



Módulo 5: Diseño de casos de prueba

5.5 Técnicas de caja blanca - estructuras de control

Técnica de estructuras de control

- Ayudan a diseñar buenas pruebas para las estructuras de control que causan más problemas
 - Condiciones (if ... then ... else)
 - Ciclos (while, do, for)
- Importante:
 - No estamos haciendo un “code review”

Para Condiciones

Técnicas para condiciones

- Técnica de Dominio
 - Aplica para operadores relacionales
 - Diseñar 3 pruebas:
 - Un valor mayor
 - Un valor menor
 - Un valor igual
- Técnica de tablas de verdad
 - Para expresiones booleanas complejas

Ejemplo tablas de verdad

if $(a < b)$ or $(b > 4)$ then

Con Falso-Verdadero

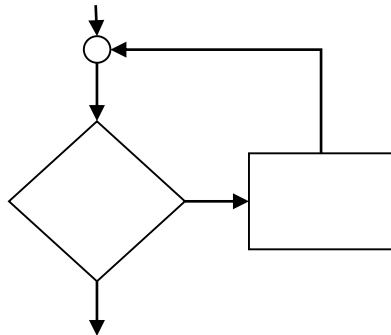
# prueba	$a < b$	$b > 4$	a	b
1	F	F	4	4
2	F	V	6	5
3	V	F	2	3
4	V	V	1	8

Con Tec. de Dominio

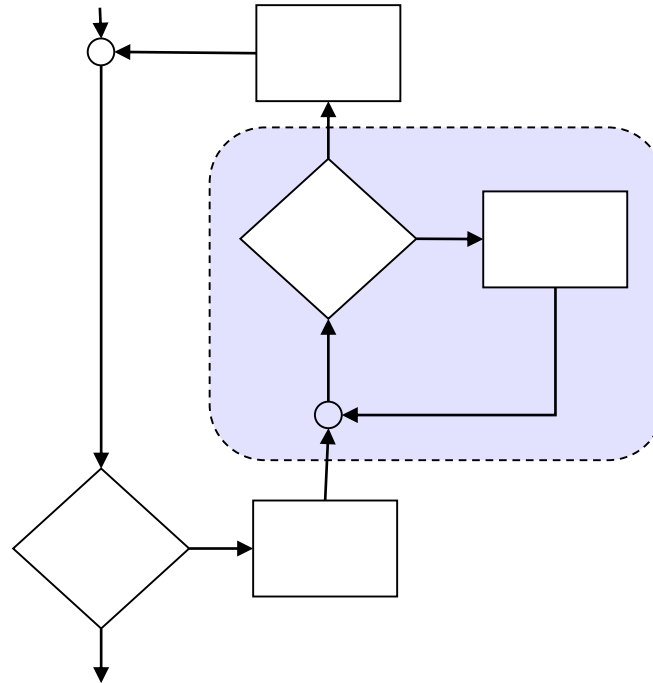
# prueba	$a < b$	$b > 4$	a	b
1	$a < b$	$b < 4$	0	1
2	$a < b$	$b = 4$	3	4
3	$a < b$	$b > 4$	4	5
4	$a = b$	$b < 4$	2	2
5	$a = b$	$b = 4$	4	4
6	$a = b$	$b > 4$	6	6
7	$a > b$	$b < 4$	4	3
8	$a > b$	$b = 4$	5	4
9	$a > b$	$b > 4$	8	7

