White-box testing techniques contd…

Dr. Sangharatna Godboley
Assistant Professor
Department of CSE
NIT Warangal

Slides Credits: Dr. Durga Prasad Mohapatra, Professor, Department of CSE, NIT Rourkela

Path Coverage

- Design test cases such that:
  - All linearly independent paths in the program are executed at least once.
- Defined in terms of
  - Control flow graph (CFG) of a program.

Path Coverage-Based Testing

- To understand the path coverage-based testing:
  - we need to learn how to draw control flow graph of a program.
- A control flow graph (CFG) describes:
  - The sequence in which different instructions of a program get executed.
  - The way control flows through the program.

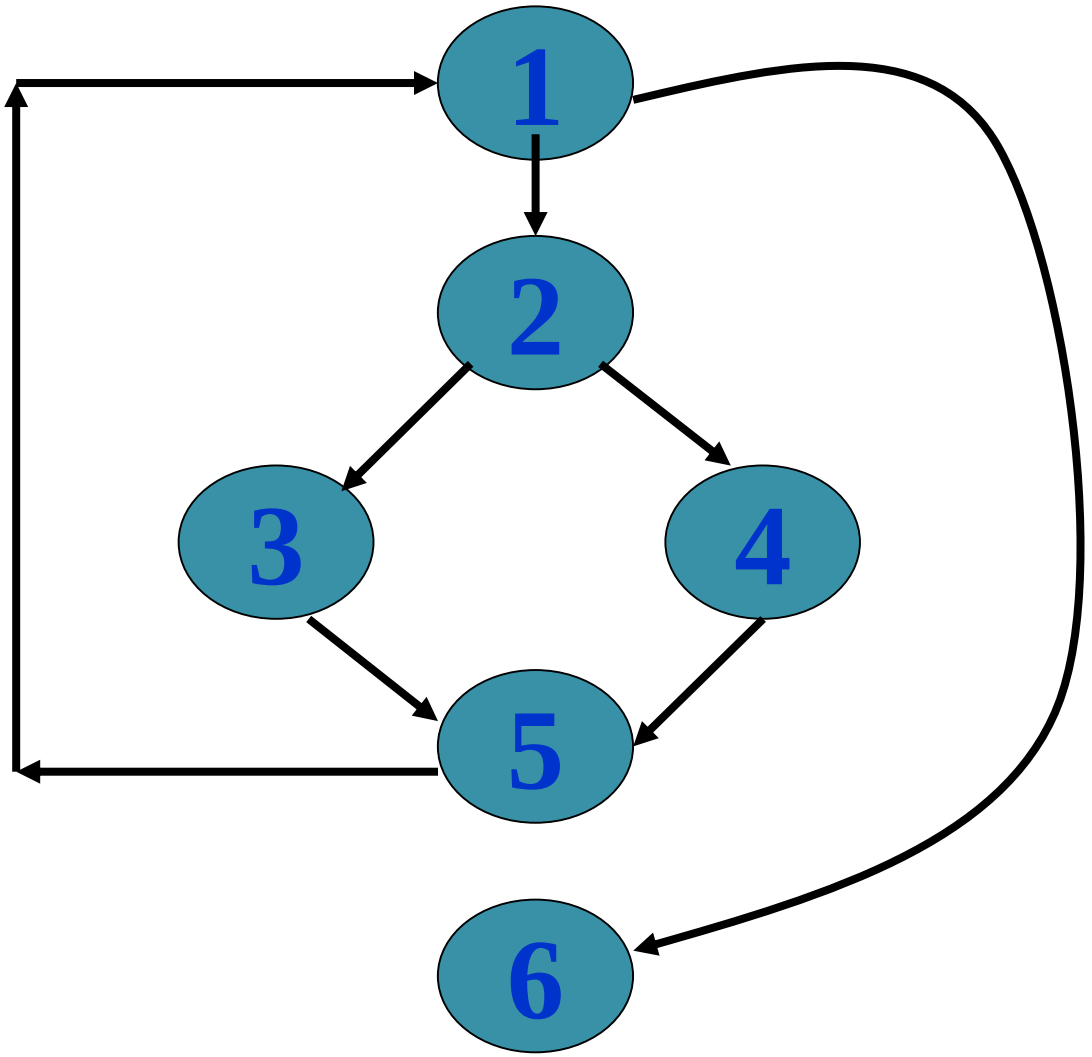
How to Draw Control Flow Graph?

- Number all statements of a program.
- Numbered statements:
  - Represent nodes of control flow graph.
- An edge from one node to another node exists:
  - If execution of the statement representing the first node
    - Can result in transfer of control to the other node.

Example

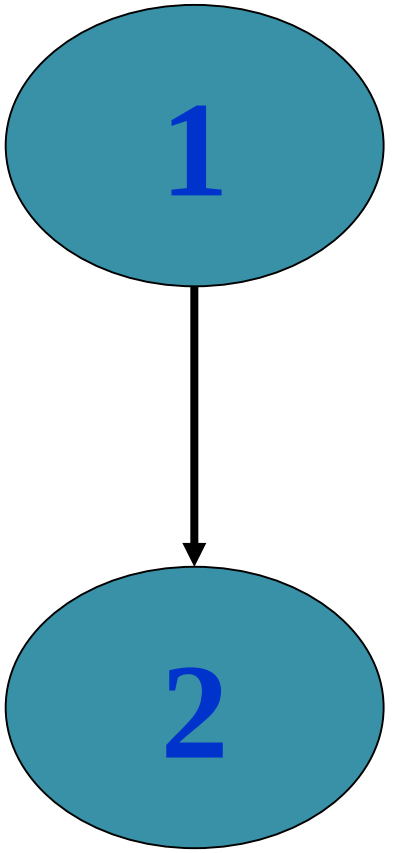
```
int f1(int x,int y){
1 while (x != y){
2    if (x>y) then
3         x=x-y;
4    else y=y-x;
5 }
6 return x;        }
```

Example Control Flow Graph



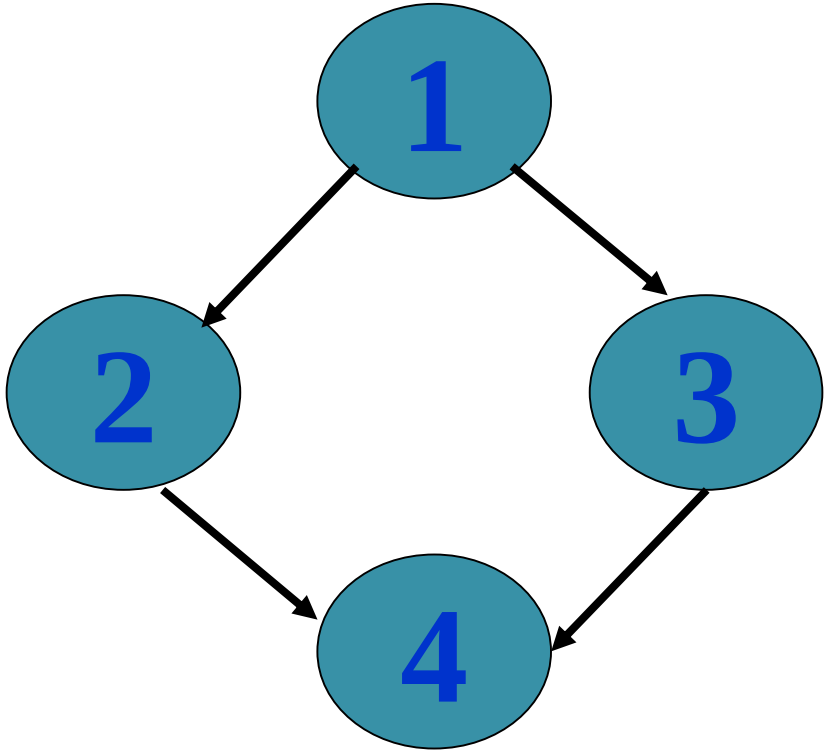
How to Draw Control flow Graph?

