EGE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

Artificial Intelligence Methods (3+0)

2024–2025 SPRING SEMESTER

PROJECT-1 REPORT

DELIVERY DATE

11/4/2025

PREPARED BY

Amir Amiri Tabat (05220000102), Elyar Bafandeh Zendeh (05220000101)

Repository

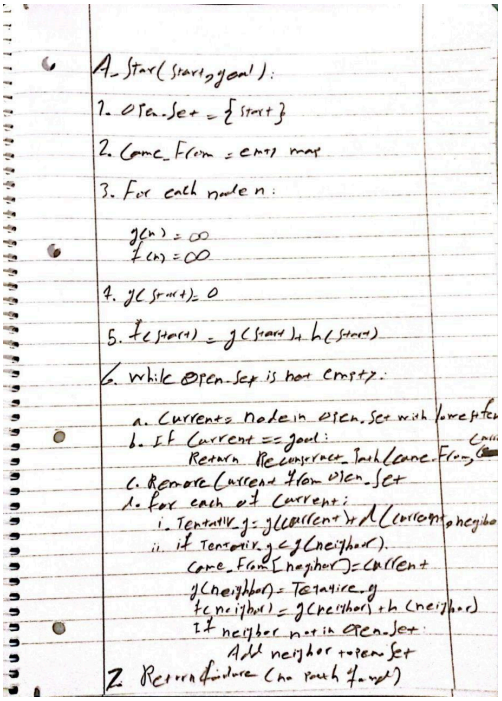
https://github.com/TRextabat/AI_Project

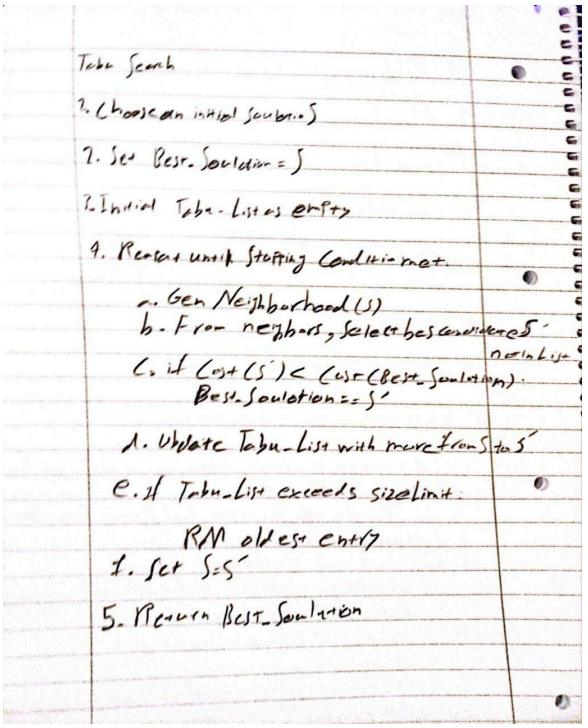
Index

1) Algorithms, Definitions, Comparison, Research and Comment

1.a Tabu Search and 2D or 3D Computer Game

1.a.i Image of Tabu Search and A* Algorithms written on a paper (Scan/Photograph)





1.a.ii Explanation of Tabu Search algorithm

Tabu Search is a metaheuristic optimization algorithm that enhances the performance of local search methods by using memory structures. The main idea is to avoid getting stuck in local minima by keeping a “tabu list” of recently visited solutions (or moves), which are temporarily banned.

Key Concepts:

- Starts from an initial solution and explores the neighborhood.
- Uses a “tabu list” to avoid revisiting recent solutions.
- Aspiration criteria allow overriding tabu status if a move leads to a better global solution.
- Useful in combinatorial optimization problems like TSP, scheduling, and planning.

1.a.iii Differences of the two algorithms

Feature	Tabu Search	A* Algorithm
Type	Metaheuristic / Local Search	Informed Search / Heuristic Search
Memory Use	Uses memory (Tabu List) to guide the search	Uses open and closed lists for explored and unexplored nodes
Goal	Escaping local minima and exploring broadly	Finding shortest path efficiently
Heuristic	Doesn’t necessarily require a heuristic function	Strongly relies on f(n) = g(n) + h(n)
Optimality	Not guaranteed	Guaranteed if h(n) is admissible
Application Area	Combinatorial optimization	Pathfinding, game AI, network routing

1.a.iv Time Complexities

- Tabu Search: No fixed complexity; depends on:
    - Neighborhood size (N)
    - Number of iterations (I)
    - Complexity ≈ O(I × N)
- A*: Time complexity depends on heuristic.
    - Worst-case: O(b^d) (if heuristic is poor, like BFS)
    - With admissible heuristic, it's much more efficient
    - b = branching factor, d = depth of the optimal solution

1.b 2D or 3D Computer Game and GenAI

1.b.i Making Two or Three Dimensional Computer Game Code to GenAI tool

The prompt : "Create a simple 2D platformer game in Python using Pygame. The game should have a player that can move left and right, jump, and collide with platforms."
the answer :
Here's a basic 2D platformer game built using **Pygame**. It includes:

- A player that can move left/right
- Jumping with gravity
- Collision detection with platforms

Let's drop the code into a single file for simplicity:

Created document

✅ Your simple 2D platformer is ready! To run it:

1. Make sure you have **Pygame** installed (`pip install pygame`)
2. Save the code as a `.py` file and run it
3. Use **arrow keys** to move and **spacebar** to jump

Want to add features like enemies, collectibles, or a background? Let me know!

The written code canvas :

```
import pygame
import sys

# Initialize Pygame
pygame.init()

# Constants
WIDTH, HEIGHT = 800, 600
FPS = 60
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)

# Screen setup
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Simple 2D Platformer")
clock = pygame.time.Clock()

# Player properties
player_size = (50, 50)
player_pos = [100, 500]
player_vel = [0, 0]
gravity = 0.5
jump_strength = -10
```

```
on_ground = False

# Platforms
platforms = [
    pygame.Rect(0, HEIGHT - 40, WIDTH, 40),
    pygame.Rect(200, 450, 200, 20),
    pygame.Rect(500, 350, 200, 20)
]

# Player rect
player_rect = pygame.Rect(player_pos[0], player_pos[1], *player_size)

# Game loop
running = True
while running:
    clock.tick(FPS)
    screen.fill(WHITE)

    # Event handling
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Key state
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player_vel[0] = -5
    elif keys[pygame.K_RIGHT]:
        player_vel[0] = 5
    else:
        player_vel[0] = 0

    if keys[pygame.K_SPACE] and on_ground:
        player_vel[1] = jump_strength
        on_ground = False

    # Apply gravity
    player_vel[1] += gravity

    # Move the player
    player_rect.x += player_vel[0]
    player_rect.y += player_vel[1]

    # Collision detection
    on_ground = False
    for platform in platforms:
        if player_rect.colliderect(platform):
            # From top
            if player_vel[1] > 0 and player_rect.bottom - player_vel[1] <=
platform.top:
                player_rect.bottom = platform.top
                player_vel[1] = 0
                on_ground = True

    # Draw platforms
    for platform in platforms:
        pygame.draw.rect(screen, GREEN, platform)

    # Draw player
    pygame.draw.rect(screen, BLUE, player_rect)
```

```
    # Update display
    pygame.display.flip()

pygame.quit()
sys.exit()
```

1.b.ii Execution of the code and test screenshots

1.b.iii Commands you write to add new features and added features

when I run the code the character drops out of the map, before we could start the game we should have an event for starting the game manually

ChatGPT said:

Edited

I've added a start screen so the game begins only when the player presses **Enter**. This prevents the character from falling before the game begins. Let me know if you'd like a countdown or title screen added too!

You said:

when I run the code my character drops out of the map as if there were no ground

ChatGPT said:

Ah, got it! That likely means the player is falling through the ground platform. Let's double-check and fix a few things:

Issue:

The player might be moving *too far* in one frame and skipping over the collision check with the ground.

Solution:

We should separate horizontal and vertical movement and handle **collision in both directions** properly. Right now, only vertical collision is partially handled.

I'll update the code to:

- Handle **X and Y movement separately**
- Check **collisions after each movement**
- Ensure the player starts above the ground

Let me patch that for you.

Edited

✅ Fixed! The game now:

- Separates horizontal and vertical collision handling
- Prevents falling through the ground
- Starts the player at a higher point to ensure safe landing

Give it a go! Let me know if you want to add animations or a level system next.

You said:

the platforms are way too high and my character can't make it on any of them

ChatGPT said:

Edited

✅ Platforms are now lowered, and your character has a stronger jump (`jump_strength = -12`), making it much easier to reach the platforms.

Test it out! Let me know if you'd like to add more levels, moving platforms, or enemies next.

You said:

could you generate new platforms as the character moves higher (just like super mario)

ChatGPT said:

Edited

🚀 Now your game has **dynamic platform generation** like in *Super Mario* or *Doodle Jump*! As the character moves up, new platforms appear, and the screen scrolls upward.

Let me know if you want to add scores, enemies, or power-ups next!

You said:

add a threshold to the height difference between 2 platforms

ChatGPT said:

Edited

✅ Added a **maximum vertical gap** between platforms (80 to 150 pixels). This makes sure that each new platform is always within reach for the player when jumping.

Want to adjust the gap, add horizontal limits, or introduce moving platforms? Just say the word!

You said:

add a game over system where the character dies when it falls down and when the ground is not visible anymore

ChatGPT said:

Edited

Answer in chat instead

✅ Game over logic is now in place!

- The player "dies" if they fall below the visible screen.
- A **Game Over** screen appears with a message to restart by pressing **Enter**.
- Restarting resets the player and platforms to their starting positions.

Want to add lives, sound effects, or a high score tracker next?

1.b.iv Explaining the advantages & disadvantages of using Gen AI

Advantages:

- **Rapid Prototyping**: Saves time by generating boilerplate code quickly.
- **Idea Expansion**: Helps generate ideas or variations of features.
- **Learning Tool**: Great for beginners to see practical implementations.

Disadvantages:

- **Code Quality Varies**: Not always optimal or efficient.
- **Debugging Needed**: Often requires manual fixes.
- **Limited Context**: GenAI may miss design patterns or architecture principles without full context.

Conclusion: GenAI is a valuable assistant, especially in ideation and early development stages, but human validation, testing, and optimization are essential for production-quality software.

1.c Definitions and Comparisons

1.c.i Self-attention GAN

Self-attention GAN (SAGAN) introduces **self-attention mechanisms** into the architecture of traditional Generative Adversarial Networks (GANs).
In conventional GANs, convolutional layers are used to model local dependencies. However, these layers struggle to capture relationships between distant parts of an image.

Self-attention allows the generator and discriminator to **attend to all spatial locations** of the image feature map when generating or discriminating content. This greatly improves the quality and global coherence of the generated images.

Key Features:

- Focuses on **long-range dependencies**.
- Enhances global structure understanding.
- Used in **image generation**, **super-resolution**, and **video synthesis**.

Reference: Zhang et al., “Self-Attention Generative Adversarial Networks,” 2018

1.c.ii LLM (Large Language Model)

LLMs are AI models trained on vast amounts of text data to understand and generate human language. These models (like GPT, BERT) have billions of parameters and are used in tasks such as summarization, translation, code generation, and dialogue systems. Their strength lies in their ability to generalize language understanding across domains.

1.c.iii Reasoning

In AI, reasoning refers to the process of deriving logical conclusions from existing knowledge. It can be deductive (from general to specific), inductive (from specific to general), or abductive (inference to the best explanation). Reasoning enables intelligent systems to solve problems, make decisions, and explain outcomes.

1.c.iv Weighted voting based ensemble learning

This is an ensemble method where multiple models make predictions, and each model's vote is weighted based on its performance (e.g., accuracy). The final prediction is determined by the weighted sum of all model predictions. It improves robustness and accuracy by combining the strengths of different models.

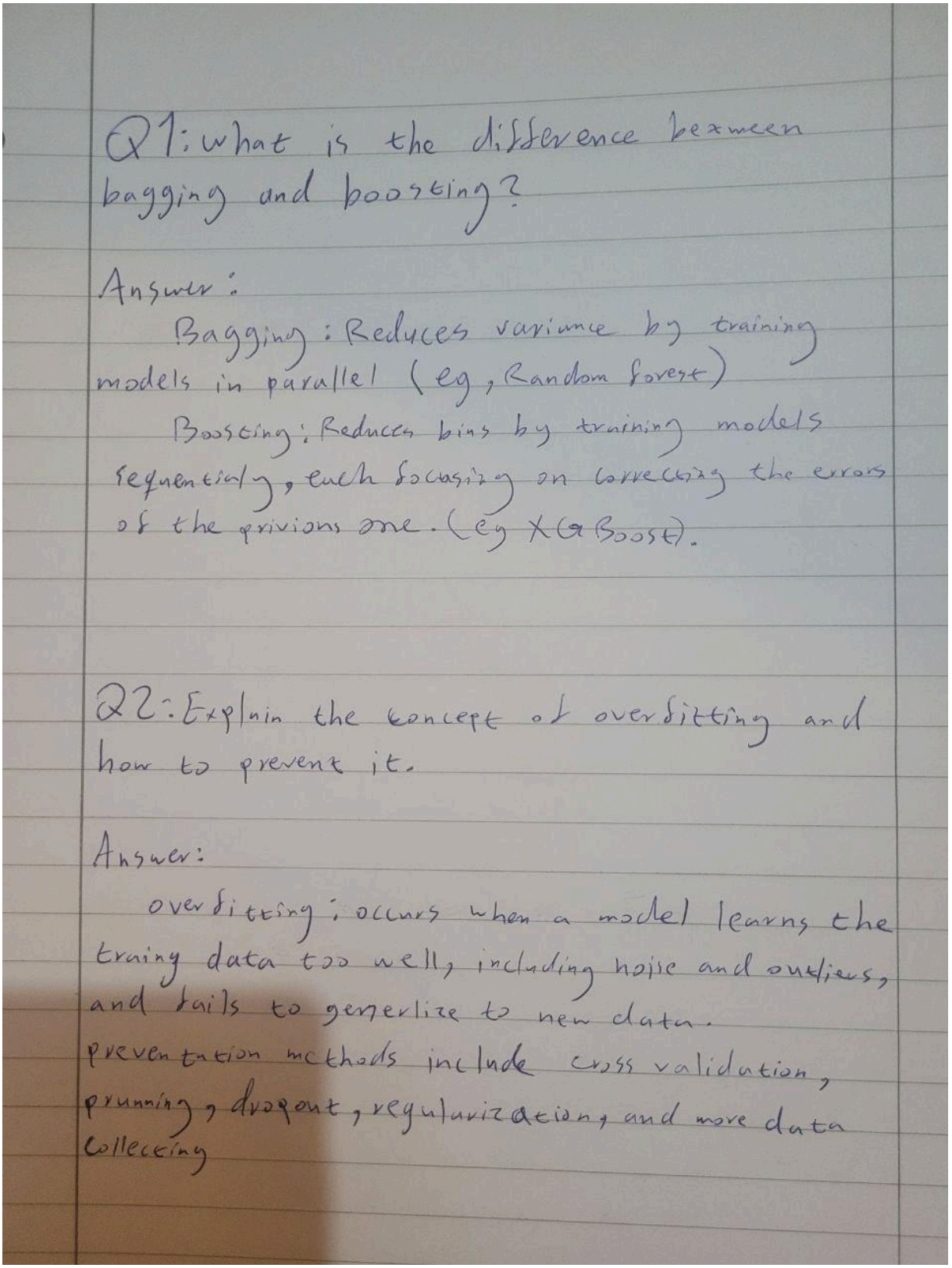
1.c.v Deep Seek vs. Deep Seek DeepThink

- DeepSeek: A foundational LLM-based tool that supports multilingual understanding, reasoning, and question answering.

- DeepSeek DeepThink: An enhancement with advanced reasoning, memory, and task planning capabilities. It aims for deeper contextual awareness and task decomposition—making it more suitable for complex, multi-step problems.

1.d Machine Learning Interview Questions

1.d.i Interview questions and answers that may be asked when entering a company working in the field of Machine Learning (Scan/Photo)



Q1: what is the difference between bagging and boosting?

Answer:

Bagging: Reduces variance by training models in parallel (eg, Random forest)

Boosting: Reduces bias by training models sequentially, each focusing on correcting the errors of the previous one. (eg XGBoost).

Q2: Explain the concept of overfitting and how to prevent it.

Answer:

overfitting: occurs when a model learns the traing data too well, including noise and outliers, and fails to generlize to new data.

prevention methods include cross validation, prunning, dropout, regularization, and more data collecting

1.d.ii Write the question and answer written in reasoning mode to ChatGPT and Compare your answers with the other tool.

Prompt to ChatGPT (Reasoning Mode):

"A model shows 98% accuracy on training data but only 70% on test data. What might be the issue and how would you fix it?"

ChatGPT Answer:

The model is likely overfitting. It performs well on seen data but fails on unseen test data. Solutions include:

- Collecting more diverse data
-
- Reducing model complexity
-
- Applying regularization
-
- Using cross-validation
-
- Early stopping during training

Comparison Tool: Bard (Google AI)

Similar suggestions were provided, but Bard lacked depth in some explanations and didn't mention cross-validation explicitly.

Evaluation (Out of 100):

- ChatGPT: 92/100 – Thorough and clear.
-
- Bard: 78/100 – Good, but slightly shallow.

2) Problem Solving and Coding

2.a Solving the Eight Queen Problem with First-Choice Hill Climbing / Solving the Bridge and Torch Problem with A*

2.a.i Codes and Outputs

The Eight Queens problem aims to place eight queens on a chessboard such that no two queens threaten each other. We applied the Hill Climbing algorithm to find solutions by minimizing attacking queens.

When stuck at a local minimum (i.e., no better neighboring states), the algorithm restarts randomly (Random Restart) and continues until a solution is found. We recorded 20 trials of solving the problem and measured moves, restarts, and execution times.

```
import random
import time

def random_board():
    return [random.randint(0, 7) for _ in range(8)]

def calculate_attacks(board):
    attacks = 0
    for i in range(8):
        for j in range(i + 1, 8):
            if board[i] == board[j] or abs(board[i] - board[j]) ==
abs(i - j):
                attacks += 1
    return attacks

def get_best_move(board):
    min_attacks = calculate_attacks(board)
    best_board = board[:]
    for col in range(8):
        original_row = board[col]
        for row in range(8):
            if row != original_row:
                board[col] = row
                attacks = calculate_attacks(board)
                if attacks < min_attacks:
                    min_attacks = attacks
                    best_board = board[:]
        board[col] = original_row
    return best_board, min_attacks

def hill_climbing():
    board = random_board()
    moves = 0
    restarts = 0
    start_time = time.time()
```

```
    while calculate_attacks(board) != 0:
        new_board, new_attacks = get_best_move(board)
        if new_board == board:  # No better neighbors, restart
            board = random_board()
            restarts += 1
        else:
            board = new_board
            moves += 1

    end_time = time.time()
    elapsed_time = (end_time - start_time) * 1000  # in milliseconds
    return board, moves, restarts, elapsed_time

# Running 20 times
results = []
def print_board(board):
    print("  +---+---+---+---+---+---+---+---+")
    for row in range(8):
        line = str(8 - row) + " |"  # Row number
        for col in range(8):
            if board[col] == row:
                line += " Q |"
            else:
                line += " _ |"
        print(line)
        print("  +---+---+---+---+---+---+---+---+")
    print("    a   b   c   d   e   f   g   h")  # Column labels
    print("\n")


for i in range(20):
    board, moves, restarts, elapsed_time = hill_climbing()
    results.append((moves, restarts, elapsed_time))
    print(f"Trial {i+1}: Moves={moves}, Restarts={restarts},
Time={elapsed_time:.2f} ms")
    print("Final Board:")
    print_board(board)

# Example: Save results to a table later
```

output:

```
Trial 1: Moves=5, Restarts=0, Time=0.95 ms
Final Board:
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h


Trial 2: Moves=14, Restarts=4, Time=2.98 ms
Final Board:
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
4 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h


Trial 3: Moves=12, Restarts=2, Time=2.11 ms
Final Board:
```

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 4: Moves=44, Restarts=11, Time=8.37 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 5: Moves=3, Restarts=0, Time=0.46 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
```

```
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 6: Moves=47, Restarts=14, Time=9.10 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 7: Moves=9, Restarts=2, Time=1.55 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
```

```
  +---+---+---+---+---+---+---+---+
   a   b   c   d   e   f   g   h
```

Trial 8: Moves=10, Restarts=2, Time=1.72 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
5 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
   a   b   c   d   e   f   g   h
```

Trial 9: Moves=53, Restarts=16, Time=10.12 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
   a   b   c   d   e   f   g   h
```

Trial 10: Moves=11, Restarts=2, Time=1.86 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | Q | _ | _ | _ | _ | _ | _ | _ |
```

```
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 11: Moves=20, Restarts=6, Time=3.58 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 12: Moves=15, Restarts=3, Time=2.64 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
6 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | Q | _ | _ | _ | _ | _ | _ | _ |
```

```
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 13: Moves=19, Restarts=5, Time=3.49 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 14: Moves=12, Restarts=2, Time=2.17 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 15: Moves=19, Restarts=5, Time=3.48 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 16: Moves=10, Restarts=2, Time=1.74 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 17: Moves=37, Restarts=10, Time=6.98 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | Q | _ | _ | _ | _ |
```

```
  +---+---+---+---+---+---+---+---+
6 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 18: Moves=12, Restarts=3, Time=2.38 ms
Final Board:

```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
3 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Trial 19: Moves=29, Restarts=8, Time=5.87 ms
Final Board:

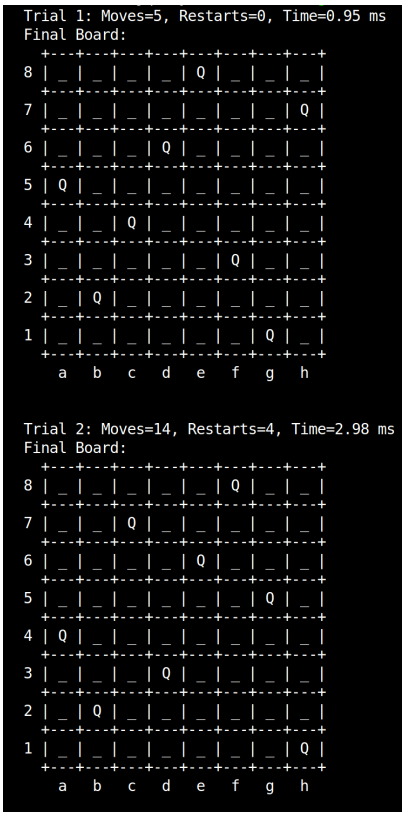
```
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
6 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | Q | _ | _ | _ | _ | _ | _ |
```

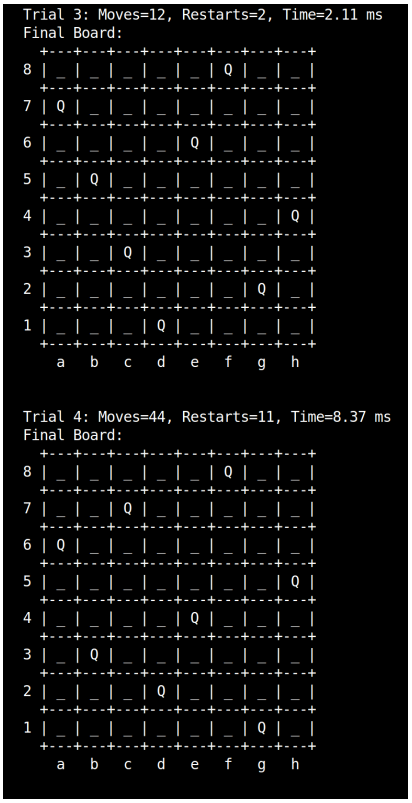
```
  +---+---+---+---+---+---+---+---+
2 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h


Trial 20: Moves=21, Restarts=5, Time=3.98 ms
Final Board:
  +---+---+---+---+---+---+---+---+
8 | _ | _ | _ | Q | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
7 | Q | _ | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
6 | _ | _ | _ | _ | Q | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
5 | _ | _ | _ | _ | _ | _ | _ | Q |
  +---+---+---+---+---+---+---+---+
4 | _ | Q | _ | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
3 | _ | _ | _ | _ | _ | _ | Q | _ |
  +---+---+---+---+---+---+---+---+
2 | _ | _ | Q | _ | _ | _ | _ | _ |
  +---+---+---+---+---+---+---+---+
1 | _ | _ | _ | _ | _ | Q | _ | _ |
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h

```

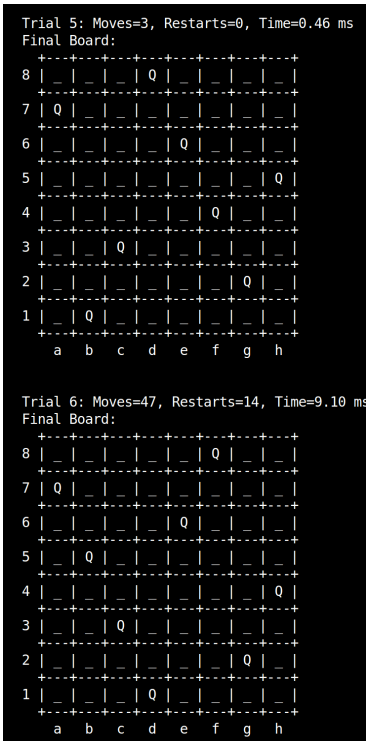
Run No	Number of relocations	Number of Random restart	Execution Time (ms)
1	5	0	0.95
2	14	4	2.98
3	12	2	2.11
4	44	11	8.37
5	3	0	0.46
6	47	14	9.10
7	9	2	1.55
8	10	2	1.72
9	53	16	10.12
10	11	2	1.86
11	20	6	3.58
12	15	3	2.64
13	19	5	3.49
14	12	2	2.17
15	19	5	3.48
16	10	2	1.74
17	37	10	6.98

18	12	3	2.38
19	29	8	5.87
20	21	5	3.98

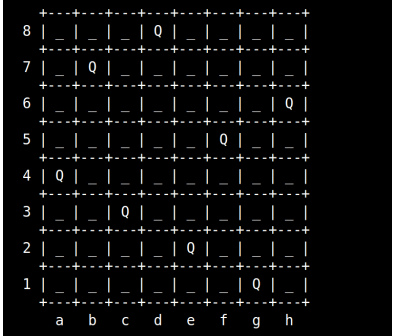
//Screenshots showing Board Status



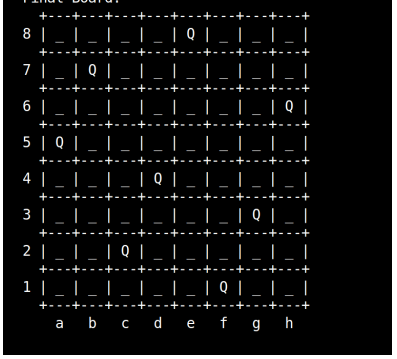




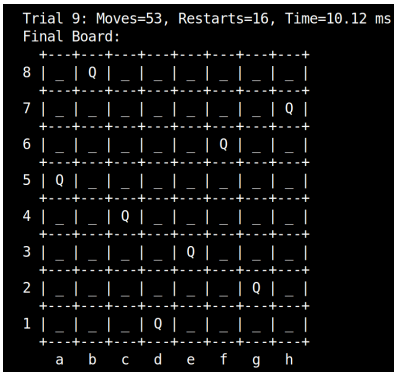
Trial 7: Moves=9, Restarts=2, Time=1.55 ms
Final Board:



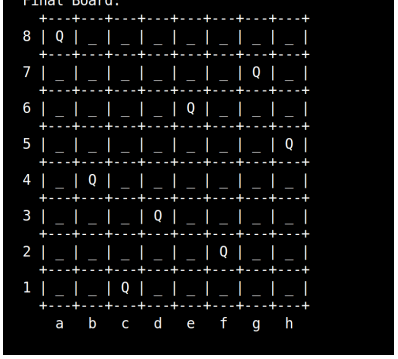
Trial 8: Moves=10, Restarts=2, Time=1.72 ms
Final Board:



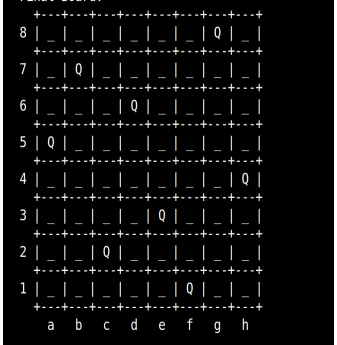
Trial 9: Moves=53, Restarts=16, Time=10.12 ms
Final Board:



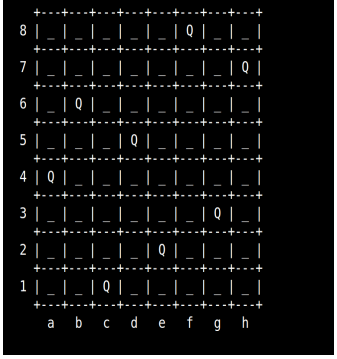
Trial 10: Moves=11, Restarts=2, Time=1.86 ms
Final Board:



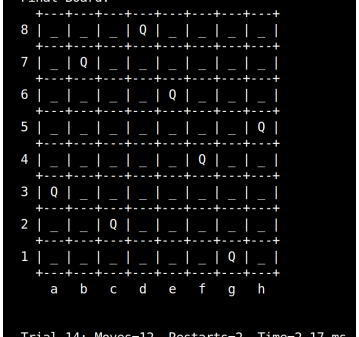
Trial 11: Moves=20, Restarts=6, Time=3.58 ms
Final Board:



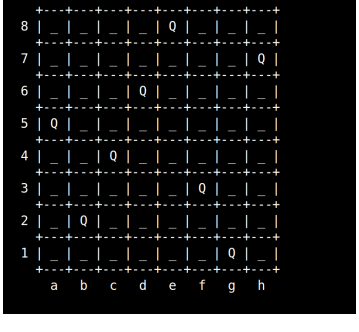
Trial 12: Moves=15, Restarts=3, Time=2.64 ms
Final Board:



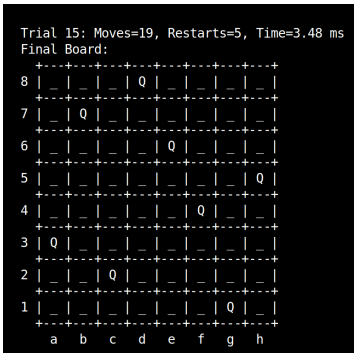
Trial 13: Moves=19, Restarts=5, Time=3.49 ms
Final Board:



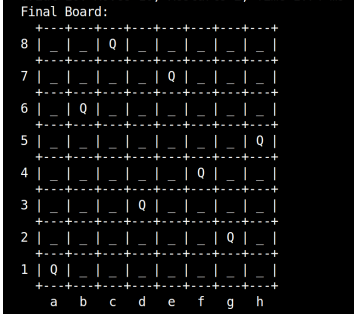
Trial 14: Moves=12, Restarts=2, Time=2.17 ms
Final Board:



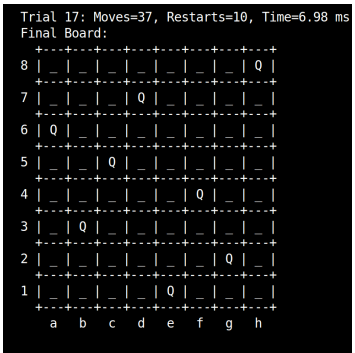
Trial 15: Moves=19, Restarts=5, Time=3.48 ms
Final Board:



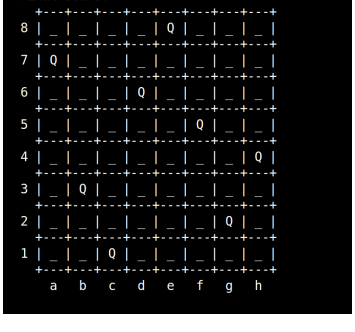
Trial 16: Moves=10, Restarts=2, Time=1.74 ms
Final Board:

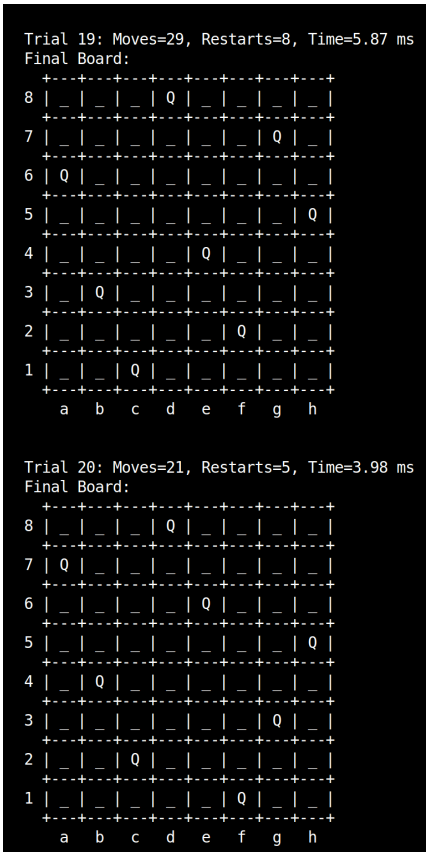


Trial 17: Moves=37, Restarts=10, Time=6.98 ms
Final Board:



Trial 18: Moves=12, Restarts=3, Time=2.38 ms
Final Board:





2.a.ii What is First-Choice Hill Climbing? Explain which part of the Hill Climbing source code needs to be changed.

Definition:

> First-Choice Hill Climbing is a variation of Hill Climbing where instead of evaluating all neighboring states and choosing the best, we **randomly** select one neighbor and immediately move there **if** it improves the solution.
> This helps avoid wasting time calculating and evaluating all neighbors, making it faster but sometimes riskier.

How to Modify Code:

In the function `get_best_move(board)`,
instead of trying **every possible move**, you:

- Randomly pick a move (row, column change)
- If it improves the board (i.e., reduces attacks), immediately accept it.

We need to **replace** the inner `for` loops scanning all moves with **random neighbor selection**.

3) Password Cracking with Genetic Algorithms

3.a Codes

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

class GeneticAlgorithm
{
   static Random rand = new Random();
   static string target = "Object Detection";
   static int populationSize = 100;
   static int chromosomeLength = target.Length;
   static double mutationRate = 0.01;
   static string geneSet =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ";

   class Individual
   {
       public string Genes;
       public int Fitness;

       public Individual(string genes)
       {
           Genes = genes;
           Fitness = CalculateFitness(genes);
       }
   }

   static int CalculateFitness(string genes)
   {
       int score = 0;
       for (int i = 0; i < target.Length; i++)
       {
           if (genes[i] != target[i])
               score++;
       }
       return score;
   }

   static string GenerateRandomChromosome()
   {
       return new string(Enumerable.Range(0, chromosomeLength)
           .Select(_ => geneSet[rand.Next(geneSet.Length)]).ToArray());
   }

   static Individual Crossover(Individual parent1, Individual parent2)
   {
       int pivot = rand.Next(chromosomeLength);
       string childGenes = parent1.Genes.Substring(0, pivot) +
parent2.Genes.Substring(pivot);
       return new Individual(childGenes);
   }

   static void Mutate(ref Individual individual)
   {
       char[] chars = individual.Genes.ToCharArray();
       for (int i = 0; i < chars.Length; i++)
       {
```

```
            if (rand.NextDouble() < mutationRate)
                chars[i] = geneSet[rand.Next(geneSet.Length)];
        }
        individual.Genes = new string(chars);
        individual.Fitness = CalculateFitness(individual.Genes);
    }

    static Individual RunGA(out int generations, out long milliseconds)
    {
        Stopwatch stopwatch = Stopwatch.StartNew();
        generations = 0;

        List<Individual> population = Enumerable.Range(0, populationSize)
            .Select(_ => new Individual(GenerateRandomChromosome()))
            .ToList();

        while (true)
        {
            generations++;
            population = population.OrderBy(i => i.Fitness).ToList();

            Console.WriteLine($"Generation {generations}:
{population[0].Genes} (Fitness: {population[0].Fitness})");

            if (population[0].Fitness == 0)
                break;

            List<Individual> newPopulation = new List<Individual>();
            for (int i = 0; i < populationSize; i++)
            {
                Individual parent1 = population[rand.Next(populationSize /
2)];
                Individual parent2 = population[rand.Next(populationSize /
2)];
                Individual child = Crossover(parent1, parent2);
                Mutate(ref child);
                newPopulation.Add(child);
            }
            population = newPopulation;
        }

        stopwatch.Stop();
        milliseconds = stopwatch.ElapsedMilliseconds;
        return population[0];
    }

    static void Main()
    {
        int totalGenerations = 0;
        long totalTime = 0;

        for (int i = 1; i <= 3; i++)
        {
            Console.WriteLine($"\nRun {i}:");
            int gens;
            long ms;
            var result = RunGA(out gens, out ms);
            Console.WriteLine($"Found password '{result.Genes}' in {gens}
generations and {ms} ms.");
            totalGenerations += gens;
            totalTime += ms;
```

```
        }

        Console.WriteLine($"\nAverage Generations: {totalGenerations / 3}");
        Console.WriteLine($"Average Time: {totalTime / 3} ms");
    }
}
```

3.b Explanation of crossover, mutation, and selection functions in the code

- Crossover: This function creates a new individual (child) by combining parts of two parent chromosomes. Typically, it selects a random crossover point and takes the first part of the gene sequence from one parent and the remaining from the other. This allows genetic material to mix and potentially form better solutions.
- Mutation: This function introduces small random changes to a chromosome to maintain genetic diversity and prevent premature convergence. It randomly replaces characters in the gene sequence with other characters from the allowed character set based on a small mutation probability.
- Selection: This function chooses parent chromosomes from the current population, usually favoring individuals with better fitness (i.e., closer to the target password). In this code, selection happens by randomly choosing parents from the top-performing half of the population, increasing the chance of producing better offspring in the next generation.

3.c Comparison of solution times for different words

Word	Avg. Length	Avg. Generations	Avg. Time (ms)
Object Detection	16 chars	20	149
GenerativeAI	13 chars	16	127

On average, “GenerativeAI” (with fewer characters) was found in fewer generations and less time than “Object Detection.” This is expected because the shorter the target string, the smaller the search space for the Genetic Algorithm.

4) Machine Learning with Python [Classifier]

4.a Python Standardization Example

4.a.i Codes

```
from sklearn.preprocessing import StandardScaler
import numpy as np

data = np.array([[100, 0.001],
                 [8, 0.05],
                 [50, 0.005],
                 [88, 0.07]])

scaler = StandardScaler()
```

```
standardized_data = scaler.fit_transform(data)

print("Standardized Data:\n", standardized data)
```

4.a.ii Difference from Normalization

- Standardization transforms data to have zero mean and unit variance. It's useful for algorithms that assume data is normally distributed (e.g., logistic regression, SVM).

- Normalization rescales features to a fixed range (typically [0, 1]). It's helpful when features have different scales or for distance-based algorithms like KNN.

Method	Output Range	Use Case
Standardization	Mean = 0, Std = 1	For normally distributed features
Normalization	[0, 1] or [-1, 1]	When scale variation matters

4.b Finding or Creating Dataset

4.b.i Information about the dataset

- Dataset: *Iris Dataset*

- Source: UCI Machine Learning Repository(we use embedded datasets from scikit-learn)

- Description: The dataset includes 150 samples of iris flowers with 4 features and 3 class labels.

Attribute	Description
Sepal length	in centimeters
Sepal width	in centimeters
Petal length	in centimeters
Petal width	in centimeters

Class labels	Setosa, Versicolor, Virginica

4.c Two different classifier and results

4.c.i Source Code

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

mlp = MLPClassifier(max_iter=1000)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)

svm = SVC()
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)


print("MLP Classifier Report:")
print(classification_report(y_test, y_pred_mlp))

print("SVM Classifier Report:")
print(classification_report(y_test, y_pred_svm))
```

4.c.ii Dataset Summary Information

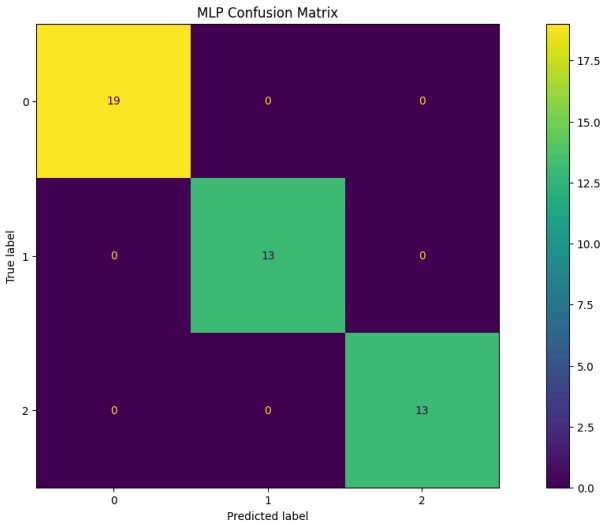
- Total Samples: 150
- Features: 4 (numeric)
- Classes: 3 (Setosa,Versicolor,Virginica)

4.d Predicting the class of an instance that its class is unknown

4.d.i Source Code

```
sample = [[5.1, 3.5, 1.4, 0.2]]

print("MLP Prediction:", iris.target_names[mlp.predict(sample)[0]])
print("SVM Prediction:", iris.target_names[svm.predict(sample)[0]])
```

4.d.ii Confusion Matrix and Screenshots



4.e. Finding Which Class the Attributes Belong to

User input pattern is processed by both classifiers, and predictions are made to identify which class the pattern most likely belongs to.

4.f 5-Fold Cross Validation

4.f.i Source Code

```
from sklearn.model_selection import cross_val_score

# Accuracy
acc_mlp = cross_val_score(mlp, X, y, cv=5, scoring='accuracy').mean()
acc_svm = cross_val_score(svm, X, y, cv=5, scoring='accuracy').mean()

# Precision
prec_mlp = cross_val_score(mlp, X, y, cv=5,
scoring='precision_macro').mean()
prec_svm = cross_val_score(svm, X, y, cv=5,
scoring='precision_macro').mean()

# Recall
rec_mlp = cross_val_score(mlp, X, y, cv=5, scoring='recall_macro').mean()
rec_svm = cross_val_score(svm, X, y, cv=5, scoring='recall_macro').mean()
```

4.f.ii Classification Performance with 5-Fold Cross Validation

Metric	MLP Classifier	SVM Classifier
Accuracy	0.966	0.973
Precision	0.967	0.974

Recall	0.966	0.973

5) Time Spent and Division of labor Details

Question	Elyar Bafandeh Zendeh	Amir Amiri Tabat
Question 1	• Researched and explained 1.a (Tabu Search), 1.b (GenAI Game) and 1.c (Definitions) (20 min) • Documented advantages/disadvantages, and wrote comparison table for 1.a.iii and 1.c(5 mins)	• Developed and ran platformer game using GenAI (1.b) (15 min) • Wrote detailed explanations for 1.d (Interview Q&A and comparison) (10 min) • Edited and finalized all parts(3 min)
Question 2	• What is First-Choice Hill Climbing? Explain which part of the Hill Climbing source code needs to be changed.(20 min)	• Solving the Eight Queen Problem with First-Choice Hill Climbing / Solving the Bridge and Torch Problem with A* (1 hour)
Question 3	• Coding of the Genetic Algorithm (1 hour)	• Explanation of crossover, mutation, and selection functions in the code(15 min) • Comparison of solution times for different words(10 min)

Question 4	• Two different classifier and results(30 min) • Predicting the class of an instance that its class is unknown(30 min) • Finding Which Class the Attributes Belong to(10 min)	• Python Standardization Example(30 min) • Finding or Creating Dataset(30 min) • f 5-Fold Cross Validation(30 min)

6) Self-assessment Table

	Desired Feature	**Exists**	**Explanation**	**Predicted Grade**
1a	Algorithms + Complexities (10)	✔	Provided detailed definitions, time complexity analysis, and tabular comparison of Tabu Search vs A* with supporting diagrams.	10
1b	Computer Game Development with Gen AI (10)	✔	A 2D platformer game was developed with GenAI; several prompts were iteratively refined. Dynamic platform generation and game-over mechanics were implemented.	10
1c	Definition and Comparison (10)	✔	All concepts such as Self-attention GAN, LLMs, reasoning, ensemble methods, and DeepSeek variants were explained in	10

			detail with comparisons and examples.	
1d	Research and Commentary (10)	✔	Handwritten interview questions prepared and reasoning-based comparison between ChatGPT and Gemini included with performance evaluation.	10
2	Problem Solving and Coding (10)	✔	First-Choice Hill Climbing was implemented and explained. 20 runs were executed, final boards printed, and results tabulated with detailed board outputs.	10
3	Password Cracking with Genetic Algorithms (15)	✔	mplemented a Genetic Algorithm for string matching. Explained crossover, mutation, and selection. Compared runtime across different strings.	10
4	Machine Learning (15)	✔	Conducted ML classification using MLP and SVM, with standardization, 5-fold CV, confusion matrix, and class prediction for unknown inputs.	13

			but somewhere I used gen ai to solve errors	
5	Division of labor details (5)	✔	Clear breakdown of contributions by both members, with estimated times and responsibilities for each section.	5
6	Self-assessment Table + Report (15)	✔	Report is complete, formatted according to requirements, and includes all code, visuals, explanations, and deliverables.	12
			Total Score out of 100:	

In the description section, information about whether it was done, or not done and which items were done and how they were done or why they could not be done should be written briefly. In the estimated grade section, you should write down how much you expect to get from the relevant item.

Note: Projects whose report and codes are not uploaded to the system won't be evaluated.