

# Calidad de Software

Complejidad Ciclomática

# Complejidad Ciclomática

Considere la siguiente pieza de código:

```
Import java.io.*;

Public class Maximo
{
    public static void main (String args[]) throws IOException
    {
        BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));
        Int x,y,z,max;

        System.out.println("Introduce x,y,z: ");
        x = Integer.parseInt (entrada.readLine());
        y = Integer.parseInt (entrada.readLine());
        z = Integer.parseInt (entrada.readLine());

        if (x>y && x>z)
            max = x;
        else
            if (z>y)
                max = z;
            else
                max = y;
        System.out.println ("El máximo es "+ max);
    }
}
```

# Complejidad Ciclomática

Para calcular la complejidad ciclomática, lo primero que tenemos que hacer es dibujar el grafo de flujo.

1. Anotar en el código los pasos del programa
2. Dibujar el grafo
3. Calcular Complejidad Ciclomática
4. Determinar caminos independientes
5. Definir conjuntos de pruebas mínimo para alcanzar criterios de cobertura:
  - Cobertura de sentencias
  - Cobertura de decisiones

# Complejidad Ciclomática

Considere la siguiente pieza de código:

```
Import java.io.*;
```

```
Public class Maximo  
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```

```
    if (x>y && x>z)  
        max = x;
```

```
    else
```

```
        if (z>y)  
            max = z;
```

```
        else
```

```
            max = y;
```

```
    System.out.println ("El máximo es "+ max);
```

```
    }
```

```
}
```

1



# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

```
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```

1

2

```
    if (x>y && x>z)  
        max = x;
```

```
    else
```

```
        if (z>y)
```

```
            max = z;
```

```
        else
```

```
            max = y;
```

```
    System.out.println ("El máximo es "+ max);
```

```
}
```

```
}
```

# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

```
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```

2

```
    → if (x>y && x>z)  
        max = x;
```

3

```
    else
```

```
        if (z>y)
```

```
            max = z;
```

```
        else
```

```
            max = y;
```

```
    System.out.println ("El máximo es "+ max);
```

```
}
```

```
}
```

1

# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

```
{
```

```
    public static void main (String args[]) throws IOException
```

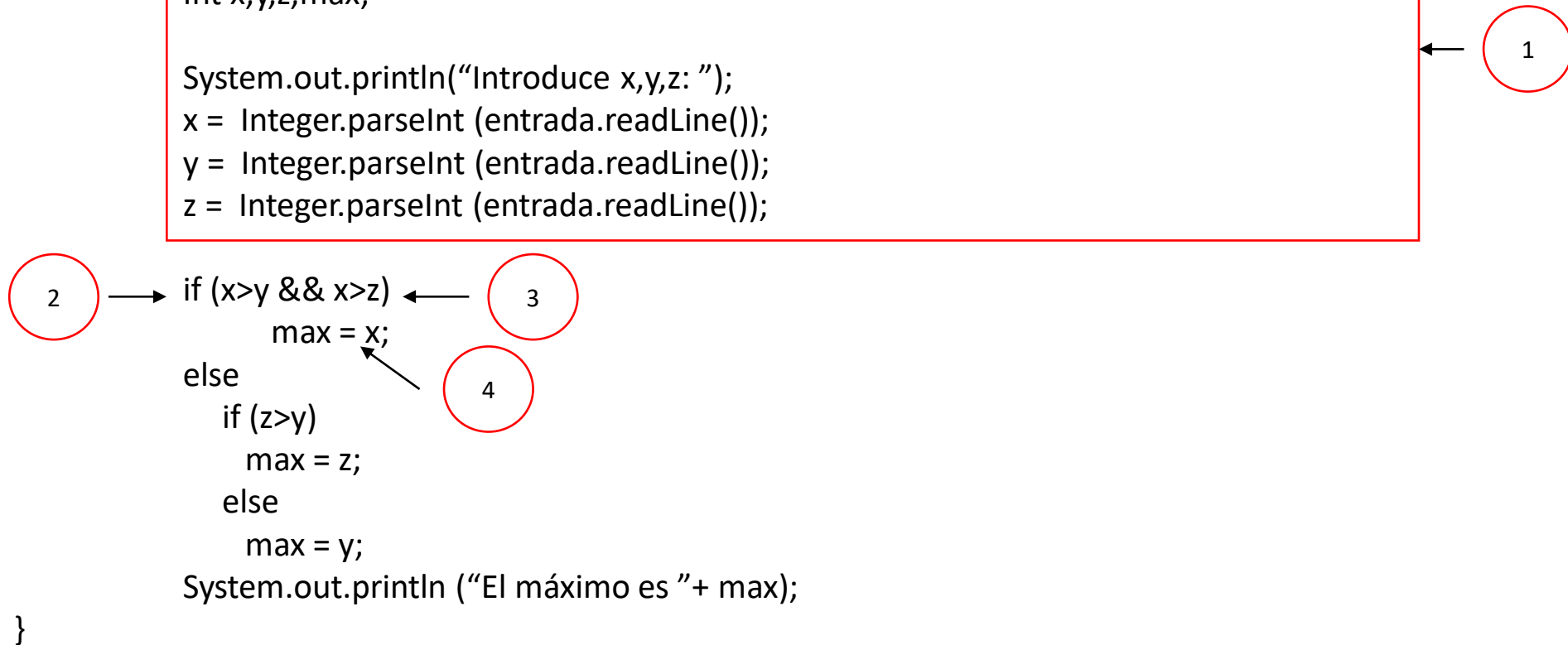
```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```

```
    if (x>y && x>z)   
        max = x;  
    else   
        if (z>y)  
            max = z;  
        else  
            max = y;  
    System.out.println ("El máximo es "+ max);  
}
```

```
}
```



# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

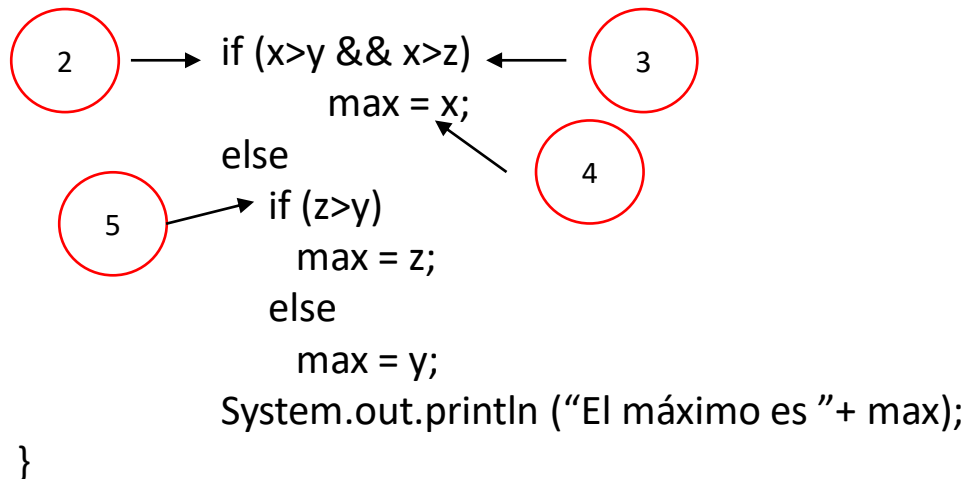
```
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```



```
}
```



# Grafo

```
import java.io.*;
```

## Public class Maximo

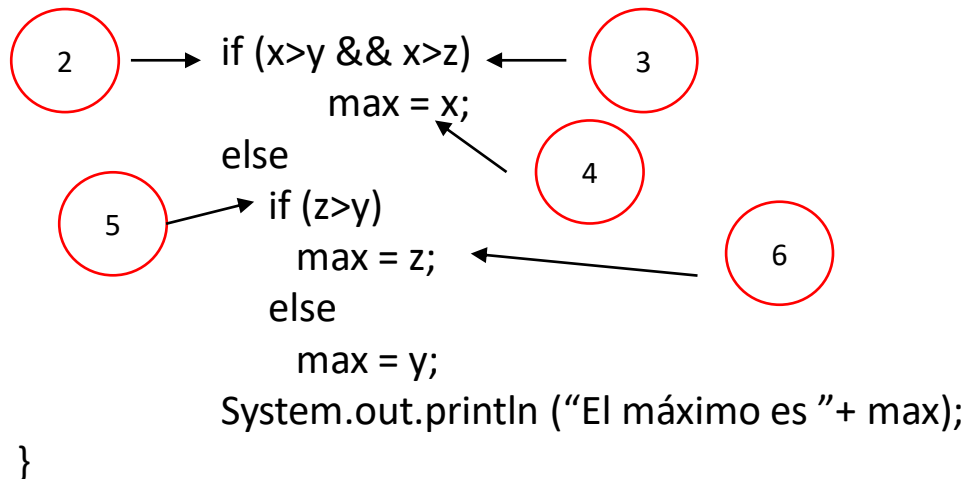
 $\{$ 

```
public static void main (String args[]) throws IOException
```

$$\{$$

```
BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
Int x,y,z,max;
```

```
System.out.println("Introduce x,y,z: ");
x = Integer.parseInt (entrada.readLine());
y = Integer.parseInt (entrada.readLine());
z = Integer.parseInt (entrada.readLine());
```



}

}

# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

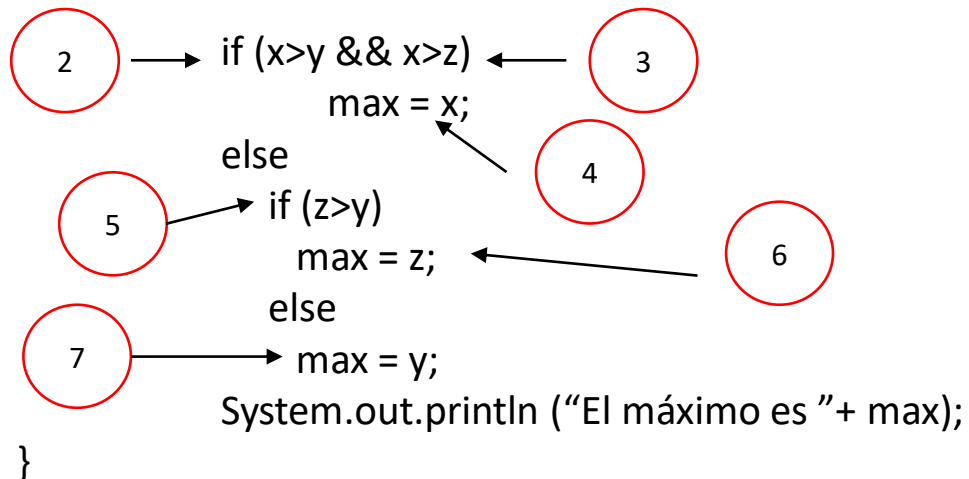
```
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```



```
}
```

# Grafo

```
Import java.io.*;
```

```
Public class Maximo
```

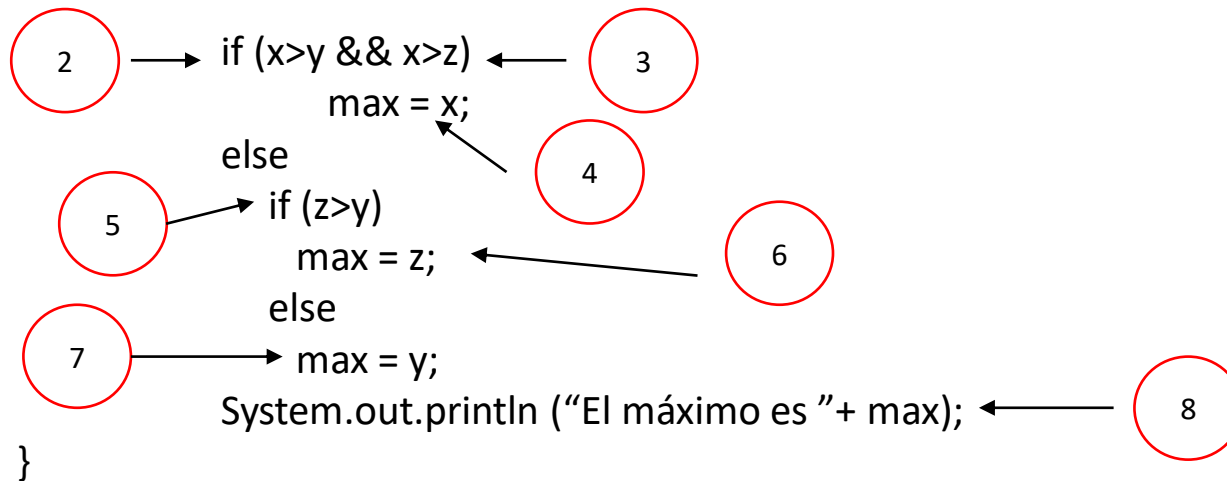
```
{
```

```
    public static void main (String args[]) throws IOException
```

```
{
```

