Proves de Caixa blanca

Índex

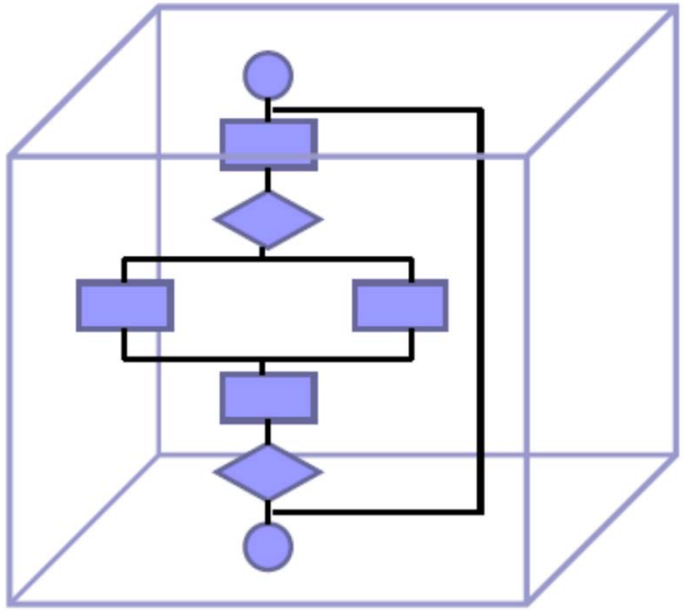
- Definició
- *Control flow* (camins bàsics)

Índex

- **Definició**
- *Control flow* (camins bàsics)

