LAPORAN TUGAS KECIL 3
IF2211 STRATEGI ALGORITMA
SEMESTER II 2022-2023

Penentuan Lintasan Terpendek dengan Algoritma UCS dan A*

Disusun oleh:

Haidar Hamda	13521105
Bintang Dwi Marthen	13521144

PROGRAM STUDI
TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO
DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2022

1. DESKRIPSI PERSOALAN

Peta suatu daerah dapat direpresentasikan menggunakan graf dengan *node* melambangan suatu lokasi dengan *edge* merepresentasikan jalan antar dua lokasi. Dengan banyaknya permutasi jalur dari dua titik, mencari jalur terpendek merupakan suatu dambaan. Untuk mencari lintasan terpendek antara dua titik terdapat berbagai pilihan algoritma: Dijkstra, BFS, Floyd-Warshall, Bellman Ford, UCS, dan A*. Dalam tugas kecil ini, mahasiswa diminta untuk membuat algoritma UCS dan A* untuk mencari lintasan terpendek antara dua titik tertentu.

2. SOURCE CODE DALAM BAHASA GOLANG

2.1. Berkas graph.go

```
package lib

import (
	"container/list"
	"fmt"
	"strconv"
)

type Edge struct {
	dest   int
	weight float32
}

/*
Graph is a struct that represents a graph
Using an adjacency list
nodes are represented by integers to save the number of nodes
adj is the adjacency list
*/
type Graph struct {
	nodes int
	adj   []*list.List
	names []string
}

func GetName(g Graph) []string {
	return g.names
}

func AddEdge(g *Graph, src, dest int, weight float32) {
	g.adj[src].PushBack(&Edge{dest, weight})
}

func NewGraph(matrix [][]float32) *Graph {
	g    :=    &Graph{len(matrix),    make([]*list.List,    len(matrix)),
make([]string, len(matrix))}

	for i := 0; i < len(matrix); i++ {
		g.adj[i] = list.New()
		g.names[i] = strconv.Itoa(i)
	}

	for i := 0; i < len(matrix); i++ {
```

```
		for j := 0; j < len(matrix[i]); j++ {
			if matrix[i][j] != 0 {
				AddEdge(g, i, j, matrix[i][j])
			}
		}
	}

	return g
}

func NewGraphNamed(matrix [][]float32, names []string) *Graph {
	g := &Graph{len(matrix), make([]*list.List, len(matrix)), names}

	for i := 0; i < len(matrix); i++ {
		g.adj[i] = list.New()
	}

	for i := 0; i < len(matrix); i++ {
		for j := 0; j < len(matrix[i]); j++ {
			if matrix[i][j] != 0 {
				AddEdge(g, i, j, matrix[i][j])
			}
		}
	}

	return g
}

func PrintGraphInfos(g *Graph) {
	fmt.Println("Graph Infos:")
	fmt.Println("Number of nodes:", g.nodes)
	fmt.Println("Adjacency List:")
	for i := 0; i < g.nodes; i++ {
		fmt.Print(g.names[i], ": ")
		if g.adj[i].Len() == 0 {
			fmt.Println("No edge")
			continue
		}
		for e := g.adj[i].Front(); e != nil; e = e.Next() {
			fmt.Printf("%s(%f)  ",  g.names[e.Value.(*Edge).dest],
e.Value.(*Edge).weight)
		}
		fmt.Println()
	}
}
```

2.2. Berkas PrioQueue.go

```
package lib

import (
	"container/heap"
	"container/list"
)

//https://pkg.go.dev/container/heap
```

```
type Item struct {
	Value      int     // The Value of the item; arbitrary.
	Priority   float32 // The Priority of the item in the queue.
	PassedNode list.List
	// The Index is needed by update and is maintained by the
heap.Interface methods.
	Index int // The Index of the item in the heap.
}

// A PriorityQueue implements heap.Interface and holds Items.
type PriorityQueue []*Item

func (pq PriorityQueue) Len() int {
	return len(pq)
}

func (pq PriorityQueue) Less(i, j int) bool {
	return pq[i].Priority < pq[j].Priority
}

func (pq PriorityQueue) Swap(i, j int) {
	pq[i], pq[j] = pq[j], pq[i]
	pq[i].Index = i
	pq[j].Index = j
}

func (pq *PriorityQueue) Push(x any) {
	n := len(*pq)
	item := x.(*Item)
	item.Index = n
	*pq = append(*pq, item)
}

func (pq *PriorityQueue) Pop() any {
	old := *pq
	n := len(old)
	item := old[n-1]
	old[n-1] = nil  // avoid memory leak
	item.Index = -1 // for safety
	*pq = old[0 : n-1]
	return item
}

func (pq *PriorityQueue) Update(item *Item, Value int, Priority float32) {
	item.Value = Value
	item.Priority = Priority
	heap.Fix(pq, item.Index)
}
```

