Caminos básicos

La prueba de caminos básicos opera de la siguiente manera:

- Convertir el código a un grafo de control.
- Calcular la complejidad ciclomática de dicho grafo y obtener los caminos independientes.
- Preparar un caso de prueba para cada camino independiente (camino básico).

En el grafo de control cumple las siguientes especificaciones:

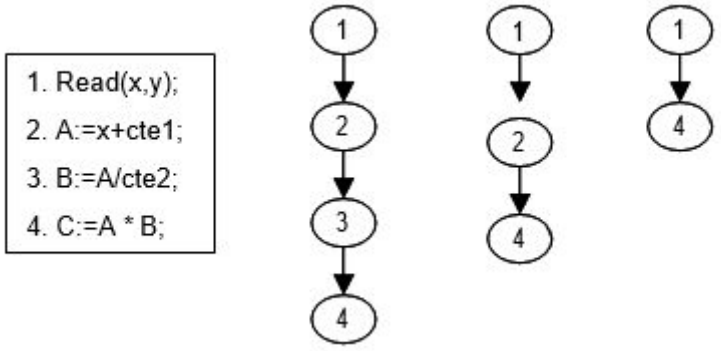
1. Existe un solo nodo inicial.
2. Existe un solo nodo final.
3. Cada nodo representa una secuencia de instrucciones.
4. La relación entre los nodos es una relación de “el control pasa a”.
5. Un nodo puede tener uno o más sucesores.
6. Un nodo puede tener uno o más antecesores.
7. Un nodo tendrá más de un sucesor si existen X caminos posibles dependiendo de condiciones.
8. Los casos de for, while, until y case se pueden reducir a grupos de nodos con dos sucesores.

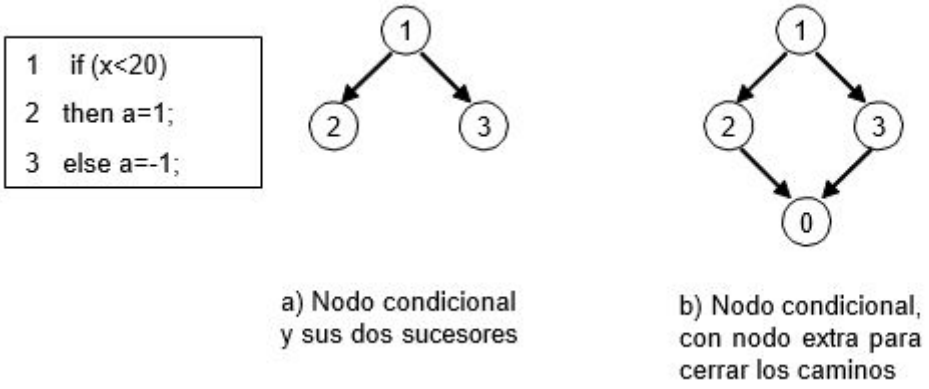
El grafo contendrá dos tipos de elementos:

- Nodos que representan una o más instrucciones secuenciales y
- Aristas que representan cambios de dirección.

En principio cada instrucción se representará como un nodo. Ahora bien, existen dos tipos de instrucción:

- Las que tienen un sucesor único, como las asignaciones y las instrucciones de lectura;
- Las que tienen dos o más sucesores posibles, como el if, la condición del for y el case.

