Cálculo de la complejidad ciclomática

Máximo común divisor entre dos números (algoritmo de Euclides): Dados dos números naturales a y b, compruebe si b es igual a cero. Si es así, a es el mayor divisor común entre los mismos, de lo contrario, repetir el proceso usando b y el resto de la división de a por b.

```
def euclid(m, n)
  if n > m
     r = m
     m = n
     n = r
  end

  r = m % n

  while r != 0
     m = n
     n = r
     r = m % n
  end
  n
end
```

Sobre la base de este grafo, podemos definir la complejidad ciclomática de un programa de la siguiente manera:

CC = NA - NN + 2

En esta fórmula:

CC es la complejidad ciclomática
NA es el número de aristas del grafo
NN es el número de nodos en el grafo

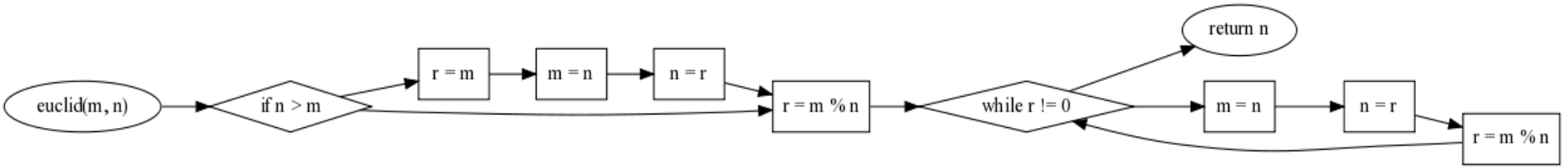
Tiene 11 nodos y 12 aristas, que nos da una complejidad ciclomática de 12-11 + 2, es decir, 3. Otra forma muy sencilla para descubrir la complejidad ciclomática es contar el número de loops cerrados en el grafo (que son formados por condicionales y loops) y añadir el número de puntos de salida. En el grafo anterior, tenemos 2 bucles cerrados (los if y while) y un punto de salida, resultando el mismo valor 3 para la complejidad de la función.

El valor de complejidad ciclomática establece un límite superior en la cantidad de pruebas necesarias para cubrir todos los caminos decisorios del código en cuestión. Este es un límite superior.

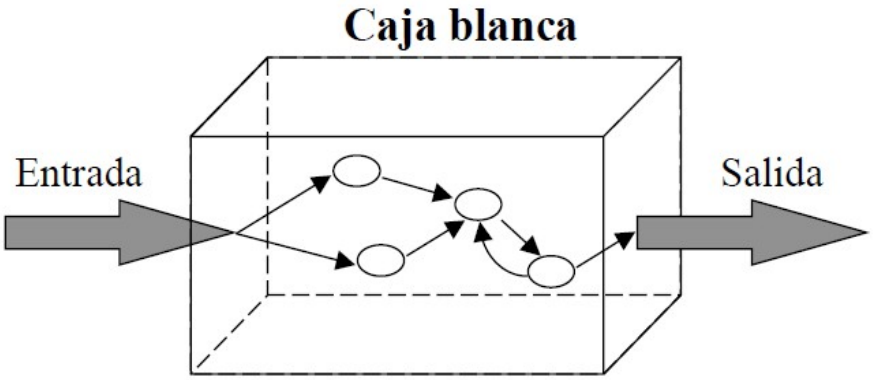
De eso se infiere que cuanto menor es la complejidad, menor es la cantidad de pruebas necesarias para el método en cuestión. Este hecho conduce a otra curiosidad: romper un método en varios reduce la complejidad de los métodos, pero aumenta la complejidad general del código y, de forma general, mantiene la comprobabilidad del programa completo en el mismo nivel.
Referencias

Basado en el trabajo de **McCabe**, los valores de referencia son los siguientes:

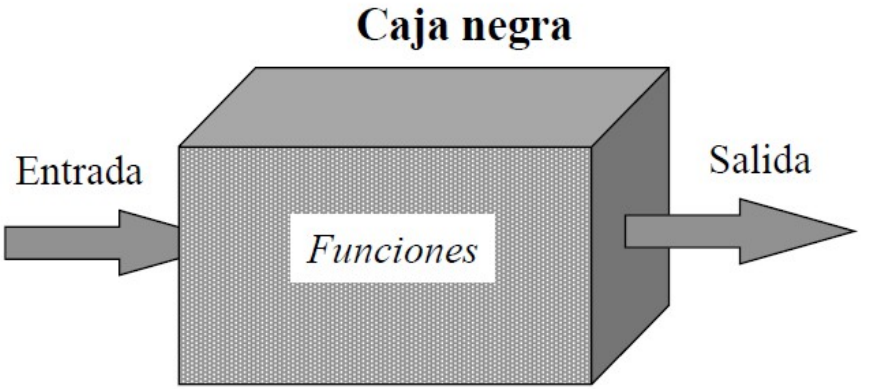
1-10, métodos sencillos, sin mucho riesgo
11-20, métodos medianamente complejos, con riesgo moderado
21-50, métodos complejos, con alto riesgo
51 o más, métodos inestables, de altísimo riesgo

