TEAM NAME : TEAMOG
MEMBER 1: SANJAY KUMAR TAMMINANI
MEMBER 2: KIMMI RISHI

PROBLEM 0: THE CLASSICAL RANDOM WALK

1. Objective

The primary goal of this problem is to establish a rigorous classical benchmark for the speed and spread of an exploratory process on the integer line. By simulating the unbiased, memory-less **Classical Random Walk (CRW)**, we compute and plot the Root-Mean-Squared Displacement (RMSD) as a function of the number of steps(N), which will serve as the baseline for evaluating the enhanced speed of the subsequent Quantum Walk (QW) problems.

2. Theoretical Background

The position X_N of a walker after N steps is the sum of N independent random variables, $s_i \in \{-1, +1\}$. The average position is $\langle X_N \rangle = 0$.

The crucial metric is the **Root-Mean-Squared Displacement (RMSD)**, which is derived from the Mean Squared Displacement (MSD), $\langle X_N^2 \rangle = N$. This relationship yields the scaling law:

$$RMSD(N) = \sqrt{\langle X_N^2 \rangle} = \sqrt{N}$$

This scaling $RMSD \propto N^{1/2}$, defines diffusive spreading.

3. Simulation Methodology

A Monte Carlo simulation was implemented in Python to model the 1D CRW.

- Metric Computed: Root-Mean-Squared Displacement (RMSD), calculated as the square root of the average of the squared position over all walks at each step N.
- Parameters:
  - Maximum Steps: $N_{max} = 1000$
  - Number of Independent Walks M = 5000
- Implementation: The moment was simulated by summing random choices of +1 and -1 for each walk. The final RMSD was calculated across the ensemble of M walks of M walks for all steps $N \in [0,1000]$

4. Results and Analysis

The simulation results confirm the theoretical prediction for the CRW.
Property = Classical Random Walk
Scalinfg Law = $RMSD(N) \propto N^{1/2}$
Behaviour = Diffusive

As shown in the plot, the **Simulated RMSD** curve closely follows the **Theoretical scaling** $(\sqrt{N})$. The slow, non-linear growth confirms the walk I a diffusive process with a scaling exponent of a = 1/2. The plot also includes the **Ballistic Scaling (O(N))** line, which establishes the expected linear growth rate for the quantum walk, highlighting the exponential advantage of the QW.

5. Instructions to Run the Code

The simulation code is saved as **classical_walk.py** in the **code/** folder.

Run the script

"python classical_walk.py"

The script will display the plot and automatically save the output image as **classical_walk_rmsd.png** in the present directory's **results/** folder **(../results/)**.

6. Conclusion

The simulation establishes the fundamental limit of classical exploration as **diffusive growth (O($\sqrt{(N)}$))**. This process the necessary benchmark for comparison with the quantum walk, which is expected to spread quadratically faster with **ballistic growth (O(N))**.

PROBLEM 1: A QUANTUM COIN FLIP (PAUL-X COIN)

1. Objective

The objective is to simulate the most basic form of a Quantum Walk (QW) using the **Paul-X gate** as the coin operator. The test determines if the quantum mechanics, without **superposition,** can introduce spreading, thereby confirming the necessity of quantum interference effects for the QW advantage.

2. Theoretical Background

The simulation models a 1D walk on a minimal 2-site lattice (positions x = {0, 1}).

- Initial State: The walker starts in a definite state: $|\Psi_0\rangle = |x = 0\rangle \otimes |c = 0\rangle$
- Coin Operator (C): The Paul-X gate (X) acts deterministically on the coin qubit: $X|c\rangle$. This operation is purely **deterministic** ($|0\rangle \leftrightarrow |1\rangle$).
- Total Step: The unitary operation $U_{step} = S \cdot C(Shift \cdot Coin)$ dictates the evolution. Since the input state is define and the oin operator is deterministic, the state $|\Psi_N\rangle$ is a single computational basis state at every step.

3. Simulation

The simulation tracks the vector $|\Psi\rangle$ over a fixed number of steps using matrix multiplication in a 4-dimensional Hilbert space (representing the two positions and two coin states)

- Steps: The number of steps is fixed in the code at **Steps = 5.**
- Code Implementation: The code uses the fixed 4 x 4 matrices for the coin ($C = I \otimes X$) and the corrected shift (S) operators to define the total step U_{step}. The final position is determined by checking which position subspace (x=0 or x=1) holds the full probability amplitude.

4. Results and Analysis

The simulation output reveals a consistent, non-exploratory path:

Observed path (for 5 steps): [0, 1, 0, 1, 0, 1]

- ***What path does he follow?***

The walker follows a fixed path of alternating positions: 0 -> 1 -> 0 -> 1 -> 0 -> 1 -> 0 ->....

