Tugas Kecil 3 IF2211 Strategi Algoritma
Semester II tahun 2022/2023
Implementasi Algoritma UCS dan A* untuk Menentukan Lintasan Terpendek

Disusun oleh:
Kelompok 101
Jericho Russel Sebastian 13521107
Asyifa Nurul Shafira 13521125

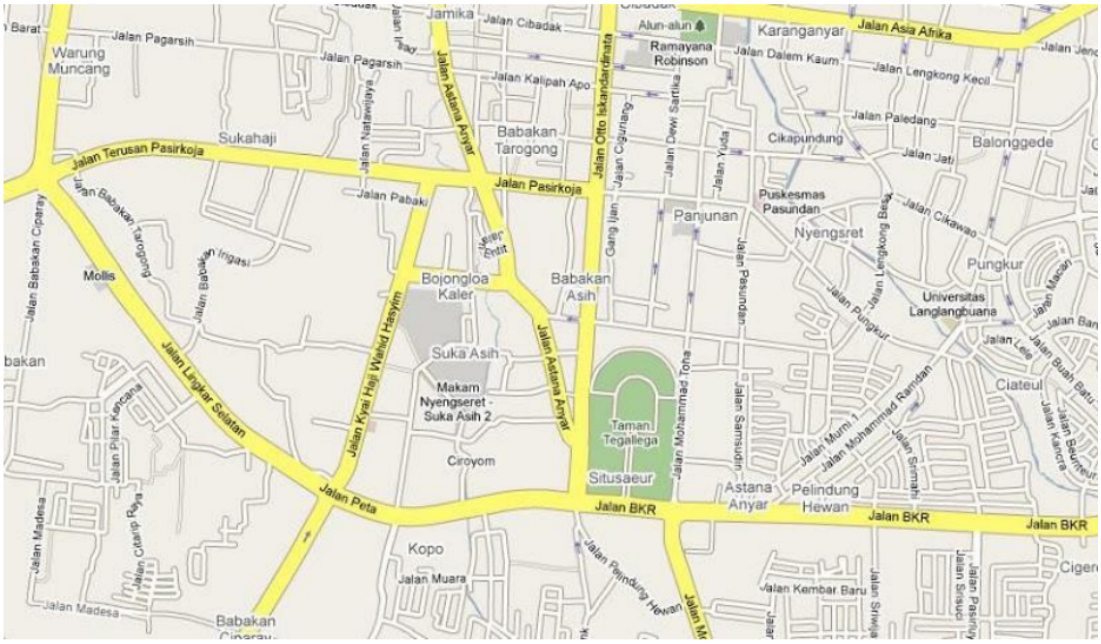
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

DAFTAR ISI

Bab 1

Deskripsi Persoalan

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 beserta jaraknya antara simpul asal dan simpul tujuan.

5. Antarmuka program bebas, apakah pakai GUI atau command line saja.
   Bonus: Bonus nilai diberikan jika dapat menggunakan Google Map API untuk menampilkan peta, membentuk graf dari peta, dan menampilkan lintasan terpendek di peta (berupa jalan yang diberi warna). Simpul graf diperoleh dari peta (menggunakan API Google Map) dengan mengklik ujung jalan atau persimpangan jalan, lalu jarak antara kedua simpul dihitung langsung dengan rumus Euclidean.

Bab 2

Implementasi Program

2.1 GraphTraversalAlgorithm.cs

```
using System;
using System.Collections.Generic;
using PathfindAllDay.Structs;

namespace PathfindAllDay.Algorithms {
    /// <summary>
    /// A class that contains a traversal algorithm over a graph instance, with modifiable g- and h-function.
    /// </summary>
    /// <typeparam name="T">Type of the graph node.</typeparam>
    public class GraphTraversalAlgorithm<T> {
        /// <summary>The internal reference to the graph.</summary>
        readonly DirectedGraph<T, double> _graph;
        /// <summary>An internal queue of open nodes.</summary>
        readonly PriorityQueue<GraphTraversalNode<T>> _open;
        /// <summary>An internal map of visited nodes.</summary>
        readonly Dictionary<T, GraphTraversalNode<T>> _closed;

        /// <summary>The g-function of the traversal algorithm.</summary>
        public Func<Info, double> GFunction { get; set; } = info => 0d;
        /// <summary>The h-function of the traversal algorithm.</summary>
        public Func<Info, double> HFunction { get; set; } = info => 0d;

        /// <summary>
        /// Instantiates a traversal algorithm over the given graph.
        /// </summary>
        /// <param name="graph">The graph to traverse.</param>
        /// <param name="minHeap">Whether to use min-heap to sort expand nodes.</param>
        public GraphTraversalAlgorithm(DirectedGraph<T, double> graph, bool minHeap = false) {
            _graph = graph;
            _open = new PriorityQueue<GraphTraversalNode<T>>(_graph.NodeCount, minHeap);
            _closed = new Dictionary<T, GraphTraversalNode<T>>();
        }

        /// <summary>
        /// Performs a traversal search over the graph to find a path that leads from <paramref name="start"/>
to <paramref name="end"/>.
        /// </summary>
        /// <param name="start">The start node of the path.</param>
        /// <param name="end">The end node of the path.</param>
```

```
        /// <returns>
        ///     An array containing the path from <paramref name="start"/> to <paramref name="end"/> if such path is found within the graph;
        ///     <see langword="null"/> otherwise.
        /// </returns>
        /// <exception cref="ArgumentNullException">Thrown when either <paramref name="start"/> or <paramref name="end"/> is <see langword="null"/>.</exception>
        /// <exception cref="ArgumentException">Thrown when either <paramref name="start"/> or <paramref name="end"/> is not contained in the graph.</exception>
        public T[] FindPath(T start, T end) {
            if(start == null || end == null) throw new ArgumentNullException();
            if(!_graph.ContainsNode(start) || !_graph.ContainsNode(end)) throw new ArgumentException("Node is not in the graph.");

            Console.WriteLine($">>> Beginning search, from: {start}, to: {end}");
            _open.Clear(); _closed.Clear();

            // Add the starting node to the open queue.
            bool success = false;
            GraphTraversalNode<T> expandNode = new GraphTraversalNode<T>(start, null);
            _open.TryEnqueue(expandNode);

            Console.WriteLine($"  Queue: {_open}");
            // Dequeue a node from the open queue and add it to the closed set. Repeat until the open set is empty.
            while(_open.TryDequeue(out expandNode)) {
                Console.WriteLine($"    Selected: {expandNode.Value}");
                _closed.Add(expandNode.Value, expandNode);

                // If the end node is reached, then a path is found: bail out early from the loop.
                success = expandNode.Value.Equals(end);
                if(success) break;

                // If not, then enumerate through the node's open neighbors...
                foreach(GraphEdge<T, double> expandEdge in _graph.OutEdges(expandNode.Value)) {
                    if(_closed.ContainsKey(expandEdge.To)) continue;

                    // ...and calculate its g-cost.
                    double gCost = GFunction(new Info(start, end, expandNode, expandEdge));
                    GraphTraversalNode<T> neighborNode = _open.Find(node => node.Value.Equals(expandEdge.To));
                    Console.WriteLine($"      Evaluating node {expandEdge.To} -> {expandEdge.From}, got {gCost}{(neighborNode != null ? " versus " + neighborNode.GCost : "")}");
```

```
                // If the neighbor hasn't been visited yet or the calculated g-cost is smaller than the stored g-cost...
                if(neighborNode == null || gCost < neighborNode.GCost) {
                    if(neighborNode == null)
                        neighborNode = new GraphTraversalNode<T>(expandEdge.To, expandNode);

                    // ...update the neighbor's g- and h-cost...
                    neighborNode.GCost = gCost;
                    neighborNode.HCost = HFunction(new Info(start, end, expandNode, expandEdge));
                    neighborNode.Parent = expandNode;

                    // ...and update the neighbor's order in the open queue.
                    if(!_open.Contains(neighborNode))
                        _open.TryEnqueue(neighborNode);
                    else
                        _open.Update(neighborNode);
                }
            }
            Console.WriteLine($"  Queue: {_open}");
        }

        T[] result = success ? expandNode.Backtrack() : null;

        Console.WriteLine($"<<< Result: {(success ? $"success [{string.Join(", ", result)}]" : "fail")}");
        return result;
    }

    /// <summary>
    /// Contains information about a node expansion step within a search instance.
    /// </summary>
    public readonly struct Info {
        /// <summary>The start node of the searched path.</summary>
        public T Start { get; }
        /// <summary>The end node of the searched path.</summary>
        public T End { get; }
        /// <summary>The currently expanded node.</summary>
        public GraphTraversalNode<T> ExpandNode { get; }
        /// <summary>The edge over which the current node is expanded.</summary>
        public GraphEdge<T, double> ExpandEdge { get; }

        /// <summary>
        /// Constructs an <see cref="Info"/> instance that contains the given information.
        /// </summary>
        /// <param name="start">The start node of the searched path.</param>
        /// <param name="end">The end node of the searched path.</param>
```

```
        /// <param name="expandNode">The currently expanded node.</param>
        /// <param name="expandEdge">The edge over which the current node is expanded.</param>
        public Info(T start, T end, GraphTraversalNode<T> expandNode, GraphEdge<T, double>
expandEdge) {
            Start = start; End = end; ExpandNode = expandNode; ExpandEdge = expandEdge;
        }
    }
  }

  /// <summary>
  /// Stores information about a node in the search tree.
  /// </summary>
  /// <typeparam name="T">Type of the graph node.</typeparam>
  public class GraphTraversalNode<T> : IPriorityQueueItem<GraphTraversalNode<T>> {
    /// <summary>The relevant graph node.</summary>
    public T Value { get; }
    /// <summary>The parent of this node in the search tree.</summary>
    public GraphTraversalNode<T> Parent { get; set; }
    /// <summary>The stored g-cost of the node.</summary>
    public double GCost { get; set; }
    /// <summary>The stored h-cost of the node.</summary>
    public double HCost { get; set; }
    public int QueueIndex { get; set; }

    /// <summary>The stored f-cost of the node.</summary>
    public double FCost => GCost + HCost;

    /// <summary>
    /// Constructs a traversal node.
    /// </summary>
    /// <param name="value">The relevant graph node.</param>
    /// <param name="parent">The parent of this node in the search tree.</param>
    public GraphTraversalNode(T value, GraphTraversalNode<T> parent) {
      Value = value; Parent = parent;
      GCost = 0f; HCost = 0f;
      QueueIndex = -1;
    }

    public int CompareTo(GraphTraversalNode<T> other) {
      return FCost.CompareTo(other.FCost);
    }

    /// <summary>
    /// Constructs a graph node array that leads from the root of the search tree to this node.
    /// </summary>
```

```
        /// <returns>An array containing the path from the search tree root to this node.</returns>
        public T[] Backtrack() {
            List<T> path = new List<T>();
            GraphTraversalNode<T> current = this;

            while(current != null) {
                path.Add(current.Value);
                current = current.Parent;
            }

            path.Reverse();
            return path.ToArray();
        }

        public override string ToString() {
            return $"({Value}{(Parent != null ? " -> " + Parent : "")}: {FCost})";
        }
    }
}
```

2.2 Graph.cs

```
using System;
using System.Collections.Generic;

namespace PathfindAllDay.Structs {
    /// <summary>
    /// Represents a graph with monodirectional edges.
    /// </summary>
    /// <typeparam name="TNode">Type of each node.</typeparam>
    /// <typeparam name="TEdgeData">Type of data stored in each edge.</typeparam>
    public class DirectedGraph<TNode, TEdgeData> {
        /// <summary>Stores each node and its adjacent nodes.</summary>
        private readonly Dictionary<TNode, LinkedList<TNode>> _adjacencyLists;
        /// <summary>Stores each edge and its stored data.</summary>
        private readonly Dictionary<(TNode, TNode), TEdgeData> _edges;

        /// <summary>The number of nodes stored in the graph.</summary>
        public int NodeCount => _adjacencyLists.Count;
        /// <summary>The number of edges stored in the graph.</summary>
        public int EdgeCount => _edges.Count;

        /// <summary>Constructs an empty graph.</summary>
        public DirectedGraph() {
            _edges = new Dictionary<(TNode, TNode), TEdgeData>();
            _adjacencyLists = new Dictionary<TNode, LinkedList<TNode>>();
```

```
    }

    /// <summary>
    /// Checks whether a node exists in the graph.
    /// </summary>
    /// <param name="node">The node to be checked.</param>
    /// <returns>Whether <paramref name="node"/> is contained in the graph.</returns>
    /// <exception cref="ArgumentNullException">Thrown when <paramref name="node"/> is <see
langword="null"/>.</exception>
    public bool ContainsNode(TNode node) {
        if(node == null) throw new ArgumentNullException();
        return _adjacencyLists.ContainsKey(node);
    }

    /// <summary>
    /// Checks whether the edge (<paramref name="from"/>, <paramref name="to"/>) exists in the graph.
    /// </summary>
    /// <param name="from">The source node of the edge.</param>
    /// <param name="to">The destination node of the edge.</param>
    /// <returns>Whether the edge (<paramref name="from"/>, <paramref name="to"/>) is contained in the
graph.</returns>
    /// <inheritdoc cref="ContainsNode(TNode)"/>
    public bool ContainsEdge(TNode from, TNode to) {
        return ContainsNode(from) && ContainsNode(to) && _edges.ContainsKey((from, to));
    }

    /// <summary>
    /// Attempts to find the edge (<paramref name="from"/>, <paramref name="to"/>) in the graph.
    /// </summary>
    /// <param name="from">The source node of the edge.</param>
    /// <param name="to">The destination node of the edge.</param>
    /// <param name="data">The data stored in the edge.</param>
    /// <returns>Whether the edge (<paramref name="from"/>, <paramref name="to"/>) is contained in the
graph.</returns>
    /// <exception cref="ArgumentNullException">Thrown when either <paramref name="from"/> or
<paramref name="to"/> is <see langword="null"/>.</exception>
    public bool TryGetEdge(TNode from, TNode to, out TEdgeData data) {
        if(from == null || to == null) {
            data = default;
            throw new ArgumentNullException();
        }
        return _edges.TryGetValue((from, to), out data);
    }

    /// <summary>
```

```
        /// Adds <paramref name="node"/> to the graph.
        /// </summary>
        /// <param name="node">The node to be added.</param>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="node"/> is <see
langword="null"/>.</exception>
        /// <exception cref="ArgumentException">Thrown when <paramref name="node"/> already exists in
the graph.</exception>
        public void AddNode(TNode node) {
            if(node == null) throw new ArgumentNullException();
            if(_adjacencyLists.ContainsKey(node)) throw new ArgumentException($"Node {node} already
exists in this graph.");

            _adjacencyLists.Add(node, null);
        }

        /// <summary>
        /// Adds the edge (<paramref name="from"/>, <paramref name="to"/>) to the graph.
        /// </summary>
        /// <param name="from">The source node of the edge.</param>
        /// <param name="to">The destination node of the edge.</param>
        /// <param name="data">The data stored in the edge.</param>
        /// <exception cref="ArgumentNullException">Thrown when either <paramref name="from"/> or
<paramref name="to"/> is <see langword="null"/>.</exception>
        /// <exception cref="ArgumentException">Thrown when <paramref name="from"/> equals <paramref
name="to"/>.</exception>
        public void AddEdge(TNode from, TNode to, TEdgeData data) {
            if(from == null || to == null) throw new ArgumentNullException();
            if(from.Equals(to)) throw new ArgumentException();

            if(!ContainsNode(from)) AddNode(from);
            if(!ContainsNode(to)) AddNode(to);

            _edges.Add((from, to), data);
            (_adjacencyLists[from] ?? (_adjacencyLists[from] = new LinkedList<TNode>())).AddFirst(to);
        }

        /// <summary>
        /// Adds the edge described by <paramref name="edge"/> to the graph.
        /// </summary>
        /// <param name="edge">The edge description.</param>
        /// <inheritdoc cref="AddEdge(TNode, TNode, TEdgeData)"/>
        public void AddEdge(GraphEdge<TNode, TEdgeData> edge) {
            AddEdge(edge.From, edge.To, edge.Data);
        }
```

```
        /// <summary>
        /// Enumerates through all the nodes stored in the graph.
        /// </summary>
        /// <returns>An <see cref="IEnumerable{T}"/> that enumerates through all the nodes stored in the graph.</returns>
        public IEnumerable<TNode> Nodes() {
            foreach(TNode node in _adjacencyLists.Keys)
                yield return node;
        }

        /// <summary>
        /// Enumerates through all edges leading out of <paramref name="from"/>.
        /// </summary>
        /// <param name="from">The source node of the edges.</param>
        /// <returns>An <see cref="IEnumerable{T}"/> that enumerates through all edges leading out of <paramref name="from"/>.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="from"/> is <see langword="null"/>.</exception>
        /// <exception cref="ArgumentException">Thrown when <paramref name="from"/> doesn't exist in the graph.</exception>
        public IEnumerable<GraphEdge<TNode, TEdgeData>> OutEdges(TNode from) {
            if(from == null) throw new ArgumentNullException();
            if(!ContainsNode(from)) throw new ArgumentException($"Graph doesn't contain the node {from}");

            if(_adjacencyLists[from] != null)
                foreach(TNode to in _adjacencyLists[from])
                    yield return new GraphEdge<TNode, TEdgeData>(from, to, _edges[(from, to)]);
        }
    }

    /// <summary>
    /// Stores all information relating to a graph edge.
    /// </summary>
    /// <typeparam name="TNode"></typeparam>
    /// <typeparam name="TEdgeData"></typeparam>
    public readonly struct GraphEdge<TNode, TEdgeData> {
        /// <summary>The source node of the edge.</summary>
        public TNode From { get; }
        /// <summary>The destination node of the edge.</summary>
        public TNode To { get; }
        /// <summary>The data stored in the edge.</summary>
        public TEdgeData Data { get; }

        /// <summary>
```

```
        /// Constructs a <see cref="GraphEdge{TNode, TEdgeData}"/> that describes the edge (<paramref
name="from"/>, <paramref name="to"/>).
        /// </summary>
        /// <param name="from">The source node of the edge.</param>
        /// <param name="to">The destination node of the edge.</param>
        /// <param name="data">The data stored in the edge.</param>
        public GraphEdge(TNode from, TNode to, TEdgeData data) {
            From = from; To = to; Data = data;
        }
    }
}
```

2.3 PriorityQueue.cs

```
using System;
using System.Text;

namespace PathfindAllDay.Structs {
    /// <summary>
    /// Represents a queue where items are ordered by priority, internally implemented using a heap.
    /// </summary>
    /// <typeparam name="T">Type of queue items.</typeparam>
    public class PriorityQueue<T> where T : IPriorityQueueItem<T> {
        /// <summary>Internal buffer that stores queue items.</summary>
        private readonly T[] _buffer;

        /// <summary>The maximum capacity of the queue.</summary>
        public int Capacity => _buffer.Length;
        /// <summary>The number of items in the queue.</summary>
        public bool IsMinHeap { get; }
        public int Count { get; private set; }

        /// <summary>
        /// Constructs an empty queue.
        /// </summary>
        /// <param name="capacity">The maximum capacity of the queue.</param>
        /// <param name="minHeap">Whether to sort the queue using min-heap rule.</param>
        /// <exception cref="ArgumentOutOfRangeException">Thrown when <paramref name="capacity"/> is
negative.</exception>
        public PriorityQueue(int capacity, bool minHeap = false) {
            if(capacity < 0)throw new ArgumentOutOfRangeException(nameof(capacity));

            _buffer = new T[capacity];
            IsMinHeap = minHeap;
            Count = 0;
        }
```

```
        /// <summary>
        /// Clears the queue.
        /// </summary>
        public void Clear() {
            for(int i = 0; i < Capacity; i++) _buffer[i] = default;
            Count = 0;
        }

        /// <summary>
        /// Checks if <paramref name="item"/> is contained in the queue.
        /// </summary>
        /// <param name="item">The item to be checked.</param>
        /// <returns>Whether <paramref name="item"/> is contained in the queue.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="item"/> is <see
langword="null"/>.</exception>
        public bool Contains(T item) {
            if(item == null) throw new ArgumentNullException();
            return _buffer.GetLowerBound(0) <= item.QueueIndex && item.QueueIndex < Count &&
item.Equals(_buffer[item.QueueIndex]);
        }

        /// <summary>
        /// Updates <paramref name="item"/>'s order in the queue. This should be called on every modification
to <paramref name="item"/>'s comparison attribute.
        /// </summary>
        /// <param name="item">The item to be updated.</param>
        public void Update(T item) {
            SortUp(item);
            SortDown(item);
        }

        /// <summary>
        /// Checks whether any item in the queue satisfies a condition.
        /// </summary>
        /// <param name="pred">The condition to check.</param>
        /// <returns>Whether at least one item satisfies <paramref name="pred"/>.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="pred"/> is <see
langword="null"/>.</exception>
        public bool Any(Predicate<T> pred) {
            if(pred == null) throw new ArgumentNullException();
            for(int i = 0; i < Count; i++) if(pred(_buffer[i])) return true;
            return false;
        }
```

```
        /// <summary>
        /// Checks whether all item in the queue satisfies a condition.
        /// </summary>
        /// <param name="pred">The condition to check.</param>
        /// <returns>Whether all item satisfies <paramref name="pred"/>.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="pred"/> is <see
langword="null"/>.</exception>
        public bool All(Predicate<T> pred) {
            if(pred == null) throw new ArgumentNullException();
            for(int i = 0; i < Count; i++) if(pred(_buffer[i])) return false;
            return true;
        }

        /// <summary>
        /// Finds the first item in the queue that satisfies a condition.
        /// </summary>
        /// <param name="pred">The condition to check.</param>
        /// <returns>The first item in the queue that satisfies <paramref name="pred"/>.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="pred"/> is <see
langword="null"/>.</exception>
        public T Find(Predicate<T> pred) {
            if(pred == null) throw new ArgumentNullException();
            for(int i = 0; i < Count; i++) if(pred(_buffer[i])) return _buffer[i];
            return default;
        }

        /// <summary>
        /// Attempts to enqueue an item into the queue.
        /// </summary>
        /// <param name="item">The item to be enqueued.</param>
        /// <returns>Whether <paramref name="item"/> is successfully enqueued.</returns>
        /// <exception cref="ArgumentNullException">Thrown when <paramref name="item"/> is <see
langword="null"/>.</exception>
        public bool TryEnqueue(T item) {
            if(item == null)throw new ArgumentNullException();
            if(Count == Capacity) return false;

            (_buffer[Count] = item).QueueIndex = Count;
            SortUp(item);
            Count++;

            return true;
        }

        /// <summary>
```

```
    /// Attempts to dequeue an item from the queue.
    /// </summary>
    /// <param name="item">The dequeued item.</param>
    /// <returns>Whether <paramref name="item"/> is succesfully dequeued.</returns>
    public bool TryDequeue(out T item) {
        if(Count == 0) {
            item = default;
            return false;
        }

        item = _buffer[0];
        Count--;
        (_buffer[0] = _buffer[Count]).QueueIndex = 0;
        SortDown(_buffer[0]);

        return true;
    }

    /// <summary>
    /// Sorts <paramref name="item"/> upwards in the heap.
    /// </summary>
    /// <param name="item">The item to be sorted upwards.</param>
    private void SortUp(T item) {
        int parentIndex = (item.QueueIndex - 1) / 2;

        while(parentIndex >= 0) {
            T parentItem = _buffer[parentIndex];
            int compare = item.CompareTo(parentItem);
            if(!IsMinHeap ? compare > 0 : compare < 0) {
                Swap(item, parentItem);
            } else return;
            parentIndex = (item.QueueIndex - 1) / 2;
        }
    }

    /// <summary>
    /// Sorts <paramref name="item"/> downwards in the heap.
    /// </summary>
    /// <param name="item">The item to be sorted downwards.</param>
    private void SortDown(T item) {
        while(true) {
            int childIndexLeft = item.QueueIndex * 2 + 1;
            int childIndexRight = item.QueueIndex * 2 + 2;
            int swapIndex;
```

```
            if(childIndexLeft < Count) {
                swapIndex = childIndexLeft;
                if(childIndexRight < Count) {
                    int compareChild = _buffer[childIndexLeft].CompareTo(_buffer[childIndexRight]);
                    if(!IsMinHeap ? compareChild < 0 : compareChild > 0) {
                        swapIndex = childIndexRight;
                    }
                }

                int compare = item.CompareTo(_buffer[swapIndex]);
                if(!IsMinHeap ? compare < 0 : compare > 0) {
                    Swap(item, _buffer[swapIndex]);
                } else return;
            } else return;
        }
    }

    /// <summary>
    /// Swaps <paramref name="a"/> and <paramref name="b"/> in the heap.
    /// </summary>
    /// <param name="a">The first item.</param>
    /// <param name="b">The second item.</param>
    private void Swap(T a, T b) {
        _buffer[a.QueueIndex] = b;
        _buffer[b.QueueIndex] = a;

        (a.QueueIndex, b.QueueIndex) = (b.QueueIndex, a.QueueIndex);
    }

    public override string ToString() {
        StringBuilder str = new StringBuilder();
        str.Append("[");
        for(int i = 0; i < Count; i++) {
            str.Append(_buffer[i].ToString());
            if(i < Count - 1) str.Append(", ");
        }
        str.Append("]");
        return str.ToString();
    }
}

/// <summary>
/// Denotes that a type can be treated as a queue item.
/// </summary>
/// <typeparam name="T"></typeparam>
```

```
    public interface IPriorityQueueItem<T> : IComparable<T> {
        /// <summary>The buffer index of the queue item.</summary>
        int QueueIndex { get; set; }
    }
}
```

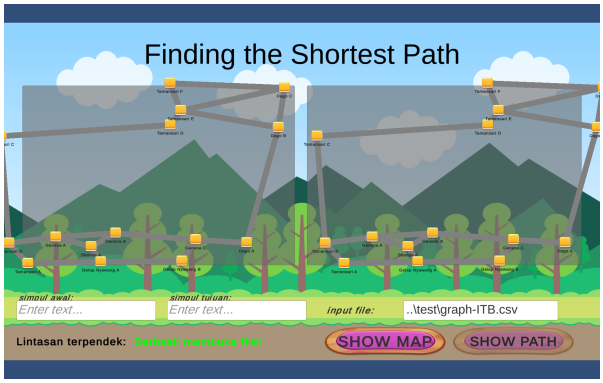
Bab 3

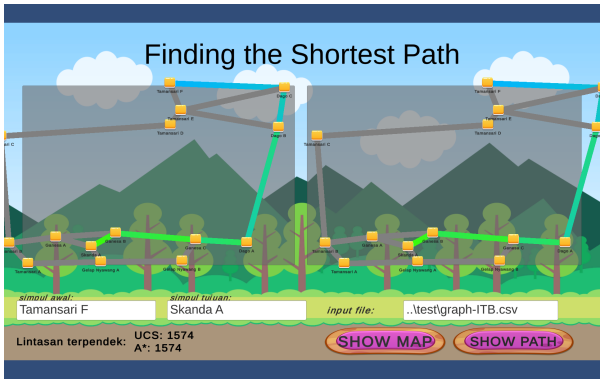
Eksekusi Program

3.1 Peta jalan sekitar kampus ITB

<table>
<tr><th>Input</th><th>Hasil</th></tr>
<tr><td>graph-ITB.csv

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=pair
N,107.608783,-6.8948617,Tamansari A
N,107.60842,-6.8937402,Tamansari B
N,107.6082665,-6.8878702,Tamansari C
N,107.6114806,-6.8872532,Tamansari D
N,107.6116764,-6.886465,Tamansari E
N,107.6114718,-6.8849629,Tamansari F
N,107.6093053,-6.8934058,Ganesa A
N,107.6104436,-6.8931905,Ganesa B
N,107.6119596,-6.8935537,Ganesa C
N,107.6129286,-6.8937204,Dago A
N,107.6135294,-6.8873914,Dago B
N,107.6136478,-6.8852158,Dago C
N,107.6101608,-6.8947687,Gelap Nyawang A
N,107.6117003,-6.8947261,Gelap Nyawang B
N,107.6099758,-6.8939384,Skanda A
E,Tamansari A,Tamansari B,131,undirected
E,Tamansari B,Tamansari C,654,undirected
E,Tamansari C,Tamansari D,362,undirected
E,Tamansari D,Tamansari E,90,undirected
E,Tamansari E,Tamansari F,169,directed
E,Dago A,Dago B,693,undirected
E,Dago B,Dago C,243,undirected
E,Tamansari A,Gelap Nyawang A,153,undirected
E,Gelap Nyawang A,Gelap Nyawang B,170,undirected
E,Gelap Nyawang B,Ganesa C,134,undirected
E,Tamansari B,Ganesa A,105,undirected
E,Ganesa A,Ganesa B,128,undirected
E,Ganesa B,Ganesa C,172,undirected
E,Ganesa C,Dago A,109,undirected
E,Ganesa A,Skanda A,137,undirected
E,Skanda A,Ganesa B,115,undirected
E,Skanda A,Gelap Nyawang A,95,undirected
E,Tamansari F,Dago C,242,directed
E,Dago C,Tamansari E,281,directed
E,Tamansari E,Dago B,227,undirected
```

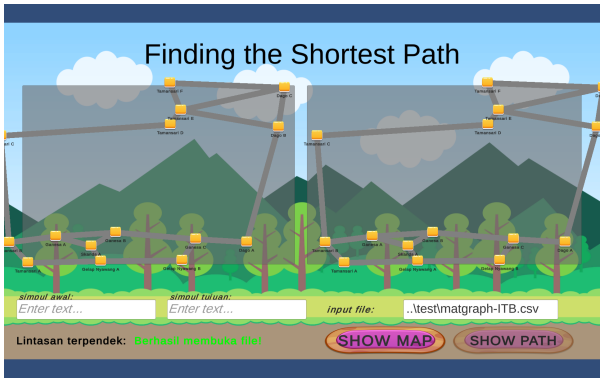
</td><td> 

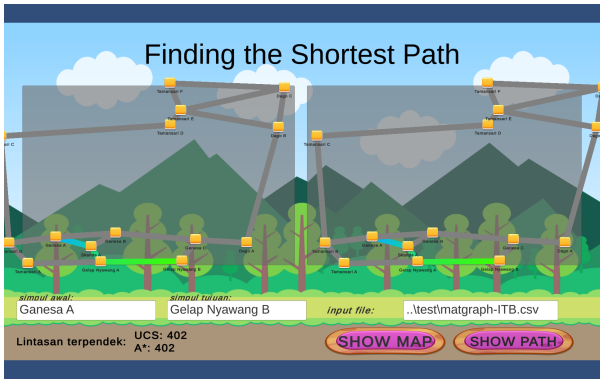


Simpul awal: Tamansari F
Simpul akhir: Skanda A
Jarak:
UCS = 1574
A* = 1574</td></tr>
</table>

matgraph-ITB.csv

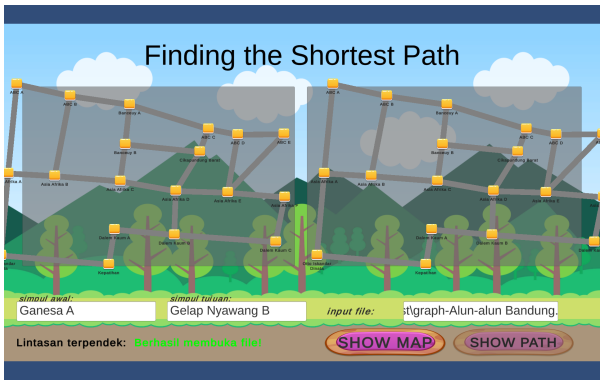
```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=matrix
N,107.608783,-6.8948617,Tamansari A
N,107.60842,-6.8937402,Tamansari B
N,107.6082665,-6.8878702,Tamansari C
N,107.6114806,-6.8872532,Tamansari D
N,107.6116764,-6.886465,Tamansari E
N,107.6114718,-6.8849629,Tamansari F
N,107.6093053,-6.8934058,Ganesa A
N,107.6104436,-6.8931905,Ganesa B
N,107.6119596,-6.8935537,Ganesa C
N,107.6129286,-6.8937204,Dago A
N,107.6135294,-6.8873914,Dago B
N,107.6136478,-6.8852158,Dago C
N,107.6101608,-6.8947687,Gelap Nyawang A
N,107.6117003,-6.8947261,Gelap Nyawang B
N,107.6099758,-6.8939384,Skanda A
E,0   131 0   0   0   0   0   0   0   0   0   153 0   0
E,131 0   654 0   0   0   105 0   0   0   0   0   0   0
E,0   654 0   362 0   0   0   0   0   0   0   0   0   0
E,0   0   362 0   90  0   0   0   0   0   0   0   0   0
E,0   0   0   90  0   169 0   0   0   0   227 0   0   0
E,0   0   0   0   0   0   0   0   0   0   0   242 0   0
E,0   105 0   0   0   0   0   128 0   0   0   0   0   137
E,0   0   0   0   0   0   128 0   172 0   0   0   0   115
E,0   0   0   0   0   0   0   172 0   109 0   0   0   134 0
E,0   0   0   0   0   0   0   0   109 0   693 0   0   0   0
E,0   0   0   0   227 0   0   0   0   693 0   243 0   0   0
E,0   0   0   0   281 0   0   0   0   0   243 0   0   0   0
E,153 0   0   0   0   0   0   0   0   0   0   0   0   170 95
E,0   0   0   0   0   0   0   0   134 0   0   0   170 0   0
E,0   0   0   0   0   0   137 115 0   0   0   0   95  0   0
```



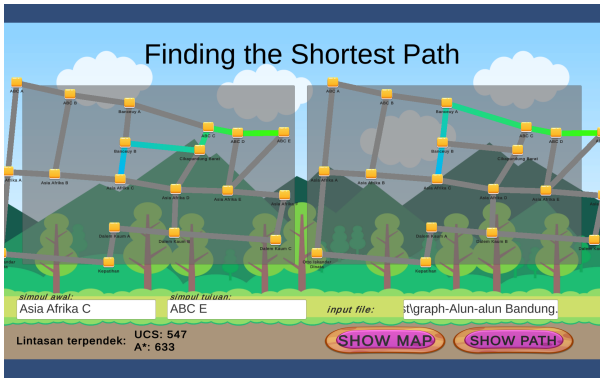


Simpul awal: Ganesa A
Simpul akhir: Gelap Nyawang B
Jarak:
UCS = 402
A* = 402

3.2 Peta jalan sekitar Alun-alun Bandung

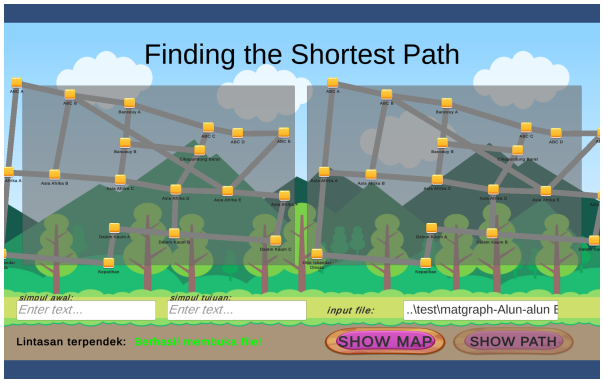
Input	Hasil
graph-Alun-alun Bandung.csv	

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=pair
N,107.6040898,-6.9208222,Asia Afrika A
N,107.6050782,-6.9208964,Asia Afrika B
N,107.6064737,-6.9210422,Asia Afrika C
N,107.607657,-6.9213172,Asia Afrika D
N,107.6087653,-6.921361,Asia Afrika E
N,107.6099786,-6.9214898,Asia Afrika F
N,107.6039371,-6.9231211,Otto Iskandar Dinata
N,107.6063511,-6.9223968,Dalem Kaum A
N,107.6076301,-6.9225448,Dalem Kaum B
N,107.6097838,-6.9227463,Dalem Kaum C
N,107.6062331,-6.9233767,Kepatihan
N,107.6042693,-6.9183202,ABC A
N,107.6054043,-6.9186524,ABC B
N,107.6083516,-6.9195765,ABC C
N,107.6089804,-6.9197364,ABC D
N,107.6099594,-6.9197204,ABC E
N,107.6066726,-6.9188953,Banceuy A
N,107.6065797,-6.9200103,Banceuy B
N,107.6081731,-6.9202192,Cikapundung Barat
E,Asia Afrika F,Asia Afrika E,135,directed
E,Asia Afrika E,Asia Afrika D,134,directed
E,Asia Afrika D,Asia Afrika C,134,directed
E,Asia Afrika C,Asia Afrika B,155,directed
E,Asia Afrika B,Asia Afrika A,110,directed
E,ABC A,Asia Afrika A,279,directed
E,ABC A,ABC B,131,directed
E,Asia Afrika B,ABC B,253,directed
E,ABC B,Banceuy A,164,directed
E,Asia Afrika C,Banceuy B,115,directed
E,Banceuy B,Banceuy A,131,directed
E,Banceuy A,ABC C,207,directed
E,Banceuy B,Cikapundung Barat,178,undirected
E,Cikapundung Barat,ABC C,74,directed
E,Asia Afrika D,Cikapundung Barat,162,directed
E,ABC C,ABC D,72,directed
E,ABC D,ABC E,108,directed
E,Asia Afrika E,ABC D,182,directed
E,ABC E,Asia Afrika E,311,directed
E,Asia Afrika F,Dalem Kaum C,142,directed
E,Dalem Kaum B,Dalem Kaum C,248,directed
E,Asia Afrika D,Dalem Kaum B,137,undirected
E,Dalem Kaum A,Dalem Kaum B,144,directed
E,Kepatihan,Dalem Kaum A,111,directed
E,Otto Iskandar Dinata,Kepatihan,255,undirected
E,Asia Afrika A,Otto Iskandar Dinata,256,directed
```



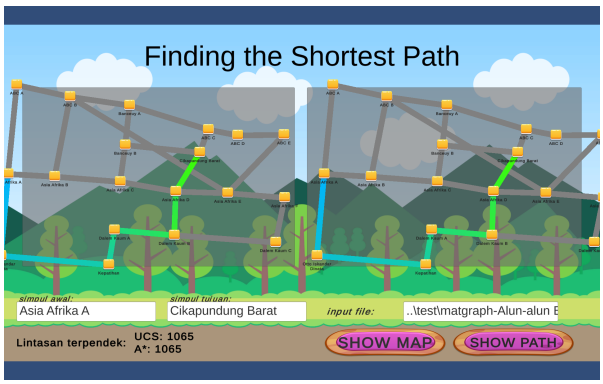
Simpul awal: Asia Afrika C
Simpul akhir: ABC E
Jarak:
UCS = 547
A* = 633

matgraph-Alun-alun Bandung.csv



<table>
<tr>
<td>

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=matrix
N,107.6040898,-6.9208222,Asia Afrika A
N,107.6050782,-6.9208964,Asia Afrika B
N,107.6064737,-6.9210422,Asia Afrika C
N,107.607657,-6.9213172,Asia Afrika D
N,107.6087653,-6.921361,Asia Afrika E
N,107.6099786,-6.9214898,Asia Afrika F
N,107.6039371,-6.9231211,Otto Iskandar Dinata
N,107.6063511,-6.9223968,Dalem Kaum A
N,107.6076301,-6.9225448,Dalem Kaum B
N,107.6097838,-6.9227463,Dalem Kaum C
N,107.6062331,-6.9233767,Kepatihan
N,107.6042693,-6.9183202,ABC A
N,107.6054043,-6.9186524,ABC B
N,107.6083516,-6.9195765,ABC C
N,107.6089804,-6.9197364,ABC D
N,107.6099594,-6.9197204,ABC E
N,107.6066726,-6.9188953,Banceuy A
N,107.6065797,-6.9200103,Banceuy B
N,107.6081731,-6.9202192,Cikapundung Barat
E,0    0    0    0    0    0    256  0    0    0    0    0    0    0    0    0    0    0    0    0
E,110  0    0    0    0    0    0    0    0    0    0    0    253  0    0    0    0    0    0    0
E,0    155  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    115  0
E,0    0    134  0    0    0    0    0    137  0    0    0    0    0    0    0    0    0    0    162
E,0    0    0    134  0    0    0    0    0    0    0    0    182  0    0    0    0    0    0    0
E,0    0    0    0    135  0    0    0    0    142  0    0    0    0    0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    255  0    0    0    0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    144  0    0    0    0    0    0    0    0    0    0    0
E,0    0    0    137  0    0    0    0    0    248  0    0    0    0    0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
E,0    0    0    0    0    0    255  111  0    0    0    0    0    0    0    0    0    0    0    0
E,279  0    0    0    0    0    0    0    0    0    0    0    131  0    0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    164  0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    72   0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    108  0    0    0    0
E,0    0    0    311  0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    207  0    0    0    0    0    0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    131  0    178  0
E,0    0    0    0    0    0    0    0    0    0    0    0    0    74   0    0    0    178  0    0
```

</td>
<td>


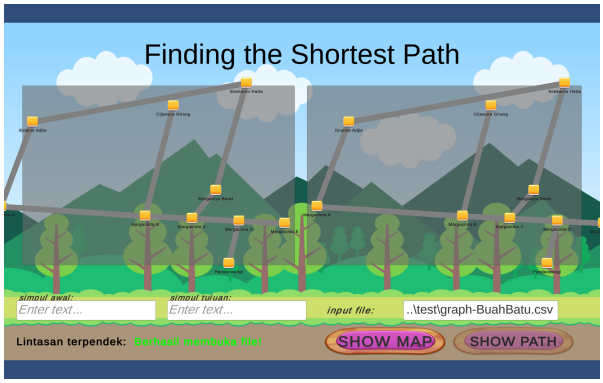
Simpul awal: Asia Afrika A
Simpul akhir: Cikapundung Barat
Jarak:
UCS = 1065
A* = 1065
</td>
</tr>
</table>

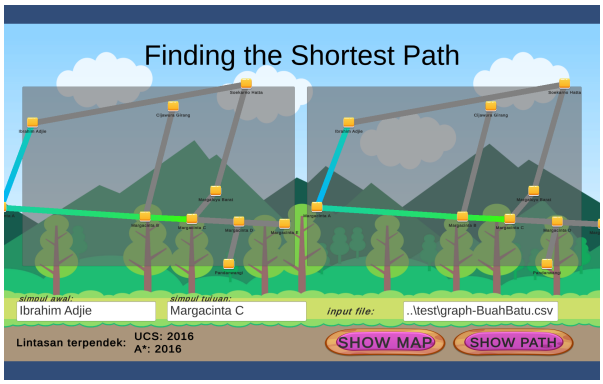
3.3 Peta jalan sekitar Buahbatu

<table>
<tr>
<th>Input</th>
<th>Hasil</th>
</tr>
<tr>
<td>

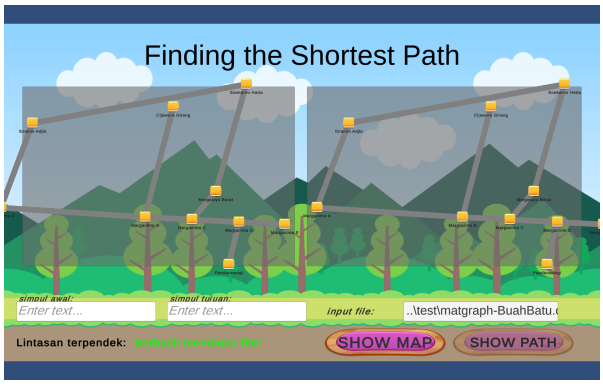
graph-BuahBatu.csv

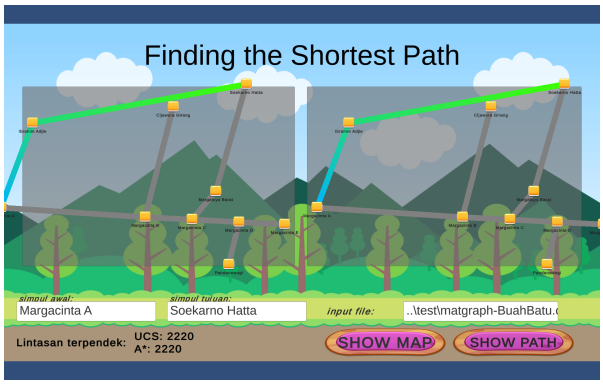
```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=pair
N,107.6401981,-6.9541001,Margacinta A
N,107.6475469,-6.9550278,Margacinta B
N,107.6499318,-6.9552515,Margacinta C
N,107.6523263,-6.9555033,Margacinta D
N,107.6546541,-6.9557245,Margacinta E
N,107.6511442,-6.9525358,Margaluyu Barat
N,107.6489786,-6.9443565,Cijawura Girang
N,107.6418074,-6.9459209,Ibrahim Adjie
N,107.6526999,-6.942156,Soekarno Hatta
N,107.6518114,-6.9596141,Pandanwangi
E,Margacinta E,Margacinta D,258,undirected
E,Margacinta D,Margacinta C,266,undirected
E,Margacinta C,Margacinta B,265,undirected
E,Margacinta B,Margacinta A,821,undirected
E,Margacinta D,Pandanwangi,462,undirected
E,Margacinta A,Ibrahim Adjie,930,undirected
E,Soekarno Hatta,Ibrahim Adjie,1290,directed
E,Margacinta C,Margaluyu Barat,332,undirected
E,Margaluyu Barat,Soekarno Hatta,1220,undirected
E,Margacinta B,Cijawura Girang,1220,undirected
```

</td>
<td>




Simpul awal: Ibrahim Adjie
</td>
</tr>
</table>

<table>
<tr><td></td><td>Simpul akhir: Margacinta C
Jarak:
UCS = 2016
A* = 2016</td></tr>
<tr><td>matgraph-BuahBatu.csv
<pre>
Node coordinate is in lon-lat format
O,format=lonlat,edge=matrix
N,107.6401981,-6.9541001,Margacinta A
N,107.6475469,-6.9550278,Margacinta B
N,107.6499318,-6.9552515,Margacinta C
N,107.6523263,-6.9555033,Margacinta D
N,107.6546541,-6.9557245,Margacinta E
N,107.6511442,-6.9525358,Margaluyu Barat
N,107.6489786,-6.9443565,Cijawura Girang
N,107.6418074,-6.9459209,Ibrahim Adjie
N,107.6526999,-6.942156,Soekarno Hatta
N,107.6518114,-6.9596141,Pandanwangi
E,0   821 0   0   0   0   0   930 0   0
E,821 0   265 0   0   0   1220 0   0   0
E,0   265 0   266 0   332 0   0   0   0
E,0   0   266 0   258 0   0   0   0   462
E,0   0   0   258 0   0   0   0   0   0
E,0   0   332 0   0   0   0   0   1220 0
E,0   1220 0   0   0   0   0   0   0   0
E,930 0   0   0   0   0   0   0   1290 0
E,0   0   0   0   0   0   0   1290 0   0
E,0   0   0   462 0   0   0   0   0   0
</pre></td><td> 



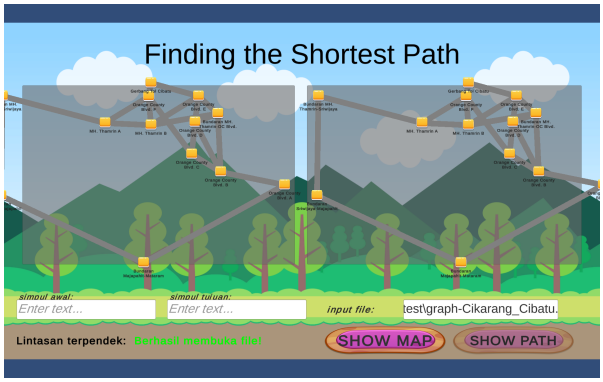
Simpul awal: Margacinta A
Simpul akhir: Soekarno Hatta
Jarak:
UCS = 2220
A* = 2220</td></tr>
</table>

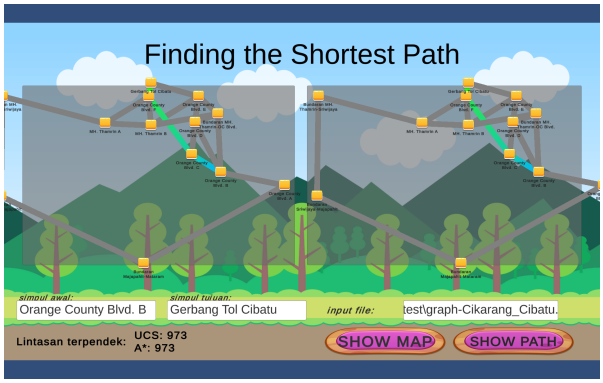
3.4 Peta jalan sebuah kawasan di kota asal

Input	Hasil

graph-Cikarang_Cibatu.csv

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=pair
N,107.159511,-6.3368296,MH. Thamrin A
N,107.1621073,-6.3370003,MH. Thamrin B
N,107.1536858,-6.3346411,Bundaran MH. Thamrin-Sriwijaya
N,107.1536028,-6.3427309,Bundaran Sriwijaya-Majapahit
N,107.1616924,-6.3481834,Bundaran Majapahit-Mataram
N,107.1696853,-6.3418601,Orange County Blvd. A
N,107.1660642,-6.3408034,Orange County Blvd. B
N,107.1644193,-6.3393646,Orange County Blvd. C
N,107.1646018,-6.3367656,Orange County Blvd. D
N,107.1648003,-6.334649,Orange County Blvd. E
N,107.1619786,-6.334681,Orange County Blvd. F
N,107.1659,-6.3360672,Bundaran MH. Thamrin-OC Blvd.
N,107.162113,-6.333583,Gerbang Tol Cibatu
E,Bundaran MH. Thamrin-Sriwijaya,Bundaran Sriwijaya-Majapahit,901,undirected
E,Bundaran Sriwijaya-Majapahit,Bundaran Majapahit-Mataram,1080,undirected
E,Bundaran Majapahit-Mataram,Orange County Blvd. A,1190,undirected
E,Bundaran MH. Thamrin-Sriwijaya,MH. Thamrin A,842,undirected
E,MH. Thamrin A,MH. Thamrin B,290,undirected
E,MH. Thamrin A,Orange County Blvd. F,501,undirected
E,MH. Thamrin B,Bundaran MH. Thamrin-OC Blvd.,435,undirected
E,Orange County Blvd. A,Orange County Blvd. B,418,undirected
E,Orange County Blvd. B,Orange County Blvd. C,254,undirected
E,Orange County Blvd. C,Orange County Blvd. D,295,undirected
E,Orange County Blvd. B,Orange County Blvd. D,603,undirected
E,Orange County Blvd. D,Orange County Blvd. E,237,undirected
E,Orange County Blvd. E,Bundaran MH. Thamrin-OC Blvd.,231,undirected
E,Orange County Blvd. E,MH. Thamrin B,506,directed
E,MH. Thamrin B,Orange County Blvd. F,274,directed
E,Bundaran MH. Thamrin-OC Blvd.,Orange County Blvd. B,528,undirected
E,Bundaran MH. Thamrin-OC Blvd.,Orange County Blvd. D,224,undirected
E,Orange County Blvd. C,Orange County Blvd. F,596,undirected
E,Orange County Blvd. F,Gerbang Tol Cibatu,123,undirected
E,Gerbang Tol Cibatu,Orange County Blvd. E,641,directed
```

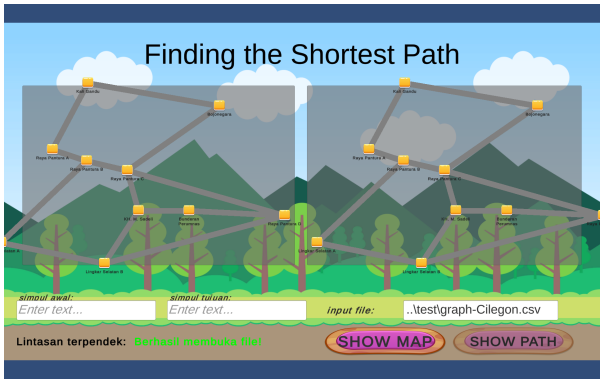


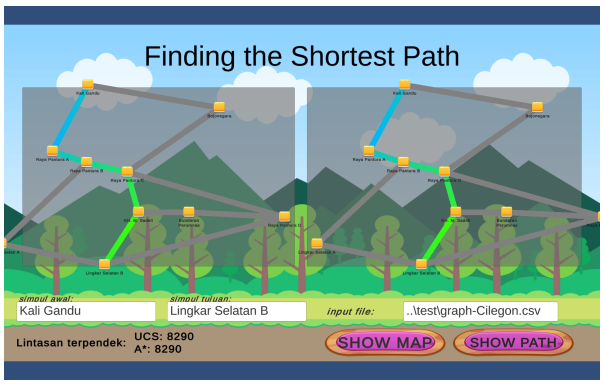


Simpul awal: Orange County Blvd. B
Simpul akhir: Gerbang Tol Cibatu
Jarak:
UCS = 973
A* = 973

graph-Cilegon.csv

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=pair
N,106.0434727,-6.011673,Raya Pantura A
N,106.0490901,-6.015428,Raya Pantura B
N,106.055773,-6.0186019,Raya Pantura C
N,106.0816198,-6.0335235,Raya Pantura D
N,106.0350139,-6.0424858,Lingkar Selatan A
N,106.0520084,-6.0493994,Lingkar Selatan B
N,106.0576071,-6.0319271,KH. M. Sadeli
N,106.0659102,-6.0319271,Bunderan Perumnas
N,106.0709099,-5.997197,Bojonegara
N,106.0492974,-5.9896366,Kali Gandu
E,Raya Pantura A,Raya Pantura B,749,undirected
E,Raya Pantura B,Raya Pantura C,821,undirected
E,Raya Pantura C,Raya Pantura D,3320,undirected
E,Raya Pantura D,Bunderan Perumnas,2450,undirected
E,Bunderan Perumnas,KH. M. Sadeli,1020,undirected
E,KH. M. Sadeli,Lingkar Selatan B,2460,undirected
E,Raya Pantura D,Lingkar Selatan B,4320,undirected
E,Lingkar Selatan B,Lingkar Selatan A,2080,undirected
E,Lingkar Selatan A,Raya Pantura B,3590,undirected
E,KH. M. Sadeli,Raya Pantura C,1630,undirected
E,Raya Pantura C,Bojonegara,3050,undirected
E,Bojonegara,Kali Gandu,2610,undirected
E,Kali Gandu,Raya Pantura A,2630,undirected
```

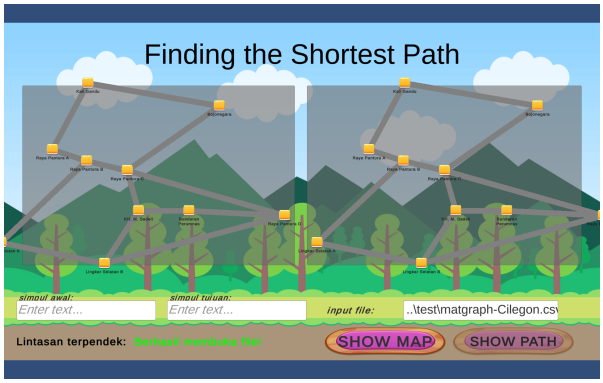


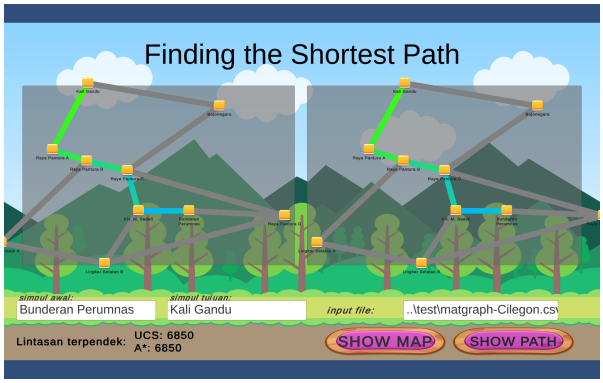


Simpul awal: Kali Gandu
Simpul akhir: Lingkar Selatan B
Jarak:
UCS = 8290
A* = 8290

matgraph-Cilegon.csv

```
# Node coordinate is in lon-lat format
O,format=lonlat,edge=matrix
N,106.0434727,-6.011673,Raya Pantura A
N,106.0490901,-6.015428,Raya Pantura B
N,106.055773,-6.0186019,Raya Pantura C
N,106.0816198,-6.0335235,Raya Pantura D
N,106.0350139,-6.0424858,Lingkar Selatan A
N,106.0520084,-6.0493994,Lingkar Selatan B
N,106.0576071,-6.0319271,KH. M. Sadeli
N,106.0659102,-6.0319271,Bunderan Perumnas
N,106.0709099,-5.997197,Bojonegara
N,106.0492974,-5.9896366,Kali Gandu
E,0    749  0    0    0    0    0    0    0    2630
E,749  0    821  0    3590 0    0    0    0    0
E,0    821  0    3320 0    0    1630 0    3050 0
E,0    0    3320 0    0    4320 0    2450 0    0
E,0    3590 0    0    0    2080 0    0    0    0
E,0    0    0    4320 2080 0    2460 0    0    0
E,0    0    1630 0    0    2460 0    1020 0    0
E,0    0    0    2450 0    0    1020 0    0    0
E,0    0    3050 0    0    0    0    0    0    2610
E,2630 0    0    0    0    0    0    0    2610 0
```





Simpul awal: Bunderan Perumnas
Simpul akhir: Kali Gandu
Jarak:
UCS = 6850
A* = 6850

Bab 4

Penutup

4.1 Kesimpulan

Algoritma *Uniform Cost Search* (UCS) dan A* merupakan algoritma yang digunakan untuk mencari lintasan terpendek dari suatu peta. Algoritma UCS termasuk *uninformed search* atau sering juga dikenal dengan *blind search*. Algoritma ini dapat menjamin menemukan lintasan terpendek, tetapi mungkin membutuhkan waktu yang lebih lama untuk penyelesaian graf yang besar dan kompleks. Sedangkan, algoritma A* termasuk *informed search* atau sering juga dikenal dengan *heuristic search*. Algoritma ini lebih efisien dan efektif dalam pencarian jalur, tetapi membutuhkan fungsi heuristik yang tepat agar dapat mencapai kinerja yang terbaik.

Dapat ditarik kesimpulan bahwa menggunakan metode pencarian lintasan terpendek dengan algoritma A* lebih efisien dibandingkan dengan menggunakan algoritma UCS. Hal ini terlihat dari hasil eksekusi pada *test case* di atas, di mana jarak yang ditempuh dengan menggunakan algoritma A* lebih pendek daripada menggunakan algoritma UCS. Meskipun ada *test case* yang menunjukkan bahwa jarak yang ditempuh sama antara kedua metode, namun penggunaan algoritma A* mengunjungi jumlah simpul yang lebih sedikit dibandingkan dengan algoritma UCS, menunjukkan bahwa waktu eksekusi dari metode A* lebih cepat daripada metode UCS.

Meskipun demikian, dalam kasus ini, algoritma UCS terbilang lebih efektif ketimbang algoritma A* dalam pencarian lintasan terpendek. Hal ini disebabkan oleh optimalitas A* yang hanya terjamin jika heuristik yang digunakan selalu menaksir *cost* lebih kecil dari *cost* sebenarnya. Sedangkan, algoritma UCS selalu menjamin bahwa lintasan yang ditemukan adalah lintasan optimal, karena simpul yang dikunjungi dalam setiap langkah selalu merupakan simpul dengan *cost* lintasan terkecil.

4.2 Saran

Bonus yang diberikan pada tugas kecil ini bagus untuk mendorong eksplorasi serta kreativitas penulis, tetapi dibutuhkan waktu yang lebih untuk merealisasikan hal tersebut. Sehingga penulis memiliki saran agar dapat lebih membagi waktu dalam pengerjaan tugas kecil ini.

LAMPIRAN

- Link Repository: JerichoFletcher/Tucil3_13521107_13521125 (github.com)
- Checklist

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	

DAFTAR PUSTAKA

Maulidevi, Nur Ulfa. (2021). “Penentuan Rute (Route/Path Planning) Bagian 1: BFS, DFS, UCS, Greedy Best First Search”. Diakses online dari Penentuan Rute (Route/Path Planning) pada 11 April 2023.

Munir, Rinaldi dan Maulidevi, Nur Ulfa. (2021). “Penentuan Rute (Route/Path Planning)Bagian 2: Algoritma A*”. Diakses online dari Penentuan Rute (Route/Path Planning) pada 11 April 2023.