**Applied Artificial Intelligence**

**Assignment - 1**

**Course Instructor**

Dr. Atif Jilani

**Submitted by**

Usman Ali……………………………………………………………..(22I-2725)

Faizan Rasheed………………………………….…………………….(22I-2734)

**Section**

SE-E

**Date**

**Saturday, March 1st, 2025**

**Spring 2025**



**Department of Software Engineering**

FAST – National University of Computer & Emerging Sciences

Islamabad Campus

Table of Contents

[1. Introduction 1](#_Toc191737842)

[2. Game Overview 1](#_Toc191737843)

[3. Implementation Details 1](#_Toc191737844)

[3.1. Search Algorithms 1](#_Toc191737845)

[1. A\* Search Algorithm: 1](#_Toc191737846)

[2. Greedy Best-First Search (GBFS): 1](#_Toc191737847)

[3. Uniform Cost Search (UCS): 1](#_Toc191737848)

[3.2. Graph Representation 1](#_Toc191737849)

[3.3. Dictionary Integration 2](#_Toc191737850)

[3.4. Game Modes 2](#_Toc191737851)

[1. Beginner Mode: 2](#_Toc191737852)

[2. Advanced Mode: 2](#_Toc191737853)

[3. Challenge Mode: 2](#_Toc191737854)

[4. Custom Mode: 2](#_Toc191737855)

[4. Game Features 2](#_Toc191737856)

[4.1. AI-Powered Hint System 2](#_Toc191737857)

[4.2. Graph Visualization 3](#_Toc191737858)

[4.3. Scoring System 3](#_Toc191737859)

[4.4. Move Limit 3](#_Toc191737860)

[5. Challenges Faced 3](#_Toc191737861)

[5.1. Dictionary Integration 3](#_Toc191737862)

[5.2. Graph Visualization 3](#_Toc191737863)

[5.3. Algorithm Optimization 3](#_Toc191737864)

[6. Code Structure 3](#_Toc191737865)

[6.1. Key Functions 3](#_Toc191737866)

[1. load\_words (): 3](#_Toc191737867)

[2. get\_neighbor\_words (word): 3](#_Toc191737868)

[3. astar (start, end), gbfs (start, end), ucs (start, end): 4](#_Toc191737869)

[4. visualize\_graph (graph, path, title, highlight\_user\_path): 4](#_Toc191737870)

[5. game\_mode (): 4](#_Toc191737871)

[6. select\_random\_words (word\_list): 4](#_Toc191737872)

[7. predefined\_challenge (): 4](#_Toc191737873)

[8. custom\_word\_ladder (): 4](#_Toc191737874)

[6.2. Data Structures 4](#_Toc191737875)

[7. Conclusion 4](#_Toc191737876)

# 1. Introduction

The **Word Ladder Adventure Game** is a fun puzzle game where players change one word into another by replacing just one letter at a time. Every step must be a real word from a set dictionary. To help players, the game includes **AI-powered hints** that suggest the shortest way to reach the final word using smart search techniques. This report explains how the game was built, how the search algorithms work, the challenges faced, and how everything comes together to make the game enjoyable and functional.

# 2. Game Overview

The game allows players to:

* **Multiple Modes:** Players can choose from predefined challenges based on difficulty level or create custom word ladders.
* **Transform Words:** Players can manually change one letter at a time to transform a start word into a target word.
* **AI Assistance:** Players can request hints from an AI assistant that uses search algorithms to suggest the next best move.
* **Graph Visualization:** The game visualizes the word ladder as a graph, with words as nodes and valid transformations as edges

# 3. Implementation Details

## 3.1. Search Algorithms

The game implements **three search algorithms** to find the shortest path between words:

### 1. A\* Search Algorithm:

* It uses both the actual cost so far (**g(n)**) and an estimated cost to the goal (**h(n)**) to find the best path.
* The (Heuristic Function) estimated cost (**h(n)**) predicts how many steps are needed to reach the target word.
* A\* is efficient and guarantees the shortest path when the heuristic is admissible.

### 2. Greedy Best-First Search (GBFS):

* It focuses on the heuristic function (h(n)) to prioritize nodes closest to the target word.
* **Performance:** GBFS is faster but does not guarantee the shortest path.

