

**LAPORAN TUGAS KECIL 03**  
**IF2211 STRATEGI ALGORITMA**

**PENYELESAIAN PERMAINAN WORD LADDER MENGGUNAKAN  
ALGORITMA UCS, GREEDY BEST FIRST SEARCH, DAN A\***



Disusun oleh:

Juan Alfred Widjaya 13522073

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**

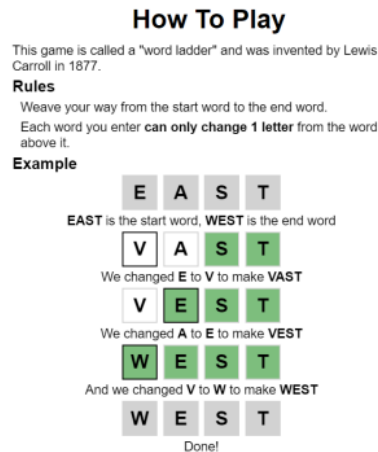
**2024**

## DAFTAR ISI

|  |    |
|--|----|
| DAFTAR ISI.....  | 1  |
| BAB I DESKRIPSI MASALAH.....                           | 2  |
| BAB II LANDASAN TEORI.....                             | 3  |
| 2.1 Algoritma Uniform Cost Search (UCS).....           | 3  |
| 2.2 Algoritma Greedy Best First Search.....            | 3  |
| 2.3 Algoritma A* Search.....                           | 4  |
| BAB III IMPLEMENTASI PROGRAM.....                      | 5  |
| 3.1. Main.java.....                                    | 5  |
| 3.2. SolverGUI.java.....                               | 5  |
| 3.3. Utils.java.....                                   | 8  |
| 3.4. AlgoResult.java.....                              | 8  |
| 3.5. Algorithm.java.....                               | 9  |
| 3.6. SearchUtil.java.....                              | 9  |
| 3.7. UCS.java.....                                     | 11 |
| 3.8. GBFS.java.....                                    | 12 |
| 3.9. ASTAR.java.....                                   | 14 |
| BAB IV IMPLEMENTASI DAN PENGUJIAN.....                 | 16 |
| 4.1. Pengujian.....                                    | 16 |
| 1.1. Pengujian Algoritma Uniform Cost Search.....      | 16 |
| 1.2. Pengujian Algoritma Greedy Best First Search..... | 21 |
| 1.3. Pengujian Algoritma A* Search.....                | 28 |
| 1.4. Hasil Pengujian.....                              | 33 |
| 4.2. Analisis.....                                     | 34 |
| 4.3. Penjelasan Bonus.....                             | 38 |
| BAB V PENUTUP.....                                     | 39 |
| 5.1. Kesimpulan.....                                   | 39 |
| 5.2. Saran.....  | 39 |
| DAFTAR REFERENSI.....                                  | 41 |
| LAMPIRAN.....  | 42 |

# BAB I

## DESKRIPSI MASALAH



Gambar 1. Ilustrasi dan Peraturan Permainan Word Ladder

(Sumber: <https://wordwormdormdork.com/>)

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. Pencarian solusi dari permainan Word Ladder bisa menggunakan beberapa algoritma pencarian seperti algoritma Uniform Cost Search (UCS), algoritma Greedy Best First Search, dan algoritma A\* search.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Algoritma Uniform Cost Search (UCS)**

Algoritma Uniform Cost Search adalah jenis pencarian tanpa informasi yang berfokus pada menemukan jalur dari titik awal ke tujuan dengan biaya total terendah. Dalam algoritma ini, setiap langkah diambil dengan mempertimbangkan biaya terendah dari titik awal ke setiap node yang terhubung. Algoritma ini bekerja di sepanjang ruang pencarian berbobot berarah, mencoba untuk menemukan jalur dengan biaya kumulatif minimum tanpa mempertimbangkan informasi tambahan tentang keadaan node atau ruang pencarian. Biasanya, algoritma ini digunakan untuk mencari jalur terpendek dalam graf berbobot, dimana setiap node diperluas berdasarkan biaya traversal dari node awal. Implementasi umum dari Uniform Cost Search melibatkan penggunaan priority queue, dengan prioritas yang ditentukan oleh biaya operasi terendah atau jarak dari node awal ke node ke- $n$  ( $g(n)$ ). Algoritma ini sering juga dianggap sebagai varian dari algoritma Dijkstra, karena fokusnya pada penurunan biaya operasi untuk mencapai tujuan. Dalam Uniform Cost Search, node sumber dimasukkan terlebih dahulu ke dalam antrian prioritas, dan kemudian node-node lainnya ditambahkan satu per satu sesuai kebutuhan.

#### **2.2 Algoritma Greedy Best First Search**

Algoritma ini memilih untuk mengekspansi node yang paling "menjanjikan" atau tampak mendekati tujuan, berdasarkan estimasi heuristik ( $f(n) = h(n)$ ). Algoritma ini menggunakan heuristik yang mengestimasi biaya terkecil dari node saat ini ke tujuan. Algoritma ini tidak memperdulikan estimasi biaya dari node  $n$  ke tujuan. Sehingga algoritma ini tidak selalu menemukan jalur dengan biaya terendah. Tetapi dengan menggunakan estimasi heuristik yang baik, algoritma ini dapat menemukan jalur lebih cepat dari algoritma UCS dalam segi waktu. Algoritma ini tidak selalu menjamin penemuan jalur terpendek dan dapat terjebak pada solusi lokal yang optimal, terutama dalam graf yang kompleks atau ketika heuristik tidak akurat.

### 2.3 Algoritma A\* Search

Algoritma A\* adalah perpaduan antara UCS dan Greedy Best First Search. Algoritma ini menggunakan fungsi biaya  $f(n) = g(n) + h(n)$ , dimana  $g(n)$  adalah biaya dari titik awal ke node  $n$ , dan  $h(n)$  adalah heuristik yang estima biaya dari node  $n$  ke tujuan. A\* mengekspansi node dengan nilai  $f(n)$  terendah. Heuristik yang digunakan harus *admissible* (tidak pernah overestimate biaya sebenarnya) untuk menjamin bahwa A\* akan menemukan jalur terpendek. A\* efisien dan efektif dalam banyak kasus karena menggabungkan keunggulan UCS dalam menemukan jalur dengan biaya minimal dan kecepatan Greedy Best First Search dalam mendekati tujuan.

## BAB III

### IMPLEMENTASI PROGRAM

Berikut merupakan dari implementasi source code program, beserta penjelasan tiap class dan method yang diimplementasi.

#### 3.1. Main.java

Main.java

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        HashSet<String> dictionary =
            Utils.initializeWordsHashSet("_dictionary_javase.txt");
        SolverGUI.run(dictionary);
    }
}
```

Di dalam file Main.java, terdapat class Main. Class Main memiliki method main yang menginisialisasi kamus bahasa inggris dalam suatu HashSet. Kamus bahasa inggris ini menyimpan seluruh kata dalam bahasa inggris. Setelah inisialisasi fungsi ini memanggil fungsi jalannya program GUI.

#### 3.2. SolverGUI.java

SolverGUI.java

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.awt.event.*;
import java.util.HashSet;

public class SolverGUI {
    private static HashSet<String> words;
    private static JFrame frame;
    private static JTextField startWordField;
    private static JTextField endWordField;
    private static JTable resultTable;
    private static DefaultTableModel model;

    public static void run(HashSet<String> dictionary) {
        words = dictionary;
        SwingUtilities.invokeLater(SolverGUI::initializeGUI);
    }

    private static void initializeGUI() {
        frame = new JFrame("Word Ladder Solver");
    }
}
```

```

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout(frame.getContentPane(), BorderLayout.Y_AXIS));
        frame.getContentPane().setBackground(Color.WHITE);

        frame.add(createWordPanel("Enter start word: ", startWordField = new
        JTextField(15)));
        frame.add(createWordPanel("Enter end word: ", endWordField = new JTextField(15)));
        frame.add(createButtonsPanel());
        setupResultTable();

        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }

    private static JPanel createWordPanel(String labelText, JTextField textField) {
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
        panel.setBackground(Color.WHITE);
        JLabel label = new JLabel(labelText);
        label.setFont(new Font("Arial", Font.BOLD, 14));
        textField.setFont(new Font("Arial", Font.BOLD, 14));
        panel.add(label);
        panel.add(textField);
        return panel;
    }

    private static JPanel createButtonsPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        panel.setBackground(Color.WHITE);
        panel.add(createButton("Search A*", "ASTAR"));
        panel.add(createButton("Search UCS", "UCS"));
        panel.add(createButton("Search GBFS", "GBFS"));
        return panel;
    }

    private static JButton createButton(String buttonText, String method) {
        JButton button = new JButton(buttonText);
        button.addActionListener(e → performSearch(method));
        styleButton(button);
        return button;
    }

    private static void styleButton(JButton button) {
        button.setFocusPainted(false);
        button.setBorderPainted(false);
        button.setBackground(new Color(0, 156, 240));
        button.setFont(new Font("Arial", Font.BOLD, 12));
        button.setForeground(Color.BLACK);
        button.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseEntered(MouseEvent e) {
                button.setBackground(new Color(0, 156, 222));
            }

            @Override
            public void mouseExited(MouseEvent e) {
                button.setBackground(new Color(0, 156, 240));
            }
        });
    }

    private static void setupResultTable() {

```

```

        model = new DefaultTableModel();
        model.addColumn("Step");
        model.addColumn("Word");
        resultTable = new JTable(model) {
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        };
        customizeTableLook();
        JScrollPane scrollPane = new JScrollPane(resultTable);
        scrollPane.setPreferredSize(new Dimension(580, 200));
        frame.add(scrollPane);
    }

    private static void customizeTableLook() {
        resultTable.setFont(new Font("Arial", Font.PLAIN, 12));
        resultTable.setRowHeight(20);
        resultTable.setFillViewportHeight(true);
        resultTable.setBackground(Color.WHITE);
        resultTable.setGridColor(Color.WHITE);
        JTableHeader tableHeader = resultTable.getTableHeader();
        tableHeader.setBackground(Color.WHITE);
        tableHeader.setForeground(Color.BLACK);
        tableHeader.setFont(new Font("Arial", Font.BOLD, 12));
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);
        resultTable.setDefaultRenderer(Object.class, centerRenderer);
    }

    private static void performSearch(String algorithm) {
        String startWord = startWordField.getText().toLowerCase();
        String endWord = endWordField.getText().toLowerCase();

        if (!isValidWord(startWord) || !isValidWord(endWord)) {
            JOptionPane.showMessageDialog(frame, "Error: One or both words are not valid English words.");
            return;
        }

        if (startWord.length() != endWord.length()) {
            JOptionPane.showMessageDialog(frame, "Error: Words have different lengths");
            return;
        }

        AlgoResult searchResult = Algorithm.search(startWord, endWord, algorithm, words);
        updateResultTable(searchResult);
    }

    private static boolean isValidWord(String word) {
        return words.contains(word);
    }

    private static void updateResultTable(AlgoResult searchResult) {
        model.setRowCount(0);
        int step = 1;
        for (String word : searchResult.getPath()) {
            model.addRow(new Object[]{step++, word.toUpperCase()});
        }
        model.addRow(new Object[]{"Execution Time (ms)", searchResult.getExecutionTime()});
        model.addRow(new Object[]{"Words Checked", searchResult.getNodeVisited()});

        for (int i = 0; i < model.getColumnCount(); i++) {
            DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
            centerRenderer.setHorizontalAlignment(JLabel.CENTER);
        }
    }

```



```

        resultTable.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);
    }
}

```

Didalam file SolverGUI.java terdapat class SolverGUI. Class SolverGUI mengatur jalannya program dan visualisasi GUI. Pemanggilan algoritma, dilakukan pada method performSearch. Method ini akan memvalidasi masukan terlebih dahulu. Jika masukan valid, maka akan dilakukan pencarian menggunakan algoritma pilihan sesuai tombol yang ditekan.

### 3.3. Utils.java

Utils.java

```

import java.io.*;
import java.util.*;

public class Utils {
    public static HashSet<String> initializeWordsHashSet(String filename) {
        HashSet<String> words = new HashSet<>();
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = br.readLine()) != null) {
                words.add(line.trim().toLowerCase());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return words;
    }
}

```

Di dalam file Utils.java, terdapat class Utils yang memiliki method statis untuk menginisialisasi HashSet kamus bahasa Inggris dengan membaca isi file txt.

### 3.4. AlgoResult.java

AlgoResult.java

```

import java.util.ArrayList;

public class AlgoResult {
    private ArrayList<String> path;
    private int executionTime;
    private int nodeVisited;

    public ArrayList<String> getPath() {
        return path;
    }

    public void setPath(ArrayList<String> path) {
        this.path = path;
    }

    public int getExecutionTime() {

```

```

        return executionTime;
    }

    public void setExecutionTime(int executionTime) {
        this.executionTime = executionTime;
    }

    public int getNodeVisited() {
        return nodeVisited;
    }

    public void setNodeVisited(int nodeVisited) {
        this.nodeVisited = nodeVisited;
    }
}

```

Di dalam file `AlgoResult.java`, terdapat class `AlgoResult`. Class ini berfungsi sebagai wrapper dari hasil pencarian. Karena merupakan sebuah wrapper, hanya terdapat fungsi getter dan setter pada class ini.