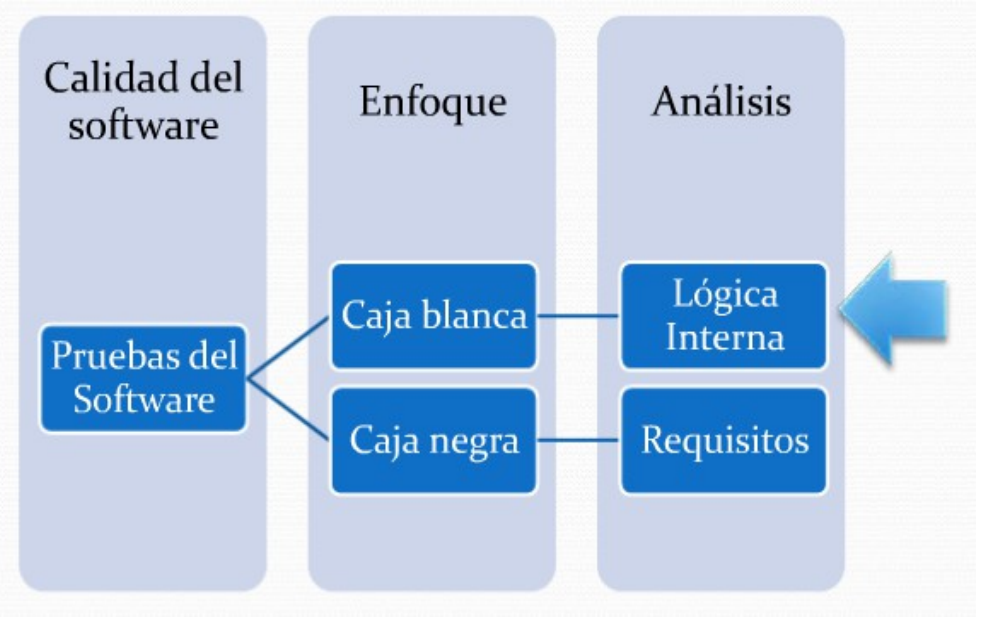


Estructural

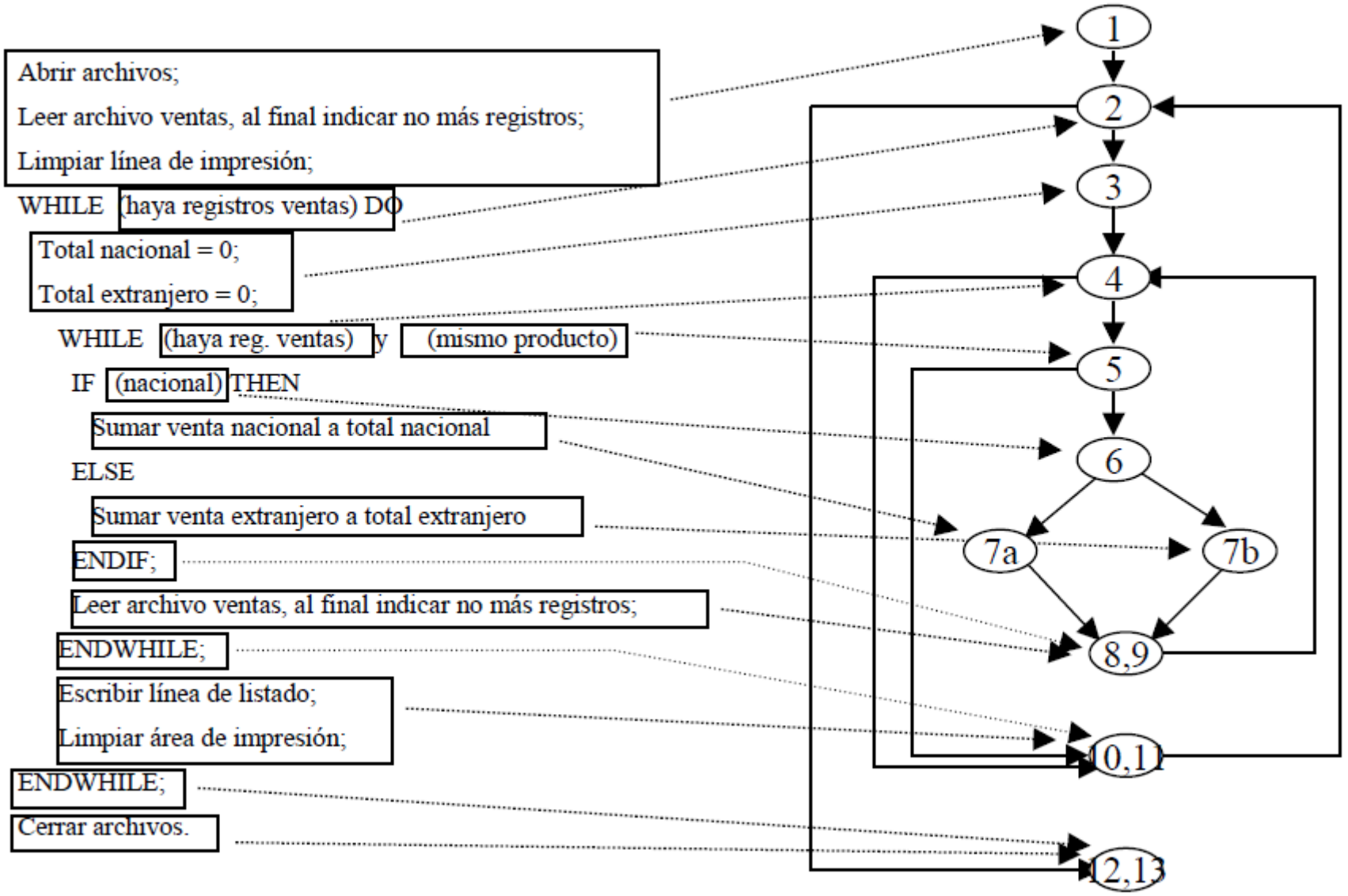


Funcional





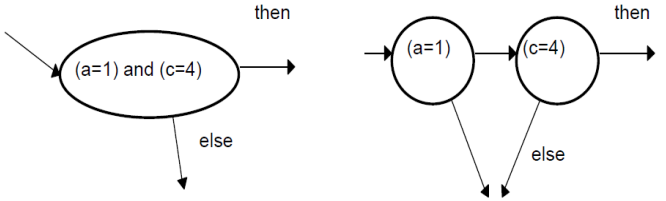
Grupo Alarcos
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha



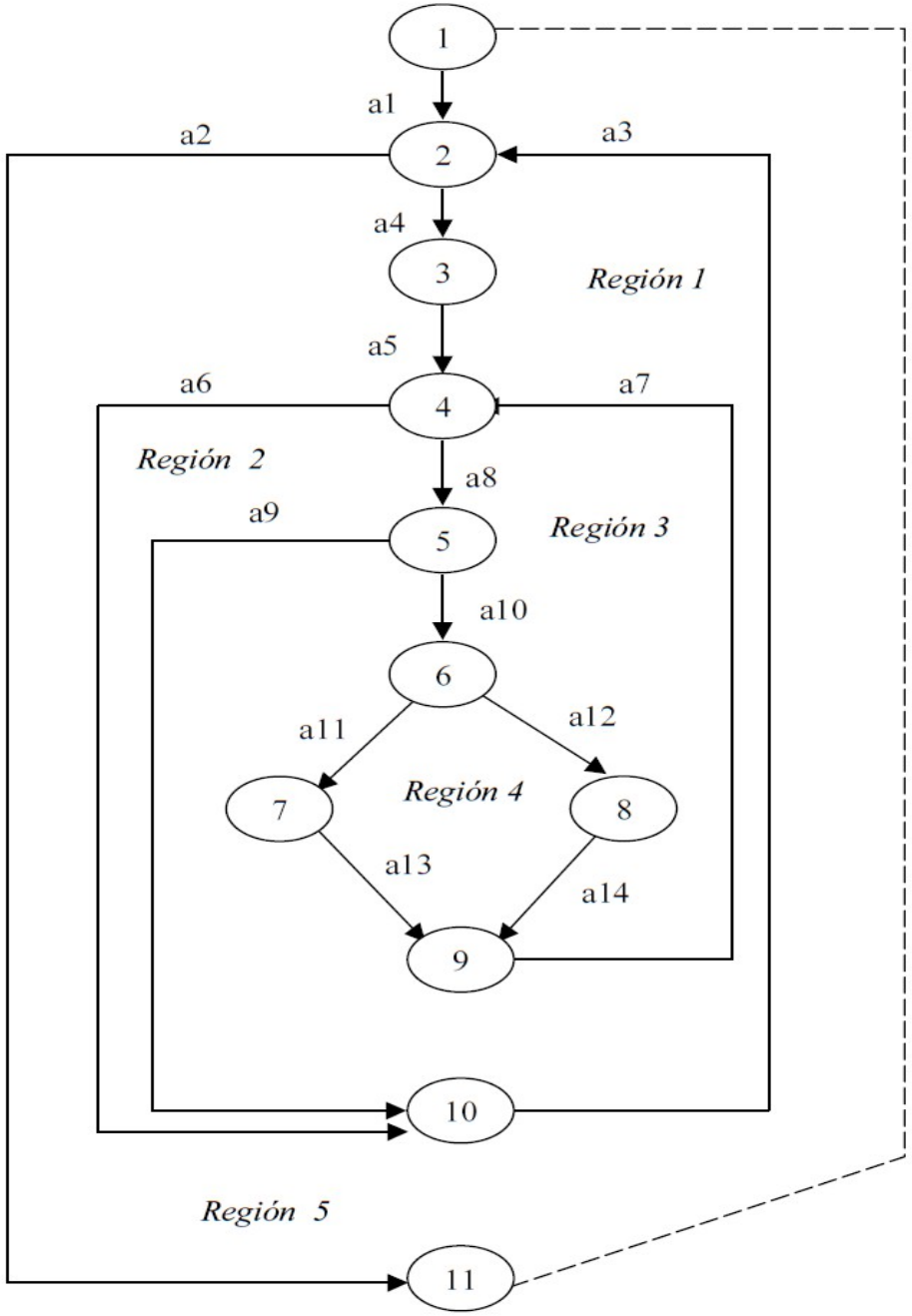
-Separar todas las condiciones

-Agrupar sentencias 'simples' en bloques

-Numerar todos los bloques de sentencias y también las condiciones



Grupo Alarcos
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha



a) V(G) = 14-11+2 = 5
b) V(G) = 5 regiones cerradas
c) V(G) = 5 condiciones

LA COMPLEJIDAD DE McCABE V(G)

- El criterio de prueba de McCabe es: Elegir tantos casos de prueba como caminos independientes (calculados como V(G))
- La experiencia en este campo asegura que:
  - V(G) marca el límite mínimo de casos de prueba para un programa
  - Cuando V(G) > 10 la probabilidad de defectos en el módulo o programa crece mucho → quizás sea interesante dividir el módulo

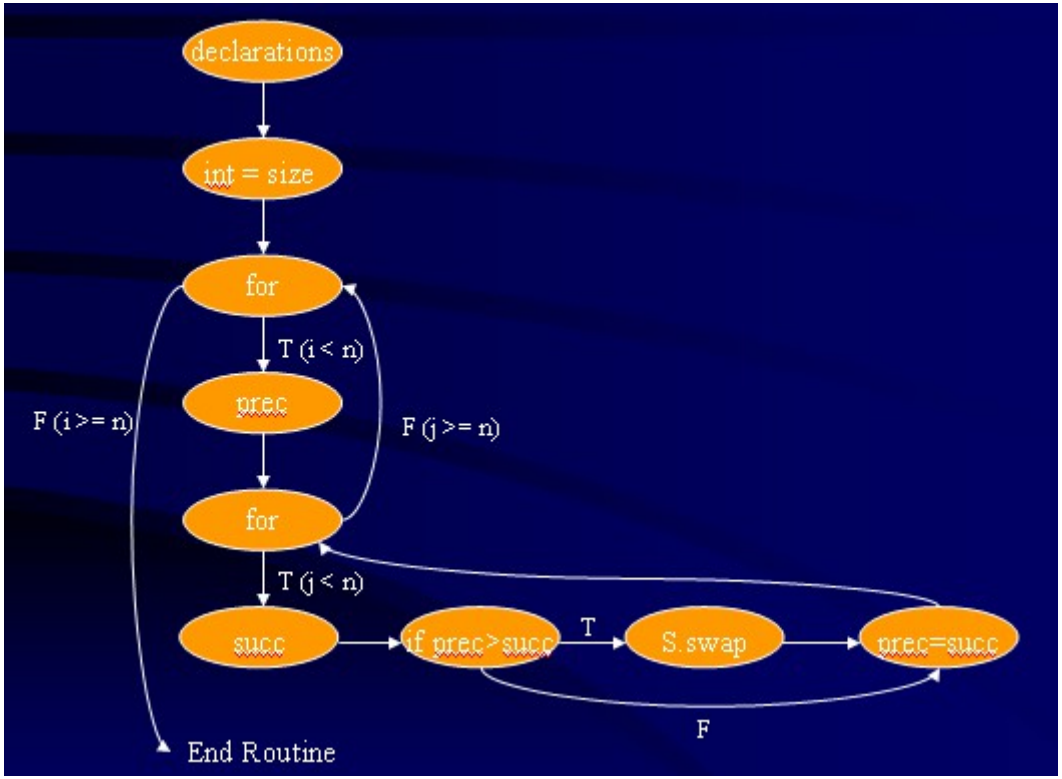
La complejidad de McCabe se puede calcular de cualquiera de estas 3 formas

1. V (G) = *a* - *n* + 2, siendo *a* el número de arcos o aristas del grafo y *n* el número de nodos.
2. V (G) = *r*, siendo *r* el número de regiones cerradas del grafo.
3. V(G) = *c* + 1, siendo *c* el número de nodos de condición.

Grupo Alarcos
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha

Calcular la complejidad ciclomática del método de ordenación de la Burbuja siguiendo el siguiente código:

```
Public static void bubbleSort2 (Sequence S) {
  Position prec, Succ;
  int n = S.size();
  for (int i = 0; i < n; i++) {
    // i-th pass
    prec = S.first();
    for (int j=1; j < n - i; j++) {
        succ = S.after(prec);
        if (valAtPos(prec) > valAtPos(succ))
            S.swapElements(prec, succ);
        rec = succ;
    }
  }
}
```



v(G) = NA - NN + 2;
v(G) = 12 - 10 + 2 = 4;
Complejidad ciclomática = 4

Module by: Miguel-Angel Sicilia.
Edited By: Verónica De la Morena
Summary Ejemplo de calcular la Complejidad Ciclomática de McCabe.