### 3. Uniform Cost Search (UCS):

* Uses the cost function (g(n)) to prioritize paths with the lowest cumulative cost.
* **Performance:** UCS guarantees the shortest path but can be slower due to its exhaustive search.

## 3.2. Graph Representation

* Words are represented as **nodes** in a graph, and valid transformations are represented as **edges**.
* The game uses the **NetworkX** and **Matplotlib** library to create and visualize the graph.
* Players can visualize the graph for each algorithm, highlighting the shortest path and possible transformations.

## 3.3. Dictionary Integration

* The game uses a dictionary of valid English words sourced from a **GitHub repository**.
* The dictionary is loaded into the game as a **set** for efficient word validation.
* Words are validated against this dictionary to ensure all transformations are valid.

## 3.4. Game Modes

The game offers multiple modes to cater to different player preferences:

### Beginner Mode:

* Simple word ladders with short transformation sequences (e.g., "cat" to "dog").
* Move limit: 10.

### Advanced Mode:

* Longer and more complex word transformations (e.g., "stone" to "money").
* Move limit: 15.

### Challenge Mode:

* Introduces obstacles such as **banned words** and **restricted letters**.
* Move limit: 20.

### Custom Mode:

* Players can input their own start and end words, provided a valid transformation path exists.
* Move limit: 20.

# 4. Game Features

## 4.1. AI-Powered Hint System

* Players can request hints from the AI assistant, which uses one of the three search algorithms to suggest the next best move.
* The AI assistant provides the shortest transformation path based on the selected algorithm.

## 4.2. Graph Visualization

* The game visualizes the word ladder as a graph, with words as nodes and transformations as edges.
* Players can view the graph for each algorithm, highlighting the shortest path and possible transformations.

## 4.3. Scoring System

* Players start with a score of **1000 points**.
* Each move reduces the score by **10 points**.
* The goal is to complete the transformation in as few moves as possible to maximize the score.

## 4.4. Move Limit

* Each game mode has a move limit to increase the challenge and strategic depth.
* If the player exceeds the move limit, the game ends, and the player loses.

# 5. Challenges Faced

## 5.1. Dictionary Integration

* Firstly, loading the dictionary from an external source was slow.
* To fix this issue, the dictionary was saved locally after the first use, making future loads much faster.

## 5.2. Graph Visualization

* Displaying large word transformation graphs with many connections was difficult.
* The **NetworkX** library and **kamada\_kawai\_layout** algorithm(to improve layout) was used to generate and display the graph, with adjustments to keep it clear and readable.

## 5.3. Algorithm Optimization

* Making the search algorithms (**A**\* and **UCS**) fast and efficient required careful fine-tuning.
* The heuristic function along with priority queues was designed to accurately estimate the number of steps needed, ensuring the best performance .

# 6. Code Structure

## 6.1. Key Functions

### load\_words ():

Loads dictionary of valid English words.

### get\_neighbor\_words (word):

Returns all neighbors of a word by replacing one letter at a time

### astar (start, end), gbfs (start, end), ucs (start, end):

Implement the A\*, GBFS, and UCS algorithms.

### visualize\_graph (graph, path, title, highlight\_user\_path):

Visualizes the word ladder graph using NetworkX and matplotlib library.

### game\_mode ():

Handles the main gameplay loop, including word transformation, AI hints, and graph visualization

### select\_random\_words (word\_list):

It randomly selects two words from pre-defined set of words as start and end words and return them.

### predefined\_challenge ():

Allow players to select a mode from beginner, advance and challenging.

### custom\_word\_ladder ():

Allow a player to enter start and word in custom mode.

## 6.2. Data Structures

* **english\_words**: A set of valid English words.
* **Node class**: Represents a word in the graph, with attributes for state, parent, and costs (g(n), h(n), f(n)).

# 7. Conclusion

The Word Ladder Adventure Game successfully meets all the key requirements of the project. It provides a fun and educational experience where players can solve word ladders on their own or with AI-powered hints.

The game uses three search algorithms (A\*, GBFS, and UCS) to show different AI search strategies. Its graph-based visualization and various difficulty levels make gameplay more engaging.