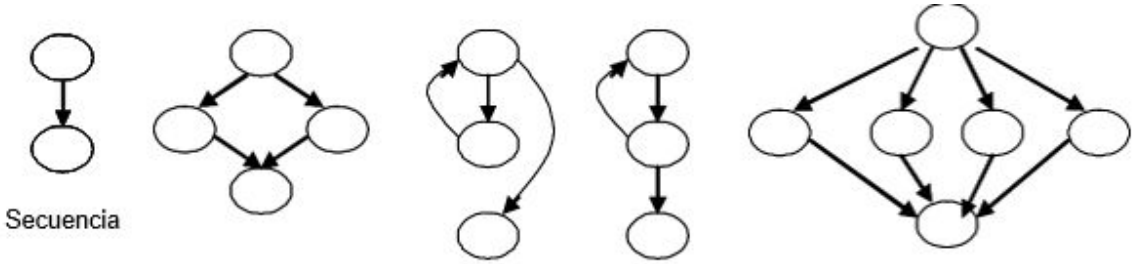




a) Nodo condicional y sus dos sucesores

b) Nodo condicional, con nodo extra para cerrar los caminos

La construcción del grafo de control puede emprenderse de dos maneras generales, ambas sistemáticas: descendente y ascendentemente. Ambas se auxilian de las estructuras típicas de programación estructurada, aunque puede usarse en casos no estructurados también. Las estructuras se muestran a continuación:



Secuencia

Condicional

Mientras ocurra

Repite hasta

Case

Existen dos métodos para generar dicho grafo de control a partir de código: Descendente y Ascendente. A partir del siguiente pseudocódigo se presentan las dos formas.

```
1  Lee llave
2  x1=lmin
3  x2=lmax
4  resp=-1
5  mientras (x1<x2-1)
6  ix=(x1+x2)/2
7     si (Lista[ix] == llave) entonces
8        resp = ix
9        x1 = ix
10       x2 = ix
11    de otro modo si (Lista[ix]<llave) entonces
12          x1 = ix
13       de otro modo x2 = ix
14 fin del mientras
15 regresa resp
```

Descendente

Con esta estrategia el programa será inicialmente un solo nodo, que se detallará cada vez más hasta donde sea necesario. Se comienza con los nodos que forman las estructuras iterativas (mientras, while, for) principales, comenzando por el principio del algoritmo. Luego se expanden las estructuras condicionales (if, case).

Como primer paso, se identifica la estructura principal que es un mientras, que abarca de la línea 6 a la 14. Cuando el mientras concluya, seguirá la instrucción 15. De acuerdo a las figuras básicas, se obtiene el grafo de la Figura 3.6 a).

En un segundo paso, el mientras (línea 5) se estructura de acuerdo a los patrones mostrados en la Figura 3.4, quedando como se muestra en la Figura 3.6 b).

En el tercer paso, el mientras tiene un if principal (línea 7) con dos ramas: la línea 10 y las líneas 11 a 13. Al concluir el if regresa al inicio del mientras. Note que todas las líneas de control regresan al nodo 5 y cuando falle el mientras, del nodo 5 seguirá el 15. El resultado se muestra en la Figura 3.6 c).

Un cuarto paso expande el nodo del if más interno (11 a 13) en dos ramas: 12 y 13. Al terminar, ambas regresan al nodo 5. El resultado se muestra en la Figura 3.6 d).

