

### **URL VIDEOS**

Explicación enunciado del problema: <https://youtu.be/DK7QwqLvy0E>

Explicación código: <https://youtu.be/WIUx0NuMRI0>

Análisis de complejidad: <https://youtu.be/-vDHKVk5lzo>

### **ENUNCIADO DEL PROBLEMA**

Policarpa tiene una secuencia favorita, la escribió en una pizarra de la siguiente manera:

Escribió el primer número a la izquierda.

El segundo número a la derecha.

El tercero lo más a la izquierda posible (pero después del primero).

El cuarto lo más a la derecha posible (pero antes del segundo).

Repite este proceso hasta escribir toda la secuencia.

Dada la secuencia final en la pizarra, reconstruye la secuencia original a.

Entrada:

$t (1 \leq t \leq 300)$  → número de casos de prueba.

Para cada caso de prueba:

$n (1 \leq n \leq 300)$  → longitud de la secuencia.

$b_1, b_2, \dots, b_n (1 \leq b_i \leq 10^9)$  → secuencia escrita en la pizarra.

**Salida:**  
Secuencia a que Policarpa escribió en la pizarra.

**URL DE GITLAB O GITHUB PROYECTO EN FORMA PRIVADA, USUARIO [madarme@ufps.edu.co](mailto:madarme@ufps.edu.co), con rol mantener**

[https://gitlab.com/ejercicios-proyecto-ada/ejercicio-3-secuencia-favorita/-/tree/main/FavoriteSequence?ref\\_type=heads](https://gitlab.com/ejercicios-proyecto-ada/ejercicio-3-secuencia-favorita/-/tree/main/FavoriteSequence?ref_type=heads)

## MÉTODO 1

```
public int[] metodoCandido(int n, int[] secuencia) {  
    int[] a = new int[n];  
    int pos = 0;  
  
    while (pos < n) {  
        for (int i = 0; i < n; i++) {  
            if (i == pos) {  
                if (i % 2 == 0) {
```

```
        a[pos] = secuencia[i / 2];
    }
    if (i % 2 == 1) {
        a[pos] = secuencia[n - 1 - (i / 2)];
    }
}
pos++;
}

return a;
}
```

El método es eficaz, puesto a que tiene una solución para todas las entradas válidas ingresadas.

**Eficacia**

Instancia 1:

TAMAÑO SECUENCIA: 11  
SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}

	<p>Resultado de la invocación: 8 2 4 4 3 9 1 7 2 8 7</p> <p>Instancia 2:</p> <p>TAMAÑO SECUENCIA: 7 SECUENCIA: {3, 4, 5, 2, 9, 1, 1}</p> <p>Resultado de la invocación: 3 1 4 1 5 9 2</p>
	<p>Instancia 3:</p> <p>TAMAÑO SECUENCIA: 48 SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}</p> <p>Resultado de la invocación: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7</p>
<b>Eficiencia</b>	Argumento (Escriba su argumento basado en el punto las instancias de abajo):

**(debe escribir tiempo que demora su solución) para cada instancia. Vea el ejemplo del siguiente ejercicio.**  
**Usé la clase StopWatch.**

El método no es el más eficiente ya que recorre el arreglo múltiples veces cuando no es necesario, haciendo que consuma tiempo realizando operaciones de más; esto se ve reflejado en la instancia N°3 donde hay una mayor cantidad de elementos y esta falencia toma mayor peso.

Instancia 1:

TAMAÑO SECUENCIA: 11  
SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}

Resultado de la invocación: **22 microsegundos**

Instancia 2:

TAMAÑO SECUENCIA: 7  
SECUENCIA: {3, 4, 5, 2, 9, 1, 1}

Resultado de la invocación: **10 microsegundos**

Instancia 3:

	TAMAÑO SECUENCIA: 48 SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}
	Resultado de la invocación: <b>180 microsegundos</b>
<b>Correctitud</b>	El algoritmo es correcto, ya que da un resultado esperado para todas las entradas posibles  Instancia 1:  TAMAÑO SECUENCIA: 11 SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}  RESPUESTA ESPERADA: 8 2 4 4 3 9 1 7 2 8 7 RESPUESTA MÉTODO CÁNDIDO: 8 2 4 4 3 9 1 7 2 8 7
	Instancia 2:  TAMAÑO SECUENCIA: 7

	<p>SECUENCIA: {3, 4, 5, 2, 9, 1, 1}</p> <p>RESPUESTA ESPERADA: 3 1 4 1 5 9 2</p> <p>RESPUESTA MÉTODO CÁNDIDO: 3 1 4 1 5 9 2</p>
	<p>Instancia 3:</p> <p>TAMAÑO SECUENCIA: 48</p> <p>SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}</p> <p>RESPUESTA ESPERADA: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7</p> <p>RESPUESTA MÉTODO CÁNDIDO: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7</p>
<b>Completitud</b>	<p>El método es completo, ya que la lógica usada para hallar la secuencia escrita, aunque es lento asegura siempre un resultado correcto, si es que este existe</p>

Instancia 1:

TAMAÑO SECUENCIA: 11

SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}

RESPUESTA ESPERADA: 8 2 4 4 3 9 1 7 2 8 7

RESPUESTA MÉTODO CÁNDIDO: 8 2 4 4 3 9 1 7 2 8 7

Instancia 2:

TAMAÑO SECUENCIA: 7

SECUENCIA: {3, 4, 5, 2, 9, 1, 1}

RESPUESTA ESPERADA: 3 1 4 1 5 9 2

RESPUESTA MÉTODO CÁNDIDO: 3 1 4 1 5 9 2

Resultado de la invocación:

Instancia 3:

TAMAÑO SECUENCIA: 48

SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2,

323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}

RESPUESTA ESPERADA: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87  
3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7

RESPUESTA MÉTODO CÁNDIDO: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7  
32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7

## MÉTODO 2

```
public int[] metodoOptimo(int n, int[] secuencia) {  
    int[] a = new int[n];  
    int pos = 0, l = 0, r = n - 1;  
  
    while (l <= r) {  
        a[pos] = secuencia[l];  
        if (pos + 1 < n) {  
            a[pos + 1] = secuencia[r];  
        }  
        pos += 2;  
        l++;  
        r--;  
    }  
    return a;  
}
```

<b>Eficacia</b>	El método es eficaz, puesto a que tiene una solución para todas las entradas válidas ingresadas
-----------------	---

	<p>Instancia 1:</p> <p>TAMAÑO SECUENCIA: 11 SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}</p> <p>Resultado de la invocación: 8 2 4 4 3 9 1 7 2 8 7</p>
	<p>Instancia 2:</p> <p>TAMAÑO SECUENCIA: 7 SECUENCIA: {3, 4, 5, 2, 9, 1, 1}</p> <p>Resultado de la invocación: 3 1 4 1 5 9 2</p>
	<p>Instancia 3:</p> <p>TAMAÑO SECUENCIA: 48 SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}</p>

	Resultado de la invocación: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7
<b>Eficiencia (debe escribir tiempo que demora su solución) para cada instancia. Vea el ejemplo del siguiente ejercicio. Usé la clase StopWatch.</b>	El método es eficiente ya que solo necesita de un recorrido para llegar a un resultado correcto, evitando las múltiples validaciones de los mismo valores
	Instancia 1:  TAMAÑO SECUENCIA: 11 SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}  Resultado de la invocación: <b>8 microsegundos</b>
	Instancia 2:  TAMAÑO SECUENCIA: 7 SECUENCIA: {3, 4, 5, 2, 9, 1, 1}

	<p>Resultado de la invocación: <b>6 microsegundos</b></p> <p>Instancia 3:</p> <p>TAMAÑO SECUENCIA: 48 SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}</p> <p>Resultado de la invocación: <b>10 microsegundos</b></p>
<b>Correctitud</b>	<p>El método es completo, ya que la lógica usada para hallar la secuencia escrita, es eficiente y asegura siempre un resultado correcto, si es que este existe</p> <p>Instancia 1:</p> <p>TAMAÑO SECUENCIA: 11 SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}</p> <p>RESPUESTA ESPERADA: 8 2 4 4 3 9 1 7 2 8 7</p>

RESPUESTA MÉTODO ÓPTIMO: 8 2 4 4 3 9 1 7 2 8 7

Instancia 2:

TAMAÑO SECUENCIA: 7

SECUENCIA: {3, 4, 5, 2, 9, 1, 1}

RESPUESTA ESPERADA: 3 1 4 1 5 9 2

RESPUESTA MÉTODO ÓPTIMO: 3 1 4 1 5 9 2

Instancia 3:

TAMAÑO SECUENCIA: 48

SECUENCIA: {9, 2, 7, 1, 5, 3, 2, 3, 4, 5, 6, 7, 87, 8, 97, 4, 2, 1, 34, 5, 677, 8, 9, 9, 7, 55, 3, 2, 323, 323, 234, 45, 7678, 8766, 34, 3434, 32, 21, 33, 3, 5, 6, 7, 8, 9, 12, 12, 45}

RESPUESTA ESPERADA: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7

RESPUESTA MÉTODO ÓPTIMO: 9 45 2 12 7 12 1 9 5 8 3 7 2 6 3 5 4 3 5 33 6 21 7 32 87 3434 8 34 97 8766 4 7678 2 45 1 234 34 323 5 323 677 2 8 3 9 55 9 7

<b>Completitud</b>	<p>El algoritmo es completo, ya que da un resultado esperado para todas las entradas posibles</p>
	<p>Instancia 1:</p> <p>TAMAÑO SECUENCIA: 11 SECUENCIA: {8, 4, 3, 1, 2, 7, 8, 7, 9, 4, 2}</p> <p>RESPUESTA ESPERADA: 8 2 4 4 3 9 1 7 2 8 7 RESPUESTA MÉTODO ÓPTIMO: 8 2 4 4 3 9 1 7 2 8 7</p>
	<p>Instancia 2:</p> <p>TAMAÑO SECUENCIA: 7 SECUENCIA: {3, 4, 5, 2, 9, 1, 1}</p> <p>RESPUESTA ESPERADA: 3 1 4 1 5 9 2 RESPUESTA MÉTODO ÓPTIMO: 3 1 4 1 5 9 2</p>