COMPLEJIDAD CICLOMÁTICA DE UN ALGORITMO O MÉTODO DE LOS CAMINOS BÁSICOS

1. COMPLEJIDAD CICLOMÁTICA. DEFINICIÓN.

La Complejidad Ciclomática (CC) determina el número de caminos básicos de un algoritmo.

Pero, ¿qué es un Camino Básico?

Un camino básico es el recorrido que podemos seguir desde un nodo inicial a un nodo final y que no contenga otros subcaminos.

2. ¿CÓMO SE CALCULA LA COMPLEJIDAD CICLOMÁTICA?

Vamos a tomar como ejemplo un fragmento de código que realiza una ordenación ascendente de un vector de 7 número enteros.

```
13  public static void main (String arg[ ])
14  {
15     int [ ] vector = new int [7];
16     int aux;
17
18     for (int i=0; i< vector.length-1; i++)
19     {
20        for (int j=0;j<vector.length-1; j++)
21        {
22           if (vector[j] < vector[j+1])
23           {
24              aux=vector[j];
25              vector[j]=vector[j+1];
26              vector[j+1]=aux;
27           } // if
28        } // for j
29     } // for i
30
31  } // main
```

Para aplicar el método no es necesario conocer en profundidad el código, tan solo es necesario conocer la sintaxis del lenguaje.

La aplicación del método se basa en **3 pasos**:

1. Obtener a partir del código el **grafo** que representa al programa.
2. Determinar a partir del grafo la complejidad ciclomática del algoritmo, o lo que es lo mismo, los **caminos básicos**.
3. Determinar cuáles son esos caminos básicos para luego generar los **juegos de prueba** que permitan que fuercen la ejecución del programa mediante el recorrido de esos caminos básicos.

Veamos ahora como pasar del código al grafo que representa al código.

1. **Numeramos todas y cada una de las instrucciones de nuestro programa**.

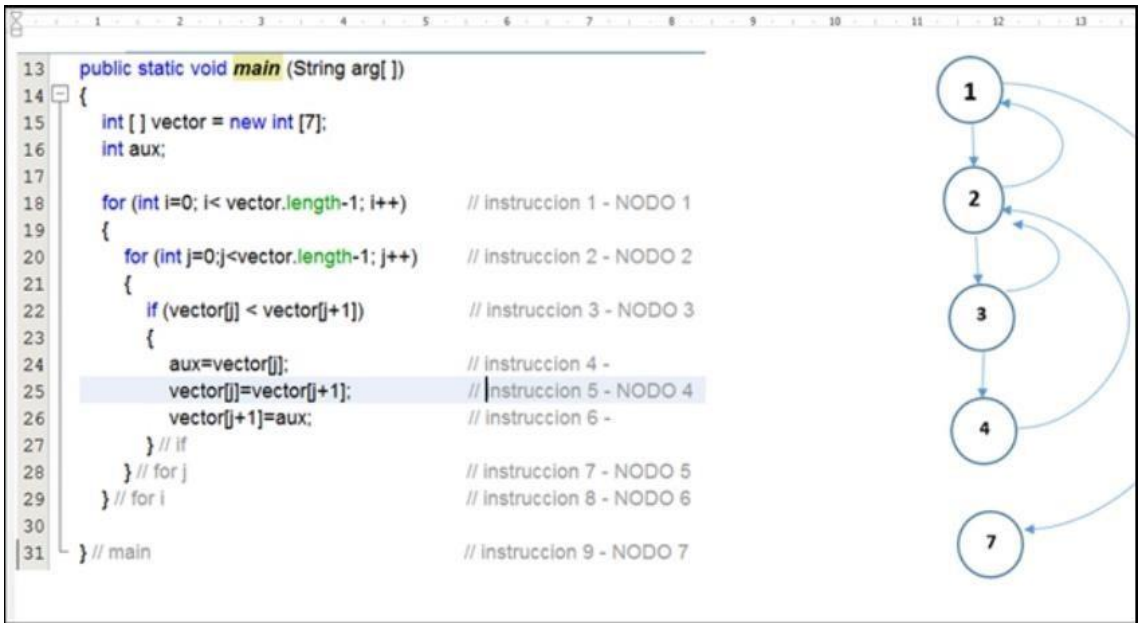
   Podemos tomar como referencia los números de líneas que nos da el propio IDE, pero por simplicidad vamos a asignar nosotros mismos una numeración a las instrucciones. ¿Cómo? Dando un número a aquellas instrucciones que se ejecutan; no entran aquí, por ejemplo, las declaraciones de variables, por no ser relevantes para lo que queremos. Sí nos interesan las instrucciones con un FOR, IF, WHILE, SWITCH..., así como las instrucciones que puedan llevar dentro.

   Una vez localizada la primera instrucción, le asignamos el número 1. Así con las demás instrucciones que tengamos. Si hay un grupo de instrucciones que van en bloque, por ejemplo, 4 instrucciones dentro de un IF, podemos asignarles un único número.

   Numeramos también las que cierran bucles FOR, IF, WHILE, SWITCH, etc. No hay problema en asignar más números de los que sean necesarios, ya que al construir el grafo nos saldrán los que necesitemos.

```
13  public static void main (String arg[ ])
14  {
15     int [ ] vector = new int [7];
16     int aux;
17
18     for (int i=0; i< vector.length-1; i++)        // instruccion 1 - NODO 1
19     {
20        for (int j=0;j<vector.length-1; j++)       // instruccion 2 - NODO 2
21        {
22           if (vector[j] < vector[j+1])            // instruccion 3 - NODO 3
23           {
24              aux=vector[j];                       // instruccion 4 -
25              vector[j]=vector[j+1];               // instruccion 5 - NODO 4
26              vector[j+1]=aux;                     // instruccion 6 -
27           } // if
28        } // for j                                 // instruccion 7 - NODO 5
29     } // for i                                    // instruccion 8 - NODO 6
30
31  } // main                                        // instruccion 9 - NODO 7
```

2. Una vez numeradas, **repasamos esas numeraciones**, por si nos puede quedar más sencillo. Una vez hecho esto, tendríamos el grafo:



Construcción del grafo correspondiente al fragmento de código.

3. Ahora **calculamos la complejidad ciclomática (CC) del algoritmo**. Para ello hay 3 formas:

a) **CC = ARISTAS - NODOS + 2**

Contamos el número de aristas o flechas del grafo, 7 aristas en el ejemplo, y contamos también el número de nodos, 5 nodos en el ejemplo.

ARISTAS: 7, NODOS: 5 → 7 -5 + 2 = 4

b) **CC = REGIONES CERRADAS + 1**

Regiones Cerradas: partes del grafo que forman un círculo completo (en este caso, 3: alrededor de los nodos 2, 3, y 4)

REGIONES CERRADAS → 3 + 1 = 4

c) Evaluar los predicados simples del programa.

CC = PREDICADOS SIMPLES + 1

Para conocer el número de predicados simples podemos hacerlo de dos formas:

i. Basarnos en los nodos que tienen dos o más salidas, que se corresponden con bifurcaciones. En el ejemplo:
   - Los nodos 1, 2 y 3 tienen dos salidas → son PREDICADOS SIMPLES.
   - El nodo 4 tiene una sola salida → no es PREDICADO SIMPLE
   - El nodo 7 no tienen ninguna salida → no es PREDICADO SIMPLE

ii. Ir directamente al código y determinar el número de condiciones que tenemos escritas. Aquí entrarían los IF, bucles… Si vemos en nuestro ejemplo, el código tiene una condición para el FOR i, otra para el FOR j y otra para el IF (3 en total).

Por tanto:

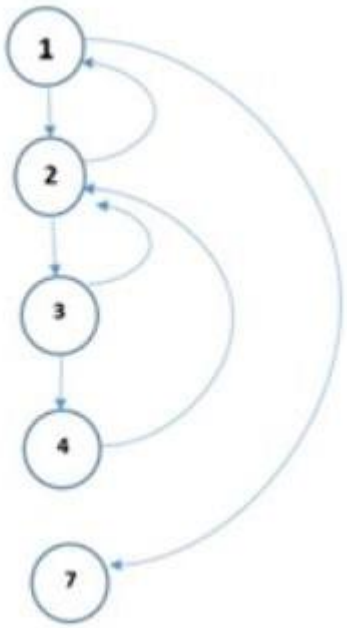
CC = PREDICADOS SIMPLES + 1→ 3 + 1 = 4

La complejidad ciclomática obtenida nos determina el número de caminos básicos distintos que tiene el algoritmo.

En nuestro ejemplo la complejidad ciclomática, CC = 4. Esto significa, que es aceptable y que hay 4 caminos básicos. ¿Cuáles son? Son los siguientes:

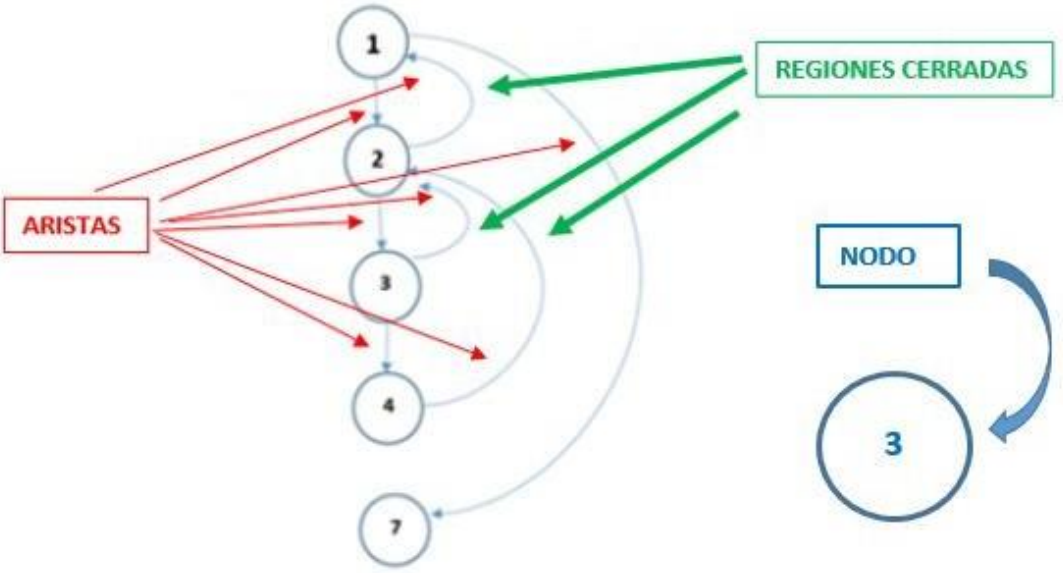
Caminos básicos:

1-7
1-2-1-7
1-2-3-2-1-7
1-2-3-4-2-1-7



A partir de aquí, se determinarían los valores de entrada que hay que pasar al programa para forzar la ejecución de cada uno de estos caminos básicos.

Elementos que intervienen en el grafo:



Predicados Simples según el código: