**Problem Name: Mirror-Zip Palindrome Matrix**

**Problem Statement:**

You are given an odd integer N (1 ≤ N ≤ 99). You need to generate an N x N matrix pattern that follows a special magical property called the **Mirror-Zip Palindrome**. The matrix must follow these rules:

1. Each row must form a palindrome.
2. The columns are formed by zipping characters from the two halves of the top row.
3. The entire matrix must be symmetric both horizontally and vertically.
4. The central row and central column must contain a zigzag alphabet pattern starting from 'A' outward and reaching 'Z' at the center.

**Brute Force:**

* Construct the matrix using nested loops for each cell (i, j).
* For the left half of the matrix (including the middle), calculate the minimum distance from center.
* Assign characters using 'A' + layer, where layer is the distance from the outer boundary.
* Mirror the left half onto the right to maintain palindrome property.
* The center cell (mid, mid) must be 'Z'.
* Overwrite center row and column with the outward zigzag alphabet patter

**Time Complexity:**

**O(N²)** — Traverse each cell of the matrix once.

**Space Complexity:**

**O(N²)** — To store the full matrix.

**Implementation:**

 Every row is a **palindrome**.

 The **middle column and row** (3rd row and 3rd column for N=5) have characters that follow an outward-in **zigzag alphabet pattern**, peaking at the center with **‘Z’**.

 The matrix shows **mirror symmetry** both horizontally and vertically.

 Each row is formed as:

* Left half → Center → Reverse of left half

 Characters are assigned using the **alphabet starting from ‘A’**, increasing alphabetically layer by layer.

**Code:**

public class MirrorZipBruteForce {

public static void printMirrorZipMatrix(int N) {

if (N % 2 == 0 || N < 1 || N > 99) {

System.out.println("N must be an odd integer between 1 and 99.");

return;

}

char[][] mat = new char[N][N];

int mid = N / 2;

// Fill rows with palindromic layers

for (int i = 0; i < N; i++) {

for (int j = 0; j <= mid; j++) {

int layer = Math.min(Math.min(i, j), Math.min(N - 1 - i, N - 1 - j));

char ch = (char) ('A' + layer);

mat[i][j] = ch;

mat[i][N - 1 - j] = ch; // mirror

}

}

// Overwrite center row and column for 'Z' logic

for (int i = 0; i < N; i++) {

int dist = Math.abs(mid - i);

mat[i][mid] = (i == mid) ? 'Z' : (char) ('A' + dist);

}

for (int j = 0; j < N; j++) {

int dist = Math.abs(mid - j);

mat[mid][j] = (j == mid) ? 'Z' : (char) ('A' + dist);

}

// Print result

for (char[] row : mat) {

for (char c : row) {

System.out.print(c + " ");

}

System.out.println();

}

}

public static void main(String[] args) {

int N = 7;

printMirrorZipMatrix(N);

}

}

**Optimized Solution:**

**Explanation:**

* Use direct computation instead of creating and storing the full matrix.
* For each cell (i, j):
  + If it's the center, print 'Z'.
  + If it's on the center row or column, use distance from the center for zigzag.
  + Else, calculate the layer using Math.min(i, j, N-1-i, N-1-j).

**Time Complexity:**

**O(N²)** — One pass over all cells.

**Space Complexity:**

**O(1)** — Only print output; no matrix storage.

**Code:**

public class MirrorZipOptimized {

public static void printMirrorZipMatrix(int N) {

if (N % 2 == 0 || N < 1 || N > 99) {

System.out.println("N must be an odd integer between 1 and 99.");

return;

}

int mid = N / 2;

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if (i == mid && j == mid) {

System.out.print("Z ");

} else {

int minRow = Math.min(i, N - 1 - i);

int minCol = Math.min(j, N - 1 - j);

int layer = Math.min(minRow, minCol);

char ch;

if (i == mid) {

ch = (char) ('A' + Math.abs(j - mid));

} else if (j == mid) {

ch = (char) ('A' + Math.abs(i - mid));

} else {

ch = (char) ('A' + layer);

}

System.out.print(ch + " ");

}

}

System.out.println();

}

}

public static void main(String[] args) {

int N = 7;

printMirrorZipMatrix(N);

}

}