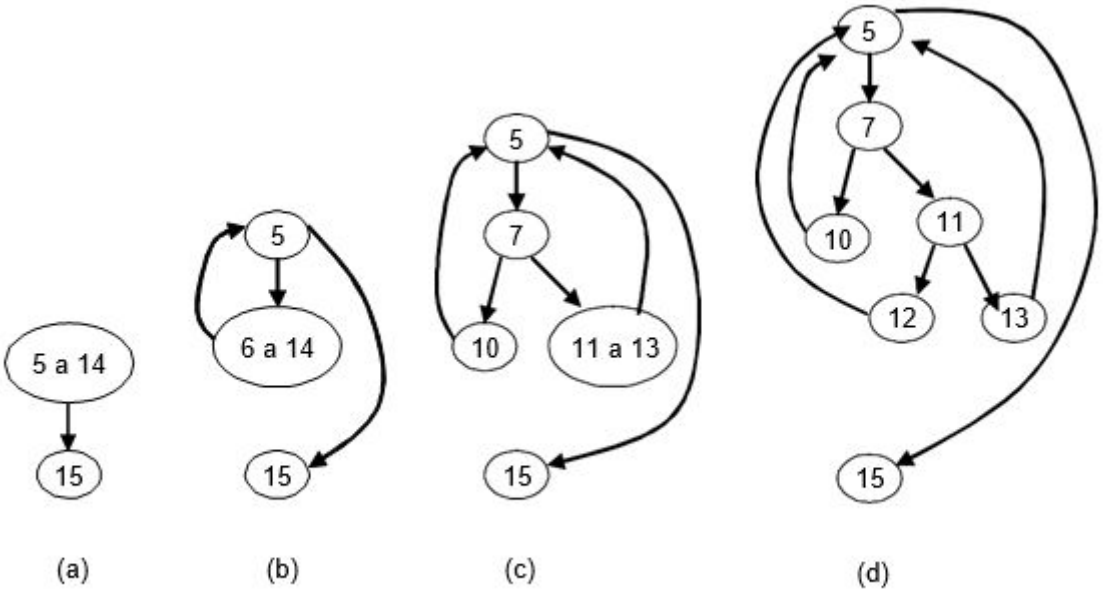


Figura 3.6 Creación descendente del grafo

Al observar las Figuras 3.6 c) y d) a veces resulta desagradable o confuso tener tantos regresos a un sólo nodo (el 5 en este caso) y algunos prefieren agregar nodos redundantes que sirven para ligar, pero no representan ninguna operación. Tales nodos se etiquetan con un 0, como se muestra en la Figura 3.7.

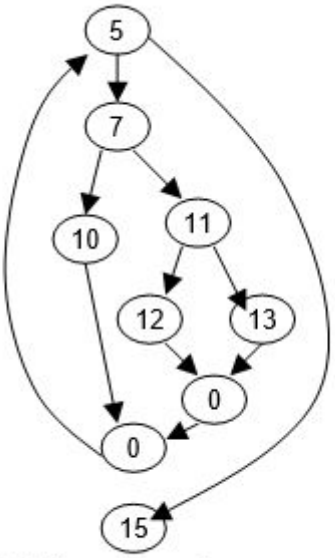


Figura 3.7 Uso de nodos conectores

Ascendente

Esta otra estrategia comienza por las instrucciones individuales y se van agrupando formando las estructuras. Usualmente se comienza por instrucciones que están dentro de

las iteraciones o condiciones más profundamente anidadas y se avanza hacia el exterior. Si se identifica un condicional, se reemplaza por la estructura correspondiente. En forma similar se aplica a iteraciones y case. En las iteraciones siempre existirá un nodo dentro de la iteración. En la Figura 3.8 se muestran varios pasos de la formación ascendente, que se describen a continuación.

En el primer paso se tiene una serie de nodos individuales. Recorriendo el grafo hasta niveles que no tengan anidamiento, se podrá identificar un grupo, correspondiente al Si de la línea 11, formando un grupo con los nodos 11, 12 y 13 (Figura 3.8 a). En este caso el uso de nodos conectores es muy útil, por lo cual se usarán en éste y los siguientes pasos.

En el siguiente paso se identifica el otro Si de la línea 7, que forma una estructura con el nodo 10 y el grupo previamente identificado (11 a 13), dando el grafo de la Figura 3.8 b).

En el cuarto paso identificamos el mientras de la línea 5, que incluye los grupos antes identificados, con su regreso, como se muestra en la Figura 3.8 c).

Finalmente, se conecta el mientras con el nodo final, quedando como en 3.8 d).

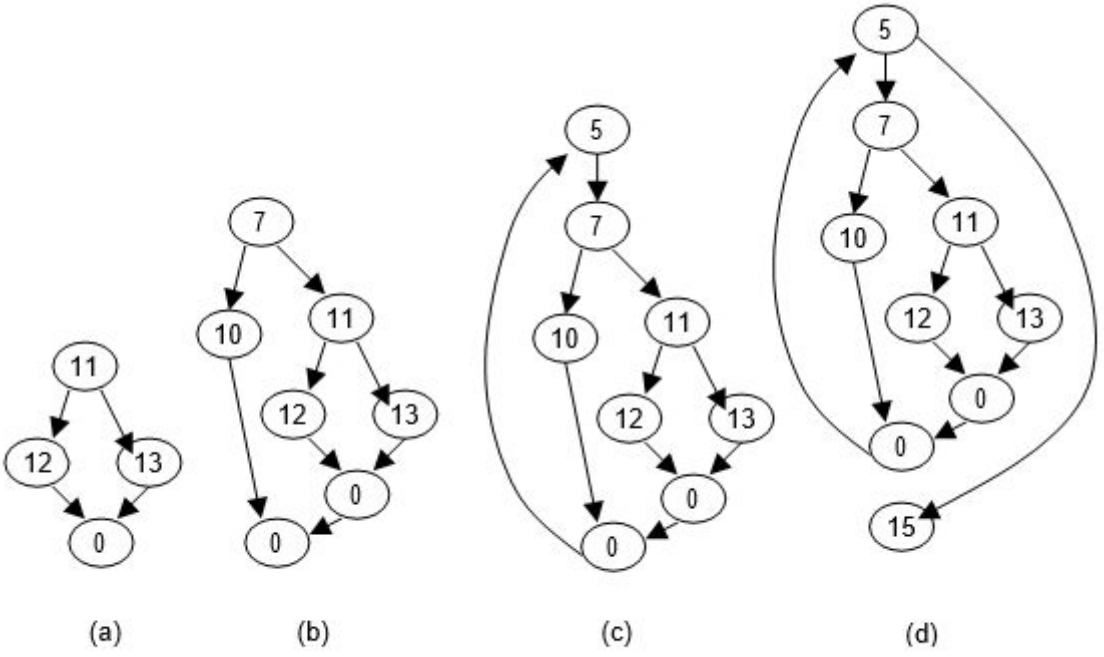


Figura 3.8 Construcción ascendente del grafo

En algunos casos, el grafo puede presentar complejidad extra cuando nos encontramos condiciones con varios predicados. De esta forma representamos esos varios predicados de la siguiente forma.

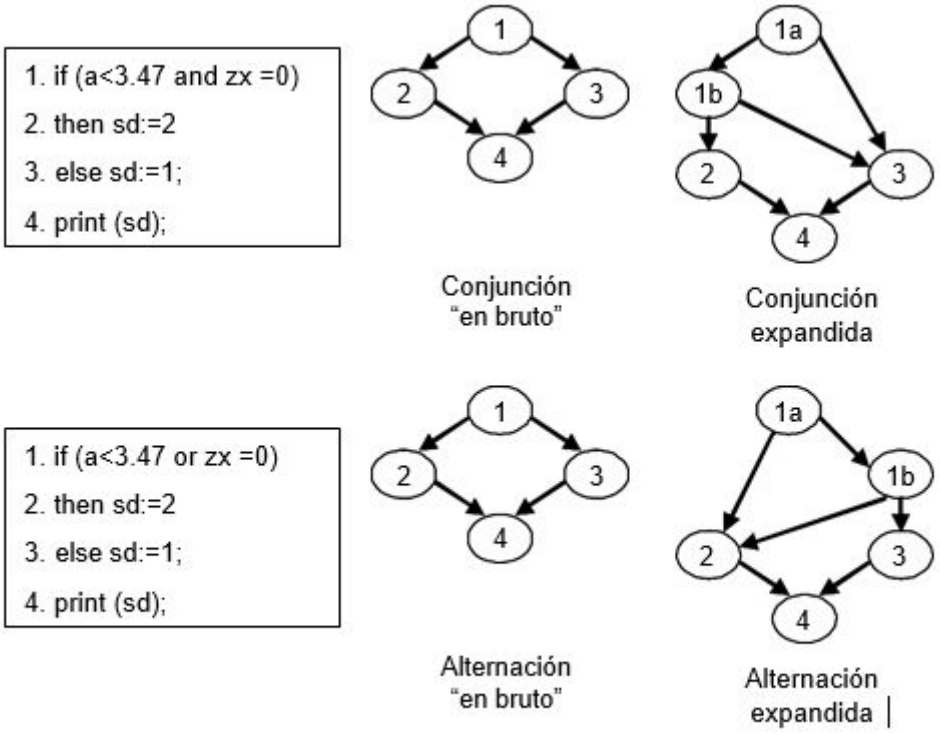


Figura 3.9 Expansión de condiciones con predicados múltiples

Referencias Bibliográficas

Fernñandez, J.M. (julio, 2010) *Capítulo 3 Caminos básicos.* Disponible en https://www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdf