UNIVERSIDAD NACIONAL DE LA MATANZA

Análisis de Software

Actividad 3

- Primer Cuatrimestre de 2021.
- Profesores:
  - Agustín Gustavo.
  - Del Ben Enzo.
  - Marcelo Vinjoy.
- Grupo 3

TABLE OF CONTENTS

CONSIGNAS

Aplicar el método de Complejidad Ciclomática (McCabe) al programa del triángulo. Desde el grafo hasta las pruebas.

RESOLUCIÓN

Código de la aplicación

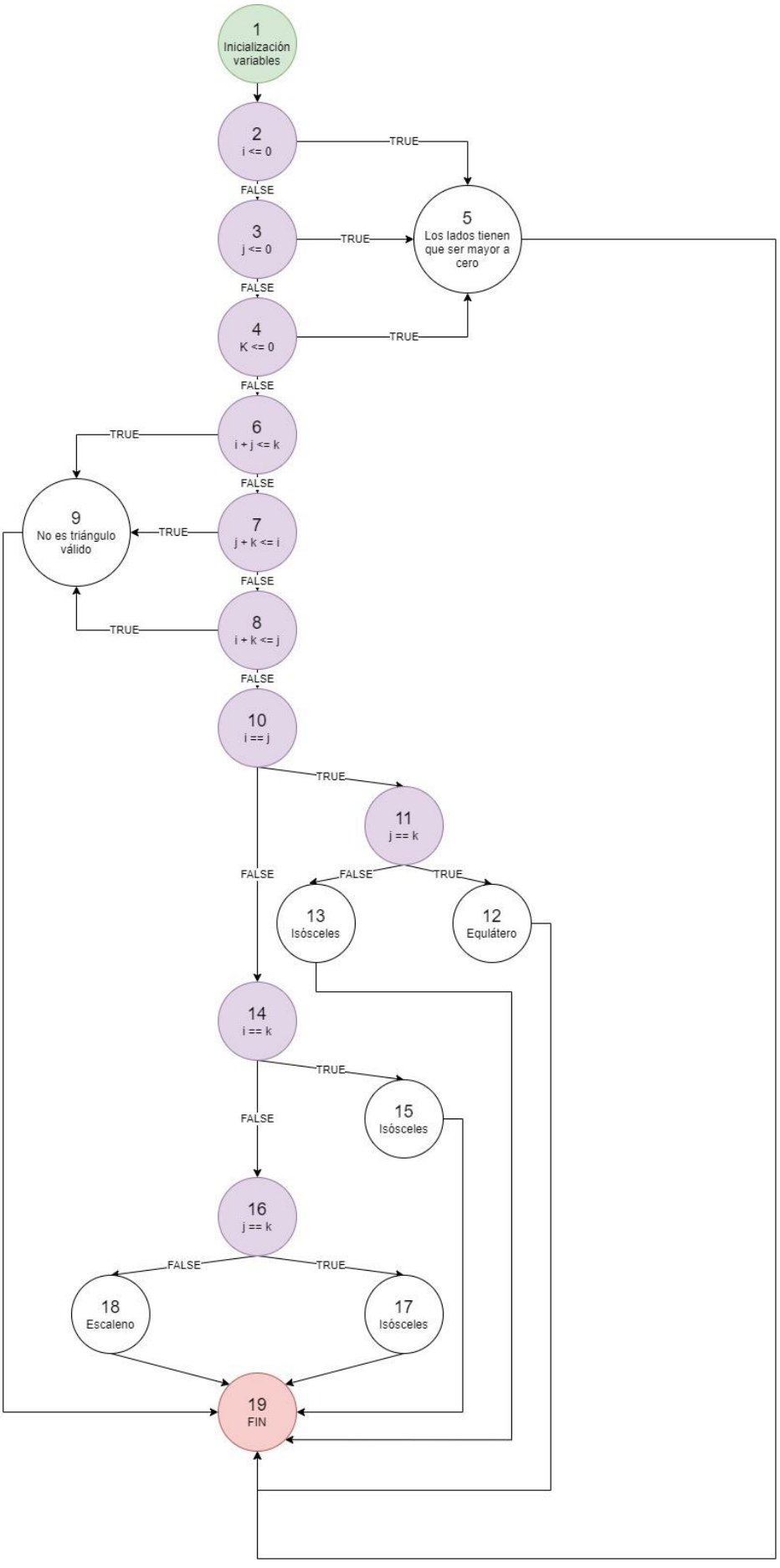
Como primer paso mostramos el código del programa triángulo.

```
public static void triangulito(Scanner in) {
        System.out.print("\nHola! Te voy a pedir 3 numeros.\n");
        System.out.printf("Enter lado a: ");
        int i = in.nextInt();
        System.out.printf("Enter lado b: ");
        int j = in.nextInt();
        System.out.printf("Enter lado c: ");
        int k = in.nextInt();
        if(i <= 0 || j <= 0 || k <= 0) {
            System.out.printf("Los lados tienen que ser mayores a 0.
            return;
        }
        if((i + j) <= k || (j + k) <= i || (i + k) <= j) {
            System.out.printf("No es triangulo valido.");
            return;
        }
        if(i == j) {
            if (j == k) {
                System.out.printf("equilátero");
                return;
            }
            System.out.printf("isósceles");
            return;
        }
        if (i == k) {
            System.out.printf("isósceles");
            return;
        }
        if (j == k) {
            System.out.printf("isósceles");
            return;
        }
        System.out.printf("escaleno");
        return;
    }
```

Grafo

Ahora, a partir del código realizamos el grafo. Usamos el siguiente código de colores:

- Verde: nodo inicial.
- Rojo: nodo final.
- Violeta: nodos predicados.

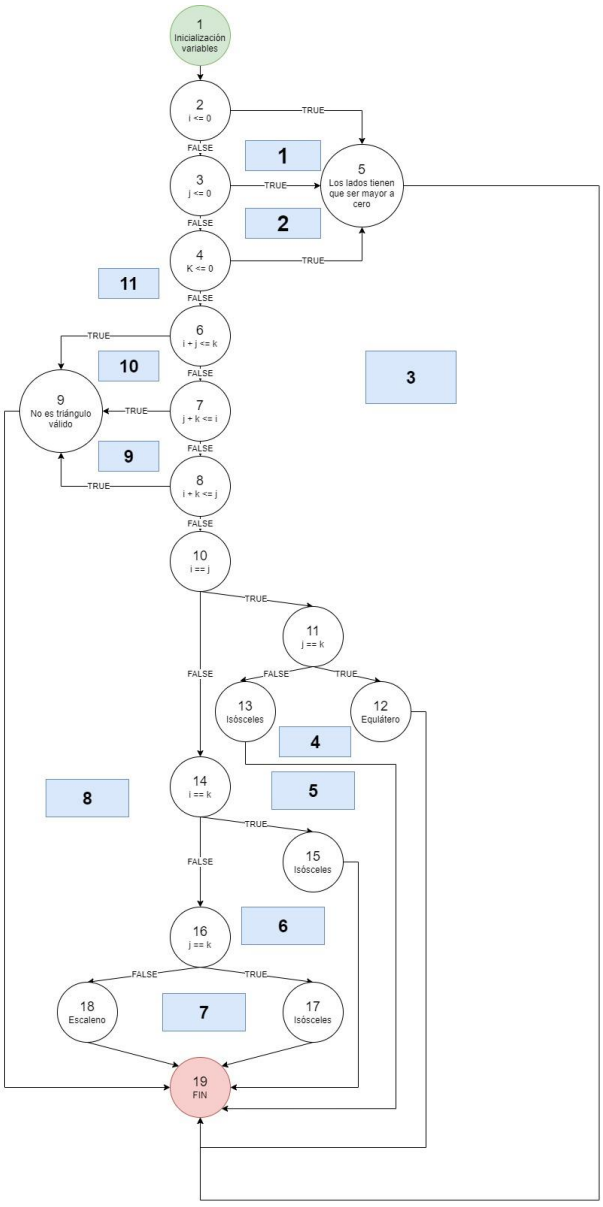

1
Inicialización variables
2
i <= 0
TRUE
FALSE
3
j <= 0
TRUE
FALSE
4
K <= 0
TRUE
FALSE
5
Los lados tienen que ser mayor a cero
6
i + j <= k
TRUE
FALSE
7
j + k <= i
TRUE
FALSE
8
i + k <= j
TRUE
FALSE
9
No es triángulo válido
10
i == j
TRUE
FALSE
11
j == k
FALSE
TRUE
12
Equlátero
13
Isósceles
14
i == k
TRUE
FALSE
15
Isósceles
16
j == k
FALSE
TRUE
17
Isósceles
18
Escaleno
19
FIN

Complejidad Ciclomática

Una vez que definimos el grafo, calculamos la complejidad ciclomática de todas las formas posibles, para verificar que se encuentre correctamente formado.

Forma 1: Regiones del Grafo

En la siguiente imagen observamos todas las regiones marcadas. Tener en cuenta que la región 11 es la externa al grafo.



Concluimos que la complejidad ciclomática es de 11, V(G) = 11.

Forma 2: Aristas y Nodos

En el grafo armado obtenemos que:

- Nodos (N) = 19.
- Aristas (E) = 28

Entonces,

$$V(G) = E - N + 2 = 28 - 19 + 2 = 11$$

Se obtuvo el mismo resultado calculado de las dos formas.

Forma 3: Nodos Predicados

Por último, observamos en el grafo que:

- Nodos Predicados (P) = 10

Entonces,

$$V(G) = P + 1 = 10 + 1 = 11$$

Conclusión: calculamos V(G) de las tres formas y obtuvimos el mismo resultado. Entonces, podremos hallar un conjunto de 11 caminos linealmente independientes.

Caminos Linealmente Independientes

Determinamos el conjunto de 11 caminos linealmente independientes.

- **CAMINO 1**: 1, 2, 5, 19.
- **CAMINO 2**: 1, 2, 3, 5, 19.
- **CAMINO 3**: 1, 2, 3, 4, 5, 19.
- **CAMINO 4**: 1, 2, 3, 4, 6, 9, 19.
- **CAMINO 5**: 1, 2, 3, 4, 6, 7, 9, 19.
- **CAMINO 6**: 1, 2, 3, 4, 6, 7, 8, 9, 19.
- **CAMINO 7**: 1, 2, 3, 4, 6, 7, 8, 10, 11, 12, 19.
- **CAMINO 8**: 1, 2, 3, 4, 6, 7, 8, 10, 11, 13, 19.
- **CAMINO 9**: 1, 2, 3, 4, 6, 7, 8, 10, 14, 15, 19.
- **CAMINO 10**: 1, 2, 3, 4, 6, 7, 8, 10, 14, 16, 17, 19.
- **CAMINO 11**: 1, 2, 3, 4, 6, 7, 8, 10, 14, 16, 18, 19.

Casos de prueba

Con los caminos linealmente independientes, armamos los casos de prueba. Cada camino linealmente independiente tendrá asociado un caso de prueba que fuerce a que se ejecuten las instrucciones especificadas en el camino.

Nro de caso de prueba	Valores de entrada	Resultado Esperado	Resultado Obtenido	Camino
1	I = -1 J = 2 K = 2	"Los lados tienen que ser mayores a cero"	"Los lados tienen que ser mayores a cero"	1
2	I = 2 J = -1 K = 2	"Los lados tienen que ser mayores a cero"	"Los lados tienen que ser mayores a cero"	2
3	I = 2 J = 2 K = -1	"Los lados tienen que ser mayores a cero"	"Los lados tienen que ser mayores a cero"	3
4	I = 10 J = 5 K = 5	"No es triángulo válido"	"No es triángulo válido"	5
5	I = 5 J = 5 K = 10	"No es triángulo válido"	"No es triángulo válido"	4
6	I = 5 J = 10 K = 5	"No es triángulo válido"	"No es triángulo válido"	6
7	I = 5 J = 6 K = 7	"Triángulo es escaleno"	"Triángulo es escaleno"	11
8	I = 2 J = 2 K = 1	"Triángulo es isosceles"	"Triángulo es isosceles"	8
9	I = 2 J = 1 K = 2	"Triángulo es isosceles"	"Triángulo es isosceles"	9
10	I = 1 J = 2 k = 2	"Triángulo es isosceles"	"Triángulo es isosceles"	10
11	I = 1 J = 1 K = 1	"Triángulo es equilatero"	"Triángulo es equilatero"	7

Conclusión

Originalmente, al testear el programa sin aplicar la metodología llegamos a tener 22 casos de pruebas. Aplicando esta metodología llegamos a 11, la mitad de lo original. Si bien en los 22 casos de pruebas iniciales llegamos a cubrir estos 11 caminos, generamos 11 casos extras que son redundantes. De esta forma se gasto tiempos y recursos de forma innecesaria.