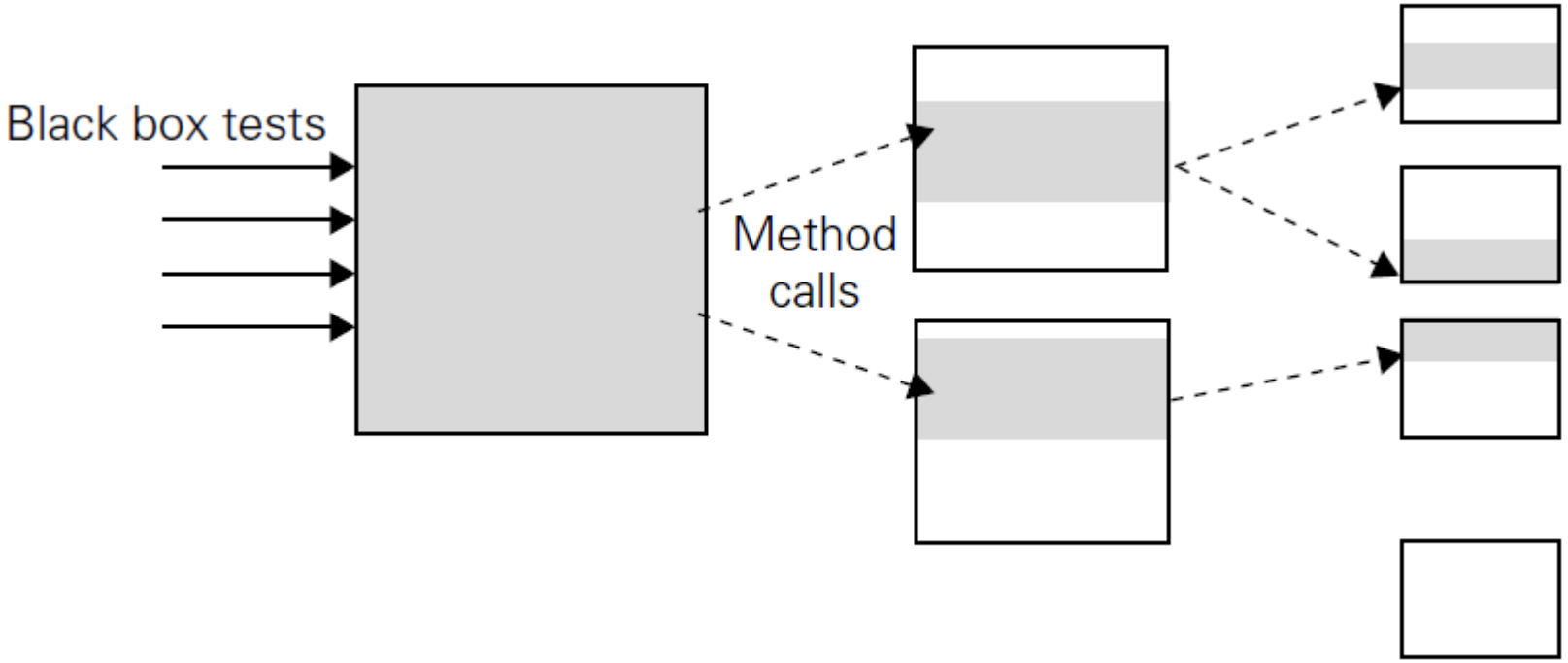
Definició

- *Testing* que analiza i té en compte l'estructura interna del sistema
- Analitza mòduls, funcions, blocs, sentencies, cobertura del codi, ...



Definició

- No sempre executem TOT el codi quan executem tot els *test case*



Black box test coverage

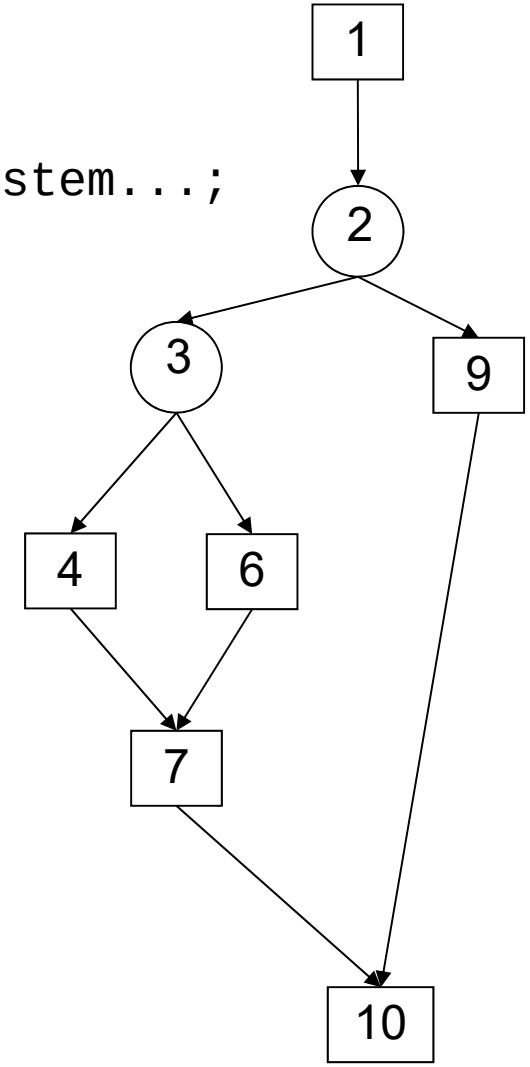
Índex

- Definició
- ***Control flow* (camins bàsics)**

Control flow

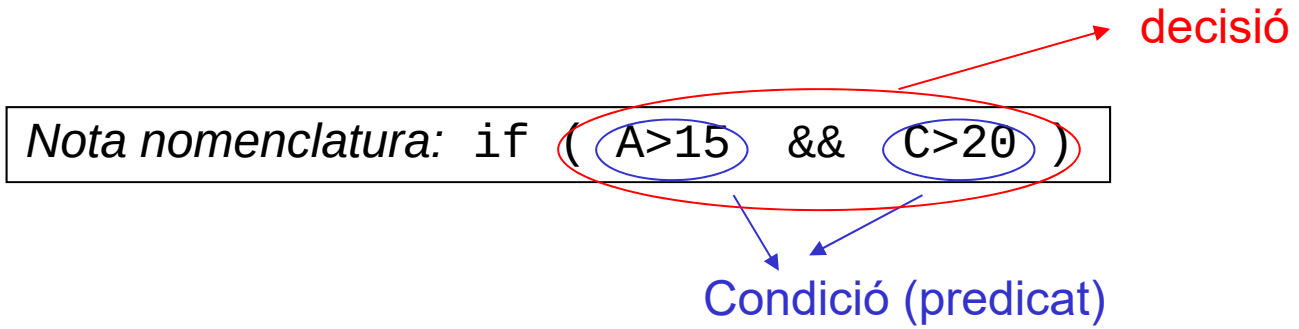
- Suposem que tenim aquest codi

```
... int A,B,C;
1  A=System.in.read(); B=System.in.read(); C=System...;
2  if (A>15 && C>20) {
3    if (B<10)
4      B=A+5;
5    else
6      B=A-5;
7  }
8  else
9    A=B+5;
10 ...
```