- **Sequence:**
  - **1 a=5;**
  - **2 b=a*b-1;**



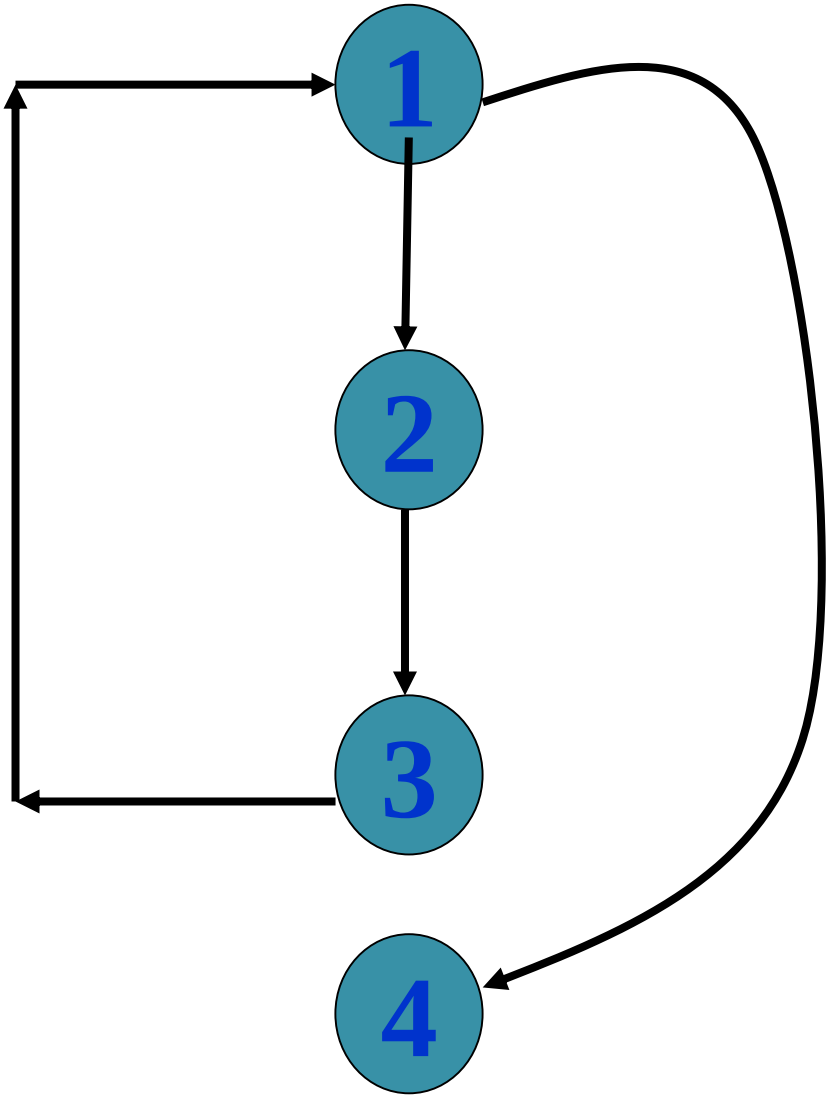
How to Draw Control Flow Graph?

- **Selection:**
  - **1 if(a>b) then**
  - **2 c=3;**
  - **3 else c=5;**
  - **4 c=c*c;**



How to Draw Control Flow Graph?

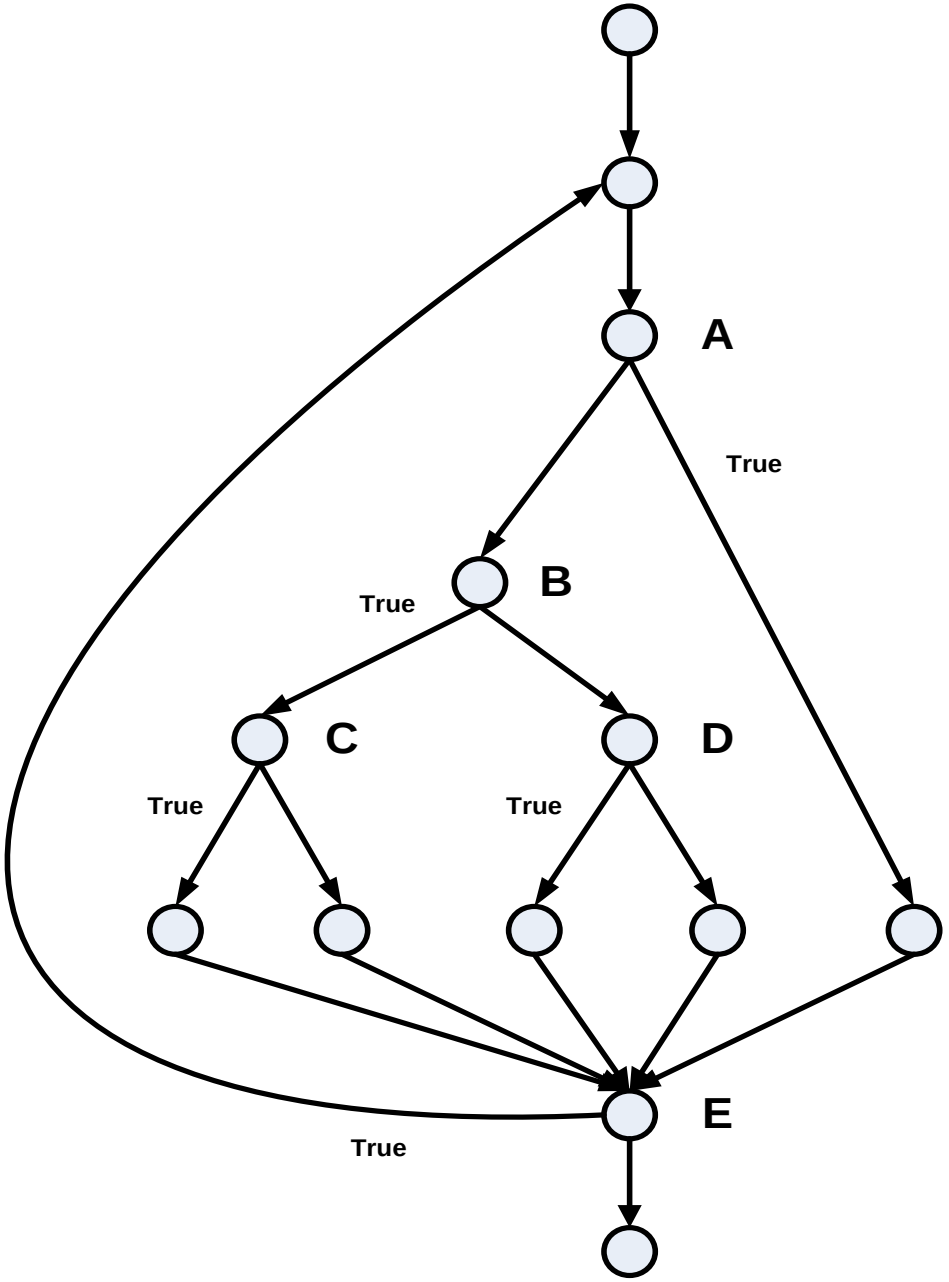
- **Iteration:**
  - **1 while(a>b){**
  - **2 b=b*a;**
  - **3 b=b-1;}**
  - **4 c=b+d;**



Example Code Fragment

```
Do
{
  if (A) then {...};
  else {
    if (B) then {
              if (C) then {...};
              else {…}
    }
    else if (D) then {...};
         else {...};
  }
}
While (E);
```

Example Control Flow Graph



Source: The Art of Software Testing – Glenford Myers

Path

- A path through a program:
  - A node and edge sequence from the starting node to a terminal node of the control flow graph.
  - There may be several terminal nodes for program.

Linearly Independent Path

- Any path through the program:
  - Introduces at least one new edge:
    - Not included in any other independent paths.

Independent path

- It is straight forward:
  - To identify linearly independent paths of simple programs.
- For complicated programs:
  - It is not easy to determine the number of independent paths.

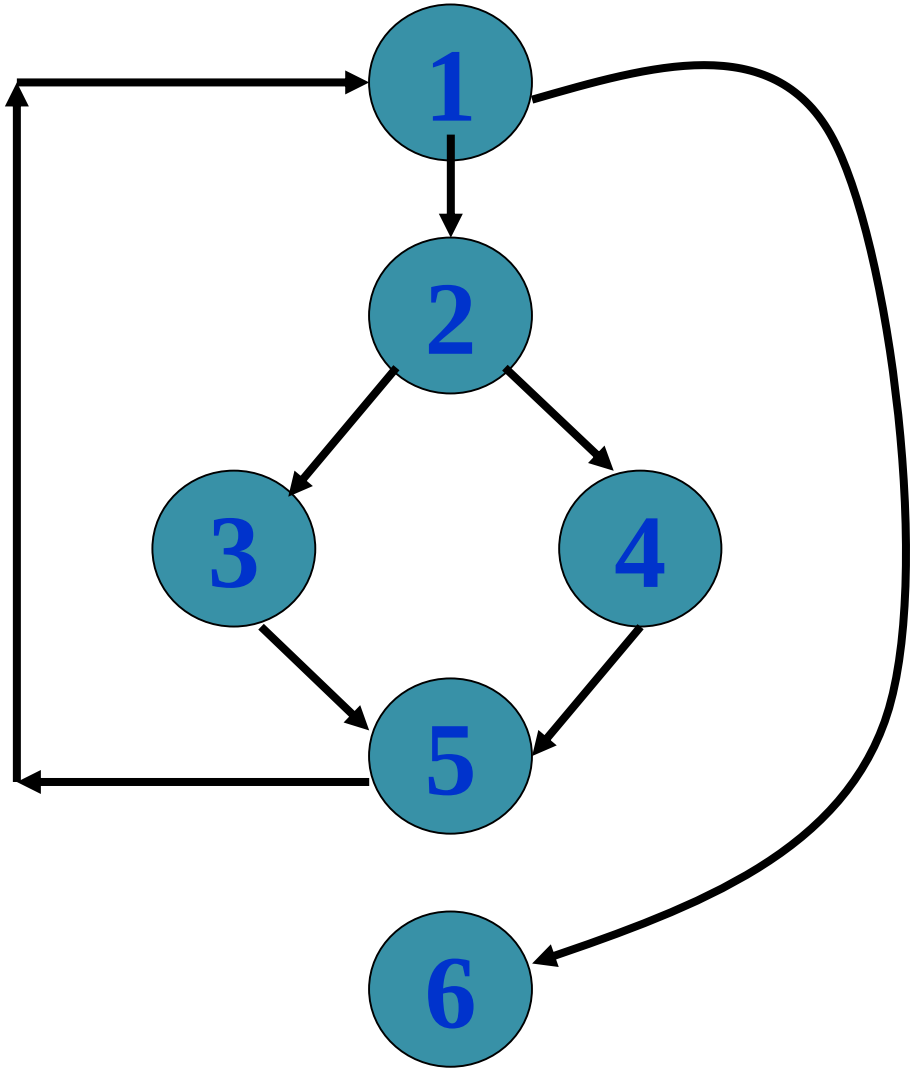
McCabe's Cyclomatic Metric

- An upper bound:
  - For the number of linearly independent paths of a program
- Provides a practical way of determining:
  - The maximum number of linearly independent paths in a program.

McCabe's Cyclomatic Metric

- Given a control flow graph G, cyclomatic complexity V(G):
  - V(G)= E-N+2
    - N is the number of nodes in G
    - E is the number of edges in G

Example Control Flow Graph

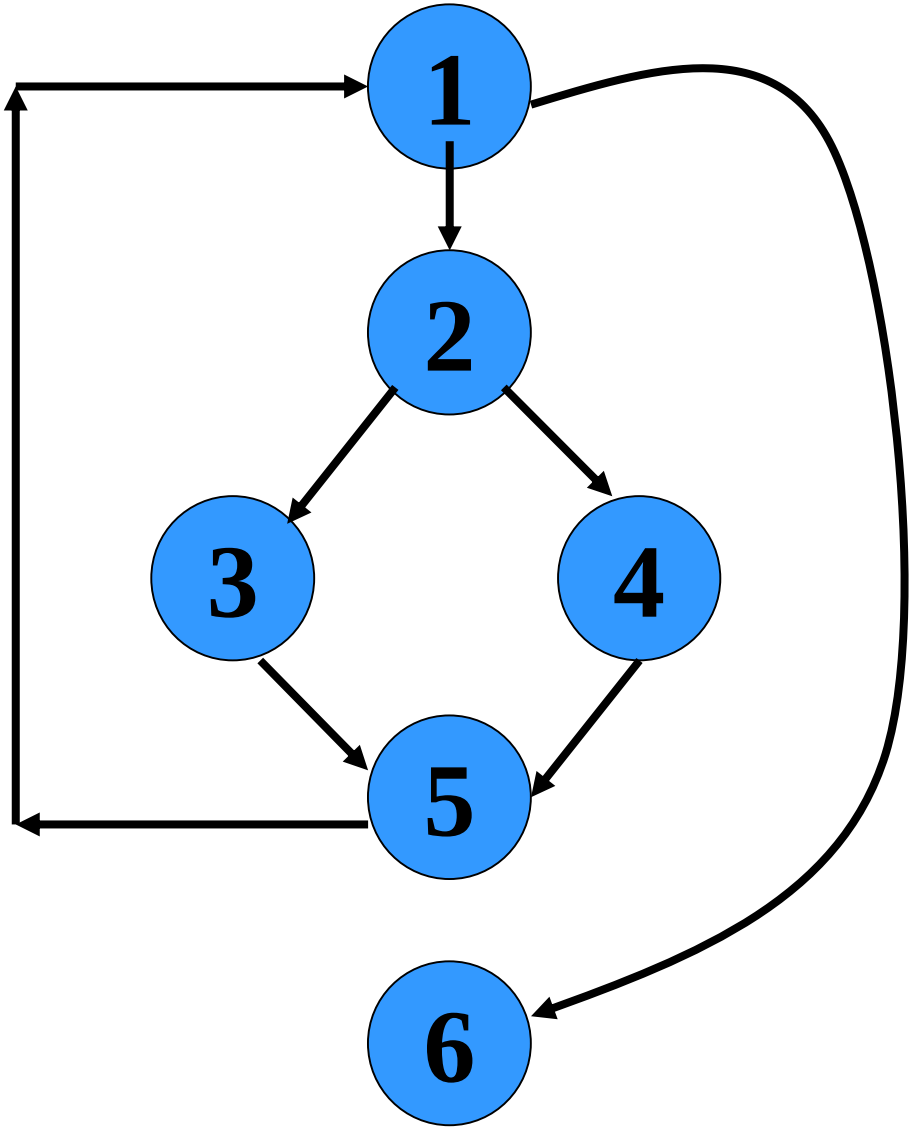


Cyclomatic complexity = 7-6+2 = 3.

Cyclomatic Complexity

- Another way of computing cyclomatic complexity:
  - inspect control flow graph
  - determine number of bounded areas in the graph
- V(G) = Total number of bounded areas + 1
  - Any region enclosed by a nodes and edge sequence.

Example Control Flow Graph



Example

- From a visual examination of the CFG:
  - Number of bounded areas is 2.
  - Cyclomatic complexity = 2+1=3.

Cyclomatic Complexity

- McCabe's metric provides:
  - A quantitative measure of testing difficulty and the ultimate reliability
- Intuitively,
  - Number of bounded areas increases with the number of decision nodes and loops.

Cyclomatic Complexity

- The first method of computing V(G) is amenable to automation:
  - You can write a program which determines the number of nodes and edges of a graph
  - Applies the formula to find V(G).

Cyclomatic Complexity

- The cyclomatic complexity of a program provides:
  - A lower bound on the number of test cases to be designed
  - To guarantee coverage of all linearly independent paths.

Cyclomatic Complexity

- A measure of the number of independent paths in a program.
- Provides a lower bound:
  - for the number of test cases for path coverage.

Cyclomatic Complexity

- Knowing the number of test cases required:
  - Does not make it any easier to derive the test cases,
  - Only gives an indication of the minimum number of test cases required.

Practical Path Testing

- The tester proposes initial set of test data :
  - Using his experience and judgment.
- A dynamic program analyzer used:
  - Measures which parts of the program have been tested
  - Result used to determine when to stop testing.

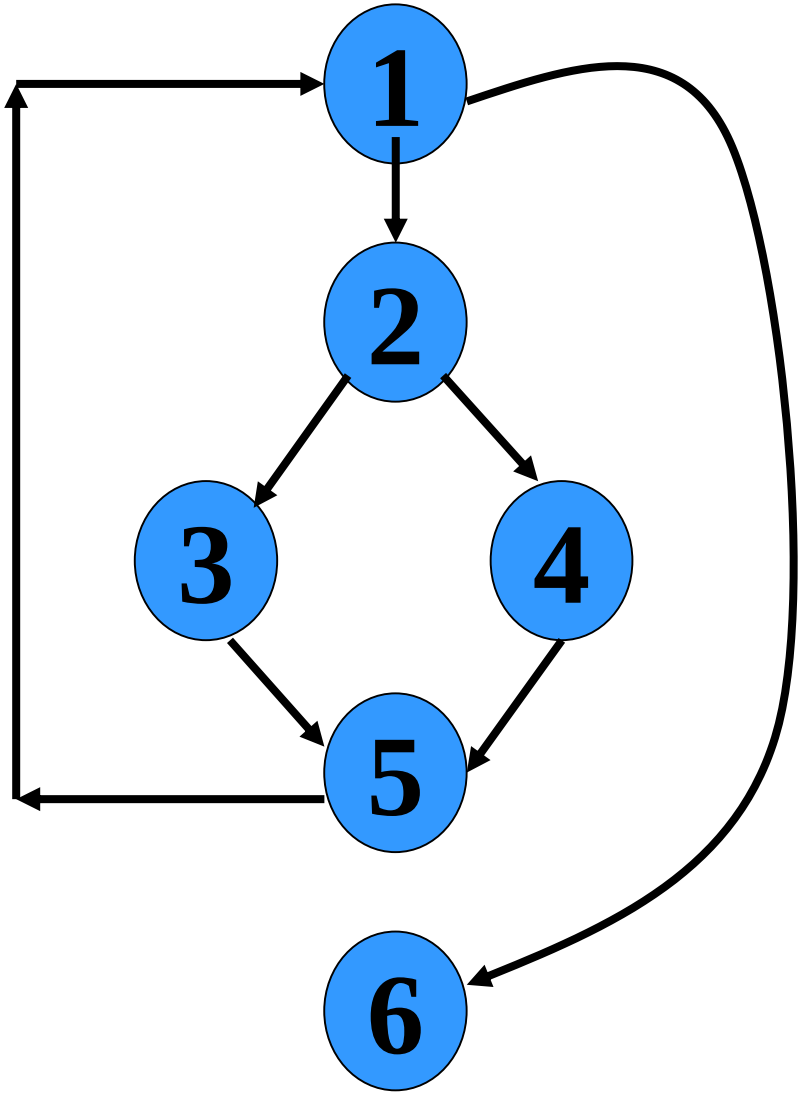
Derivation of Test Cases

- Draw control flow graph.
- Determine V(G).
- Determine the set of linearly independent paths.
- Prepare test cases:
  - to force execution along each path.

Example

```
int f1(int x,int y){
1 while (x != y){
2     if (x>y) then
3           x=x-y;
4     else y=y-x;
5 }
6 return x;          }
```

Example Control Flow Diagram



Derivation of Test Cases

- **Number of independent paths: 3**
  - **1,6** **test case (x=1, y=1)**
  - **1,2,3,5,1,6** **test case(x=2, y=1)**
  - **1,2,4,5,1,6** **test case(x=1, y=2)**

An Interesting Application of Cyclomatic Complexity

- Relationship exists between:
  - McCabe's metric
  - The number of errors existing in the code,
  - The time required to find and correct the errors.

Cyclomatic Complexity

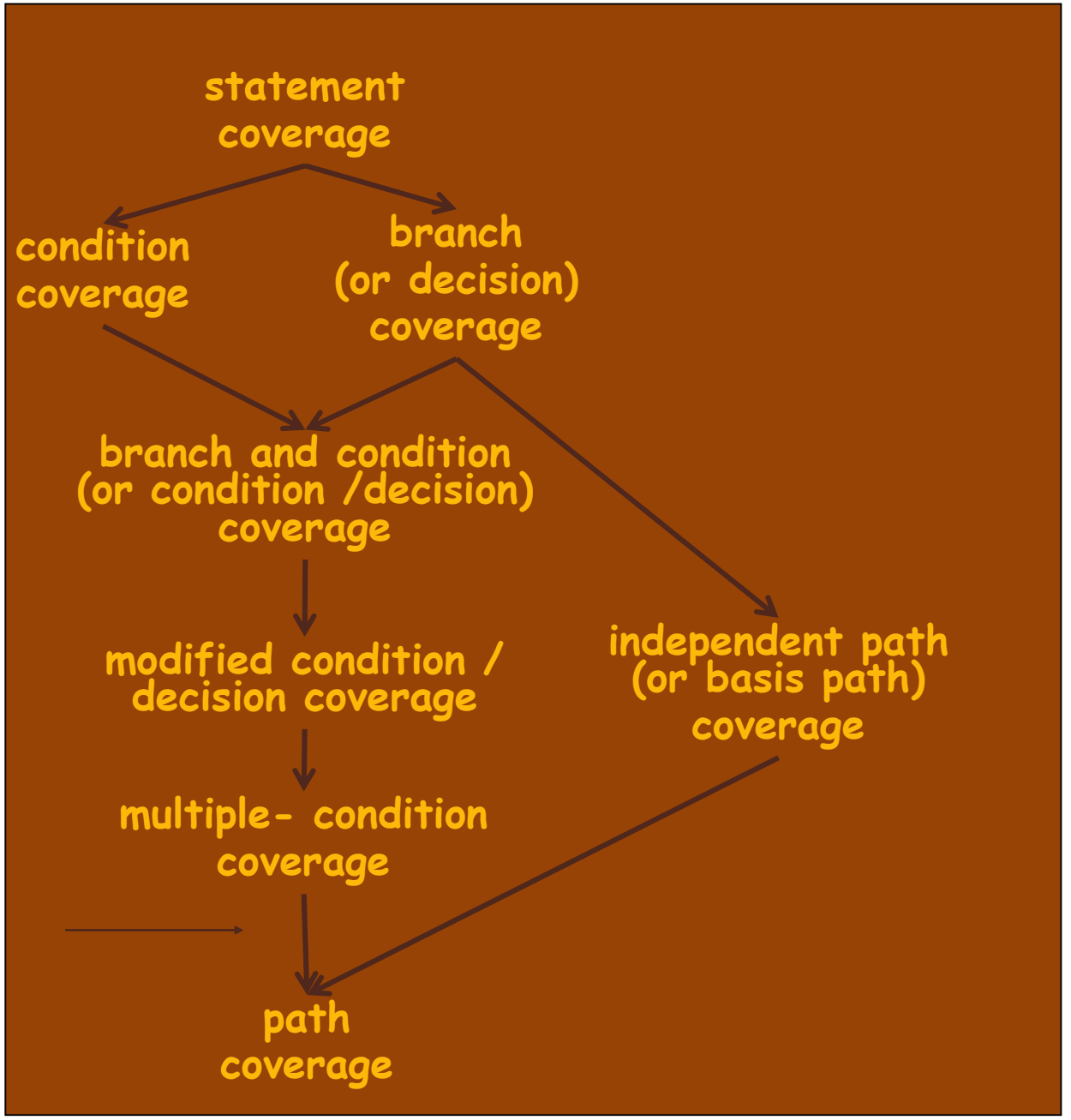
- Cyclomatic complexity of a program:
  - Also indicates the psychological complexity of a program.
  - Difficulty level of understanding the program.

Cyclomatic Complexity

- From maintenance perspective,
  - Limit cyclomatic complexity of modules
    - To some reasonable value.
  - Good software development organizations:
    - Restrict cyclomatic complexity of functions to a maximum of ten or so.

White-Box Testing : Summary

weakest



only if paths across composite conditions are distinguished

strongest

Summary

- There are two approaches to testing:
  - black-box testing and
  - white-box testing.
- Designing test cases for black box testing:
  - does not require any knowledge of how the functions have been designed and implemented.
  - Test cases can be designed by examining only SRS document.

Summary

- White box testing:
  - Requires knowledge about internals of the software.
  - Design and code is required.
- We have discussed a few white-box test strategies.
  - Statement coverage
  - branch coverage
  - condition coverage
  - path coverage

Summary

- A stronger testing strategy:
  - Provides more number of significant test cases than a weaker one.
  - Condition coverage is strongest among strategies we discussed.
- We discussed McCabe's Cyclomatic complexity metric:
  - Provides an upper bound for linearly independent paths
  - Correlates with understanding, testing, and debugging difficulty of a program.

Thank you