

IF2211 STRATEGI ALGORITMA
TUGAS KECIL 3
Penyelesaian Permainan Word Ladder Menggunakan
Algoritma UCS, Greedy Best First Search, dan A*



Dipersiapkan oleh:

Ahmad Naufal Ramadan - 13522005

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2024

Daftar Isi

Daftar Isi	2
1. Deskripsi Tugas	3
2. Analisis dan Implementasi	3
a. Uniform Cost Search (UCS)	3
b. Greedy Best First Search (GBFS)	4
c. A*	5
3. Source Code Program	7
App.java	7
Tuple.java	14
StringUtil.java	15
NodeUtil.java	15
BFS.java	17
UCS.java	19
GBFS.java	20
AStar.java	22
4. Uji Kasus	25
Tabel 4.1 Tangkapan Layar Hasil Uji Kasus	25
5. Analisis Perbandingan Algoritma UCS, Greedy Best First Search, dan A*	43
6. Penjelasan Implementasi Bonus	44
Lampiran	45

1. Deskripsi Tugas

Word ladder (juga dikenal sebagai Doublets, word-links, change-the-word puzzles, paragrams, laddergrams, atau word golf) adalah salah satu permainan kata yang terkenal bagi seluruh kalangan. Word ladder ditemukan oleh Lewis Carroll, seorang penulis dan matematikawan, pada tahun 1877. Pada permainan ini, pemain diberikan dua kata yang disebut sebagai start word dan end word. Untuk memenangkan permainan, pemain harus menemukan rantai kata yang dapat menghubungkan antara start word dan end word. Banyaknya huruf pada start word dan end word selalu sama. Tiap kata yang berdekatan dalam rantai kata tersebut hanya boleh berbeda satu huruf saja. Pada permainan ini, diharapkan solusi optimal, yaitu solusi yang meminimalkan banyaknya kata yang dimasukkan pada rantai kata.

2. Analisis dan Implementasi

a. Uniform Cost Search (UCS)

Algoritma Uniform Cost Search (UCS) adalah algoritma pencarian graf berbasis cost yang digunakan untuk mencari jalur terpendek dari satu titik ke titik lainnya. Algoritma ini memanfaatkan struktur data priority queue untuk memilih node dengan cost terendah. Cost dari sebuah node ($g(n)$) dalam UCS adalah nilai dari fungsi yang mengukur jarak dari node awal ke node ke- n . Dalam konteks permainan Word Ladder, $g(n)$ dihitung sebagai panjang path dari node awal ke node saat ini, yang mana path ini merupakan serangkaian kata yang berbeda satu huruf dari kata sebelumnya. Karena setiap langkah dalam UCS hanya mengubah satu huruf pada kata, implementasi UCS dalam Word Ladder serupa dengan implementasi algoritma Breadth-First Search (BFS) dalam hal pencarian jalur terpendek.

Algoritma UCS yang diimplementasikan serupa dengan BFS, penjelasan langkah-langkahnya adalah sebagai berikut:

1. Inisialisasi queue sebagai priority queue (menggunakan PriorityQueue di Java) yang mengurutkan elemen berdasarkan cost, serta inisialisasi variabel visited sebagai Set dan parent sebagai Map untuk melacak node yang telah dikunjungi dan hubungan parent-child.

2. Selama priority queue tidak kosong, lakukan langkah-langkah berikut:

- Ambil node dengan cost terendah dari priority queue. Ini adalah node yang akan dieksplorasi selanjutnya.
- Jika node tersebut adalah node tujuan, maka pencarian selesai.

- Jika tidak, untuk setiap node tetangga yang dapat dicapai dari node saat ini, lakukan langkah berikut:
 1. Hitung cost baru untuk mencapai node tetangga tersebut melalui node saat ini.
 2. Jika cost baru lebih kecil dari cost yang sebelumnya telah diperhitungkan untuk node tetangga tersebut, update cost node tetangga tersebut dengan cost baru dan tambahkan node tetangga tersebut ke priority queue.
 - 3. Selama pencarian, simpan jalur yang diambil untuk mencapai setiap node, sehingga ketika node tujuan ditemukan, jalur terpendek dapat direkonstruksi dari node tujuan ke node awal.
 - 4. Jika dalam proses pencarian ditemukan bahwa cost untuk mencapai suatu node lebih kecil daripada cost yang telah diperhitungkan sebelumnya, maka cost tersebut akan di-update dan pencarian akan melanjutkan eksplorasi dari node tersebut untuk mencari jalur yang lebih pendek.
 - 5. Jika ada lebih dari satu jalur untuk mencapai suatu node, maka UCS akan mempertimbangkan semua jalur tersebut dan memilih jalur dengan cost terendah.
- b. Greedy Best First Search (GBFS)
- Algoritma Greedy Best First Search (GBFS) adalah algoritma pencarian graf yang mirip dengan UCS namun hanya mempertimbangkan nilai heuristik untuk memilih node yang akan dieksplorasi selanjutnya. Berbeda dengan UCS yang memperhitungkan cost sejauh ini ($g(n)$), GBFS hanya mempertimbangkan nilai heuristik yang menunjukkan perkiraan jarak dari node saat ini ke node tujuan ($h(n)$). Dalam konteks permainan Word Ladder, GBFS dapat diimplementasikan dengan mempertimbangkan heuristik yang menunjukkan jumlah karakter yang berbeda antara kata saat ini dan kata tujuan. Dengan demikian, GBFS cenderung memilih node yang dianggap "lebih dekat" ke node tujuan, tanpa mempertimbangkan total cost sejauh ini. Namun, perlu dicatat bahwa GBFS tidak menjamin menemukan jalur terpendek karena keputusan pencarian berdasarkan heuristik mungkin terjebak pada minimum lokal.
- Penjelasan algoritma GBFS yang diimplementasikan adalah sebagai berikut:
1. Inisialisasi priority queue sebagai PriorityQueue yang mengurutkan elemen berdasarkan nilai heuristik ($h(n)$) yang diberikan oleh fungsi heuristic. Selain itu,

inisialisasi variabel visited sebagai Set dan parent sebagai Map untuk melacak node yang telah dikunjungi dan hubungan parent-child.

2. Fungsi hamming digunakan untuk menghitung jarak heuristik antara node saat ini dengan node tujuan, yaitu jumlah huruf yang berbeda pada posisi yang sama. Fungsi ini digunakan dalam perhitungan nilai heuristik untuk setiap node.

3. Selama priority queue tidak kosong, lakukan langkah-langkah berikut:

- Ambil node dengan nilai heuristik terendah dari priority queue. Ini adalah node yang akan dieksplorasi selanjutnya.
- Jika node tersebut adalah node tujuan, maka pencarian selesai.
- Jika tidak, untuk setiap node tetangga yang dapat dicapai dari node saat ini, lakukan langkah berikut:
 1. Hitung nilai heuristik untuk node tetangga tersebut.
 2. Tambahkan node tetangga tersebut ke priority queue, dengan nilai heuristik sebagai kriteria pengurutan.
 3. Tandai node tetangga tersebut sebagai dikunjungi, dan catat hubungan parent-child antara node saat ini dan node tetangga.

4. Selama pencarian, simpan jalur yang diambil untuk mencapai setiap node, sehingga ketika node tujuan ditemukan, jalur terpendek dapat direkonstruksi dari node tujuan ke node awal.

5. Jika ada lebih dari satu jalur untuk mencapai suatu node, GBFS akan mempertimbangkan node-node tersebut dan memilih node dengan nilai heuristik yang paling kecil

c. A*

Algoritma A* (A-star) adalah algoritma pencarian graf yang menggabungkan konsep dari UCS dan GBFS dengan mempertimbangkan total cost dari node awal ke node saat ini ($g(n)$) ditambah dengan nilai heuristik yang menunjukkan perkiraan jarak dari node saat ini ke node tujuan ($h(n)$). Total cost ini dinyatakan sebagai $f(n) = g(n) + h(n)$. Algoritma A* menggunakan nilai $f(n)$ untuk memilih node yang akan dieksplorasi selanjutnya.

Dalam konteks permainan Word Ladder, A* dapat diimplementasikan dengan mempertimbangkan total cost yang terdiri dari jumlah langkah yang telah diambil dari node awal ke node saat ini ($g(n)$) dan jumlah karakter yang berbeda antara kata saat ini dan kata tujuan ($h(n)$). Dengan demikian, A* cenderung memilih node yang memiliki total cost paling rendah dengan menggabungkan efisiensi dari UCS dalam menemukan jalur terpendek dengan kecerdasan

heuristik dari GBFS. Oleh karena itu, algoritma A* secara teoritis dapat menemukan jalur terpendek lebih efisien dibandingkan dengan algoritma UCS..

Jarak Hamming memberikan perkiraan terendah dari jumlah langkah yang diperlukan untuk mengubah satu kata menjadi kata lainnya. Oleh karena itu, dalam konteks Word Ladder, jarak Hamming antara kata saat ini dan kata tujuan sebenarnya tidak akan melebihi jumlah langkah yang diperlukan untuk mencapai kata tujuan. Jadi, heuristik jarak Hamming adalah admissible untuk digunakan dalam algoritma A* pada permainan Word Ladder.

Penjelasan algoritma A* yang diimplementasikan adalah sebagai berikut:

1. Inisialisasi priority queue sebagai PriorityQueue yang mengurutkan elemen berdasarkan nilai fungsi evaluasi ($f(n)$), yang merupakan jumlah cost dari node awal ke node saat ini ditambah dengan nilai heuristik dari node saat ini ke node tujuan. Selain itu, inisialisasi variabel visited sebagai Set untuk melacak node yang telah dikunjungi.
2. Fungsi hamming digunakan untuk menghitung jarak heuristik antara dua kata, yaitu jumlah huruf yang berbeda pada posisi yang sama. Fungsi ini digunakan dalam perhitungan nilai heuristik untuk setiap node.
3. Fungsi cost digunakan untuk menghitung jumlah langkah yang telah diambil dari node awal ke node saat ini.
4. Fungsi evaluasi digunakan untuk menghitung nilai $f(n) = g(n) + h(n)$, yaitu total cost dari node awal ke node saat ini ditambah dengan nilai heuristik dari node saat ini ke node tujuan.
5. Selama priority queue tidak kosong, lakukan langkah-langkah berikut:
 - Ambil node dengan nilai $f(n)$ terkecil dari priority queue. Ini adalah node yang akan dieksplorasi selanjutnya.
 - Jika node tersebut adalah node tujuan, maka pencarian selesai.
 - Jika tidak, untuk setiap node tetangga yang dapat dicapai dari node saat ini, lakukan langkah berikut:
 1. Tambahkan node tetangga tersebut ke priority queue dengan nilai $f(n)$ yang sesuai.
 2. Tandai node tetangga tersebut sebagai dikunjungi.
6. Selama pencarian, simpan jalur yang diambil untuk mencapai setiap node, sehingga ketika node tujuan ditemukan, jalur terpendek dapat direkonstruksi dari node tujuan ke node awal.

7. Jika ada lebih dari satu jalur untuk mencapai suatu node, A* akan mempertimbangkan semua jalur tersebut dan memilih jalur dengan nilai $f(n)$ yang paling kecil. Algoritma A* menggunakan nilai $f(n)$ untuk memastikan bahwa jalur yang diambil memiliki total cost yang minimum dari node awal ke node tujuan.

3. Source Code Program

App.java

```
package com.sandwichese.app;

import java.io.File;
import java.util.ArrayList;
import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;
import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;

public class App extends JFrame implements ActionListener {
    private JTextField startField, goalField;
    private JComboBox<String> methodComboBox;
    private JTable pathTable;
    private JTextArea resultArea;

    // Dark Purple
    private static Color primaryColor = Color.decode("#2C001E");
    // Orange
    private static Color secondaryColor = Color.decode("#DD4814");
    // White
    private static Color textWhiteColor = Color.decode("#FFFFFF");
    // Dark Gray
    private static Color textGrayColor = Color.decode("#333333");

    public App() {
        setTitle("Word Ladder Solver");
        setSize(1200, 800);
    }
}
```

```
setDefaultCloseOperation(EXIT_ON_CLOSE);
setLocationRelativeTo(null);

Font font = new Font("Arial", Font.PLAIN, 24);

JPanel panel = new JPanel();
panel.setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;

JLabel startLabel = new JLabel("Enter starting word:");
startLabel.setFont(font);
gbc.gridx = 0;
gbc.gridy = 0;
panel.add(startLabel, gbc);

startField = new JTextField(15);
startField.setFont(font);
gbc.gridx = 1;
gbc.gridy = 0;
panel.add(startField, gbc);

JLabel goalLabel = new JLabel("Enter goal word:");
goalLabel.setFont(font);
gbc.gridx = 0;
gbc.gridy = 1;
panel.add(goalLabel, gbc);

goalField = new JTextField(15);
goalField.setFont(font);
gbc.gridx = 1;
gbc.gridy = 1;
panel.add(goalField, gbc);

JLabel methodLabel = new JLabel("Select method:");
methodLabel.setFont(font);
gbc.gridx = 0;
gbc.gridy = 2;
```



```

panel.add(methodLabel, gbc);

String[] methods = { "UCS", "GBFS", "A*" };
methodComboBox = new JComboBox<>(methods);
methodComboBox.setFont(font);
gbc.gridx = 1;
gbc.gridy = 2;
panel.add(methodComboBox, gbc);

JButton solveButton = new JButton("Solve");
solveButton.setUI(new CustomButtonUI(secondaryColor));
solveButton.addActionListener(this);
solveButton.setFont(font);
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
gbc.anchor = GridBagConstraints.CENTER;
panel.add(solveButton, gbc);

add(panel);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    String start = startField.getText().toLowerCase();
    String goal = goalField.getText().toLowerCase();
    String method =
methodComboBox.getSelectedItem().toString().toLowerCase();

    if (method.equals("a*")) {
        method = "astar";
    }

    JFrame resultFrame = new JFrame("Result");
    resultFrame.setSize(1200, 800);
    resultFrame.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    resultFrame.setLocationRelativeTo(null);
    JPanel resultPanel = new JPanel();

```

```

resultPanel.setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;

if (start.length() != goal.length()) {
    resultArea = new JTextArea("Words must have the same length",
5, 30);

    resultArea.setFont(new Font("Arial", Font.PLAIN, 24));
    resultArea.setLineWrap(true);
    resultArea.setWrapStyleWord(true);
    resultArea.setEditable(false);

    gbc.gridx = 0;
    gbc.gridy = 0;
    resultPanel.add(resultArea, gbc);

    resultFrame.add(resultPanel);
    resultFrame.setVisible(true);
    return;
}

String wordListFile = "./wordlist/" + start.length() + ".txt";
File wordList = new File(wordListFile);

        if (!NodeUtil.isInDictionary(start, wordList) ||
!NodeUtil.isInDictionary(goal, wordList)) {
    resultArea = new JTextArea("Words must be in the dictionary",
5, 30);

    resultArea.setFont(new Font("Arial", Font.PLAIN, 24));
    resultArea.setLineWrap(true);
    resultArea.setWrapStyleWord(true);
    resultArea.setEditable(false);

    gbc.gridx = 0;
    gbc.gridy = 0;
    resultPanel.add(resultArea, gbc);

    resultFrame.add(resultPanel);

```

```

        resultFrame.setVisible(true);
        return;
    }

    Runtime runtime = Runtime.getRuntime();
    runtime.gc();
    long startMemory = runtime.totalMemory() - runtime.freeMemory();

    long startTime = System.nanoTime();
    Tuple<ArrayList<String>, Integer> pathTuple = null;
    switch (method) {
        case "ucs":
            UCS ucs = new UCS(start, goal);
            pathTuple = ucs.search(wordList);
            break;
        case "gbfs":
            GBFS gbfs = new GBFS(start, goal);
            pathTuple = gbfs.search(wordList);
            break;
        case "astar":
            AStar astar = new AStar(start, goal);
            pathTuple = astar.search(wordList);
            break;
        default:
            resultArea.setText("Invalid method");
            return;
    }

    long endMemory = runtime.totalMemory() - runtime.freeMemory();

    long memory = endMemory - startMemory;

    long endTime = System.nanoTime();
    if (!pathTuple.x.isEmpty()) {
        String[] columnNames = { "Step", "Word" };
        DefaultTableModel model = new DefaultTableModel(columnNames,
0);

        for (int i = 0; i < pathTuple.x.size(); i++) {
            String step = Integer.toString(i);

```

```

String word =
StringUtil.convertToTitleCaseSplitting(pathTuple.x.get(i), " ");
    model.addRow(new String[] { step, word });
}

pathTable = new JTable(model);
pathTable.setFont(new Font("Arial", Font.PLAIN, 24));
pathTable.setRowHeight(30);
pathTable.setEnabled(false);
JScrollPane scrollPane = new JScrollPane(pathTable);
gbc.gridx = 0;
gbc.gridy = 0;
resultPanel.add(scrollPane, gbc);

    resultArea = new JTextArea("Number of nodes visited: " +
pathTuple.y + "\nTime taken: "
        + (endTime - startTime) / 1000000 + " ms\n" + "Memory
usage: " + memory + " bytes", 5, 30);
    resultArea.setFont(new Font("Arial", Font.PLAIN, 24));
    resultArea.setLineWrap(true);
    resultArea.setWrapStyleWord(true);
    resultArea.setEditable(false);

gbc.gridx = 0;
gbc.gridy = 1;
resultPanel.add(resultArea, gbc);

resultFrame.add(resultPanel);
resultFrame.setVisible(true);
} else {
    resultArea = new JTextArea("No path found from " + start + "
to " + goal + "\nNumber of nodes visited: "
        + pathTuple.y + "\nTime taken: "
        + (endTime - startTime) / 1000000 + " ms\n" + "Memory
usage: " + memory + " bytes", 5, 30);
    resultArea.setFont(new Font("Arial", Font.PLAIN, 24));
    resultArea.setLineWrap(true);
    resultArea.setWrapStyleWord(true);
    resultArea.setEditable(false);
}

```

```

        gbc.gridx = 0;
        gbc.gridy = 0;
        resultPanel.add(resultArea, gbc);

        resultFrame.add(resultPanel);
        resultFrame.setVisible(true);
    }
}

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
        e.printStackTrace();
    }

    UIManager.put("Button.background", secondaryColor);
    UIManager.put("Button.foreground", textWhiteColor);
    UIManager.put("Panel.background", primaryColor);
    UIManager.put("Label.foreground", textWhiteColor);
    UIManager.put("TextField.background", textWhiteColor);
    UIManager.put("TextField.foreground", textGrayColor);
    UIManager.put("ComboBox.background", textWhiteColor);
    UIManager.put("ComboBox.foreground", textGrayColor);
    UIManager.put("ComboBox.selectionBackground", secondaryColor);
    UIManager.put("ComboBox.selectionForeground", textWhiteColor);
    UIManager.put("Table.background", textWhiteColor);
    UIManager.put("Table.foreground", textGrayColor);
    UIManager.put("Table.selectionBackground", secondaryColor);
    UIManager.put("Table.selectionForeground", textWhiteColor);
    UIManager.put("Table.gridColor", secondaryColor);
    UIManager.put("TextArea.background", textWhiteColor);
    UIManager.put("TextArea.foreground", textGrayColor);

    SwingUtilities.invokeLater(App::new);
}

```

```

public static class CustomButtonUI extends BasicButtonUI {
    private Color backgroundColor;

    public CustomButtonUI(Color backgroundColor) {
        this.backgroundColor = backgroundColor;
    }

    @Override
    public void paint(Graphics g, JComponent c) {
        AbstractButton b = (AbstractButton) c;
        ButtonModel model = b.getModel();
        Color color = model.isPressed() ? backgroundColor.darker() :
backgroundColor;

        g.setColor(color);
        g.fillRect(0, 0, c.getWidth(), c.getHeight());

        super.paint(g, c);
    }
}

```

File App.java merupakan entry point dari program yang memiliki static method main. Di dalam file App.java, didefinisikan class App yang diturunkan dari class JFrame dan mengimplementasikan interface ActionListener. Class App selain berfungsi sebagai entry point juga yang bertugas untuk membangun GUI pada aplikasi yang dibuat.

Tuple.java

```

package com.sandwichese.app;

public class Tuple<T, U> {
    public final T x;
    public final U y;

    public Tuple(T x, U y) {
        this.x = x;
        this.y = y;
    }
}

```

File Tuple.java berisi class Tuple yang merupakan class generic yang dapat menyimpan pasangan data bertipe objek apapun,

StringUtil.java

```
package com.sandwichese.app;

import java.util.Arrays;
import java.util.stream.Collectors;

public class StringUtil {
    public static String convertToTitleCaseSplitting(String text, String separator) {
        if (text == null || text.isEmpty()) {
            return text;
        }

        return Arrays
            .stream(text.split(separator))
            .map(word -> word.isEmpty()
                ? word
                : Character.toUpperCase(word.charAt(0)) + word
                    .substring(1)
                    .toLowerCase())
            .collect(Collectors.joining(separator));
    }
}
```

File StringUtil.java berisi class StringUtil yang berisi method utilitas yang digunakan untuk memformat string. Hanya terdapat satu method pada class ini, yaitu method untuk mengubah suatu string menjadi title case.

NodeUtil.java

```
package com.sandwichese.app;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
```

```

public class NodeUtil {
    public static boolean isNeighbor(String word1, String word2) {
        int count = 0;
        for (int i = 0; i < word1.length(); i++) {
            if (word1.charAt(i) != word2.charAt(i)) {
                count++;
            }
        }
        return count == 1;
    }

    public static ArrayList<String> getNeighbors(String word, File wordList) {
        ArrayList<String> neighbors = new ArrayList<String>();
        try {
            Scanner myReader = new Scanner(wordList);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                if (isNeighbor(word, data)) {
                    neighbors.add(data);
                }
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
            e.printStackTrace();
        }

        return neighbors;
    }

    public static Boolean isInDictionary(String word, File wordList) {
        try {
            Scanner myReader = new Scanner(wordList);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                if (word.equals(data)) {
                    myReader.close();
                    return true;
                }
            }
        }
    }
}

```



```

    }
    }
    myReader.close();
} catch (FileNotFoundException e) {
    System.out.println("File not found");
    e.printStackTrace();
}
return false;
}
}

```

File NodeUtil.java berisi class NodeUtil yang berisi method utilitas yang digunakan untuk operasi yang berhubungan dengan node. Terdapat method untuk mengecek apakah suatu kata bertetanggan dengan kata lain, method yang mengembalikan semua tetangga dari suatu kata, dan method untuk mengecek apakah suatu kata berada pada kamus yang digunakan.

BFS.java

```

package com.sandwichese.app;

import java.io.File;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Map;
import java.util.Queue;
import java.util.Set;

public class BFS {
    private String start;
    private String goal;

    public BFS(String start, String goal) {
        this.start = start;
        this.goal = goal;
    }

    public Tuple<ArrayList<String>, Integer> search(File wordlist) {

```

```

Queue<ArrayList<String>> queue = new LinkedList<>();
Set<String> visited = new HashSet<>();
Map<String, String> parent = new HashMap<>();

ArrayList<String> initialPath = new ArrayList<>();
initialPath.add(start);
queue.add(initialPath);
visited.add(start);
parent.put(start, null);

int currentDepth = 0;

while (!queue.isEmpty()) {
    ArrayList<String> currentPath = queue.poll();
    String current = currentPath.get(currentPath.size() - 1);

    if (currentPath.size() > currentDepth) {
        currentDepth = currentPath.size();
        System.out.println("Searching depth " + currentDepth);
    }

    if (current.equals(goal)) {
        Tuple<ArrayList<String>, Integer> result = new
Tuple<>(currentPath, visited.size());
        return result;
    }

    ArrayList<String> neighbors = NodeUtil.getNeighbors(current,
wordlist);
    for (String neighbor : neighbors) {
        if (!visited.contains(neighbor)) {
            ArrayList<String> newPath = new
ArrayList<>(currentPath);
            newPath.add(neighbor);
            queue.add(newPath);
            visited.add(neighbor);
            parent.put(neighbor, current);
        }
    }
}

```

```

    }

    Tuple<ArrayList<String>, Integer> result = new Tuple<>>(new
ArrayList<>>(), visited.size());
    return result;
}
}

```

File BFS.java berisi class BFS yang memiliki method search yang mengimplementasikan algoritma BFS pada permainan Word Ladder dan mengembalikan Tuple yang berisi path dan banyaknya node yang dikunjungi. Algoritma ini adalah algoritma yang akan digunakan pada class UCS.java

UCS.java

```

package com.sandwicheseese.app;

import java.io.File;
import java.util.ArrayList;

public class UCS {
    // Pada UCS permainan word ladder, definisi dari g(n) adalah jarak
    // dari node
    // awal ke node saat ini. Karena pada permainan word ladder kita hanya
    // diperbolehkan untuk mengubah satu huruf pada setiap langkahnya,
    // maka jarak
    // dari node awal ke node saat ini adalah jumlah huruf yang berbeda
    // antara kedua
    // node tersebut. Oleh karena itu, essentially UCS pada permainan word
    // ladder
    // adalah sama dengan BFS.

    private String start;
    private String goal;

    public UCS(String start, String goal) {
        this.start = start;
        this.goal = goal;
    }
}

```

```

    public Tuple<ArrayList<String>, Integer> search(File wordlist) {
        BFS bfs = new BFS(start, goal);
        return bfs.search(wordlist);
    }
}

```

File UCS.java berisi class UCS yang memiliki method search yang memanggil method search dari objek BFS dan mengembalikan nilainya.

GBFS.java

```

package com.sandwichese.app;

import java.io.File;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Set;

public class GBFS {
    // Definisi h(n) pada Greedy Best First Search (GBFS) adalah jarak
    hamming dari
    // node saat ini ke node tujuan. Jarak hamming adalah jumlah huruf
    yang berbeda
    // pada posisi yang sama antara kedua node tersebut. Himpunan calon
    solusi yang
    // di-generate oleh GBFS adalah himpunan yang berisi node-node yang
    memiliki
    // nilai h(n) yang paling kecil.

    private String start;
    private String goal;

    private int hamming(String word1, String word2) {
        int count = 0;
        for (int i = 0; i < word1.length(); i++) {
            if (word1.charAt(i) != word2.charAt(i)) {

```

```

        count++;
    }
}
return count;
}

private int heuristic(ArrayList<String> path) {
    String current = path.get(path.size() - 1);
    return hamming(current, goal);
}

public GBFS(String start, String goal) {
    this.start = start;
    this.goal = goal;
}

public Tuple<ArrayList<String>, Integer> search(File wordlist) {
    PriorityQueue<ArrayList<String>> queue = new
PriorityQueue<>(Comparator.comparingInt(this::heuristic));
    Set<String> visited = new HashSet<>();
    Map<String, String> parent = new HashMap<>();

    ArrayList<String> initialPath = new ArrayList<>();
    initialPath.add(start);
    queue.add(initialPath);
    visited.add(start);
    parent.put(start, null);

    int currentDepth = 0;

    while (!queue.isEmpty()) {
        ArrayList<String> currentPath = queue.poll();
        String current = currentPath.get(currentPath.size() - 1);

        if (currentPath.size() > currentDepth) {
            currentDepth = currentPath.size();
            System.out.println("Searching depth " + currentDepth);
        }
    }
}

```

```

        if (current.equals(goal)) {
            Tuple<ArrayList<String>, Integer> result = new
Tuple<>(currentPath, visited.size());
            return result;
        }

        ArrayList<String> neighbors = NodeUtil.getNeighbors(current,
wordlist);
        for (String neighbor : neighbors) {
            if (!visited.contains(neighbor)) {
                ArrayList<String> newPath = new
ArrayList<>(currentPath);
                newPath.add(neighbor);
                queue.add(newPath);
                visited.add(neighbor);
                parent.put(neighbor, current);
            }
        }
    }

    Tuple<ArrayList<String>, Integer> result = new Tuple<>(new
ArrayList<>(), visited.size());
    return result;
}
}

```

File GBFS.java berisi class GBFS yang memiliki method search yang mengimplementasikan algoritma GBFS pada permainan Word Ladder dan mengembalikan Tuple yang berisi path dan banyaknya node yang dikunjungi.

AStar.java

```

package com.sandwichese.app;

import java.io.File;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashSet;
import java.util.PriorityQueue;
import java.util.Set;

```

```

public class AStar {
    // Pada A* permainan word ladder, definisi dari g(n) adalah jarak dari
    node
    // awal ke node saat ini. Definisi dari h(n) adalah jarak hamming dari
    node saat
    // ini ke node tujuan. Fungsi f(n) adalah g(n) + h(n). Himpunan calon
    solusi
    // yang di-generate oleh A* adalah himpunan yang berisi node-node yang
    memiliki
    // nilai f(n) yang paling kecil. A* adalah gabungan dari UCS dan GBFS.

    private String start;
    private String goal;

    private int hamming(String word1, String word2) {
        int count = 0;
        for (int i = 0; i < word1.length(); i++) {
            if (word1.charAt(i) != word2.charAt(i)) {
                count++;
            }
        }
        return count;
    }

    private int heuristic(ArrayList<String> path) {
        String current = path.get(path.size() - 1);
        return hamming(current, goal);
    }

    private int cost(ArrayList<String> path) {
        return path.size() - 1;
    }

    private int function(ArrayList<String> path) {
        return cost(path) + heuristic(path);
    }

    public AStar(String start, String goal) {

```

```

        this.start = start;
        this.goal = goal;
    }

    public Tuple<ArrayList<String>, Integer> search(File wordlist) {
        PriorityQueue<ArrayList<String>> queue = new
PriorityQueue<>(Comparator.comparingInt(this::function));
        Set<String> visited = new HashSet<>();

        ArrayList<String> initialPath = new ArrayList<>();
        initialPath.add(start);
        queue.add(initialPath);
        visited.add(start);

        int currentDepth = 0;

        while (!queue.isEmpty()) {
            ArrayList<String> currentPath = queue.poll();
            String current = currentPath.get(currentPath.size() - 1);

            if (currentPath.size() > currentDepth) {
                currentDepth = currentPath.size();
                System.out.println("Searching depth " + currentDepth);
            }

            if (current.equals(goal)) {
                Tuple<ArrayList<String>, Integer> result = new
Tuple<>(currentPath, visited.size());
                return result;
            }

            ArrayList<String> neighbors = NodeUtil.getNeighbors(current,
wordlist);
            for (String neighbor : neighbors) {
                if (!visited.contains(neighbor)) {
                    ArrayList<String> newPath = new
ArrayList<>(currentPath);
                    newPath.add(neighbor);
                    queue.add(newPath);
                }
            }
        }
    }

```



```

        visited.add(neighbor);
    }
}

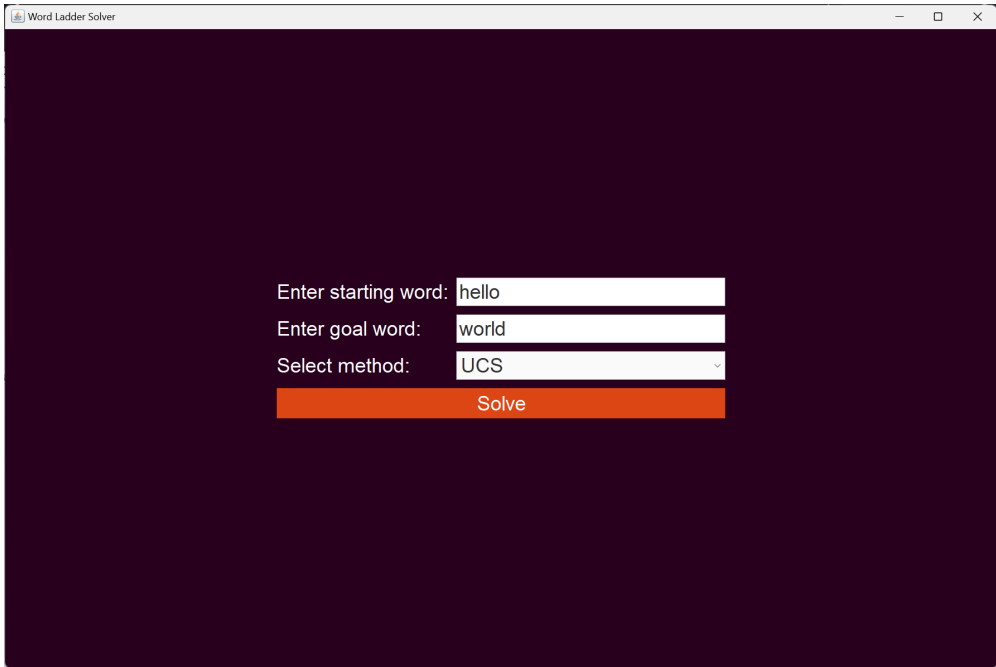
Tuple<ArrayList<String>, Integer> result = new Tuple<>(new
ArrayList<>(), visited.size());
return result;
}
}

```

File AStar.java berisi class AStar yang memiliki method search yang mengimplementasikan algoritma A* pada permainan Word Ladder dan mengembalikan Tuple yang berisi path dan banyaknya node yang dikunjungi.