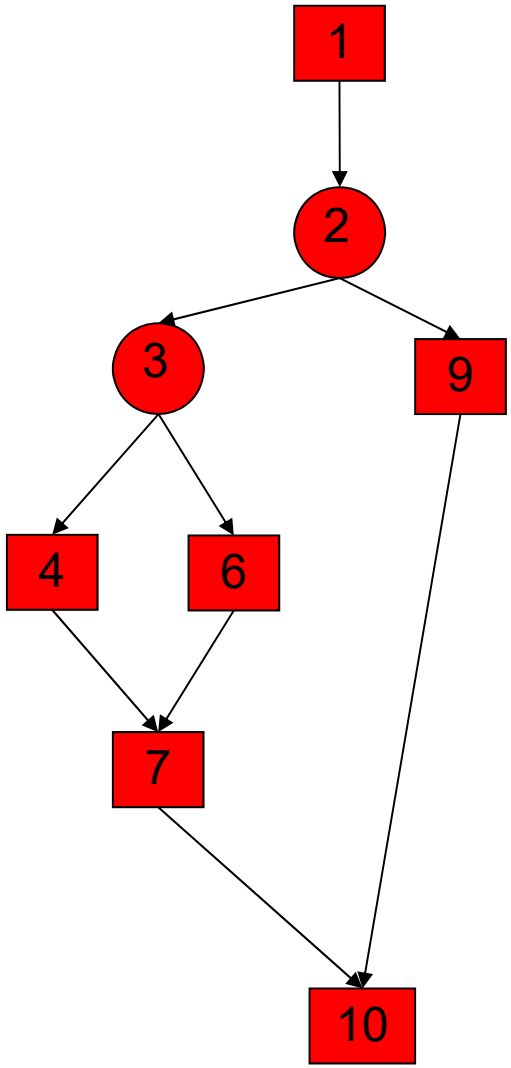
Control flow

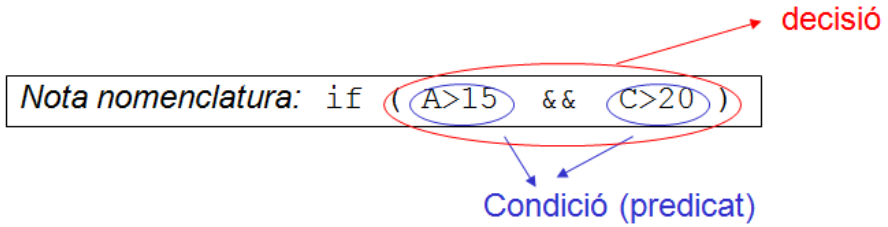
- **Tipus**: Els test cases es generen de manera que ...
  - *Statement coverage*: ... totes les sentències s'executen com a mínim un cop.
  - *Decision (branch) coverage*: ... totes les **decisions** (p.ex. condicionals-`if`) prenen els valors *true* o *false*.
  - *Condition coverage*: ... totes les condicions (**predicats**) que formen l'expressió lògica d'una **decisió** prenen els valors *true* o *false*.
  - *Path coverage*: ... s'executen tots els *path*.