Para Ciclos

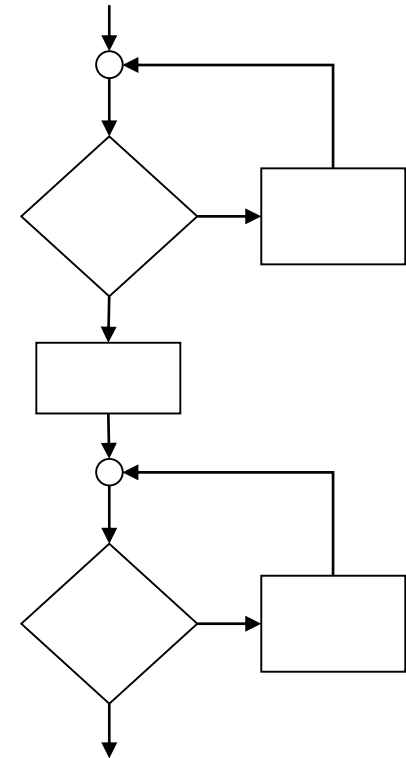
3 Tipos de Ciclos



Sencillo

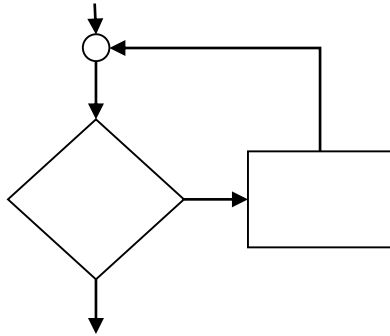


Anidado



Concatenado

Casos de prueba para ciclos sencillos

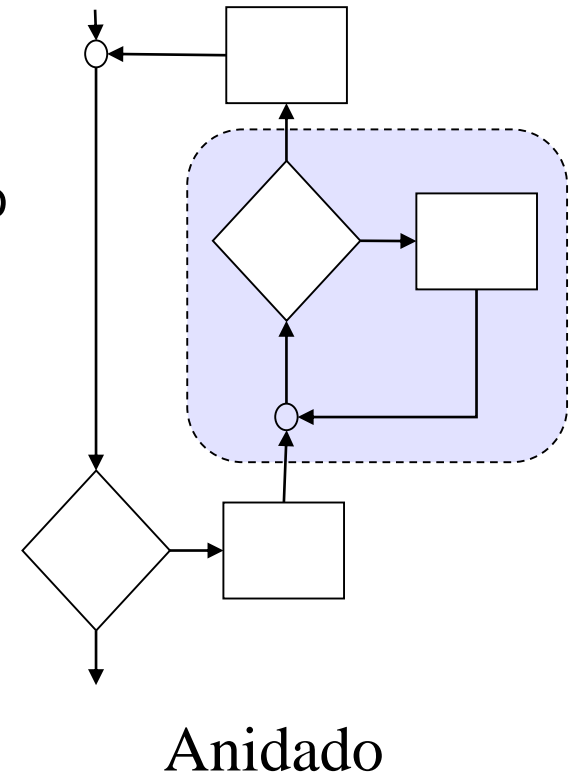


Sencillo

- Si existe un máximo de repeticiones, diseñar 6 pruebas:
 - 1) Cero repeticiones (saltar todo el ciclo)
 - 2) Una sola repetición
 - 3) Dos repeticiones
 - 4) max-1 repeticiones
 - 5) max repeticiones
 - 6) max+1 repeticiones
- Si no existe un máximo de repeticiones, diseñar 4 pruebas:
 - 1) Cero repeticiones (saltar todo el ciclo)
 - 2) Una sola repetición
 - 3) Dos repeticiones
 - 4) Un número muy grande de repeticiones

Casos de prueba para ciclos anidados

- Seguir el siguiente proceso:
 - Empezar por el ciclo mas interior
 - Utilizar reglas de ciclo sencillo al ciclo interior
 - Mantener ciclos externos en su valor mínimo (al menos 1)
 - Trabajar hacia fuera
 - Aplicar pruebas de ciclo sencillo al siguiente ciclo
 - Mantener ciclos externos en su valor mínimo (al menos 1)
 - Mantener ciclos internos en un valor típico