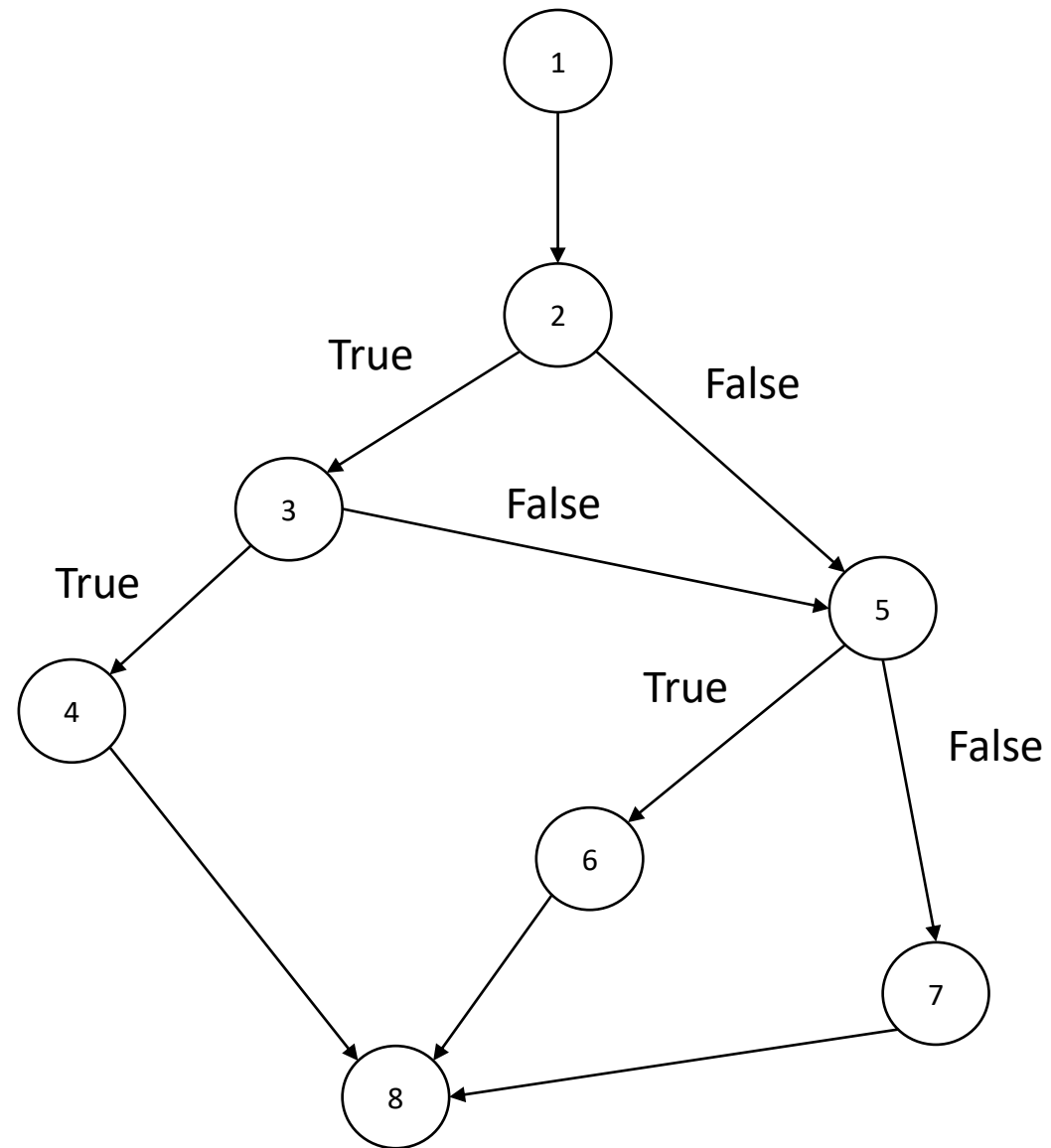
```
    BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));  
    Int x,y,z,max;
```

```
    System.out.println("Introduce x,y,z: ");  
    x = Integer.parseInt (entrada.readLine());  
    y = Integer.parseInt (entrada.readLine());  
    z = Integer.parseInt (entrada.readLine());
```

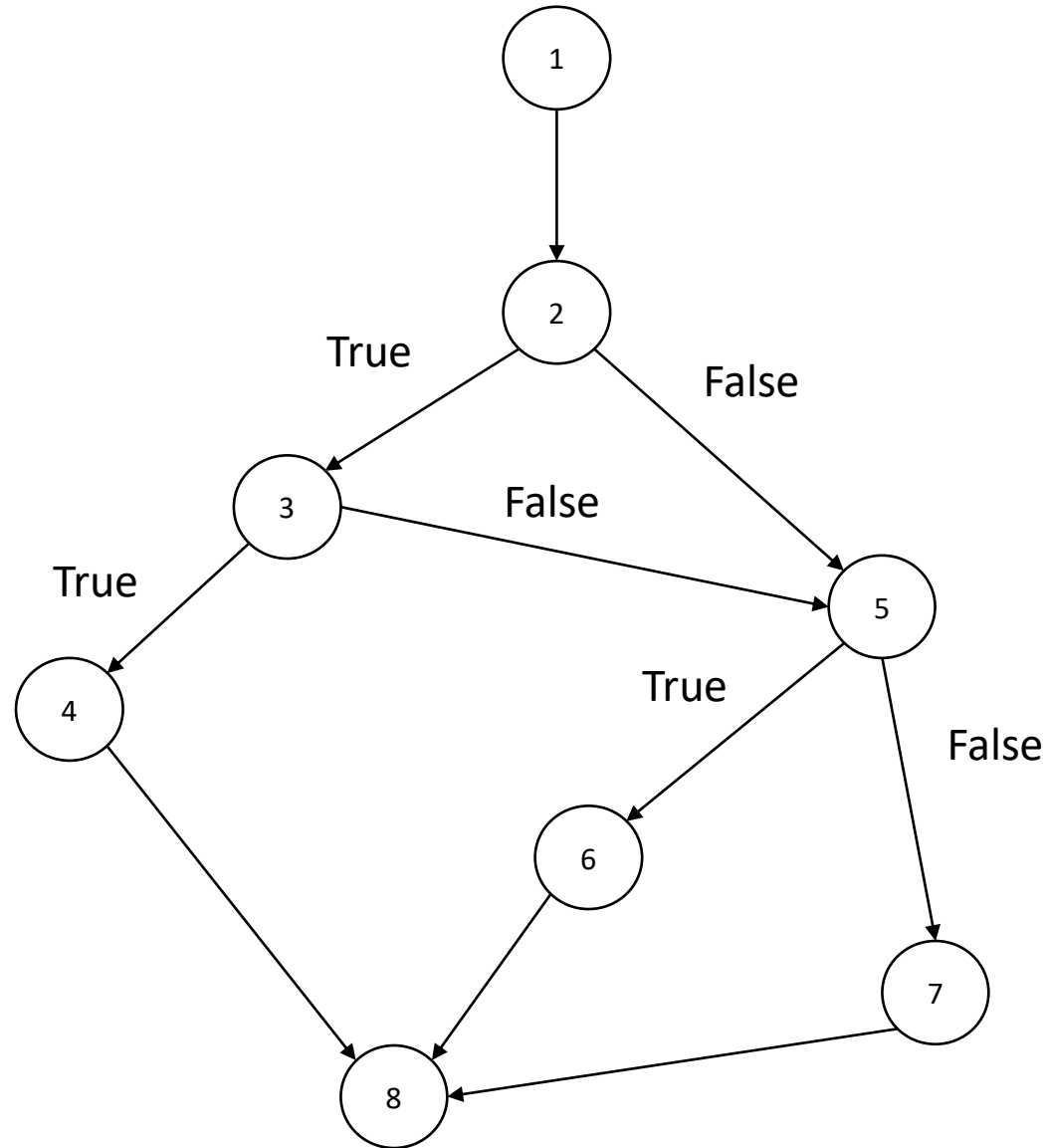


```
}
```

# Grafo



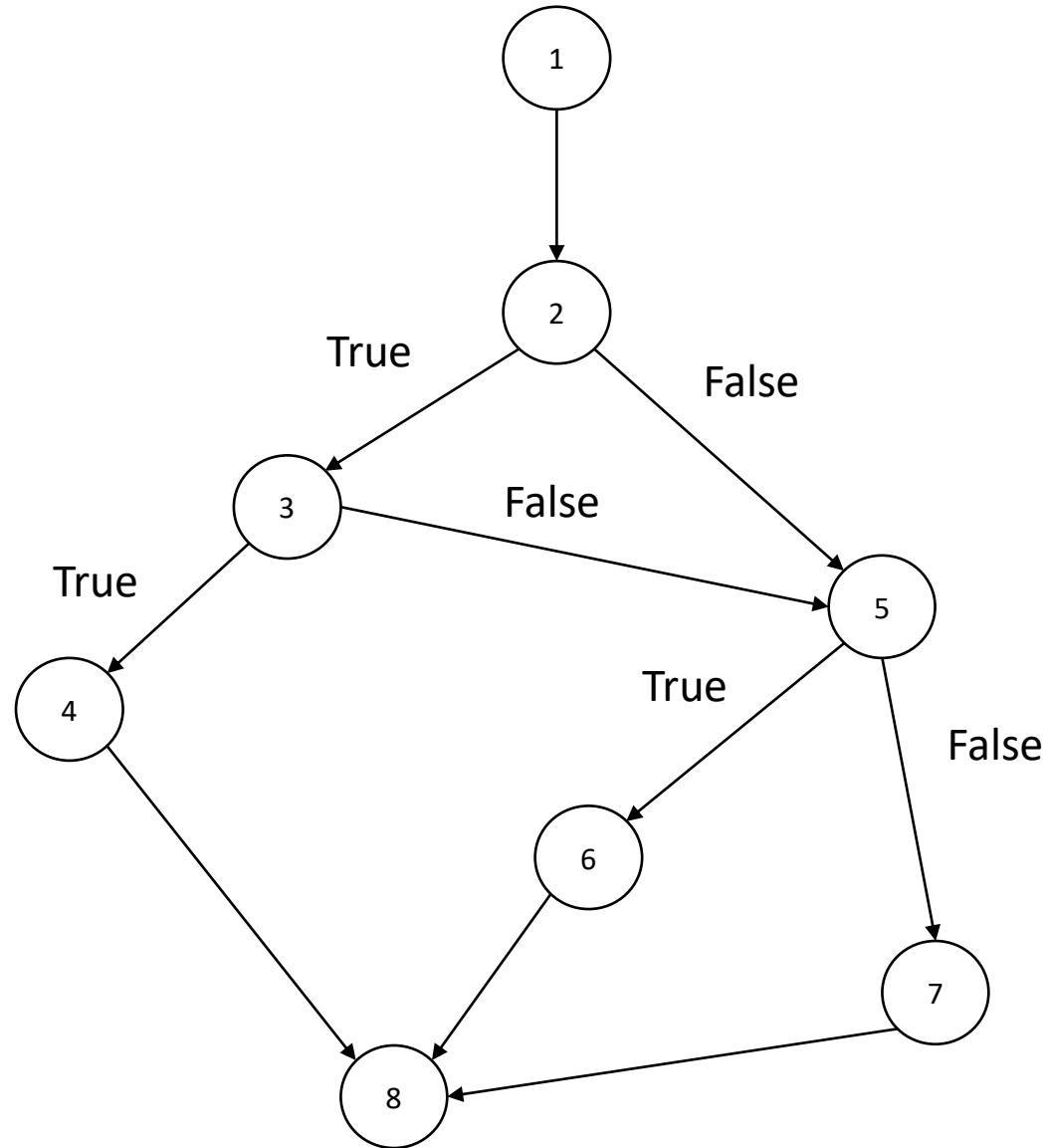
# Grafo



## Calculo de Complejidad

$$\text{Complejidad} = \text{Aristas} - \text{Nodos} + 2$$

# Grafo



## Calculo de Complejidad

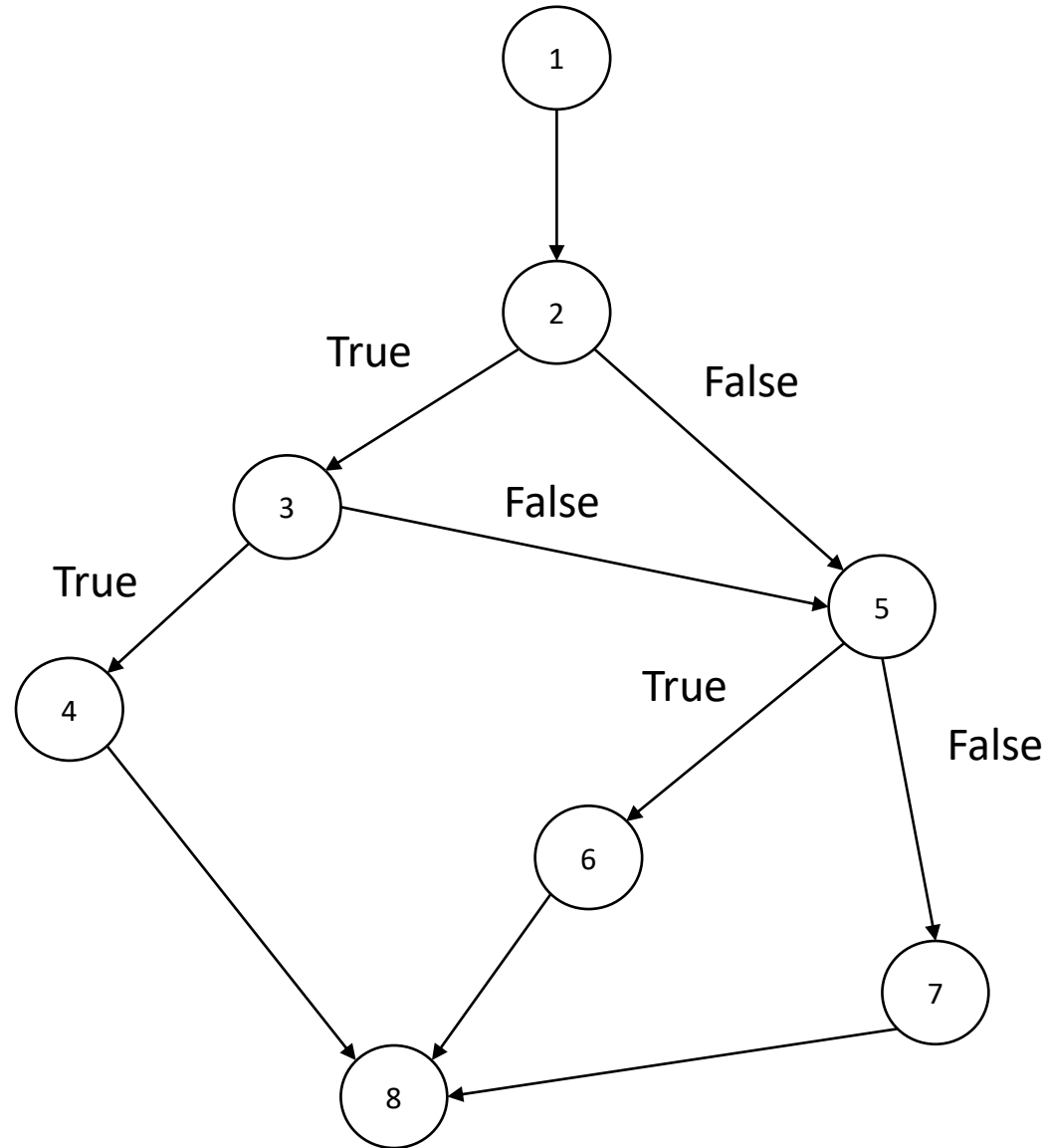
Complejidad = Aristas – Nodos + 2

En nuestro caso =  $10 - 8 + 2 = 4$

Luego

Complejidad = 4

# Grafo



## Calculo de Complejidad

Complejidad = Aristas – Nodos + 2

En nuestro caso =  $10 - 8 + 2 = 4$

Luego

Complejidad = 4

Tendremos cuatro (4) caminos independientes, que, mirando el grafo, son los siguientes:

Camino 1: **1 – 2 – 3 – 4 – 8**

Camino 2: **1 – 2 – 3 – 5 – 6 – 8**

Camino 3: **1 – 2 – 5 – 6 – 8**

Camino 4: **1 – 2 – 5 – 7 – 8**

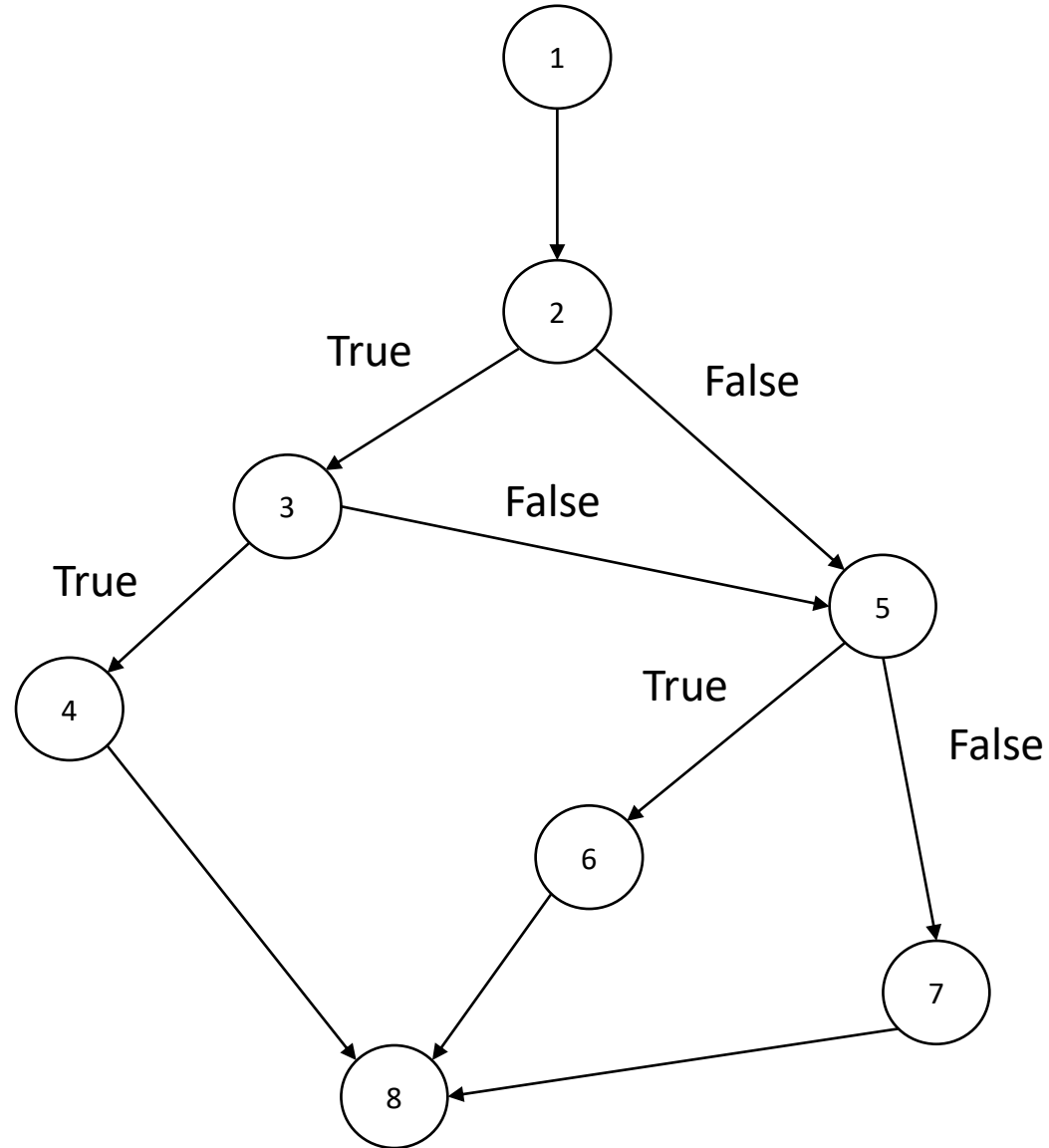
# Definición Conjuntos de Pruebas

## Cobertura de Sentencias

Definir conjuntos de pruebas mínimos para ejecutar cada sentencia/instrucción al menos una vez



# Cobertura de Sentencias



Camino 1: **1 – 2 – 3 – 4 – 8**

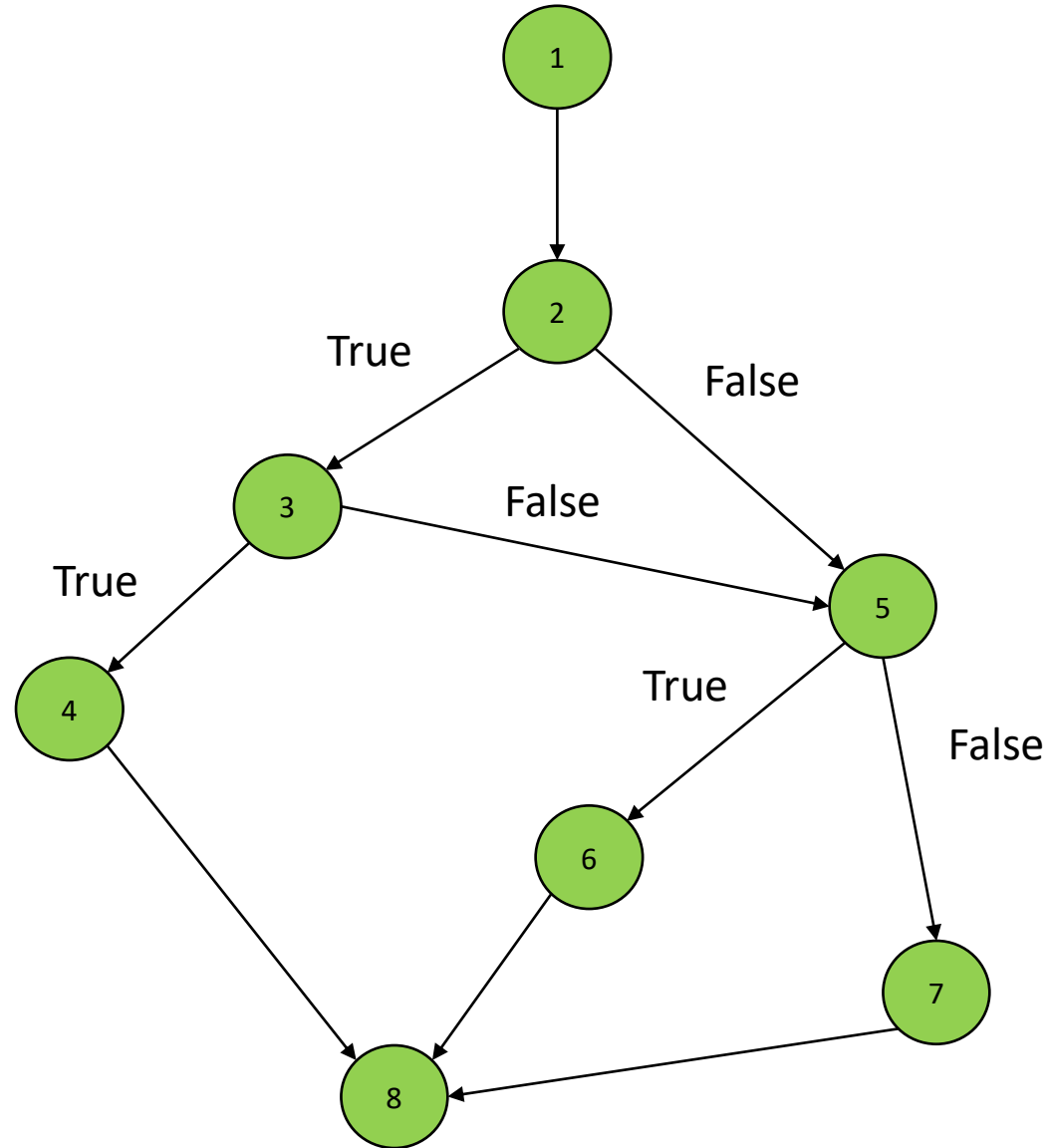
Camino 2: **1 – 2 – 3 – 5 – 6 – 8**

Camino 3: **1 – 2 – 5 – 6 – 8**

Camino 4: **1 – 2 – 5 – 7 – 8**

En este caso los caminos 1, 2 y 4 son suficientes

# Cobertura de Sentencias



Camino 1: **1 – 2 – 3 – 4 – 8**

Camino 2: **1 – 2 – 3 – 5 – 6 – 8**

Camino 3: **1 – 2 – 5 – 6 – 8**

Camino 4: **1 – 2 – 5 – 7 – 8**

En este caso los caminos 1, 2 y 4 son suficientes

# Cobertura de Sentencias

Camino	Características	Caso de Prueba		
		x	y	z
Camino 1	$x > y, x > z$	10	3	3
Camino 2	$y < x < z$	5	2	10
Camino 4	$x < y, z < y$	5	10	5

# Cobertura de Sentencias

Camino	Características	Caso de Prueba		
		x	y	z
Camino 1	$x > y, x > z$	10	3	3
Camino 2	$y < x < z$	5	2	10
Camino 4	$x < y, z < y$	5	10	5

Caso de Prueba 1 (Camino 1). Ejecutaremos un caso en el que  $x > y$  y  $x > z$ , como por ejemplo  $x = 10, y = 3$  y  $z = 3$

Caso de prueba 2 (Camino 2). Ejecutaremos un caso en el que  $y < x < z$ , como por ejemplo  $(x, y, z) = (5, 2, 10)$

Caso de prueba 3 (Camino 4). Ejecutaremos un caso en el que  $x < y$  y  $z < y$ , como por ejemplo  $(x, y, z) = (5, 10, 5)$

# Cobertura de Decisiones

Escribir los casos suficientes para que cada condición tenga al menos un resultado verdadero y otro falso. En este caso, si se emplean los mismos caminos y casos de prueba anteriores, es posible abarcar la cobertura de decisiones.

# Cobertura de Decisiones

Camino	Características	Caso de Prueba		
		x	y	z
Camino 1	$x > y, x > z$	10	3	3
Camino 2	$y < x < z$	5	2	10
Camino 4	$x < y, z < y$	5	10	5

# Criterios de Complejidad

Complejidad	Significado
$\leq 10$	Métodos sencillos, sin mucho riesgo
$> 10 \dots \leq 20$	Métodos medianamente complejos, con riesgo moderado
$> 20 \dots \leq 50$	Métodos complejos, con alto riesgo
$> 50$	Métodos inestables, de altísimo riesgo

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática. Defina conjuntos de pruebas mínimo para criterios de cobertura de sentencias y de decisiones.


```
i = 0;
n = 4;
while (i < n-1) do
    j = i + 1;
    while (j < n) do
        if A[i] < A[j] then
            swap(A[i], A[j]);
        j = j + 1;
    end do;
    i = i + 1;
end do;
```



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.

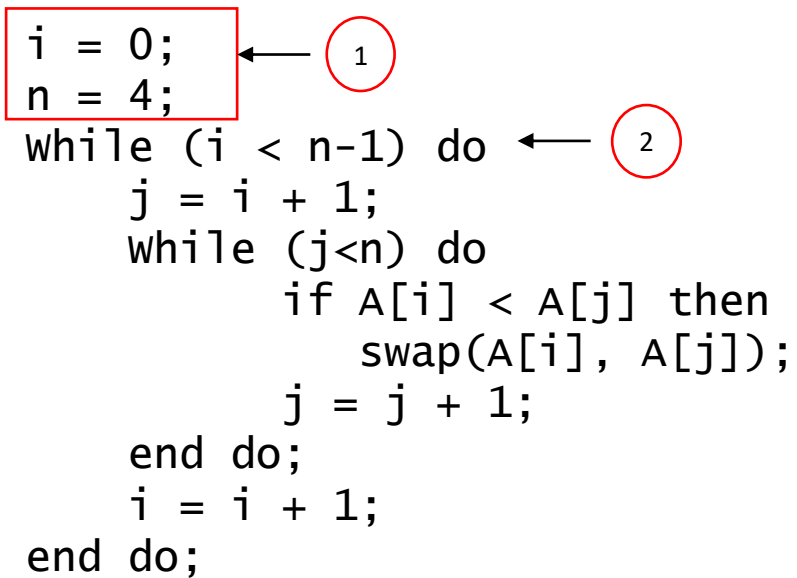
```
i = 0;  
n = 4;  
while (i < n-1) do  
    j = i + 1;  
    while (j < n) do  
        if A[i] < A[j] then  
            swap(A[i], A[j]);  
        j = j + 1;  
    end do;  
    i = i + 1;  
end do;
```



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.

```
i = 0;
n = 4;
while (i < n-1) do
    j = i + 1;
    while (j < n) do
        if A[i] < A[j] then
            swap(A[i], A[j]);
        j = j + 1;
    end do;
    i = i + 1;
end do;
```



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.

```

i = 0;
n = 4;
while (i < n-1) do
  j = i + 1;
  while (j < n) do
    if A[i] < A[j] then
      swap(A[i], A[j]);
    j = j + 1;
  end do;
  i = i + 1;
end do;
```

Control flow graph annotations:

- Node 1: Entry point to the initialization block.
- Node 2: Entry point to the outer while loop.
- Node 3: Entry point to the inner while loop.

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.

```

i = 0;
n = 4;
while (i < n-1) do
  j = i + 1;
  while (j < n) do
    if A[i] < A[j] then
      swap(A[i], A[j]);
    j = j + 1;
  end do;
  i = i + 1;
end do;
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.

```

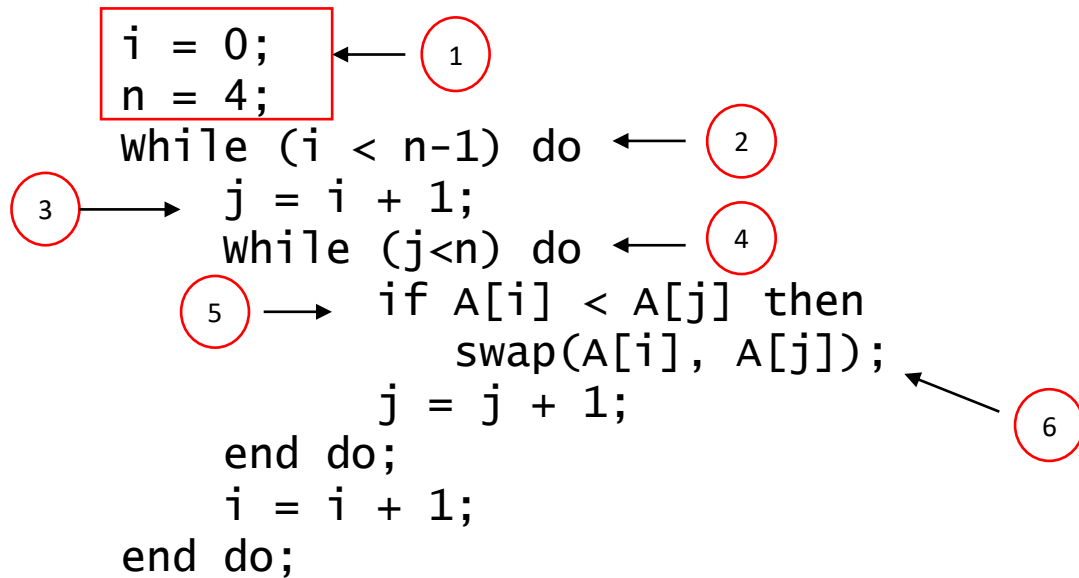
  i = 0;
  n = 4;
while (i < n-1) do
  j = i + 1;
  while (j < n) do
    if A[i] < A[j] then
      swap(A[i], A[j]);
      j = j + 1;
    end do;
    i = i + 1;
  end do;
end do;
```