- ***<u>Does this behaviour look familiar?</u>***
  Yes. This behaviour is a classical period - 2 cycle. It is deterministic and shows zero spreading, indicating that the quantum system is not utilizing any probabilistic or superposition-based exploration.
    - <u>Behaviour Classification:</u> The walk is limited O(1) **spreading** (constant displacement). This confirms that simply using a qubit is not enough; the walk only exhibits classical dynamics because the Paul-X coin fails to introduce the necessary **superposition** of position and coin states.

5. Instructions To Run The Code

The simulation code is saved as "**quantum_walk_x_coin.py**" in the "**code/**" folder.

6. Conclusion

The simulation of the QW with the Paul-X coin demonstrates a **failure to achieve quantum advantage**. The walk remains a **classical, oscillatory, and non-exploratory** process due to the lack of superposition. This result is crucial, as it isolates the finding that the coin operation must actively generate superposition (as the Hadamard coin will in the nest problem) to unlock the enhanced **ballistic spreading** of a true quantum walk.

PROBLEM 2: THE SUPEROSED WALKER (HADAMARD COIN)

1. Objective

The primary objective is to simulate the standard 1D Quantum walk (Qw) with the Hadamard gate as the coin operator.This aims to demonstrate the powerful effect of **superposition** and **interference**, and to confirm the predicted **ballistic spreading** that significantly outperforms the classical random walk benchmark established in Problem 0.

2. Theoretical Background

The quantum walk introduces randomness through the Hadamard coin, which creates superposition at every step.

- Hadamard Coin (C): The Hadamard Coin (H) is applied to the coin qubit: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This operation places the qubit into a superposition of moving left ($|0\rangle$) and moving right ($|1\rangle$).
- Initial State: The walk starts at $|\Psi_0\rangle = |x = 0\rangle \otimes |0\rangle$.
- Evolution: The combined state evolves over N steps under the unitary operation $U_{step} = S \cdot (I \otimes H)$. This process allows the walker to explore all possible paths simultaneously in a superposition of positions.
- Spreading: The curcial quantum effect is that the amplitudes interfere with each other-some paths constructively reinforce, while others destructively cancel. Theoretically, this leads to **ballistic spreading**, where the Root-Mean-Squared Displacement (RMSD) grows linearly with time: $RMSD(N) \propto O(N)$.

Position amplitude for Early Steps (Analytical Insight)

$$|Step(t)|PositionAmplitudes|\Psi_t\rangle|MaxPosition||:---:|:---:|:--:||1|\frac{1}{\sqrt{2}}|-1,0\rangle+\frac{1}{\sqrt{2}}|1,1\rangle|\pm 1||2|\frac{1}{2}|-2,0\rangle+\frac{1}{2}(|0,0\rangle+|0,1\rangle)-\frac{1}{2}|2,1\rangle|\pm 2|$$

3. Simulation

The simulation was performed using a matrix approach to calculate the final state vector after N Steps.

- Steps: The simulation was run for a fixed N = 50 steps.
- Method: The total unitary operation U_{step} was applied N times to the initial state using matrix exponentiation ($\Psi_N = (U_{step})^N \Psi_0$).
- Output Matrix: The final state vector Ψ_N was used to compute the probability distribution P(x) for all possible positions $x \in [-N, N]$.

4. Results and Analysis

The resulting probability distribution plot (for N = 50) is shown below.

- **Distribution Shape:** The distribution is highly non-Gaussian, characterized by **two prominent peaks** located far from the origin and near the edges of the possible position range. This distinct shape is the signature of quantum interference.
- **Peak Location:** For N=50 steps, the maximum position reached by the probability clusters is approximately $x \approx \pm 40$.
- **Scaling Check:** The classical random walk ($RMSD \propto \sqrt{N}$) would only spread to $RMSD \approx \sqrt{50} \approx 7.1$ positions. The QW, however, concentrates its probability far beyond this point.

Does this spread faster that the classical random walk?
Why?

Yes, the quantum walk spreads quadratically faster.
The $RMSD$of the quantum walk scales as O(N) (ballistic spreading), whereas the classical random walk scales as $O(\sqrt{N})$ (diffusive spreading). The concentration of probability near the maximum possible displacement (40 out of a possible 50) confirms this linear relationship. The quadratic speedup is achieved because the superposition of paths allows amplitudes to interfere constructively at the edges of the distribution, leading to rapid, directed movement, rather than the slow, erratic spreading of the classical case.

5. Instructions to run the code

The simulation code is saved as "**quantum_walk_hadamard.py**" in the "**code/**" folder.

The script will display the plot automatically save the output image as "**hadamard_walk_distribution.png"** in the parent directory "**results/"** folder **(../results/)**.

6. Conclusion

The simulation of the QW with the Hadamard coin successfully demonstrates the core advantage of quantum computation: **ballistic spreading** $(RMSD \propto O(N))$. By introducing superposition via the Hadamard gate, the walker is able to leverage **quantum interference** to suppress probabilities near the center and concentrate them at the extremes. This quadratic speedup over the classical walk $(RMSD \propto O(\sqrt{N}))$ is the foundational principle used in algorithms like Grover's search.

PROBLEM 3: GRAPH-BASED COMPUTATION (SEARCH ON A CYCLE)

1. Objective:

The objective is to compare the performance of the **Quantum Walk (QW)** against the **Classical Random Walk (CRW)** for a search problem on a simple graph. This demonstrates how the QW's interference properties affect the probability of finding a specific target vertex on a closed-loop graph.

2. Theoretical Background:

The simulation takes place on a **cycle graph** with L=5 vertices (0, 1, 2, 3, 4) in a periodic 1D structure. The walker starts at vertex x=0, and the target vertex is x=3. The comparison is performed over a fixed T=8 steps.

- **QW Operator:** The total step operator, $U_{step} = S \cdot C$, utilizes the Hadamard coin (H) and a periodic shift (S) operator to move the walker based on the coin state, ensuring the walk wraps around the graph (e.g.,$4 \rightarrow 0 and 0 \rightarrow 4$).
- **Initial State (QW):** The initial state is $|\Psi_0\rangle = |x = 0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, placed in a superposition of moving both left and right.
- **CRW Operator:** The classical process is governed by a stochastic transition matrix P, where the probability of moving left or right from any vertex is 0.5.
- **Spreading and Search:** On a cycle of size L, the CRW takes $O(L^2)$time to find a target with high probability, while the QW provides a **quadratic speedup**, achieving the same in O(L) time. However, due to strong interference on a small, symmetric graph, the QW probability *oscillates* and may be lower than the classical probability at specific non-optimal times, such as T=8.

3. Simulation:

The simulation calculates the success probability P(Target = 3)after T = 8 steps for both models.

- Evolution Method: To ensure numerical stability and avoid errors associated with "**np.linalg.matrix_power**", the code implements a robust iterative

method for both walks, expilicitly applying the evolution matrix (U_{step} or P)T times in a loop.

- Parameters used:
  - L(Graph Size) = 5
  - T (Steps) = 8
  - Target_V = 3

4. Results and Analysis:

The simulation yields the following probabilities

Does the Quantum Walk outperform the Classical Walk at $T=8$?

No, the Quantum Walk does not outperform the Classical Walk at $T=8$.

- **Classical Random Walk (≈0.2227):** The classical walker, having spread out, has a probability slightly above the steady-state uniform value of 0.20 (or 1/5).
- **Quantum Walk (≈ 0.1328):** The QW probability is significantly **lower** than the CRW. This demonstrates the effect of **destructive quantum interference**. At T=8, the wave function's constructive components have moved past the target or reinforced another vertex, causing the probability at the target to be suppressed.

5. Instructions to run the code:

Run the script with "**python graph_walks.py**". The results for the success probabilities will be printed directly to the console.

6. Conclusion:

The QW's advantage lies in its O(L) *time complexity* for search, not its probability at every arbitrary time step. The low probability at T=8 confirms that on a cycle, the QW exhibits a periodic, oscillatory behavior, meaning a successful search requires stopping the walk at an **optimal interference time** (which is not T=8).

PROBLEM 7: QUANTUM APPROXIMATION OPTIMIZATION ALGORITHM (QAOA)

1. Objective:

The objective was to simulate the performance of the Quantum Approximate Optimization Algorithm (QAOA) on the **3-node MaxCut graph** (a triangle graph) at a fixed depth of $\mathbf{p = 1}$. Specifically, the goal was to observe and quantify how the expected cut value $\langle C \rangle$ degrades as a function of the **Amplitude Damping noise probability (p_{AD})**.

2. Theoretical Background:

A. The MaxCut Problem

The goal of MaxCut is to partition the vertices of a graph into two sets such that the number of edges connecting vertices in different sets (the "cut") is maximized. For the 3-node graph (a triangle), the maximum possible cut value is $\mathbf{3.0}$.

B. QAOA

QAOA is a hybrid quantum-classical algorithm designed to find approximate solutions to combinatorial optimization problems. It uses a sequence of two types of unitary operators, applied p times:

- **Cost Unitary ($U_C(\gamma)$):** Implements the problem's objective function.
- **Mixer Unitary ($U_B(\beta)$):** Introduces superposition and entanglement.

C. Amplitude Damping Noise

Amplitude Damping models energy dissipation in a quantum system, specifically the spontaneous decay of an excited state $|\mathbf{1}\rangle$ to the ground state $|\mathbf{0}\rangle$. It is described by two Kraus operators:

$$\mathcal{E}(\rho) = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger$$

Where:

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p_{AD}} \end{pmatrix}, \quad K_1 = \begin{pmatrix} 0 & \sqrt{p_{AD}} \\ 0 & 0 \end{pmatrix}$$

The simulation uses the **density matrix (ρ) formalism** to track the state's evolution and the application of this noise channel after every unitary gate layer.