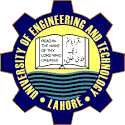
**DSA Gaming Hub**



**Session 2023 – 2027**

**Submitted by:**

ESA IRFAN 2023-CS-104

Waleed Sarfaraz 2023-CS-102

Muhammad Owais 2023-CS-131

**Supervised by:**

Mr. Nazeef ul Haq

Mr. Waseem

**Course:**

CSC-200 Data Structures and Algorithms

**Department of Computer Science**

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[**1.1 Overview** 3](#_Toc183364705)

[**1.2 Objectives** 3](#_Toc183364706)

[**1.3 Games** 3](#_Toc183364707)

[1.3.1 Stack related 3](#_Toc183364708)

[1.3.1.1 Tower of Hanoi 3](#_Toc183364709)

[1.3.1.2. Fill Up the Bottle 3](#_Toc183364710)

[1.3.2 Queue related 4](#_Toc183364711)

[Dine Dash 4](#_Toc183364712)

[1.3.3 Linked List related 4](#_Toc183364713)

[1.3.3.1 Snake Evolution 4](#_Toc183364714)

[1.3.3.2 Twist and Climb 5](#_Toc183364715)

[1.3.4 Graph related 6](#_Toc183364716)

[1.3.4.1 One Way Out 6](#_Toc183364717)

[1.3.4.2 Dog’s Chase 6](#_Toc183364718)

[1.3.4.3 Trail of Secret 6](#_Toc183364719)

[1.3.4.4 Railroad Tycoon Challenge 7](#_Toc183364720)

[1.3.5 Array related 7](#_Toc183364721)

[1.3.5.1 Tic Tac Toe 7](#_Toc183364722)

[1.3.5.2 Memory Match 8](#_Toc183364723)

[1.3.5.3 Word Search 8](#_Toc183364724)

[1.3.6 Trees Related 9](#_Toc183364725)

[1.3.6.1 Balanced Tree Builder 9](#_Toc183364726)

[1.3.6.2 Traversal Tycoon 9](#_Toc183364727)

# **1.1 Overview**

This project aims to create an interactive platform where users can play short games that are designed to demonstrate and reinforce the concepts of various data structures and algorithms. Each game will focus on a specific data structure or algorithm and present its real-world applications in an engaging and intuitive way.

# **1.2 Objectives**

* **Educational Goals**: Provide an interactive way to learn and understand the implementation and functionality of data structures and algorithms.
* **Engagement**: Gamify learning to make it fun, especially for students and coding enthusiasts.
* **Hands-on Experience**: Allow users to practice applying data structures and algorithms in problem-solving scenarios.

# **1.3 Games**

## 1.3.1 Stack related

### ****Tower of Hanoi****

**Concept**: Players must move a stack of disks from one peg to another while following these rules:

* Only one disk can be moved at a time.
* A larger disk cannot be placed on top of a smaller disk.
* All disks must be moved to the target peg in the minimum number of moves.

**Gameplay Features**:

* Three pegs and a stack of disks with different sizes.
* Visual feedback for invalid moves.
* Display the minimum number of moves required and track the player's move count.

**Learning Outcomes**:

* Understand stack operations: push (place disk) and pop (remove disk).
* Practice recursive problem-solving logic.
* Explore the relationship between stacks and recursion.

### ****1.3.1.2. Fill Up the Bottle****

**Concept**: Players are given several bottles (representing stacks) filled with colored balls in a random order. The objective is to sort each bottle such that all balls in a bottle are of the same color.

**Rules**:

* A ball can only be moved to another bottle if the top ball in the target bottle is of the same color or the bottle is empty.
* Players can use one or more empty bottles as temporary storage.

**Gameplay Features**:

* Drag-and-drop mechanics for moving balls between bottles.
* Highlight valid moves for user guidance.
* Timer or move counter for an additional challenge.
* Success message upon sorting all bottles correctly.

**Learning Outcomes**:

* Understand stack operations: push (add ball to a bottle) and pop (remove ball from a bottle).
* Apply logical thinking to manage constraints and use temporary stacks effectively.
* Visualize the LIFO principle in a practical scenario.

## 1.3.2 Queue related

### Dine Dash

**Concept**: The game simulates a restaurant where players manage customer queues using the FIFO (First-In-First-Out) principle. VIP customers disrupt the regular order, requiring players to prioritize them while maintaining efficiency and customer satisfaction.

**Gameplay**:

* **Customers**: Regular customers are served in the order they arrive, while VIP customers must be served first, overriding the queue rules.
* **Objective**: Serve as many customers as possible before their patience runs out, balancing VIP and regular customer demands.
* **Challenges**: Limited kitchen capacity and random VIP arrivals increase the game's difficulty.

**Learning Outcomes**:  
The game introduces queue management concepts, combining basic queues for regular customers with priority queues for VIPs. It teaches real-time problem-solving and logical decision-making under constraints.

## 1.3.3 Linked List related

### ****1.3.3.1 Snake Evolution****

**Concept**: The game expands on the classic Snake concept, with a twist: if the snake's head collides with any part of its body, the collided node and all subsequent nodes are removed from the snake's body. This creates an added layer of strategy, requiring players to carefully navigate their path to avoid collisions and maintain the snake's length.

**Gameplay**:

* The snake grows longer with each collected item, represented as a dynamically linked list of nodes.
* When the snake collides with its own body:
  + The segment at the collision point and all nodes after it are deleted.
  + If only one segment remains, the game ends.
* Players must balance growth with careful path planning to avoid reducing the snake's length.

**Objective**:  
Survive as long as possible by collecting items, growing the snake, and avoiding collisions that could shorten the snake or end the game.

**Learning Outcomes**:  
This game highlights the functionality of linked lists through dynamic node addition, deletion, and traversal. It teaches logical thinking, strategic planning, and reinforces concepts of pointer manipulation in a visually engaging and interactive manner.

### 1.3.3.2 Twist and Climb

**Concept**: This is a multiplayer game where the traditional Snake and Ladder board is implemented as a **doubly linked list**. Each square of the board is represented as a node in the list, allowing forward and backward traversal. Players navigate through the nodes based on dice rolls, encountering snakes and ladders that alter their positions dynamically.

**Gameplay**:

* Each node contains the square's position and information about any special action (e.g., a snake or ladder).
* Players move across the board using dice rolls. If a player lands on a node with a ladder, they move forward to the destination node. If they land on a snake, they slide back to a previous node.
* The game supports multiple players, and their positions are tracked as pointers to nodes.
* The use of a doubly linked list allows for additional mechanics, such as backward traversal for certain game rules or undoing a move.

**Learning Outcome**:

* Understand the construction and traversal of a **doubly linked list**.
* Explore how to dynamically manipulate data structures to simulate game mechanics.
* Learn how to manage multiple players using separate pointers, demonstrating real-world applications of linked lists.

## 1.3.4 Graph related

### ****1.3.4.1**** One Way Out

**Concept**: The game is based on the Eulerian path concept. Players must traverse a graph such that each edge is visited exactly once without retracing their steps.

**Gameplay**:

* The graph consists of nodes (locations) and edges (paths between locations).
* Players start at a given node and attempt to visit every edge exactly once.
* Challenges include choosing the correct traversal order and avoiding getting stuck.
* Bonus levels introduce weighted edges, requiring players to balance edge traversal with cost.

**Learning Outcome**:

* Understanding Eulerian paths and circuits.
* Exploring practical graph traversal and edge constraints.
* Applying problem-solving to ensure a complete traversal.

### ****1.3.4.2 Dog’s Chase****

**Concept**: This game is based on shortest path algorithms. A dog tied to a rope must catch a thief by calculating the shortest path on a randomly generated graph. The rope length is equal to the shortest path distance. Players must determine the correct path within two tries to catch the thief.

**Gameplay**:

* The graph is randomly generated with nodes (locations) and weighted edges (distances).
* Players are given the starting position of the dog and the thief’s location.
* Players calculate the shortest path using algorithms like Dijkstra’s or A\*.
* If the dog chooses the correct path in the first attempt, it catches the thief. If the dog fails, it gets one more attempt. Failing twice means the thief escapes, and the player loses the game.
* Bonus levels introduce obstacles or moving thieves, adding complexity.

**Learning Outcome**:

* Understanding shortest path algorithms like Dijkstra’s and their applications.
* Practical application of weighted graphs.
* Developing problem-solving and logical thinking to analyze paths effectively.

### 1.3.4.3 Trail of Secret

**Concept: Trail of Secrets** is a graph-based exploration game where players must navigate from a starting node to a destination while avoiding hidden traps. These traps are revealed only when visited, challenging players to strategize and adapt. The game leverages BFS and DFS traversal methods for problem-solving.

**Gameplay:**

The maze is represented as a graph of nodes and edges. Some nodes are hidden traps that send players back to the start when triggered. Players must find the destination while minimizing encounters with traps. BFS can be used to find the shortest path, while DFS allows deeper exploration.

**Learning Outcomes:**

Gain practical understanding of graph traversal techniques (BFS and DFS). Develop logical problem-solving and decision-making skills. Understand risk management by navigating unknown paths efficiently.

### ****1.3.4.4 Railroad Tycoon Challenge****

**Concept**: Players must build a railway network connecting multiple cities with the minimum construction cost.

**Gameplay:**

* Each city represents a node, and potential railway tracks between cities are edges with costs.
* Players select the cheapest available tracks to connect all cities.
* If a chosen track forms a cycle, it is discarded.
* Players compete to complete the network first or with the lowest cost.

**Learning Outcomes:**

* Application of Kruskal's Algorithm for finding Minimum Spanning Trees (MSTs).
* Logical decision-making in selecting the most efficient paths.

## 1.3.5 Array related

### 1.3.5.1 Tic Tac Toe

**Concept**: Tic Tac Toe is a classic two-player strategy game played on a 3x3 grid. The goal is for one player to align three of their marks (X or O) in a row—horizontally, vertically, or diagonally—before their opponent.

**Gameplay:**

* The game starts with an empty 3x3 grid.
* Two players take turns marking one available cell with their symbol (either X or O).
* After each move, the system checks if the latest move results in a win by forming a continuous row, column, or diagonal of identical marks.
* If all cells are filled without any player achieving three marks in a row, the game ends in a draw.

**Learning Outcomes:**

* Practice manipulating arrays by representing the game grid as a 2D array.
* Learn to implement win-checking algorithms for rows, columns, and diagonals.
* Enhance understanding of turn-based logic and handling user input dynamically.

### ****1.3.5.2 Memory Match****

**Concept**: A game where players flip cards on a grid to find matching pairs.

**Gameplay**:

* The grid is filled with pairs of hidden symbols or numbers.
* Players select two cells to "flip" and reveal their contents.
  + If the symbols match, they remain visible.
  + If they don’t match, they are flipped back.
* The game ends when all pairs are matched.

**Learning Outcomes**:

* Practice storing and accessing data in 2D arrays.
* Implement logic for comparing elements.
* Enhance focus on turn-based gameplay and event handling.

### ****1.3.5.3 Word Search****

**Concept**: Players find hidden words in a grid filled with random letters.

**Gameplay**:

* A grid contains hidden words placed horizontally, vertically, or diagonally.
* Players select letters in sequence to form a word.
* The game provides a list of words to find, and players race against time.

**Learning Outcomes**:

* Practice searching patterns in 2D arrays.
* Implement algorithms for traversing diagonally.
* Build logic for validating player inputs.

## 1.3.6 Trees Related

### 1.3.6.1 Balanced Tree Builder

**Concept**: In this game, players are given a set of random nodes and must organize them into a balanced AVL tree. The tree must meet the AVL property, where the balance factor of each node (the difference between the height of the left and right subtrees) must be either -1, 0, or 1.

**Gameplay**:

* Players are provided with a set of nodes (usually integers) and tasked with inserting them into a binary search tree (BST).
* As each node is inserted, the tree must be balanced according to the AVL tree property.
* Players must detect when an imbalance occurs and perform rotations (left or right) to restore the balance.
* The game may include time limits to balance the tree and additional challenges such as inserting nodes in random order or after specific operations (like removing a node).
* Players are awarded points based on the speed and accuracy of balancing the tree, and bonus points are given for creating a perfectly balanced tree after inserting all nodes.

**Learning Outcome**:

* **Tree Construction**: Players will learn how to build a binary search tree (BST) and understand the importance of node placement.
* **AVL Tree Balancing**: Players will gain hands-on experience with AVL tree balancing and learn how to identify when the tree is unbalanced and apply rotations to restore balance.
* **Efficient Traversal**: Players will understand the impact of balance on tree traversal efficiency, particularly in terms of search times and operations.

### 1.3.6.2 Traversal Tycoon

**Concept**: In this game, players are given a tree (either a binary tree or a binary search tree) and must correctly identify the in-order, pre-order, and post-order traversal sequences for the tree.

**Gameplay**:

* The player is presented with a tree (e.g., a binary tree with nodes arranged in a random structure).
* The objective is to write or select the correct traversal orders:
* **In-order Traversal**: Visit left subtree, root, then right subtree.
* **Pre-order Traversal**: Visit root, left subtree, then right subtree.
* **Post-order Traversal**: Visit left subtree, right subtree, then root.
* The player can choose from multiple-choice options for each traversal type, or in a more advanced version, they can manually write the correct sequence of nodes.
* The game may include challenges such as trees with different structures (balanced, unbalanced, skewed), and players must adjust their traversal techniques based on the tree’s structure.
* A timer is used to encourage quick decision-making, and points are awarded for correctness and speed.

**Learning Outcome**:

* **Traversal Techniques**: Players will deepen their understanding of tree traversal methods (in-order, pre-order, and post-order) and their respective use cases.
* **Binary Tree Structure**: Players will visualize how different tree structures impact traversal results.
* **Efficiency**: Players will learn the importance of traversal in solving tree-based problems and the time complexity of different traversal methods.