

LAPORAN TUGAS KECIL III
STRATEGI ALGORITMA

**IMPLEMENTASI ALGORITMA UCS DAN A* UNTUK
MENENTUKAN LINTASAN TERPENDEK**

**Diajukan sebagai tugas kecil Mata Kuliah IF2211 Strategi Algoritma pada Semester II
Tahun Akademik 2022/2023**



Disusun Oleh:
Jason Rivalino (13521008)
Juan Christopher Santoso (13521116)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

DAFTAR ISI

DAFTAR ISI.....	3
BAB I: DESKRIPSI MASALAH.....	4
BAB II: TEORI SINGKAT.....	5
I. Teori Singkat Algoritma Uniform-Cost Search (UCS).....	5
II. Teori Singkat Algoritma A-Star (A*).....	6
BAB III: PENJELASAN ALGORITMA & SOURCE PROGRAM YANG DIBUAT.....	7
I. Penjelasan Algoritma Penyelesaian Masalah.....	7
II. Penjelasan Algoritma Program yang Dibuat.....	7
BAB IV: EKSPERIMEN.....	23
I. Hasil Running Program.....	23
II. Uji Coba Hasil Program.....	24
BAB V: KESIMPULAN & KOMENTAR.....	30
REFERENSI.....	31
LAMPIRAN.....	32

BAB I

DESKRIPSI MASALAH

Algoritma UCS (Uniform cost search) dan A* (atau *A-Star*) dapat digunakan untuk menentukan lintasan terpendek dari suatu titik ke titik lain. Dalam tugas kecil ini tugas yang diberikan adalah untuk 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.

BAB II

TEORI SINGKAT

I. Teori Singkat Algoritma *Uniform-Cost Search* (UCS)

Algoritma *Uniform-Cost Search* atau UCS merupakan algoritma yang melakukan pencarian berdasarkan biaya kumulatif terendah untuk menemukan jalur dari *node* awal menuju ke *node* tujuan. Algoritma ini umumnya digunakan untuk menemukan rute dengan nilai biaya akumulasi minimum di dalam graf berbobot.

Cara kerja dari algoritma UCS sebenarnya mirip dengan algoritma BFS, yaitu sama-sama menggunakan *Queue*. Hanya saja, dalam algoritma UCS, *queue* yang digunakan merupakan *priority queue* dengan prioritas diurutkan berdasarkan nilai kumulatif terendah hingga tertinggi. Algoritma ini juga dapat disebut sebagai varian dari algoritma Dijkstra karena kita tidak memasukan semua simpul dalam *queue*, tetapi hanya memasukkan *node* sumber, lalu *node* lainnya satu per satu dimasukkan sesuai kebutuhan. Dalam setiap iterasi, akan diperiksa apakah item sudah masuk dalam *priority queue*. Jika sudah masuk, maka melakukan kunci penurunan dan jika belum, item akan dimasukkan dalam *priority queue*.

Untuk *pseudocode* dari algoritma *Uniform-Cost Search* sendiri adalah sebagai berikut:

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node  $\leftarrow$  a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier  $\leftarrow$  a priority queue ordered by PATH-COST, with node as the only element
  explored  $\leftarrow$  an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node  $\leftarrow$  POP(frontier) /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child  $\leftarrow$  CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier  $\leftarrow$  INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
```

Sumber:<https://cs.stackexchange.com/questions/149259/uniform-cost-search-with-backtracking-and-additional-constraint>

II. Teori Singkat Algoritma A-Star (A*)

Algoritma A* merupakan algoritma pencarian untuk menemukan jalur terpendek antara *node* awal dengan *node* akhir. Algoritma ini sering digunakan untuk menemukan rute dengan jalur terpendek dalam penjelajahan peta.

Cara kerja dari algoritma ini adalah dengan melakukan pencarian jarak terpendek dengan metode penjelajahan graph (*graph traversal*). Algoritma ini mengaplikasikan graf berbobot pada penerapan algoritmanya yang mewakili biaya pengambilan setiap jalur atau tindakan. Pencarian jalur dengan algoritma ini bersifat optimal (mendapatkan hasil yang paling kecil dalam hal biaya) dan lengkap (mendapatkan semua hasil yang mungkin dari suatu masalah). Akan tetapi, pencarian dengan algoritma ini membutuhkan banyak ruang penyimpanan karena akan menyimpan segala kemungkinan jalur.

Untuk *pseudocode* dari algoritma A* sendiri adalah sebagai berikut:

```

1: table ← hash table
2: for all reads read do
3:   trim_N(read) {remove leading and trailing Ns}
4:   replace_N(read) {replace internal Ns with A, C, G, T}
5:   for all k-mers kmer in read do
6:     hash(kmer)
7:   end for
8: end for
9: for all k-mers kmer in table do
10:  if table[kmer] = 1 then
11:    remove(kmer, table)
12:  end if
13: end for
14: for all reads read do
15:   errors ← find_errors(read) {locate all errors in read}
16:   pos ← next(errors)
17:   while pos ≥ 0 do
18:     {explore possible corrections at position}
19:     subs ← substitutions(read, pos)
20:     ins ← insertions(read, pos)
21:     dels ← deletions(read, pos)
22:     {select the best correction at position}
23:     correction ← best(subs, ins, dels)
24:     if valid(correction) then
25:       apply(correction)
26:       errors ← find_errors(read)
27:     else
28:       homops ← homopolymers(read, pos)
29:       correction ← best(homops)
30:       if valid(correction) then
31:         apply(correction)
32:         errors ← find_errors(read)
33:       end if
34:     end if
35:     pos ← next(errors)
36:   end while
37:   if low_information(read) then
38:     filter(read)
39:   end if
40: end for
```

Sumber:https://www.researchgate.net/figure/Algorithm-pseudocode-A-pseudocode-for-the-error-correction-algorithm_fig3_270907635

BAB III

PENJELASAN ALGORITMA & SOURCE PROGRAM YANG DIBUAT

I. Penjelasan Algoritma Penyelesaian Masalah

Untuk tahapan dari penyelesaian masalah pencarian rute terdekat dengan algoritma UCS dan A* adalah sebagai berikut:

- 1) Inisialisasi array untuk menyimpan rute dan *queue* untuk menyimpan jalur rute
- 2) Memasukkan simpul awal sebagai rute awal ke dalam *queue*
- 3) Melakukan pengecekan dengan loop, jika panjang dari *queue* bernilai 0, berarti tidak ada solusi
- 4) Melakukan *dequeue* pada *queue* untuk mengambil urutan rute yang akan dilakukan pengecekan
- 5) Simpul yang akan dianalisis diambil dari elemen terakhir pada rute yang telah *dequeue*
- 6) Melakukan pengecekan terhadap simpul yang masih tersedia, jika masih ada membuat array baru untuk memasukkan rute yang masih tersedia dan dibandingkan dengan rute yang sudah dilewati
- 7) Jika simpul yang tersedia berhasil melewati pengecekan, maka simpul tersebut akan dimasukkan ke dalam urutan rute
- 8) Melakukan sorting *queue* terurut menaik (jika algoritma UCS diurutkan berdasarkan nilai *weight*, jika algoritma A* diurutkan berdasarkan nilai *weight* $g(n) + \text{nilai heuristik A}^* h(n)$)
- 9) Program akan terus berlanjut hingga solusi sudah ditemukan, algoritma akan dihentikan dan array penyimpan rute akan disimpan. Jika masih belum ditemukan, rute tersebut akan di-*enqueue* ke dalam queue dan kembali dilakukan iterasi

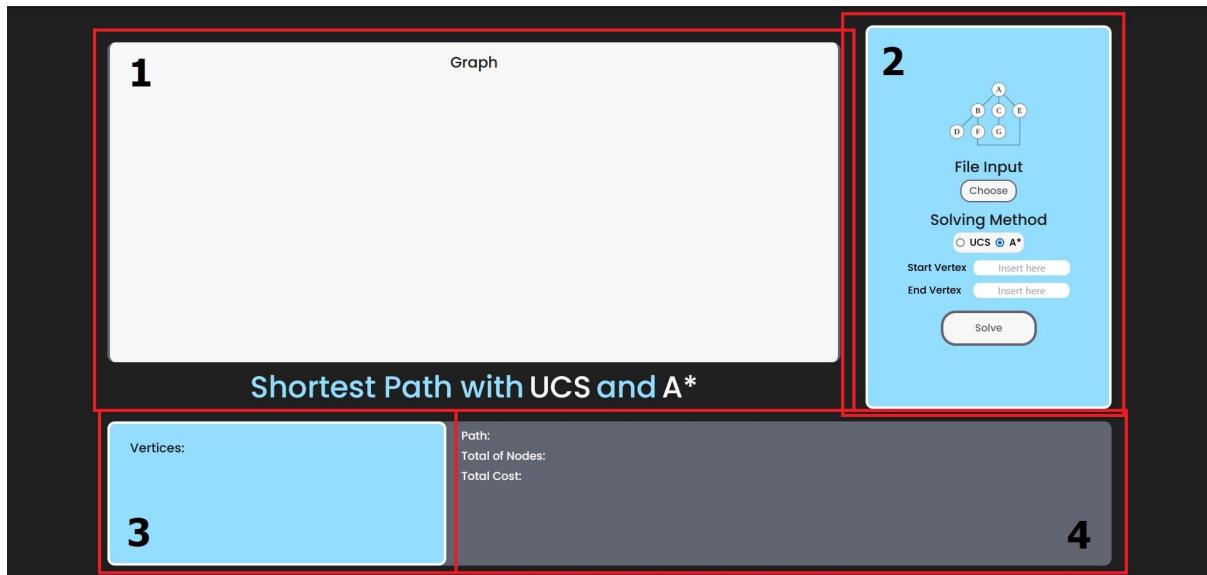
II. Penjelasan Algoritma Program yang Dibuat

Pada program yang dibuat, program terdiri atas bagian *front-end* dan juga bagian *back-end*. Untuk *front-end* dibuat untuk memunculkan tampilan *interface* dalam bentuk *web-based*. Sedangkan, untuk *back-end*, bagian ini merupakan bagian untuk mengimplementasikan fungsi-fungsi untuk melakukan rute pencarian.

IF2211 Strategi Algoritma

1) Penjelasan *Front-End* Program

Untuk *front-end* yang dibuat, kami menggunakan HTML dan CSS dengan dibantu oleh *framework* Tailwind CSS untuk membuat bentuk *front-end* dari programnya. Bentuk program yang dibuat berbasis *web-based* dengan tampilan dan fungsi-fungsi tiap bagian adalah sebagai berikut:



Fungsi-fungsi bagian yang terdapat pada tampilan program adalah sebagai berikut:

1. Visualisasi graph
 2. Input *file.txt* dan memilih metode antara UCS atau A* dan memilih vertex awal dan vertex akhir
 3. Menampilkan semua vertex yang ada pada graph
 4. Menampilkan rute yang dilewati, jumlah nodes, dan jumlah biaya (*cost*) dari node awal menuju ke node akhir
- 2) Penjelasan *Back-End* Program

Untuk *back-end* yang ada pada program kami, terdapat beberapa file yang memiliki fungsinya masing-masing. Beberapa file tersebut antara lain *graph.go*, *io.go*, *queue.go*, *route.go*, dan *solver.go*.

a) Graph.go

Program pada file ini terdiri dari berbagai macam fungsi untuk pengolahan simpul dan sisi pada graf yang ingin diproses. Untuk tipe data yang dibuat pada file program ini antara lain:

Nama Tipe Data	Deskripsi
Vertex	Tipe data yang terdiri atas variabel key dengan tipe data string, longPos dengan tipe data float, dan latPos dengan tipe data float
Edge	Tipe data yang terdiri atas variable startVertex dengan tipe data Vertex, endVertex dengan tipe data Vertex, dan weight dengan tipe data float
Graph	Tipe data yang terdiri atas nVertex dengan tipe data integer, vertices dengan tipe data array of Vertex, nEdge dengan tipe data integer, dan edges dengan tipe data array of edge.

