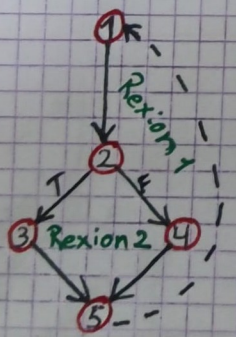
**Diagramas de grafos e complexidade McCabe**

Método calcularDivision

```
1  Public class Division {
   public float calcularDivision (float dividendo, float divisor) throws
   Exception {
2      if (divisor == 0) {
3          throw (new Exception ("Error. O divisor non pode
           ser 0."));
       }
4      float resultado = dividendo / divisor;
       return resultado;
   }
5  }
```




Complexidade:

V(G) = a - n + 2

V(G) = 5 - 5 + 2

V(G) = 2

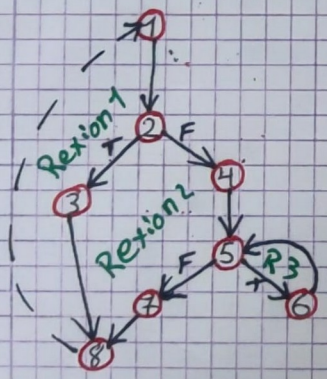
Camiños:

1-> 1-2-3-5

2-> 1-2-4-5

# Método factorial

```
1  Public class Factorial {
     public float factorial (byte n) throws Exception {
2      if (n < 0) {
3        throw new Exception ("Error. O número ten que
         Ser >= 0");
       }
4      float resultado = 1;
5      for (int i = 2; i <= n; i++) {
6        resultado *= i;
       }
7      return resultado;
     }
8  }
```




Complexidade:

$V(G) = a - n + 2$

$V(G) = 9 - 8 + 2$

$V(G) = 3$

Caminos:

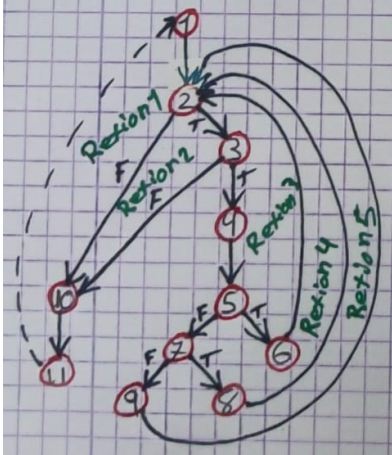
1-> 1-2-3-8

2-> 1-2-4-5-6-5-...

3-> 1-2-4-5-7-8

# Método busca

```
Public class OperacionsArrays {
  Public boolean busca (char c, char[] v) {
    int a, z, i;
1   a = 0;
    z = v.length - 1;
    boolean resultado = false;
    while[2] (a <= z && [3] resultado == false)) {
4     i = (a + z) / 2;
5     if (v[i] == c) {
6       resultado = true;
      }
      else {
7       if (v[i] < c) {
8         a = i + 1;
        }
        else {
9         z = i - 1;
        }
      }
10  }
    return resultado;
11  }
}
```




**Complexidade**

V(G) = a - n + 2

V(G) = 14 - 11 + 2

V(G) = 5

**Caminos**

1-> 1-2-10-11

2-> 1-2-3-10-11

3-> 1-2-3-4-5-6-2...

4-> 1-2-3-4-5-7-8-2...

5-> 1-2-3-4-5-7-9-2...