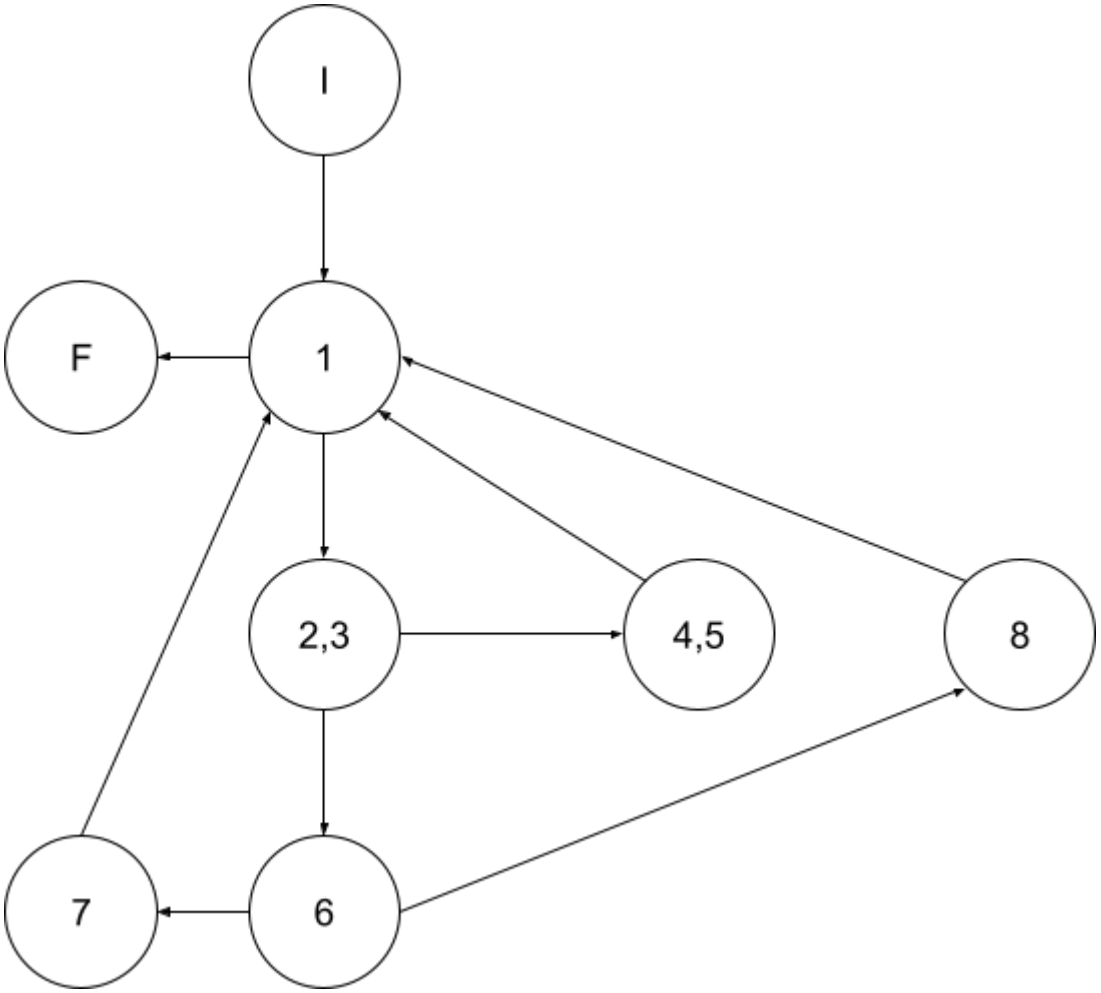
Práctica 1: Pruebas del software

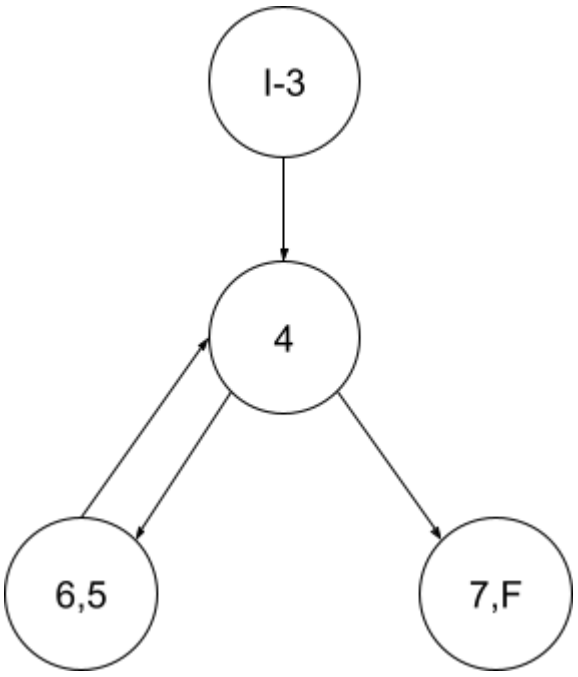
Primero: Grafo1



- Número de nodos: 8
- Número de aristas: 10
- Número de regiones: 4
- Número de nodos predicado: 3
- Complejidad ciclomática: 4
- Secuencia de nodos de todos los caminos del conjunto básico:
  - C1: I,1,2,3,4,5,1,F
  - C2: I,1,2,3,6,7,1,F
  - C3: I,1,2,3,6,8,1,F
  - C4: I,1,F

Segundo: Grafo2. Factorial

```
I     public static int factorial(int n) {
1     int resultado;
2     resultado = 1;
3,4,5 for (int i = 2; i <= n; i++) {
6     resultado = resultado * i;
      }
7     return resultado;
F     }
```

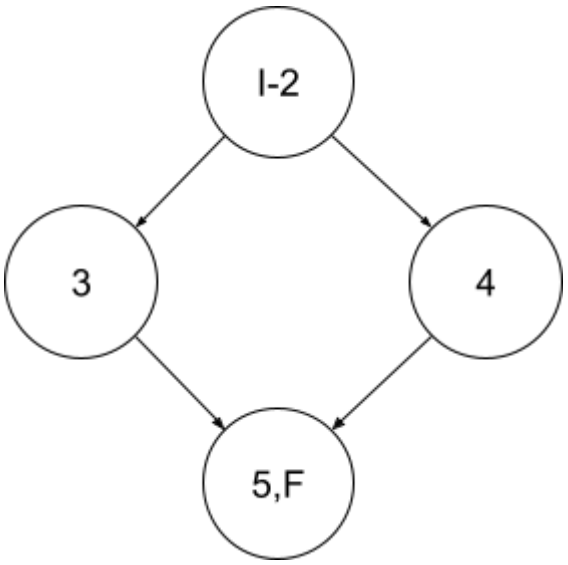


- ➢ Calcula la complejidad ciclomática: 2
- ➢ Evalúa el riesgo: Bajo
- ➢ Define el conjunto básico de caminos:
  - C1: I-3,4,7,F
  - C2: I-3,4,6,5,4,7,F
- ➢ Define un caso de prueba para cada camino (valores de entrada y resultado esperado):

Camino	Valor de n	Resultado esperado
C1	1	1
C2	2	2

Tercero: Grafo3. Divisible

```
I     public boolean divisible(int multiplo, int divisor) {
1     boolean resultado;
2     if (multiplo % divisor == 0) {
3     resultado = true;
      } else {
4     resultado = false;
      }
5     return resultado;
F     }
```

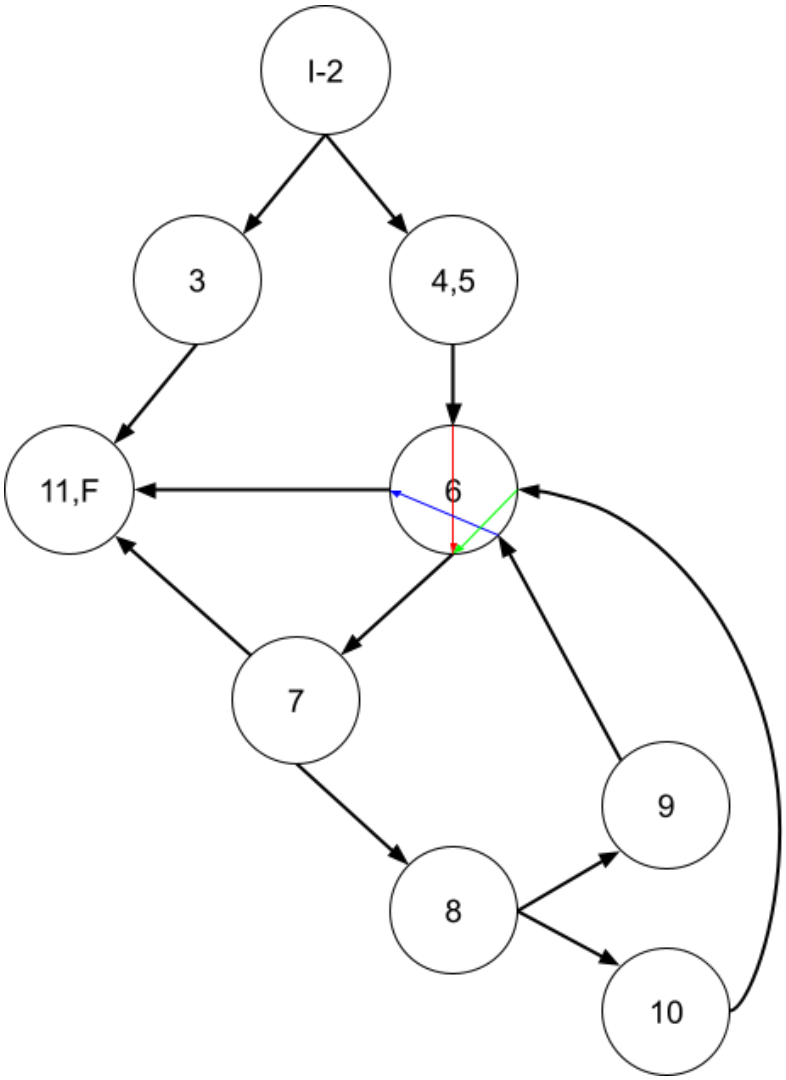


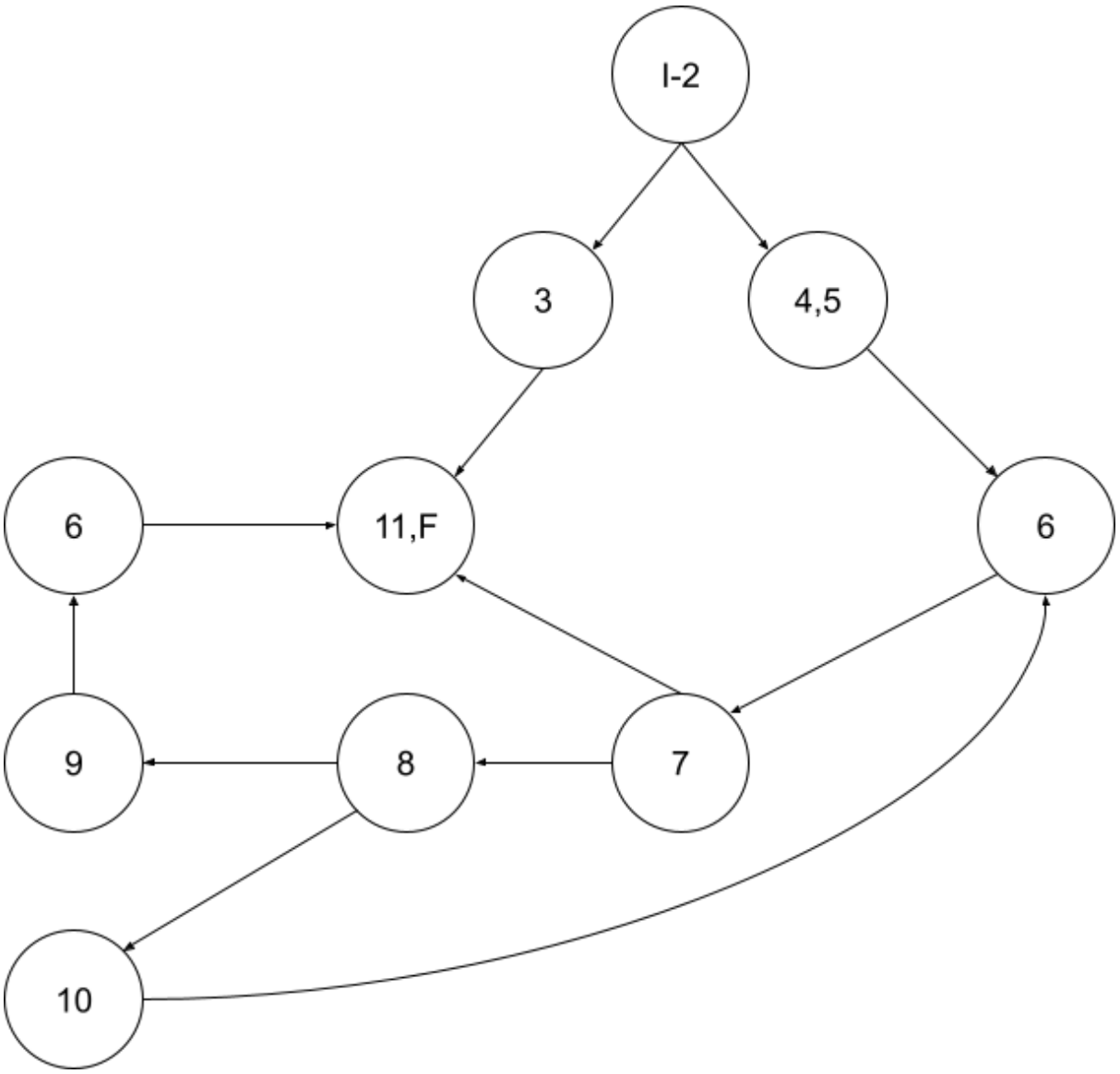
- ➢ Calcula la complejidad ciclomática: 2
- ➢ Evalúa el riesgo: Bajo
- ➢ Define el conjunto básico de caminos:
  - ○ C1: I-2,3,5,F
  - ○ C2: I-2,4,5,F
- ➢ Define un caso de prueba para cada camino (valores de entrada y resultado esperado):

Camino	Valor de multiplo	Valor de divisor	Resultado esperado
C1	4	2	Verdadero
C2	3	7	Falso

Cuarto: Grafo4. esPrimo

```
I       public boolean esPrimo(int n) {
1       boolean primo;
2       if (n <= 1) {
3       primo = false;
        } else {
4       primo = true;
5       int i = 2;
6,7     while (primo && i <= n / 2) {
8       if (divisible(n, i)) {
9       primo = false;
        } else {
10      i++;
        }
        }
        }
11      return primo;
F       }
```





- Calcula la complejidad ciclomática: 4
- Evalúa el riesgo: Bajo
- Define el conjunto básico de caminos:
    - C1: I-2,3,11,F
    - C2: I-2,4,5,6,7,11,F
    - C3: I-2,4,5,6,7,8,9,6,11,F
    - C4: I-2,4,5,6,7,8,10,6,7,11,F
- Define casos de prueba para cada camino (valores de entrada y resultado esperado):

Camino	Valor de n	Resultado esperado
C1	1	false
C2	2	true
C3	4	false
C4	5	true

Quinto: Clases de equivalencia

`factorial(int n)`		
Clases de equivalencia	Valores elegidos	Resultado esperado
n >= 2	n = 4	24
n < 2	n = 1	1
n < 0	n = -3	excepción
n no es un entero	n = 4.23	excepción
n > 2147483647 (mayor que el valor máximo de int)	5147483647	excepción

`divisible(int multiplo, int divisor)`		
Clases de equivalencia	Valores elegidos	Resultado esperado
multiplo es divisible por divisor	multiplo = 15, divisor = 3	true
multiplo no es divisible por divisor	multiplo = 10, divisor = 3	false
multiplo y divisor son negativos	multiplo = -15, divisor = -3	true
multiplo es negativo y divisor es positivo	multiplo = -15, divisor = 3	true
multiplo es positivo y divisor es negativo	multiplo = 15, divisor = -3	true
Divisor es 0	multiplo = 10, divisor = 0	excepción
multiplo y divisor son iguales a cero	multiplo = 0, divisor = 0	excepción

`esPrimo(int n)`		
Clases de equivalencia	Valores elegidos	Resultado esperado
n es primo	n = 7	true
n no es primo	n = 4	false
n = 1	n = 1	false
n < 0	n = -5	excepción
n > 2147483647 (mayor que el valor máximo de int)	n = 5147483647	excepción