```

8  type Vertex struct {
9    key string
10   longPos float64
11   latPos float64
12 }
13
14 type Edge struct {
15   startVertex Vertex
16   endVertex   Vertex
17   weight      float64
18 }
19
20 type Graph struct {
21   nVertex  int
22   vertices []*Vertex
23   nEdge    int
24   edges    []*Edge
25 }
26

```

Untuk fungsi yang terdapat dalam file program ini antara lain:

Nama Fungsi/Prosedur	Deskripsi
func (v Vertex) calculateDistance(v2 Vertex) float64	Fungsi untuk menghitung jarak antar dua simpul dengan rumus <i>Euclidean</i>
func (v Vertex) copyVertex(v2 Vertex)	Prosedur untuk <i>copy</i> vertex yang telah terisi sebelumnya
func (g *Graph) AddVertex(k string)	Fungsi untuk memasukkan vertex

IF2211

Strategi Algoritma

	kedalam array of vertex
func (g *Graph) AddEdge(v1 Vertex, v2 Vertex, w float64)	Fungsi untuk memasukkan edge kedalam array of edge
func (g *Graph) GetVertex(k string) *Vertex	Fungsi untuk mencari vertex dalam graph
func (g *Graph) GetEdgeWithStartV(startV Vertex) []*Edge	Fungsi untuk mencari edge awal dalam graph
func (g *Graph) GetEdgeWithEndV(endV Vertex) []*Edge	Fungsi untuk mencari edge akhir dalam graph
func IsContainVertex(vertices []*Vertex, vertex Vertex) bool	Fungsi untuk menentukan apakah sebuah vertex berada di dalam array of vertex
func IsContainEdge(edges []*Edge, edge Edge) bool	Fungsi untuk menentukan apakah sebuah edge berada di dalam array of edge
func IsSameVertex(v1 Vertex, v2 Vertex) bool	Fungsi untuk menentukan apakah dua buah vertex sama atau tidak
func (g *Graph) DisplayGraph()	Fungsi untuk menampilkan jumlah dari vertex dan juga edge

```

27 func (v Vertex) calculateDistance(v2 Vertex) float64 {
28     return math.Sqrt(math.Pow(v2.latPos-v.latPos, 2) + math.Pow(v2.longPos-v.longPos, 2))
29 }
30
31 func (v Vertex) copyVertex(v2 Vertex) {
32     v.key = v2.key
33     v.latPos = v2.latPos
34     v.longPos = v2.longPos
35 }
36
37 func (g *Graph) AddVertex(k string) {
38     v := Vertex{
39         key: k,
40     }
41     if !IsContainVertex(g.vertices, v) {
42         g.vertices = append(g.vertices, &v)
43         g.nVertex++
44     }
45 }
46
47 func (g *Graph) AddEdge(v1 Vertex, v2 Vertex, w float64) {
48     e := Edge{
49         startVertex: v1,
50         endVertex: v2,
51         weight: w,
52     }
53     if !IsContainEdge(g.edges, e) {
54         g.edges = append(g.edges, &e)
55         g.nEdge++
56     }
57 }
58

```

IF2211

Strategi Algoritma

```
59 ˜ func (g *Graph) GetVertex(k string) *Vertex {
60      for i := 0; i < g.nVertex; i++ {
61          if g.vertices[i].key == k {
62              return g.vertices[i]
63          }
64      }
65      return nil
66  }
67
68 ˜ func (g *Graph) GetEdgeWithStartV(startV Vertex) []*Edge {
69      result := []*Edge{}
70      for i := 0; i < g.nEdge; i++ {
71          if IsSameVertex(g.edges[i].startVertex, startV) {
72              result = append(result, g.edges[i])
73          }
74      }
75      return result
76  }
77
78 ˜ func (g *Graph) GetEdgeWithEndV(endV Vertex) []*Edge {
79      result := []*Edge{}
80      for i := 0; i < g.nEdge; i++ {
81          if IsSameVertex(g.edges[i].endVertex, endV) {
82              result = append(result, g.edges[i])
83          }
84      }
85      return result
86  }
87
88 ˜ func IsContainVertex(vertices []*Vertex, vertex Vertex) bool {
89      for i := 0; i < len(vertices); i++ {
90          if IsSameVertex(*vertices[i], vertex) {
91              return true
92          }
93      }
94      return false
95  }
96
```

```
97  func IsContainEdge(edges []*Edge, edge Edge) bool {
98      for i := 0; i < len(edges); i++ {
99          if edges[i].startVertex == edge.startVertex && edges[i].endVertex == edge.endVertex {
100              return true
101          }
102      }
103      return false
104  }
105
106 func IsSameVertex(v1 Vertex, v2 Vertex) bool {
107     return v1.key == v2.key && v1.latPos == v2.latPos && v1.longPos == v2.longPos
108 }
109
```

IF2211 Strategi Algoritma

```
110 func (g *Graph) DisplayGraph() {
111     fmt.Printf("Amount of Vertices : %d\n", g.nVertex)
112     for i := 0; i < g.nVertex; i++ {
113         fmt.Printf("%s (%.2f, %.2f)\n", g.vertices[i].key, g.vertices[i].longPos, g.vertices[i].latPos)
114     }
115     fmt.Printf("Amount of Edges : %d\n", g.nEdge)
116     for i := 0; i < g.nEdge; i++ {
117         fmt.Printf("| \"%s\" -> (%.2f) -> \"%s\" |\n", g.edges[i].startVertex.key, g.edges[i].weight, g.edges[i].endVertex.key)
118     }
119 }
120 }
```

b) Io.go

Program pada file ini terdiri dari berbagai macam fungsi untuk membaca input graf yang terdapat pada file.txt untuk kemudian diproses di dalam program. Untuk fungsi-fungsi yang terdapat dalam file program ini antara lain:

Nama Fungsi/Prosedur	Deskripsi
func IdentifyAmountVertex(path string) int	Fungsi untuk menghitung jumlah vertex yang terdapat pada file graf yang akan dibaca
func ReadFileToGraph(path string) *Graph	Fungsi untuk membaca file.txt dan mengubah graf dari file menjadi graf yang akan diproses dalam program
func AskingUserInput() string	Fungsi untuk meminta input file kepada pengguna
func AskUserStartEndVertices () (string, string)	Fungsi untuk meminta simpul awal dan simpul akhir yang akan dicari kepada pengguna

```
13 v func IdentifyAmountVertex(path string) int {
14     file, err := os.Open(path)
15 v     if err != nil {
16         log.Fatal(err)
17     }
18     count := 0
19     content := bufio.NewScanner(file)
20     content.Split(bufio.ScanLines)
21 v     for content.Scan() {
22         count++
23     }
24     return count/2
25 }
26 }
```

IF2211

Strategi Algoritma

```
27 func ReadFileToGraph(path string) *Graph {
28     g := &Graph{}
29     idx := 0
30     tempIdx := 0
31
32     n := IdentifyAmountVertex(path)
33     file, err := os.Open(path)
34     if err != nil {
35         log.Fatal(err)
36     }
37
38     content := bufio.NewScanner(file)
39     content.Split(bufio.ScanLines)
40     for content.Scan() {
41         if idx < n {
42             vInfo := strings.Split(content.Text(), " ")
43             g.AddVertex(vInfo[0])
44             long, _ := strconv.ParseFloat(vInfo[1], 64)
45             g.vertices[idx].longPos = float64(long)
46             lat, _ := strconv.ParseFloat(vInfo[2], 64)
47             g.vertices[idx].latPos = float64(lat)
48             idx++
49         } else {
50             line := strings.Split(content.Text(), " ")
51             for i:= 0 ; i < n; i++{
52                 val, _ := strconv.ParseFloat(line[i], 64)
53                 if (val != 0){
54                     g.AddEdge(*g.vertices[tempIdx], *g.vertices[i], val)
55                 }
56             }
57             tempIdx++
58         }
59     }
60     return g
61 }
```

```
62
63 func AskingUserInput() string {
64     curDir, _ := os.Getwd()
65     var name string
66     fmt.Print("Which file do you want to use: ")
67     fmt.Scanln(&name)
68     name = name + ".txt"
69     fullPath := path.Join(curDir, "test", name)
70
71     return fullPath
72 }
73
74 func AskUserStartEndVertices () (string, string) {
75     var startV string
76     var endV string
77     fmt.Print("What is the name of the start Vertex: ")
78     fmt.Scanln(&startV)
79     fmt.Print("What is the name of the end Vertex: ")
80     fmt.Scanln(&endV)
81     return startV, endV
82 }
83 }
```

c) Queue.go

Program pada file ini terdiri dari berbagai macam fungsi untuk pengelolaan *queue* rute yang nantinya akan digunakan untuk menyimpan rute yang telah dilewati untuk pencarian solusi rute dengan nilai *weight* yang paling kecil. Untuk tipe data yang dibuat pada file program ini antara lain:

Nama Tipe Data	Deskripsi
QueueRoute	Tipe data yang terdiri atas variabel buffer dengan tipe data array of route dan nRoute dengan tipe data integer

```
3 ✓ type QueueRoute struct {
4     buffer []*Route
5     nRoute int
6 }
7
```

Untuk fungsi yang terdapat dalam file program ini antara lain:

Nama Fungsi/Prosedur	Deskripsi
func (q *QueueRoute) Enqueue(r Route)	Prosedur untuk memasukkan rute ke dalam array <i>queue</i> dengan sistem FIFO
func (q *QueueRoute) Dequeue() *Route	Fungsi untuk mengeluarkan rute dari dalam array <i>queue</i> dengan sistem FIFO
func (q *QueueRoute) DequeueAt(idx int) Route	Fungsi untuk mengeluarkan rute dengan indeks tertentu dari dalam array <i>queue</i> rute
func (q *QueueRoute) SortAscending()	Prosedur untuk mengurutkan <i>queue</i> secara terurut naik berdasarkan <i>weight</i> dari rutennya (kondisi <i>weight</i> saja)
func (q *QueueRoute) SortAStarAscending()	Prosedur untuk mengurutkan <i>queue</i> secara terurut naik berdasarkan <i>weight</i> dari rutennya (kondisi <i>weight</i> + heuristic A*)
func (q *QueueRoute) DisplayQueue()	Prosedur untuk menampilkan <i>queue</i> rute

IF2211

Strategi Algoritma

```
8  func (q *QueueRoute) Enqueue(r Route) {
9    q.buffer = append(q.buffer, &r)
10   q.nRoute++
11 }
12
13 func (q *QueueRoute) Dequeue() *Route {
14   temp := *q.buffer[0]
15   q.buffer = q.buffer[1:]
16   q.nRoute--
17   return &temp
18 }
19
20 func (q *QueueRoute) DequeueAt(idx int) Route {
21   temp := *q.buffer[idx]
22   for i := idx; i < q.nRoute-1; i++ {
23     q.buffer[i] = q.buffer[i+1]
24   }
25   q.nRoute--
26   return temp
27 }
28
```

```
29 func (q *QueueRoute) SortAscending() {
30   if q.nRoute > 1 {
31     for i := 1; i < q.nRoute; i++ {
32       temp := &Route{}
33       temp.CopyConstructorRoute(q.buffer[i])
34       j := i - 1
35       for temp.IsLessWeight(*q.buffer[j]) && j > 0 {
36         q.buffer[j+1].CopyRoute(q.buffer[j])
37         j--
38       }
39       if !temp.IsLessWeight(*q.buffer[j]) {
40         q.buffer[j+1].CopyRoute(temp)
41       } else {
42         q.buffer[j+1].CopyRoute(q.buffer[j])
43         q.buffer[j].CopyRoute(temp)
44       }
45     }
46   }
47 }
48
49 func (q *QueueRoute) SortAStarAscending() {
50   if q.nRoute > 1 {
51     for i := 1; i < q.nRoute; i++ {
52       temp := &Route{}
53       temp.CopyConstructorRoute(q.buffer[i])
54       j := i - 1
55       for temp.IsAStarLess(*q.buffer[j]) && j > 0 {
56         q.buffer[j+1].CopyRoute(q.buffer[j])
57         j--
58       }
59       if !temp.IsAStarLess(*q.buffer[j]) {
60         q.buffer[j+1].CopyRoute(temp)
61       } else {
62         q.buffer[j+1].CopyRoute(q.buffer[j])
63         q.buffer[j].CopyRoute(temp)
64       }
65     }
66   }
67 }
68
```

```
69 func (q *QueueRoute) DisplayQueue() {
70     for _, r := range q.buffer {
71         r.DisplayRoute()
72     }
73 }
74 }
```

d) Route.go

Program pada file ini terdiri dari berbagai macam fungsi untuk memproses dan mengkalkulasikan hasil rute yang telah dilewati dari vertex awal menuju ke vertex tujuan (memproses vertex mana saja yang dilalui dan menghitung *weightnya*). Untuk tipe data yang dibuat pada file program ini antara lain:

Nama Tipe Data	Deskripsi
Route	Tipe data yang terdiri atas variabel buffer dengan tipe data array of vertex, nVertex dengan tipe data integer, accWeight dengan tipe data float, aStarDistance dengan tipe data float

```
5 type Route struct {
6     buffer      []*Vertex
7     nVertex     int
8     accWeight   float64
9     aStarDistance float64
10 }
```

Untuk fungsi yang terdapat dalam file program ini antara lain:

Nama Fungsi/Prosedur	Deskripsi
func (r *Route) InsertLastVertex(v Vertex)	Prosedur untuk memasukkan vertex terakhir kedalam array of vertex
func (r *Route) DeleteFirstVertex() Vertex	Fungsi untuk menghapus vertex pertama yang ada dalam array of vertex
func (r *Route) GetLastVertex() Vertex	Fungsi untuk mencari vertex terakhir yang ada dalam array of vertex
func (r *Route) IsEmpty() bool	Fungsi untuk mengecek apakah

IF2211

Strategi Algoritma

	array of vertex kosong atau tidak (jika kosong tidak ada rute)
func (r *Route) IsLessWeight(r2 Route) bool	Fungsi untuk mengecek dan membandingkan <i>weight</i> dari rute
func (r *Route) IsAStarLess(r2 Route) bool	Fungsi untuk mengecek dan membandingkan (<i>weight</i> dari rute + nilai heuristic A*)
func (r *Route) CopyConstructorRoute(r2 *Route)	Prosedur untuk <i>copy constructor</i> rute dari <i>default array of route</i>
func (r *Route) CopyRoute(r2 *Route)	Prosedur untuk <i>copy</i> rute yang telah terisi sebelumnya
func (r *Route) DisplayRoute()	Prosedur untuk menampilkan <i>weight</i> dari rute dan nilai heuristik A*

```

11
12 func (r *Route) InsertLastVertex(v Vertex) {
13     r.buffer = append(r.buffer, &v)
14     r.nVertex++
15 }
16
17 func (r *Route) DeleteFirstVertex() Vertex {
18     temp := *r.buffer[0]
19     r.buffer = r.buffer[1:]
20     r.nVertex--
21     return temp
22 }
23
24 func (r *Route) GetLastVertex() Vertex {
25     return *r.buffer[r.nVertex-1]
26 }
27
28 func (r *Route) IsEmpty() bool {
29     return r.nVertex == 0
30 }
31
32 func (r *Route) IsLessWeight(r2 Route) bool {
33     return r.accWeight < r2.accWeight
34 }
35
36 func (r *Route) IsAStarLess(r2 Route) bool {
37     return r.accWeight+r.aStarDistance < r2.accWeight+r2.aStarDistance
38 }
39
40 func (r *Route) CopyConstructorRoute(r2 *Route) {
41     // Make sure that r is empty
42     r.nVertex = r2.nVertex
43     r.accWeight = r2.accWeight
44     r.aStarDistance = r2.aStarDistance
45     for i := range r2.buffer {
46         r.buffer = append(r.buffer, r2.buffer[i])
47     }
48 }
49

```

```
50 func (r *Route) CopyRoute(r2 *Route) {
51     r.nVertex = r2.nVertex
52     r.accWeight = r2.accWeight
53     r.aStarDistance = r2.aStarDistance
54     r.buffer = r2.buffer[:r2.nVertex]
55 }
56
57 func (r *Route) DisplayRoute() {
58     for i := 0; i < r.nVertex; i++ {
59         fmt.Print(r.buffer[i].key, " ")
60     }
61     fmt.Println(" |AccWeight: ", r.accWeight, " |AStarDistance: ", r.aStarDistance)
62 }
63 }
64 }
```

e) Solver.go

Program pada file ini terdiri dari berbagai macam fungsi untuk mengaplikasikan algoritma pencarian rute dengan UCS dan juga A*. Untuk tipe data yang dibuat pada file program ini antara lain:

Nama Tipe Data	Deskripsi
Solver	Tipe data yang terdiri atas variabel solRoute dengan tipe data Route

```
7 type Solver struct {
8     |     solRoute Route
9 }
```

Untuk fungsi yang terdapat dalam file program ini antara lain:

Nama Fungsi/Prosedur	Deskripsi
func (s *Solver) SolveUCS(g Graph, startVKey string, endVKey string)	Prosedur untuk menentukan rute dengan nilai terendah menggunakan algoritma UCS (aplikasi algoritma UCS)
func (s *Solver) SolveAStar(g Graph, startVKey string, endVKey string)	Prosedur untuk menentukan rute dengan nilai terendah menggunakan algoritma A* (aplikasi algoritma A*)
func AvailableEdges(r Route, edges []*Edge) []*Edge	Fungsi untuk mengecek edge yang masih tersedia untuk dilewati
func (s *Solver) DisplaySolutionRoute()	Fungsi untuk menampilkan rute solusi

IF2211

Strategi Algoritma

func IsSolution(r Route, endVKey string) bool	Fungsi untuk mengecek apakah sudah mencapai titik akhir solusi route
---	--

```
11 func (s *Solver) SolveUCS(g Graph, startVKey string, endVKey string) {
12
13     // Declaring Variables
14     q := &QueueRoute{}
15     curRoute := &Route{}
16     // Preparing Variables Setup
17     curRoute.InsertLastVertex(*g.GetVertex(startVKey))
18     q.Enqueue(*curRoute)
19
20     count := 0
21     check := false
22     for !check {
23
24         if q.nRoute == 0 {
25             break
26         }
27
28         curRoute = q.Dequeue()
29         curVertex := curRoute.GetLastVertex()
30
31         if IsSolution(*curRoute, endVKey) {
32             check = true
33             break
34         }
35
36         availableEdges := AvailableEdges(*curRoute, g.EdgeWithStartV(curVertex))
37         if len(availableEdges) != 0 {
38             temp := &Route{}
39             temp.CopyConstructorRoute(curRoute)
40             for i, e := range availableEdges {
41                 if i > 0 {
42                     temp.CopyRoute(curRoute)
43                 }
44                 temp.InsertLastVertex(e.endVertex)
45                 temp.accWeight += e.weight
46                 q.Enqueue(*temp)
47             }
48         }
49         q.SortAscending()
50         count++
51     }
52
53     if !check {
54         fmt.Println("No Solution Found")
55     } else {
56         s.solRoute.CopyConstructorRoute(curRoute)
57     }
58 }
59 }
```

IF2211

Strategi Algoritma

```
60  func (s *Solver) SolveAStar(g Graph, startVKey string, endVKey string) {
61  // Declaring Variables
62  q := &QueueRoute{}
63  curRoute := &Route{}
64  // Preparing Variables Setup
65  curRoute.InsertLastVertex(*g.GetVertex(startVKey))
66  q.Enqueue(*curRoute)
67
68  goalVertex := g.GetVertex(endVKey)
69
70  count := 0
71  check := false
72  for !check {
73
74      if q.nRoute == 0 {
75          break
76      }
77
78
79      curRoute = q.Dequeue()
80      curVertex := curRoute.GetLastVertex()
81
82      if IsSolution(*curRoute, endVKey) {
83          check = true
84          break
85      }
86
87      availableEdges := AvailableEdges(*curRoute, g.EdgeWithStartV(curVertex))
88      if len(availableEdges) != 0 {
89          temp := &Route{}
90          temp.CopyConstructorRoute(curRoute)
91          for i, e := range availableEdges {
92              if i > 0 {
93                  temp.CopyRoute(curRoute)
94              }
95              temp.InsertLastVertex(e.endVertex)
96              temp.accWeight += e.weight
97              temp.aStarDistance = e.endVertex.calculateDistance(*goalVertex)
98              q.Enqueue(*temp)
99          }
100     }
101     q.SortAStarAscending()
102     count++
103 }
104
105 if !check {
106     fmt.Println("No Solution Found")
107 } else {
108     s.solRoute.CopyConstructorRoute(curRoute)
109 }
110 }
```

```
112 func AvailableEdges(r Route, edges []*Edge) []*Edge {
113     result := []*Edge{}
114     for _, e := range edges {
115         if !IsContainVertex(r.buffer, e.endVertex) {
116             result = append(result, e)
117         }
118     }
119     return result
120 }
```

IF2211 Strategi Algoritma

```
122 func (s *Solver) DisplaySolutionRoute() {
123     s.solRoute.DisplayRoute()
124 }
125
126 func IsSolution(r Route, endVKey string) bool {
127     return r.GetLastVertex().key == endVKey
128 }
129
```

f) Main.go

Program pada file ini berfungsi untuk menjalankan program secara keseluruhan untuk menampilkan *web-based interface* dari *front-end* dan menjalankan algoritma pencarian di *back-end*. Untuk struktur programnya sendiri adalah sebagai berikut:

```
1 package main
2
3 import (
4     "RoutePlanner/src/backend"
5     "fmt"
6     "html/template"
7     "net/http"
8     "os"
9     "path"
10 )
11
12 type Data struct {
13     Algorithm string
14     FileName  string
15     Vertices   string
16     Solution   string
17     TotalNodes int
18     TotalCost  float64
19 }
20
21 func main() {
22     assetsPath := path.Join(getSrcPath(), "frontend", "assets")
23
24     http.HandleFunc("/", homePageHandler)
25     http.HandleFunc("/result", processHandler)
26     http.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir(assetsPath))))
27
28     fmt.Println("Server started at => localhost:9000")
29     http.ListenAndServe(":9000", nil)
30 }
31
32 func homePageHandler(w http.ResponseWriter, r *http.Request) {
33
34     if r.Method == "GET" {
35         var filePath = path.Join(getSrcPath(), "frontend", "index.html")
36         var tmpl = template.Must(template.New("form").ParseFiles(filePath))
37
38         err := tmpl.Execute(w, nil)
39         if err != nil {
40             http.Error(w, err.Error(), http.StatusInternalServerError)
41         }
42     }
43 }
44
45 }
```

IF2211

Strategi Algoritma

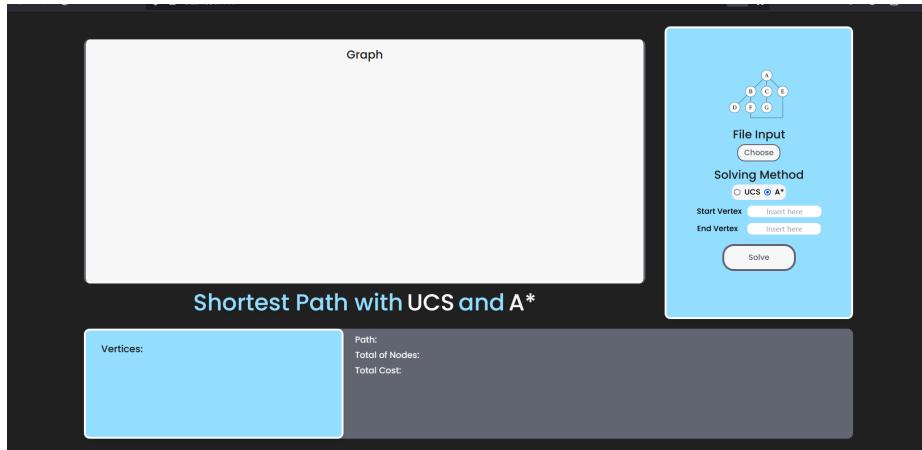
```
47 func processHandler(w http.ResponseWriter, r *http.Request) {
48     if r.Method == "POST" {
49
50         var filePath = path.Join(getSrcPath(), "frontend", "result.html")
51         var tmpl = template.Must(template.New("result").ParseFiles(filePath))
52
53         if err := r.ParseForm(); err != nil {
54             http.Error(w, err.Error(), http.StatusInternalServerError)
55         }
56
57         var algo = r.FormValue("algoritma")
58         var file = r.FormValue("myfile")
59         var start = r.FormValue("startV")
60         var end = r.FormValue("endV")
61
62         g := &backend.Graph{}
63         s := &backend.Solver{}
64
65         fmt.Println(algo)
66         fmt.Println(file)
67         fmt.Println(start)
68         fmt.Println(end)
69
70         root, _ := os.Getwd()
71         absPath := path.Join(root, "test", file)
72
73         g = backend.ReadFileToGraph(absPath)
74         if algo == "ucs" {
75             s.SolveUCS(*g, start, end)
76         } else {
77             s.SolveAStar(*g, start, end)
78         }
79         g.VisualizeGraph(s.GetSolutionRoute())
80         var verticesBuffer string = g.GraphVerticesToString()
81         var solutionBuffer string = s.SolutionRouteToString()
82
83         var data = Data{
84             Algorithm: algo,
85             FileName: file,
86             Vertices: verticesBuffer,
87             Solution: solutionBuffer,
88             TotalNodes: s.GetSolutionNodes(),
89             TotalCost: s.GetSolutionCost(),
90         }
91
92         if err := tmpl.Execute(w, data); err != nil {
93             http.Error(w, err.Error(), http.StatusInternalServerError)
94         }
95     }
96     http.Error(w, "", http.StatusBadRequest)
97 }
98
99 func getSrcPath() string {
100     curDir, _ := os.Getwd()
101     return path.Join(curDir, "src")
102 }
103 }
```

BAB IV

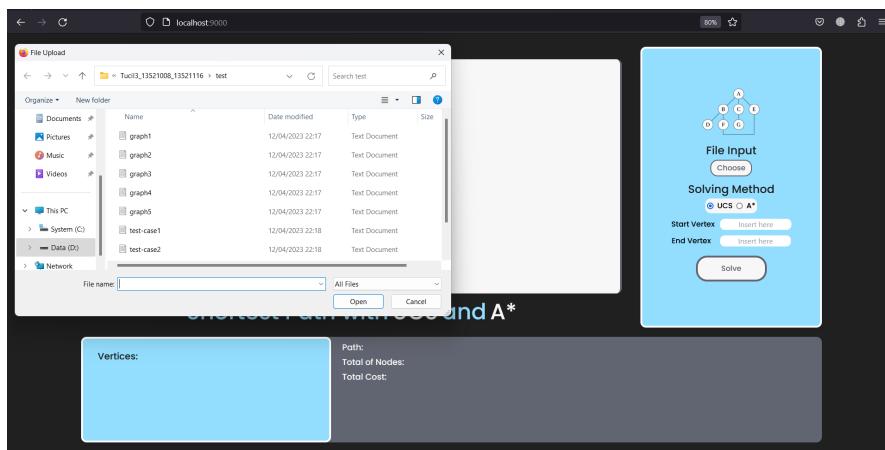
EKSPERIMEN

I. Hasil Running Program

- a) Tampilan awal saat membuka program



- b) Tampilan saat memilih *file.txt* yang ingin digunakan



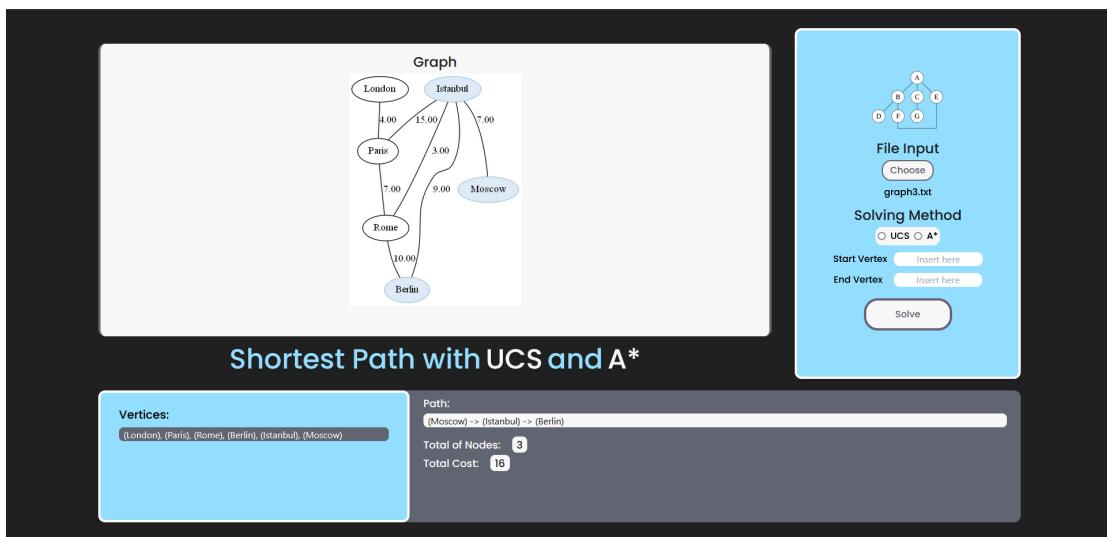
- c) Tampilan saat memilih algoritma dan input vertex awal dan akhir



IF2211

Strategi Algoritma

- d) Tampilan hasil pencarian rute



II. Uji Coba Hasil Program

- a) test-case1.txt (Peta jalan sekitar kampus ITB/Dago/Bandung Utara)

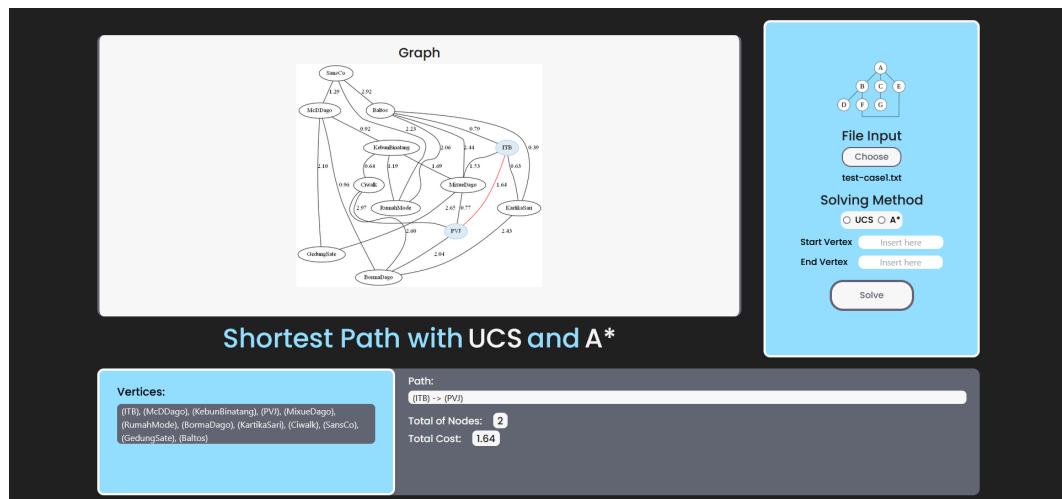
Isi file:

```
1 ITB 107.61057996449114, -6.891409734473638
2 McDago 107.61360178927141, -6.884577742189036
3 KebunBinatang 107.60757344796498, -6.889632298572686
4 PVJ 107.59605120589964, -6.888509420661611
5 MixueDago 107.61682218556373, -6.8787802629261146
6 RumahMode 107.5995493440505, -6.881778897104148
7 BormaDago 107.6180412651226, -6.87619610302739
8 KartikaSari 107.61268944786919, -6.896951963931746
9 Ciwalk 107.60433898080416, -6.894748291409175
10 SansCo 107.61873578556376, -6.874388705858442
11 GedungSate 107.6177515180798, -6.902955257097195
12 Baltos 107.60906077809669, -6.896966838334653
13 0 0 1.64 1.53 0 0 0.63 0 0 0 0.79
14 0 0 0.92 0 0 0 0.96 0 0 1.29 2.10 0
15 0 0.92 0 0 1.69 1.19 0 0 0.64 0 0 0
16 1.64 0 0 0 0.77 0 2.04 0 2.97 0 0 0
17 1.53 0 1.69 0.77 0 0 0 0 0 2.65 2.44
18 0 0 1.19 0 0 0 0 0 0 2.23 0 2.06
19 0 0.96 0 2.04 0 0 0 2.43 2.60 0 0 0
20 0.63 0 0 0 0 0 2.43 0 0 0 0 0.39
21 0 0 0.64 2.97 0 0 2.60 0 0 0 0 0
22 0 1.29 0 0 0 2.23 0 0 0 0 0 2.92
23 0 2.10 0 0 2.65 0 0 0 0 0 0 0
24 0.79 0 0 0 2.44 2.06 0 0.39 0 2.92 0 0
```

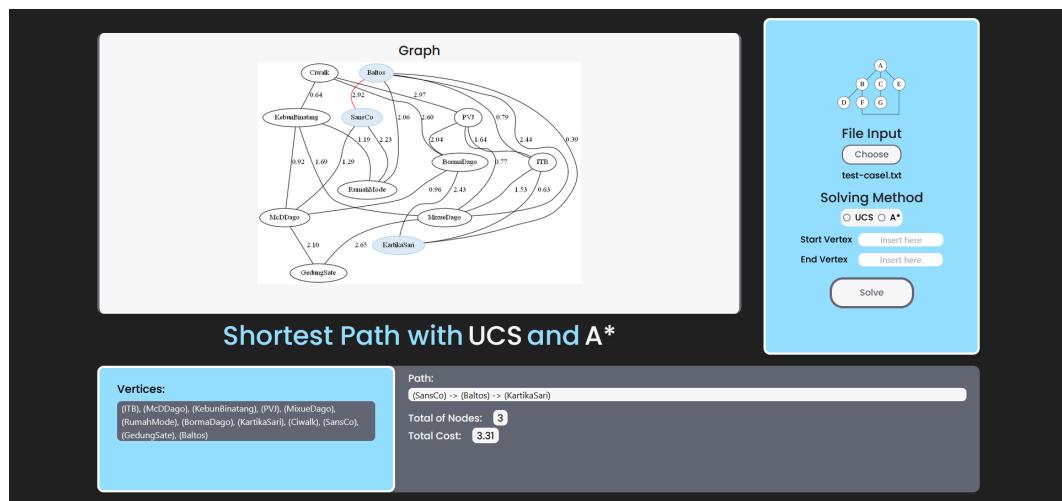
IF2211

Strategi Algoritma

1. Contoh test program (ITB -> PVJ)



2. Contoh test program (SansCo -> Kartika Sari)



b) test-case2.txt (Peta jalan sekitar Alun-alun Bandung)

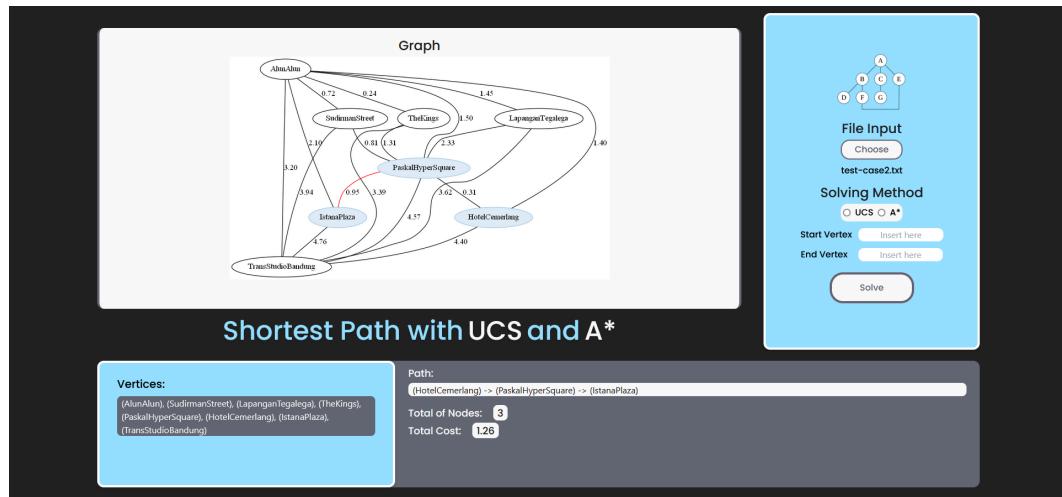
Isi file:

```
1  AlunAlun 107.60791636060287, -6.921503451421524
2  SudirmanStreet 107.60064256305091, -6.920379685167328
3  LapanganTegalega 107.60511893121488, -6.933654471607951
4  TheKings 107.60493781227642, -6.9228424417379575
5  PaskalHyperSquare 107.5956618100845, -6.912911575094042
6  HotelCemerlang 107.59758749905599, -6.912117133868341
7  IstanaPlaza 107.59687611543947, -6.905488904843278
8  TransStudioBandung 107.63589802418775, -6.924403236307638
9  0 0.72 1.45 0.24 1.50 1.40 2.10 3.20
10 0.72 0 0 0 0.81 0 0 3.94
11 1.45 0 0 0 2.33 0 0 3.62
12 0.24 0 0 0 1.31 0 0 3.39
13 1.50 0.81 2.33 1.31 0 0.31 0.95 4.57
14 1.40 0 0 0 0.31 0 0 4.40
15 2.10 0 0 0 0.95 0 0 4.76
16 3.20 3.94 3.62 3.39 4.57 4.40 4.76 0
```

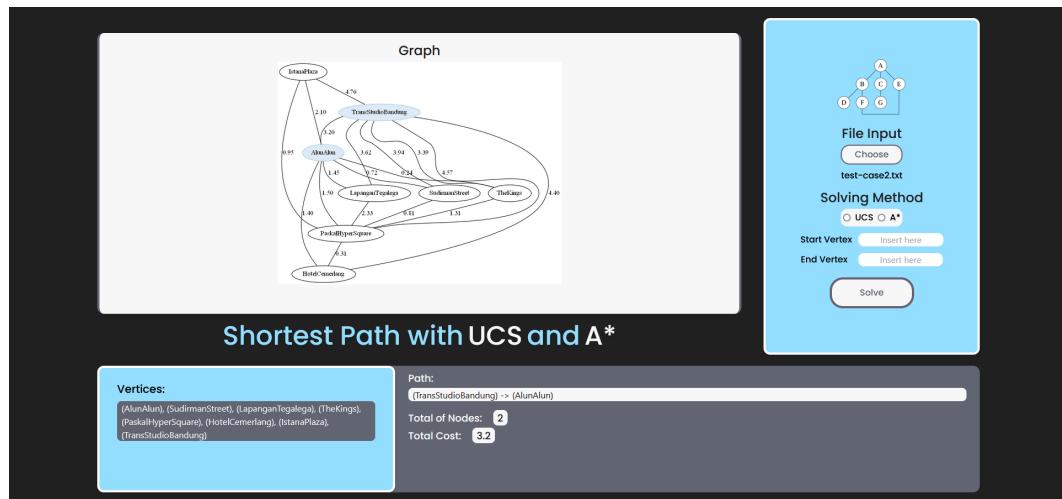
IF2211

Strategi Algoritma

- Contoh test program (Hotel Cemerlang -> Istana Plaza)



- Contoh test program (Trans Studio Bandung -> Alun-Alun)



- test-case3.txt (Peta jalan sekitar Buahbatu atau Bandung Selatan)

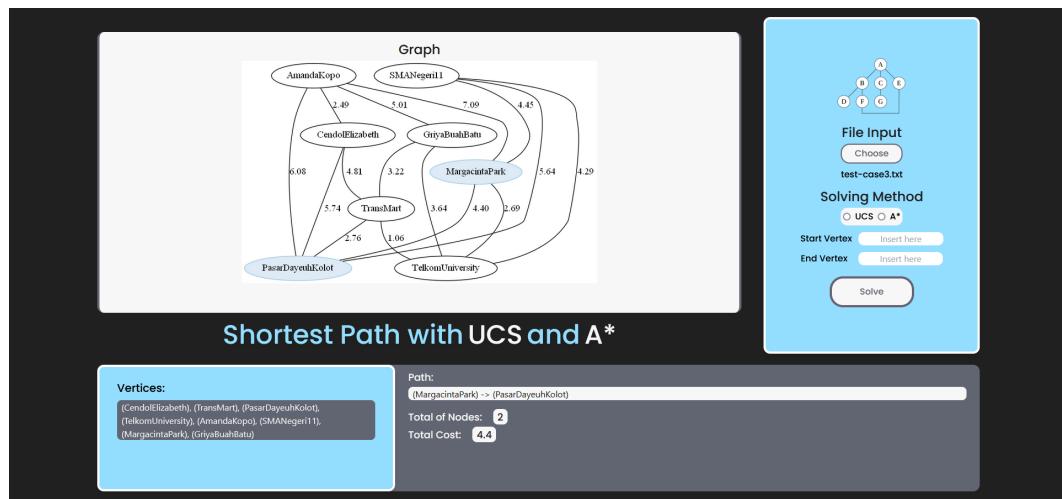
Isi file:

```
1 CendolElizabeth 107.60265377421953, -6.9416339162848475
2 TransMart 107.63879836852355, -6.965794316496801
3 PasarDayeuhKolot 107.62710113357753, -6.9885263401812985
4 TelkomUniversity 107.63168539704971, -6.9729856960029135
5 AmandaKopo 107.58368502219332, -6.954880645965372
6 SMANegeri11 107.61044287590494, -6.940651428732937
7 MargacintaPark 107.64794759704962, -6.954986694975226
8 GriyaBuahBatu 107.62673469704959, -6.940641994985418
9 0 4.81 5.74 0 2.49 0 0 0
10 4.81 0 2.76 1.06 0 0 0 3.22
11 5.74 2.76 0 0 6.08 5.64 4.40 0
12 0 1.06 0 0 0 4.29 2.69 3.64
13 2.49 0 6.08 0 0 0 7.09 5.01
14 0 0 5.64 4.29 0 0 4.45 0
15 0 0 4.40 2.69 7.09 4.45 0 0
16 0 3.22 0 3.64 5.01 0 0 0
```

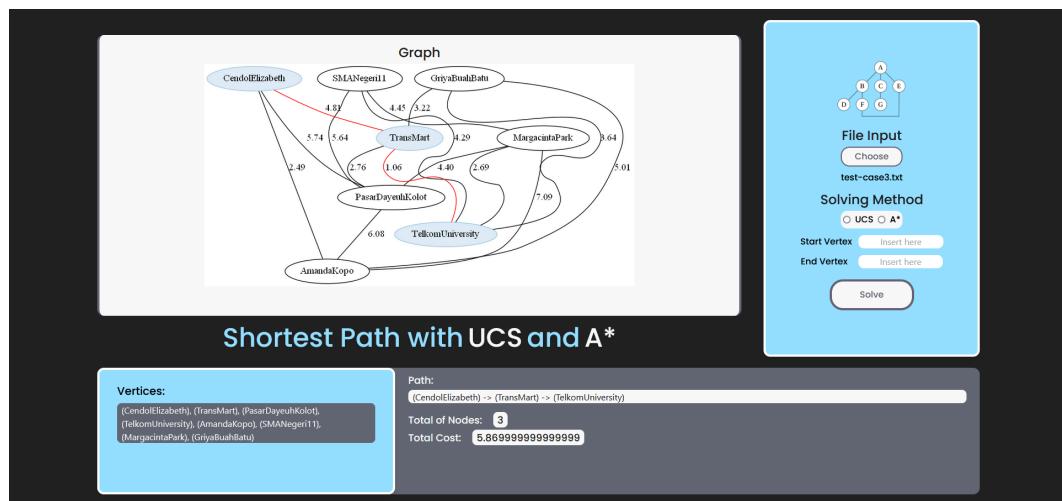
IF2211

Strategi Algoritma

1. Contoh test program (Margacinta Park -> Pasar Dayeah Kolot)



2. Contoh test program (Cendol Elizabeth -> Telkom University)



d) test-case4.txt (Peta jalan sebuah kawasan di kota asal (Harapan Indah))

Isi file:

```

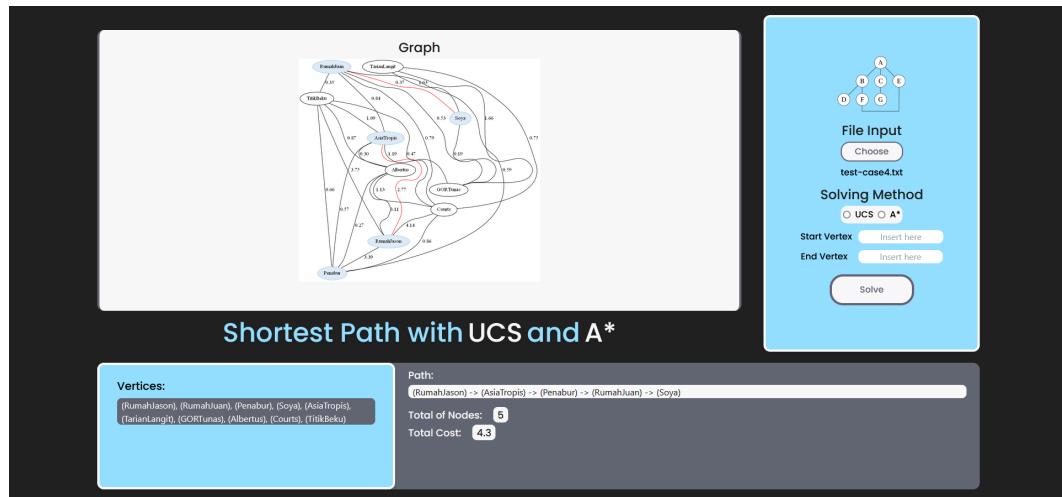
1 RumahJason 106.98173454067636, -6.1494458049796945
2 RumahJuan 106.97874554278651, -6.179320758817187
3 Penabur 106.97362361828729, -6.178132298018893
4 Soya 106.98194163206286, -6.178900181105388
5 AsiaTropis 106.97522158330068, -6.172735298363075
6 TarianLangit 106.97576972410556, -6.192225111081525
7 GORTunas 106.98344006531882, -6.179473694141415
8 Albertus 106.97426750215179, -6.175399478498407
9 Courts 106.97470599284478, -6.185600302949796
10 TitikBeku 106.9776017256263, -6.182191253432929
11 0 0 3.39 3.32 2.77 5.04 3.51 3.11 4.14 3.75
12 0 0 0.59 0.37 0.84 1.49 0.53 0.66 0.79 0.35
13 3.39 0.59 0 0.94 0.57 1.62 1.11 0.27 0.86 0.66
14 3.32 0.37 0.94 0 0.98 1.63 0.19 0.93 1.09 0.62
15 2.77 0.84 0.57 0.98 0 2.16 1.19 0.30 1.41 1.09
16 5.04 1.49 1.62 1.63 2.16 0 1.66 1.88 0.75 1.10
17 3.51 0.53 1.11 0.19 1.19 1.66 0 1.11 1.18 0.73
18 3.11 0.66 0.27 0.93 0.30 1.88 1.11 0 1.13 0.87
19 4.14 0.79 0.86 1.09 1.41 0.75 1.18 1.13 0 0.47
20 3.75 0.35 0.66 0.62 1.09 1.10 0.73 0.87 0.47 0

```

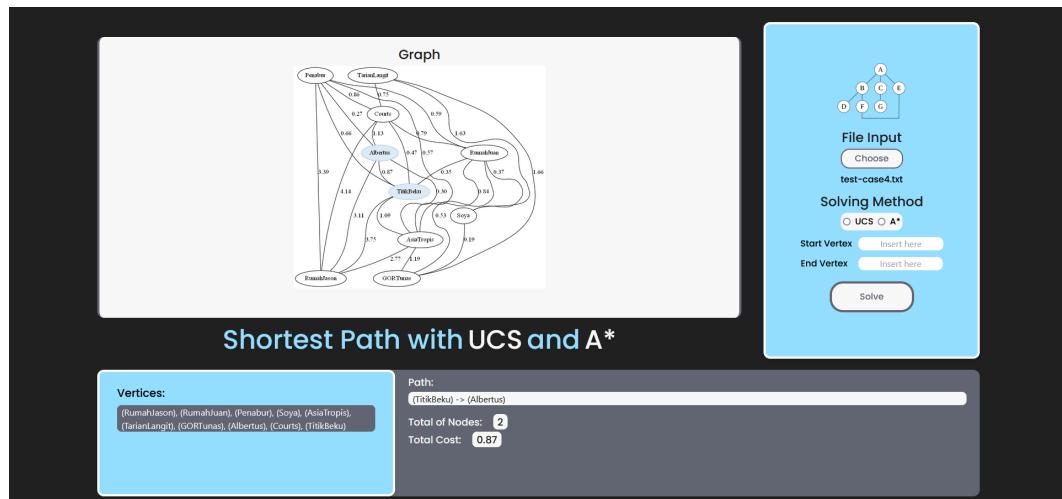
IF2211

Strategi Algoritma

1. Contoh test program (Rumah Jason -> Soya)

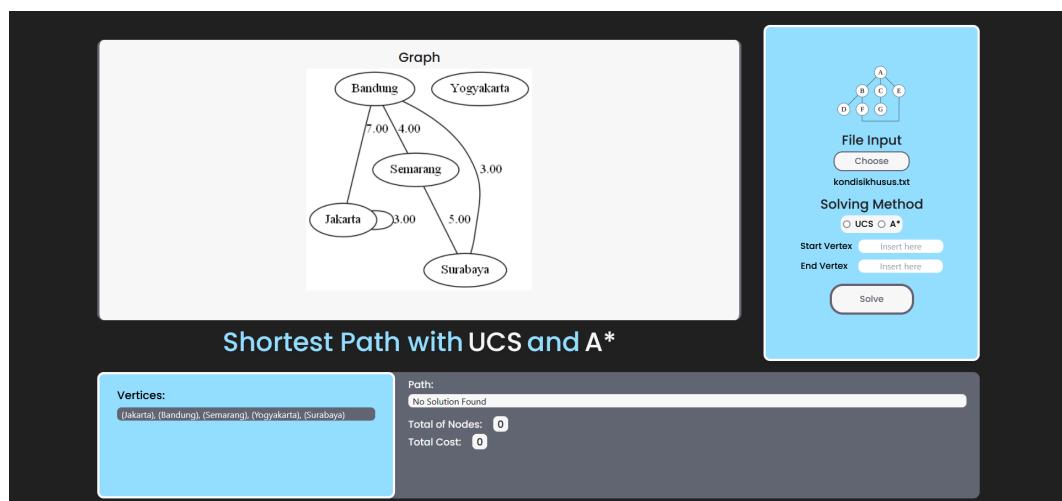


2. Contoh test program (Titik Beku -> Albertus)



Kondisi khusus:

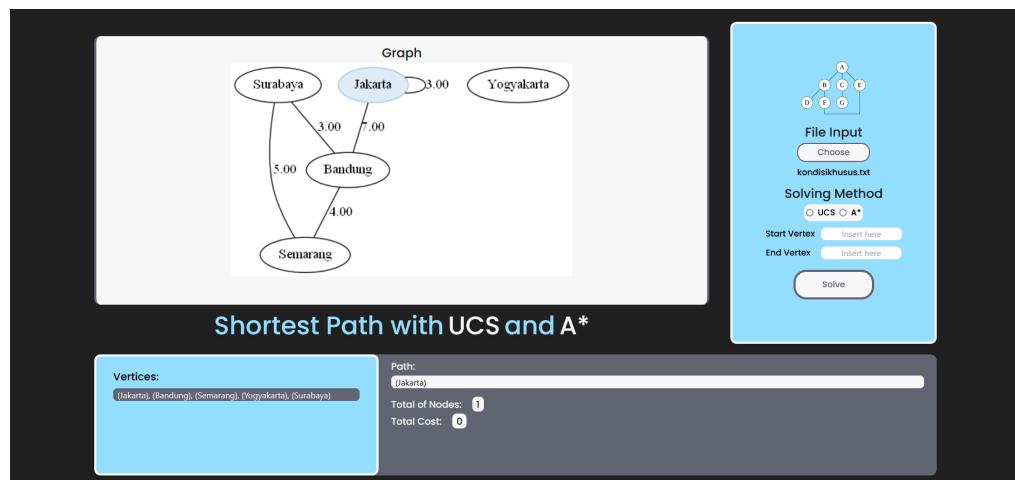
1. Ketika mencari rute untuk vertex yang terisolasi



IF2211

Strategi Algoritma

2. Ketika mencari rute awal dan rute akhir vertex yang sama



BAB V

KESIMPULAN & KOMENTAR

Kesimpulan yang didapat dari penggerjaan tugas kecil ini adalah sebagai berikut:

- 1) Algoritma *Uniform Cost-Search* dan A* dapat dipergunakan untuk melakukan pencarian jarak terdekat dengan nilai yang paling optimal.
- 2) Persoalan pencarian jarak yang ada pada peta dapat diselesaikan dengan algoritma *Uniform Cost-Search* dan A* untuk menemukan solusi optimalnya.

Komentar:

- 1) Jason Rivalino:

tugasnya sebenarnya seru, cuma waktunya kalo bisa lebih lama lagi agar mungkin tugas yang kami buat bisa lebih inovatif soalnya cape jujur. Overall, udah good

- 2) Juan Christopher Santoso:

hehehehe xixixi, terima kasih sudah membuat saya kurang tidur 😊

REFERENSI

- [1] “*Apa itu Uniform-Cost Search? Pengertian dan Cara Kerjanya.*” Trivusi. <https://www.trivusi.web.id/2022/10/apa-itu-algoritma-uniform-cost-search.html> (Diakses pada tanggal 10 April 2023)
- [2] “*Algoritma A* (A Star): Pengertian, Cara Kerja, dan Kegunaannya.*” Trivusi. <https://www.trivusi.web.id/2023/01/algoritma-a-star.html> (Diakses pada tanggal 10 April 2023)

LAMPIRAN

Pranala Github: https://github.com/Gulilil/Tucil3_13521008_13521116

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		✓