# Assignment 01: Applied AI

**Abdul Rafay i22-8762**

**Sumeed Jawad Kanwar i22-2652**

**Word Ladder Adventure Game Documentation**

**Project Overview**

The Word Ladder Adventure Game is a web-based implementation of Lewis Carroll's classic word game, where players transform one word into another by changing a single letter at a time, with each intermediate step being a valid word. This implementation features multiple difficulty levels, custom word challenges, and AI-powered hint systems using various search algorithms.

**Core Features**

**Game Modes**

* **Easy**: 3-4 letter words with no move limits
* **Medium**: 5-6 letter words with 10-12 move limits
* **Hard**: 7+ letter words with 8-12 move limits
* **Challenge**: 4-7 letter words with 6-10 move limits and banned words

**Player Experience**

* Retro-inspired pixel art interface with a cyberpunk aesthetic
* Real-time move tracking and scoring
* Timer to track completion speed
* leaderboard system

**AI Assistance**

The game implements four distinct search algorithms for providing hints:

1. **Breadth-First Search (BFS)**: Finds the shortest path by exploring all neighbors at the current depth before moving to the next level
2. **Uniform Cost Search (UCS)**: Considers path costs to find the optimal solution
3. **A\* Search:** Combines path cost with heuristic estimation for efficient pathfinding
4. **Greedy Best-First Search**: Uses heuristic estimation to select the most promising next word

**Technical Implementation**

**Backend (Python/Flask)**

* Word validation against a CSV dictionary
* Path-finding algorithms implementation
* Score calculation based on move efficiency and difficulty
* Graph visualization generation using NetworkX and Matplotlib

**Frontend (HTML/CSS/JavaScript)**

* Responsive design with retro gaming aesthetics
* Interactive gameplay with real-time feedback
* Dynamic visual elements for user engagement
* Tabbed interface for game, leaderboard, and information

**Data Structures**

* Words represented as nodes in a graph
* Valid transformations (one-letter changes) as edges
* Priority queues for optimal path finding
* Sets for efficient word validation

**Algorithm Details**

**Word Neighbor Generation**

def get\_neighbors(word, mode="normal"):

    """Find all valid words differing by one letter."""

    neighbors = []

    for i in range(len(word)):

        for letter in "abcdefghijklmnopqrstuvwxyz":

            new\_word = word[:i] + letter + word[i+1:]

            if new\_word in valid\_words and new\_word != word:

                # In challenge mode, check if the word is banned

                if mode == "challenge" and new\_word in banned\_words:

                    continue

                neighbors.append(new\_word)

    return neighbors

**Heuristic Function**

For informed search algorithms (A\* and Greedy):

def heuristic(word):

        return sum(1 for a, b in zip(word, end\_word) if a != b)

This calculates Hamming distance - the number of positions where letters differ.

**Scoring System**

Scores are calculated based on:

* How close the player's solution is to the optimal path
* The difficulty level chosen
* Perfect scores (1000 base points) for matching the optimal path length
* Penalties for additional moves beyond the optimal solution

**Future Enhancements**

1. Persistent user accounts and statistics tracking
2. Additional game modes with time-based challenges
3. Expanded dictionary and multi-language support
4. Mobile application version
5. Educational mode with algorithm visualization and explanation
6. Daily challenges with global leaderboards

**Conclusion**

The Word Ladder Adventure Game successfully implements a classic word puzzle with modern AI techniques, providing both entertainment and educational value. The integration of multiple search algorithms offers insight into different approaches to problem-solving, while the game mechanics and retro aesthetic create an engaging user experience.