Análisis del Problema y como lo resolvi: Wondersquare

Función **distanceCell(p0, p1)**

Primero, implementé una función para calcular la distancia entre dos puntos en mi matriz. En esta función, decidí utilizar la distancia entre dos puntos porque necesitaba saber cuántos "pasos" se requieren para ir de un punto a otro. Recibo dos puntos como tuplas y calculo el máximo entre las diferencias absolutas de sus coordenadas x e y.

Función **wondersquare(n)**

Esta es la función principal donde creo la matriz del wondersquare:
Para resolver esto, primero calculé que necesitaba una matriz de tamaño 2n-1 para que quedara cuadrada y el centro estuviera en medio. Ubiqué el centro en (n-1, n-1) y luego construí la matriz calculando la distancia de cada posición al centro, sumándole 1 para que empezara desde 1 en lugar de 0.

printMat(mat)

Para mostrar la matriz par que se vea alineada, creé esta función de impresión:

Converti todos los números a striings y unirlos sin espacios para que se vierabien.

pregunta()

Esta es prácticamente la función principal que maneja la interacción con la persona:

Aquí me aseguro de validar que el número ingresado sea válido (mayor o igual a 1) antes de proceder a crear e imprimir el wondersquare.

2. Modelo de Caja Negra

Para estructurar el problema o la solución, identifiqué:

Entradas: Un número n que el usuario ingresa (debe ser mayor o igual a 1)

Proceso: En mi solución, el proceso sigue estos pasos:

1. Verifico que el número sea válido.
2. Calculo el tamaño de la matriz como 2n-.
3. Genero la matriz calculando las distancias desde cada punto al centro.
4. Formateo la matriz para su impresión.

Salidas: Mi programa puede producir dos tipos de salidas:

- Si el usuario ingresa un número menor a 1, muestro un mensaje de error
- Si el número es válido, imprimo la matriz del wondersquare donde:

- El centro tiene el número 1
- Los números aumentan en 1 mientras más lejos estén del centr
- Los números se muestran juntos sin espacios

Análisis: Palíndromos

1. Análisis del Problema y Mi Solución

Para resolver este problema, use una función que verifica si una cadena es un palíndromo utilizando dos índices que se mueven desde los extremos hacia el otro extremo, uno desde el la izquierda haasta la derecha y uno de la derechha hasta la izquierda:

Mi solución funciona de la siguiente manera:

1. Creo dos variables inicio y fin:
    - inicio empieza en la primera posición (índice 0).
    - fin empieza en la última posición (longitud - 1), ya que la primera posicion estaria en el indice 0 y la ultima posición estaría en el indice longuitud – 1.
2. Utilizo un ciclo while que continúa mientras inicio sea menor que fin. En cada iteración:
    - Comparo los caracteres en las posiciones inicio y fin.
    - Si son diferentes, retorno False porque no es un palíndromo.
    - Si son iguales, muevo el índice inicio una posición hacia la derecha y fin una posición hacia la izquierda.
3. Si el ciclo termina sin encontrar caracteres diferentes, significa que la palabra es un palíndromo y retorno True, de lo contrario retorno false si se cumple la condicion de que inicio es diferente de fin.

Por ejemplo, para la palabra "anitalavalatina":

- Primera iteración: comparo 'a' con 'a'.
- Segunda iteración: comparo 'n' con 'n'.
- Y así sucesivamente hasta confirmar que todos los caracteres coinciden.

2. Modelo de Caja Negra

Entradas

- Una cadena de texto s (string).

Proceso

1. Inicialización de índices en los extremos de la cadena.

2. Comparación de caracteres desde los extremos hacia el centro.
3. Movimiento de índices hacia el centro si los caracteres coinciden.

Salidas

- True: si la cadena es un palíndromo.
- False: si la cadena no es un palíndromo.

Análisis: MCD

1. Para resolver este problema, utilicé una sola función (MCD). La solución busca los divisores comunes más grandes de ambos números usando una iteración desde 2 hasta el menor de los dos números.

Mi solución funciona de la siguiente manera:

- Defino una función mcd(a, b) que toma dos números enteros positivos como entrada.
- Inicializo una variable mcd_resultado en 1, ya que el MCD mínimo posible es 1.
- Utilizo unn for que recorre los números desde 2 hasta el menor de los dos números (min(a, b) + 1).

En cada iteración:

- Verifico si div divide exactamente a a y b.
- Si es así, actualizo mcd_resultado con div, manteniendo el mayor divisor encontrado.
- Al finalizar el ciclo, retorno mcd_resultado como el MCD de a y b.

Por ejemplo, para los números (6, 12):

- Inicio con mcd_resultado = 1.
- Itero probando divisores comunes: 2, 3 y 6.
- Como 6 es el mayor divisor común, la función retorna 6.

2. Modelo de Caja Negra

Entradas

- Dos números enteros positivos a y b.

Proceso

- Inicializacióon de mcd_resultado en 1.
- Iteración desde 2 hasta min(a, b).
- Verificación de divisibilidad de a y b.
- Actualización del mayor divisor común encontrado.

Salidas

- Un número entero que representa el Máximo Común Divisor (MCD) de a y b.