2.3. Berkas Utils.go

```
package lib

import (
	"bufio"
	"container/list"
	"errors"
```

```
	"fmt"
	"log"
	"math"
	"os"
	"strconv"
	"strings"
)

func itemInList(item Item, list list.List) bool {
	for e := list.Front(); e != nil; e = e.Next() {
		//println("lis", e.Value.(*Item).Value)
		if item.Value == e.Value.(*Item).Value {
			return true
		}
	}
	return false
}

func euclideanDistance(x1, y1, x2, y2 float32) float32 {
	return  float32(math.Sqrt(float64((x1-x2)*(x1-x2)  +  (y1-y2)*(y1-
y2))))
}

func ReadFiletoGraph(dir string) (*Graph, []float32, []float32, error) {
	f, err := os.Open(dir)
	if err != nil {
		//log.Fatal(err)
		return nil, nil, nil, err
	}
	defer f.Close()
	scanner := bufio.NewScanner(f)
	scanner.Scan()
	nodeCount, err := strconv.ParseInt(scanner.Text(), 10, 64)
	if err != nil {
		//log.Fatal(err)
		return nil, nil, nil, err
	}
	//println(parseInt)
	names := make([]string, int(nodeCount))
	xarr := make([]float32, int(nodeCount))
	yarr := make([]float32, int(nodeCount))
	if nodeCount < 8 {
		return nil, nil, nil, errors.New("invalid node count")
	}
	for i := int64(0); i < nodeCount; i++ {
		scanner.Scan()
		tmp := strings.Split(scanner.Text(), " ")
		if len(tmp) != 3 {
			return  nil,  nil,  nil,  errors.New("invalid  argument
number")
		}
		names[i] = tmp[0]
		x, err := strconv.ParseFloat(tmp[1], 64)
		if err != nil {
			//log.Fatal(err)
			return nil, nil, nil, err
```

```
			}
			xarr[i] = float32(x)
			y, err := strconv.ParseFloat(tmp[2], 64)
			if err != nil {
				//log.Fatal(err)
				return nil, nil, nil, err
			}
			yarr[i] = float32(y)
		}
		matrix := make([][]float32, nodeCount)
		for i := range matrix {
			matrix[i] = make([]float32, nodeCount)
		}
		for j := 0; j < int(nodeCount); j++ {
			scanner.Scan()
			tmp := strings.Split(scanner.Text(), " ")
			if len(matrix[j]) != len(tmp) {
				return nil, nil, nil, errors.New("out of bound")
			}
			for i := 0; i < len(tmp); i++ {
				parseF, err := strconv.ParseFloat(tmp[i], 64)
				if err != nil {
					//log.Fatal(err)
					return nil, nil, nil, err
				}
				if parseF < 0 {
					return  nil,  nil,  nil,  errors.New("negative
weight")
				}
				matrix[j][i] = float32(parseF)
			}
		}
		return NewGraphNamed(matrix, names), xarr, yarr, nil
}

func NameToIndex(g Graph, name string) int {
		for i := 0; i < len(g.names); i++ {
			if g.names[i] == name {
				return i
			}
		}
		return -1
}

func PathToName(g Graph, i Item) []string {
		path := make([]string, 0)
		for e := i.PassedNode.Front(); e != nil; e = e.Next() {
			path = append(path, g.names[e.Value.(int)])
		}
		return path
}

func FileNameParse() (Graph, []float32, []float32) {
		args := os.Args[1:]
		if len(args) != 1 {
			log.Fatal("invalid argument number")
```

```
	}
	g, x, y, err := ReadFiletoGraph(args[0])
	if err != nil {
		log.Fatal(err)
	}
	return *g, x, y
}

func RangedInput(min, max int) int {
	var input int
	for {
		fmt.Scanln(&input)
		if input >= min && input <= max {
			break
		}
		fmt.Println("invalid input")
	}
	return input
}

func PrintPath(g Graph, i Item) {
	if i.Priority == -1 {
		fmt.Println("No path found")
		return
	}
	fmt.Println("Path: ", PathToName(g, i))
	fmt.Println("Cost: ", i.Priority)
}
```