### 3.5. Algorithm.java

#### Algorithm.java

```

import java.util.HashSet;

public class Algorithm {
    public static AlgoResult search(String startWord, String endWord, String algorithm,
        HashSet<String> dictionary) {
        Runtime runtime = Runtime.getRuntime();
        long beforeMemory, afterMemory, usedMemory;

        runtime.gc();
        beforeMemory = runtime.totalMemory() - runtime.freeMemory();

        AlgoResult result;
        if ("UCS".equals(algorithm)) {
            result = UCS.uniformCostSearch(startWord, endWord, dictionary);
        } else if ("GBFS".equals(algorithm)) {
            result = GBFS.greedyBestFirstSearch(startWord, endWord, dictionary);
        } else {
            result = ASTAR.aStarSearch(startWord, endWord, dictionary);
        }

        runtime.gc();
        afterMemory = runtime.totalMemory() - runtime.freeMemory();
        usedMemory = afterMemory - beforeMemory;

        System.out.println(algorithm + " used memory (bytes): " + usedMemory);

        return result;
    }
}

```

Di dalam file `Algorithm.java`, terdapat class `Algorithm` yang memiliki method statis untuk pemanggilan fungsi pencarian berdasarkan masukan. Method statis ini akan mengembalikan objek `AlgoResult` yang merupakan hasil dari pencarian dan juga mendisplay memori yang dipakai algoritma.

### 3.6. SearchUtil.java

#### SearchUtil.java

```
import java.util.ArrayList;
import java.util.HashSet;

class Node {
    String word;
    Node parent;
    int value;

    Node(String word, Node parent, int value) {
        this.word = word;
        this.parent = parent;
        this.value = value;
    }
}

public class SearchUtil {
    public static ArrayList<String> getPossibleWords(String word, HashSet<String>
dictionary) {
        ArrayList<String> possibleWordList = new ArrayList<>();
        char[] chars = word.toCharArray();
        for (int i = 0; i < chars.length; i++) {
            char originalChar = chars[i];
            for (char c = 'a'; c ≤ 'z'; c++) {
                if (c ≠ originalChar) {
                    chars[i] = c;
                    String temp = new String(chars);
                    if (dictionary.contains(temp)) {
                        possibleWordList.add(temp);
                    }
                }
            }
            chars[i] = originalChar;
        }
        return possibleWordList;
    }

    public static int getHeuristicValue(String word, String goal) {
        int mismatch = 0;
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) ≠ goal.charAt(i)) {
                mismatch++;
            }
        }
        return mismatch;
    }
}
```

Di dalam file SearchUtil.java, terdapat dua class yakni class Node dan class SearchUtil. Class Node berfungsi untuk merepresentasikan node yang memiliki atribut word (kata yang ia representasikan), parent (merefereasikan Node parent), dan value (merepresentasikan cost, cost disesuaikan pada algoritma). Pada SearchUtil.java terdapat dua method statis. Method getPossibleWords, mencari seluruh kata valid dalam kamus bahasa inggris yang didapat dari suatu kata awal dengan mengubah salah satu hurufnya. Method

getHeuristicValue, mencari nilai heuristik suatu kata dengan mencari jumlah huruf yang tidak sesuai dengan kata target. Sebagai contoh jika kata target adalah **ABCD**, sedangkan kata yang dicek adalah **BBCE**, maka nilai heuristik dari **BBCE** adalah 2, karena terdapat dua huruf (**BBCE**) yang tidak sesuai dengan kata target. Sehingga semakin banyak kata yang cocok, maka nilai heuristiknya akan lebih rendah.

### 3.7. UCS.java

UCS.java

```
import java.util.*;

public class UCS {
    public static AlgoResult uniformCostSearch(String startWord, String endWord,
        HashSet<String> dictionary) {

        // NOTE: For calculating memory usage
        Runtime runtime = Runtime.getRuntime();
        long beforeMemory, afterMemory, usedMemory;
        runtime.gc();
        beforeMemory = runtime.totalMemory() - runtime.freeMemory();
        //

        long startTime = System.currentTimeMillis();

        PriorityQueue<Node> prioQueue = new PriorityQueue<>(Comparator.comparingInt(node →
        node.value));
        HashMap<String, Integer> visitedCost = new HashMap<>();
        prioQueue.offer(new Node(startWord, null, 0));
        visitedCost.put(startWord, 0);

        Node finalNode = null;
        int nodesVisited = 0;

        while (!prioQueue.isEmpty()) {
            Node current = prioQueue.poll();
            nodesVisited++;

            if (current.word.equals(endWord)) {
                finalNode = current;
                break;
            }

            for (String possibleWord : SearchUtil.getPossibleWords(current.word,
                dictionary)) {
                int newCost = current.value + 1;
                if (!visitedCost.containsKey(possibleWord) || newCost <
                visitedCost.get(possibleWord)) {
                    visitedCost.put(possibleWord, newCost);
                    prioQueue.offer(new Node(possibleWord, current, newCost));
                }
            }
        }

        // NOTE: For calculating memory usage
        runtime.gc();
        afterMemory = runtime.totalMemory() - runtime.freeMemory();
    }
}
```

```

        usedMemory = afterMemory - beforeMemory;
        System.out.println("[UCS] Start: " + startWord.toUpperCase());
        System.out.println("[UCS] End : " + endWord.toUpperCase());
        System.out.println("[UCS] used memory (bytes): " + usedMemory);
        //

        long endTime = System.currentTimeMillis();
        AlgoResult result = new AlgoResult();
        if (finalNode != null) {
            ArrayList<String> path = new ArrayList<>();
            for (Node n = finalNode; n != null; n = n.parent) {
                path.add(n.word);
            }
            Collections.reverse(path);
            result.setPath(path);
        } else {
            ArrayList<String> path = new ArrayList<>();
            if (startWord == endWord) {
                path.add(startWord);
            } else {
                path.add("No path found");
            }
            result.setPath(path);
        }
        result.setNodeVisited(nodesVisited);
        result.setExecutionTime((int) (endTime - startTime));

        return result;
    }
}

```

Di dalam file UCS.java, terdapat class UCS yang memiliki satu method pencarian. Algoritma ini menggunakan PriorityQueue untuk menyimpan dan mengurutkan node berdasarkan biaya (biaya transformasi) dari kata awal terendah, dan HashMap untuk melacak biaya minimal untuk mencapai setiap kata yang sudah dikunjungi. Setiap kata yang dihasilkan dari transformasi yang valid dan belum dikunjungi atau memiliki biaya lebih rendah dari biaya tercatat sebelumnya, akan ditambahkan ke dalam queue. Proses ini berlanjut hingga kata tujuan ditemukan atau queue kosong. Hasil pencarian termasuk jalur kata, jumlah kata yang dikunjungi, dan waktu eksekusi algoritma. Memori yang digunakan juga dicetak pada terminal.

### 3.8. GBFS.java

GBFS.java

```

import java.util.*;

public class GBFS {
    public static AlgoResult greedyBestFirstSearch(String startWord, String endWord,
        HashSet<String> dictionary) {

        // NOTE: For calculating memory usage
        Runtime runtime = Runtime.getRuntime();

```

```

long beforeMemory, afterMemory, usedMemory;
runtime.gc();
beforeMemory = runtime.totalMemory() - runtime.freeMemory();
//

long startTime = System.currentTimeMillis();

PriorityQueue<Node> prioQueue = new PriorityQueue<>(Comparator.comparingInt(node →
node.value));
HashSet<String> queued = new HashSet<>();
prioQueue.offer(new Node(startWord, null, SearchUtil.getHeuristicValue(startWord,
endWord)));
queued.add(startWord);

Node finalNode = null;
int nodesVisited = 0;

while (!prioQueue.isEmpty()) {
    Node current = prioQueue.poll();
    nodesVisited++;

    if (current.word.equals(endWord)) {
        finalNode = current;
        break;
    }

    for (String possibleWord : SearchUtil.getPossibleWords(current.word,
dictionary)) {
        if (!queued.contains(possibleWord)) {
            queued.add(possibleWord);
            prioQueue.offer(new Node(possibleWord, current,
SearchUtil.getHeuristicValue(possibleWord, endWord)));
        }
    }
}

// NOTE: For calculating memory usage
runtime.gc();
afterMemory = runtime.totalMemory() - runtime.freeMemory();
usedMemory = afterMemory - beforeMemory;
System.out.println("[GBFS] Start: " + startWord.toUpperCase());
System.out.println("[GBFS] End : " + endWord.toUpperCase());
System.out.println("[GBFS] used memory (bytes): " + usedMemory);
//

long endTime = System.currentTimeMillis();
AlgoResult result = new AlgoResult();
if (finalNode != null) {
    ArrayList<String> path = new ArrayList<>();
    for (Node n = finalNode; n != null; n = n.parent) {
        path.add(n.word);
    }
    Collections.reverse(path);
    result.setPath(path);
} else {
    ArrayList<String> path = new ArrayList<>();
    if (startWord == endWord) {
        path.add(startWord);
    } else {
        path.add("No path found");
    }
    result.setPath(path);
}
}

```

```

        result.setNodeVisited(nodesVisited);
        result.setExecutionTime((int) (endTime - startTime));

        return result;
    }
}

```

Di dalam file GBFS.java, terdapat class GBFS yang memiliki satu method pencarian. Algoritma ini menggunakan PriorityQueue untuk menyimpan dan mengurutkan node berdasarkan nilai heuristik (didapat dari fungsi getHeuristicValue) terendah, dan HashSet untuk melacak setiap kata yang sudah dikunjungi. Setiap kata yang dihasilkan dari transformasi yang valid dan belum dikunjungi sebelumnya, akan ditambahkan ke dalam queue. Proses ini berlanjut hingga kata tujuan ditemukan atau queue kosong. Hasil pencarian termasuk jalur kata, jumlah kata yang dikunjungi, dan waktu eksekusi algoritma. Memori yang digunakan juga dicetak pada terminal.

### 3.9. ASTAR.java

ASTAR.java

```

import java.util.*;

public class ASTAR {
    public static AlgoResult aStarSearch(String startWord, String endWord, HashSet<String>
        dictionary) {

        // NOTE: For calculating memory usage
        Runtime runtime = Runtime.getRuntime();
        long beforeMemory, afterMemory, usedMemory;
        runtime.gc();
        beforeMemory = runtime.totalMemory() - runtime.freeMemory();
        //

        long startTime = System.currentTimeMillis();

        PriorityQueue<Node> prioQueue = new PriorityQueue<>(Comparator.comparingInt(node ->
            node.value));
        HashMap<String, Integer> visitedCost = new HashMap<>();
        prioQueue.offer(new Node(startWord, null, 0 +
            SearchUtil.getHeuristicValue(startWord, endWord)));
        visitedCost.put(startWord, 0);

        Node finalNode = null;
        int nodesVisited = 0;

        while (!prioQueue.isEmpty()) {
            Node current = prioQueue.poll();

            if (visitedCost.get(current.word) < current.value -
                SearchUtil.getHeuristicValue(current.word, endWord)) {
                continue;
            }

            nodesVisited++;

```

```

        if (current.word.equals(endWord)) {
            finalNode = current;
            break;
        }

        for (String possibleWord : SearchUtil.getPossibleWords(current.word,
dictionary)) {
            int gNew = visitedCost.get(current.word) + 1;
            int fNew = gNew + SearchUtil.getHeuristicValue(possibleWord, endWord);
            if (!visitedCost.containsKey(possibleWord) || gNew <
visitedCost.get(possibleWord)) {
                visitedCost.put(possibleWord, gNew);
                prioQueue.offer(new Node(possibleWord, current, fNew));
            }
        }
    }
}

// NOTE: For calculating memory usage
runtime.gc();
afterMemory = runtime.totalMemory() - runtime.freeMemory();
usedMemory = afterMemory - beforeMemory;
System.out.println("[A*] Start: " + startWord.toUpperCase());
System.out.println("[A*] End : " + endWord.toUpperCase());
System.out.println("[A*] used memory (bytes): " + usedMemory);
//

long endTime = System.currentTimeMillis();
AlgoResult result = new AlgoResult();
if (finalNode != null) {
    ArrayList<String> path = new ArrayList<>();
    for (Node n = finalNode; n != null; n = n.parent) {
        path.add(n.word);
    }
    Collections.reverse(path);
    result.setPath(path);
} else {
    ArrayList<String> path = new ArrayList<>();
    if (startWord == endWord) {
        path.add(startWord);
    } else {
        path.add("No path found");
    }
    result.setPath(path);
}
result.setNodeVisited(nodesVisited);
result.setExecutionTime((int) (endTime - startTime));

return result;
}
}

```

Di dalam file ASTAR.java, terdapat class ASTAR yang memiliki satu method pencarian. Algoritma ini menggunakan PriorityQueue untuk menyimpan dan mengurutkan node berdasarkan nilai heuristik (didapat dari fungsi getHeuristicValue) ditambah dengan jarak atau biaya transformasi, dari yang terendah, dan HashMap untuk melacak biaya transformasi (jarak) minimal untuk mencapai setiap kata yang sudah dikunjungi. Setiap kata yang dihasilkan dari transformasi yang valid dan belum dikunjungi atau memiliki biaya (jarak) lebih rendah dari biaya tercatat sebelumnya, akan ditambahkan ke dalam queue.



Proses ini berlanjut hingga kata tujuan ditemukan atau queue kosong. Hasil pencarian termasuk jalur kata, jumlah kata yang dikunjungi, dan waktu eksekusi algoritma. Memori yang digunakan juga dicetak pada terminal.