Ejemplo

Indicar los valores de los parámetros N y M para cada uno de los casos de prueba que se deben diseñar al utilizar la técnica pruebas de **ciclos** en el siguiente código:

```
// Regresa la suma de los datos del arreglo A.  
// El arreglo tiene 100 renglones y 70 columnas máximo.  
// Típicamente tiene 20 renglones y 5 columnas.  
int suma (int N, int M)  
{ int res = 0;  
  for (int i = 0; i < N; i++)  
    for (int j = 0; j < M; j++)  
      res += A [i] [j];  
  return res;  
}
```

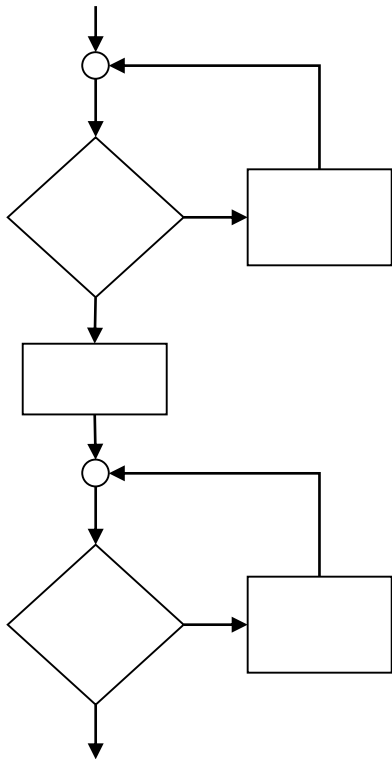
Respuesta

N	M
1	0
1	1
1	2
1	69
1	70
1	71

N	M
0	5
1	5
2	5
99	5
100	5
101	5

	Ciclo sencillo
	Valor mínimo
	Valor típico

Casos de prueba para ciclos concatenados



- Seguir el siguiente proceso:
 - Verificar si existe interdependencia entre ellos
 - Dos ciclos son interdependientes si el segundo utiliza alguna variable cuyo valor depende de lo calculado en el primero
 - Si existe interdependencia
 - Aplicar pruebas de ciclo anidado (como si el segundo ciclo estuviera dentro del primero)
 - Si no existe interdependencia (si son independientes)
 - Aplicar pruebas de ciclo sencillo a cada uno

Actividad individual

Indicar los valores de los parámetros N y M para cada uno de los casos de prueba que se deben diseñar al utilizar la técnica pruebas de **ciclos** en el siguiente código:

```
// Función que regresa la sumatoria de los elementos de "B"  
// que son mayores a la sumatoria de "A".  
// El tamaño del arreglo A es de 90, pero normalmente tiene 6.  
// El tamaño del arreglo B es de 120, pero típicamente tiene 3.  
int sumaMayor (int N, int M) {  
    int temp = suma = 0;  
    for (int i = 0; i < N; i++)  
        temp = temp + A[i];  
    for (int i = 0; i < M; i++)  
        if (B[i] > temp)  
            suma = suma + B[i];  
    return suma;  
}
```

Respuesta

N	M
1	0
1	1
1	2
1	119
1	120
1	121

N	M
0	3
1	3
2	3
89	3
90	3
91	3

Respuesta

(si hubieran sido **independientes**)

N	M
0	0
1	1
2	2
89	119
90	120
91	121

Ejemplo completo

Indicar los valores de los parámetros N, M y P para cada uno de los casos de prueba que se deben diseñar al utilizar la técnica pruebas para **condiciones** y para **ciclos** en el siguiente código.

```
// Función que regresa cuantos datos dentro del arreglo B son mayores al
// promedio de los datos del arreglo A.
// El tamaño del arreglo A es 500 x 50, pero normalmente es de 20 x 5.
// El tamaño del arreglo B es de 40, pero típicamente tiene 10.
int cant_mayor (int N, int M, int P)
{   if ( (N<1) || (M<1) || (P<1) ) return 0;
    int res = prom = 0;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < M; j++)
            prom += A [i] [j];
    prom = prom / ( N * M );
    for (int i = 0; i < P; i++)
        if ( B [i] > prom ) res++;
    return res;
}
```


Actividad individual

Realiza el paso de análisis para la primera condición y todos los ciclos:

```
// Función que regresa cuantos datos dentro del arreglo B son mayores al  
// promedio de los datos del arreglo A.  
// El tamaño del arreglo A es 500 x 50, pero normalmente es de 20 x 5.  
// El tamaño del arreglo B es de 40, pero típicamente tiene 10.
```

```
int cant_mayor (int N, int M, int P)
```

```
{  if ( (N<1) || (M<1) || (P<1) ) return 0;  
    int res = prom = 0;  
    for (int i = 0; i < N; i++)  
        for (int j = 0; j < M; j++)  
            prom += A [i] [j];  
    prom = prom / ( N * M );  
    for (int i = 0; i < P; i++)  
        if ( B [i] > prom ) res++;  
    return res;  
}
```

Paso 1 Análisis: Técnica Condiciones

```
// Función que regresa cuantos datos
// dentro del arreglo B son mayores al
// promedio de los datos del arreglo A.
// El tamaño del arreglo A es 500 x 50,
// pero normalmente es de 20 x 5.
// El tamaño de B es de 40,
// pero típicamente tiene 10.
```

```
int cant_mayor (int N, int M, int P)
{ if ( (N<1) || (M<1) || (P<1) ) return 0;
  int res = prom = 0;
  for (int i = 0; i < N; i++)
    for (int j = 0; j < M; j++)
      prom += A [i] [j];
  prom = prom / ( N * M );
  for (int i = 0; i < P; i++)
    if ( B [i] > prom ) res++;
  return res;
}
```

N<1	M<1	P<1
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

Paso 1 Análisis:

Técnica Condiciones

```
// Función que regresa cuantos datos  
// dentro del arreglo B son mayores al  
// promedio de los datos del arreglo A.  
// El tamaño del arreglo A es 500 x 50,  
// pero normalmente es de 20 x 5.  
// El tamaño de B es de 40,  
// pero típicamente tiene 10.
```

```
int cant_mayor (int N, int M, int P)  
{  if ( (N<1) || (M<1) || (P<1) ) return 0;  
    int res = prom = 0;  
    for (int i = 0; i < N; i++)  
        for (int j = 0; j < M; j++)  
            prom += A [i] [j];  
    prom = prom / ( N * M );  
    for (int i = 0; i < P; i++)  
        if ( B [i] > prom ) res++;  
    return res;  
}
```

**Asegurar que el arreglo B
contenga datos:**

B[i]
<prom
=prom
>prom

Paso 1 Análisis:

Técnica Ciclos (interdependientes)

```
// Función que regresa cuantos datos
// dentro del arreglo B son mayores al
// promedio de los datos del arreglo A.
// El tamaño del arreglo A es 500 x 50,
// pero normalmente es de 20 x 5.
// El tamaño de B es de 40,
// pero típicamente tiene 10.
```

```
int cant_mayor (int N, int M, int P)
{  if ( (N<1) || (M<1) || (P<1) ) return 0;
    int res = prom = 0;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < M; j++)
            prom += A [i] [j];
    prom = prom / ( N * M );
    for (int i = 0; i < P; i++)
        if ( B [i] > prom ) res++;
    return res;
}
```

N	M	P
1	1	0
1	1	1
1	1	2
1	1	39
1	1	40
1	1	41

N	M	P
1	0	10
1	1	10
1	2	10
1	49	10
1	50	10
1	51	10

N	M	P
0	5	10
1	5	10
2	5	10
499	5	10
500	5	10
501	5	10

	Ciclo sencillo
	Valor mínimo
	Valor típico

Paso 2 Síntesis: Eliminar repetidos

N	M	P
1	1	0
1	1	1
1	1	2
1	1	39
1	1	40
1	1	41

N	M	P
1	0	10
1	1	10
1	2	10
1	49	10
1	50	10
1	51	10

N	M	P
0	5	10
1	5	10
2	5	10
499	5	10
500	5	10
501	5	10

N<1	M<1	P<1
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

**Asegurar que el arreglo B
contenga datos:**

B[i]
< prom
= prom
> prom

 = repetido

Paso 2 Síntesis: Seleccionar valores

N	M	P
1	1	0
1	1	1
1	1	2
1	1	39
1	1	40
1	1	41

N	M	P
1	0	10
1	1	10
1	2	10
1	49	10
1	50	10
1	51	10

N	M	P
0	5	10
1	5	10
2	5	10
499	5	10
500	5	10
501	5	10

N<1	M<1	P<1
3	-1	0
-2	4	-3
-4	0	5
0	-1	-2

Asegurar que el arreglo B
contenga datos:

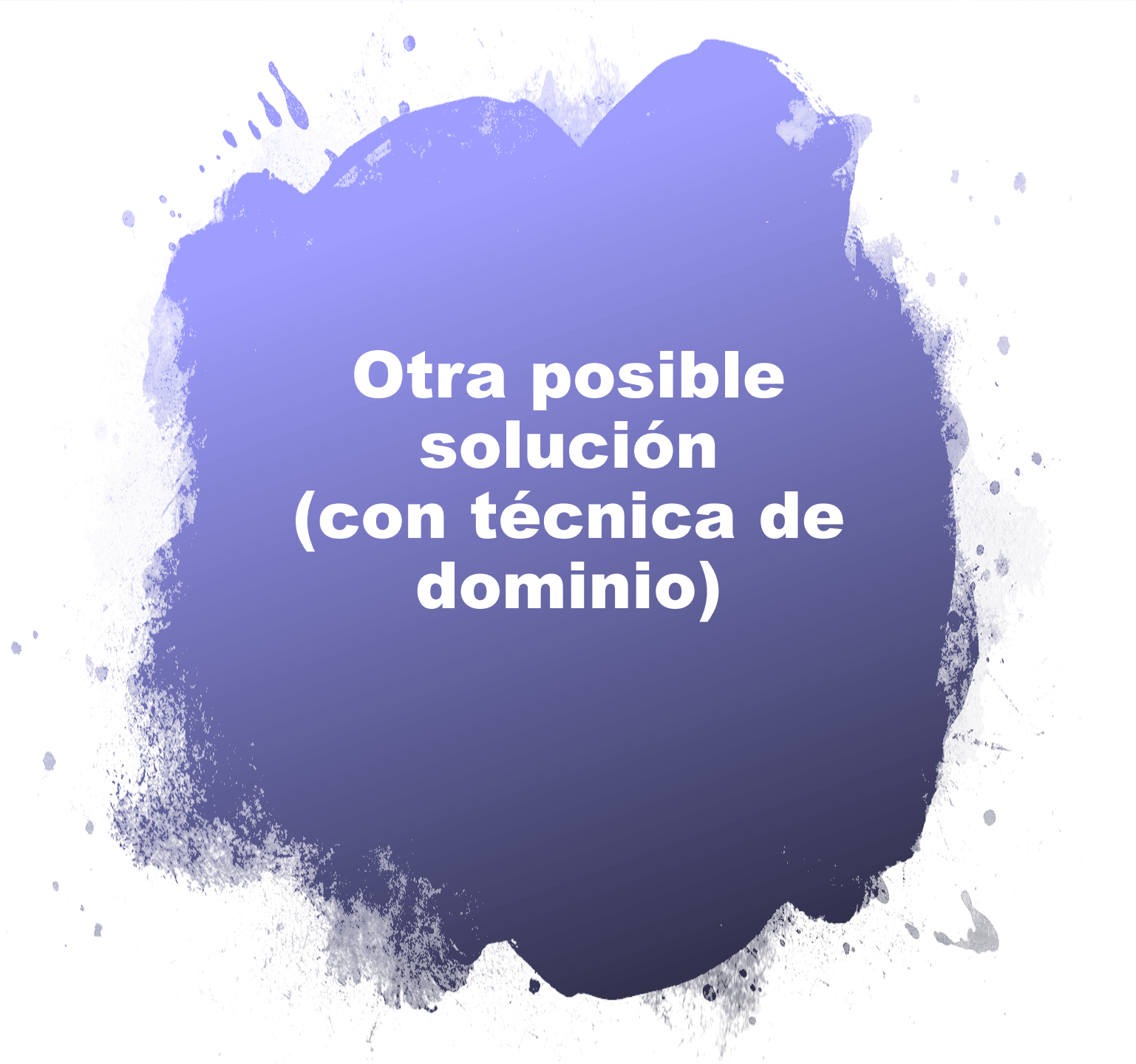
B[i]
< prom
= prom
> prom

22 casos de prueba

Paso 3:

Diseño de los casos de prueba

- Escribir los valores de entrada de cada caso de prueba
- Escribir las salidas esperadas de acuerdo a los requerimientos



**Otra posible
solución
(con técnica de
dominio)**

Tabla de verdad con técnica de dominio

N	M	P
<1	<1	<1
<1	<1	=1
<1	<1	>1
<1	=1	<1
<1	=1	=1
<1	=1	>1
<1	>1	<1
<1	>1	=1
<1	>1	>1

N	M	P
=1	<1	<1
=1	<1	=1
=1	<1	>1
=1	=1	<1
=1	=1	=1
=1	=1	>1
=1	>1	<1
=1	>1	=1
=1	>1	>1

N	M	P
>1	<1	<1
>1	<1	=1
>1	<1	>1
>1	=1	<1
>1	=1	=1
>1	=1	>1
>1	>1	<1
>1	>1	=1
>1	>1	>1

Síntesis (eliminar repetidos)

N	M	P
1	1	0
1	1	1
1	1	2
1	1	39
1	1	40
1	1	41

N	M	P
1	0	10
1	1	10
1	2	10
1	49	10
1	50	10
1	51	10

N	M	P
0	5	10
1	5	10
2	5	10
499	5	10
500	5	10
501	5	10

N	M	P
<1	<1	<1
<1	<1	=1
<1	<1	>1
<1	=1	<1
<1	=1	=1
<1	=1	>1
<1	>1	<1
<1	>1	=1
<1	>1	=1
<1	>1	>1

N	M	P
=1	<1	<1
=1	<1	=1
=1	<1	>1
=1	=1	<1
=1	=1	=1
=1	=1	>1
=1	>1	<1
=1	>1	=1
=1	>1	=1
=1	>1	>1

N	M	P
>1	<1	<1
>1	<1	=1
>1	<1	>1
>1	=1	<1
>1	=1	=1
>1	=1	>1
>1	>1	<1
>1	>1	=1
>1	>1	=1
>1	>1	>1

Asegurar que el arreglo B contenga datos:

B[i]
< prom
= prom
> prom

■ = repetido

Síntesis (seleccionar valores)

N	M	P
1	1	0
1	1	1
1	1	2
1	1	39
1	1	40
1	1	41

N	M	P
1	0	10
1	1	10
1	2	10
1	49	10
1	50	10
1	51	10

N	M	P
0	5	10
1	5	10
2	5	10
499	5	10
500	5	10
501	5	10

N	M	P
0	-2	0
-1	-1	1
-2	0	2
0	1	-4
0	1	1
0	1	7
0	2	-8
0	4	1

N	M	P
1	0	-2
1	-1	1
1	2	0
1	3	1

N	M	P
2	-3	-1
3	-8	1
4	-6	5
5	1	-2
6	1	1
7	1	8
8	6	-4
2	4	1

Asegurar que el arreglo B contenga datos:

B[i]
< prom
= prom
> prom

38 casos de prueba