         curr.g + distance,
        trace:     append(append([]int{}, curr.trace...), neighbour),
      }
      pq.Enqueue(next)
    }
  }
  return []int{}, 0
}
```

2.5 A* Algorithm

```
package models
import (
  "github.com/NicholasLiem/Tucil3_13521083_13521135/utils"
  "math"
)
type astarnode struct {
  nodeIndex int
  f         float64
  g         float64
  h         float64
  trace     []int
}

func calculateH(graph Graph, current, destination int, weightRange float64) float64 {
  node1 := graph.Nodes[current]
  node2 := graph.Nodes[destination]
  lat1 := node1.Latitude
  lat2 := node2.Latitude
  lon1 := node1.Longitude
  lon2 := node2.Longitude

  return utils.Distance(lat1, lon1, lat2, lon2, weightRange)
}
```

```go

func getWeightRange(graph Graph) float64 {
  min := math.MaxFloat64
  max := 0.0
  for from := range graph.Edges {
    for _, weight := range graph.Edges[from] {
      if weight < min {
        min = weight
      } else if weight > max {
        max = weight
      }
    }
  }
  return max - min
}

func AStarSearch(graph Graph, src, dest int) ([]int, float64) {
  pq := NewPriorityQueue(func(value astarnode) float64 {
    return -value.f
  })
  weighRange := getWeightRange(graph)

  pq.Enqueue(astarnode{nodeIndex: src, f: 0, g: 0, trace: []int{src}})

  visited := map[int]bool{}

  for !pq.IsEmpty() {
    curr := pq.Dequeue()
    if visited[curr.nodeIndex] {
      continue
    }
    if curr.nodeIndex == dest {
      return curr.trace, curr.g
    }

    for neighbour, distance := range graph.Edges[curr.nodeIndex] {
      if visited[neighbour] {
        continue
      }
      x := astarnode{
        nodeIndex: neighbour,
        g:         curr.g + distance,
        h:         calculateH(graph, curr.nodeIndex, dest, weighRange),
        trace:     append(append([]int{}, curr.trace...), neighbour),
      }
      x.f = x.g + x.h
      pq.Enqueue(x)
    }
  }
  return []int{}, 0
}
```

3 Input Output Program

3.1 File Test Case

Penjelasan singkat file *test case* adalah sebagai berikut. Baris pertama merupakan jumlah simpul yang akan dimasukkan, kemudian diisi dengan nama simpul serta letak *longitude* dan *latitude*nya. Kemudian, di bawahnya diisi dengan matriks *adjacency* yang berisi jarak dari simpul n ke simpul m, berurutan dari simpul yang pertamakali dipanggil sampai paling terakhir.

3.1.1 Peta Jalan Sekitar Kampus ITB/Dago/Bandung Utara

```
13
Kubus, -6.893311689500024, 107.61054852083436
Titik Tengah, -6.890915377731855, 107.61034091465416
Parkiran Labtek V, -6.891023877300416, 107.60945879583674
GKUB, -6.890455545882821, 107.60872239575617
Intel, -6.89036884274947, 107.61043670639772
Gedung LFM, -6.891156756723175, 107.61162847754282
Kantin Bengkok, -6.889862510450611, 107.61152439273246
Gedung FTMD, -6.889898445152209, 107.6086521160611
Labtek V, -6.89060627683087, 107.6097059748216
Labtek VI, -6.890066361526841, 107.60963831969185
Labtek VII, -6.890071528190981, 107.6107962632752
Labtek VIII, -6.890500360487576, 107.61083529508082
Pusat Gema, -6.889841034853903, 107.6103502670937
0 254.11 0 0 0 0 0 0 0 0 0 0 0
254.11 0 100.6 0 66.49 133.73 0 0 0 0 56.75 55.57 0
0 100.6 0 95.23 0 0 0 0 42.27 0 0 0 0
0 0 95.23 0 0 0 0 62.75 96.84 96.02 0 0 0
0 66.49 0 0 0 0 0 0 89.42 88.77 63.34 44.98 55.69
0 133.73 0 0 0 0 124.66 0 0 0 0 104.23 0
0 0 0 0 0 124.66 0 0 0 0 82.97 0 133.54
0 0 0 62.75 0 0 0 0 0 80.83 0 0 169.74
0 0 42.27 96.84 89.42 0 0 0 0 42.05 0 0 0
0 0 0 96.02 88.77 0 0 80.83 42.05 0 0 0 102.4
0 56.75 0 0 63.34 0 82.97 0 0 0 0 54.33 56.84
0 55.57 0 0 44.98 104.23 0 0 0 0 54.33 0 0
0 0 0 0 55.69 0 133.54 169.74 0 102.4 56.84 0 0
```

3.1.2 Peta Jalan Sekitar Alun-Alun Bandung

```
8
Museum KAA, -6.921284918454274, 107.60957943240983
Simpang Braga, -6.919723089454119, 107.609940342634
Jurnal Risa, -6.918789028521304, 107.60948363279562
Braga City Walk, -6.91713043077999, 107.60883021614896
Museum Mandala, -6.9174506555519075, 107.61103859891946
Bakmi Rica Kejaksaan, -6.918415434001528, 107.61186984411852
SPBU, -6.919859329250897, 107.61201040242935
The Centre of Bandung, -6.921387017201472, 107.611106436847
```

```
0 176.98 0 0 0 0 0 162.5
176.98 0 120.86 0 0 0 229.44 0
0 120.86 0 185.92 0 268.73 0 0
0 0 185.92 0 232.2 0 0 0
0 0 0 232.2 0 166.93 0 0
0 0 268.73 0 166.93 0 160.47 0
0 229.44 0 0 0 160.47 0 202.82
162.5 0 0 0 0 0 202.82 0
```

3.1.3 Peta Jalan Sekitar Buahbatu atau Bandung Selatan

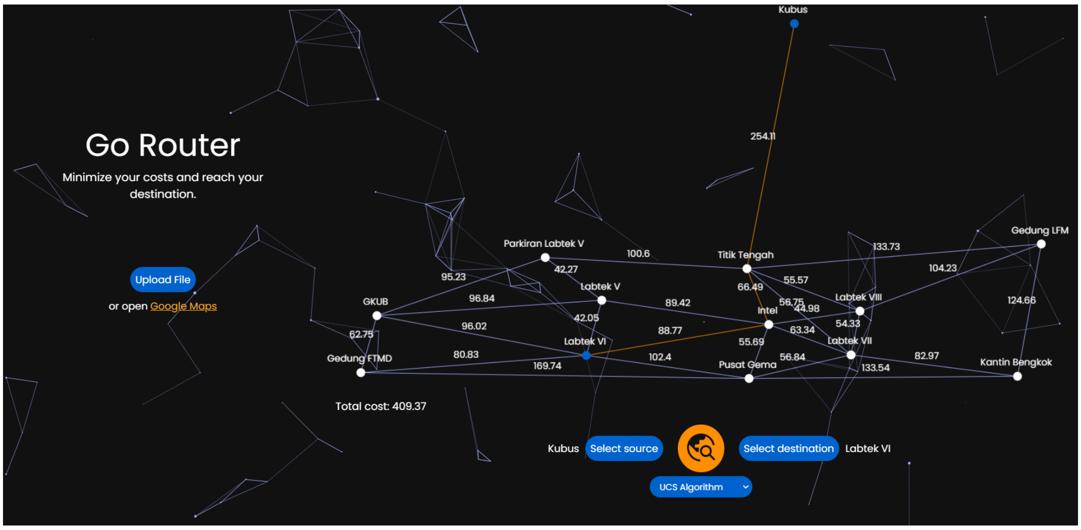
```
8
Universitas Islam Nusantara, -6.945832500507639, 107.64434603645425
Edelweiss Hospital, -6.943617699023188, 107.64979863586177
Ciduran Selatan Bypass, -6.9421024125955135, 107.652860318267
Masjid Miftahul Jannah, -6.937807631379689, 107.65290393209608
RSU Pindad, -6.9401974740772765, 107.64600424305948
Gedung Graha SembadhaVicaya, -6.933174808590936, 107.6467978495106
Direktorat Perhub AD, -6.930144163663747, 107.64755672810774
Pertigaan Terusan PSM, -6.930812727620998, 107.6545234905284
0 601.97 0 0 625.17 0 0 0
601.97 0 386.76 0 0 0 0 0
0 386.76 0 460.15 0 0 0 0
0 0 460.15 0 780.39 0 0 776.34
625.17 0 0 780.39 0 753.74 0 0
0 0 0 0 753.74 0 375.02 0
0 0 0 0 0 375.02 0 756.08
0 0 0 776.34 0 0 756.08 0
```

3.1.4 Peta Jalan Kawasan di Kota Asalmu

```
8
Aula Simfonia Jakarta, -6.152915184814156, 106.84323972638512
SPBU Industri, -6.149236517856851, 106.83881761490679
Halte Gunung Sahari, -6.146214386015122, 106.83419713078882
Mangga Dua Square, -6.138661093651788, 106.83140256250385
Ancol, -6.129132124652569, 106.83342039736061
Stasiun Ancol, -6.127991521394342, 106.84620906276047
PRJ, -6.143296175290157, 106.84608069458818
Pademangan, -6.14178440927396, 106.8385809532435
0 603.46 0 0 0 0 1070 0
603.46 0 572.02 0 0 0 0 753.55
0 572.02 0 882.51 1920 0 0 0
0 0 882.51 0 1110 0 0 1020
0 0 1920 1110 0 1310 0 0
0 0 0 0 1310 0 1750 1710
1070 0 0 0 0 1750 0 779.58
0 753.55 0 1020 0 1710 779.58 0
```

3.2 I/O Program

3.2.1 Test Case Sekitar Labtek



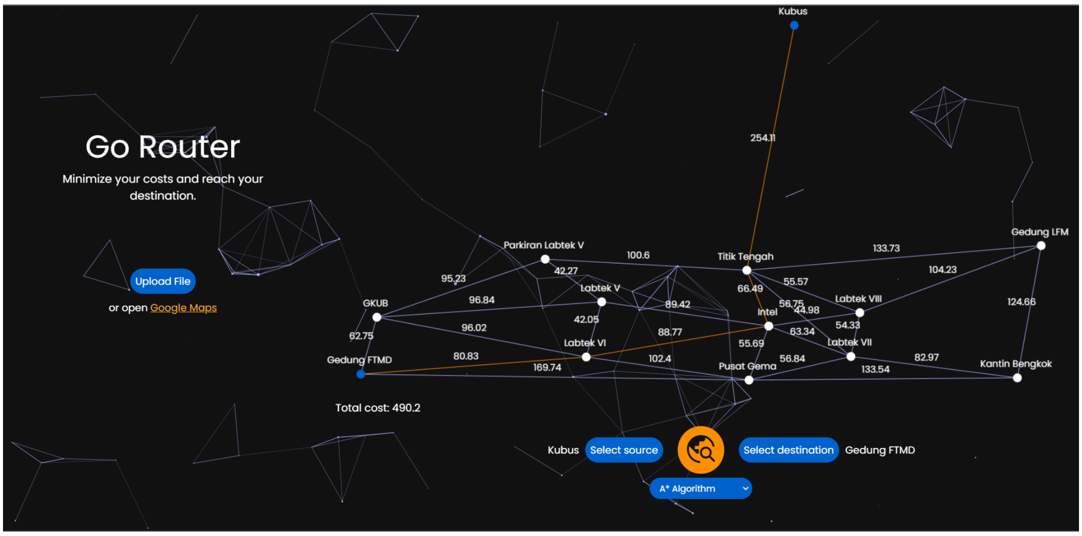
Go Router

Minimize your costs and reach your destination.

Total cost: 409.37

Kubus Select source Select destination Labtek VI

UCS Algorithm



Go Router

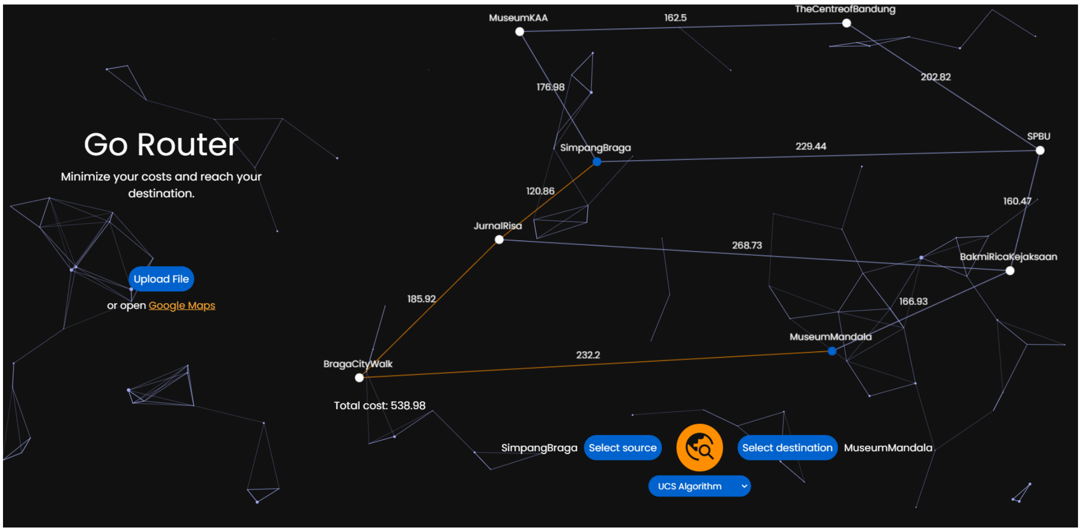
Minimize your costs and reach your destination.

Total cost: 490.2

Kubus Select source Select destination Gedung FTMD

A* Algorithm

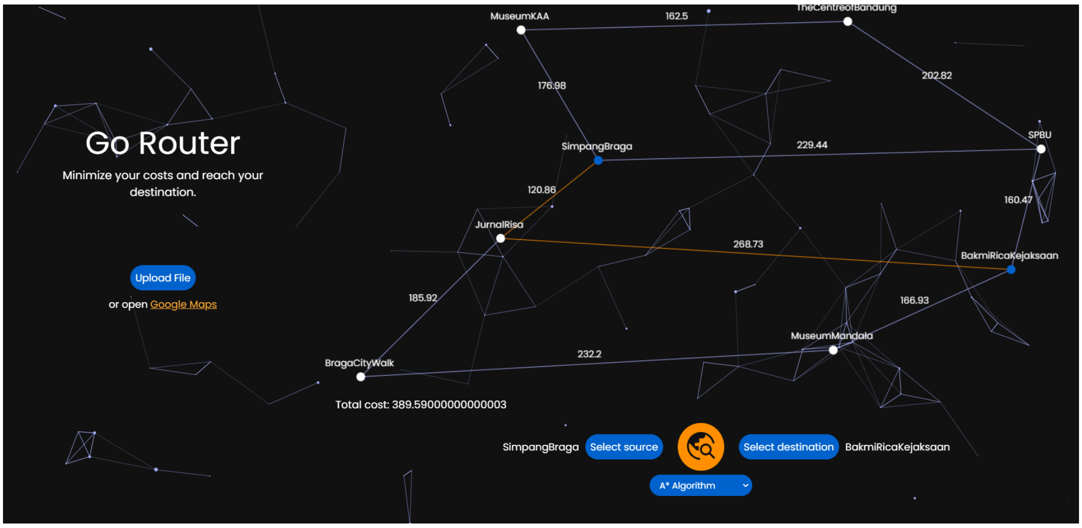
3.2.2 Test Case Alun-Alun Bandung



Total cost: 538.98

SimpangBraga Select source Select destination MuseumMandala

UCS Algorithm

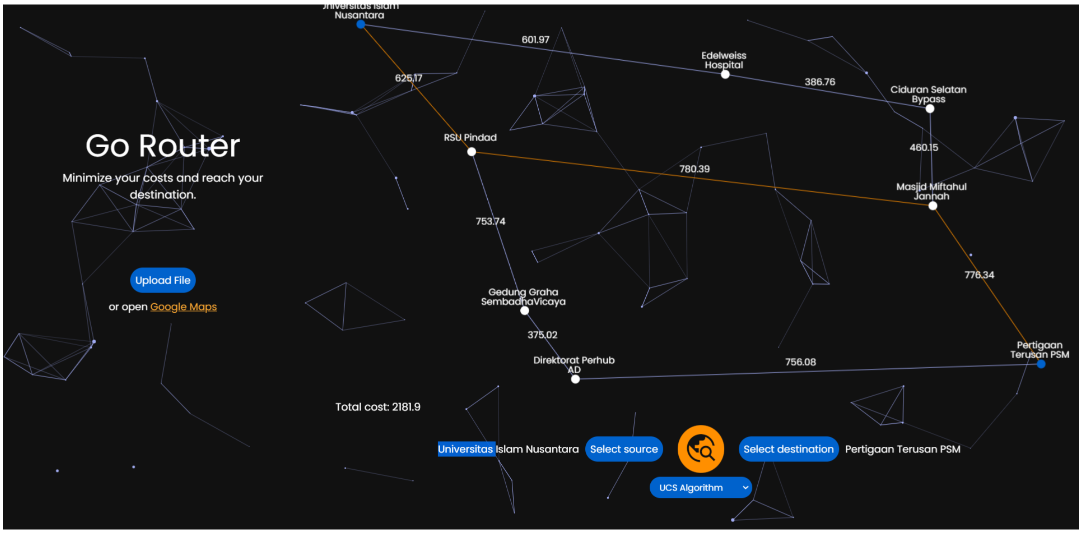


Total cost: 389.59000000000003

SimpangBraga Select source Select destination BakmiRicaKejaksaan

A* Algorithm

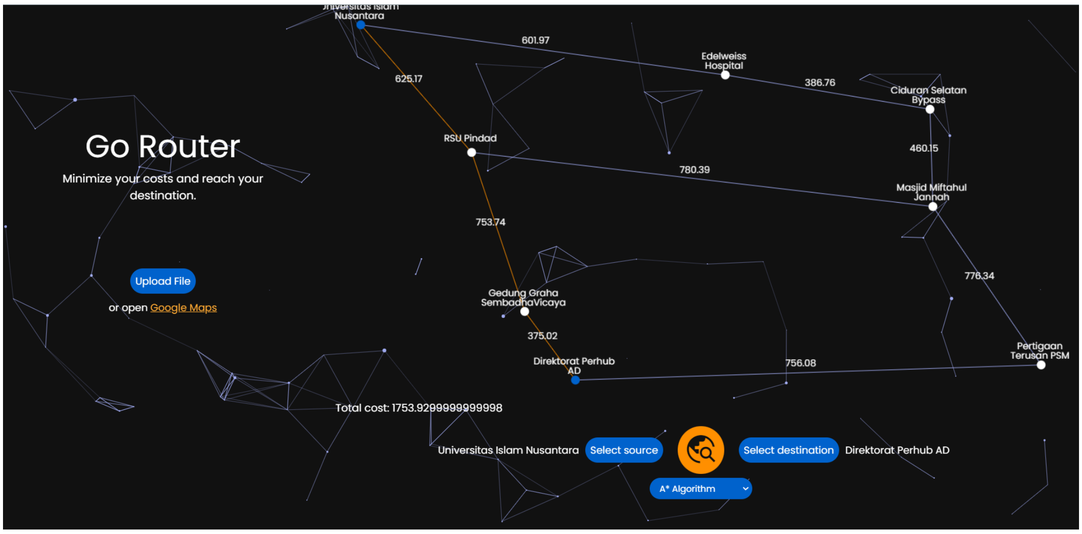
3.2.3 Test Case Buahbatu



Total cost: 2181.9

Universitas Islam Nusantara Select source Select destination Pertigaan Terusan PSM

UCS Algorithm

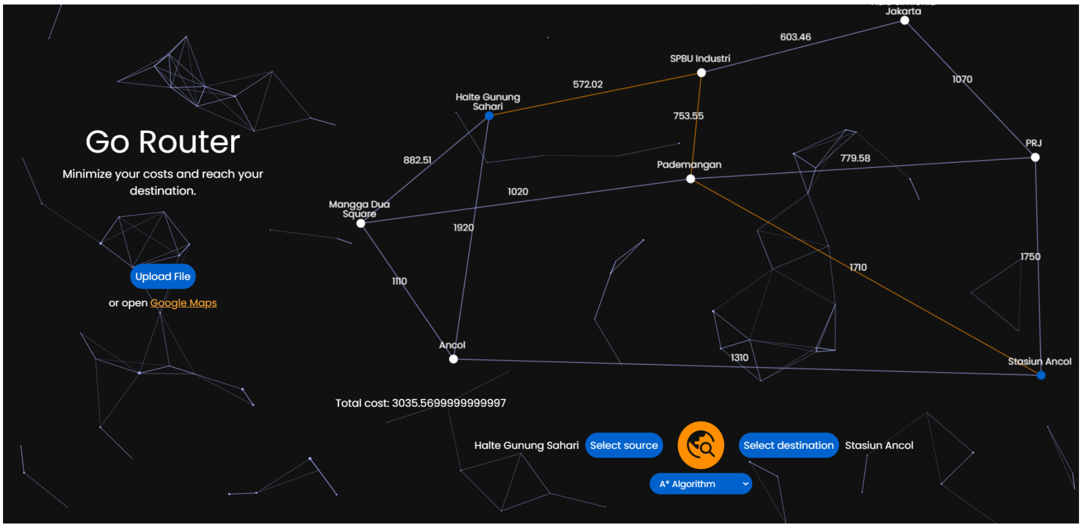


Total cost: 1753.9299999999998

Universitas Islam Nusantara Select source Select destination Direktorat Perhub AD

A* Algorithm

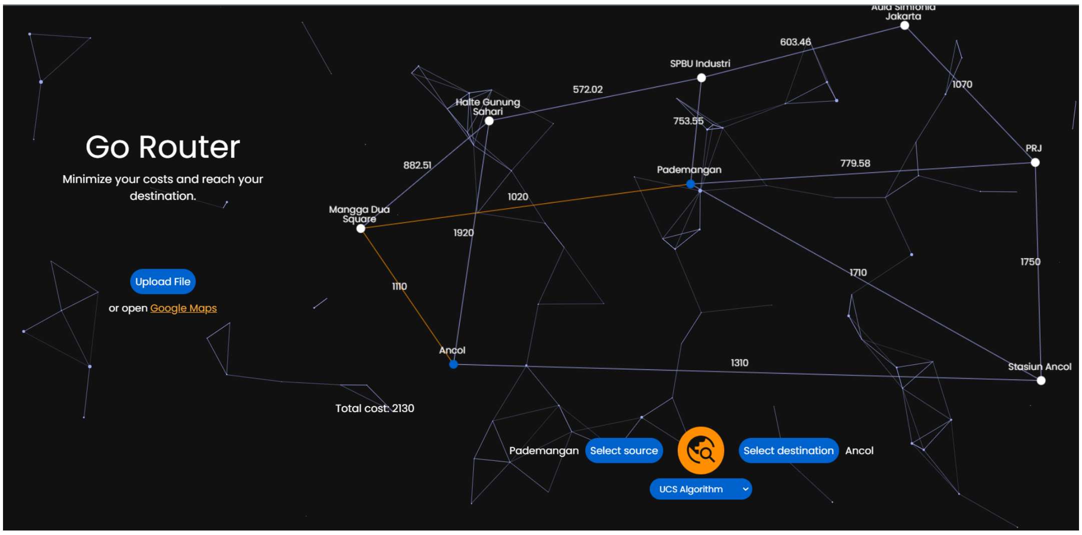
3.2.4 Test Case Jakarta



Total cost: 3035.5699999999997

Halte Gunung Sahari Select source Select destination Stasiun Ancol

A* Algorithm



Total cost: 2130

Pademangan Select source Select destination Ancol

UCS Algorithm

4 Kesimpulan

4.1 Kesimpulan

Dari tugas kecil ini, kami belajar beberapa hal. Hal pertama yang kami pelajari adalah tentang bagaimana mengimplementasikan algoritma UCS dan A* pada bahasa pemrograman Go. Hal kedua adalah tentang *web-development* khususnya bagaimana menyambungkan bagian *front-end* dengan *back-end*. Hal terakhir yang kami pelajari adalah mengutilisasi penggunaan *branching* dan melakukan unit testing untuk setiap algoritma yang telah kami buat supaya *readibility* bagi sesama dan pembaca lebih baik.

4.2 Komentar

Kalau mau buat bonus, harus nyambung sama Google Maps API, harus masukin kartu kredit/kartu debit. Lain kali mungkin kalau bonus boleh dipertimbangkan lagi kira-kira lebih mudah pengaksesan datanya.

5 Checklist

Poin	Judul Fitur	Ya	Tidak
1	Program dapat menerima input graf	✓	
2	Program dapat menghitung lintasan terpendek dengan UCS	✓	
3	Program dapat menghitung lintasan terpendek dengan A*	✓	
4	Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
5	Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta serta lintasan terpendek pada peta		✓
6	Tambahan: Membuat visualizer sendiri tanpa library bawaan	✓	

6 Daftar Pustaka

Munir, Rinaldi. *Bagian 1: BFS, DFS, UCS, Greedy Best First Search.* 2023. URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf (visited on 04/10/2023).

— *Bagian 2: Algoritma A*.* 2023. URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf (visited on 04/10/2023).

7 Lampiran

7.1 Link Repository GitHub

- Repo GitHub [Click Me!]
- https://github.com/NicholasLiem/Tucil3_13521083_13521135