```

graph TD
  1((1)) --> B1[i = 0;  
n = 4;]
  B1 --> 2((2))
  2 --> B2[while (i < n-1) do]
  B2 --> 3((3))
  3 --> B3[j = i + 1;]
  B3 --> 4((4))
  4 --> B4[while (j < n) do]
  B4 --> 5((5))
  5 --> B5[if A[i] < A[j] then  
  swap(A[i], A[j]);  
  j = j + 1;  
end do;  
i = i + 1;]
  B5 --> 2
```

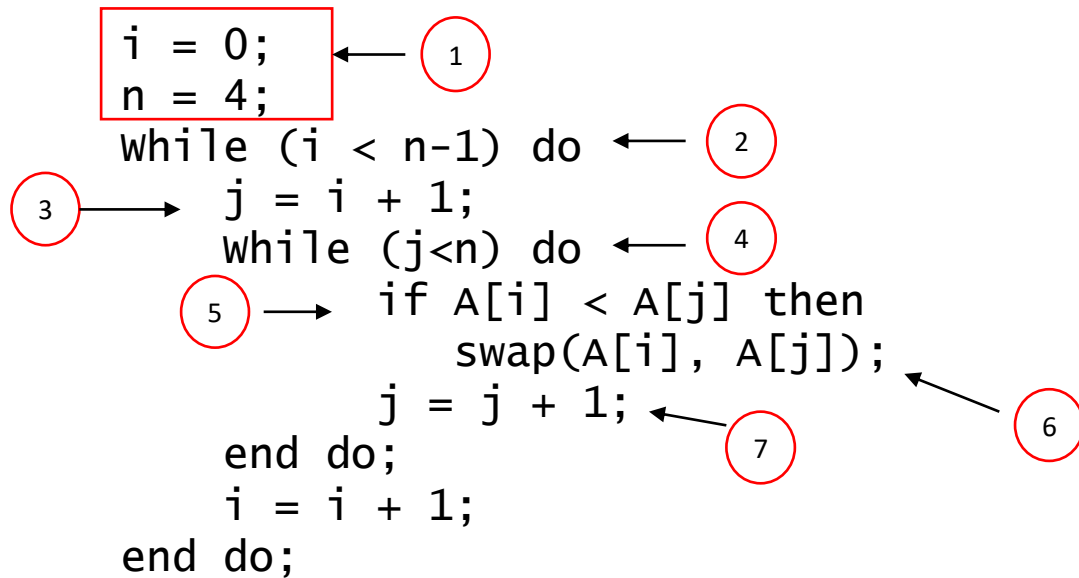
# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.



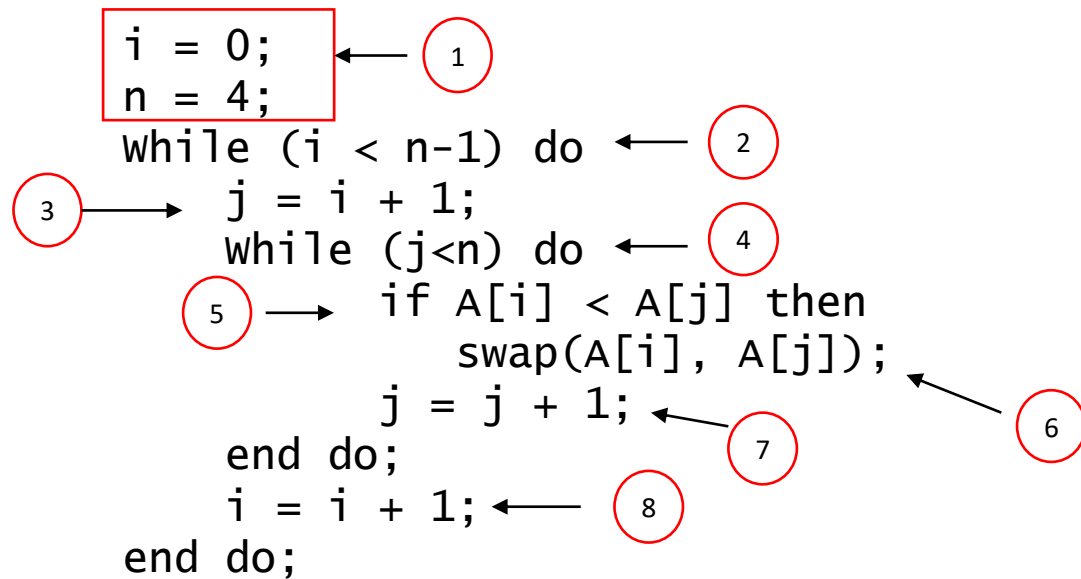
# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.



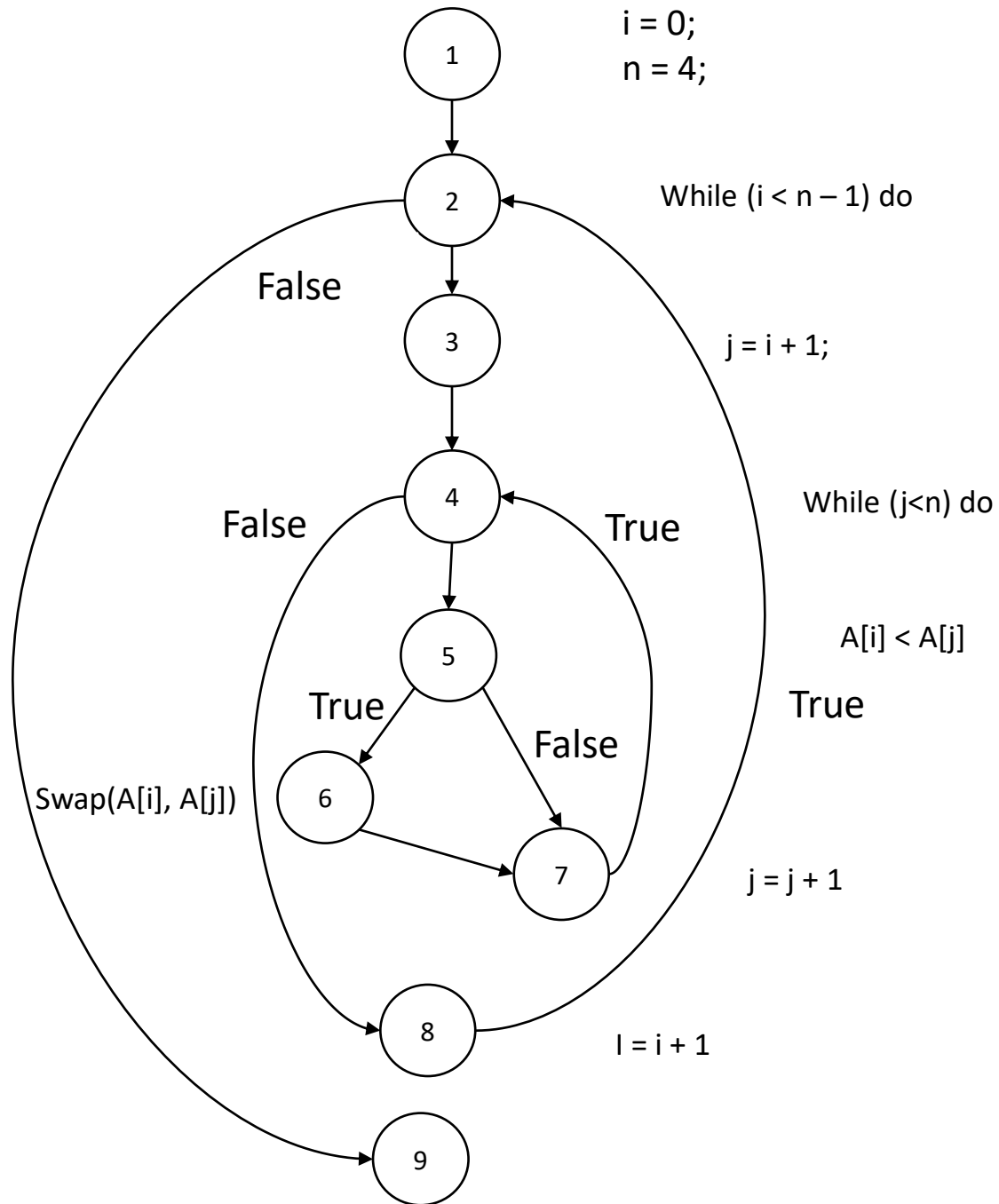
# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática.





# Grafo



Cálculo de la Complejidad

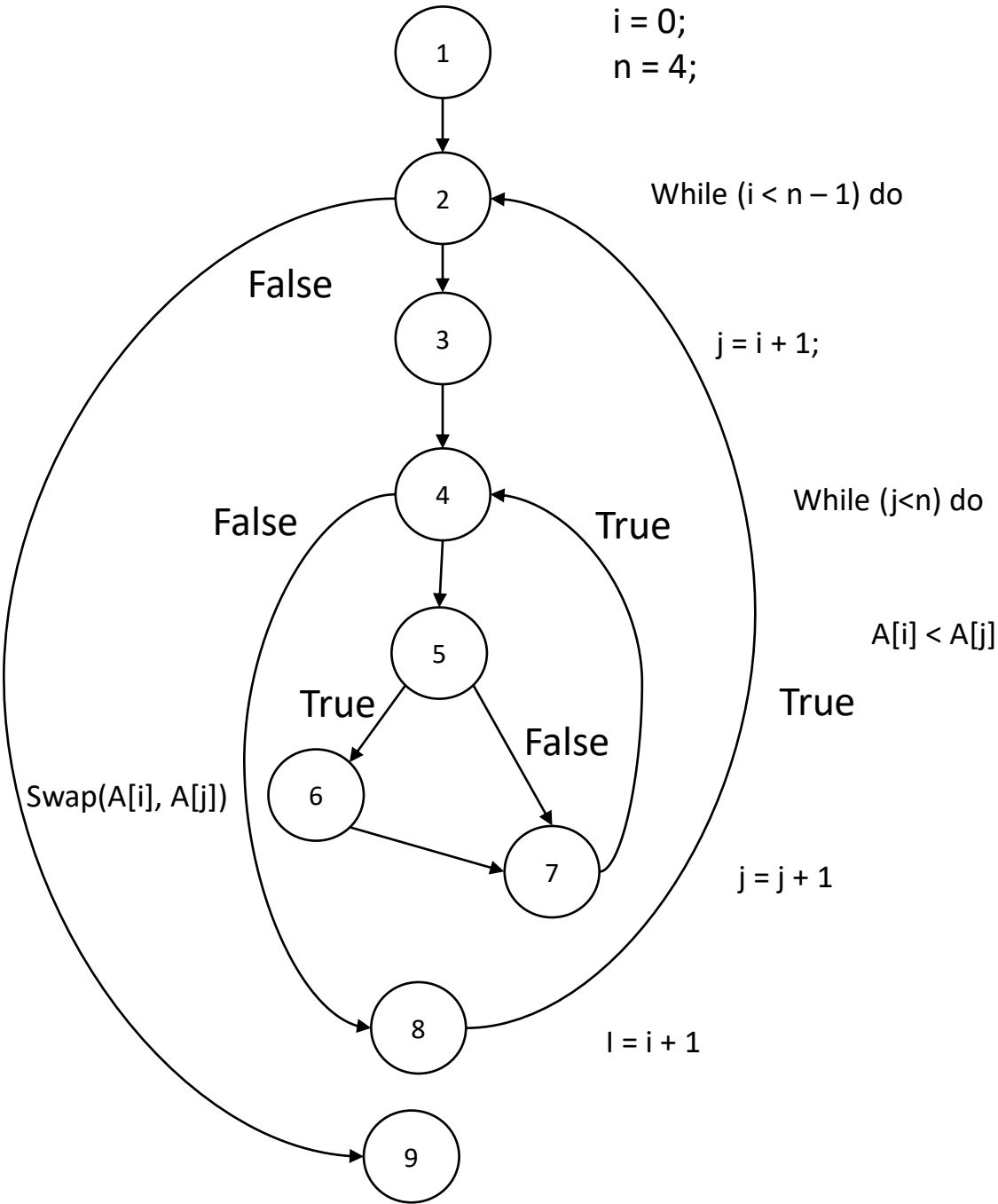
**Complejidad = Aristas – Nodos + 2**

**Aristas = 11**

**Nodos = 9**

**Complejidad = 11 – 9 + 2 = 4**

# Grafo



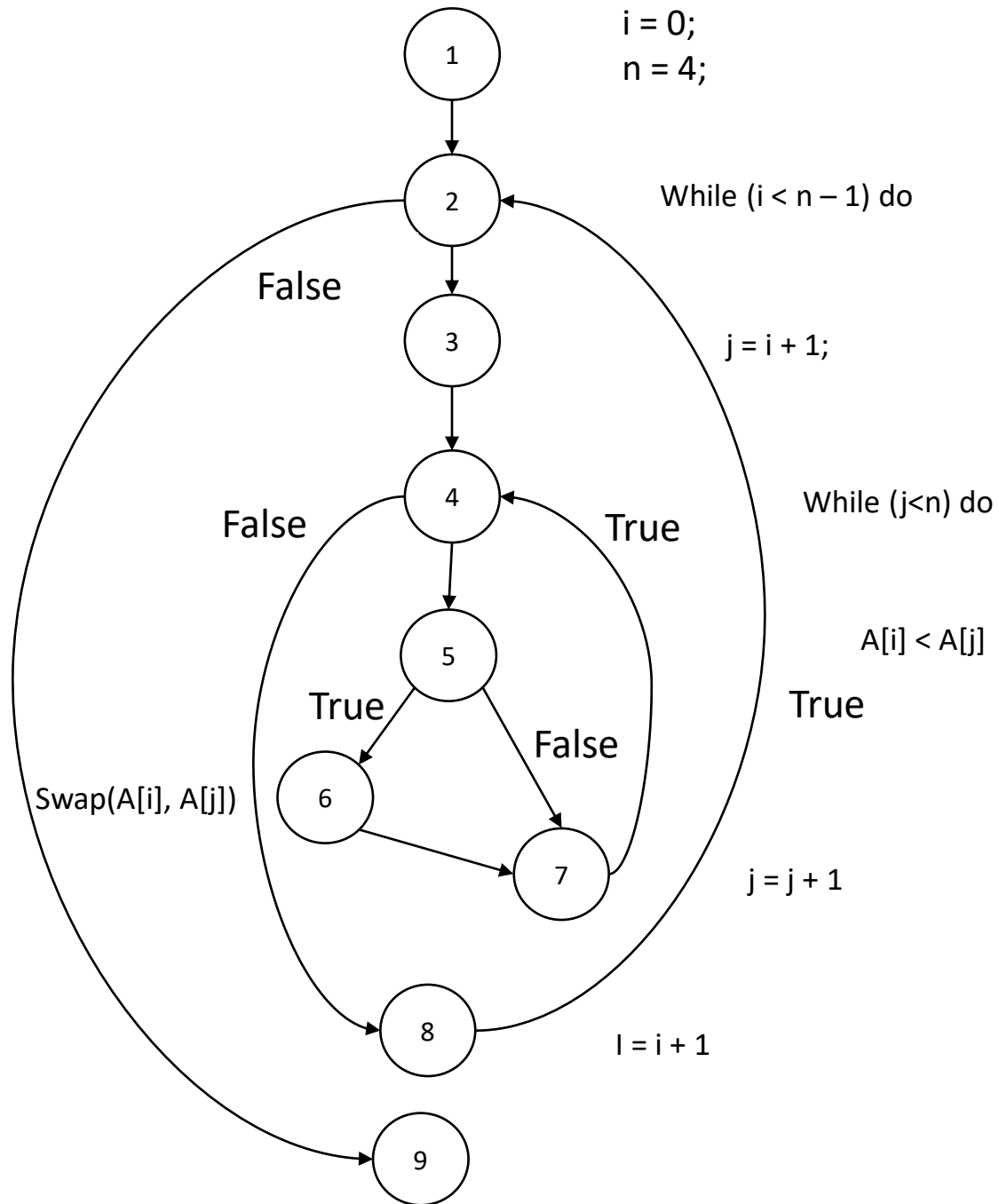
Camino 1: **1 – 2 - 9**

Camino 2: **1 – 2 – 3 – 4 – 8 – 2 - 9**

Camino 3: **1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 8 – 2 - 9**

Camino 4: **1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 2 - 9**

# Grafo



Camino 1: **1 – 2 - 9**

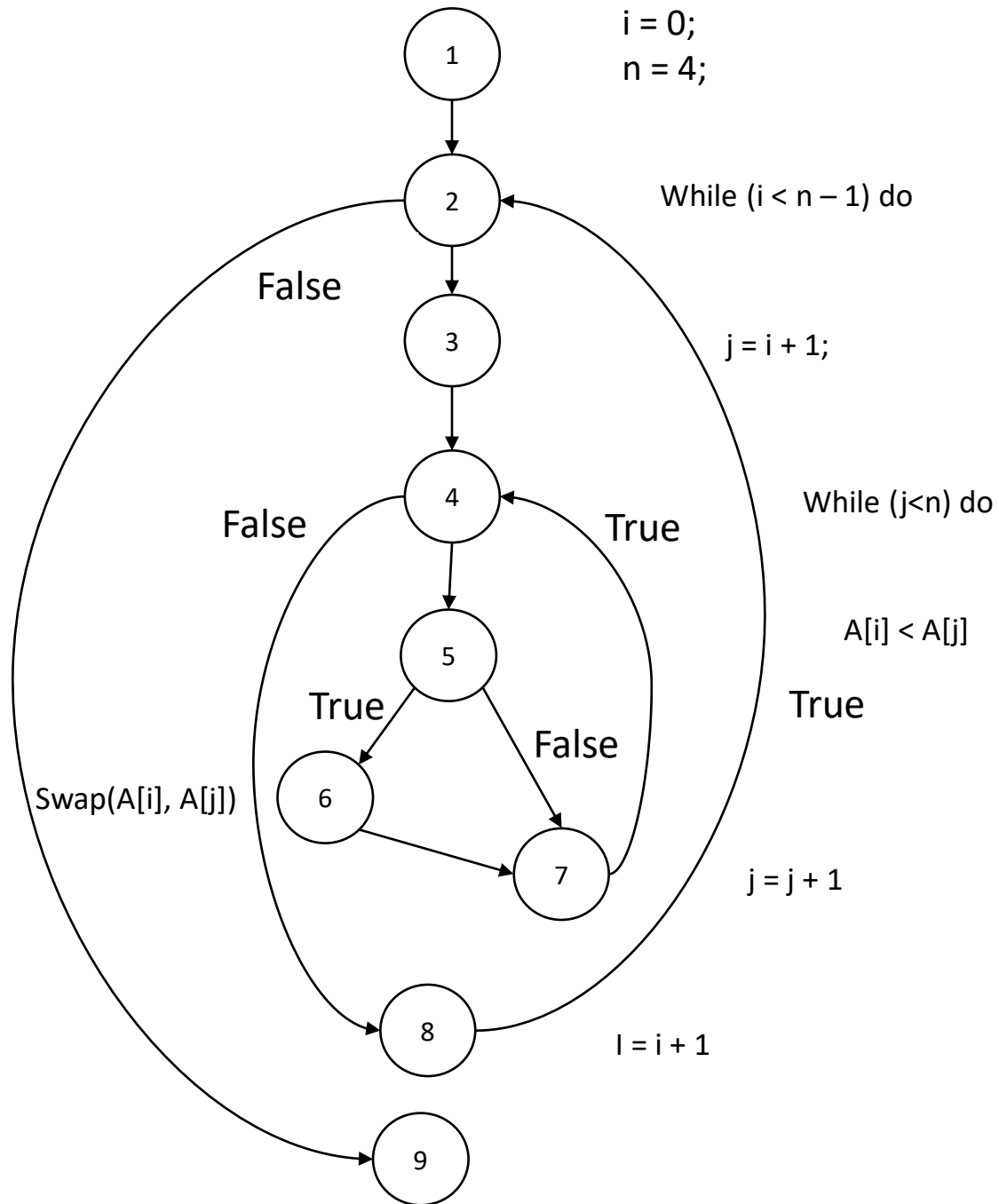
Camino 2: **1 – 2 – 3 – 4 – 8 – 2 - 9**

Camino 3: **1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 8 – 2 - 9**

Camino 4: **1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 2 - 9**

**¿Hay algún camino que sea redundante?**

# Grafo



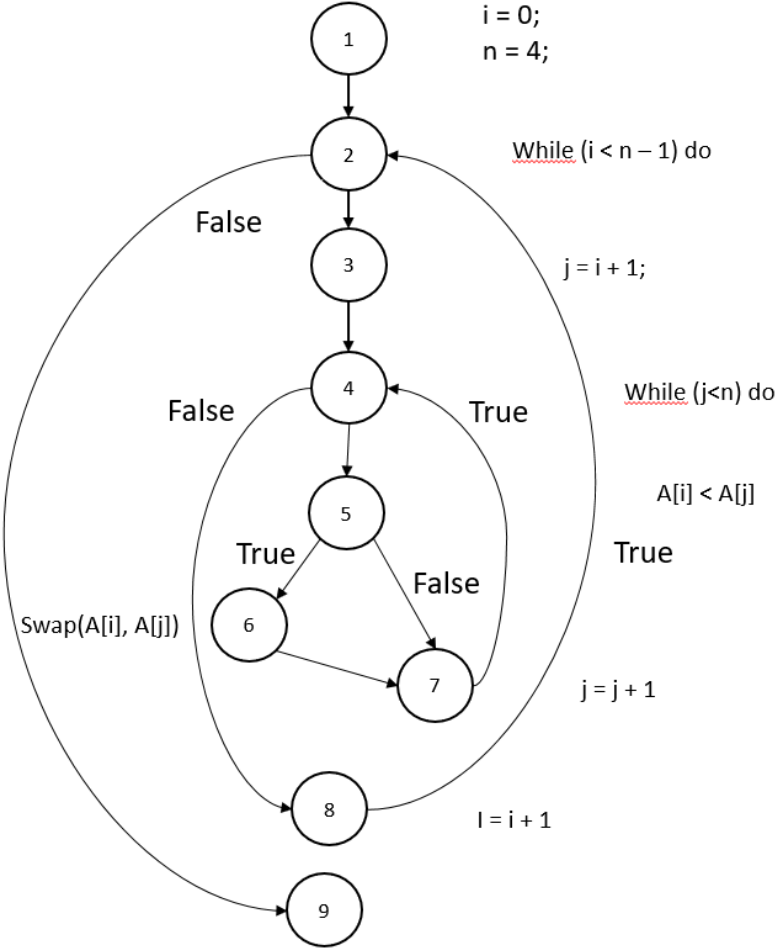
Camino 1: **1 – 2 - 9**

Camino 2: **1 – 2 – 3 – 4 – 8 – 2 - 9**

Camino 3: **1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 8 – 2 - 9**

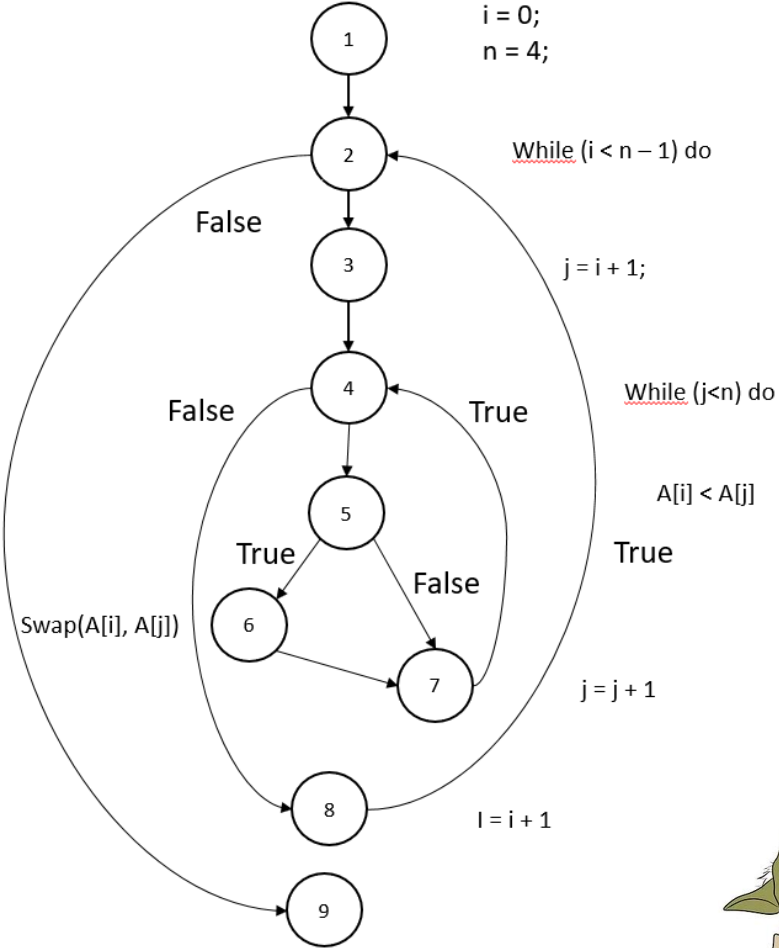
Camino 4: **1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 2 - 9**

**¿Hay algún camino que sea redundante?**



- Camino 1: 1 – 2 - 9  
Camino 2: 1 – 2 – 3 – 4 – 8 – 2 - 9  
Camino 3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 8 – 2 - 9  
Camino 4: 1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 2 - 9

Camino	Características	Caso de Prueba					
		i	n	j	A[1]	A[2]	A[3]
1- 2 - 9	$i \geq n - 1, j \geq n$	10	10	11	x	x	x
1 – 2 – 3 – 4 – 8 – 2 - 9	$i < n - 1, j \geq n$	0	4	4	x	x	x
1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 8 – 2 - 9	$i < n - 1, j < n, A[i] < A[j]$	0	4	1	6	7	8
1 - 2 – 3 – 4 – 5 – 7 – 4 – 8 – 2 - 9	$i < n - 1, j < n, A[i] \geq A[j]$	0	4	1	8	7	6



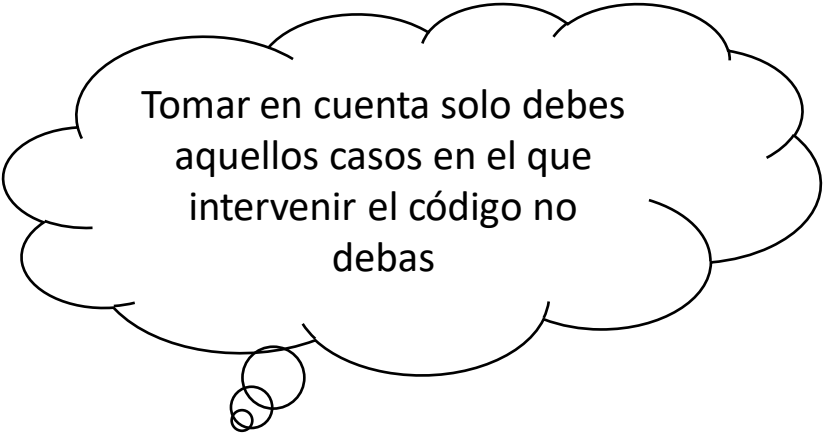
Camino 1: 1 - 2 - 9

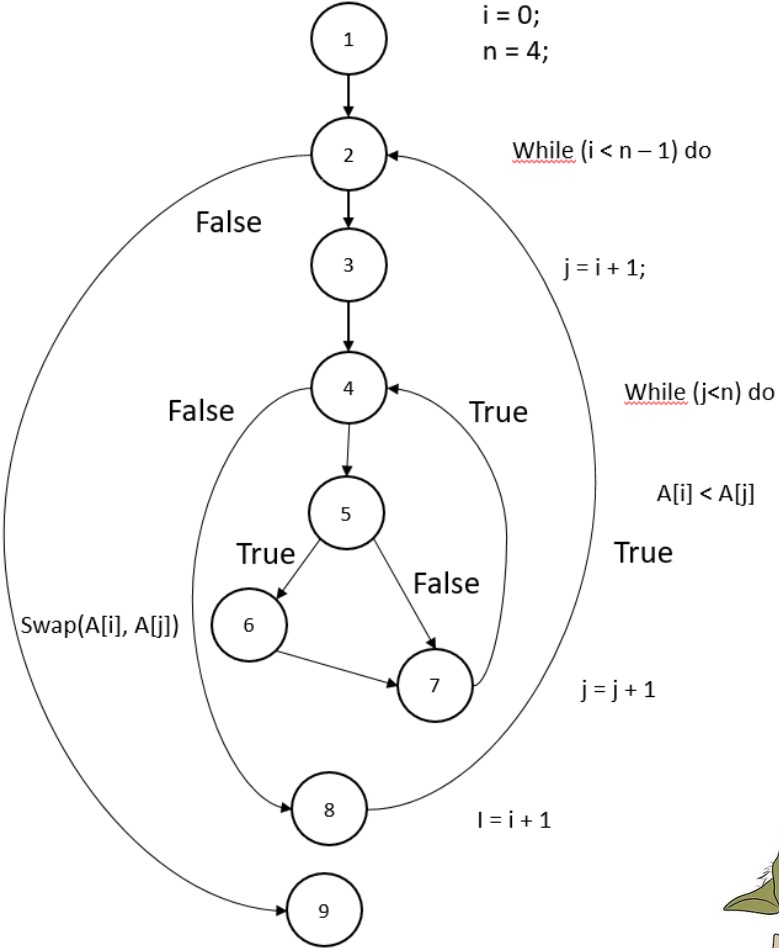
Camino 2: 1 - 2 - 3 - 4 - 8 - 2 - 9

Camino 3: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 8 - 2 - 9

Camino 4: 1 - 2 - 3 - 4 - 5 - 7 - 4 - 8 - 2 - 9

Camino	Características	Caso de Prueba					
		i	n	j	A[1]	A[2]	A[3]
1- 2 - 9	$i \geq n - 1, j \geq n$	10	10	11	x	x	x
1 - 2 - 3 - 4 - 8 - 2 - 9	$i < n - 1, j \geq n$	0	4	4	x	x	x
1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 8 - 2 - 9	$i < n - 1, j < n, A[i] < A[j]$	0	4	1	6	7	8
1 - 2 - 3 - 4 - 5 - 7 - 4 - 8 - 2 - 9	$i < n - 1, j < n, A[i] \geq A[j]$	0	4	1	8	7	6





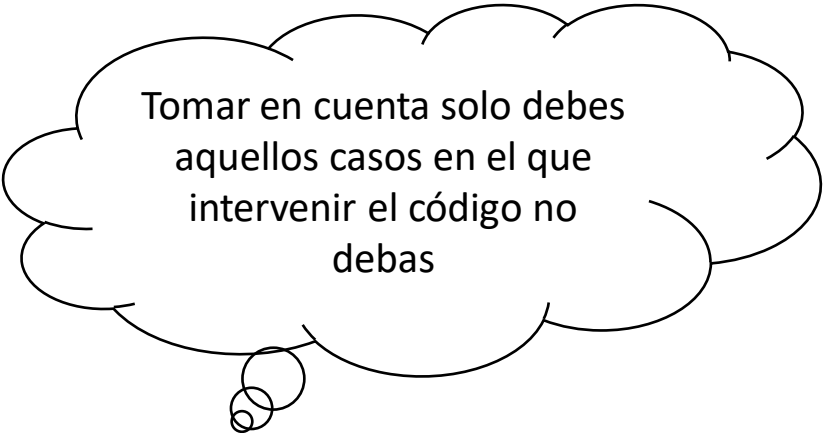
Camino 1: 1 - 2 - 9

Camino 2: 1 - 2 - 3 - 4 - 8 - 2 - 9

Camino 3: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 8 - 2 - 9

Camino 4: 1 - 2 - 3 - 4 - 5 - 7 - 4 - 8 - 2 - 9

Camino	Características	Caso de Prueba					
		i	n	j	A[1]	A[2]	A[3]
1- 2 - 9	$i \geq n - 1, j \geq n$	10	10	11	x	x	x
1 - 2 - 3 - 4 - 8 - 2 - 9	$i < n - 1, j \geq n$	0	4	4	x	x	x
1 - 2 - 3 - 4 - 5 - 6 - 7 - 4 - 8 - 2 - 9	$i < n - 1, j < n, A[i] < A[j]$	0	4	1	6	7	8
1 - 2 - 3 - 4 - 5 - 7 - 4 - 8 - 2 - 9	$i < n - 1, j < n, A[i] \geq A[j]$	0	4	1	8	7	6



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
}
for (int i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        sumaF+=cm[i][j];
    }
    if(sumaF!=15){
        break;
    }else{conta++;}
    sumaF=0;
    sumaC3+=cm[i][2];
    sumaC2+=cm[i][1];
    sumaC+=cm[i][0];
    sumaD1+=cm[i][i];
    sumaD2+=cm[i][j-1];
}
if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
    System.out.println("SI");
}else{
    System.out.println("NO");
}
```



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
}
for (int i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        sumaF+=cm[i][j];
    }
    if(sumaF!=15){
        break;
    }else{conta++;}
    sumaF=0;
    sumaC3+=cm[i][2];
    sumaC2+=cm[i][1];
    sumaC+=cm[i][0];
    sumaD1+=cm[i][i];
    sumaD2+=cm[i][j-1];
}
if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
    System.out.println("SI");
}else{
    System.out.println("NO");
}
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
}
for (int i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        sumaF+=cm[i][j];
    }
    if(sumaF!=15){
        break;
    }else{conta++;}
    sumaF=0;
    sumaC3+=cm[i][2];
    sumaC2+=cm[i][1];
    sumaC+=cm[i][0];
    sumaD1+=cm[i][i];
    sumaD2+=cm[i][j-1];
}
if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
    System.out.println("SI");
}else{
    System.out.println("NO");
}
```

1

2

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
2 → for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) { ← 3  
        cm[i][k]=sc.nextInt();  
    }  
}  
for (int i = 0; i < 3; i++) {  
    for (j = 0; j < 3; j++) {  
        sumaF+=cm[i][j];  
    }  
    if(sumaF!=15){  
        break;  
    }else{conta++;}  
    sumaF=0;  
    sumaC3+=cm[i][2];  
    sumaC2+=cm[i][1];  
    sumaC+=cm[i][0];  
    sumaD1+=cm[i][i];  
    sumaD2+=cm[i][j-1];  
}  
if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
    System.out.println("SI");  
}else{  
    System.out.println("NO");  
}
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
2 → for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) { ← 3  
        4 → cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}
```

The diagram illustrates a control flow graph for the provided Java code. The nodes are numbered 1 through 5, each enclosed in a red circle. Arrows indicate the flow from these nodes to specific lines of code:

- Node 1 points to the first line of code: `Scanner sc=new Scanner(System.in);`
- Node 2 points to the start of the first nested loop: `for (int i = 0; i < 3; i++) {`
- Node 3 points to the inner loop's body: `cm[i][k]=sc.nextInt();`
- Node 4 points to the closing brace of the inner loop: `}`
- Node 5 points to the start of the second nested loop: `for (int i = 0; i < 3; i++) {`

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}
```

```
graph TD
    1((1)) --> L1[Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];]
    2((2)) --> L2[for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}]
    3((3)) --> L3[for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}]
    4((4)) --> L4[cm[i][k]=sc.nextInt();  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}]
    5((5)) --> L5[for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}]
    6((6)) --> L6[for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}]
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}
```

Diagram illustrating the control flow graph (CFG) nodes and their connections:

- Node 1: Start of the Scanner declaration.
- Node 2: Entry to the first for loop (`for (int i = 0; i < 3; i++)`).
- Node 3: Entry to the inner for loop (`for (int k = 0; k < 3; k++)`).
- Node 4: Statement `cm[i][k]=sc.nextInt();`.
- Node 5: Entry to the second for loop (`for (int i = 0; i < 3; i++)`).
- Node 6: Entry to the inner for loop (`for (j = 0; j < 3; j++)`).
- Node 7: Statement `sumaF+=cm[i][j];`.

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);  
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;  
int[][] cm = new int[3][3];  
for (int i = 0; i < 3; i++) {  
    for (int k = 0; k < 3; k++) {  
        cm[i][k]=sc.nextInt();  
    }  
    for (int i = 0; i < 3; i++) {  
        for (j = 0; j < 3; j++) {  
            sumaF+=cm[i][j];  
        }  
        if(sumaF!=15){  
            break;  
        }else{conta++;}  
        sumaF=0;  
        sumaC3+=cm[i][2];  
        sumaC2+=cm[i][1];  
        sumaC+=cm[i][0];  
        sumaD1+=cm[i][i];  
        sumaD2+=cm[i][j-1];  
    }  
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){  
        System.out.println("SI");  
    }else{  
        System.out.println("NO");  
    }  
}
```

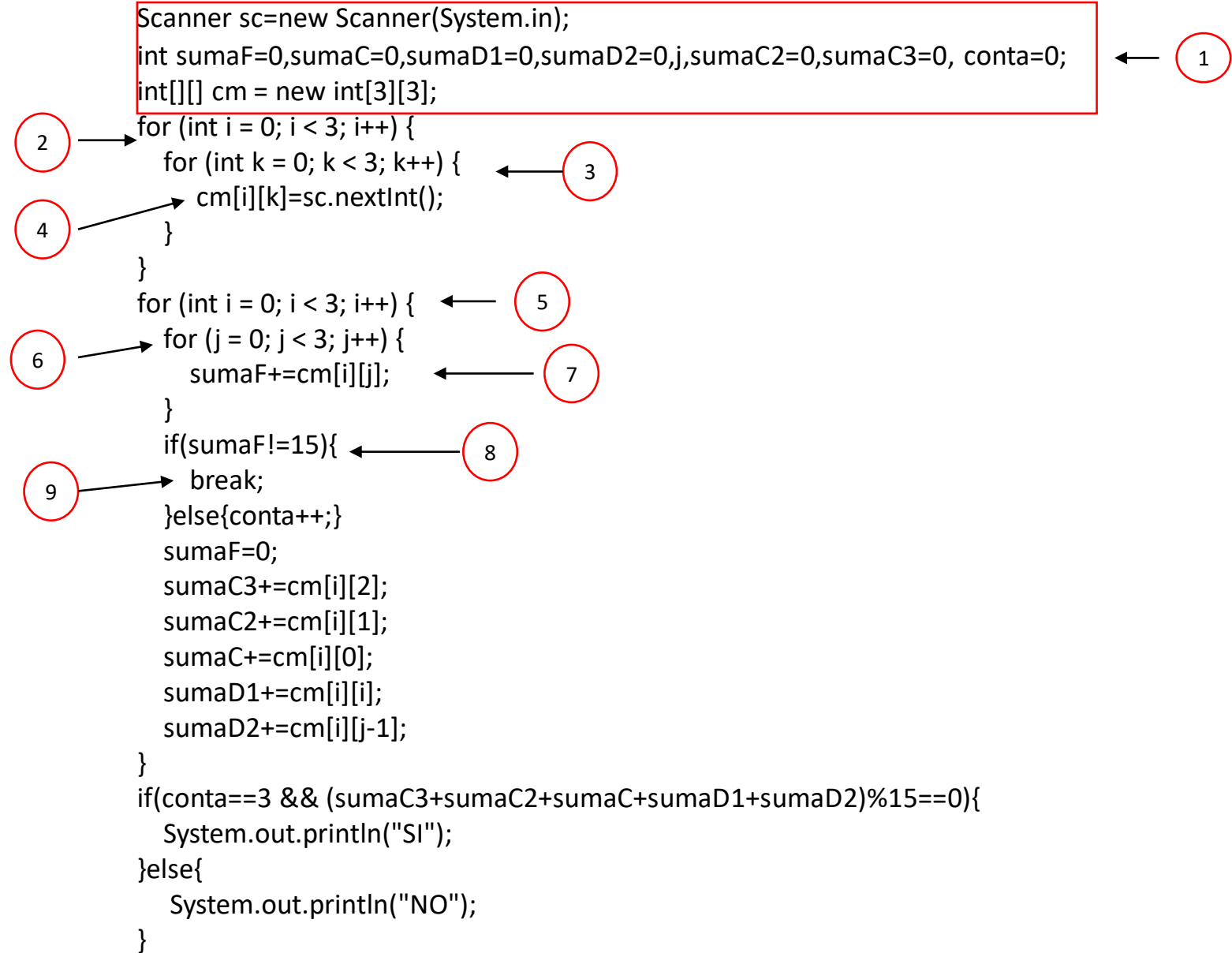
Diagram illustrating the control flow of the provided Java code, with nodes marked by red circles and arrows indicating the flow:

- Node 1: Start of the Scanner declaration.
- Node 2: Entry to the first for loop (for (int i = 0; i < 3; i++) {
- Node 3: Entry to the inner for loop (for (int k = 0; k < 3; k++) {
- Node 4: Statement cm[i][k]=sc.nextInt();
- Node 5: Entry to the second for loop (for (int i = 0; i < 3; i++) {
- Node 6: Entry to the inner for loop (for (j = 0; j < 3; j++) {
- Node 7: Statement sumaF+=cm[i][j];
- Node 8: Statement if(sumaF!=15){



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
    for (int i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            sumaF+=cm[i][j];
        }
        if(sumaF!=15){
            break;
        }else{conta++;}
        sumaF=0;
        sumaC3+=cm[i][2];
        sumaC2+=cm[i][1];
        sumaC+=cm[i][0];
        sumaD1+=cm[i][i];
        sumaD2+=cm[i][j-1];
    }
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
        System.out.println("SI");
    }else{
        System.out.println("NO");
    }
}
```

1

2

3

4

5

6

7

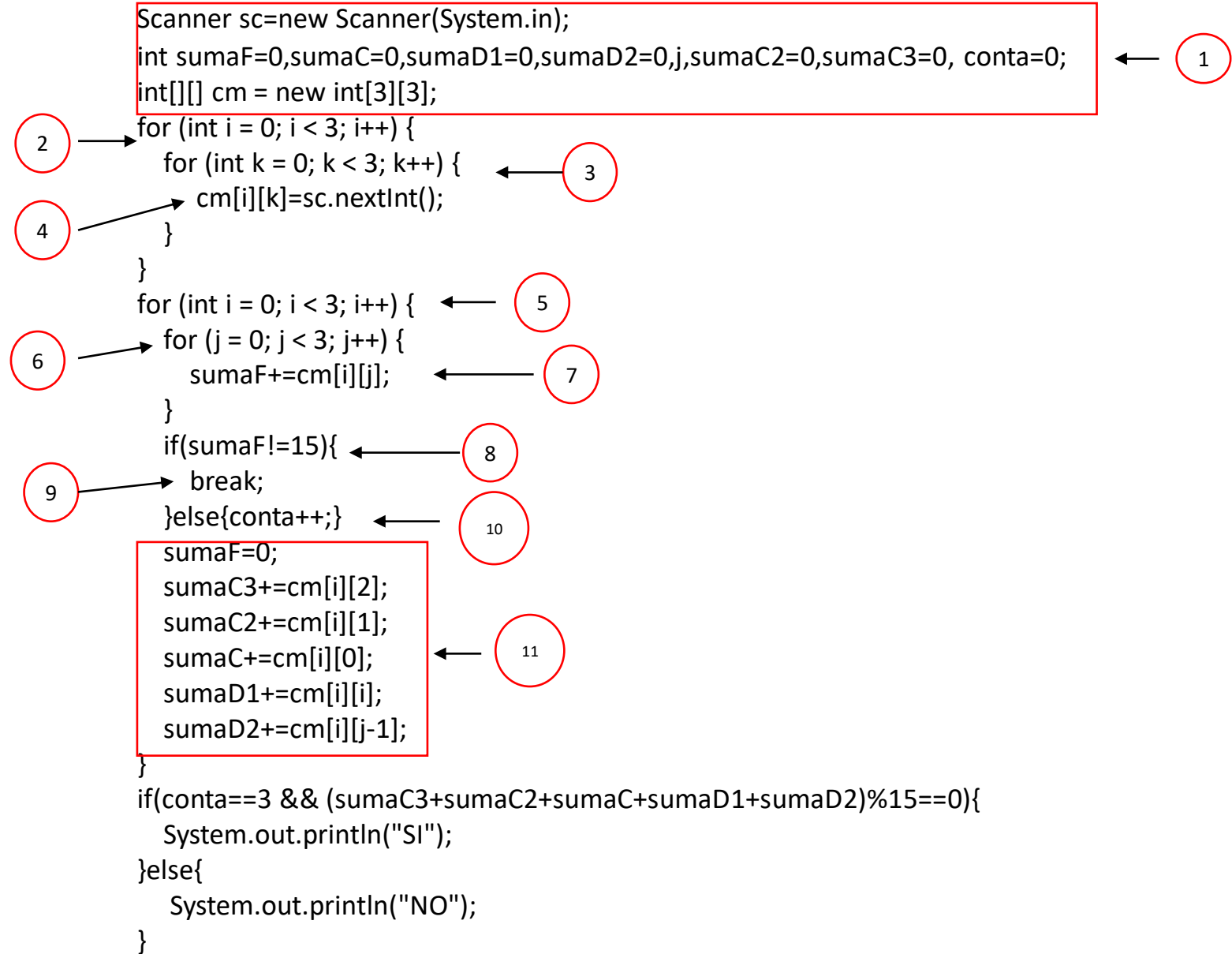
8

9

10

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.



# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
    for (int i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            sumaF+=cm[i][j];
        }
        if(sumaF!=15){
            break;
        }else{conta++;}
        sumaF=0;
        sumaC3+=cm[i][2];
        sumaC2+=cm[i][1];
        sumaC+=cm[i][0];
        sumaD1+=cm[i][i];
        sumaD2+=cm[i][j-1];
    }
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
        System.out.println("SI");
    }else{
        System.out.println("NO");
    }
}
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
    for (int i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            sumaF+=cm[i][j];
        }
        if(sumaF!=15){
            break;
        }else{conta++;}
        sumaF=0;
        sumaC3+=cm[i][2];
        sumaC2+=cm[i][1];
        sumaC+=cm[i][0];
        sumaD1+=cm[i][i];
        sumaD2+=cm[i][j-1];
    }
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
        System.out.println("SI");
    }else{
        System.out.println("NO");
    }
}
```

← 1

2 →

← 3

4 →

← 5

6 →

← 7

9 →

← 8

← 10

← 11

13 →

12 →

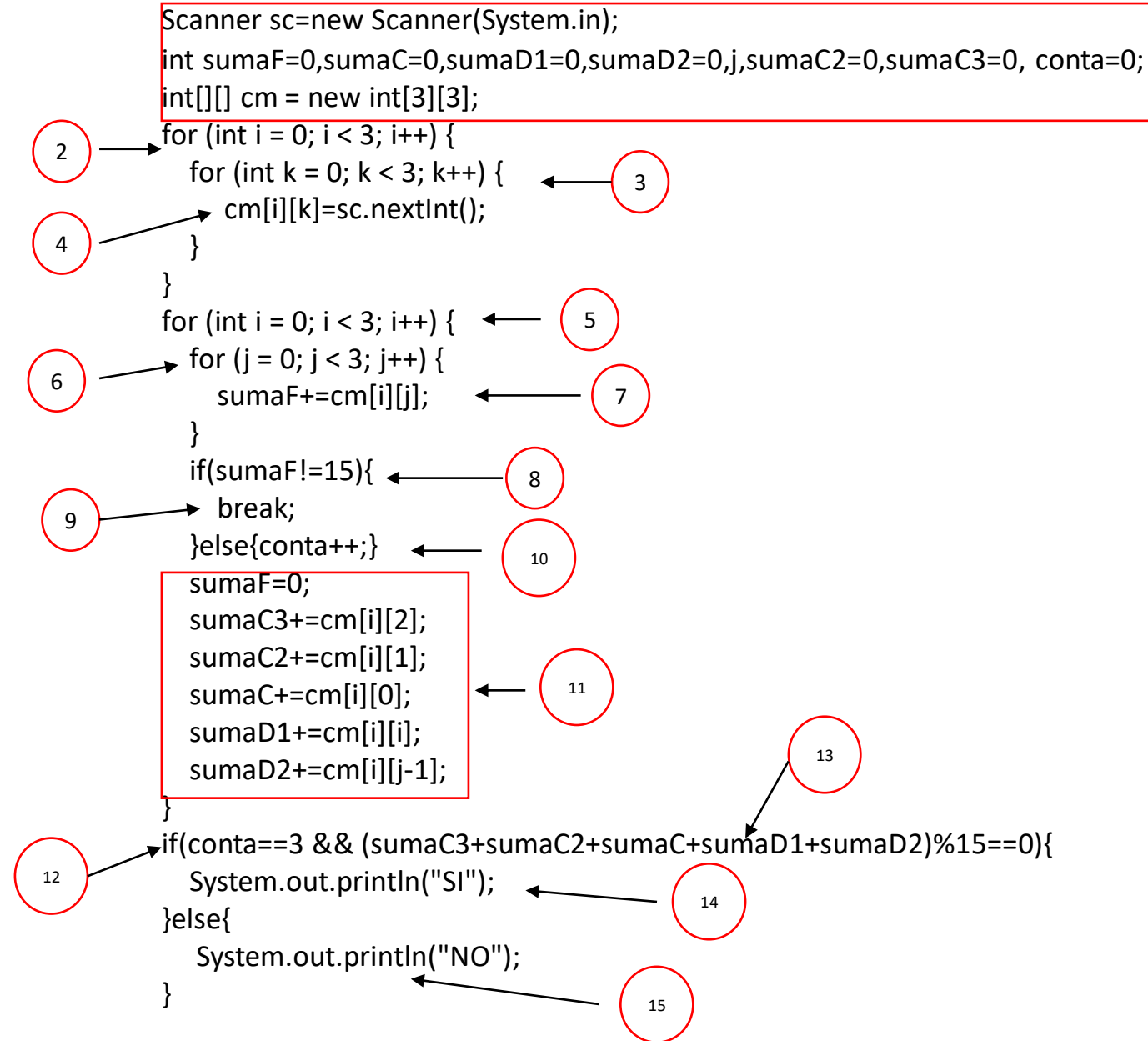
# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

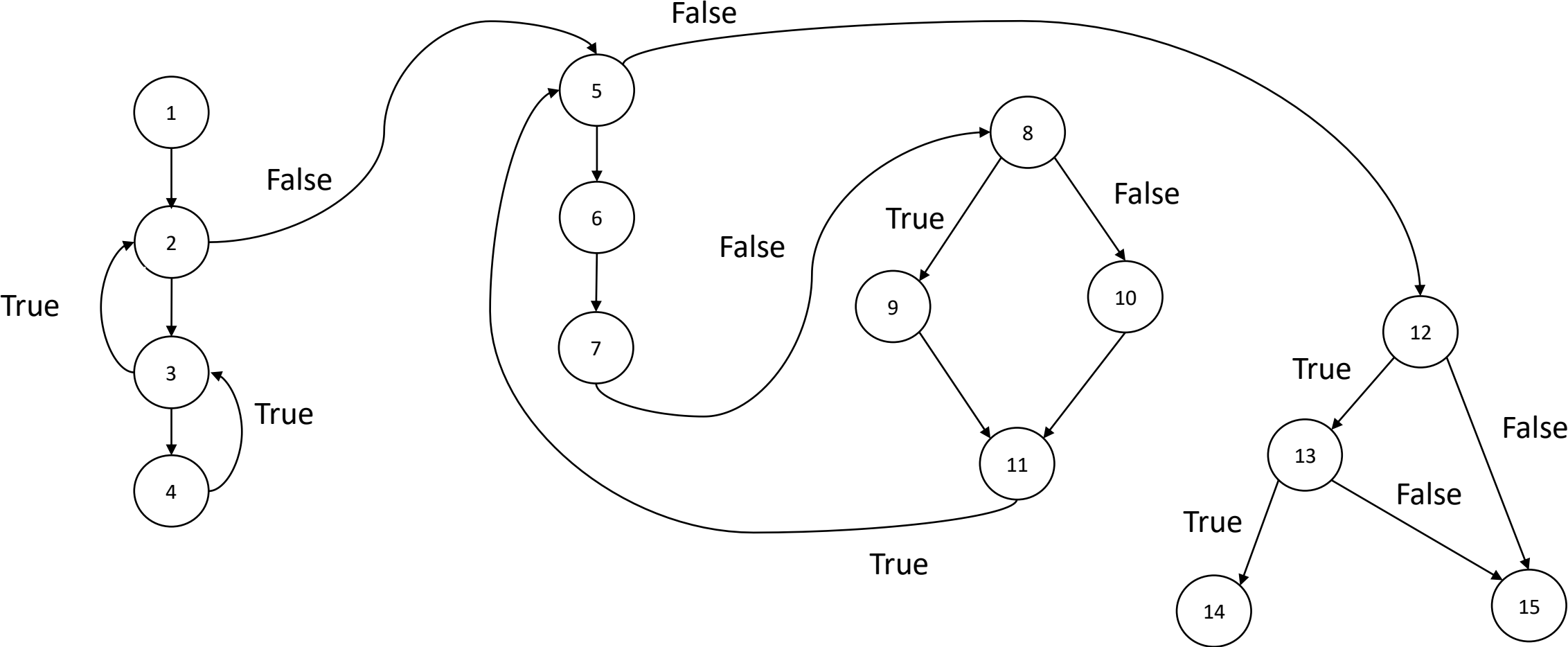
```
Scanner sc=new Scanner(System.in);
int sumaF=0,sumaC=0,sumaD1=0,sumaD2=0,j,sumaC2=0,sumaC3=0, conta=0;
int[][] cm = new int[3][3];
for (int i = 0; i < 3; i++) {
    for (int k = 0; k < 3; k++) {
        cm[i][k]=sc.nextInt();
    }
    for (int i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            sumaF+=cm[i][j];
        }
        if(sumaF!=15){
            break;
        }else{conta++;}
        sumaF=0;
        sumaC3+=cm[i][2];
        sumaC2+=cm[i][1];
        sumaC+=cm[i][0];
        sumaD1+=cm[i][i];
        sumaD2+=cm[i][j-1];
    }
    if(conta==3 && (sumaC3+sumaC2+sumaC+sumaD1+sumaD2)%15==0){
        System.out.println("SI");
    }else{
        System.out.println("NO");
    }
}
```

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo y calcule la complejidad ciclomática y defina los conjuntos de pruebas mínimos para alcanzar los criterios de cobertura de: sentencias y decisiones.

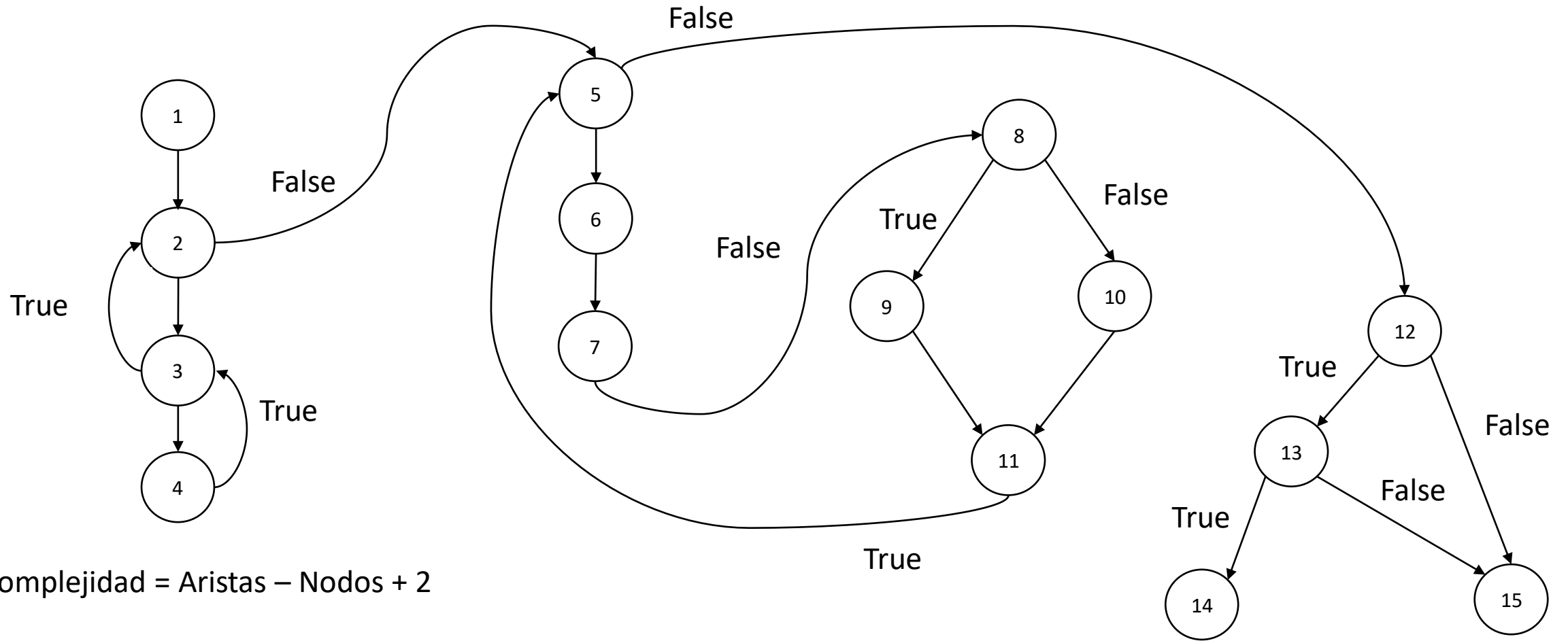


# Grafo





# Cálculo Complejidad



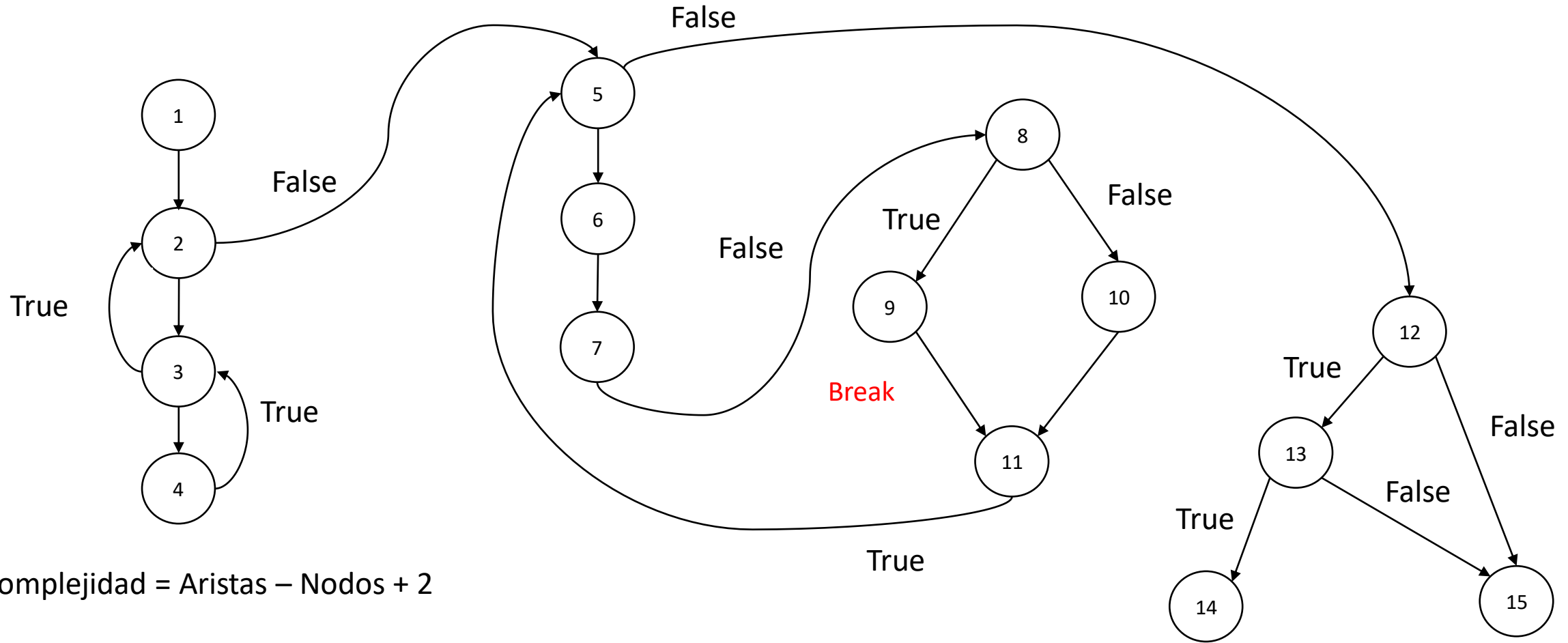
Complejidad = Aristas – Nodos + 2

En nuestro caso =  $19 - 15 + 2 = 6$

Luego

Complejidad = 6

# Cálculo Complejidad



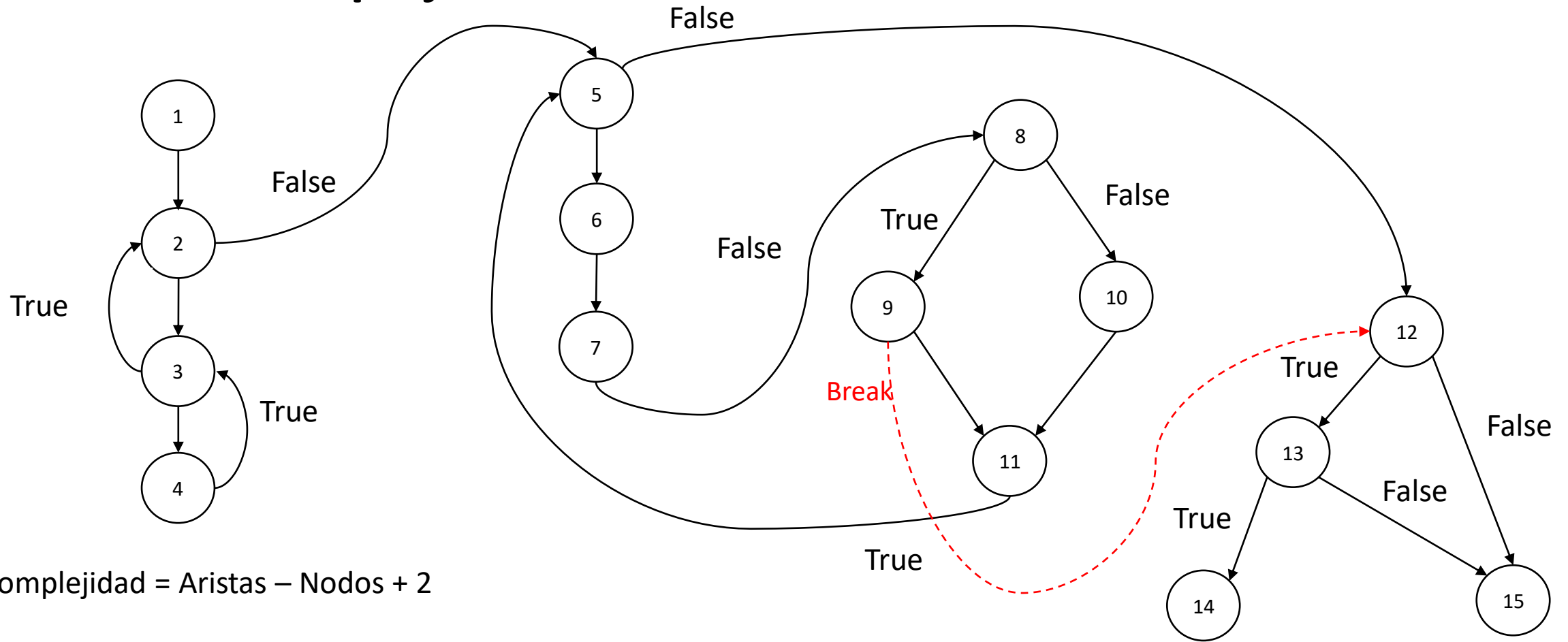
Complejidad = Aristas – Nodos + 2

En nuestro caso =  $19 - 15 + 2 = 6$

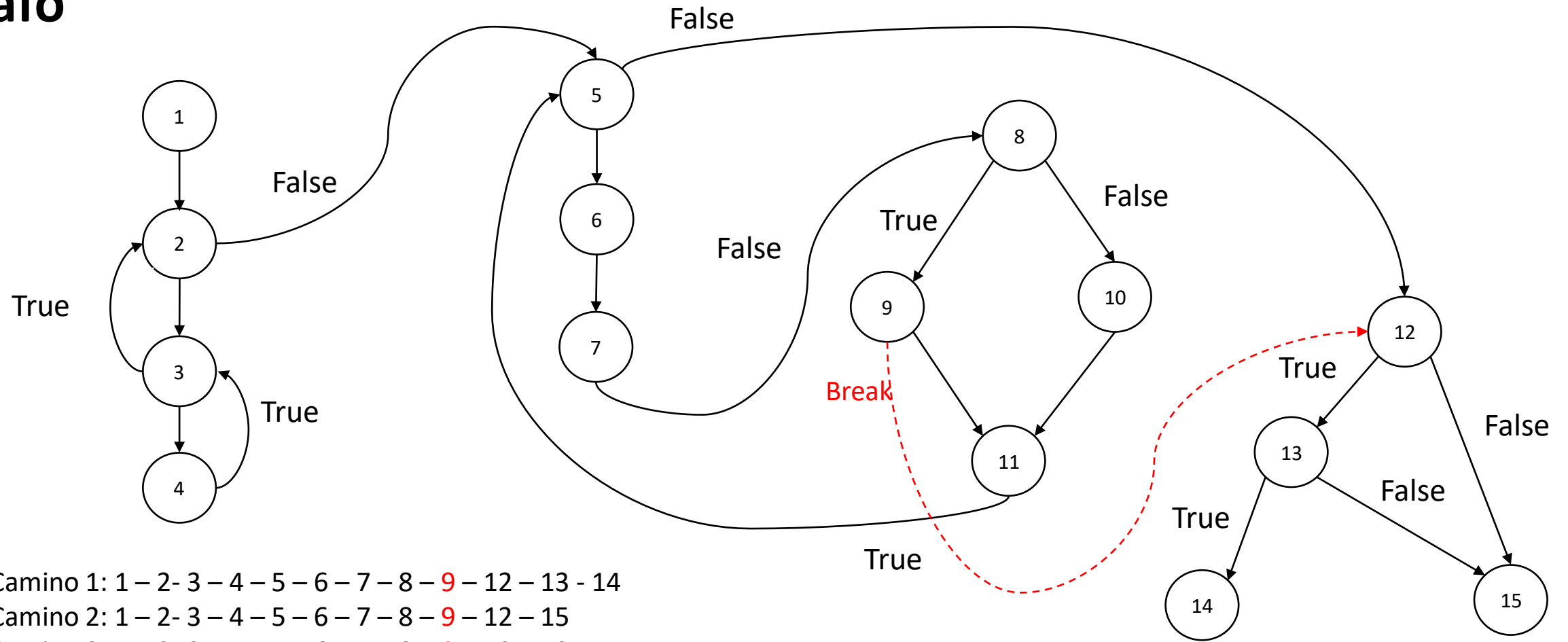
Luego

Complejidad = 6

# Cálculo de Complejidad



# Grafo



Camino 1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14

Camino 2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 15

Camino 3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 15

Camino 4: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 5 – 12 – 13 – 14

Camino 5: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 5 – 12 – 13 – 15

Camino 6: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10 – 11 – 5 – 12 – 15

Camino 7: 1 – 2 – 5 – 12 – 15

# Ejercicios

Dado el siguiente programa, presente el grafo del mismo.

*Aporte*                      *Alumno:*  
*Sebastián Sandoval*

Nodo	Instrucción
1	while (x < 100) {
2	if (a[x] % 2 == 0) {
3	parity = 0;
	}
	else {
4	parity = 1;
5	}
6	switch (parity) {
	case 0:
7	println("a[" + i + "] is even");
	case 1:
8	println( "a[" + i + "] is odd");
	default:
9	println("Unexpected error");
	}
10	x++;
	}
11	p = true;

Nodo	Instrucción
1	while (x < 100) {
2	if (a[x] % 2 == 0) {
3	parity = 0;
	}
	else {
4	parity = 1;
5	}
6	switch (parity) {
	case 0:
7	println("a[" + i + "] is even");
	case 1:
8	println( "a[" + i + "] is odd");
	default:
9	println("Unexpected error");
	}
10	x++;
	}
11	p = true;