4. Uji Kasus

Tabel 4.1 Tangkapan Layar Hasil Uji Kasus

No	Metode	Hasil Pengujian
1	UCS	

		<div><div>Result</div><table><tr><th>Step</th><th>Word</th></tr><tr><td>0</td><td>Hello</td></tr><tr><td>1</td><td>Helly</td></tr><tr><td>2</td><td>Holly</td></tr><tr><td>3</td><td>Hooly</td></tr><tr><td>4</td><td>Wooly</td></tr><tr><td>5</td><td>Woold</td></tr><tr><td>6</td><td>World</td></tr></table><div>Number of nodes visited: 6481 Time taken: 5688 ms Memory usage: 145776992 bytes</div></div>	Step	Word	0	Hello	1	Helly	2	Holly	3	Hooly	4	Wooly	5	Woold	6	World
Step	Word																	
0	Hello																	
1	Helly																	
2	Holly																	
3	Hooly																	
4	Wooly																	
5	Woold																	
6	World																	
GBFS		<div><div>Word Ladder Solver</div><div><div>Enter starting word: hello</div><div>Enter goal word: world</div><div>Select method: GBFS</div><div>Solve</div></div></div>																

	<div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div>
--	--

		<div><div>Result</div><table><tr><th>Step</th><th>Word</th></tr><tr><td>0</td><td>Hello</td></tr><tr><td>1</td><td>Hollo</td></tr><tr><td>2</td><td>Holly</td></tr><tr><td>3</td><td>Wolly</td></tr><tr><td>4</td><td>Wooly</td></tr><tr><td>5</td><td>Woold</td></tr><tr><td>6</td><td>World</td></tr></table><div>Number of nodes visited: 240 Time taken: 98 ms Memory usage: 19972416 bytes</div></div>	Step	Word	0	Hello	1	Hollo	2	Holly	3	Wolly	4	Wooly	5	Woold	6	World
Step	Word																	
0	Hello																	
1	Hollo																	
2	Holly																	
3	Wolly																	
4	Wooly																	
5	Woold																	
6	World																	
2	UCS	<div><div>Word Ladder Solver</div><div><div>Enter starting word:</div><div>count</div></div><div><div>Enter goal word:</div><div>world</div></div><div><div>Select method:</div><div>UCS</div></div><div>Solve</div></div>																

A screenshot of a software window titled "Result". It displays a word ladder solution. The solution is presented in a table with two columns: "Step" and "Word". The steps are numbered 0 through 9, and the words are: Count, Mount, Moult, Mouls, Molls, Mells, Melos, Melon, Meson, and Mason. Below the table, a summary of the search process is provided: "Number of nodes visited: 8373", "Time taken: 8492 ms", and "Memory usage: 190331040 bytes".

GBFS

A screenshot of a software window titled "Word Ladder Solver". It features three input fields for user configuration: "Enter starting word:" with the value "count", "Enter goal word:" with the value "world", and "Select method:" with a dropdown menu showing "GBFS". Below these fields is a prominent orange button labeled "Solve".

Result

Step	Word
0	Count
1	Mount
2	Mound
3	Moued
4	Mosed
5	Mased
6	Mases
7	Manes
8	Manos
9	Manon
10	Mason

Number of nodes visited: 199
Time taken: 38 ms
Memory usage: 44087616 bytes

A*

Word Ladder Solver

Enter starting word:

count

Enter goal word:

world

Select method:

A*

Solve

		<div><div>Result</div><table><tr><th>Step</th><th>Word</th></tr><tr><td>0</td><td>Count</td></tr><tr><td>1</td><td>Mount</td></tr><tr><td>2</td><td>Moult</td></tr><tr><td>3</td><td>Mouls</td></tr><tr><td>4</td><td>Molls</td></tr><tr><td>5</td><td>Mells</td></tr><tr><td>6</td><td>Melos</td></tr><tr><td>7</td><td>Melon</td></tr><tr><td>8</td><td>Meson</td></tr><tr><td>9</td><td>Mason</td></tr></table><div>Number of nodes visited: 679 Time taken: 198 ms Memory usage: 64081400 bytes</div></div>	Step	Word	0	Count	1	Mount	2	Moult	3	Mouls	4	Molls	5	Mells	6	Melos	7	Melon	8	Meson	9	Mason
Step	Word																							
0	Count																							
1	Mount																							
2	Moult																							
3	Mouls																							
4	Molls																							
5	Mells																							
6	Melos																							
7	Melon																							
8	Meson																							
9	Mason																							
3	UCS	<div><div>Word Ladder Solver</div><div><div>Enter starting word:</div><div>bard</div></div><div><div>Enter goal word:</div><div>mint</div></div><div><div>Select method:</div><div>UCS</div></div><div>Solve</div></div>																						

The screenshot shows a window titled "Result" with a dark blue background. It displays a word ladder solution in a table with two columns: "Step" and "Word". The steps are numbered 0 to 4, showing the progression from "Bard" to "Band", "Bant", "Bint", and finally "Mint". Below the table, a light gray rectangular area is present. At the bottom, a white box contains performance statistics: "Number of nodes visited: 3521", "Time taken: 1114 ms", and "Memory usage: 149623120 bytes".

Step	Word
0	Bard
1	Band
2	Bant
3	Bint
4	Mint

Number of nodes visited: 3521
Time taken: 1114 ms
Memory usage: 149623120 bytes

GBFS

The screenshot shows a window titled "Word Ladder Solver" with a dark blue background. It contains three input fields: "Enter starting word:" with the value "bard", "Enter goal word:" with the value "mint", and "Select method:" with a dropdown menu showing "GBFS". Below these fields is a prominent orange button labeled "Solve".

Enter starting word:

Enter goal word:

Select method:

	<div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 665"><div data-bbox="431 210 1414 665">Result</div><table data-bbox="662 310 1190 457"><thead><tr><th>Step</th><th>Word</th></tr></thead><tbody><tr><td>0</td><td>Bard</td></tr><tr><td>1</td><td>Band</td></tr><tr><td>2</td><td>Bant</td></tr><tr><td>3</td><td>Bint</td></tr><tr><td>4</td><td>Mint</td></tr></tbody></table></div><div data-bbox="662 674 1190 793"><div data-bbox="662 674 1190 793">Number of nodes visited: 82 Time taken: 4 ms Memory usage: 10478704 bytes</div></div></div>	Step	Word	0	Bard	1	Band	2	Bant	3	Bint	4	Mint
Step	Word												
0	Bard												
1	Band												
2	Bant												
3	Bint												
4	Mint												
A*	<div data-bbox="431 911 1414 1572"><div data-bbox="431 911 1414 1572">Word Ladder Solver</div><div data-bbox="703 1182 1151 1323"><div data-bbox="703 1182 1151 1213">Enter starting word: <input data-bbox="883 1182 1151 1213" type="text" value="bard"/></div><div data-bbox="703 1218 1151 1251">Enter goal word: <input data-bbox="883 1218 1151 1251" type="text" value="mint"/></div><div data-bbox="703 1255 1151 1287">Select method: <input data-bbox="883 1255 1151 1287" type="text" value="A*"/></div><div data-bbox="703 1291 1151 1323"><div data-bbox="703 1291 1151 1323">Solve</div></div></div></div>												

		<div><div>Result</div><table><tr><th>Step</th><th>Word</th></tr><tr><td>0</td><td>Bard</td></tr><tr><td>1</td><td>Band</td></tr><tr><td>2</td><td>Bant</td></tr><tr><td>3</td><td>Bint</td></tr><tr><td>4</td><td>Mint</td></tr></table><div>Number of nodes visited: 131 Time taken: 11 ms Memory usage: 23068672 bytes</div></div>	Step	Word	0	Bard	1	Band	2	Bant	3	Bint	4	Mint
Step	Word													
0	Bard													
1	Band													
2	Bant													
3	Bint													
4	Mint													
4	UCS	<div><div>Word Ladder Solver</div><div><div>Enter starting word:</div><div>Blue</div></div><div><div>Enter goal word:</div><div>Tart</div></div><div><div>Select method:</div><div>UCS</div></div><div>Solve</div></div>												

Result	
Step	Word
0	Blue
1	Blur
2	Baur
3	Barr
4	Bart
5	Tart
Number of nodes visited: 4563 Time taken: 1962 ms Memory usage: 184450592 bytes	

GBFS

Word Ladder Solver	
Enter starting word:	Blue
Enter goal word:	Tart
Select method:	GBFS
Solve	

	<div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"><div data-bbox="431 210 1414 869"></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div></div>
--	--

		<div><div>Result</div><table><tr><th>Step</th><th>Word</th></tr><tr><td>0</td><td>Blue</td></tr><tr><td>1</td><td>Blur</td></tr><tr><td>2</td><td>Baur</td></tr><tr><td>3</td><td>Barr</td></tr><tr><td>4</td><td>Bart</td></tr><tr><td>5</td><td>Tart</td></tr></table><div>Number of nodes visited: 137 Time taken: 13 ms Memory usage: 22857568 bytes</div></div>	Step	Word	0	Blue	1	Blur	2	Baur	3	Barr	4	Bart	5	Tart
Step	Word															
0	Blue															
1	Blur															
2	Baur															
3	Barr															
4	Bart															
5	Tart															
5	UCS	<div><div>Word Ladder Solver</div><div><div>Enter starting word: boyish</div><div>Enter goal word: tartar</div><div>Select method: UCS</div><div>Solve</div></div></div>														

		<div data-bbox="433 205 1422 871"><p>Result</p><p>No path found from boyish to tartar Number of nodes visited: 6 Time taken: 17 ms Memory usage: 4666024 bytes</p></div>
	GBFS	<div data-bbox="433 909 1422 1575"><p>Word Ladder Solver</p><p>Enter starting word: <input type="text" value="boyish"/></p><p>Enter goal word: <input type="text" value="tartar"/></p><p>Select method: <input type="text" value="GBFS"/></p><p><input type="button" value="Solve"/></p></div>

		<div data-bbox="433 205 1422 871"><p>Result</p><p>No path found from boyish to tartar Number of nodes visited: 6 Time taken: 17 ms Memory usage: 23633968 bytes</p></div>
	A*	<div data-bbox="433 909 1422 1575"><p>Word Ladder Solver</p><p>Enter starting word: <input type="text" value="boyish"/></p><p>Enter goal word: <input type="text" value="tartar"/></p><p>Select method: <input type="text" value="A*"/></p><p><input type="button" value="Solve"/></p></div>

		<div data-bbox="435 205 1422 871"><p>Result</p><p>No path found from boyish to tartar Number of nodes visited: 6 Time taken: 16 ms Memory usage: 4676064 bytes</p></div>
6	UCS	<div data-bbox="435 909 1422 1572"><p>Word Ladder Solver</p><p>Enter starting word: <input type="text" value="Tartar"/></p><p>Enter goal word: <input type="text" value="Horses"/></p><p>Select method: <input type="text" value="UCS"/></p><p><input type="button" value="Solve"/></p></div>

Result

Step	Word
0	Tartar
1	Tarter
2	Carter
3	Cartes
4	Carses
5	Corses
6	Horses

Number of nodes visited: 2930
Time taken: 3280 ms
Memory usage: 20835952 bytes

GBFS

Word Ladder Solver

Enter starting word: Tartar

Enter goal word: Horses

Select method: GBFS

Solve

Result

Step	Word
0	Tartar
1	Tarter
2	Darter
3	Dorter
4	Dorser
5	Dorses
6	Horses

Number of nodes visited: 47
Time taken: 21 ms
Memory usage: 23808312 bytes

A*

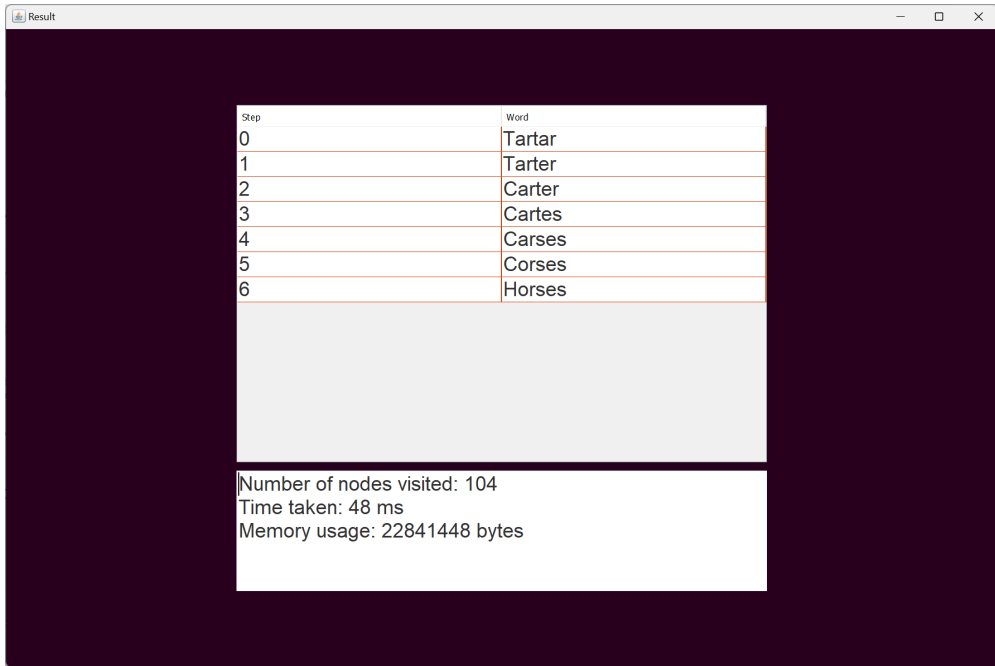
Word Ladder Solver

Enter starting word: Tartar

Enter goal word: Horses

Select method: A*

Solve



The screenshot shows a window titled "Result" with a dark purple background. In the center, a white box contains a table with two columns: "Step" and "Word". The table lists a sequence of words from "Tartar" to "Horses" over 6 steps. Below the table, a light gray rectangular area is present. At the bottom of the white box, performance statistics are displayed: "Number of nodes visited: 104", "Time taken: 48 ms", and "Memory usage: 22841448 bytes".

Step	Word
0	Tartar
1	Tarter
2	Carter
3	Cartes
4	Carses
5	Corses
6	Horses

Number of nodes visited: 104
Time taken: 48 ms
Memory usage: 22841448 bytes

5. Analisis Perbandingan Algoritma UCS, Greedy Best First Search, dan A*

Berdasarkan hasil uji kasus yang telah dilakukan, dapat dilihat perbandingan antara ketiga algoritma yang digunakan dari beberapa aspek, diantaranya adalah panjang path yang dihasilkan, jumlah node yang dikunjungi, waktu eksekusi, dan penggunaan memory.

Panjang path yang dihasilkan dari algoritma UCS dan A* selalu sama, yaitu keduanya selalu menghasilkan hasil yang optimal. Berbeda dengan algoritma GBFS, panjang path yang dihasilkan tidak selalu optimal. Hal ini karena algoritma GBFS hanya mempertimbangkan cost heuristik tanpa memikirkan panjang path saat ini.

Jumlah node yang dikunjungi dari ketiga algoritma selalu berbeda. Hal ini karena ketiga algoritma memiliki sedikit perbedaan dalam proses pencariannya. Pencarian dengan menggunakan algoritma UCS cenderung akan mengunjungi lebih banyak node dibandingkan kedua algoritma lainnya karena algoritma UCS, seperti halnya algoritma BFS, akan mengecek seluruh path yang panjangnya sama terlebih dahulu tanpa ada informasi lain yang memudahkan pencarian. Pencarian dengan algoritma GBFS cenderung akan mengunjungi node lebih sedikit dibandingkan kedua algoritma lainnya karena algoritma GBFS hanya akan mengecek node yang memiliki nilai heuristik yang paling kecil tanpa memikirkan panjang path yang dihasilkan. Pencarian dengan algoritma A* berada di tengah antara algoritma UCS dan GBFS karena algoritma A* merupakan gabungan dari kedua algoritma tersebut yang memperhatikan panjang path dengan pertimbangan nilai heuristiknya.

Waktu eksekusi dari ketiga algoritma ini berbanding lurus dengan jumlah node yang dikunjungi. Semakin banyak node yang dikunjungi maka semakin lama waktu pencarian. Oleh

karena itu, algoritma UCS memiliki waktu eksekusi paling buruk dibandingkan dengan kedua algoritma lainnya sedangkan algoritma GBFS memiliki waktu eksekusi paling baik dibandingkan dengan kedua algoritma lainnya. Algoritma A* sekali lagi memiliki waktu eksekusi yang lebih lama dibandingkan algoritma GBFS tetapi jauh lebih baik dibandingkan dengan algoritma UCS.

Penggunaan memory dari ketiga algoritma ini juga berbanding lurus dengan jumlah node yang dikunjungi. Semakin banyak node yang dikunjungi maka semakin besar jumlah memory yang digunakan untuk melakukan pencarian, begitu juga sebaliknya. Pencarian dengan algoritma UCS cenderung memakan lebih banyak memory dibandingkan kedua algoritma lainnya dan algoritma GBFS cenderung memakan memory paling sedikit dibandingkan kedua algoritma lainnya. Algoritma A* memakan memory yang mirip dengan algoritma GBFS tetapi masih lebih banyak dibandingkan algoritma GBFS.

6. Penjelasan Implementasi Bonus

Implementasi bonus pada tugas kecil ini adalah pengembangan GUI. Pada tugas kecil ini, saya menggunakan Java Swing untuk membuat GUI nya. GUI yang diimplementasikan dirancang untuk menerima input dari pengguna berupa start word dan goal word yang akan divalidasi nilainya. Pengguna dapat memilih algoritma apa yang akan digunakan untuk pencarian. Setelah mengisi input, pengguna dapat melakukan pencarian dengan menekan tombol Solve. Hasil dari pencarian ditampilkan dalam format tabel dan text area.

Lampiran

Link Github: https://github.com/SandWithCheese/Tucil3_13522005

Poin	ya	tidak
1. Program berhasil dijalankan.	✓	
2. Program dapat menemukan rangkaian kata dari start word ke end word sesuai aturan permainan dengan algoritma UCS	✓	
3. Solusi yang diberikan pada algoritma UCS optimal	✓	
4. Program dapat menemukan rangkaian kata dari start word ke end word sesuai aturan permainan dengan algoritma Greedy Best First Search	✓	
5. Program dapat menemukan rangkaian kata dari start word ke end word sesuai aturan permainan dengan algoritma A*	✓	
6. Solusi yang diberikan pada algoritma A* optimal	✓	
7. [Bonus]: Program memiliki tampilan GUI	✓	