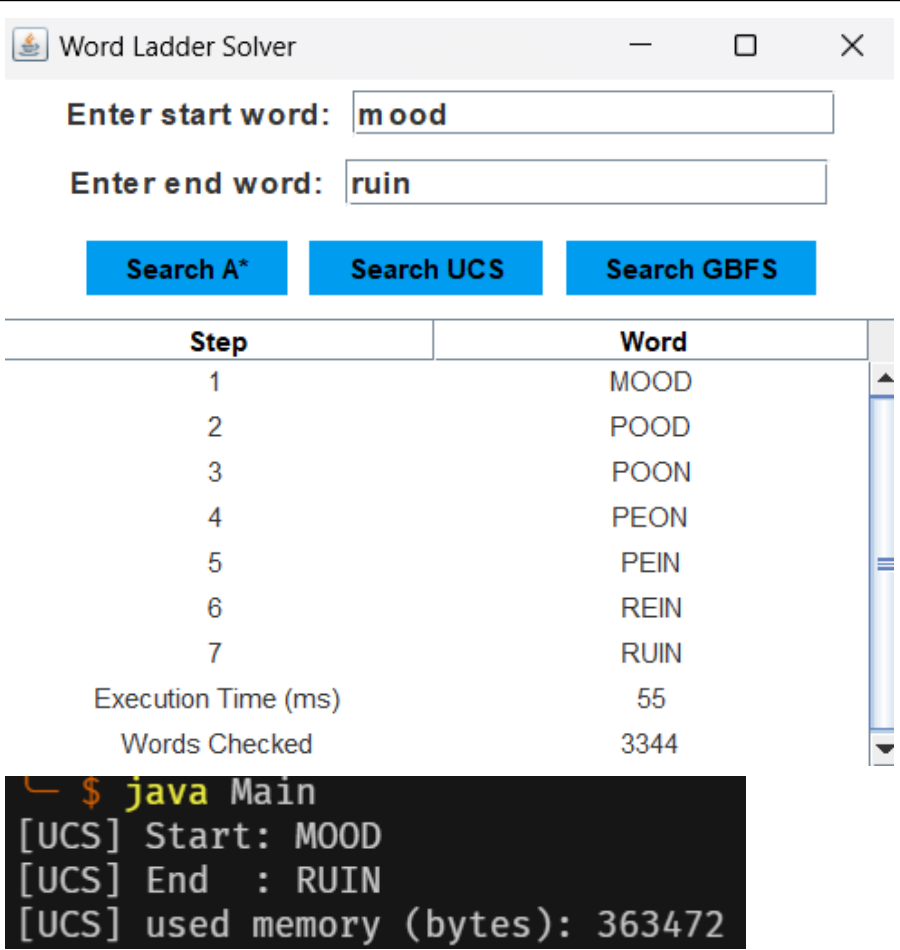
## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Pengujian

##### 1.1. Pengujian Algoritma Uniform Cost Search

Tabel 1. Pengujian Algoritma Uniform Cost Search

| No.  | Hasil Pengujian  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
|------|--|------|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|
| 1    |  <p>The screenshot shows a Java application window titled "Word Ladder Solver". It has two input fields: "Enter start word:" with the value "mood" and "Enter end word:" with the value "ruin". Below these are three buttons: "Search A*", "Search UCS" (which is highlighted), and "Search GBFS". Below the buttons is a table showing the search path:</p> <table border="1"> <thead> <tr> <th>Step</th> <th>Word</th> </tr> </thead> <tbody> <tr><td>1</td><td>MOOD</td></tr> <tr><td>2</td><td>POOD</td></tr> <tr><td>3</td><td>POON</td></tr> <tr><td>4</td><td>PEON</td></tr> <tr><td>5</td><td>PEIN</td></tr> <tr><td>6</td><td>REIN</td></tr> <tr><td>7</td><td>RUIN</td></tr> </tbody> </table> <p>Below the table, it shows "Execution Time (ms)" as 55 and "Words Checked" as 3344. At the bottom, there is a terminal window showing the command prompt output:</p> <pre> \$ java Main [UCS] Start: MOOD [UCS] End : RUIN [UCS] used memory (bytes): 363472 </pre> | Step | Word | 1 | MOOD | 2 | POOD | 3 | POON | 4 | PEON | 5 | PEIN | 6 | REIN | 7 | RUIN |
| Step | Word   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 1    | MOOD   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 2    | POOD   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 3    | POON   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 4    | PEON   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 5    | PEIN   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 6    | REIN   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 7    | RUIN   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |

2

Word Ladder Solver

Enter start word:

Enter end word:

| Step | Word  |
|------|-------|
| 1    | LOVER |
| 2    | LOVES |
| 3    | LORES |
| 4    | TORES |
| 5    | TORTS |
| 6    | TOOTS |
| 7    | TOOTH |
| 8    | TROTH |
| 9    | FROTH |
| 10   | FROSH |
| 11   | FRESH |

Execution Time (ms) 60

Words Checked 6726

```

$ java Main
[UCS] Start: LOVER
[UCS] End : FRESH
[UCS] used memory (bytes): 682896
  
```

3

Word Ladder Solver

Enter start word:

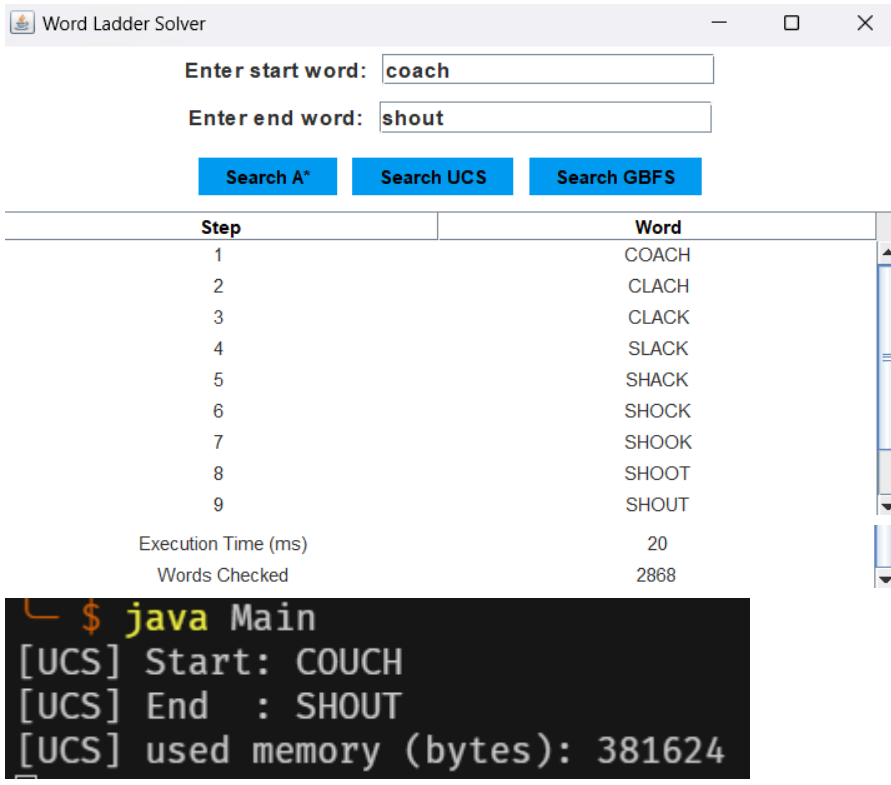
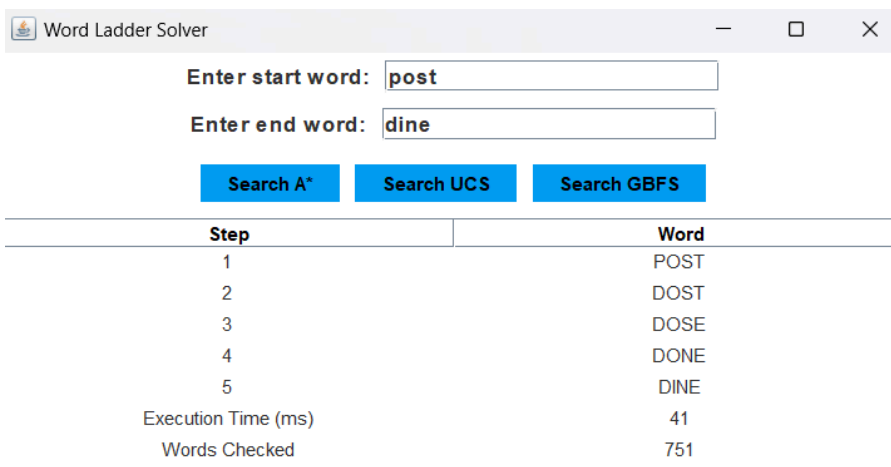
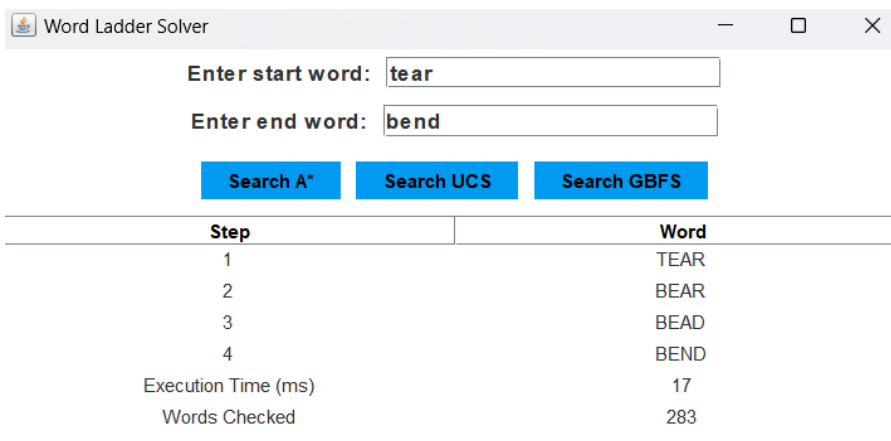
Enter end word:

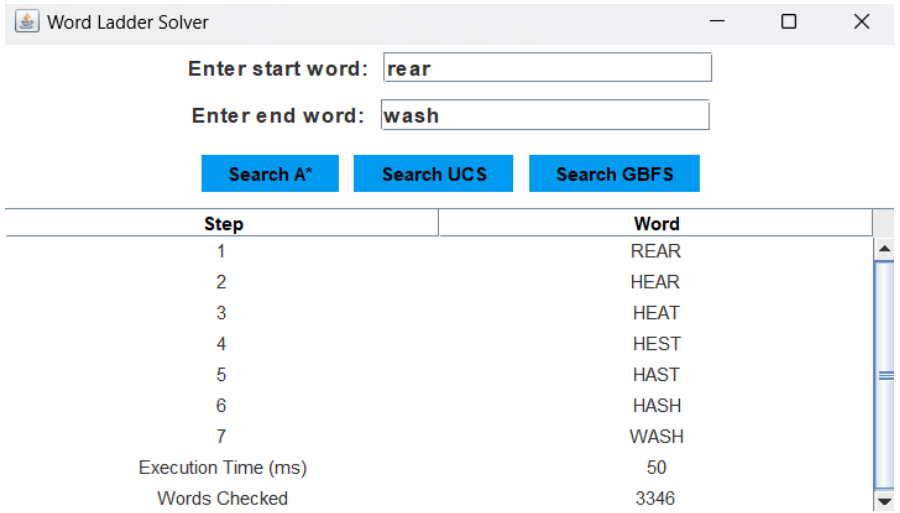
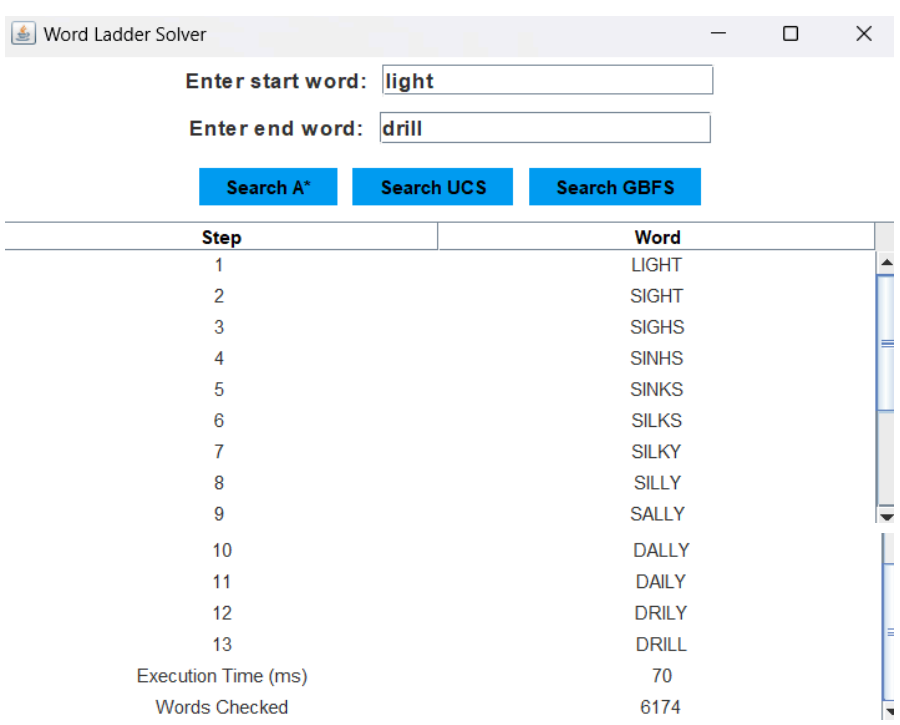
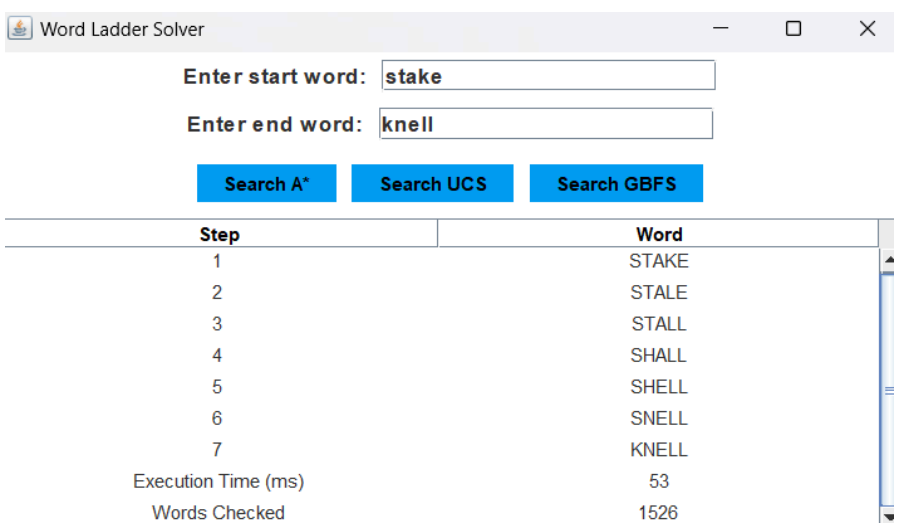
| Step | Word  |
|------|-------|
| 1    | STORY |
| 2    | STORE |
| 3    | SHORE |
| 4    | SHIRE |
| 5    | SHIRR |
| 6    | SHIER |
| 7    | SHIED |
| 8    | SHRED |
| 9    | SIRED |
| 10   | TIRED |

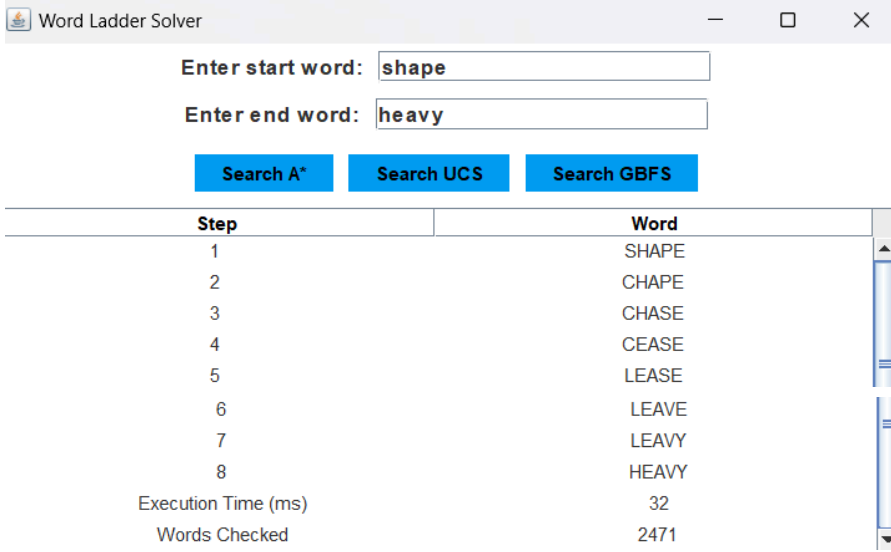
Execution Time (ms) 21

Words Checked 3203

|                     | <pre> \$ java Main [UCS] Start: STORY [UCS] End  : TIRED [UCS] used memory (bytes): 498376 </pre>  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
|---------------------|--|------|------|---|---------------|---------------------|--------|---------------|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|--------|----|--------|----|--------|----|--------|----|--------|----|--------|----|--------|----|--------|----|--------|---------------------|----|---------------|------|
| 4                   | <div> <div>Word Ladder Solver</div> <div> Enter start word: <input type="text" value="branch"/> Enter end word: <input type="text" value="remain"/> </div> <div> <div>Search A*</div> <div>Search UCS</div> <div>Search GBFS</div> </div> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr> <td>1</td><td>NO PATH FOUND</td></tr> <tr> <td>Execution Time (ms)</td><td>2</td></tr> <tr> <td>Words Checked</td><td>43</td></tr> </tbody> </table> <pre> \$ java Main [UCS] Start: BRANCH [UCS] End  : REMAIN [UCS] used memory (bytes): 11352 </pre> </div>  | Step | Word | 1 | NO PATH FOUND | Execution Time (ms) | 2      | Words Checked | 43     |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| Step                | Word   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 1                   | NO PATH FOUND  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| Execution Time (ms) | 2  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| Words Checked       | 43   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 5                   | <div> <div>Word Ladder Solver</div> <div> Enter start word: <input type="text" value="temple"/> Enter end word: <input type="text" value="happen"/> </div> <div> <div>Search A*</div> <div>Search UCS</div> <div>Search GBFS</div> </div> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>TEMPLE</td></tr> <tr><td>2</td><td>SEMPLE</td></tr> <tr><td>3</td><td>SIMPLE</td></tr> <tr><td>4</td><td>RIMPLE</td></tr> <tr><td>5</td><td>RUMPLE</td></tr> <tr><td>6</td><td>RUMBLE</td></tr> <tr><td>7</td><td>BUMBLE</td></tr> <tr><td>8</td><td>BURBLE</td></tr> <tr><td>9</td><td>BURGLE</td></tr> <tr><td>10</td><td>BURGEE</td></tr> <tr><td>11</td><td>BARGEE</td></tr> <tr><td>12</td><td>BARGED</td></tr> <tr><td>13</td><td>BARKED</td></tr> <tr><td>14</td><td>HARKED</td></tr> <tr><td>15</td><td>HARPED</td></tr> <tr><td>16</td><td>HAPPED</td></tr> <tr><td>17</td><td>HAPPEN</td></tr> <tr> <td>Execution Time (ms)</td><td>18</td></tr> <tr> <td>Words Checked</td><td>2059</td></tr> </tbody> </table> <pre> \$ java Main [UCS] Start: TEMPLE [UCS] End  : HAPPEN [UCS] used memory (bytes): 99544 </pre> </div> | Step | Word | 1 | TEMPLE        | 2                   | SEMPLE | 3             | SIMPLE | 4 | RIMPLE | 5 | RUMPLE | 6 | RUMBLE | 7 | BUMBLE | 8 | BURBLE | 9 | BURGLE | 10 | BURGEE | 11 | BARGEE | 12 | BARGED | 13 | BARKED | 14 | HARKED | 15 | HARPED | 16 | HAPPED | 17 | HAPPEN | Execution Time (ms) | 18 | Words Checked | 2059 |
| Step                | Word   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 1                   | TEMPLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 2                   | SEMPLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 3                   | SIMPLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 4                   | RIMPLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 5                   | RUMPLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 6                   | RUMBLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 7                   | BUMBLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 8                   | BURBLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 9                   | BURGLE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 10                  | BURGEE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 11                  | BARGEE   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 12                  | BARGED   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 13                  | BARKED   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 14                  | HARKED   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 15                  | HARPED   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 16                  | HAPPED   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| 17                  | HAPPEN   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| Execution Time (ms) | 18   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |
| Words Checked       | 2059   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |    |        |                     |    |               |      |

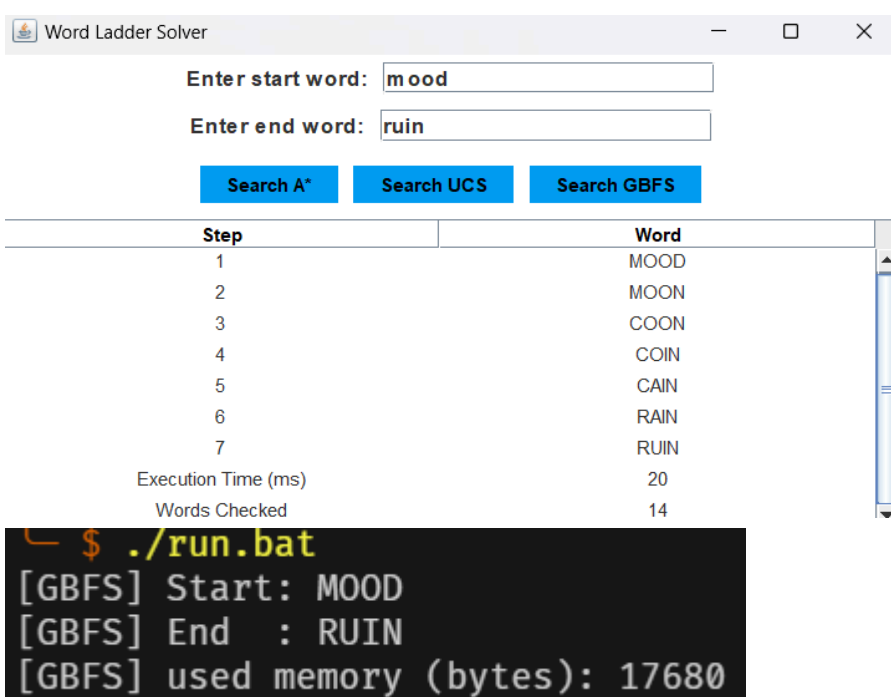
| 6    |  <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="coach"/></p> <p>Enter end word: <input type="text" value="shout"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th> <th>Word</th> </tr> </thead> <tbody> <tr><td>1</td><td>COACH</td></tr> <tr><td>2</td><td>CLACH</td></tr> <tr><td>3</td><td>CLACK</td></tr> <tr><td>4</td><td>SLACK</td></tr> <tr><td>5</td><td>SHACK</td></tr> <tr><td>6</td><td>SHOCK</td></tr> <tr><td>7</td><td>SHOOK</td></tr> <tr><td>8</td><td>SHOOT</td></tr> <tr><td>9</td><td>SHOUT</td></tr> </tbody> </table> <p>Execution Time (ms) 20</p> <p>Words Checked 2868</p> <pre> \$ java Main [UCS] Start: COUCH [UCS] End : SHOUT [UCS] used memory (bytes): 381624 </pre> | Step | Word | 1 | COACH | 2 | CLACH | 3 | CLACK | 4 | SLACK | 5 | SHACK | 6 | SHOCK | 7 | SHOOK | 8 | SHOOT | 9 | SHOUT |
|------|--|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 1    | COACH  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 2    | CLACH  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 3    | CLACK  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 4    | SLACK  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 5    | SHACK  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 6    | SHOCK  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 7    | SHOOK  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 8    | SHOOT  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 9    | SHOUT  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 7    |  <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="post"/></p> <p>Enter end word: <input type="text" value="dine"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th> <th>Word</th> </tr> </thead> <tbody> <tr><td>1</td><td>POST</td></tr> <tr><td>2</td><td>DOST</td></tr> <tr><td>3</td><td>DOSE</td></tr> <tr><td>4</td><td>DONE</td></tr> <tr><td>5</td><td>DINE</td></tr> </tbody> </table> <p>Execution Time (ms) 41</p> <p>Words Checked 751</p>  | Step | Word | 1 | POST  | 2 | DOST  | 3 | DOSE  | 4 | DONE  | 5 | DINE  |   |       |   |       |   |       |   |       |
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 1    | POST   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 2    | DOST   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 3    | DOSE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 4    | DONE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 5    | DINE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 8    |  <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="tear"/></p> <p>Enter end word: <input type="text" value="bend"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th> <th>Word</th> </tr> </thead> <tbody> <tr><td>1</td><td>TEAR</td></tr> <tr><td>2</td><td>BEAR</td></tr> <tr><td>3</td><td>BEAD</td></tr> <tr><td>4</td><td>BEND</td></tr> </tbody> </table> <p>Execution Time (ms) 17</p> <p>Words Checked 283</p>   | Step | Word | 1 | TEAR  | 2 | BEAR  | 3 | BEAD  | 4 | BEND  |   |       |   |       |   |       |   |       |   |       |
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 1    | TEAR   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 2    | BEAR   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 3    | BEAD   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 4    | BEND   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |

| 9    |  <p>Word Ladder Solver</p> <p>Enter start word: rear</p> <p>Enter end word: wash</p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>REAR</td></tr> <tr><td>2</td><td>HEAR</td></tr> <tr><td>3</td><td>HEAT</td></tr> <tr><td>4</td><td>HEST</td></tr> <tr><td>5</td><td>HAST</td></tr> <tr><td>6</td><td>HASH</td></tr> <tr><td>7</td><td>WASH</td></tr> </tbody> </table> <p>Execution Time (ms) 50</p> <p>Words Checked 3346</p>   | Step | Word | 1 | REAR  | 2 | HEAR  | 3 | HEAT  | 4 | HEST  | 5 | HAST  | 6 | HASH  | 7 | WASH  |   |       |   |       |    |       |    |       |    |       |    |       |
|------|--|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|----|-------|----|-------|----|-------|----|-------|
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 1    | REAR   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 2    | HEAR   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 3    | HEAT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 4    | HEST   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 5    | HAST   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 6    | HASH   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 7    | WASH   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 10   |  <p>Word Ladder Solver</p> <p>Enter start word: light</p> <p>Enter end word: drill</p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>LIGHT</td></tr> <tr><td>2</td><td>SIGHT</td></tr> <tr><td>3</td><td>SIGHS</td></tr> <tr><td>4</td><td>SINHS</td></tr> <tr><td>5</td><td>SINKS</td></tr> <tr><td>6</td><td>SILKS</td></tr> <tr><td>7</td><td>SILKY</td></tr> <tr><td>8</td><td>SILLY</td></tr> <tr><td>9</td><td>SALLY</td></tr> <tr><td>10</td><td>DALLY</td></tr> <tr><td>11</td><td>DAILY</td></tr> <tr><td>12</td><td>DRILY</td></tr> <tr><td>13</td><td>DRILL</td></tr> </tbody> </table> <p>Execution Time (ms) 70</p> <p>Words Checked 6174</p> | Step | Word | 1 | LIGHT | 2 | SIGHT | 3 | SIGHS | 4 | SINHS | 5 | SINKS | 6 | SILKS | 7 | SILKY | 8 | SILLY | 9 | SALLY | 10 | DALLY | 11 | DAILY | 12 | DRILY | 13 | DRILL |
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 1    | LIGHT  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 2    | SIGHT  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 3    | SIGHS  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 4    | SINHS  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 5    | SINKS  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 6    | SILKS  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 7    | SILKY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 8    | SILLY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 9    | SALLY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 10   | DALLY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 11   | DAILY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 12   | DRILY  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 13   | DRILL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 11   |  <p>Word Ladder Solver</p> <p>Enter start word: stake</p> <p>Enter end word: knell</p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>STAKE</td></tr> <tr><td>2</td><td>STALE</td></tr> <tr><td>3</td><td>STALL</td></tr> <tr><td>4</td><td>SHALL</td></tr> <tr><td>5</td><td>SHELL</td></tr> <tr><td>6</td><td>SNELL</td></tr> <tr><td>7</td><td>KNELL</td></tr> </tbody> </table> <p>Execution Time (ms) 53</p> <p>Words Checked 1526</p>  | Step | Word | 1 | STAKE | 2 | STALE | 3 | STALL | 4 | SHALL | 5 | SHELL | 6 | SNELL | 7 | KNELL |   |       |   |       |    |       |    |       |    |       |    |       |
| Step | Word   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 1    | STAKE  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 2    | STALE  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 3    | STALL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 4    | SHALL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 5    | SHELL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 6    | SNELL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 7    | KNELL  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |

| 12   |  <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="shape"/></p> <p>Enter end word: <input type="text" value="heavy"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>SHAPE</td></tr> <tr><td>2</td><td>CHAPE</td></tr> <tr><td>3</td><td>CHASE</td></tr> <tr><td>4</td><td>CEASE</td></tr> <tr><td>5</td><td>LEASE</td></tr> <tr><td>6</td><td>LEAVE</td></tr> <tr><td>7</td><td>LEAVY</td></tr> <tr><td>8</td><td>HEAVY</td></tr> </tbody> </table> <p>Execution Time (ms) 32</p> <p>Words Checked 2471</p> | Step | Word | 1 | SHAPE | 2 | CHAPE | 3 | CHASE | 4 | CEASE | 5 | LEASE | 6 | LEAVE | 7 | LEAVY | 8 | HEAVY |
|------|---|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|
| Step | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 1    | SHAPE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 2    | CHAPE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 3    | CHASE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 4    | CEASE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 5    | LEASE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 6    | LEAVE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 7    | LEAVY   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |
| 8    | HEAVY   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |

## 1.2. Pengujian Algoritma Greedy Best First Search

Tabel 2. Pengujian Algoritma Greedy Best First Search

| No.  | Hasil Pengujian   |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
|------|---|------|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|
| 1    |  <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="mood"/></p> <p>Enter end word: <input type="text" value="ruin"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>MOOD</td></tr> <tr><td>2</td><td>MOON</td></tr> <tr><td>3</td><td>COON</td></tr> <tr><td>4</td><td>COIN</td></tr> <tr><td>5</td><td>CAIN</td></tr> <tr><td>6</td><td>RAIN</td></tr> <tr><td>7</td><td>RUIN</td></tr> </tbody> </table> <p>Execution Time (ms) 20</p> <p>Words Checked 14</p> <pre> \$ ./run.bat [GBFS] Start: MOOD [GBFS] End : RUIN [GBFS] used memory (bytes): 17680 </pre> | Step | Word | 1 | MOOD | 2 | MOON | 3 | COON | 4 | COIN | 5 | CAIN | 6 | RAIN | 7 | RUIN |
| Step | Word  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 1    | MOOD  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 2    | MOON  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 3    | COON  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 4    | COIN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 5    | CAIN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 6    | RAIN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |
| 7    | RUIN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |

2

Word Ladder Solver

Enter start word:

Enter end word:

Search A\* Search UCS Search GBFS

| Step | Word  |
|------|-------|
| 1    | LOVER |
| 2    | LOVES |
| 3    | LOXES |
| 4    | FOXES |
| 5    | FIXES |
| 6    | FINES |
| 7    | FINDS |
| 8    | FENDS |
| 9    | FEEDS |
| 10   | FEELS |
| 11   | FELLS |
| 12   | FELTS |
| 13   | FEATS |
| 14   | FRATS |
| 15   | FRITS |
| 16   | FRITH |
| 17   | FROTH |
| 18   | FROSH |
| 19   | FRESH |

Execution Time (ms) 24

Words Checked 65

```

$ ./run.bat
[GBFS] Start: LOVER
[GBFS] End : FRESH
[GBFS] used memory (bytes): 60072
  
```

3

Word Ladder Solver

Enter start word:

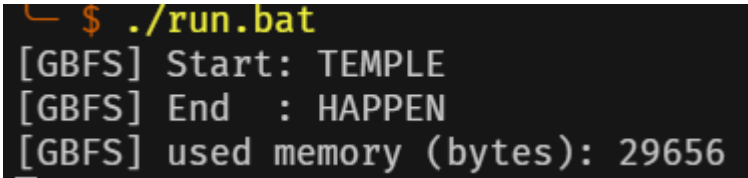
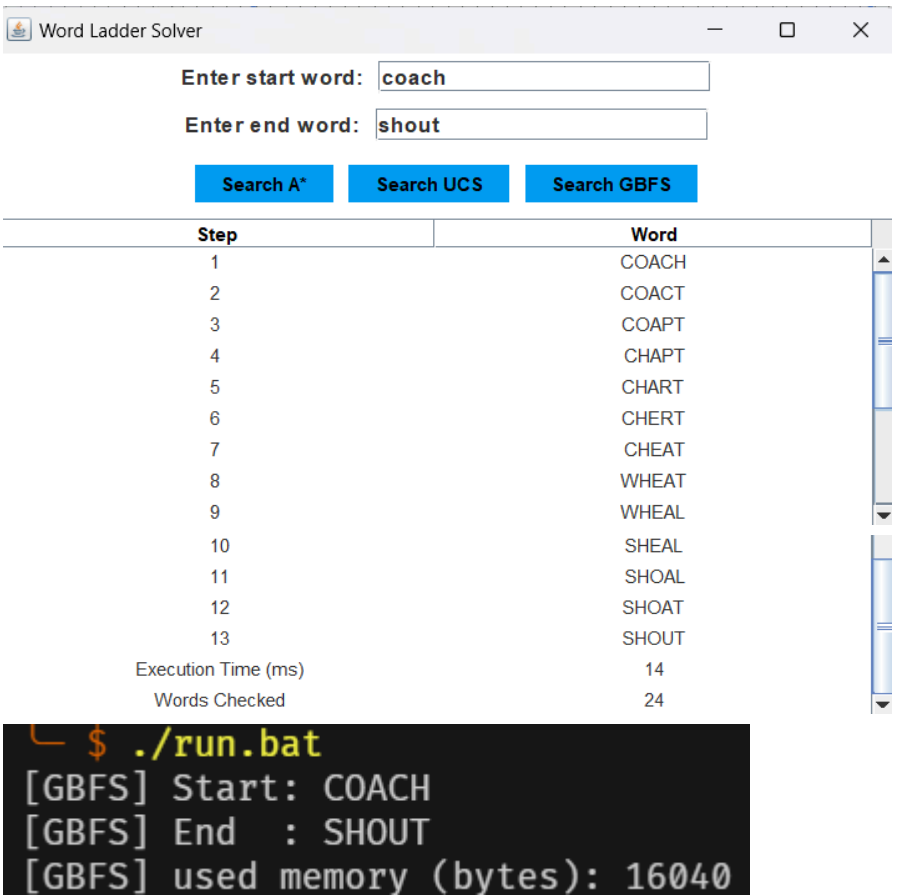
Enter end word:

Search A\* Search UCS Search GBFS

| Step | Word  |
|------|-------|
| 1    | STORY |
| 2    | STORK |
| 3    | STOOK |
| 4    | STOOD |
| 5    | SNOOD |
| 6    | SNOOT |
| 7    | SNOUT |
| 8    | STOUT |
| 9    | STRUT |



|                     | <pre> 10          STRUM 11          SCRUM 12          SCRAM 13          SCRAP 14          STRAP 15          STREP 16          STREW 17          SHREW 18          SHRED 19          SIRED 20          TIRED Execution Time (ms)      26 Words Checked           36 </pre> <pre> \$ ./run.bat [GBFS] Start: STORY [GBFS] End  : TIRED [GBFS] used memory (bytes): 21104 </pre>   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
|---------------------|---|------|------|---|---------------|---------------------|--------|---------------|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|--------|
| 4                   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="branch"/></p> <p>Enter end word: <input type="text" value="remain"/></p> <p> <input type="button" value="Search A*"/> <input type="button" value="Search UCS"/> <input type="button" value="Search GBFS"/> </p> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr> <td>1</td><td>NO PATH FOUND</td></tr> <tr> <td>Execution Time (ms)</td><td>23</td></tr> <tr> <td>Words Checked</td><td>43</td></tr> </tbody> </table> <pre> \$ ./run.bat [GBFS] Start: BRANCH [GBFS] End  : REMAIN [GBFS] used memory (bytes): 11064 </pre>   | Step | Word | 1 | NO PATH FOUND | Execution Time (ms) | 23     | Words Checked | 43     |   |        |   |        |   |        |   |        |   |        |   |        |
| Step                | Word  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 1                   | NO PATH FOUND   |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| Execution Time (ms) | 23  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| Words Checked       | 43  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 5                   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="temple"/></p> <p>Enter end word: <input type="text" value="happen"/></p> <p> <input type="button" value="Search A*"/> <input type="button" value="Search UCS"/> <input type="button" value="Search GBFS"/> </p> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>TEMPLE</td></tr> <tr><td>2</td><td>SEMPLE</td></tr> <tr><td>3</td><td>SIMPLE</td></tr> <tr><td>4</td><td>RIMPLE</td></tr> <tr><td>5</td><td>RUMPLE</td></tr> <tr><td>6</td><td>RUMPLY</td></tr> <tr><td>7</td><td>RUMBLY</td></tr> <tr><td>8</td><td>HUMBLY</td></tr> <tr><td>9</td><td>HUBBLY</td></tr> </tbody> </table> | Step | Word | 1 | TEMPLE        | 2                   | SEMPLE | 3             | SIMPLE | 4 | RIMPLE | 5 | RUMPLE | 6 | RUMPLY | 7 | RUMBLY | 8 | HUMBLY | 9 | HUBBLY |
| Step                | Word  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 1                   | TEMPLE  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 2                   | SEMPLE  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 3                   | SIMPLE  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 4                   | RIMPLE  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 5                   | RUMPLE  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 6                   | RUMPLY  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 7                   | RUMBLY  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 8                   | HUMBLY  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |
| 9                   | HUBBLY  |      |      |   |               |                     |        |               |        |   |        |   |        |   |        |   |        |   |        |   |        |

|   |   |
|---|---|
|   | <pre> 22 23 24 25 26 27 28 Execution Time (ms) Words Checked HAGGED HANGED HANTED HASTED HASPED HAPPED HAPPEN 23 77 </pre>  |
| 6 |    |

| 7    | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="bulb"/></p> <p>Enter end word: <input type="text" value="wait"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>BULB</td></tr> <tr><td>2</td><td>BULK</td></tr> <tr><td>3</td><td>BALK</td></tr> <tr><td>4</td><td>WALK</td></tr> <tr><td>5</td><td>WARK</td></tr> <tr><td>6</td><td>WART</td></tr> <tr><td>7</td><td>WAIT</td></tr> </tbody> </table> <p>Execution Time (ms) 13</p> <p>Words Checked 8</p>  | Step | Word | 1 | BULB | 2 | BULK | 3 | BALK | 4 | WALK | 5 | WARK | 6 | WART | 7 | WAIT |   |      |   |      |    |      |    |      |    |      |    |      |
|------|---|------|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|----|------|----|------|----|------|----|------|
| Step | Word  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 1    | BULB  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 2    | BULK  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 3    | BALK  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 4    | WALK  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 5    | WARK  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 6    | WART  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 7    | WAIT  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 8    | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="oven"/></p> <p>Enter end word: <input type="text" value="knot"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>OVEN</td></tr> <tr><td>2</td><td>EVEN</td></tr> <tr><td>3</td><td>EVES</td></tr> <tr><td>4</td><td>EYES</td></tr> <tr><td>5</td><td>EYAS</td></tr> <tr><td>6</td><td>ERAS</td></tr> <tr><td>7</td><td>EROS</td></tr> <tr><td>8</td><td>BROS</td></tr> <tr><td>9</td><td>BOOS</td></tr> <tr><td>10</td><td>BOOT</td></tr> <tr><td>11</td><td>SOOT</td></tr> <tr><td>12</td><td>SNOT</td></tr> <tr><td>13</td><td>KNOT</td></tr> </tbody> </table> <p>Execution Time (ms) 20</p> <p>Words Checked 29</p> | Step | Word | 1 | OVEN | 2 | EVEN | 3 | EVES | 4 | EYES | 5 | EYAS | 6 | ERAS | 7 | EROS | 8 | BROS | 9 | BOOS | 10 | BOOT | 11 | SOOT | 12 | SNOT | 13 | KNOT |
| Step | Word  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 1    | OVEN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 2    | EVEN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 3    | EVES  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 4    | EYES  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 5    | EYAS  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 6    | ERAS  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 7    | EROS  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 8    | BROS  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 9    | BOOS  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 10   | BOOT  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 11   | SOOT  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 12   | SNOT  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 13   | KNOT  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 9    | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="slap"/></p> <p>Enter end word: <input type="text" value="calm"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>SLAP</td></tr> <tr><td>2</td><td>CLAP</td></tr> <tr><td>3</td><td>CLAM</td></tr> <tr><td>4</td><td>CRAM</td></tr> <tr><td>5</td><td>CRAW</td></tr> <tr><td>6</td><td>CROW</td></tr> <tr><td>7</td><td>CHOW</td></tr> <tr><td>8</td><td>CHON</td></tr> <tr><td>9</td><td>CHIN</td></tr> </tbody> </table>   | Step | Word | 1 | SLAP | 2 | CLAP | 3 | CLAM | 4 | CRAM | 5 | CRAW | 6 | CROW | 7 | CHOW | 8 | CHON | 9 | CHIN |    |      |    |      |    |      |    |      |
| Step | Word  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 1    | SLAP  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 2    | CLAP  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 3    | CLAM  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 4    | CRAM  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 5    | CRAW  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 6    | CROW  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 7    | CHOW  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 8    | CHON  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |
| 9    | CHIN  |      |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |   |      |    |      |    |      |    |      |    |      |

|                     | <div> <div>10</div> <div>CARN</div> <div>11</div> <div>CARN</div> <div>12</div> <div>CARK</div> <div>13</div> <div>CALK</div> <div>14</div> <div>CALM</div> <div>Execution Time (ms)</div> <div>14</div> <div>Words Checked</div> <div>31</div> </div>  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
|---------------------|---|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|---------------------|----|---------------|-----|
| 10                  | <div> <div>Word Ladder Solver</div> <div> <div>Enter start word: shelf</div> <div>Enter end word: budge</div> <div>Search A*</div> <div>Search UCS</div> <div>Search GBFS</div> </div> <table> <tr> <th>Step</th><th>Word</th></tr> <tr><td>1</td><td>SHELF</td></tr> <tr><td>2</td><td>SHELL</td></tr> <tr><td>3</td><td>SHEOL</td></tr> <tr><td>4</td><td>SHOOL</td></tr> <tr><td>5</td><td>SHOOT</td></tr> <tr><td>6</td><td>SHOTT</td></tr> <tr><td>7</td><td>SHOTE</td></tr> <tr><td>8</td><td>SMOTE</td></tr> <tr><td>9</td><td>SMITE</td></tr> <tr><td>34</td><td>BUNKO</td></tr> <tr><td>35</td><td>BUNKS</td></tr> <tr><td>36</td><td>BULKS</td></tr> <tr><td>37</td><td>BULKY</td></tr> <tr><td>38</td><td>BULGY</td></tr> <tr><td>39</td><td>BULGE</td></tr> <tr><td>40</td><td>BUDGE</td></tr> <tr><td>Execution Time (ms)</td><td>17</td></tr> <tr><td>Words Checked</td><td>190</td></tr> </table> </div> | Step | Word | 1 | SHELF | 2 | SHELL | 3 | SHEOL | 4 | SHOOL | 5 | SHOOT | 6 | SHOTT | 7 | SHOTE | 8 | SMOTE | 9 | SMITE | 34 | BUNKO | 35 | BUNKS | 36 | BULKS | 37 | BULKY | 38 | BULGY | 39 | BULGE | 40 | BUDGE | Execution Time (ms) | 17 | Words Checked | 190 |
| Step                | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 1                   | SHELF   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 2                   | SHELL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 3                   | SHEOL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 4                   | SHOOL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 5                   | SHOOT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 6                   | SHOTT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 7                   | SHOTE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 8                   | SMOTE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 9                   | SMITE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 34                  | BUNKO   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 35                  | BUNKS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 36                  | BULKS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 37                  | BULKY   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 38                  | BULGY   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 39                  | BULGE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 40                  | BUDGE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| Execution Time (ms) | 17  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| Words Checked       | 190   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 11                  | <div> <div>Word Ladder Solver</div> <div> <div>Enter start word: swell</div> <div>Enter end word: basic</div> <div>Search A*</div> <div>Search UCS</div> <div>Search GBFS</div> </div> <table> <tr> <th>Step</th><th>Word</th></tr> <tr><td>1</td><td>SWELL</td></tr> <tr><td>2</td><td>SWILL</td></tr> <tr><td>3</td><td>STILL</td></tr> <tr><td>4</td><td>STILT</td></tr> <tr><td>5</td><td>STINT</td></tr> <tr><td>6</td><td>SAINT</td></tr> <tr><td>7</td><td>SAINS</td></tr> <tr><td>8</td><td>WAINS</td></tr> <tr><td>9</td><td>WAILS</td></tr> </table> </div>   | Step | Word | 1 | SWELL | 2 | SWILL | 3 | STILL | 4 | STILT | 5 | STINT | 6 | SAINT | 7 | SAINS | 8 | WAINS | 9 | WAILS |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| Step                | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 1                   | SWELL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 2                   | SWILL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 3                   | STILL   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 4                   | STILT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 5                   | STINT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 6                   | SAINT   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 7                   | SAINS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 8                   | WAINS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |
| 9                   | WAILS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |     |

|                     | <div> <div>10</div> <div>BAILS</div> </div> <div> <div>11</div> <div>BAITS</div> </div> <div> <div>12</div> <div>BASTS</div> </div> <div> <div>13</div> <div>BASIS</div> </div> <div> <div>14</div> <div>BASIC</div> </div> <div> <div>Execution Time (ms)</div> <div>15</div> </div> <div> <div>Words Checked</div> <div>32</div> </div>   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
|---------------------|---|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|----|-------|---------------------|----|---------------|----|
| 12                  | <div> <div>Word Ladder Solver</div> <div> <div>Enter start word: stack</div> <div>Enter end word: frame</div> <div> <div>Search A*</div> <div>Search UCS</div> <div>Search GBFS</div> </div> </div> <table> <tr> <th>Step</th><th>Word</th></tr> <tr><td>1</td><td>STACK</td></tr> <tr><td>2</td><td>SLACK</td></tr> <tr><td>3</td><td>FLACK</td></tr> <tr><td>4</td><td>FLANK</td></tr> <tr><td>5</td><td>FRANK</td></tr> <tr><td>6</td><td>TRANK</td></tr> <tr><td>7</td><td>TRACK</td></tr> <tr><td>8</td><td>TRACE</td></tr> <tr><td>9</td><td>GRACE</td></tr> <tr><td>10</td><td>GRATE</td></tr> <tr><td>11</td><td>PRATE</td></tr> <tr><td>12</td><td>PRATS</td></tr> <tr><td>13</td><td>FRATS</td></tr> <tr><td>14</td><td>FRAPS</td></tr> <tr><td>15</td><td>FLAPS</td></tr> <tr><td>16</td><td>FLAMS</td></tr> <tr><td>17</td><td>FLAME</td></tr> <tr><td>18</td><td>FRAME</td></tr> <tr> <td>Execution Time (ms)</td><td>16</td></tr> <tr> <td>Words Checked</td><td>72</td></tr> </table> </div> | Step | Word | 1 | STACK | 2 | SLACK | 3 | FLACK | 4 | FLANK | 5 | FRANK | 6 | TRANK | 7 | TRACK | 8 | TRACE | 9 | GRACE | 10 | GRATE | 11 | PRATE | 12 | PRATS | 13 | FRATS | 14 | FRAPS | 15 | FLAPS | 16 | FLAMS | 17 | FLAME | 18 | FRAME | Execution Time (ms) | 16 | Words Checked | 72 |
| Step                | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 1                   | STACK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 2                   | SLACK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 3                   | FLACK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 4                   | FLANK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 5                   | FRANK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 6                   | TRANK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 7                   | TRACK   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 8                   | TRACE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 9                   | GRACE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 10                  | GRATE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 11                  | PRATE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 12                  | PRATS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 13                  | FRATS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 14                  | FRAPS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 15                  | FLAPS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 16                  | FLAMS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 17                  | FLAME   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| 18                  | FRAME   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| Execution Time (ms) | 16  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |
| Words Checked       | 72  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |    |       |                     |    |               |    |

### 1.3. Pengujian Algoritma A\* Search

Tabel 3. Pengujian Algoritma A\*

| No. | Hasil Pengujian |
|-----|-----------------|
|-----|-----------------|

1

Word Ladder Solver

Enter start word: mood

Enter end word: ruin

Search A\*

Search UCS

Search GBFS

| Step | Word |
|------|------|
| 1    | MOOD |
| 2    | MOOL |
| 3    | MOIL |
| 4    | ROIL |
| 5    | RAIL |
| 6    | RAIN |
| 7    | RUIN |

Execution Time (ms)

2

Words Checked

174

```

$ java Main
[A*] Start: MOOD
[A*] End : RUIN
[A*] used memory (bytes): 94768

```

2

Word Ladder Solver

Enter start word: lover

Enter end word: fresh

Search A\*

Search UCS

Search GBFS

| Step | Word  |
|------|-------|
| 1    | LOVER |
| 2    | LOVES |
| 3    | LORES |
| 4    | BORES |
| 5    | BORTS |
| 6    | BOOTS |
| 7    | BOOTH |
| 8    | BROTH |
| 9    | FROTH |
| 10   | FROSH |
| 11   | FRESH |

Execution Time (ms)

61

Words Checked

1714

```

$ java Main
[A*] Start: LOVER
[A*] End : FRESH
[A*] used memory (bytes): 363864

```

3

Word Ladder Solver

Enter start word:

Enter end word:

| Step                | Word  |
|---------------------|-------|
| 1                   | STORY |
| 2                   | STORK |
| 3                   | STIRK |
| 4                   | STIRS |
| 5                   | STIES |
| 6                   | STIED |
| 7                   | SHIED |
| 8                   | SHRED |
| 9                   | SIREN |
| 10                  | TIRED |
| Execution Time (ms) | 11    |
| Words Checked       | 194   |

```

$ java Main
[A*] Start: STORY
[A*] End : TIRED
[A*] used memory (bytes): 62992
  
```

4

Word Ladder Solver

Enter start word:

Enter end word:

| Step                | Word          |
|---------------------|---------------|
| 1                   | NO PATH FOUND |
| Execution Time (ms) | 0             |
| Words Checked       | 43            |

```

$ java Main
[A*] Start: BRANCH
[A*] End : REMAIN
[A*] used memory (bytes): 11312
  
```

5

Word Ladder Solver

Enter start word:

Enter end word:

| Step | Word   |
|------|--------|
| 1    | TEMPLE |
| 2    | SEMPLE |
| 3    | SIMPLE |
| 4    | RIMPLE |
| 5    | RUMPLE |
| 6    | RUMBLE |
| 7    | BUMBLE |
| 8    | BURBLE |
| 9    | BURGLE |
| 10   | BURGEE |
| 11   | BARGEE |
| 12   | BARGED |
| 13   | BARKED |
| 14   | HARKED |
| 15   | HARPED |
| 16   | HAPPED |
| 17   | HAPPEN |

Execution Time (ms) 16

Words Checked 160

```

$ java Main
[A*] Start: TEMPLE
[A*] End : HAPPEN
[A*] used memory (bytes): 46640
  
```

6

Word Ladder Solver

Enter start word:

Enter end word:

| Step | Word  |
|------|-------|
| 1    | COACH |
| 2    | CLACH |
| 3    | CLACK |
| 4    | CLOCK |
| 5    | CHOCK |
| 6    | SHOCK |
| 7    | SHOOK |
| 8    | SHOOT |
| 9    | SHOUT |

Execution Time (ms) 15

Words Checked 78

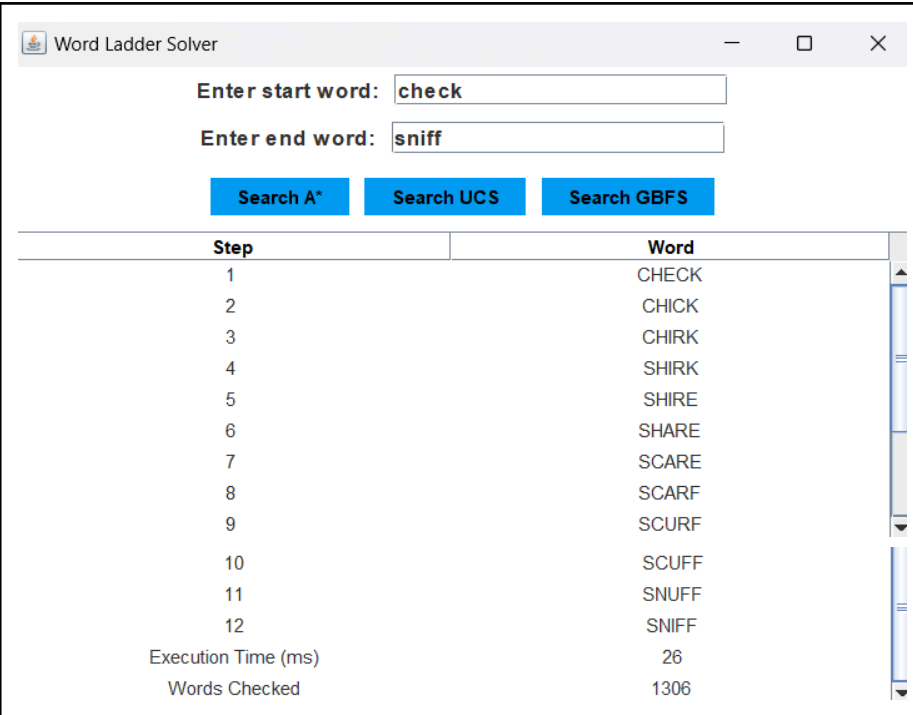
```

$ java Main
[A*] Start: COUCH
[A*] End : SHOUT
[A*] used memory (bytes): 22080
  
```



| 7                   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="sock"/></p> <p>Enter end word: <input type="text" value="bomb"/></p> <p>Search A* Search UCS Search GBFS</p> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>SOCK</td></tr> <tr><td>2</td><td>BOCK</td></tr> <tr><td>3</td><td>BOOK</td></tr> <tr><td>4</td><td>BOOB</td></tr> <tr><td>5</td><td>BOMB</td></tr> <tr><td>Execution Time (ms)</td><td>21</td></tr> <tr><td>Words Checked</td><td>17</td></tr> </tbody> </table>  | Step | Word | 1 | SOCK | 2 | BOCK | 3 | BOOK | 4 | BOOB | 5 | BOMB | Execution Time (ms) | 21   | Words Checked       | 17   |               |      |   |      |    |      |    |      |                     |    |               |     |
|---------------------|---|------|------|---|------|---|------|---|------|---|------|---|------|---------------------|------|---------------------|------|---------------|------|---|------|----|------|----|------|---------------------|----|---------------|-----|
| Step                | Word  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 1                   | SOCK  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 2                   | BOCK  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 3                   | BOOK  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 4                   | BOOB  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 5                   | BOMB  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Execution Time (ms) | 21  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Words Checked       | 17  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 8                   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="cart"/></p> <p>Enter end word: <input type="text" value="shed"/></p> <p>Search A* Search UCS Search GBFS</p> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>CART</td></tr> <tr><td>2</td><td>CARD</td></tr> <tr><td>3</td><td>SARD</td></tr> <tr><td>4</td><td>SURD</td></tr> <tr><td>5</td><td>SUED</td></tr> <tr><td>6</td><td>SHED</td></tr> <tr><td>Execution Time (ms)</td><td>16</td></tr> <tr><td>Words Checked</td><td>30</td></tr> </tbody> </table>   | Step | Word | 1 | CART | 2 | CARD | 3 | SARD | 4 | SURD | 5 | SUED | 6                   | SHED | Execution Time (ms) | 16   | Words Checked | 30   |   |      |    |      |    |      |                     |    |               |     |
| Step                | Word  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 1                   | CART  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 2                   | CARD  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 3                   | SARD  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 4                   | SURD  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 5                   | SUED  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 6                   | SHED  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Execution Time (ms) | 16  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Words Checked       | 30  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 9                   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="inch"/></p> <p>Enter end word: <input type="text" value="sofa"/></p> <p>Search A* Search UCS Search GBFS</p> <table> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>INCH</td></tr> <tr><td>2</td><td>ITCH</td></tr> <tr><td>3</td><td>ETCH</td></tr> <tr><td>4</td><td>EACH</td></tr> <tr><td>5</td><td>MACH</td></tr> <tr><td>6</td><td>MACS</td></tr> <tr><td>7</td><td>MOCS</td></tr> <tr><td>8</td><td>MODS</td></tr> <tr><td>9</td><td>SODS</td></tr> <tr><td>10</td><td>SODA</td></tr> <tr><td>11</td><td>SOFA</td></tr> <tr><td>Execution Time (ms)</td><td>17</td></tr> <tr><td>Words Checked</td><td>201</td></tr> </tbody> </table> | Step | Word | 1 | INCH | 2 | ITCH | 3 | ETCH | 4 | EACH | 5 | MACH | 6                   | MACS | 7                   | MOCS | 8             | MODS | 9 | SODS | 10 | SODA | 11 | SOFA | Execution Time (ms) | 17 | Words Checked | 201 |
| Step                | Word  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 1                   | INCH  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 2                   | ITCH  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 3                   | ETCH  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 4                   | EACH  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 5                   | MACH  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 6                   | MACS  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 7                   | MOCS  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 8                   | MODS  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 9                   | SODS  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 10                  | SODA  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| 11                  | SOFA  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Execution Time (ms) | 17  |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |
| Words Checked       | 201   |      |      |   |      |   |      |   |      |   |      |   |      |                     |      |                     |      |               |      |   |      |    |      |    |      |                     |    |               |     |

| 10   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="exile"/></p> <p>Enter end word: <input type="text" value="hover"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>EXILE</td></tr> <tr><td>2</td><td>AXILE</td></tr> <tr><td>3</td><td>AXILS</td></tr> <tr><td>4</td><td>ARILS</td></tr> <tr><td>5</td><td>ARIAS</td></tr> <tr><td>6</td><td>ARRAS</td></tr> <tr><td>7</td><td>AURAS</td></tr> <tr><td>8</td><td>AURES</td></tr> <tr><td>9</td><td>LURES</td></tr> <tr><td>10</td><td>LORES</td></tr> <tr><td>11</td><td>LOVES</td></tr> <tr><td>12</td><td>LOVER</td></tr> <tr><td>13</td><td>HOVER</td></tr> </tbody> </table> <p>Execution Time (ms) 21</p> <p>Words Checked 173</p> | Step | Word | 1 | EXILE | 2 | AXILE | 3 | AXILS | 4 | ARILS | 5 | ARIAS | 6 | ARRAS | 7 | AURAS | 8 | AURES | 9 | LURES | 10 | LORES | 11 | LOVES | 12 | LOVER | 13 | HOVER |
|------|---|------|------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|---|-------|----|-------|----|-------|----|-------|----|-------|
| Step | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 1    | EXILE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 2    | AXILE   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 3    | AXILS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 4    | ARILS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 5    | ARIAS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 6    | ARRAS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 7    | AURAS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 8    | AURES   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 9    | LURES   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 10   | LORES   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 11   | LOVES   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 12   | LOVER   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 13   | HOVER   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 11   | <p>Word Ladder Solver</p> <p>Enter start word: <input type="text" value="swing"/></p> <p>Enter end word: <input type="text" value="yearn"/></p> <p>Search A* Search UCS Search GBFS</p> <table border="1"> <thead> <tr> <th>Step</th><th>Word</th></tr> </thead> <tbody> <tr><td>1</td><td>SWING</td></tr> <tr><td>2</td><td>SWANG</td></tr> <tr><td>3</td><td>SWANS</td></tr> <tr><td>4</td><td>SPANS</td></tr> <tr><td>5</td><td>SPARS</td></tr> <tr><td>6</td><td>SEARS</td></tr> <tr><td>7</td><td>YEARS</td></tr> <tr><td>8</td><td>YEARN</td></tr> </tbody> </table> <p>Execution Time (ms) 15</p> <p>Words Checked 68</p>  | Step | Word | 1 | SWING | 2 | SWANG | 3 | SWANS | 4 | SPANS | 5 | SPARS | 6 | SEARS | 7 | YEARS | 8 | YEARN |   |       |    |       |    |       |    |       |    |       |
| Step | Word  |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 1    | SWING   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 2    | SWANG   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 3    | SWANS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 4    | SPANS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 5    | SPARS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 6    | SEARS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 7    | YEARS   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |
| 8    | YEARN   |      |      |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |   |       |    |       |    |       |    |       |    |       |

|    |  |
|----|--|
| 12 |  |
|----|--|

#### 1.4. Hasil Pengujian

Tabel 3. Keterangan Hasil Pengujian

| Warna | Keterangan    |
|-------|---------------|
|       | Paling baik   |
|       | Kedua terbaik |
|       | Paling buruk  |

Tabel 4. Hasil Pengujian Kasus Sama

| Parameter                           | Algoritma | TC.1 | TC.2 | TC.3 | TC.4 | TC.5 | TC.6 |
|-------------------------------------|-----------|------|------|------|------|------|------|
| Jumlah kata pada jalur yang didapat | UCS       | 7    | 11   | 10   | -    | 17   | 9    |
|                                     | GBFS      | 7    | 19   | 20   | -    | 28   | 13   |
|                                     | A*        | 7    | 11   | 10   | -    | 17   | 9    |
| Jumlah kata yang dicek              | UCS       | 3344 | 6726 | 3203 | 43   | 2059 | 2868 |
|                                     | GBFS      | 14   | 65   | 36   | 43   | 77   | 24   |
|                                     | A*        | 174  | 1714 | 194  | 43   | 160  | 78   |

|                               |      |        |        |        |       |       |        |
|-------------------------------|------|--------|--------|--------|-------|-------|--------|
| Waktu eksekusi (ms)           | UCS  | 55     | 60     | 21     | 2     | 18    | 20     |
|                               | GBFS | 20     | 24     | 26     | 23    | 23    | 14     |
|                               | A*   | 2      | 61     | 11     | 0     | 16    | 15     |
| Memori yang digunakan (bytes) | UCS  | 363472 | 682896 | 498376 | 11352 | 99544 | 381624 |
|                               | GBFS | 17860  | 60072  | 21104  | 11064 | 29656 | 16040  |
|                               | A*   | 94768  | 363864 | 62992  | 11312 | 46640 | 22080  |

## 4.2. Analisis

Program ini mengimplementasikan tiga algoritma pencarian. Berikut penjelasan langkah masing - masing algoritma.

Berikut langkah - langkah algoritma UCS pada program ini:

1. Masukkan kata awal ke dalam antrian prioritas.
2. Ambil kata paling depan dari antrian.
3. Cek apakah kata merupakan kata target. Jika ya, maka pencarian selesai. Jika tidak, maka cari seluruh kata yang dapat dihasilkan dari kata awal dengan cara mengganti salah satu huruf. Hanya mengambil kata - kata yang valid (ada dalam kamus bahasa Inggris).
4. Kata - kata yang dihasilkan tersebut memiliki jarak dari kata awal ( $g(n)$ ) yakni jarak kata yang dicek ke kata awal ditambah 1, karena ada penggantian satu huruf.
5. Jika kata yang dihasilkan belum pernah dicek atau kata tersebut memiliki  $g(n)$  lebih kecil dari  $g(n)$  kata yang sama pada pengecekan sebelumnya, maka masukkan kata tersebut ke dalam antrian sesuai prioritas, dimana  $g(n)$  terkecil akan ditaruh paling depan.
6. Ulangi langkah dari langkah kedua hingga seluruh kemungkinan telah dicek atau kata target ditemukan.

Berikut langkah - langkah algoritma Greedy Best First Search pada program ini:

1. Masukkan kata awal ke dalam antrian prioritas.
2. Ambil kata paling depan dari antrian.
3. Cek apakah kata merupakan kata target. Jika ya, maka pencarian selesai. Jika tidak, maka cari seluruh kata yang dapat dihasilkan dari kata awal dengan cara

mengganti salah satu huruf. Hanya mengambil kata - kata yang valid (ada dalam kamus bahasa Inggris).

4. Kata - kata yang dihasilkan tersebut memiliki nilai  $f(n)$  masing - masing, nilai  $f(n)$  diambil dari nilai heuristik ( $h(n)$ ) kata tersebut. Nilai heuristik dihitung dengan menghitung berapa banyak huruf yang tidak cocok terhadap kata target. Contoh jika kata target adalah ABC, maka kata ADF memiliki nilai heuristik 2 karena D dan F tidak cocok.
7. Jika kata yang dihasilkan belum pernah dicek, maka masukkan kata tersebut ke dalam antrian sesuai prioritas, dimana kata dengan nilai  $f(n)$  terkecil akan ditaruh paling depan.
8. Ulangi langkah dari langkah kedua hingga seluruh kemungkinan telah dicek atau kata target ditemukan.

Berikut langkah - langkah algoritma A\* pada program ini:

1. Masukkan kata awal ke dalam antrian prioritas.
2. Ambil kata paling depan dari antrian.
3. Cek apakah kata merupakan kata target. Jika ya, maka pencarian selesai. Jika tidak, maka cari seluruh kata yang dapat dihasilkan dari kata awal dengan cara mengganti salah satu huruf. Hanya mengambil kata - kata yang valid (ada dalam kamus bahasa Inggris).
4. Kata - kata yang dihasilkan tersebut memiliki nilai  $f(n)$  yang dihitung dari nilai heuristik ( $h(n)$ ) ditambah dengan jarak kata tersebut dari kata awal yang didapat dari jarak kata yang dicek ditambah 1 ( $g(n)$ ). Nilai heuristik dihitung dengan menghitung berapa banyak huruf yang tidak cocok terhadap kata target.
5. Jika kata yang dihasilkan belum pernah dicek atau  $g(n)$  yang dihasilkan lebih kecil dari  $g(n)$  kata tersebut pada pengecekan sebelumnya, maka masukkan kata tersebut ke dalam antrian sesuai prioritas, dimana kata dengan  $f(n)$  akan ditaruh paling depan.
6. Ulangi langkah dari langkah kedua hingga seluruh kemungkinan telah dicek atau kata target ditemukan.

Berdasarkan langkah - langkah tersebut, fungsi  $g(n)$  merupakan jarak dari kata awal ke kata yang sedang dicek. Yang berarti banyak perubahan huruf yang sudah terjadi dari kata awal hingga ke kata yang sedang dicek. Dan Fungsi evaluasi  $f(n)$  yang digunakan pada algoritma Greedy Best First search merupakan nilai fungsi heuristik  $h(n)$  yang didapat dari menghitung banyaknya huruf yang tidak cocok saat perbandingan kata yang sedang dicek dengan kata target. Sedangkan  $f(n)$  yang digunakan pada algoritma A\* merupakan jumlah dari nilai heuristik ditambah dengan jarak dari kata awal ke kata yang dimaksud  $h(n) + g(n)$ .

Heuristik yang menghitung jumlah huruf yang tidak cocok antara dua kata dapat dianggap sebagai heuristik yang admissible dalam konteks algoritma A\*, karena ia selalu mengunderestimate atau sama dengan jumlah langkah minimal yang sebenarnya diperlukan untuk mengubah satu kata menjadi kata lain melalui serangkaian kata yang valid dalam bahasa Inggris. Untuk kasus kata BOAT ke READ, heuristik ini memberikan nilai 3, yang berarti ada tiga huruf yang harus diubah. Namun, nilai ini tidak mempertimbangkan apakah perubahan setiap huruf menghasilkan kata yang valid dalam bahasa Inggris. Misalnya, mengubah B menjadi R di langkah pertama mungkin tidak langsung menghasilkan kata yang valid, sehingga perlu langkah tambahan untuk mencapai kata yang valid sebelum mencapai READ. Oleh karena itu, perubahan huruf dari B ke R, O ke E, dan T ke D tidak selalu langsung menghasilkan kata-kata yang valid, sehingga langkah-langkah nyata yang dibutuhkan bisa lebih banyak. Dalam kasus ini, heuristik tersebut mengunderestimate jumlah langkah yang sebenarnya diperlukan karena menganggap setiap perubahan huruf bisa langsung dilakukan tanpa mempertimbangkan validitas kata. Hal ini membuatnya admissible karena heuristik yang admissible tidak boleh melebihi biaya sebenarnya untuk mencapai tujuan, dan heuristik ini memenuhi kriteria tersebut dengan asumsi bahwa setiap perubahan huruf adalah langkah yang valid dan mungkin, meskipun dalam praktiknya bisa memerlukan lebih banyak langkah.

Dalam kasus word ladder, tiap perubahan satu huruf memang dianggap memiliki biaya satu, yang berarti setiap transisi dari satu kata ke kata lain, jika melibatkan perubahan satu huruf, memiliki biaya yang sama. Dalam konteks ini, algoritma Uniform Cost Search (UCS) akan berperilaku mirip dengan Breadth-First Search (BFS) karena kedua algoritma tersebut akan menjelajahi semua kemungkinan perubahan satu huruf pada level yang sama sebelum beralih ke level berikutnya.

Dengan kata BED sebagai contoh, kata-kata yang dihasilkan dari perubahan satu huruf pada posisi yang berbeda (seperti (A-Z)ED, B(A-Z)D, BE(A-Z)) semuanya akan memiliki biaya satu dari BED. Karena biaya transisi yang seragam (setiap transisi sama dengan satu), UCS, yang biasanya memprioritaskan node dengan total biaya terendah yang telah dikumulatifkan, akan berperilaku seperti BFS yang memprioritaskan penjelajahan berdasarkan kedalaman level, keduanya akan menjelajahi semua node (kata) pada kedalaman tertentu sebelum beralih ke kedalaman berikutnya. Oleh karena itu, dalam konteks word ladder dengan biaya perubahan huruf yang seragam, UCS tidak memiliki keuntungan dalam hal efisiensi pencarian dibandingkan dengan BFS, dan kedua metode tersebut akan menghasilkan path yang sama dalam hal urutan dan biaya, asalkan tidak ada batasan atau preferensi tambahan yang diterapkan.

Dalam konteks algoritma A\* yang menggunakan heuristik yang admissible, A\* memang dapat lebih efisien dibandingkan dengan Uniform Cost Search (UCS) dalam kasus word ladder, terutama karena A\* menggabungkan biaya yang telah dikeluarkan (cost from start) dan biaya yang diperkirakan untuk mencapai tujuan (heuristic cost to goal) untuk menentukan node mana yang harus dieksplorasi selanjutnya. UCS, yang berperilaku seperti Breadth-First Search (BFS) dalam kasus biaya transisi yang seragam, tidak memperhitungkan jarak perkiraan dari node saat ini ke node tujuan. Sebaliknya, UCS hanya memperhitungkan biaya yang telah dikeluarkan untuk mencapai node saat ini, menjelajahi semua kemungkinan node pada kedalaman tertentu sebelum beralih ke kedalaman berikutnya. Dalam hal ini, UCS dapat dikatakan bekerja seperti brute-force karena menjelajahi semua kemungkinan tanpa preferensi atau prediksi tentang node mana yang lebih mungkin mencapai tujuan lebih cepat. Di sisi lain, A\* menggunakan heuristik untuk memperkirakan biaya dari node saat ini ke node tujuan, yang membantu algoritma untuk memprioritaskan node yang lebih mungkin mendekatkan kepada solusi, berdasarkan estimasi biaya total yang lebih rendah. Hal ini memungkinkan A\* untuk secara efisien memfokuskan pencarian pada path yang lebih menjanjikan, mengurangi jumlah node yang perlu dieksplorasi dibandingkan dengan UCS. Karena heuristiknya admissible, yaitu selalu mengunderestimate atau setara dengan biaya sebenarnya dari node saat ini ke node tujuan, A\* dijamin akan menemukan solusi yang optimal. A\* tidak akan menjelajahi semua kata pada kedalaman tertentu secara tidak perlu,

melainkan memprioritaskan kata-kata yang lebih dekat dengan tujuan berdasarkan perhitungan heuristik, sehingga membuat pencarian lebih cepat dan lebih efisien daripada UCS dalam kasus ini.

Dalam word ladder, GBFS bisa cepat mencapai tujuan karena langsung mengarah ke kata-kata yang secara visual atau superficial lebih mirip dengan kata tujuan, namun hal ini seringkali mengabaikan jalan yang mungkin lebih panjang secara jumlah kata namun lebih pendek dalam hal jumlah perubahan huruf yang diperlukan. Ini artinya, meskipun GBFS mungkin lebih cepat dalam menemukan sebuah solusi dibandingkan dengan UCS atau A\*, solusi yang dihasilkan tidak selalu optimal. Path yang dihasilkan oleh GBFS tidak mencerminkan jumlah huruf yang telah diganti secara akurat, karena algoritma ini tidak mempertimbangkan biaya total dari perubahan yang telah dilakukan, sehingga dapat melewati jalur yang secara keseluruhan memiliki biaya lebih rendah namun memerlukan lebih banyak langkah yang terlihat tidak menjanjikan pada awalnya.

Berdasarkan data pengujian yang disajikan dalam Tabel 4 pada subbab 4.1 Pengujian, algoritma Uniform Cost Search (UCS) memerlukan pengecekan yang lebih banyak, sehingga membutuhkan lebih banyak memori. Hal ini disebabkan karena tiap algoritma menyimpan setiap kata yang dicek untuk menghindari pemeriksaan ulang terhadap kata yang sama. Sementara itu, algoritma A\* dan Greedy Best First Search (GBFS) memerlukan memori yang lebih sedikit karena jumlah pengecekan yang mereka lakukan lebih sedikit.

|      |  |
|------|--|
| UCS  | <pre> └─ \$ java Main [UCS] Start: LOVER [UCS] End  : FRESH [UCS] used memory (bytes): 439232 </pre>   |
| GBFS | <pre> └─ \$ java Main [GBFS] Start: LOVER [GBFS] End  : FRESH [GBFS] used memory (bytes): 71800 </pre> |
| A*   | <pre> └─ \$ java Main [A*] Start: LOVER [A*] End  : FRESH [A*] used memory (bytes): 363864 </pre>      |



Contoh kasus TC.2 dari LOVER ke FRESH. Test dijalankan dengan menjalankan program berulang sehingga tidak ada cache yang dipakai.

Dampak ini juga terlihat pada waktu eksekusi. UCS, dengan jumlah pengecekan yang paling banyak, memerlukan waktu eksekusi yang lebih lama dibandingkan algoritma lain. A\* umumnya lebih cepat dari UCS karena mengkombinasikan biaya yang telah dikeluarkan dengan estimasi heuristik ke tujuan untuk memprioritaskan node yang dijelajahi, sehingga efisiensi pencariannya lebih tinggi. GBFS, yang hanya memfokuskan pada heuristik tanpa mempertimbangkan biaya yang dikeluarkan, memiliki waktu eksekusi yang sangat cepat, namun dalam beberapa kasus bisa lebih lambat dari A\*. Mengenai optimalitas path, UCS dan A\* selalu menghasilkan path yang optimal, yang menjamin jalur terpendek berdasarkan biaya yang dikeluarkan untuk mencapai setiap node. Ini berbeda dengan GBFS yang tidak mempertimbangkan biaya keseluruhan, sehingga seringkali menghasilkan path yang tidak optimal.

#### **4.3. Penjelasan Bonus**

Bonus yang dikerjakan adalah GUI. GUI yang dibuat menggunakan Swing dan bahasa pemrograman Java. Antarmuka dari program ini dapat menerima masukan start word dan end word. Yang perlu melewati validasi. Antarmuka ini juga menyajikan tiga tombol pencarian untuk algoritma yang berbeda. Hasil dari pencarian akan ditampilkan pada antarmuka dengan format tabel.

## **BAB V**

### **PENUTUP**

#### **5.1. Kesimpulan**

Dari pengujian yang dilakukan terhadap tiga algoritma pencarian dalam konteks word ladder, Uniform Cost Search (UCS), Greedy Best First Search (GBFS), dan A\* dapat disimpulkan beberapa hal. Pertama, UCS, meskipun menghasilkan path yang optimal, memerlukan waktu eksekusi yang lebih lama dan memori yang lebih besar dibandingkan dengan dua algoritma lainnya. Hal ini disebabkan oleh pendekatannya yang exhaustive dalam menjelajahi semua kemungkinan transisi pada setiap level. Kedua, GBFS menawarkan kecepatan eksekusi yang sangat cepat karena fokusnya yang kuat pada heuristik, namun seringkali mengorbankan optimalitas path karena tidak mempertimbangkan biaya total yang dikeluarkan. Terakhir, algoritma A\* menunjukkan keseimbangan terbaik antara kecepatan dan efisiensi, dengan menggabungkan biaya yang telah dikeluarkan dan estimasi heuristik ke tujuan untuk memprioritaskan pencarian, yang seringkali menghasilkan solusi optimal dengan penggunaan memori yang lebih efisien dan waktu eksekusi yang lebih singkat dibandingkan dengan UCS.

Dalam konteks aplikasi word ladder yang ditest, A\* secara konsisten mengungguli UCS dalam hal kecepatan tanpa mengorbankan optimalitas path yang ditemukan, menjadikannya pilihan yang sangat baik untuk aplikasi yang membutuhkan kombinasi antara efisiensi waktu dan penggunaan sumber daya. Sementara itu, GBFS mungkin cocok untuk skenario di mana kecepatan adalah prioritas utama dan keakuratan atau minimalisasi langkah tidak terlalu kritis. Implementasi UCS, meskipun kurang efisien dalam hal sumber daya dan waktu, tetap relevan dalam kasus di mana kepastian menemukan jalur terpendek adalah mutlak diperlukan. Oleh karena itu, pemilihan algoritma dapat disesuaikan berdasarkan kebutuhan spesifik penggunaan, dengan pertimbangan terhadap trade-off antara kecepatan, memori, dan keakuratan hasil yang dihasilkan.

#### **5.2. Saran**

Untuk meningkatkan efektivitas dan efisiensi sistem pencarian word ladder, disarankan untuk mengembangkan heuristik yang lebih akurat untuk algoritma GBFS

dan A\*, mengoptimasi penggunaan memori pada UCS, dan menerapkan caching untuk mengurangi redundansi dalam perhitungan. Penerapan paralelisme dapat mempercepat proses dan meningkatkan penggunaan sumber daya hardware yang tersedia. Evaluasi dan tes berkelanjutan dengan dataset yang bervariasi juga penting untuk memastikan adaptabilitas dan relevansi sistem dalam menghadapi berbagai skenario penggunaan.

## DAFTAR REFERENSI

- Munir, Rinaldi. Penentuan rute (Route/Path Planning) - Bagian 1. Diakses 4 Mei 2024.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf>
- Munir, Rinaldi. Penentuan rute (Route/Path Planning) - Bagian 2. Diakses 4 Mei 2024.  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf>
- Trivusi. Apa itu Uniform-Cost Search? Pengertian dan Cara Kerjanya. Diakses 4 Mei 2024.  
<https://www.trivusi.web.id/2022/10/apa-itu-algoritma-uniform-cost-search.html>

## LAMPIRAN

Link Github: [https://github.com/juanaw6/Tucil3\\_13522073](https://github.com/juanaw6/Tucil3_13522073)

Tabel Kelayakan Program

| 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. <b>[Bonus]:</b> Program memiliki tampilan GUI  | ✓  |       |