Statement coverage

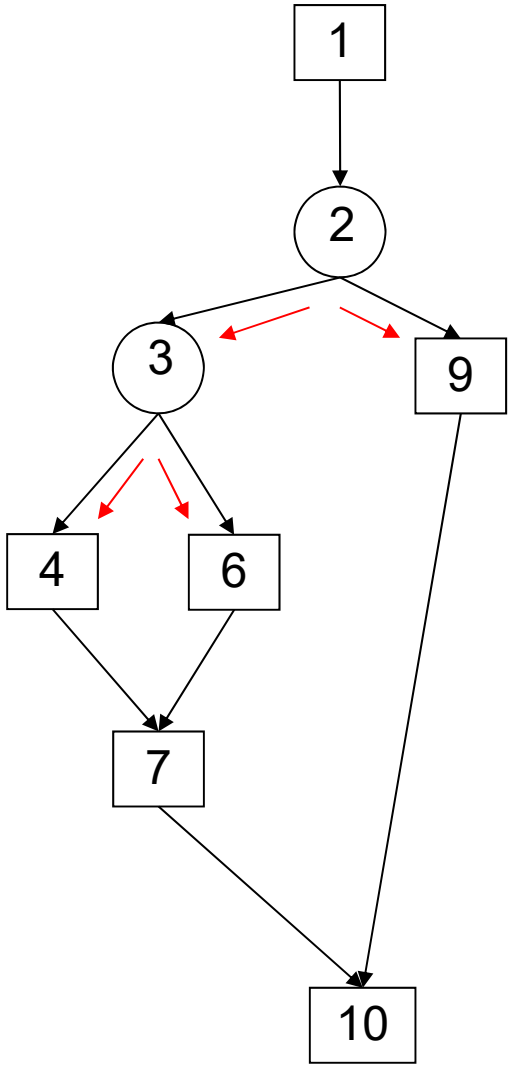
- Executar totes les sentències

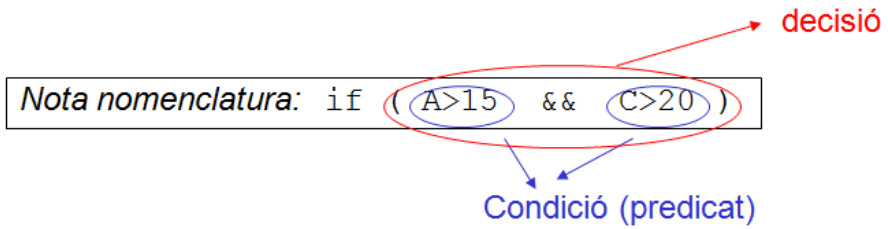




Decision (*branch*) *coverage*

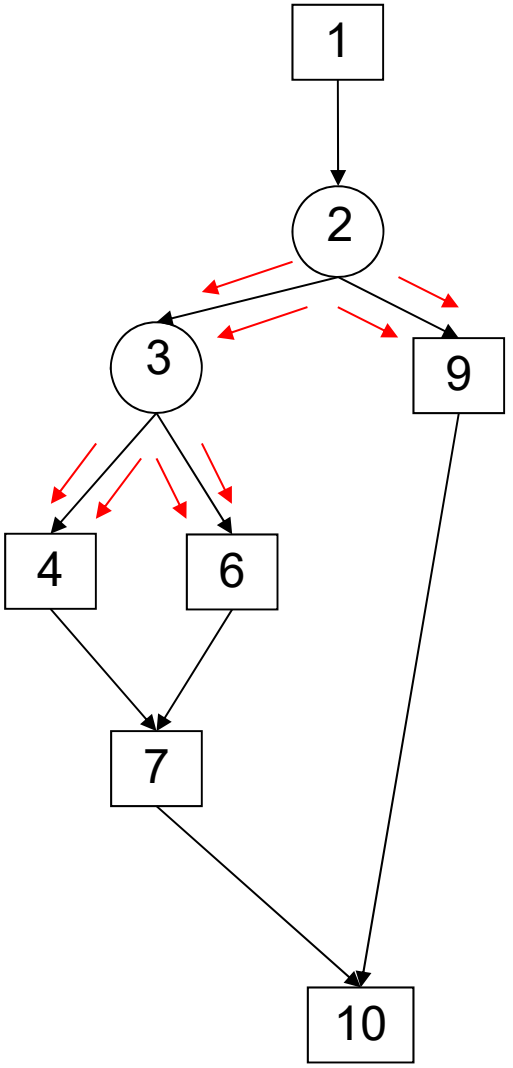
- Totes les **decisions** prenen els valors *true* o *false*

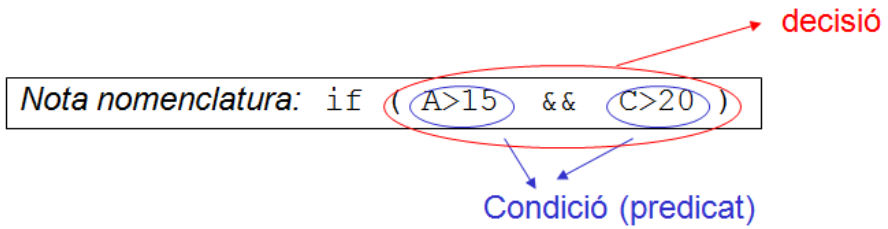




Condition coverage

- Totes les **condicions** (predicats) que formen l'expressió lògica d'una decisió prenen els valors *true* o *false*.

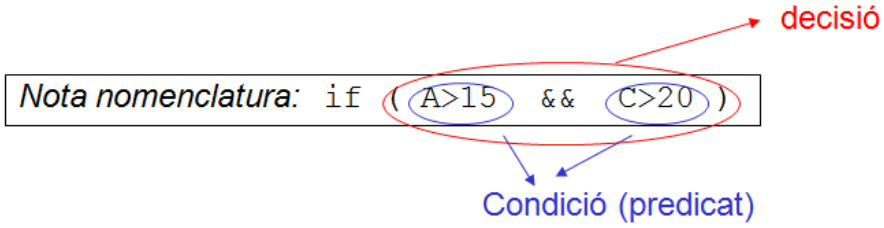




Condition coverage

- *Composed condition*:
  ((X and Y) or C)
  - Els casos de prova han de cobrir tots els valors *true* o *false* de cada condició (predicat) de l'expressió lògica.
    - Explosió de combinacions ⇒ només coverage per *simple condition ...*
    - ... però *simple condition coverage* **no** assegura *decision coverage*
- *Simple condition*: (A)
  - Degenera a *decision coverage*

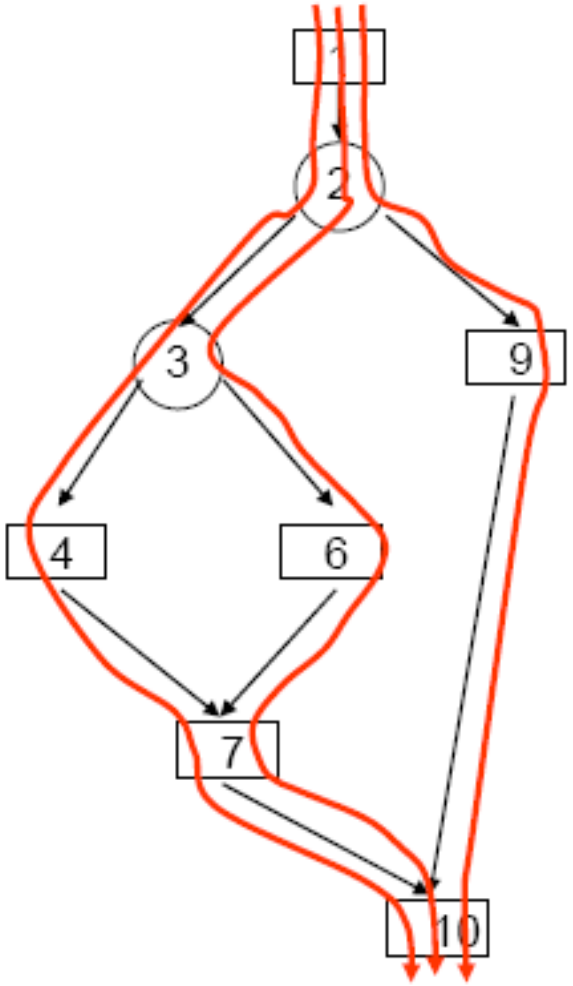
X	Y	Z	(X and Y) or C
true	true	true	true
true	true	false	true
true	false	true	true
true	false	false	false
false	true	true	true
false	true	false	false
false	false	true	true
false	false	false	false



Path coverage

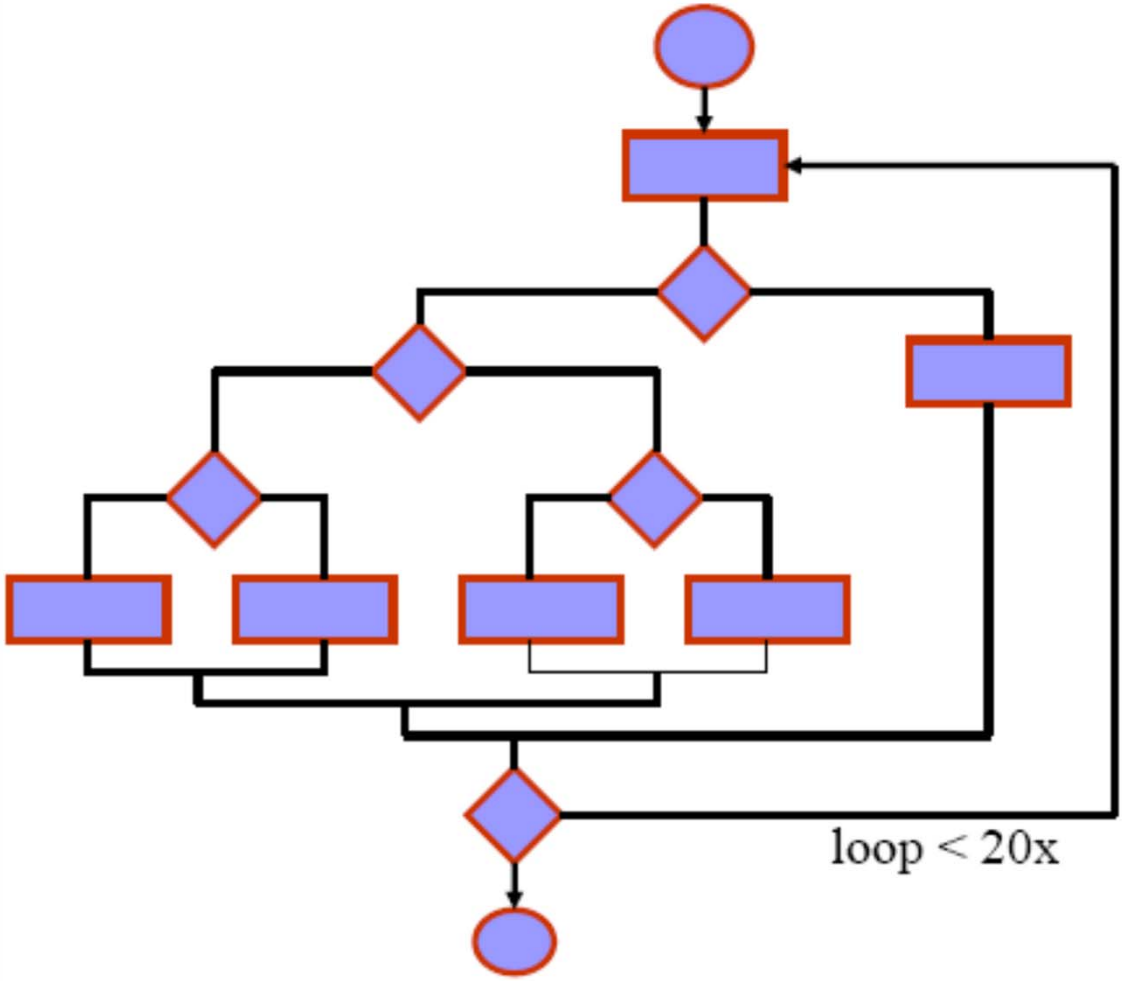
- S'executen tots els *path*

- Per què?
  - **Errors** lògics i assumpcions incorrectes són **inversament proporcionals** a la **probabilitat d'execució del *path***.
  - És probable que un *path* **no testejat** contingui **errors**



Path coverage

- El testing exhaustiu **NO** és possible



Hi ha 10^{14} *paths* possibles!

Si cadascun s'executa en 1ms trigariem 3170 anys a testejar aquest programa!!

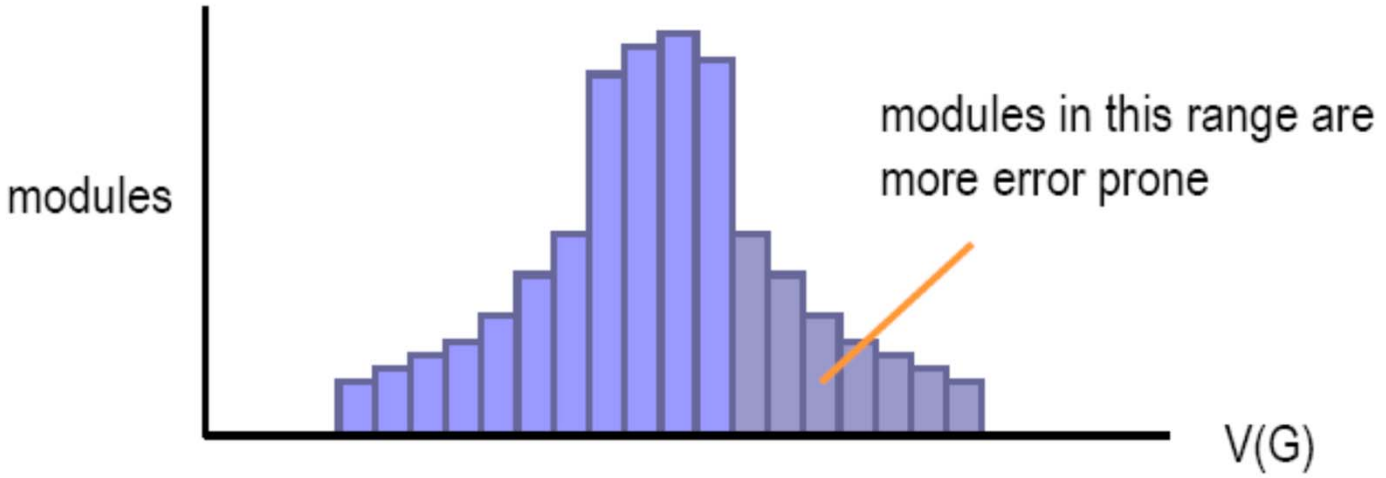
Path coverage

- Passos per realitzar *path coverage*:
  - Convertir el codi (o unitat) en un *flow graph* (diagrama de fluxe)
  - Calcular una mesura de la **complexitat lògica** de la unitat.
  - Utilitzar la mesura per derivar un conjunt “bàsic” de *paths* independents i definir els seus *test cases*.

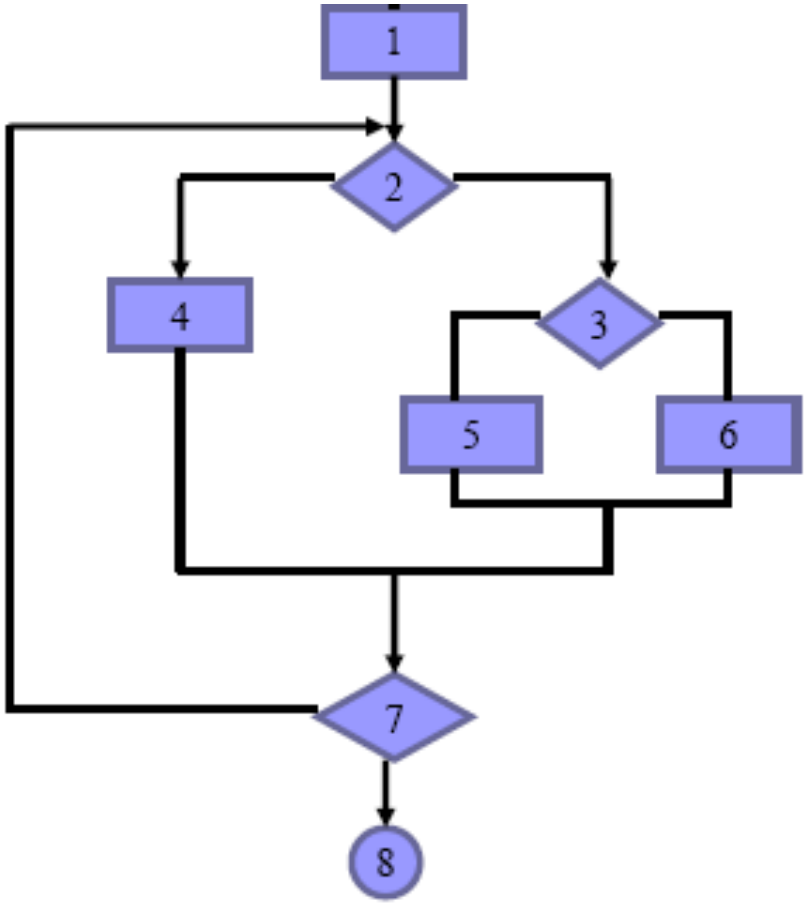
- *Nota*: El conjunt de *paths* per fer *path coverage* **NO** és únic!

Path coverage

- Complexitat ciclomàtica
  - És una mètrica, V(G), que descriu la complexitat lògica d'un diagrama de fluxe (*flow graph*), G.
  - V(G) = E – N + 2 on E=#arcs i N=#nodes a G.
  - Estudis demostren que
    - V(G) està directament relacionat al nombre d'errors
    - V(G)=10 és un límit superior pràctic a l'hora de fer el testing



Path coverage

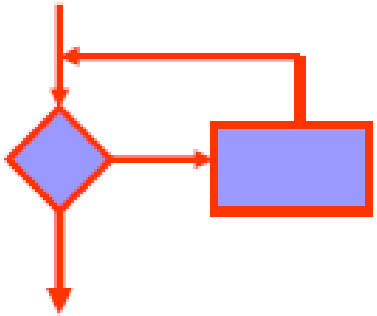


- Trobem els *paths* independents:
  - V(G) = 10 arcs – 8 nodes + 2 = 4 *paths*
  - *Path* 1: 1,2,3,6,7,8
  - *Path* 2: 1,2,3,5,7,8
  - *Path* 3: 1,2,4,7,8
  - *Path* 4: 1,2,4,7,2,4 ... 7,8
- Definim *test cases* per executar aquests *paths*, i per tant paràmetres que ens facin executar aquests *paths*.

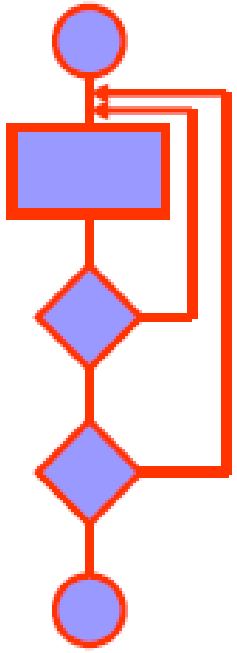
Loop testing

- Tipus de *loops*

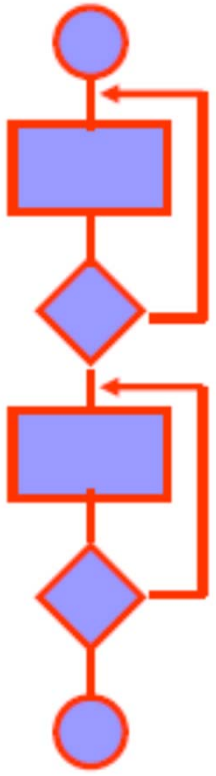
Loop simple



Loops aniuats

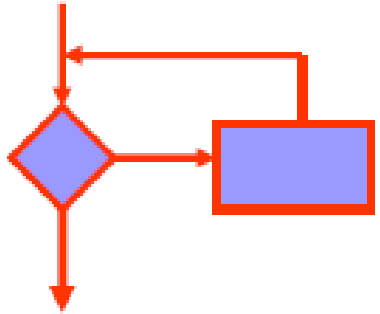


Loops concatenats



Loop testing

- *Loop* simple
  - *Test cases*
    - Evitar el *loop*
    - Una passada pel *loop*
    - Dues passades pel *loop*
    - *m* passades pel *loop m<n*
    - *(n-1)*, *n* passades pel *loop* (*n* és el nombre màxim de passades)



on *n* és el nombre màxim possible de passades

Pregunta: Aquest esquema equival a?

Valors límit pel nombre de passades pel *loop*

Loop testing

- *Loops* aniuats
  - Si extenem el test pel loop simple

    $\Rightarrow$ explosió en el nombre de tests
  - Reduim el nombre de tests
    - Començar amb un test **simple** pel *loop* més **interior**, fixant els demés *loops* al valor mínim
    - Testejar un *loop* més **extern** (com si fos un *loop* **simple**) mantenint el nombre d'iteracions dels *loops* **interiors** a valors **habituals**.
    - Continuar fins testejar tots els *loops*.