2.4. Berkas astar.go

```
package lib

import (
	"container/heap"
	"container/list"
)

func euclideanCalculator(x, y []float32, end int) []float32 {
	euclid := make([]float32, len(x))
	for i := 0; i < len(x); i++ {
		euclid[i] = euclideanDistance(x[i], y[i], x[end], y[end])
	}
	return euclid
}

func Astar(g Graph, x, y []float32, start, end int) *Item {
	// init variables
	euclid := euclideanCalculator(x, y, end)
	expandedWeight := euclid
	pq := make(PriorityQueue, 0)
	heap.Init(&pq)
	heap.Push(&pq, &Item{start, euclid[start], list.List{}, 0})
	visited := list.New()
	// start algorithm
	for pq.Len() > 0 {
		currentItem := heap.Pop(&pq).(*Item)
```

```
            currentNodeNumber := currentItem.Value
            currentNodeCost := currentItem.Priority
            currentPassed := list.New()
            currentPassed.PushBackList(&currentItem.PassedNode)
            currentPassed.PushBack(currentNodeNumber)

            // update expandedWeight
            for e := g.adj[currentNodeNumber].Front(); e != nil; e =
e.Next() {
                  if       expandedWeight[e.Value.(*Edge).dest]       >
currentNodeCost+e.Value.(*Edge).weight+euclid[e.Value.(*Edge).dest] {
                        expandedWeight[e.Value.(*Edge).dest]          =
currentNodeCost + e.Value.(*Edge).weight + euclid[e.Value.(*Edge).dest]
                  }
            }

            if currentNodeNumber == end {
                  var dist float32 = 0
                  for e := currentPassed.Front(); e != nil && e.Next()
!= nil; e = e.Next() {
                        for f := g.adj[e.Value.(int)].Front(); f != nil;
f = f.Next() {
                              if        f.Value.(*Edge).dest        ==
e.Next().Value.(int) {
                                    dist += f.Value.(*Edge).weight
                              }
                        }
                  }
                  return &Item{
                        Value:      currentNodeNumber,
                        Priority:   float32(dist),
                        PassedNode: *currentPassed,
                        Index:      0,
                  }
            } else {
                  if !itemInList(*currentItem, *visited) {
                        visited.PushBack(currentItem)
                        for e := g.adj[currentNodeNumber].Front(); e !=
nil; e = e.Next() {
                              tmp    :=    &Item{e.Value.(*Edge).dest,
currentNodeCost + e.Value.(*Edge).weight, *currentPassed, len(pq)}
                              heap.Push(&pq, tmp)
                              pq.Update(tmp,                tmp.Value,
currentNodeCost+e.Value.(*Edge).weight+expandedWeight[tmp.Value])
                        }
                  }
            }
      }
      return &Item{
            Value:      -1,
            Priority:   -1,
            PassedNode: list.List{},
            Index:      -1,
      }
}
```

2.5. Berkas UCS.go

```
package lib

import (
"container/heap"
"container/list"
)

func UCS(g Graph, startNode int, goalNode int) *Item {
nodeQueue := make(PriorityQueue, 0)
heap.Init(&nodeQueue)
heap.Push(&nodeQueue, &Item{startNode, 0, list.List{}, 0})
visited := list.New()
//fmt.Println("nq", nodeQueue.Len())
for nodeQueue.Len() > 0 {
	currentItem := heap.Pop(&nodeQueue).(*Item)
	currentNodeNumber := currentItem.Value
	currentNodeCost := currentItem.Priority
	currentPassed := list.New()
	currentPassed.PushBackList(&currentItem.PassedNode)
	currentPassed.PushBack(currentNodeNumber)
	//fmt.Println("currnod", currentNodeNumber, currentNodeCost)
	//for e := currentPassed.Front(); e != nil && e.Value != nil; e =
e.Next() {
	//	fmt.Println("v", e.Value)
	//}
	if currentNodeNumber == goalNode {
		return &Item{
			Value:	currentNodeNumber,
			Priority:	currentNodeCost,
			PassedNode: *currentPassed,
			Index:	0,
		}
	} else {
		if !itemInList(*currentItem, *visited) {
			visited.PushBack(currentItem)
			for e := g.adj[currentNodeNumber].Front(); e != nil;
e = e.Next() {
				//fmt.Println("1", e.Value.(*Edge).dest)
				tmp	:=	&Item{e.Value.(*Edge).dest,
currentNodeCost + e.Value.(*Edge).weight, *currentPassed, len(nodeQueue)}
				heap.Push(&nodeQueue, tmp)
				nodeQueue.Update(tmp,		tmp.Value,
currentNodeCost+e.Value.(*Edge).weight)
			}
		}
	}
}
return &Item{
	Value:	-1,
	Priority:	-1,
	PassedNode: list.List{},
	Index:	-1,
}
//PrintGraphInfos(g)
}
```

2.6. Berkas main.go

```
package main

import (
	"fmt"
	"main/lib"
)

func main() {
	g, x, y := lib.FileNameParse()
	lib.PrintGraphInfos(&g)
	fmt.Println("Do you want to use a* or ucs?")
	fmt.Println("1. A*")
	fmt.Println("2. UCS")
	algo := lib.RangedInput(1, 2)
	fmt.Println("Below are the names of the nodes:")
	for i := 0; i < len(lib.GetName(g)); i++ {
		fmt.Println(i, lib.GetName(g)[i])
	}
	fmt.Println("Please enter the start node:")
	start := lib.RangedInput(0, len(lib.GetName(g))-1)
	fmt.Println("Please enter the end node:")
	end := lib.RangedInput(0, len(lib.GetName(g))-1)
	var res *lib.Item
	if algo == 1 {
		res = lib.Astar(g, x, y, start, end)
	} else {
		res = lib.UCS(g, start, end)
	}
	lib.PrintPath(g, *res)
}
```

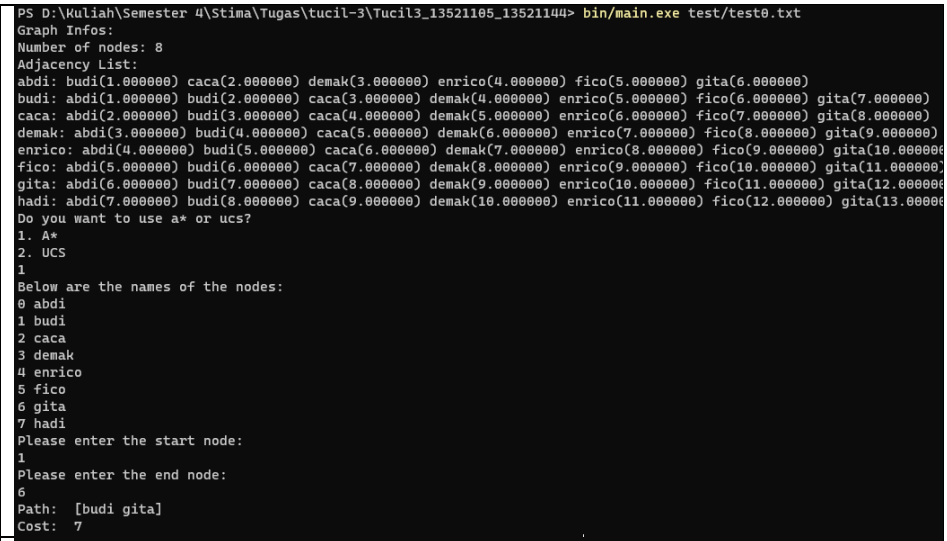
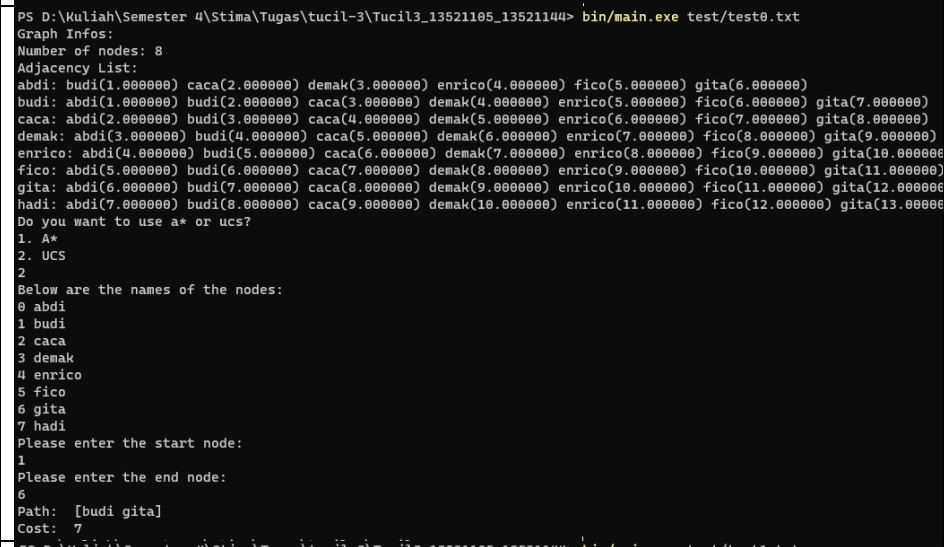
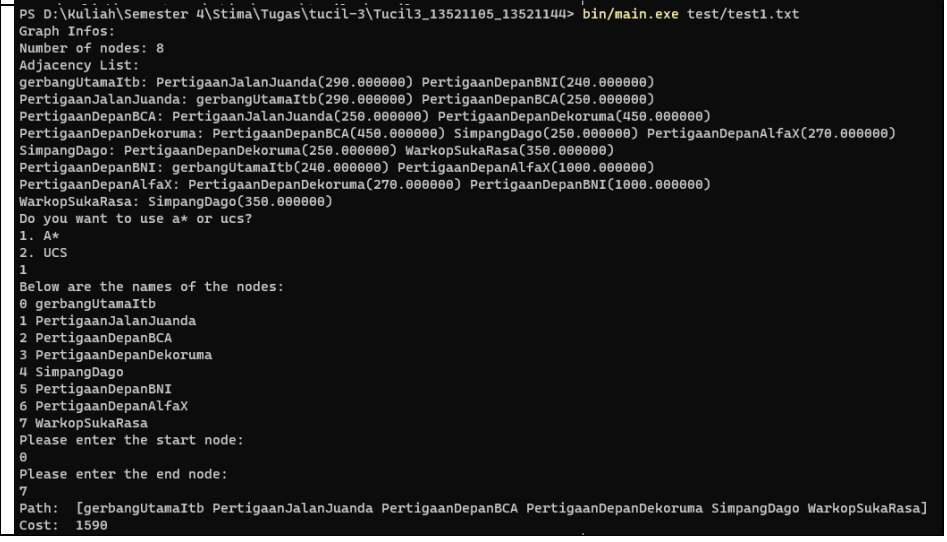
3. HASIL EKSEKUSI PROGRAM

Kasus	Tangkapan Layar
Tidak ada Berkas Masukan	`PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe` `2023/04/12 02:32:00 invalid argument number`
Berkas Masukan Tidak Ada	`PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test6.tx` `2023/04/12 02:32:54 open test/test6.txt: The system cannot find the file specified.`
Berkas Masukan Tidak Valid	`PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test5.t` `2023/04/12 02:32:33 invalid node count`

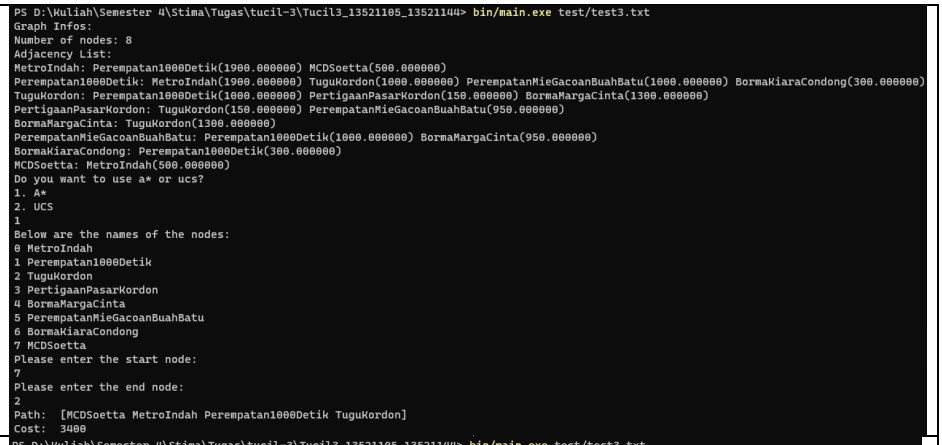
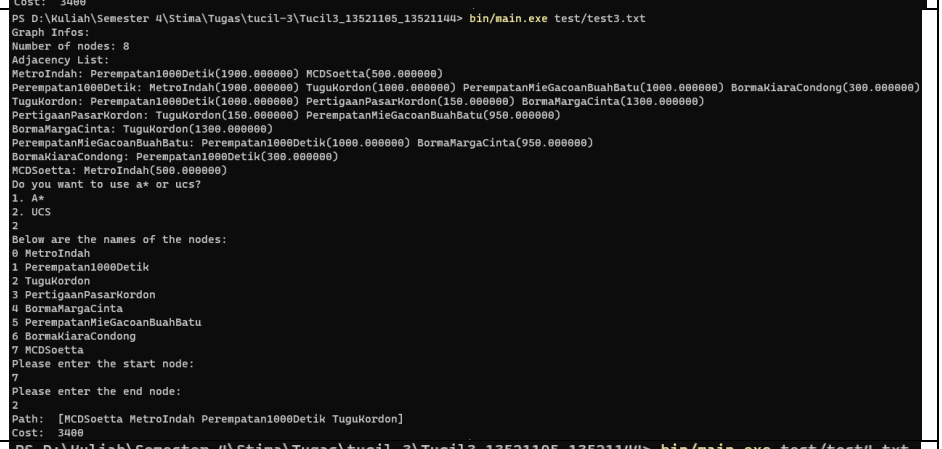
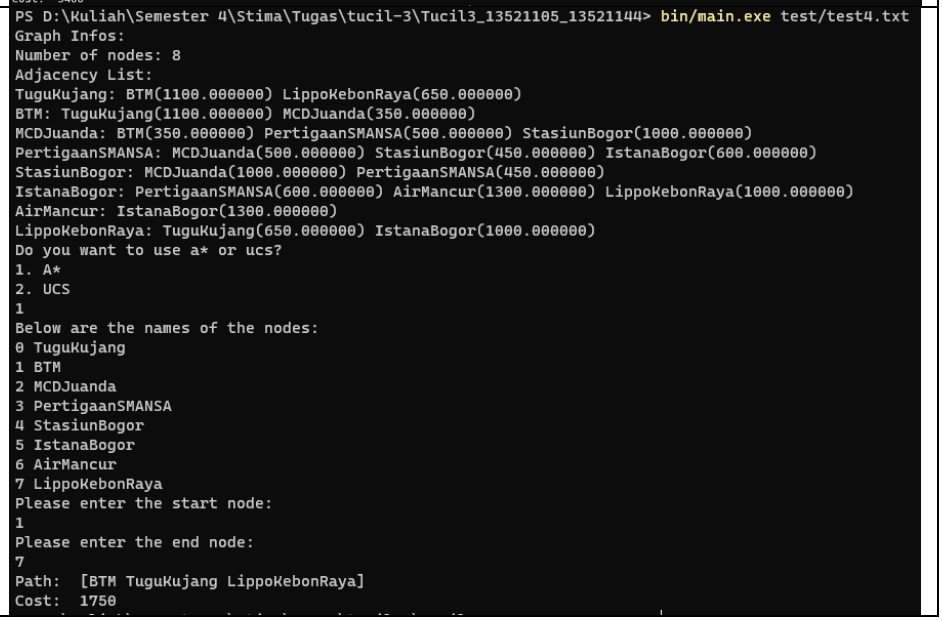
Berkas test.txt dari n0 ke n8 menggunakan algoritma UCS	`PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test.`
	`Graph Infos:`
	`Number of nodes: 8`
	`Adjacency List:`
	`n1: n2(2.000000) n3(3.000000)`
	`n2: n1(2.000000) n4(4.000000)`
	`n3: n1(3.000000) n2(3.000000) n5(5.000000)`
	`n4: n2(4.000000) n3(5.000000) n5(6.000000)`
	`n5: n4(6.000000) n6(7.000000)`
	`n6: n5(7.000000) n7(8.000000)`
	`n7: n6(8.000000)`
	`n8: No edge`
	`Do you want to use a* or ucs?`
	`1. A*`
	`2. UCS`
	`2`
	`Below are the names of the nodes:`
	`0 n1`
	`1 n2`
	`2 n3`
	`3 n4`
	`4 n5`
	`5 n6`
	`6 n7`
	`7 n8`
	`Please enter the start node:`
	`0`
	`Please enter the end node:`
	`7`
	`No path found`

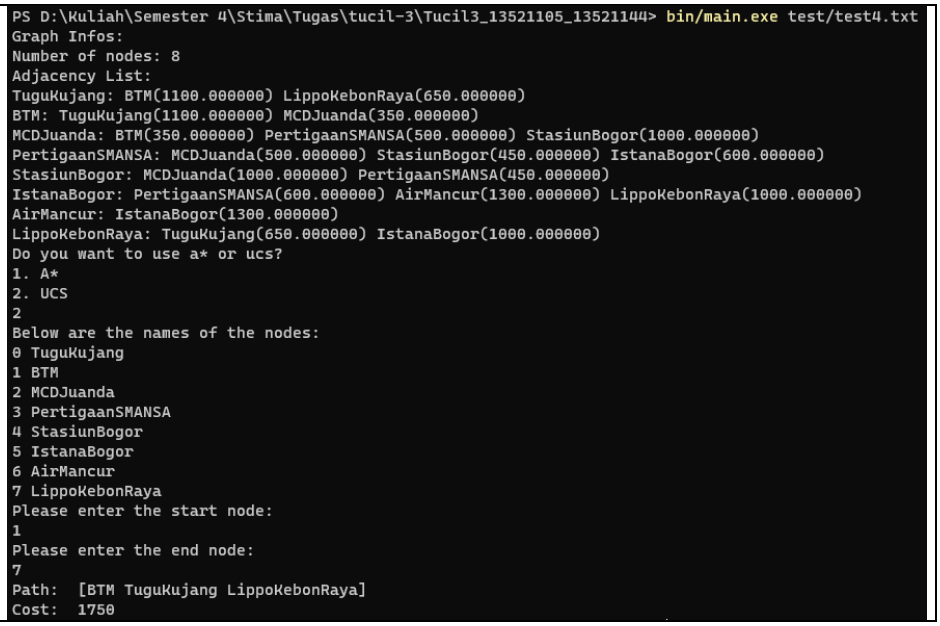
Berkas test.txt dari n0 ke n8 menggunakan algoritma A*	`PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test.`
	`Graph Infos:`
	`Number of nodes: 8`
	`Adjacency List:`
	`n1: n2(2.000000) n3(3.000000)`
	`n2: n1(2.000000) n4(4.000000)`
	`n3: n1(3.000000) n2(3.000000) n5(5.000000)`
	`n4: n2(4.000000) n3(5.000000) n5(6.000000)`
	`n5: n4(6.000000) n6(7.000000)`
	`n6: n5(7.000000) n7(8.000000)`
	`n7: n6(8.000000)`
	`n8: No edge`
	`Do you want to use a* or ucs?`
	`1. A*`
	`2. UCS`
	`1`
	`Below are the names of the nodes:`
	`0 n1`
	`1 n2`
	`2 n3`
	`3 n4`
	`4 n5`
	`5 n6`
	`6 n7`
	`7 n8`
	`Please enter the start node:`
	`0`
	`Please enter the end node:`
	`7`
	`No path found`

Berkas test.txt dari n0 ke n7 menggunakan algoritma A*	PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/ Graph Infos: Number of nodes: 8 Adjacency List: n1: n2(2.000000) n3(3.000000) n2: n1(2.000000) n4(4.000000) n3: n1(3.000000) n2(3.000000) n5(5.000000) n4: n2(4.000000) n3(5.000000) n5(6.000000) n5: n4(6.000000) n6(7.000000) n6: n5(7.000000) n7(8.000000) n7: n6(8.000000) n8: No edge Do you want to use a* or ucs? 1. A* 2. UCS 1 Below are the names of the nodes: 0 n1 1 n2 2 n3 3 n4 4 n5 5 n6 6 n7 7 n8 Please enter the start node: 0 Please enter the end node: 6 Path: [n1 n3 n5 n6 n7] Cost: 23
Berkas test.txt dari n0 ke n7 menggunakan algoritma UCS	PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/ Graph Infos: Number of nodes: 8 Adjacency List: n1: n2(2.000000) n3(3.000000) n2: n1(2.000000) n4(4.000000) n3: n1(3.000000) n2(3.000000) n5(5.000000) n4: n2(4.000000) n3(5.000000) n5(6.000000) n5: n4(6.000000) n6(7.000000) n6: n5(7.000000) n7(8.000000) n7: n6(8.000000) n8: No edge Do you want to use a* or ucs? 1. A* 2. UCS 2 Below are the names of the nodes: 0 n1 1 n2 2 n3 3 n4 4 n5 5 n6 6 n7 7 n8 Please enter the start node: 0 Please enter the end node: 6 Path: [n1 n3 n5 n6 n7] Cost: 23

Berkas test0.txt dari budi ke gita menggunakan algoritma A*	
Berkas test0.txt dari budi ke gita menggunakan algoritma UCS	
Berkas test1.txt dari gerbangUtamaITB ke WarkopSukaRasa menggunakan algoritma A*	

Berkas test1.txt dari gerbangUtamaITB ke WarkopSukaRasa menggunakan algoritma UCS	PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test1.txt Graph Infos: Number of nodes: 8 Adjacency List: gerbangUtamaItb: PertigaanJalanJuanda(290.000000) PertigaanDepanBNI(240.000000) PertigaanJalanJuanda: gerbangUtamaItb(290.000000) PertigaanDepanBCA(250.000000) PertigaanDepanBCA: PertigaanJalanJuanda(250.000000) PertigaanDepanDekoruma(450.000000) PertigaanDepanDekoruma: PertigaanDepanBCA(450.000000) SimpangDago(250.000000) PertigaanDepanAlfaX(270.000000) SimpangDago: PertigaanDepanDekoruma(250.000000) WarkopSukaRasa(350.000000) PertigaanDepanBNI: gerbangUtamaItb(240.000000) PertigaanDepanAlfaX(1000.000000) PertigaanDepanAlfaX: PertigaanDepanDekoruma(270.000000) PertigaanDepanBNI(1000.000000) WarkopSukaRasa: SimpangDago(350.000000) Do you want to use a* or ucs? 1. A* 2. UCS 2 Below are the names of the nodes: 0 gerbangUtamaItb 1 PertigaanJalanJuanda 2 PertigaanDepanBCA 3 PertigaanDepanDekoruma 4 SimpangDago 5 PertigaanDepanBNI 6 PertigaanDepanAlfaX 7 WarkopSukaRasa Please enter the start node: 0 Please enter the end node: 7 Path: [gerbangUtamaItb PertigaanJalanJuanda PertigaanDepanBCA PertigaanDepanDekoruma SimpangDago WarkopSukaRasa] Cost: 1590
Berkas test2.txt dari SimpangBraga ke Norsefiicden menggunakan algoritma A*	PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test2.txt Graph Infos: Number of nodes: 8 Adjacency List: AlunAlun: PerigaanKAASukarno(180.000000) Norsefiicden(350.000000) MuseumKAA: PertigaanJlBragaKaa(71.000000) PerigaanKAASukarno(44.000000) PertigaanJlBragaKaa: MuseumKAA(71.000000) SimpangBraga(190.000000) SimpangBraga: PertigaanJlBragaKaa(190.000000) PertigaanDepanBTN(110.000000) PertigaanDepanBTN: SimpangBraga(110.000000) PerigaanKAASukarno(180.000000) PerigaanKAASukarno: AlunAlun(180.000000) MuseumKAA(44.000000) PertigaanDepanBTN(180.000000) Norsefiicden: AlunAlun(350.000000) StasiunTimur(650.000000) StasiunTimur: Norsefiicden(650.000000) Do you want to use a* or ucs? 1. A* 2. UCS 1 Below are the names of the nodes: 0 AlunAlun 1 MuseumKAA 2 PertigaanJlBragaKaa 3 SimpangBraga 4 PertigaanDepanBTN 5 PerigaanKAASukarno 6 Norsefiicden 7 StasiunTimur Please enter the start node: 3 Please enter the end node: 6 Path: [SimpangBraga PertigaanDepanBTN PerigaanKAASukarno AlunAlun Norsefiicden] Cost: 820
Berkas test2.txt dari SimpangBraga ke Norsefiicden menggunakan algoritma UCS	PS D:\Kuliah\Semester 4\Stima\Tugas\tucil-3\Tucil3_13521105_13521144> bin/main.exe test/test2.txt Graph Infos: Number of nodes: 8 Adjacency List: AlunAlun: PerigaanKAASukarno(180.000000) Norsefiicden(350.000000) MuseumKAA: PertigaanJlBragaKaa(71.000000) PerigaanKAASukarno(44.000000) PertigaanJlBragaKaa: MuseumKAA(71.000000) SimpangBraga(190.000000) SimpangBraga: PertigaanJlBragaKaa(190.000000) PertigaanDepanBTN(110.000000) PertigaanDepanBTN: SimpangBraga(110.000000) PerigaanKAASukarno(180.000000) PerigaanKAASukarno: AlunAlun(180.000000) MuseumKAA(44.000000) PertigaanDepanBTN(180.000000) Norsefiicden: AlunAlun(350.000000) StasiunTimur(650.000000) StasiunTimur: Norsefiicden(650.000000) Do you want to use a* or ucs? 1. A* 2. UCS 2 Below are the names of the nodes: 0 AlunAlun 1 MuseumKAA 2 PertigaanJlBragaKaa 3 SimpangBraga 4 PertigaanDepanBTN 5 PerigaanKAASukarno 6 Norsefiicden 7 StasiunTimur Please enter the start node: 3 Please enter the end node: 6 Path: [SimpangBraga PertigaanDepanBTN PerigaanKAASukarno AlunAlun Norsefiicden] Cost: 820

Berkas test3.txt dari MCDSoetta ke TuguKordon menggunakan algoritma A*	
Berkas test3.txt dari MCDSoetta ke TuguKordon menggunakan algoritma UCS	
Berkas test4.txt dari BTM ke LippoKebun Raya menggunakan algoritma A*	

Berkas test4.txt dari BTIM ke LippoKebun Raya menggunakan algoritma UCS	

4. KESIMPULAN DAN KOMENTAR

4.1. Kesimpulan

1. Algoritma A* dan UCS memberikan solusi yang optimal
2. Algoritma A* dan UCS dapat mendeteksi bila tidak ada jalur
3. Algoritma A* dan UCS adalah algoritma yang *complete*

4.2. Komentar

1. Tugas kecil menarik dengan bonus menggunakan API Google Maps
2. Bonus menggunakan API relatif menyulitkan bagi mahasiswa yang tidak memiliki akses ke kartu kredit

5. LAMPIRAN

5.1. Pranala *Repository GitHub*
https://github.com/Marthenn/Tucil3_13521105_13521144

5.2. Tabel Ketercapaian Program

Poin	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		