



Uso de técnicas de caja negra y blanca

Alejandra Milena Orrego Higueta

Institución Universitaria Tecnológico de Antioquia

Facultad de ingeniería

Pruebas de software

Medellín-Antioquia.

Marzo 2023

## Contenido

<b>Introducción .....</b>	<b>3</b>
<b>Tecnicas de caja Negra .....</b>	<b>3</b>
<b>Particion de Equivalencias .....</b>	<b>6</b>
<b>Análisis de valores Limiter.....</b>	<b>8</b>
<b>Diagrama de transición de estados.....</b>	<b>10</b>
<b>Tablas de decisión .....</b>	<b>12</b>
<b>Tecnicas de caja Blanca.....</b>	<b>14</b>
<b>Cobertura de sentencias. ....</b>	<b>16</b>
<b>Cobertura de decisión.....</b>	<b>18</b>
<b>Cobertura de Caminos .....</b>	<b>22</b>
<b>Pruebas de Condición y Cobertura .....</b>	<b>26</b>

## Introducción

En este proyecto se hará uso de las técnicas de Caja negra y caja blanca para realizar pruebas de software a 3 requisitos funcionales.

Se hará uso de 3 técnicas de caja negra las cuales son:

***Partición de equivalencia*** esta técnica permite examinar los valores válidos e inválidos de las entradas existentes en el software, de esta manera se detectan los errores mucho más eficientemente.

***Análisis de valores límite*** en esta técnica los casos de prueba son diseñados basándose en los valores límite.

***Tablas de decisión*** también conocida como tabla de causa o efecto es una técnica de enfoque sistemático en el que varias combinaciones de entrada y su respectivo comportamiento del sistema se capturan en forma de tabla.

En cuanto las técnicas de caja blanca se usar 4 las cuales son:

***Cobertura de Sentencias:*** Comprueba que todas las sentencias se ejecuten al menos una vez.

***Cobertura de Decisión:*** Ejecuta casos de prueba de modo que cada decisión se pruebe al menos una vez a Verdadero (True) y otra a Falso (False).

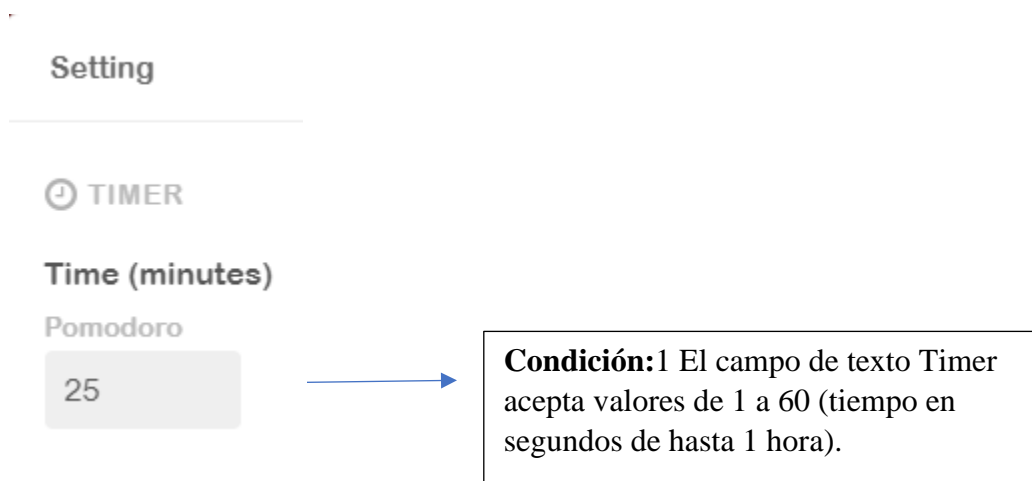
***Cobertura de Caminos:*** Se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa.

***Pruebas de condición y cobertura*** es esta tenemos tanto la cobertura de condición simple y la cobertura de decisión múltiple.

Estos son los requisitos funcionales que pasaran por las pruebas de caja negra y blanca.

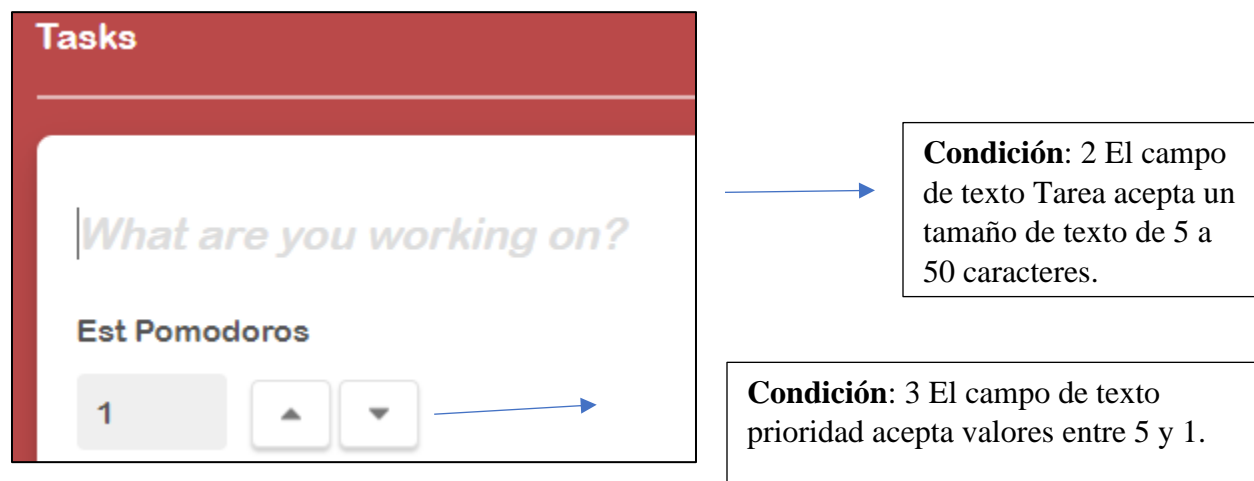
### Requisito funcional #1

Se debe tener el tiempo en minutos de cada pomo doro este puede ser por default o ingresado por el usuario.



### Requisito funcional #2

Se requiere un botón para agregar tareas y que esas tareas se listen según la prioridad indicada donde 1 es alta prioridad y 5 sin prioridad, también es necesario que la tarea tenga caracteres.



Como podemos observar tenemos 3 requisitos funcionales a los cuales se le crearan clases validas e invalidas para así reducir al máximo los casos de prueba ya que con estas clases tendremos cubiertas todas las posibilidades.

### Partición de equivalencias

Partición de equivalencias		
Condición de entrada	Clases válidas	Clases Inválidas
Tiempo en minutos del pomo doro Números de 1 a 60	1) $1 \leq \text{Tiempo} \leq 60$	2) $\text{Tiempo} \leq 0$ 3) $\text{Tiempo} \geq 61$ 4) Tiempo no es número
Prioridad de tareas Números de 1 a 5	5) $1 \leq \text{Prioridad} \leq 5$	6) $\text{Prioridad} \leq 0$ 7) $\text{Prioridad} \geq 6$ 8) Prioridad no es número
Texto con tarea El número de caracteres permitido es de 5 a 50	9) $5 \leq \text{Tamaño texto} \leq 50$	10) $\text{Texto} \geq 51$ 11) $\text{Texto} \leq 4$

### Casos de prueba para el pomo doro

Tiempo de 1 a 60
Prioridad de 1 a 5
Tamaño texto de 5 a 50

TC	Variable	Clase de equivalencia	Estado	Dato
*	Tiempo	EC1: $1 \leq \text{Tiempo} \leq 60$	Válido	25
		EC2: $\text{Tiempo} \leq 0$	No válido	-20
		EC3: $\text{Tiempo} \geq 61$	No válido	80
		EC4: Tiempo no es número	No válido	'cien'
*	Prioridad	EC5: $1 \leq \text{Prioridad} \leq 5$	Válido	3
		EC6: $\text{Prioridad} \leq 0$	No válido	0
		EC7: $\text{Prioridad} \geq 6$	No válido	10
		EC8: Prioridad no es número	No válido	'Alta'
*	Tarea	EC9: $5 \leq \text{Tamaño texto} \leq 50$	Válido	30
		EC10: $\text{Texto} \geq 51$	No válido	66
		EC11: $\text{Texto} \leq 4$	No válido	2

### Comparación CE valida con invalida

#### Casos de prueba Requisito #1

Variable	Clase de equivalencia	Estado	Dato	TC1	TC2	TC3	TC4
Tiempo	EC1: 1=< Tiempo <=60	Válido	25	*			
	EC2: Tiempo <= 0	No válido	-20		*		
	EC3: Tiempo >= 61	No válido	80			*	
	EC4: Tiempo no es número	No válido	'cien'				*

#### Casos de prueba requisito #2

Variable	Clase de equivalencia	Estado	Dato	TC1	TC2	TC3	TC4	TC5	TC6
Prioridad	EC5: 1=< Prioridad <=5	Válido	3	*	*	*			
	EC6: Prioridad <= 0	No válido	0				*		
	EC7: Prioridad >= 6	No válido	10					*	
	EC8: Prioridad no es número	No válido	'Alta'						*
Tarea	EC9: 5=< Tamaño texto <=50	Válido	30	*			*	*	*
	EC10: Texto >=51	No válido	66		*				
	EC11: Texto <=4	No válido	2			*			

## Análisis de valor limite

### Casos de prueba para el pomo doro

análisis valores Limites	Valores validos	Valores inválidos
	0,00;0,01; 59,00;60,00	-0,01; 61,00

TC	Variable	Clase de equivalencia	Estado	Dato
*	Tiempo	EC1: 1=< Tiempo <=60	Válido	1
		EC2: Tiempo <= 0	No válido	0
		EC3: Tiempo >= 61	No válido	61
		EC4: Tiempo no es número	No válido	“hola”
*	Prioridad	EC5: 1=< Prioridad <=5	Válido	5
		EC6: Prioridad <= 0	No válido	0
		EC7: Prioridad >= 6	No válido	6
		EC8: Prioridad no es número	No válido	“hola”
*	Tarea	EC9: 5=< Tamaño texto <=50	Válido	50
		EC10: Texto >=51	No válido	51
		EC11: Texto <=4	No válido	4



### Comparación CE valida con invalida

#### Casos de prueba Requisito #1

Variable	Clase de equivalencia	Estado	Dato	TC1	TC2	TC3	TC4
Tiempo	EC1: 1=< Tiempo <=60	Válido	60	*			
	EC2: Tiempo <= 0	No válido	0		*		
	EC3: Tiempo >= 61	No válido	61			*	
	EC4: Tiempo no es número	No válido	'cien'				*

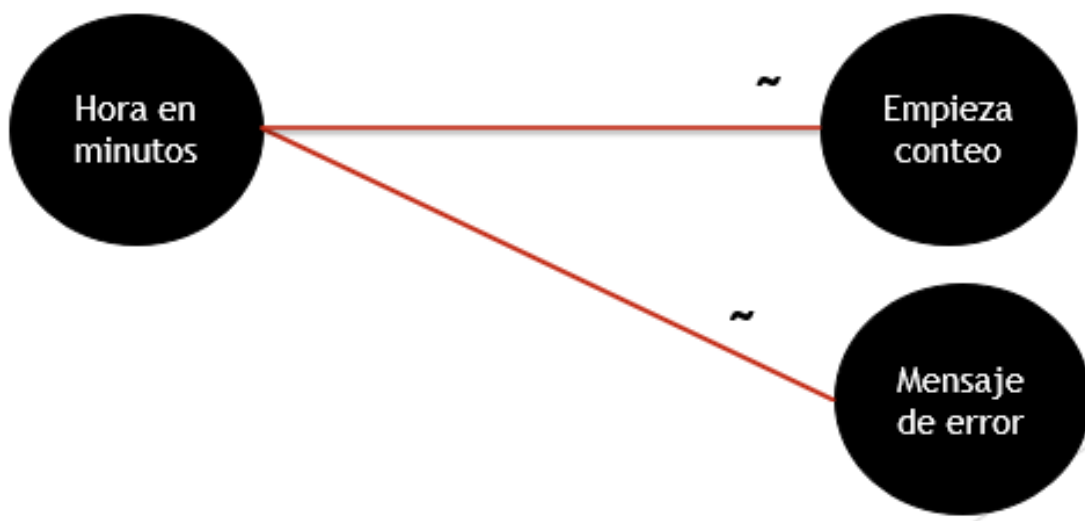
#### Casos de prueba requisito #2

Variable	Clase de equivalencia	Estado	Dato	TC1	TC2	TC3	TC4	TC5	TC6	TC7
Prioridad	EC5: 1=< Prioridad <=5	Válido	1	*						
	EC6: Prioridad <= 0	No válido	0,00		*					
	EC7: Prioridad >= 6	No válido	5,99			*				
	EC8: Prioridad no es número	No válido	'Alta'				*			
Tarea	EC9: 5=< Tamaño texto <=50	Válido	50					*		
	EC10: Texto >=51	No válido	50,99						*	
	EC11: Texto <=4	No válido	3,99							*

## Tablas de decisión

### Requisito #1

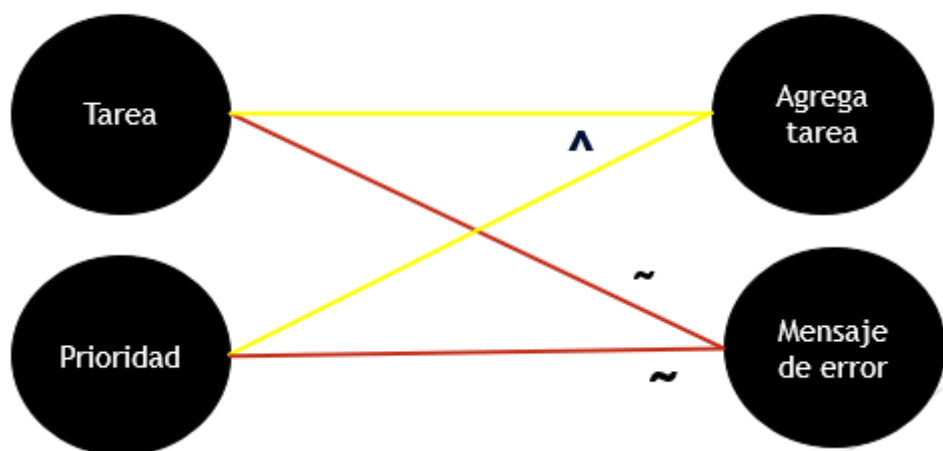
Ingresa la hora si la hora es correcta empieza el conteo y sino mostrara mensaje de error



		TC1	TC2
Precondiciones (causas)	Hora	T	F
Actividades (efectos)	Inicio conteo	T	F
	Mensaje de error	F	T

**Requisito #2**

Ingresa el nombre tarea y la prioridad si son válidos inicia conteo y sino mensaje de error

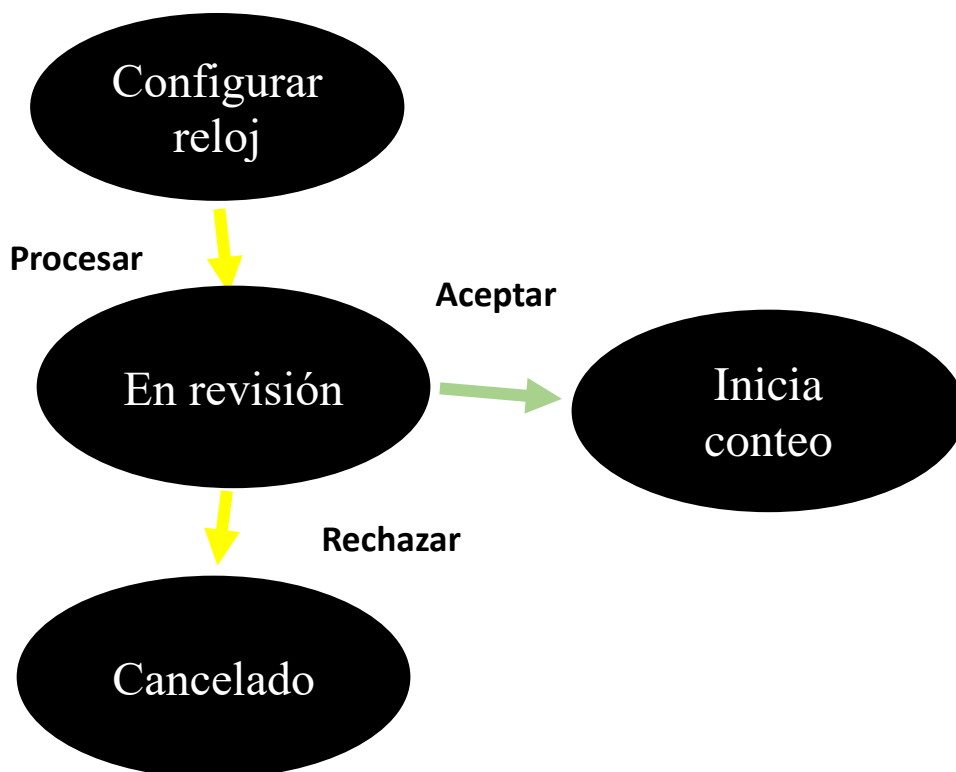


		<i>TC1</i>	<i>TC2</i>	<i>TC3</i>	<i>TC4</i>
<b>Precondiciones (causas)</b>	<i>Tarea</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>
	<i>Prioridad</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<b>Actividades (efectos)</b>	<i>Agrega Tarea</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
	<i>Mensaje de error</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>

### Diagrama Transición de estados

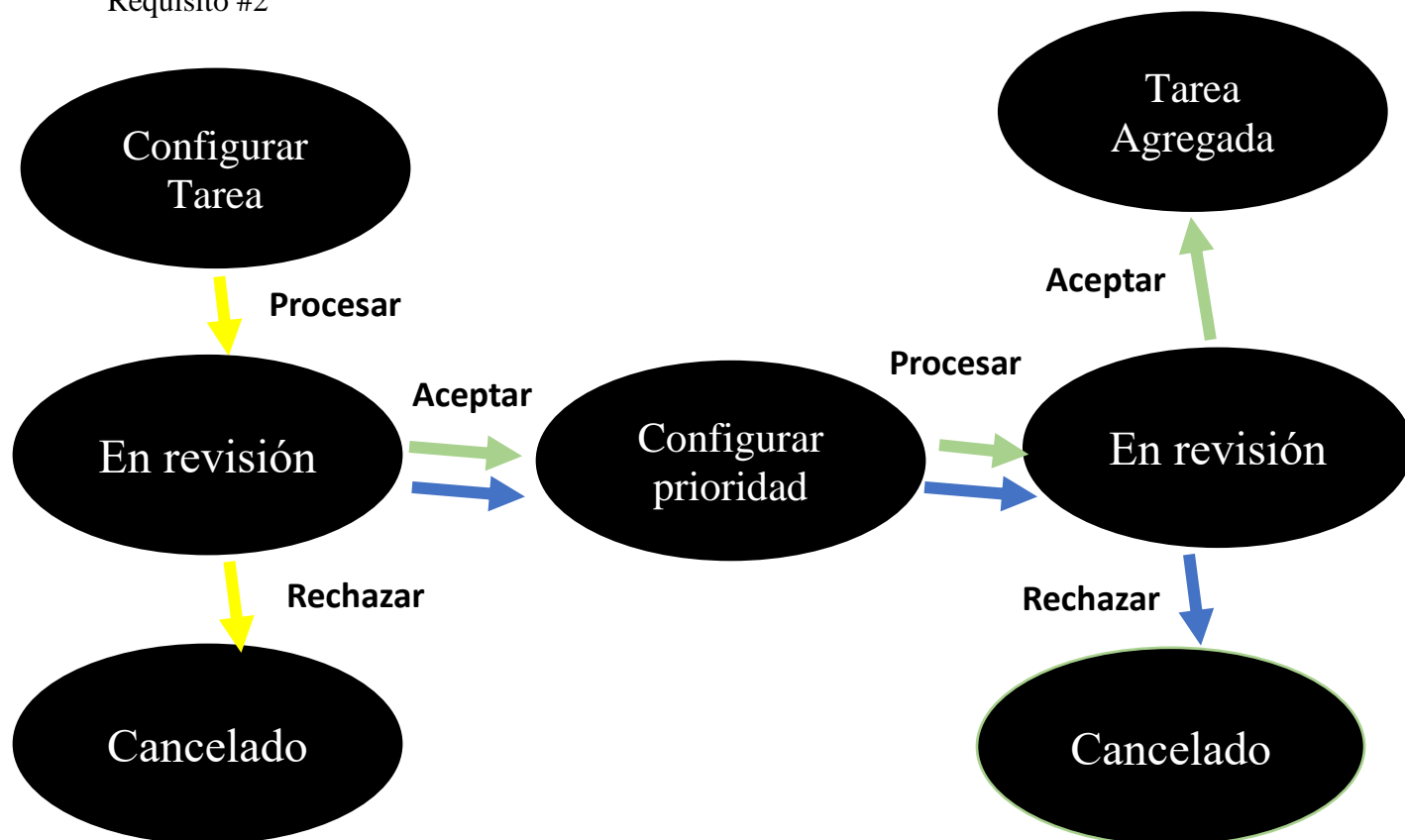
Aplicación pomo doro

Requisito #1



Caso de prueba	Estado 1	Estado 2	Estado 3	Estado final
TC1	Configurar reloj	En revisión	Cancelado	Cancelado
TC2	Configurar reloj	En revisión	Inicia Conteo	Inicia conteo

Requisito #2



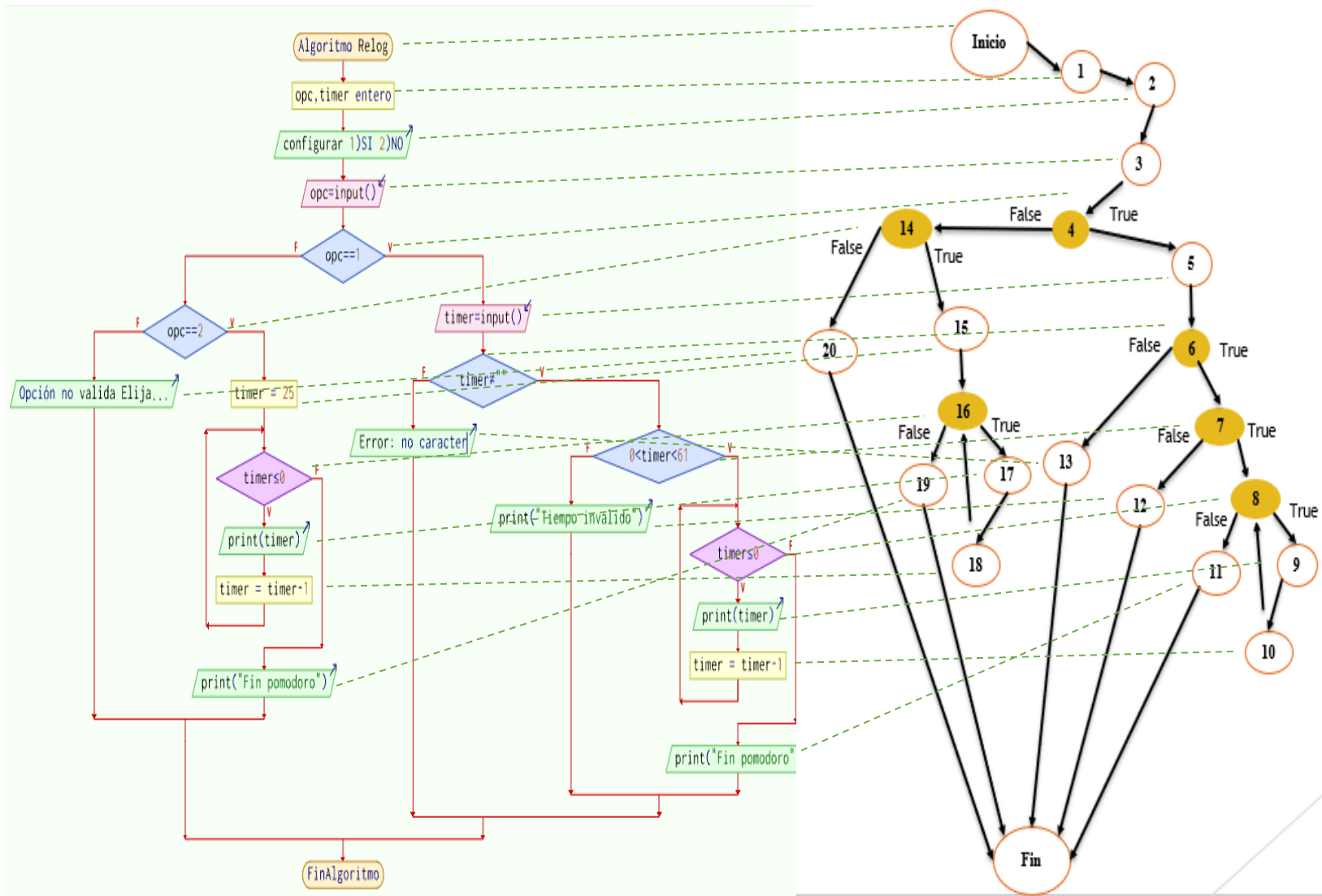
Caso de prueba	Estado 1	Estado 2	Estado 3	Estado 4	Estado 5	Estado final
TC1	Configurar tarea	En revisión	Cancelado			Cancelado
TC2	Configurar tarea	En revisión	Configurar prioridad	En revisión	Tarea Agregada	Tarea agregada
TC3	Configurar tarea	En revisión	Configurar prioridad	En revisión	Cancelado	Cancelado

## Técnicas de caja Negra

### Algoritmo requisito # 1 Python

```
reloj.py > ...
1  def relog(timer):
2      #ponemos el reloj en marcha
3      if (timer>0 and timer<61):
4          #empieza a correr conteo de forma desendente
5              while (timer>=0):
6                  #inicia conteo
7                  print(str(timer) + " minutos")#conteo minutos
8                  timer=timer -1;
9      else:
10         print("tiempo incorrecto")
11
12 print("¿Deseas configurar el reloj? (Responde con número)\n1) SI\n2) NO")
13 try:
14     opc=int(input())
15     if(opc==1):#Cambiar timer
16         #evaluamos que el tiempo sea correcto
17         timer=int(input("Ingresa el Tiempo en minutos: "))
18         relog(timer)
19     elif(opc==2):#Opción no valida
20         #tiempo por defecto
21         timer=int(25)
22         print("Tiempo por defecto 25 minutos")
23         relog(timer)
24     else:{
25         print("Opción no valida elija (1 ó 2) ")
26     }
27 except:
28     print("Error: No ingrese Caracteres")
29
```

## Diagrama de flujo y diagrama de grafo



## Complejidad Ciclomática

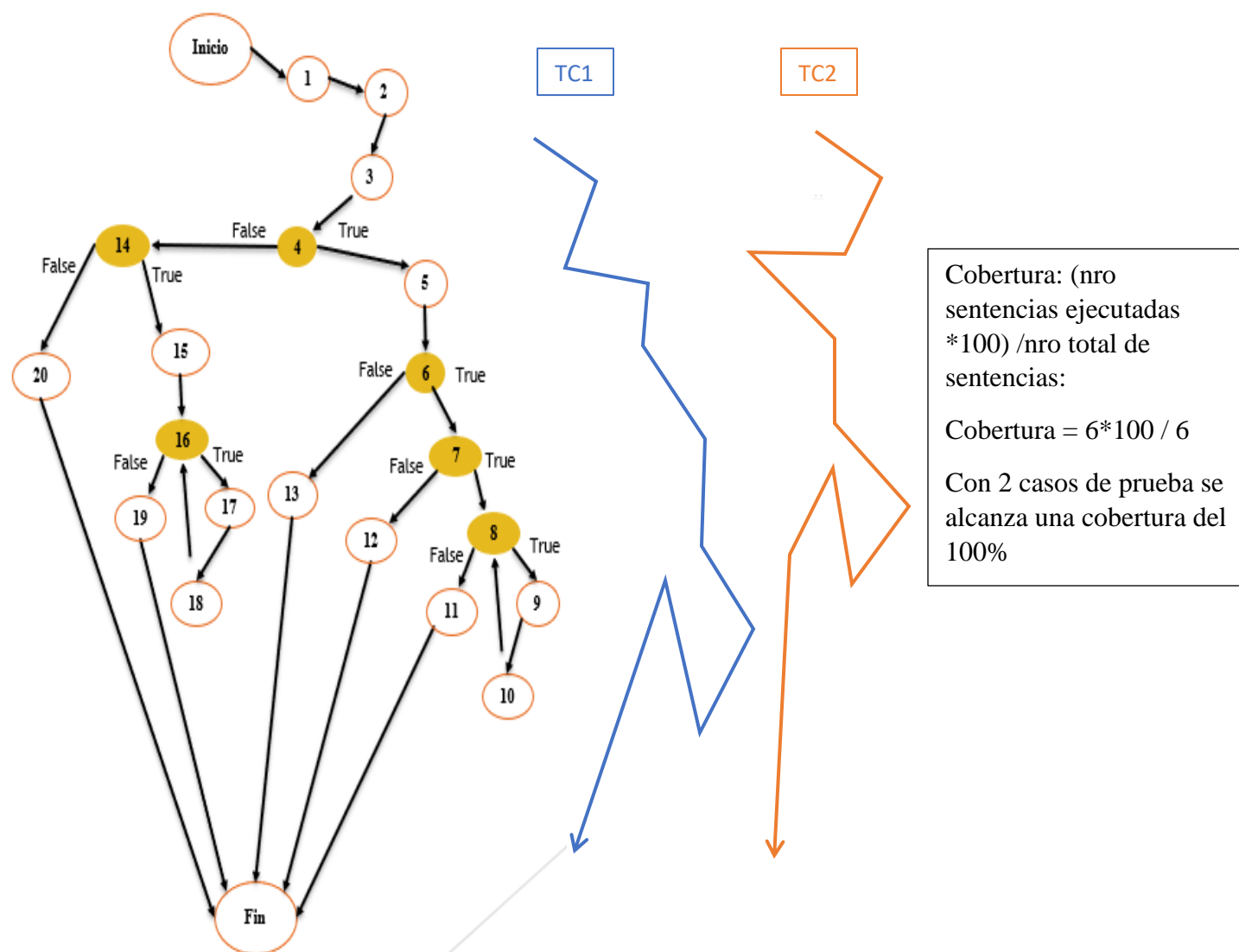
$$V(G) = P + 1$$



P = número de nodos predicados contenidos en el grafo: 6

$$V(G) = 6 \text{ nodos predicados} + 1 = 7$$

$$\text{Complejidad ciclomática} = 7$$

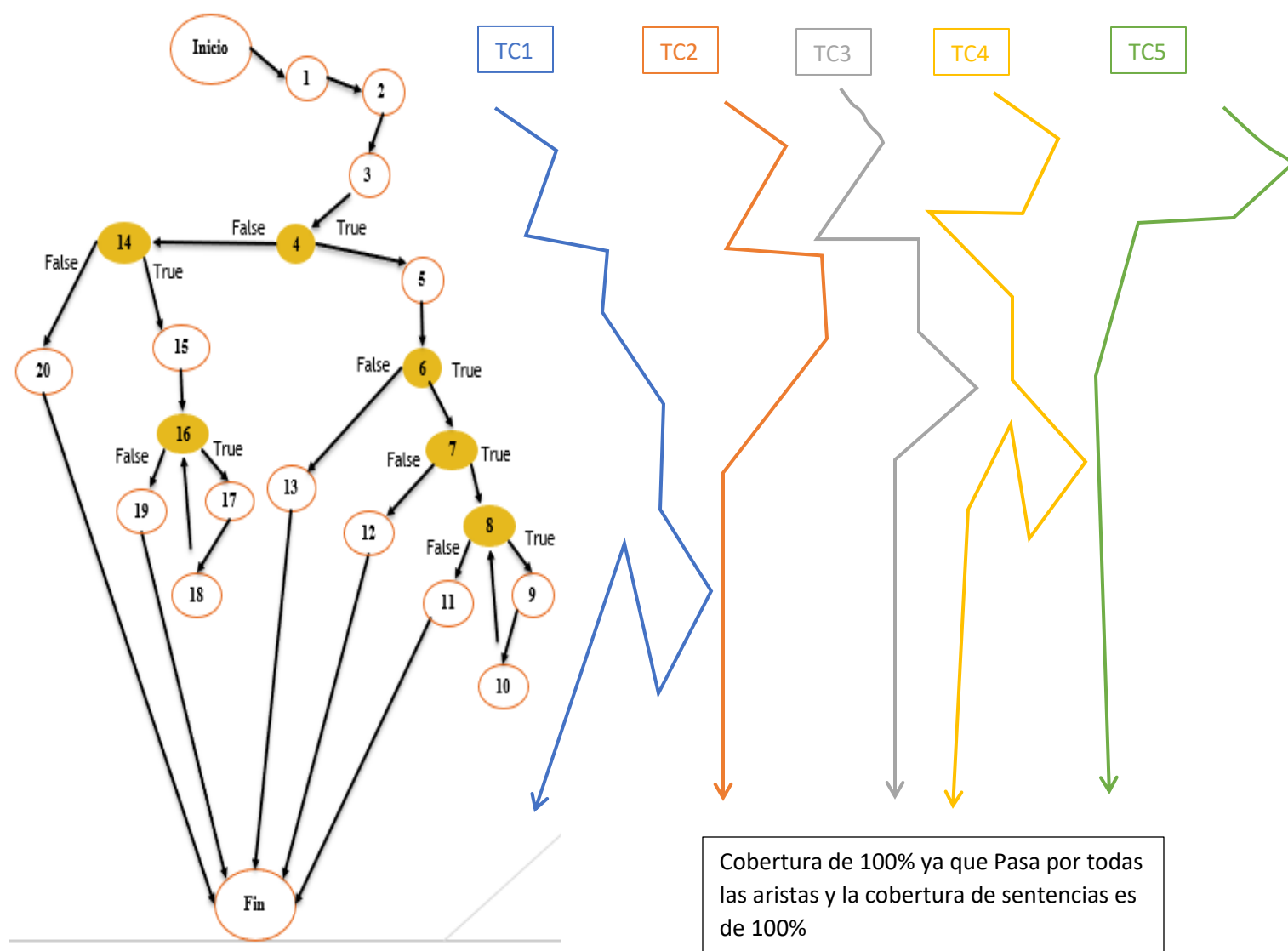
## Técnica Cobertura de sentencias





TC	Camino	Entrada	Prueba	Salida
<b>TC1</b>	1,2,3,4,5,6,8,9,10,11, F	Opc==1; True Timer != " "; True Timer<=0; True Timer<=0; False	Opc=1 Timer=30	30-1 hasta que llegue a Cero. 
<b>TC2</b>	1,2,3,4,14,15,16,17,18,19, F	Opc==2; True Timer<=0; True Timer<=0; False	Opc=2 Timer=25	25-1 hasta que llegue a Cero. 



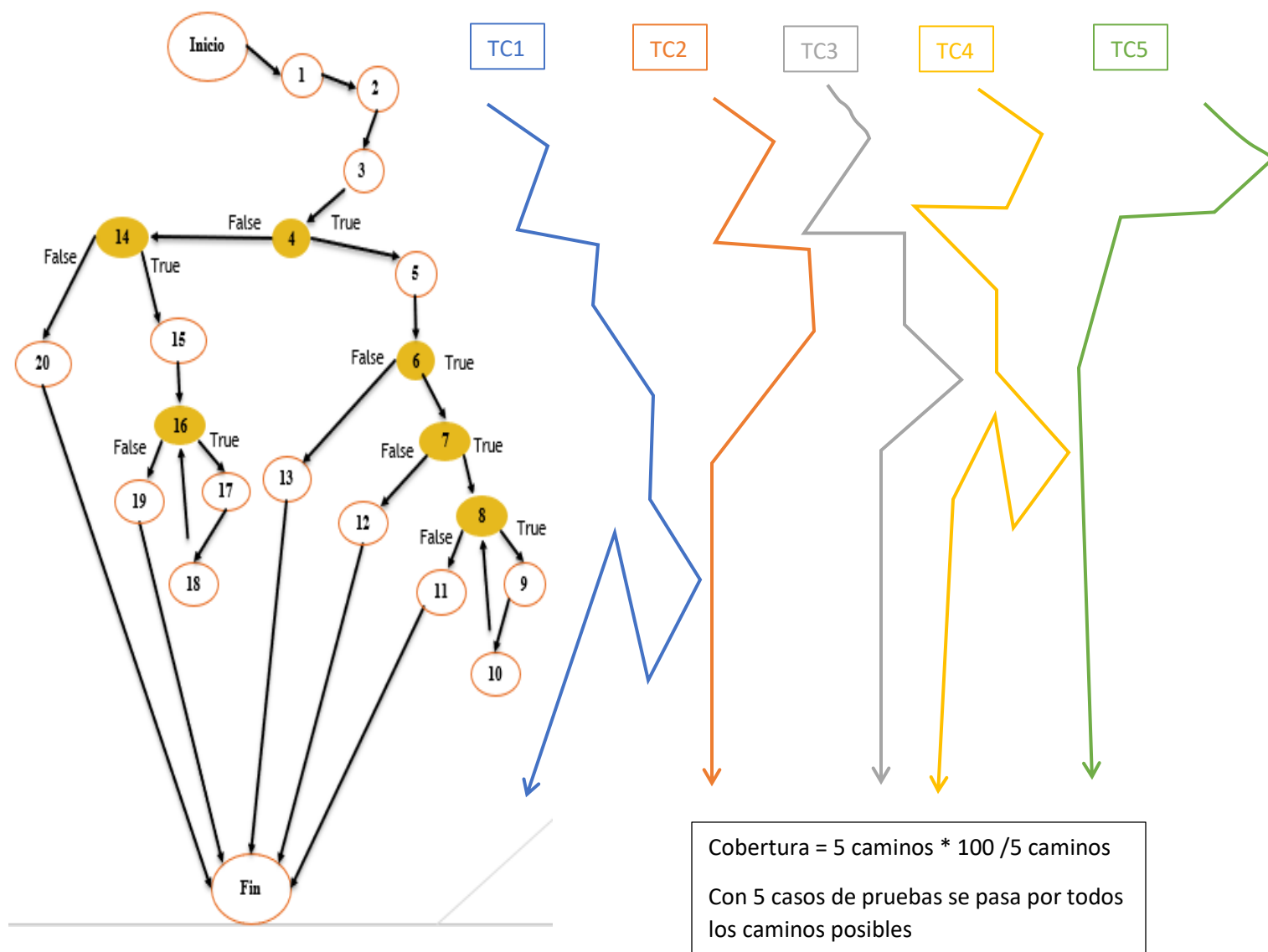
## Técnica Cobertura de decisión





TC	Camino	Entrada	Prueba	Salida
TC1	1,2,3,4,5,6,8,9,10,11, F	Opc==1; True Timer != " "; True Timer<=0; True Timer<=0; False	Opc=1 Timer=30	30-1 hasta que llegue a Cero. 
TC2	1,2,3,4,5,6,13, F	Opc==1; True Timer != " "; False	Opc=1 Timer='doce'	Error El tiempo no puede ser Carácter.
TC3	1,2,3,4,5,6,7,12, F	Opc=1; True 0<Timer<61; False	Opc=1 Timer= 70	Error Tiempo invalido

<b>TC4</b>	1,2,3,4,14,15,16,17,18,19, F	Opc==2; True Timer<=0; True Timer<=0; False	Opc=2 Timer=25	25-1 hasta que llegue a Cero. 
<b>TC5</b>	1,2,3,4,14,20, F	Opc==2; False	Opc=3	Opción no valida

### Técnica cobertura de caminos



TC	Camino	Entrada	Prueba	Salida
<b>TC1</b>	1,2,3,4,5,6,8,9,10,11, F	Opc==1; True Timer != " "; True Timer<=0; True Timer<=0; False	Opc=1 Timer=30	30-1 hasta que llegue a Cero. 

<b>TC2</b>	1,2,3,4,5,6,13, F	Opc==1; True Timer 1=" "; False	Opc=1 Timer='doce'	Error El tiempo no puede ser Carácter.
<b>TC3</b>	1,2,3,4,5,6,7,12, F	Opc=1; True 0<Timer>61; False	Opc=1 Timer= 70	Error Tiempo invalido
<b>TC4</b>	1,2,3,4,14,15,16,17,18,19, F	Opc==2; True Timer<=0; True Timer<=0; False	Opc=2 Timer=25	25-1 hasta que llegue a Cero. 
<b>TC5</b>	1,2,3,4,14,20, F	Opc==2; False	Opc=3	Opción no valida

### Prueba de condición y cobertura

Test Case	Opción	Tiempo	
<b>TC1</b> Opc=1 Timer=30	Opc=1(True)	Timer !=" "(True) and Timer<61 and Timer>0(True)	Opc=1 and Timer=30 (True)
<b>TC2</b> Opc=1 Timer='doce'	Opc=1(True)	Timer !=" "(False) and Timer<61 and Timer>0(False)	Opc=1 and Timer="dice"(False)
<b>TC3</b> Opc=1 Timer=70	Opc=1(True)	Timer !=" "(True) and Timer<61 and Timer>0(False)	Opc=1 and Timer=70 (False)
<b>TC4</b> Opc=2	Opc=1(False) Opc=2(True)	Timer<61 and Timer>0(True)	Opc=2 and Timer=25 (True)
<b>TC5</b> Opc=3	Opc=1(False) Opc=2(False)	*	Opc=3 (False)

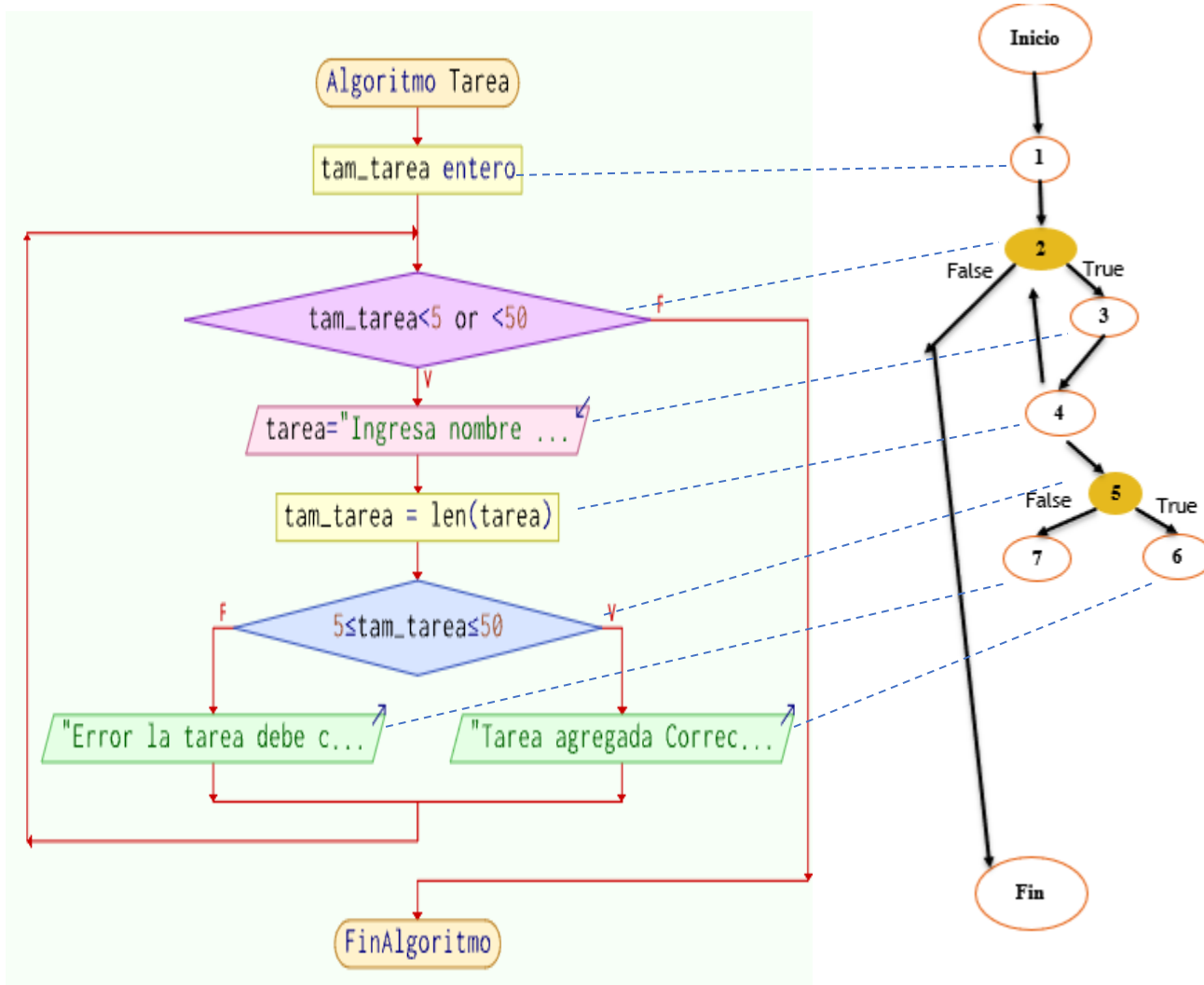
## Algoritmo requisito # 2

```

tarea.py > ...
1  #pedir una tarea
2  tam_tarea=0
3  while tam_tarea<5 or tam_tarea>50:
4      tarea=input("Ingresa el nombre de la tarea: ")
5      tam_tarea=len(tarea)
6      if tam_tarea<=50 and tam_tarea>=5:
7          print("Tarea agregada correctamente")
8      else:
9          print("Error La tarea debe contener entre 5 y 50 caracteres")
10

```

## Diagrama de flujo y diagrama de grafo



## Complejidad Ciclomática

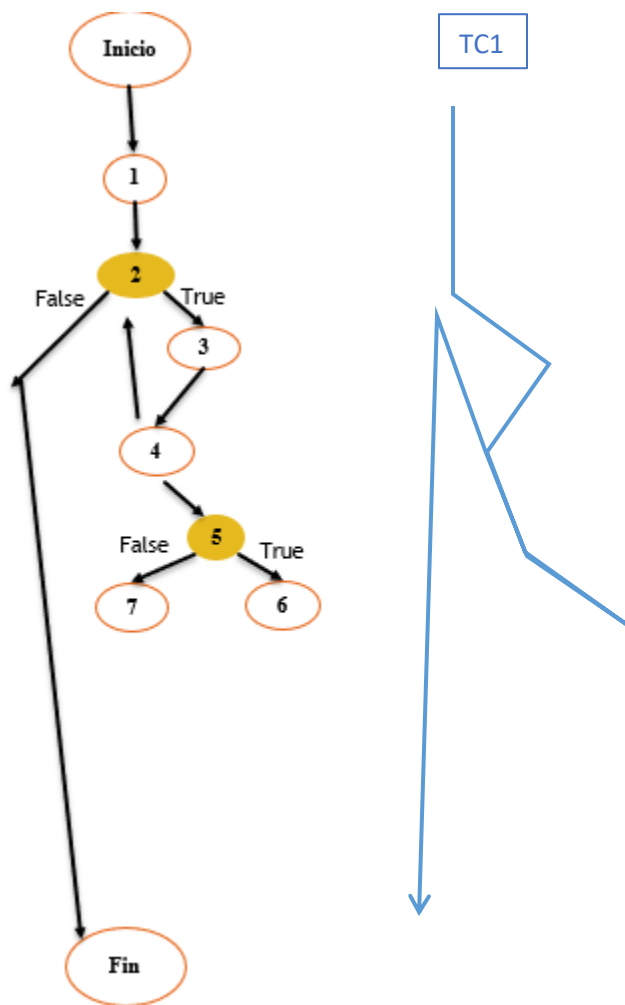
$$V(G)=P+1$$

P= número de nodos predicados contenidos en el grafo: 2

$$V(G) = 2 \text{ nodos predicados} + 1 = 3$$

Complejidad ciclomática = 3

## Técnica Cobertura de sentencias



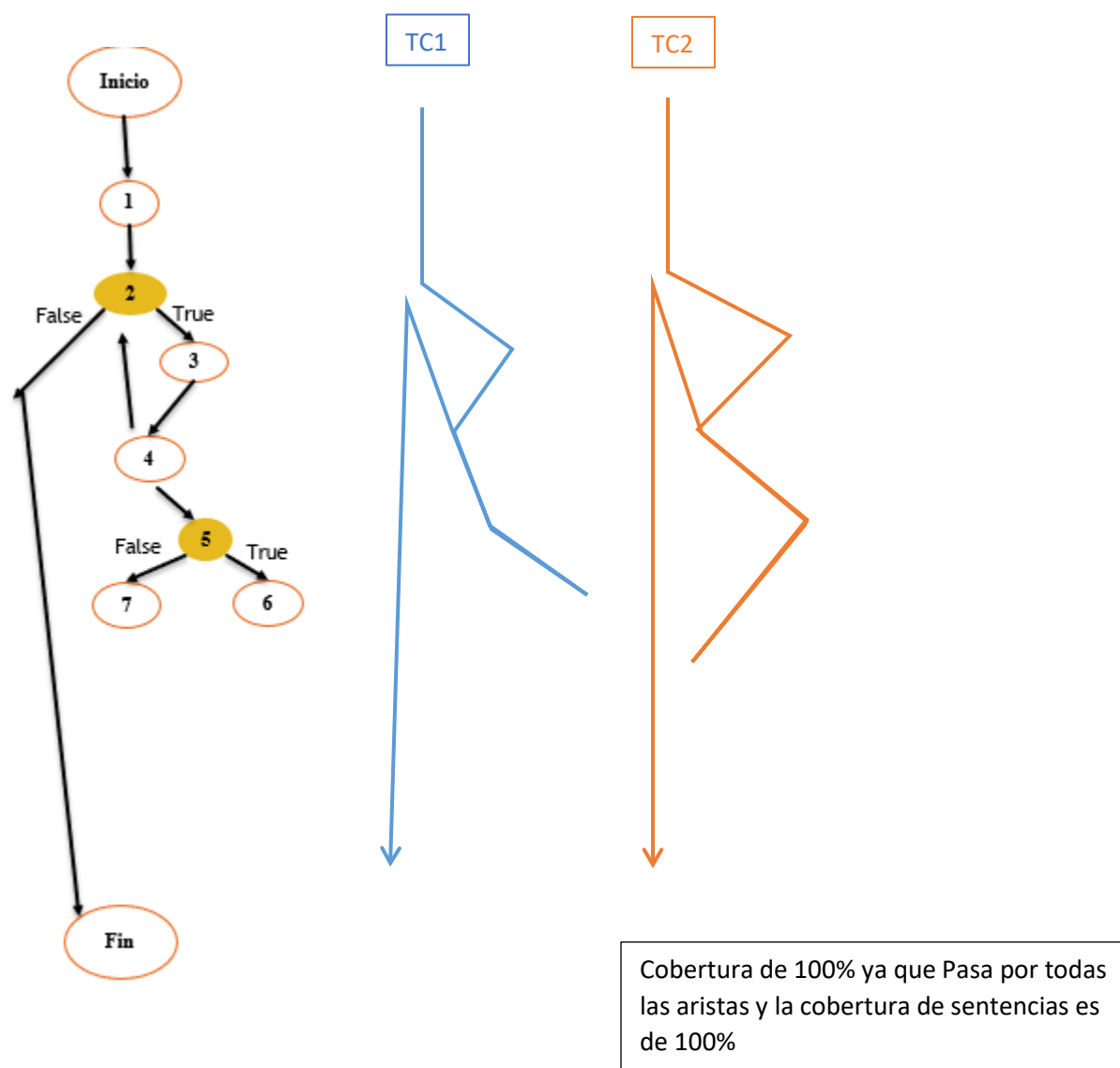
Cobertura: (nro  
sentencias ejecutadas  
\*100) / nro total de  
sentencias:

$$\text{Cobertura} = 2 * 100 / 2$$

Con 1 casos de prueba se  
alcanza una cobertura del  
100%

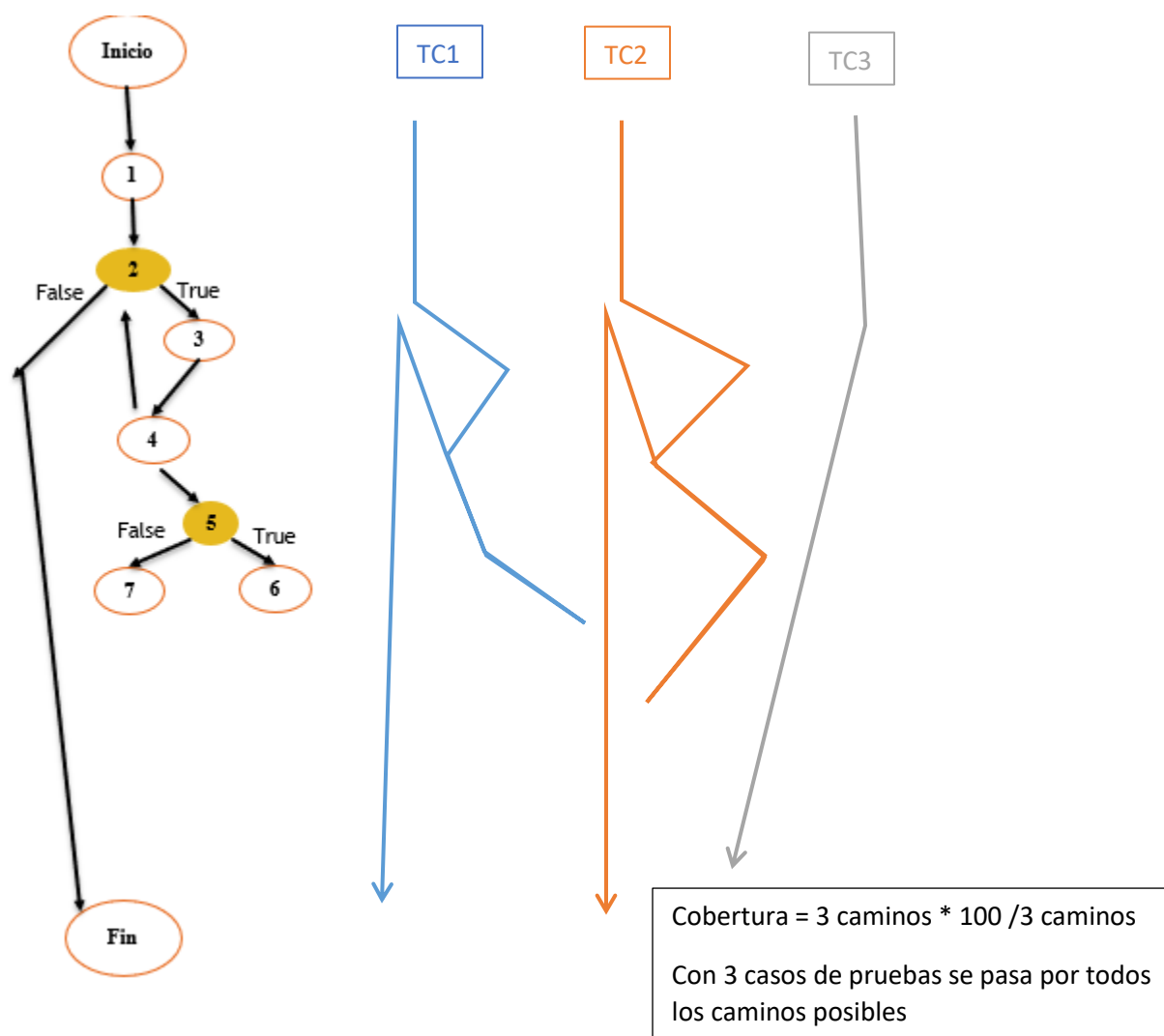
TC	Camino	Entrada	Prueba	Salida
TC1	1,2,3,4,5,6, F	Tam_tarea= 0 ; Tam_tarea<5 or Tam_tarea>50; True Tam_tarea=Tarea.lenght() Tam_tarea<=50 and >=5(True)	Tarea="Programar"	Tarea agregada con exito

### Técnica Cobertura de decisión



TC	Camino	Entrada	Prueba	Salida
<b>TC1</b>	1,2,3,4,5,6,2 F	Tam_tarea<5 or Tam_tarea>50; True Tam_tarea=Tarea.lenght() Tam_tarea<=50 and >=5(True)	Tam_tarea= 0 ; Tarea="Programar" Tam_tarea=9	Tarea agregada con éxito
<b>TC2</b>	1,2,3,4,5,7,2 F	Tam_tarea<5 or Tam_tarea>50; True Tam_tarea=Tarea.lenght() Tam_tarea<=50 and >=5(False)	Tam_tarea= 0 ; Tarea="hula" Tam_tarea=4	Error La tarea debe contener entre 5 y 50 caracteres

### Técnica cobertura de caminos



TC	Camino	Entrada	Prueba	Salida
<b>TC1</b>	1,2,3,4,5,6,2 F	Tam_tarea<5 or Tam_tarea>50; True Tam_tarea=Tarea.lenght() Tam_tarea<=50 and >=5(True)	Tam_tarea= 0 ; Tarea="" Programar" Tam_tarea=9	Tarea agregada con éxito
<b>TC2</b>	1,2,3,4,5,7,2 F	Tam_tarea<5 or Tam_tarea>50; True Tam_tarea=Tarea.lenght() Tam_tarea<=50 and >=5(False)	Tam_tarea= 0 ; Tarea=""hula" Tam_tarea=4	Error La tarea debe contener entre 5 y 50 caracteres
<b>TC3</b>	1,2, F	Tam_tarea<5 or Tam_tarea>50; True	Tam_tarea=5	Programa no entra en el ciclo

### Prueba de condición y cobertura

Test Case	Tamaño Tarea inicio	Tamaño Tarea fin
<b>TC1</b> Tam_tarea=0 Tarea=""Programar" Tam_tarea=9	Tam_tarea<5 or Tam_tarea>50(True)	Tam_tarea>=5 or Tam_tarea<=(True)
<b>TC2</b> Tam_tarea=0 Tarea=""eli" Tam_tarea=3	Tam_tarea<5 or Tam_tarea>50(True)	Tam_tarea>=5 or Tam_tarea<=(False)
<b>TC3</b> Tam_tarea=5	Tam_tarea<5 or Tam_tarea>50(False)	*
<b>TC4</b> Tam_tarea=0 Tarea=""texto con 60 caracteres..." Tam_tarea=60	Tam_tarea<5 or Tam_tarea>50(True)	Tam_tarea>=5 or Tam_tarea<=(False)



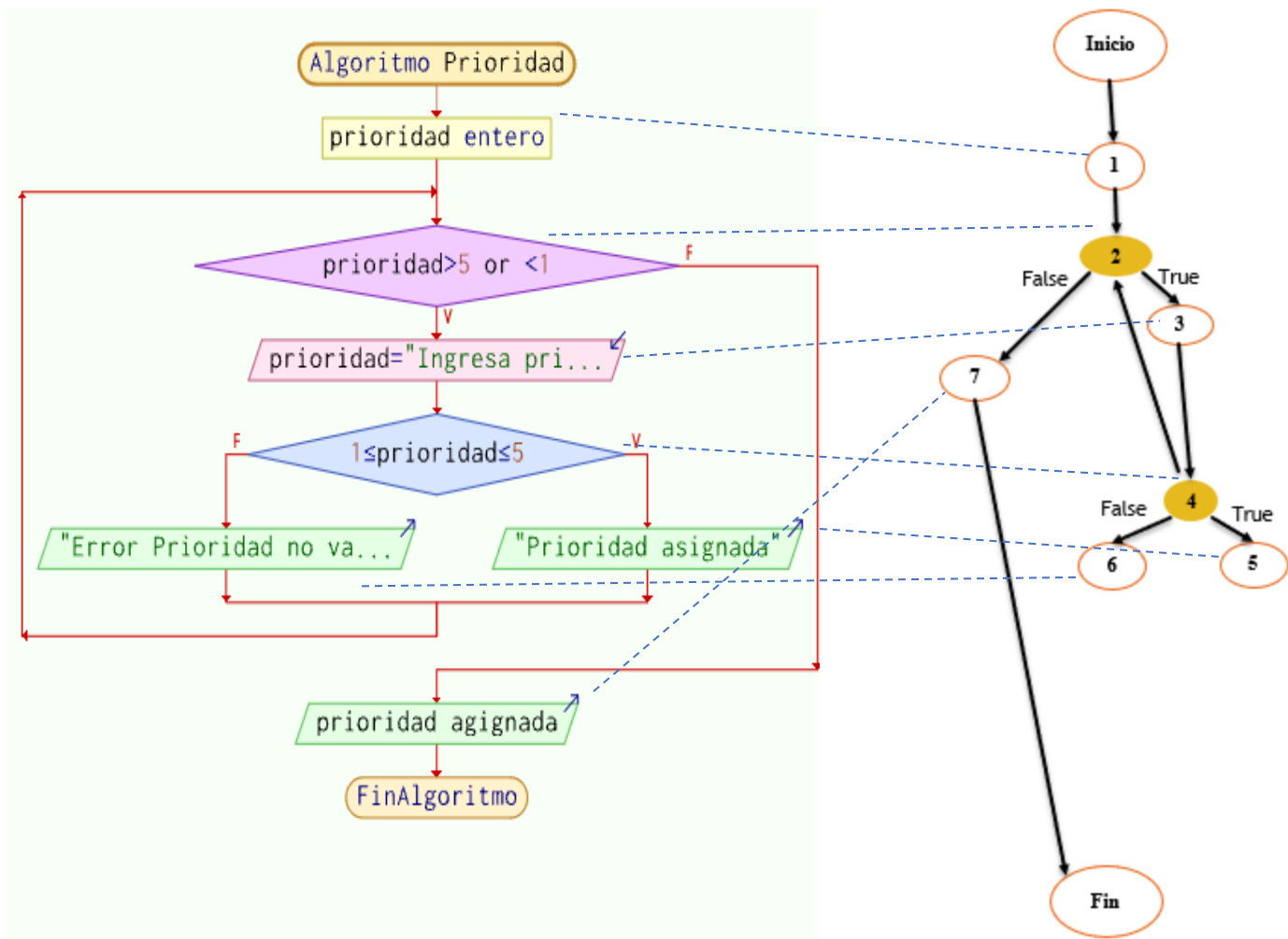
## Algoritmo requisito # 2 parte 2

```

prioridad.py > ...
1  # pedir prioridad
2  prioridad = 0
3  while prioridad < 1 or prioridad > 5:
4      prioridad = int(input("Ingresa la prioridad de la tarea: donde " +
5          "\n1 es prioridad Alta y 5 prioridad muy baja\n"))
6      if prioridad <= 5 and prioridad >= 1:
7          print("prioridad agignada correctamente")
8      else:
9          print("Error la prioridad debe ir de 1 a 5 ")
10

```

## Diagrama de flujo y diagrama de grafo



## Complejidad Ciclomática

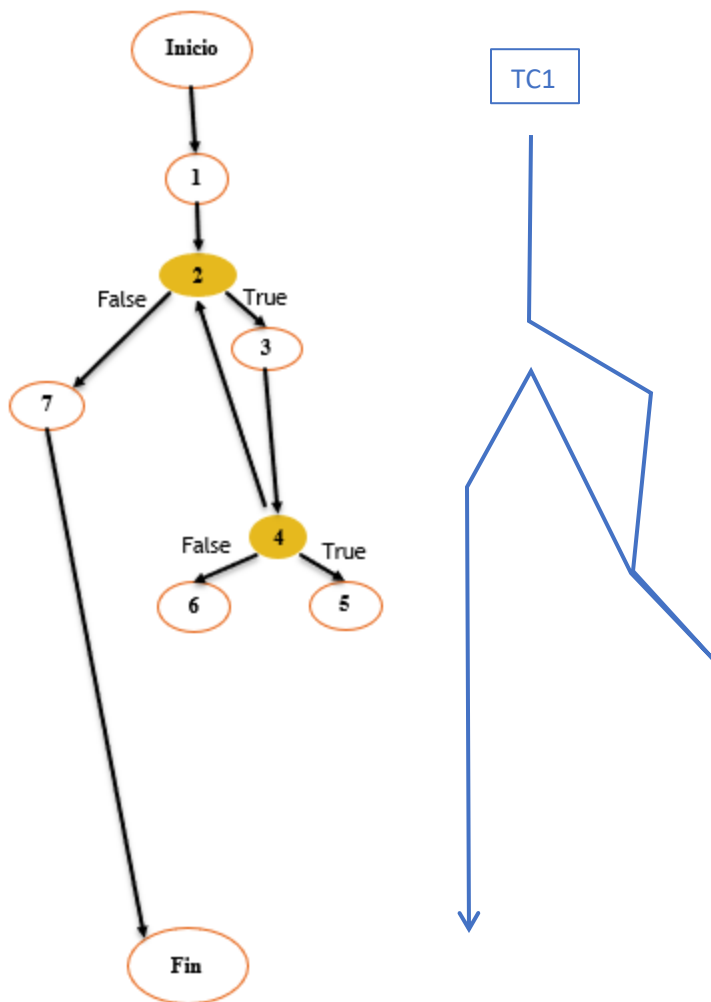
$$V(G)=P+1$$

P= número de nodos predicados contenidos en el grafo: 2

$$V(G) = 2 \text{ nodos predicados} + 1 = 3$$

Complejidad ciclomática = 3

## Técnica Cobertura de sentencias



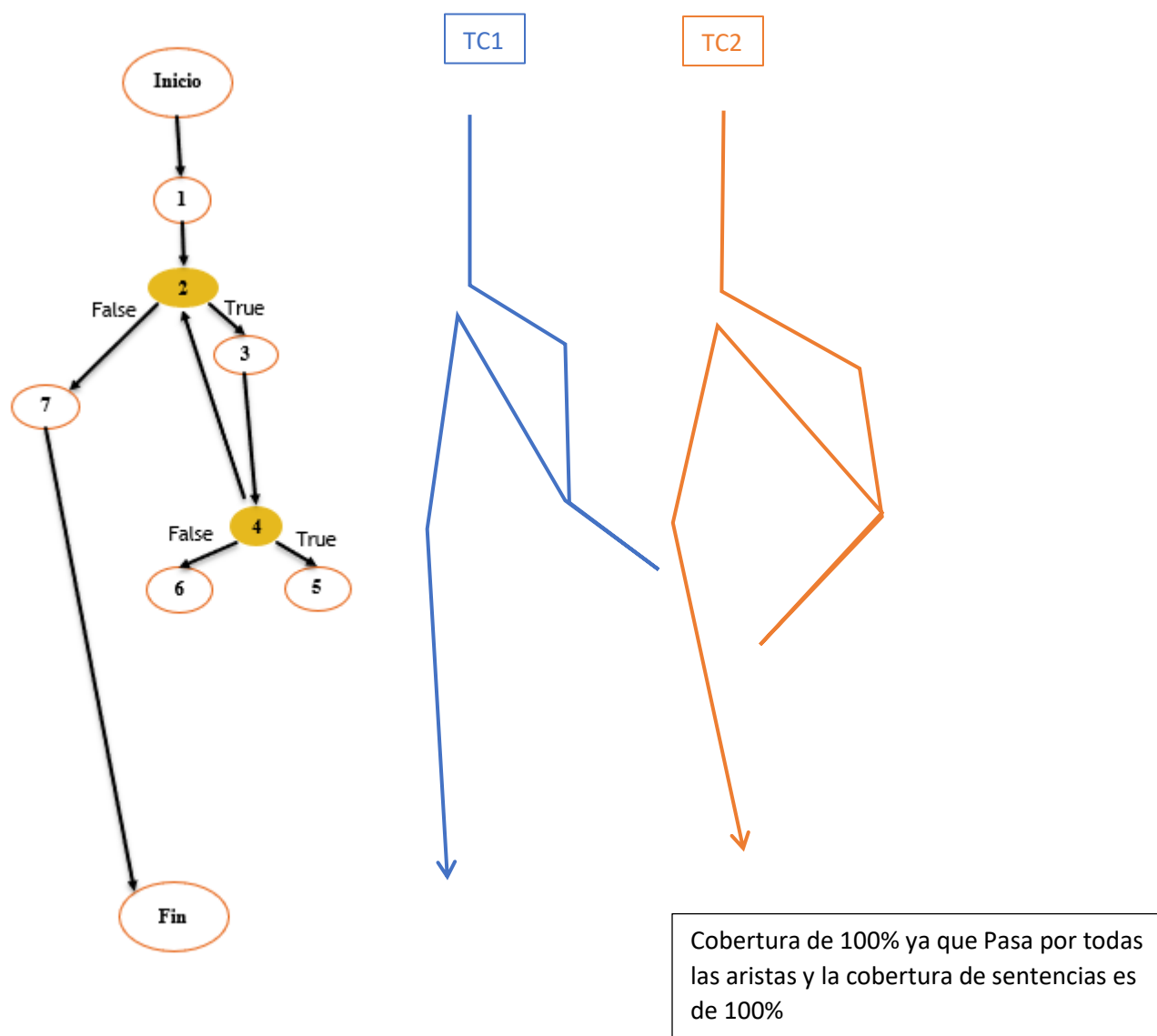
Cobertura: (nro  
sentencias ejecutadas  
\*100) / nro total de  
sentencias:

$$\text{Cobertura} = 2 * 100 / 2$$

Con 1 casos de prueba se  
alcanza una cobertura del  
100%

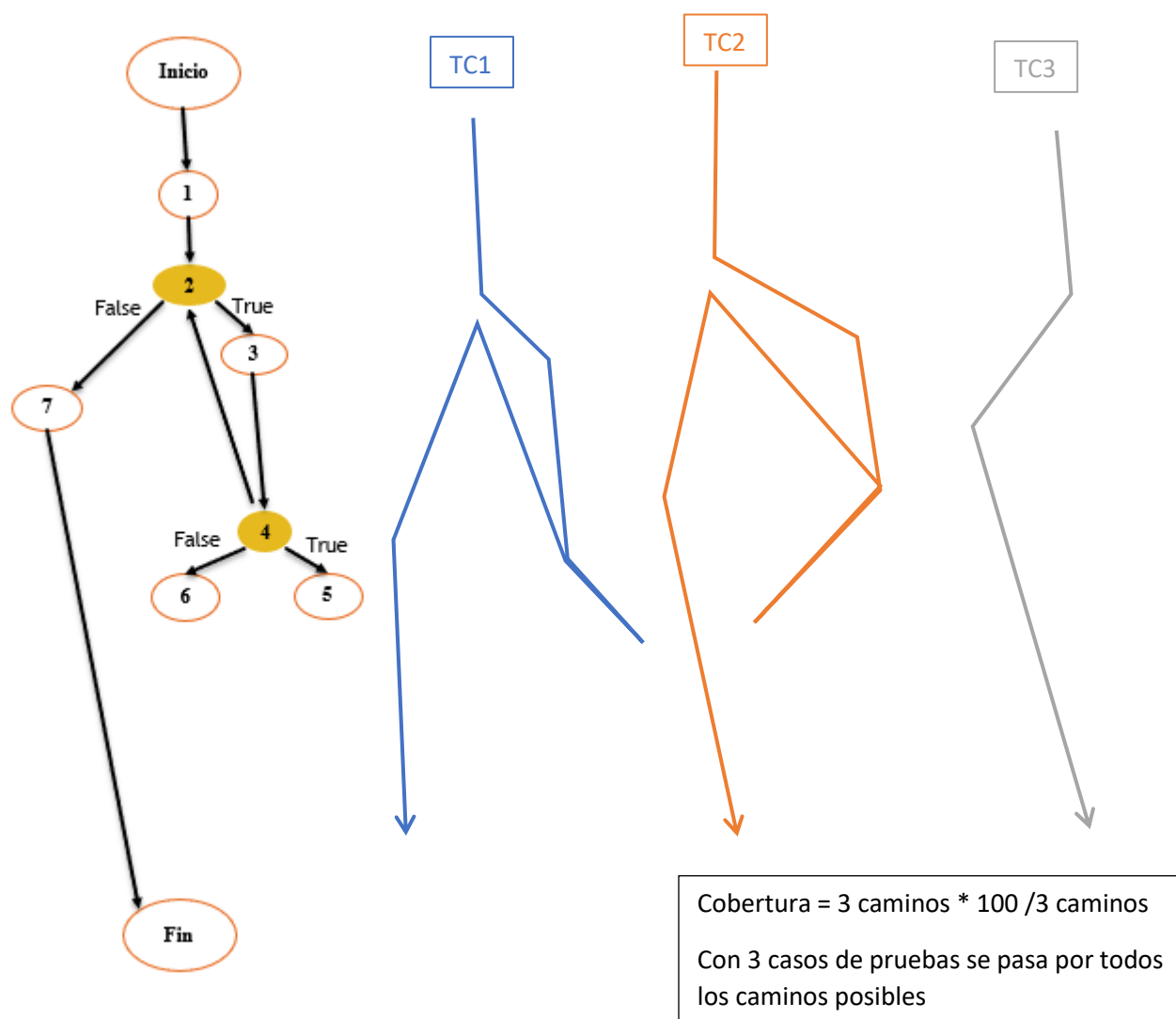
TC	Camino	Entrada	Prueba	Salida
TC1	1,2,3,4,5,7 F	Prioridad >5 or Prioridad <1; True ----- Prioridad = input() Prioridad <=50 and >=5(True)	Prioridad= 0 ----- Prioridad = 5	Prioridad asignada correctamente

### Técnica Cobertura de decisión



TC	Camino	Entrada	Prueba	Salida
<b>TC1</b>	1,2,3,4,5,2,7 F	Prioridad>5 or Prioridad<1; True ----- Prioridad=input() Prioridad <=50 and >=5(True)	Prioridad= 0 ; ----- Prioridad=3 ;	Prioridad asignada con exito
<b>TC2</b>	1,2,3,4,6,2,7 F	Prioridad>5 or Prioridad<1; True ----- Prioridad=input() Prioridad <=50 and >=5(True)	Prioridad=0; ----- Prioridad=8	Error prioridad no valida

### Técnica cobertura de caminos



TC	Camino	Entrada	Prueba	Salida
TC1	1,2,3,4,5,2,7 F	Prioridad>5 or Prioridad<1; True ----- Prioridad=input() Prioridad <=50 and >=5(True))	Prioridad= 0 ; ----- Prioridad=3 ;	Prioridad asignada con exito
TC2	1,2,3,4,6,2,7 F	Prioridad>5 or Prioridad<1; True ----- Prioridad=input() Prioridad <=50 and >=5(True)	Prioridad=0; Prioridad=8	Error prioridad no valida
TC3	1,2,7, F	Prioridad>5 or Prioridad<1; False	Prioridad=;	Programa no entra en el ciclo

### Prueba de condición y cobertura

Test Case	prioridad
TC1 Prioridad=3	Prioridad <=50 and >=5(True)
TC2 Prioridad =8	Prioridad <=50 and >=5(False)