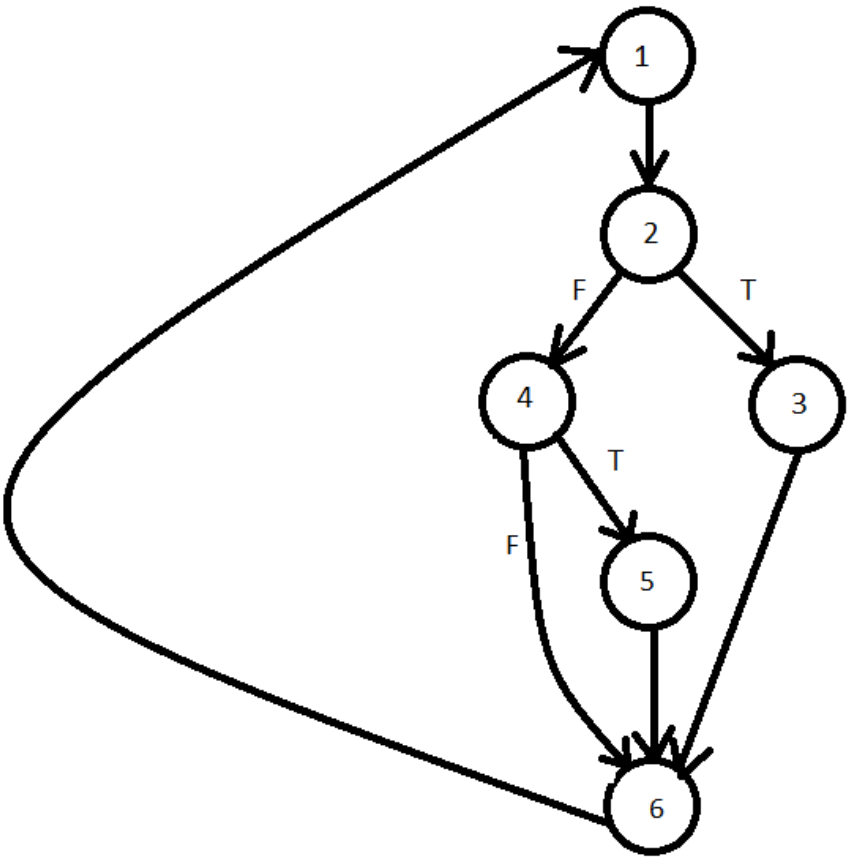
Sergio Otero Castro 1ºDAW

Ejercicios Repaso

- Ejercicio 1:



Complejidad de McCabe: 3

Caminos posibles:

1-2-3-6
1-2-4-6
1-2-4-5-6

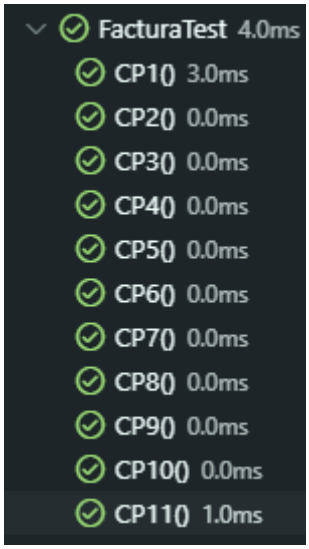
- Ejercicio 2:

```java
public int calcularPrezokWh(){
    int prezokWh=0;
    if (consumokWh==0) {
        return 0;
    }else if (consumokWh < 0){
        throw new NotValidValueException(message: "El consumo supera el límite permitido.");
    }else if (consumokWh<=300){
        prezokWh=9;
    }else if(consumokWh<=600){
        prezokWh=8;
    }else if(consumokWh<=1000){
        prezokWh=6;
    }else if (consumokWh<=2000){
        prezokWh=5;
    }else{
        throw new NotValidValueException(message: "El consumo supera el límite permitido.");
    }
    return prezokWh;
```

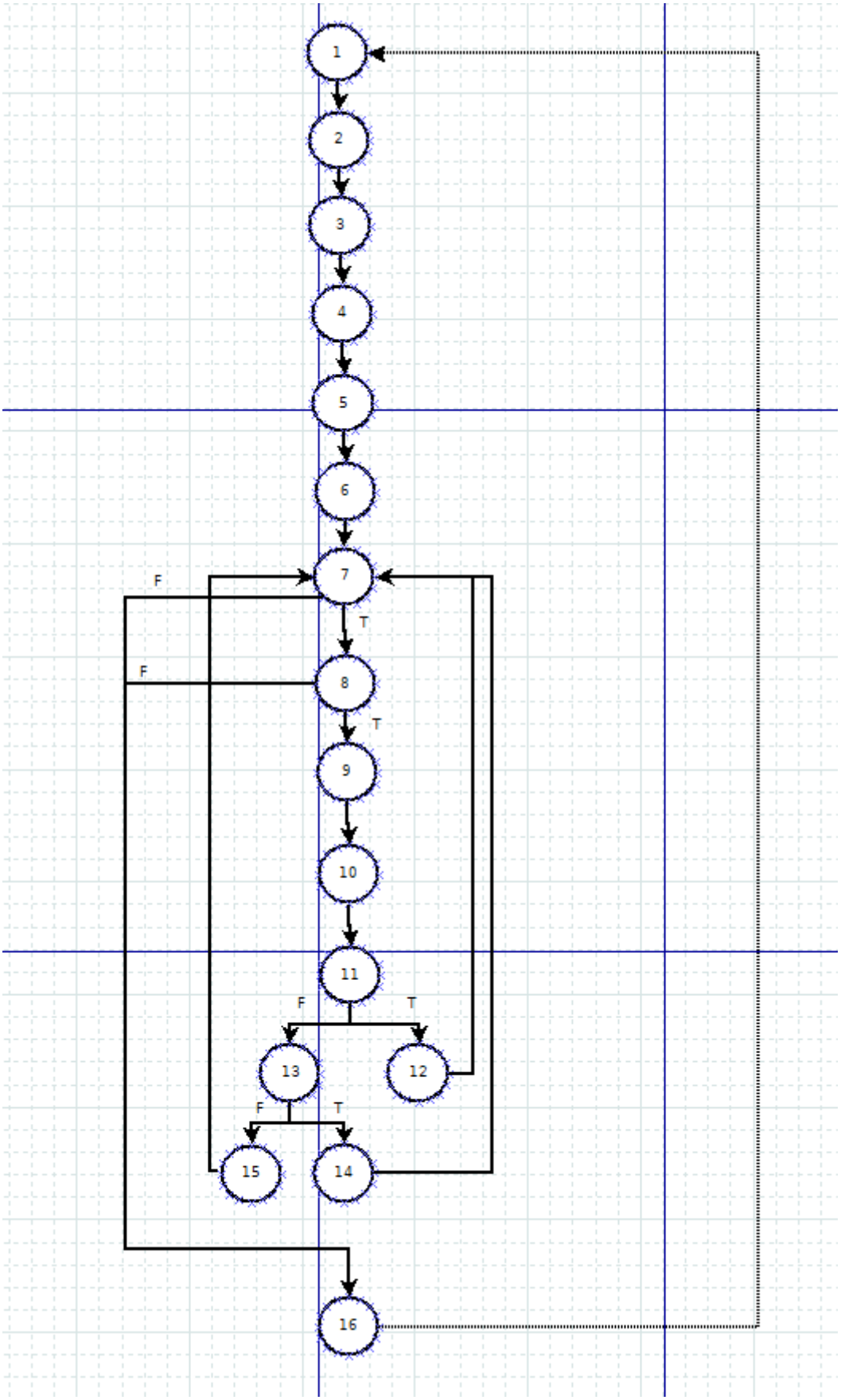
Caso de proba	*Entrada*	*Saída esperada*
CP1	0	0
CP2	300	9
CP3	299	9
CP4	301	8
CP5	600	8
CP6	601	6
CP7	1000	6
CP8	1001	5
CP9	2000	5
CP10	2001	Error
CP11	-1	Error

```java
public class FacturaTest {

    @Test
    public void CP1(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 0);
        assertEquals(expected: 0, f.calcularPrezokWh());
    }

    @Test
    public void CP2(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 300);
        assertEquals(expected: 9, f.calcularPrezokWh());
    }

    @Test
    public void CP3(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 299);
        assertEquals(expected: 9, f.calcularPrezokWh());
    }

    @Test
    public void CP4(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 301);
        assertEquals(expected: 8, f.calcularPrezokWh());
    }

    @Test
    public void CP5(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 600);
        assertEquals(expected: 8, f.calcularPrezokWh());
    }

    @Test
    public void CP6(){
        Factura f = new Factura();
        f.setConsumokWh(consumokWh: 601);
        assertEquals(expected: 6, f.calcularPrezokWh());
    }
```

```java
@Test
public void CP7(){
    Factura f = new Factura();
    f.setConsumokWh(consumokWh: 1000);
    assertEquals(expected: 6, f.calcularPrezokWh());
}

@Test
public void CP8(){
    Factura f = new Factura();
    f.setConsumokWh(consumokWh: 1001);
    assertEquals(expected: 5, f.calcularPrezokWh());
}

@Test
public void CP9(){
    Factura f = new Factura();
    f.setConsumokWh(consumokWh: 2000);
    assertEquals(expected: 5, f.calcularPrezokWh());
}

@Test
public void CP10() throws NotValidValueException{
    Factura f = new Factura();
    f.setConsumokWh(consumokWh: 2001);
}

@Test
public void CP11() throws NotValidValueException{
    Factura f = new Factura();
    f.setConsumokWh(-1);
}
```



- Ejercicio 3:

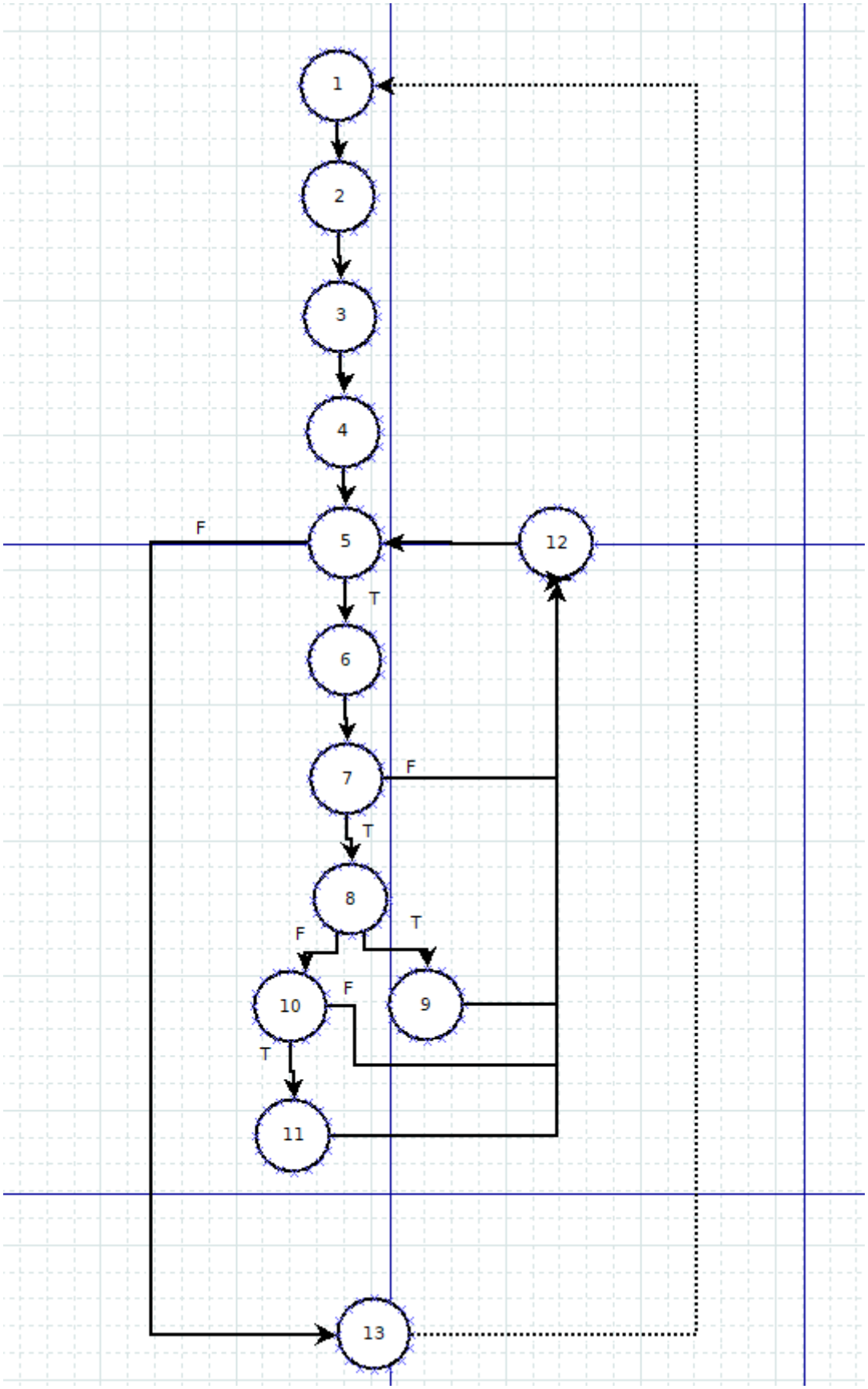


Complejidad de Mccabe: 5

Caminos posibles:

- 1-2-3-4-5-6-7-16
- 1-2-3-4-5-6-7-8-16
- 1-2-3-4-5-6-7-8-9-10-11-12-7-...-16
- 1-2-3-4-5-6-7-8-9-10-11-12-13-14-7-...-16
- 1-2-3-4-5-6-7-8-9-10-11-12-13-15-7-...-16

- Ejercicio 4:



Complejidad de McCabe: 5

Posibles caminos:

- 1-2-3-4-5-13
- 1-2-3-4-5-6-7-12-5-...-13
- 1-2-3-4-5-6-7-8-9-12-5-...-13
- 1-2-3-4-5-6-7-8-10-12-5-...-13
- 1-2-3-4-5-6-7-8-10-11-12-5-...-13

- Ejercicio 5:

```java
public class Calculadora {

    public int suma(int a, int b){
        return a + b;
    }

    public int resta(int a, int b){
        return a - b;
    }

    public int multiplicacion(int a, int b){
        return a * b;
    }

    public int division(int a, int b){

        if (b == 0) {
            throw new IllegalArgumentException(s: "Divisor cannot be zero.");
        }

        return a / b;

    }

}
```

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculadoraTest {

    @Test
    public void CP1(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 9, c.suma(a: 5, b: 4));
    }

    @Test
    public void CP2(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 1, c.suma(a: 5, -4));
    }

    @Test
    public void CP3(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 5, c.resta(a: 5, b: 0));
    }

    @Test
    public void CP4(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 1, c.resta(a: 5, b: 4));
    }

    @Test
    public void CP5(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 7, c.resta(a: 5, -2));
    }

    @Test
    public void CP6(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 20, c.multiplicacion(a: 5, b: 4));
    }
```

```java
    @Test
    public void CP7(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 0, c.multiplicacion(a: 5, b: 0));
    }

    @Test
    public void CP8(){
        Calculadora c = new Calculadora();
        assertEquals(-15, c.multiplicacion(a: 5, -3));
    }

    @Test
    public void CP9(){
        Calculadora c = new Calculadora();
        assertEquals(expected: 2, c.division(a: 5, b: 2));
    }

    @Test (expected = IllegalArgumentException.class)
    public void CP10(){
        Calculadora c = new Calculadora();
        c.division(a: 5, b: 0);
    }

    @Test
    public void CP11(){
        Calculadora c = new Calculadora();
        assertEquals(-3, c.division(a: 9, -3));
    }

}
```