Explicación del Código: Walk on a multiplication table

En este documento se explica cómo resolver el problema “Walk on Multiplication Table” mediante dos enfoques:

1. Método Cándido
2. Método Óptimo

Ambos métodos están implementados en Java y reciben como parámetro un número long n, el valor que Takashi desea encontrar dentro de la tabla de multiplicación infinita. El objetivo de ambos algoritmos es calcular la mínima cantidad de movimientos necesarios para llegar a una celda cuyo valor sea n, recordando que Takashi solo puede moverse a la derecha o hacia abajo, comenzando desde la posición (1,1).

1. Método Cándido (Ingenuo)

Idea general

Este método simula literalmente la búsqueda en la tabla de multiplicación. Recorre todas las posibles combinaciones de filas (i) y columnas (j) desde 1 hasta n, verificando si la multiplicación i * j produce el valor n.

Aunque este enfoque es correcto, es extremadamente costoso porque revisa todas las celdas de un cuadrado n × n, lo cual implica una complejidad de $O(n^2)$.

Pasos principales del método

1. Se inicializa una variable minMoves con el valor máximo posible (Long.MAX_VALUE), donde se almacenará la cantidad mínima de movimientos encontrada.
2. Se utilizan dos ciclos for anidados:
    - El primero recorre las filas i desde 1 hasta n.
    - El segundo recorre las columnas j desde 1 hasta n.
3. En cada celda (i, j) se calcula el producto i * j.
    - Si el resultado es igual a n, significa que Takashi puede llegar a esta casilla para obtener el valor deseado.
4. Una vez encontrada una casilla válida, se calculan los movimientos necesarios:
    - Desde la posición inicial (1,1), llegar a (i, j) implica:
    - movimientos = (i - 1) + (j - 1)
    - Esto representa el número de movimientos hacia abajo y hacia la derecha.
5. Si la cantidad de movimientos hallada es menor que la almacenada en minMoves, se actualiza.
6. Al finalizar los ciclos, se retorna minMoves.

Resultado

El método funciona correctamente para valores pequeños, pero su rendimiento se degrada rápidamente para entradas grandes debido al número de iteraciones.

2. Método Óptimo

Idea general

La clave para optimizar el problema está en notar que si i * j = n, entonces ambos valores son divisores de n. Esto significa que no necesitamos recorrer toda la tabla, sino únicamente buscar los divisores de n. Además, los divisores pueden encontrarse recorriendo solamente desde 1 hasta $\sqrt{n}$, reduciendo la complejidad a $O(\sqrt{n})$, lo cual es muchísimo más rápido.

Pasos del método

1. Se crea una variable mayor, que almacenará el divisor más grande encontrado en la iteración.
2. Se recorre un ciclo for desde 1 hasta $\sqrt{n}$.
3. Para cada número i:
    - Si n % i == 0, entonces i es divisor y n / i también.
    - Se guarda i como último divisor válido encontrado.
4. Una vez finalizado el ciclo, el divisor más grande encontrado será el valor i que maximiza la eficiencia al moverse en la tabla.
5. Con ese divisor mayor, se calcula el número de movimientos usando la fórmula:
6. movimientos = (mayor - 1) + (n / mayor - 1)

O equivalente:

movimientos = mayor + (n / mayor) - 2

7. Finalmente, se retorna el número mínimo de movimientos.

Resultado

El método es:

- Más rápido
- Más eficiente para cualquier valor grande de n
- Igual de exacto que el método cándido

3. Comparación de rendimiento

Para comparar ambos métodos, se utilizaron varias instancias y la clase Stopwatch para medir el tiempo de ejecución:

Instancia 1: n = 10

- Método cándido: 142 microsegundos
- Método óptimo: 36 microsegundos
  Ambos retornan el valor correcto: 5 movimientos.

Instancia 2: n = 9146

- Método cándido: 62,771 microsegundos
- Método óptimo: 25 microsegundos
  Ambos producen la misma respuesta: 301 movimientos.

Instancia 3: valor long muy grande

- El método cándido no logra producir una respuesta.
- El método óptimo sí devuelve el resultado en un tiempo conveniente.

Conclusión

El análisis demuestra que:

- El método cándido es correcto, pero completamente ineficiente para valores medianos y grandes debido a su naturaleza $O(n^2)$.
- El método óptimo se basa en propiedades matemáticas del problema (los divisores de n) y reduce enormemente el trabajo requerido.
- Para cualquier aplicación práctica o valores grandes de n, el método óptimo es la única alternativa viable.