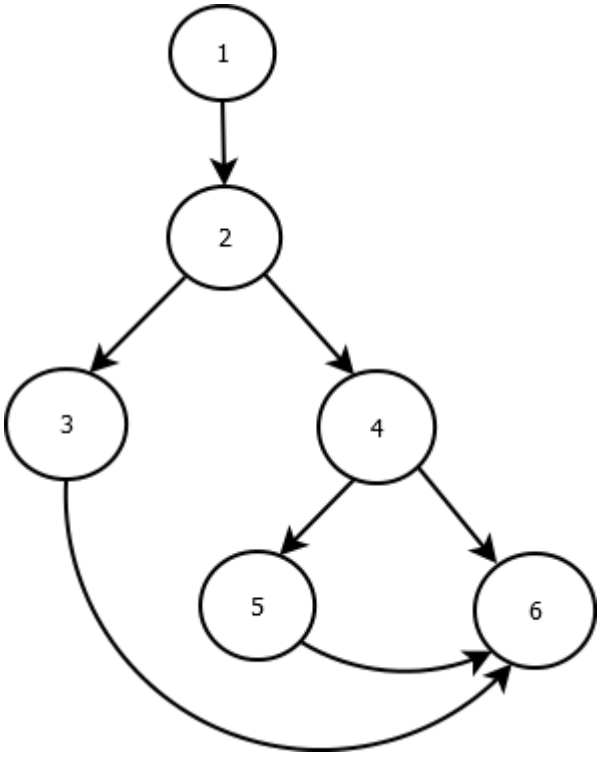
EJERCICIO 1. Consultar positivo



- Cantidad de regiones: 3
- Cantidad de nodos: 7
- Cantidad de aristas: 8
- Complejidad ciclomática:
  a) V(G) = a – n + 2 = 7 – 6+ 2 = 3
  b) V(G) = r = 3
  c) V(G) = c + 1 = 2 + 1 = 3
- Caminos posibles:
  a) 1, 2, 3, 4, 7
  b) 1, 2, 3, 5, 6, 7
  c) 1, 2, 3, 5, 7

EJERCICIO 2. Factura eléctrica

La clase tiene los siguientes problemas:

- Cuando el consumo es igual o menor que 0, es decir, 0 o un valor negativo, va a entender que el precio va a ser de 9 centimos ya que cumple la condicion de <=300, pero lo logico es que el precio sea de 0 centimos.

```
63      @Test
64      public void test7() {
65          Factura factura = new Factura();
66          factura.setConsumokWh(-1);
67          int resultadoEsperado = 0;
68          int resultadoObtenido = factura.calcularPrezokWh();
69          Assertions.assertEquals(resultadoEsperado, resultadoObtenido);
70      }
71      @Test
72      public void test8() {
73          Factura factura = new Factura();
74          factura.setConsumokWh(-1);
75          int resultadoEsperado = 9;
76          int resultadoObtenido = factura.calcularPrezokWh();
77          Assertions.assertEquals(resultadoEsperado, resultadoObtenido);
78      }
79  }
```

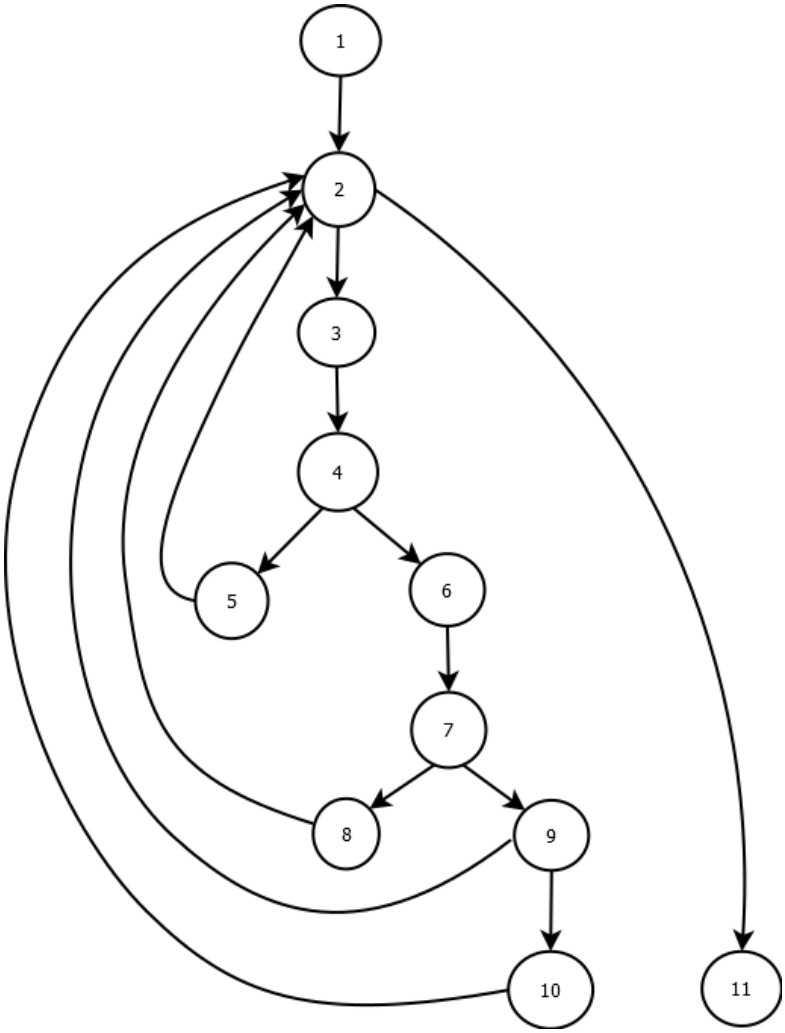
- Cuando el consumo es mayor que 2000, va a entender que el precio va a ser de 0 centimos ya que no cumple ninguna condicion. Lo logico es que a partir de 2000 de consumo, se establezca un precio, por ejemplo 4 centimos.

```
80      @Test
81      public void test9() {
82          Factura factura = new Factura();
83          factura.setConsumokWh(consumokWh: 2500);
84          int resultadoEsperado = 0;
85          int resultadoObtenido = factura.calcularPrezokWh();
86          Assertions.assertEquals(resultadoEsperado, resultadoObtenido);
87      }
88
89      @Test
90      public void test10() {
91          Factura factura = new Factura();
92          factura.setConsumokWh(consumokWh: 2500);
93          int resultadoEsperado = 4;
94          int resultadoObtenido = factura.calcularPrezokWh();
95          Assertions.assertEquals(resultadoEsperado, resultadoObtenido);
96      }
97  }
```

Ahora realizadas las pruebas, corrigo el codigo, añadiendo lo siguiente:

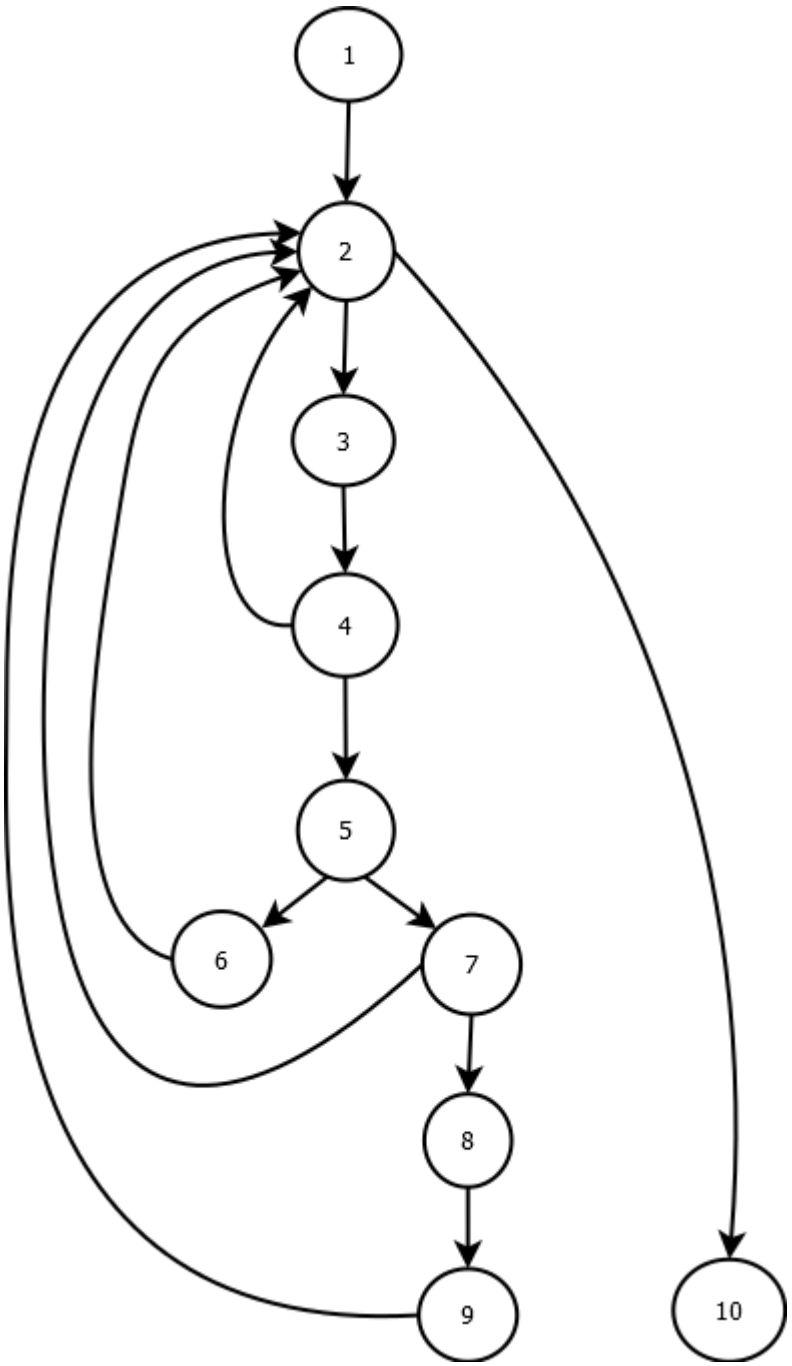
```
public int calcularPrezokWh(){
    int prezokWh=0;
    if(consumokWh<=0){
        return prezokWh;
    }else if (consumokWh<=300){
        prezokWh=9;
    }else if(consumokWh<=600){
        prezokWh=8;
    }else if(consumokWh<=1000){
        prezokWh=6;
    }else if (consumokWh<=2000){
        prezokWh=5;
    }else if(consumokWh>2000){
        prezokWh=4;
    }
    return prezokWh;
}
}
```

EJERCICIO 3. Arrays



- Cantidad de regiones: 5
- Cantidad de nodos: 11
- Cantidad de aristas: 14
- Complejidad ciclomática:
  a) V(G) = a – n + 2 = 14 – 11+ 2 = 5
  b) V(G) = r = 5
  c) V(G) = c + 1 = 4 + 1 = 5
- Caminos posibles:
  a) 1, 2, 11
  b) 1, 2, 3, 4, 5, 2, 11
  c) 1, 2 , 3, 4, 6, 7, 8, 2, 11
  d) 1, 2 , 3, 4, 6, 7, 9, 10, 2, 11
  e) 1, 2 , 3, 4, 6, 7, 9, 11

EJERCICIO 4. Acrónimos



- Cantidad de regiones: 5
- Cantidad de nodos: 10
- Cantidad de aristas: 13
- Complejidad ciclomática:
  a) V(G) = a – n + 2 = 13 – 10 + 2 = 5
  b) V(G) = r = 5
  c) V(G) = c + 1 = 4 + 1 = 5
- Caminos posibles:
  a) 1, 2, 10
  b) 1, 2, 3, 4, 2, 10
  c) 1, 2 , 3, 4, 5, 6, 2, 10
  d) 1, 2 , 3, 4, 5, 7, 2, 10
  e) 1, 2 , 3, 4, 5, 7, 8, 9, 2, 10

EJERCICIO 5. Pruebas unitarias con Junit

Una vez creado el codigo y haciendo comprobaciones, encontre un problema en mi codigo, el cual, al intentar dividir cualquier numero entre el numero divisor “0”, nos da un error. Para solucionar esto creo una excepcion.

Codigo:

```java
package repaso;

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) throws Exception {
        if (b == 0) {
            throw new Exception(message: "No se puede dividir entre cero");
        }
        return a / b;
    }

}
```

Pruebas con JUnit: