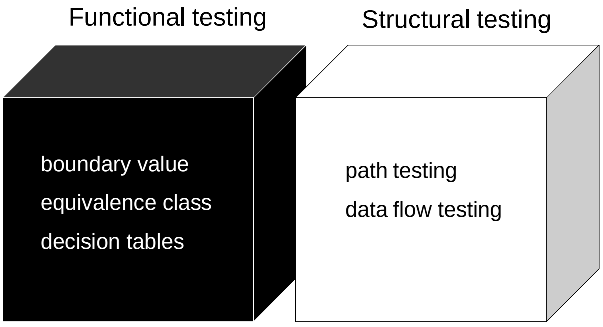
9. Path testing

Control Flow Graph
Graphs
DD-paths
Independent paths
McCabe’s Cyclomatic Metric
Derivation of test cases

Techniques



Functional testing

- boundary value
- equivalence class
- decision tables

Structural testing

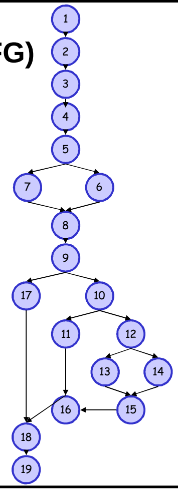
- path testing
- data flow testing

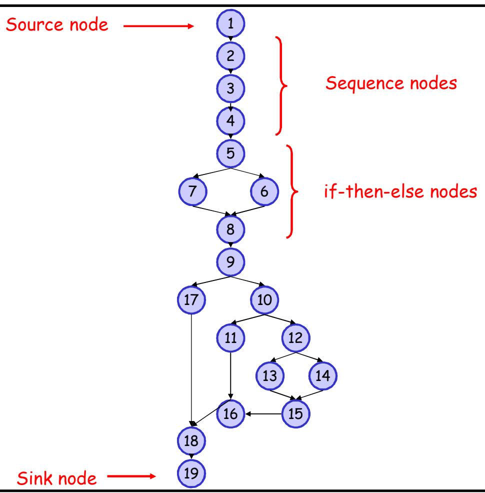
Path testing

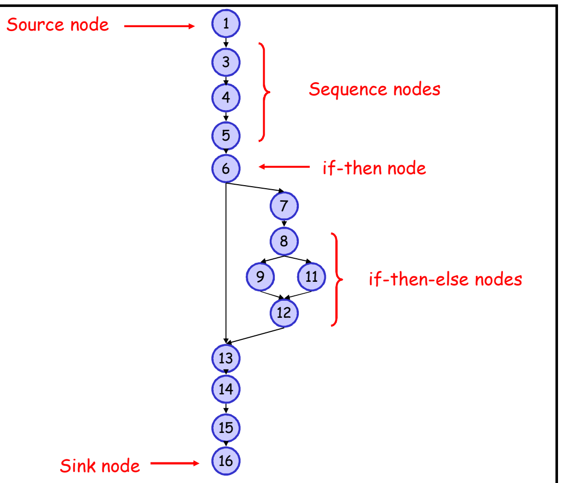
- Structural testing method
- Based on the source code / pseudocode of the program or the system, and NOT on its specification
- Primarily used for testing imperative-style programs/designes
- Can be applied at different levels of granularity

Control Flow Graph (CFG)

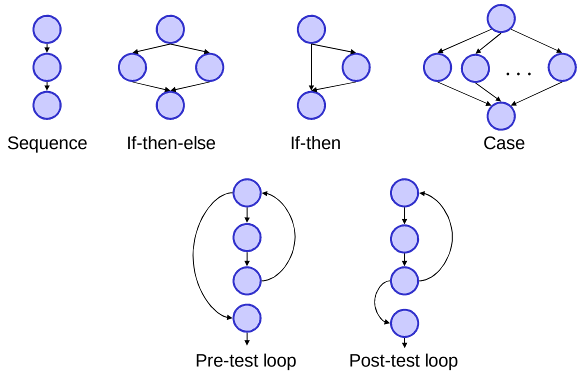
```
1   program TRIANGLE
2   input (a)
3   input (b)
4   input (c)
5   if (a<b+c) AND (b<a+c) AND (c<a+b)
6       then IsATriangle = T
7       else IsATriangle = F
8   endif
9   if IsATriangle
10      then if (a=b) AND (b=c)
11          then Output = “Equilateral”
12          else if (a != b) AND (b != c) AND (a != c)
13               then Output = “Scalene”
14               else Output = “Isosceles”
15          endif
16      endif
17  else Output = “Not a triangle”
18  endif
19  end TRIANGLE
```



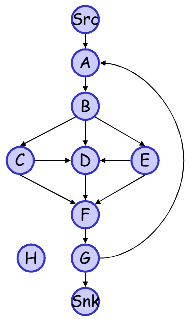




Summary of the node types



Graphs - terminology



- Nodes (n) and edges (e)
- Directed / undirected graph
- Degree of a node deg(n)
  - The number of edges that have that node as an endpoint
- Indegree of a node
  - The number of edges that have the node as a terminal node
- Outdegree of a node
  - The number of edges that have the node as a start node
- Number of components (p)
  - Component is a maximal set of connected nodes
- Cyclomatic number of graph G

  V(G) = e – n + 2p

Graphs - exercise

- Specify degree of nodes A to H
- Specify indeg of nodes B and D
- Specify outdeg of nodes B and D
- What is indeg of the source node?
- What is outdeg of the sink node?
- Compute the cyclomatic number of the graph

Graphs - exercise

- Specify degree of nodes A to H

A	B	C	D	E	F	G	H
3	4	3	4	3	4	3	0

- Specify indeg of nodes B and D
  - indeg(B) = 1 indeg(D) = 3
- Specify outdeg of nodes B and D
  - outdeg(B) = 3 outdeg(D) = 1
- What is indeg of the source node?
  - Always 0
- What is outdeg of the sink node?
  - Always 0
- Compute the cyclomatic number of the graph
  - V(G) = e – n + 2p = 13 – 10 + 4 = 7

DD – path (Logical Branch)

- **DD: Decision – to – Decision path**
- A DD-path is a sub-path in a program graph fulfilling one of the conditions below:
  1. It consists of a single node with indeg = 0
  2. It consists of a single node with outdeg = 0

     Ensures the unique source and sink nodes
  3. It consists of a single node with indeg ≥ 2 or outdeg ≥ 2

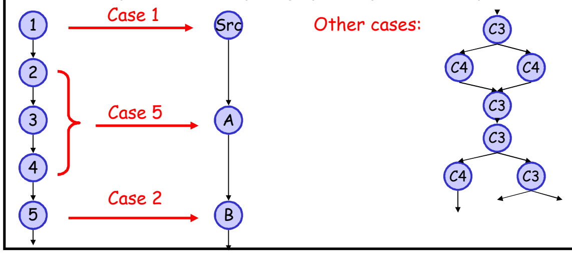
     No node is contained in more than one DD-path
  4. It consists of a single node with indeg = 1 and outdeg = 1

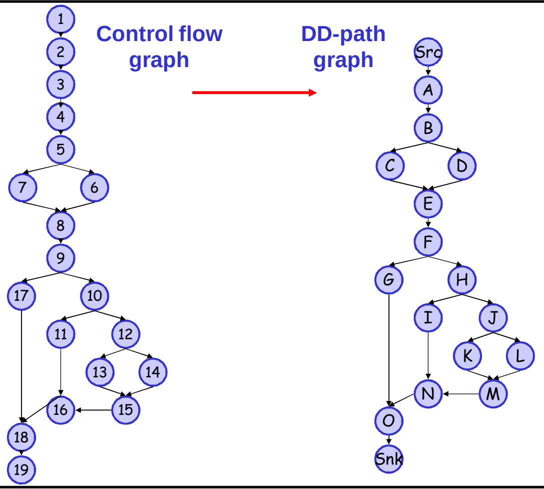
     Ensures the “one fragment one DD path” mapping
  5. It is a maximal chain of length ≥ 1

     Single entry – single exit sequence of nodes

DD-path graphs

- A CFG can be broken into DD-paths
- Each DD-path is collapsed into an individual node
- The resulting graph is called a DD-path graph of the program
- Every node in a DD-path graph is equivalent to a predicate



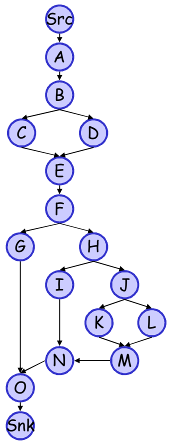


Independent (basis) paths

- Independent path is a path through a DD-path graph of the program

  (i.e. it a graph which has at least one source node and one sink node)

  which cannot be reproduced from other paths by
  - Addition (i.e. one path following another)
  - Repetition (e.g. loop)
- The concept is similar to that of the “basis functions”



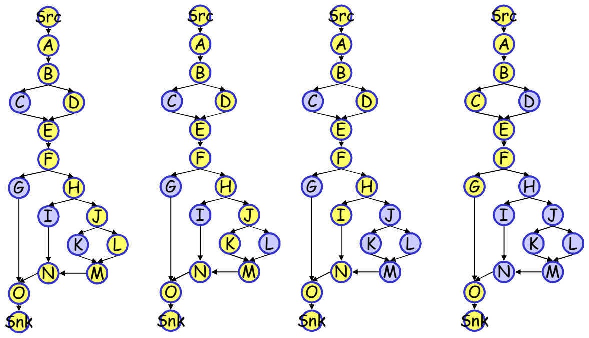
Independent paths

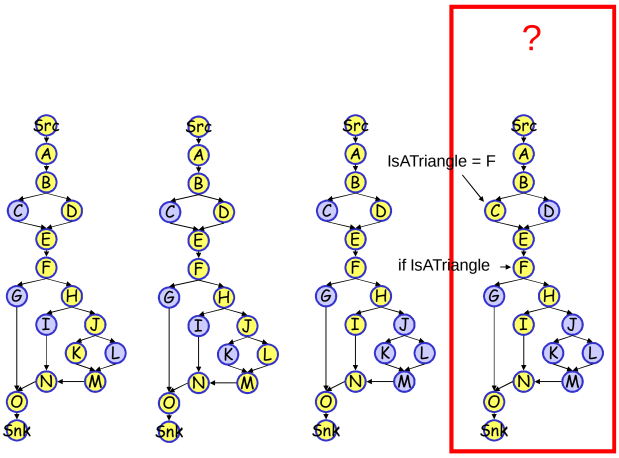
Find at least one independent path in the graph

P1: Src-A-B-D-E-F-H-J-K-M-N-O-Snk
P2: Src-A-B-D-E-F-H-J-L-M-N-O-Snk
P3: Src-A-B-D-E-F-H-I-N-O-Snk
P4: Src-A-B-C-E-F-G-O-Snk

P1: Scalene
P2: Isosceles
P3: Equilateral
P4: Not a triangle

Src-A-B-D-E-F-H-J-K-M-N-O-Snk
Src-A-B-D-E-F-H-J-L-M-N-O-Snk
Src-A-B-D-E-F-H-I-N-O-Snk
Src-A-B-C-E-F-G-O-Snk





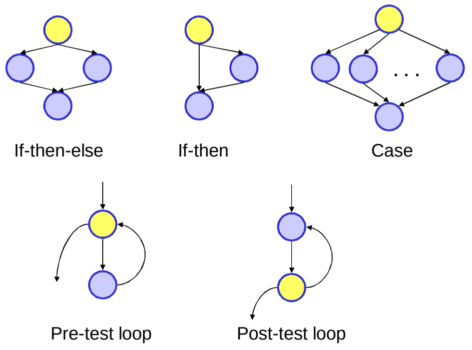
Independent paths

Lesson: **Paths must be feasible**

Generating independent paths

- Generate one feasible path (a “baseline” path)
- Generate further paths by “flipping” each decision point in turn
  - Decision point is a node with outdegree ≥ 2
  - “Flipping” is taking a different edge than those taken previously
  - A “technically” feasible path may not be feasible “logically” (according to the logic of the program)

Decision points: nodes with outdeg ≥ 2



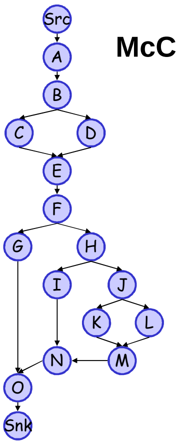
McCabe cyclomatic metric

- The number of independent paths can be predicted from a DD-path graph of the program
- Cyclomatic number of graph G

  V(G) = e – n + 2p

  gives (*approximately!*) the number of independent paths

McCabe cyclomatic metric



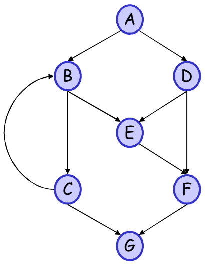
V(G) = e – n + 2p

V(G) = 20 – 17 + 2 = 5

Five independent paths

Exercise

Compute McCabe cyclomatic metric for the graph



V(G) = e – n + 2p

V(G) = 10 – 7 + 2 = 5
5 independent paths

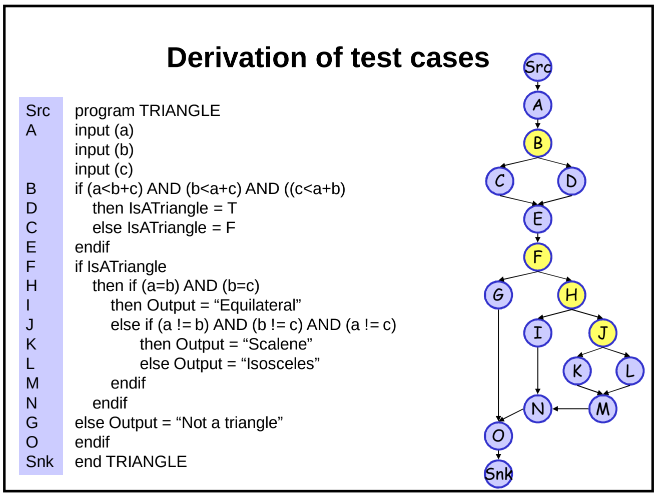
P1:	A-B-C-G
P2:	A-B-C-B-C-G
P3:	A-B-E-F-G
P4:	A-D-E-F-G
P5:	A-D-F-G

McCabe cyclomatic metric

- In practical terms the McCabe's cyclomatic metric defines a lower bound on the number of tests for the Path Coverage
- The metric also gives an intuitive feel for program complexity in terms of the number of decision nodes and loops

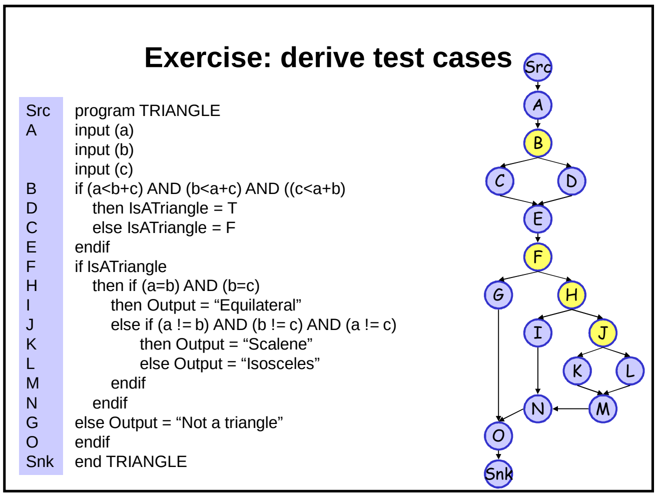
Derivation of test cases

- Determine a DD-path graph for the program
- Determine the cyclomatic number V(G)
  - this tells you approximately how many tests to generate
- Generate test cases in the same way as the Independent Paths are generated
  - i.e. each test case will represent a different combination of the states of the Gate Variables
- There exist tools for semi-automation of the Path Testing



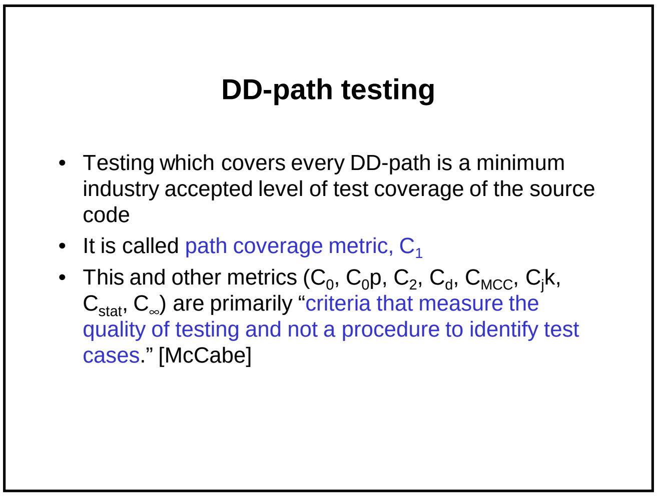
Derivation of test cases

```
Src   program TRIANGLE
A     input (a)
      input (b)
      input (c)
B     if (a<b+c) AND (b<a+c) AND ((c<a+b)
D        then IsATriangle = T
C        else IsATriangle = F
E     endif
F     if IsATriangle
H        then if (a=b) AND (b=c)
I           then Output = “Equilateral”
J           else if (a != b) AND (b != c) AND (a != c)
K                then Output = “Scalene”
L                else Output = “Isosceles”
M           endif
N        endif
G     else Output = “Not a triangle”
O     endif
Snk   end TRIANGLE
```



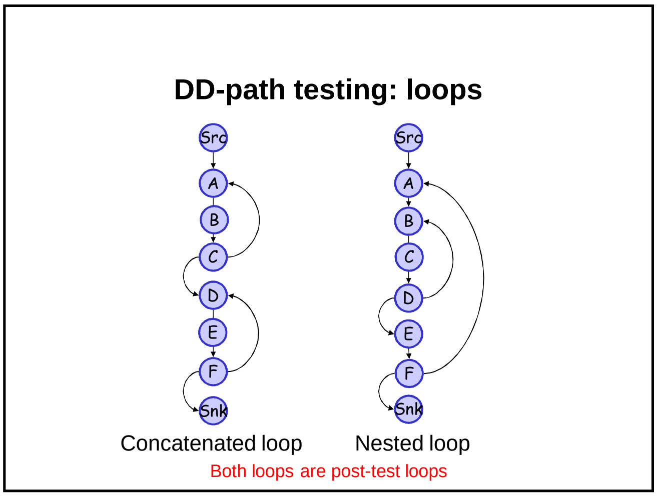
Exercise: derive test cases

```
Src   program TRIANGLE
A     input (a)
      input (b)
      input (c)
B     if (a<b+c) AND (b<a+c) AND ((c<a+b)
D        then IsATriangle = T
C        else IsATriangle = F
E     endif
F     if IsATriangle
H        then if (a=b) AND (b=c)
I           then Output = “Equilateral”
J           else if (a != b) AND (b != c) AND (a != c)
K                then Output = “Scalene”
L                else Output = “Isosceles”
M           endif
N        endif
G     else Output = “Not a triangle”
O     endif
Snk   end TRIANGLE
```



DD-path testing

- Testing which covers every DD-path is a minimum industry accepted level of test coverage of the source code
- It is called path coverage metric, C_1
- This and other metrics (C_0, C_0p, C_2, C_d, C_{MCC}, C_jk, C_{stat}, C_∞) are primarily “criteria that measure the quality of testing and not a procedure to identify test cases.” [McCabe]

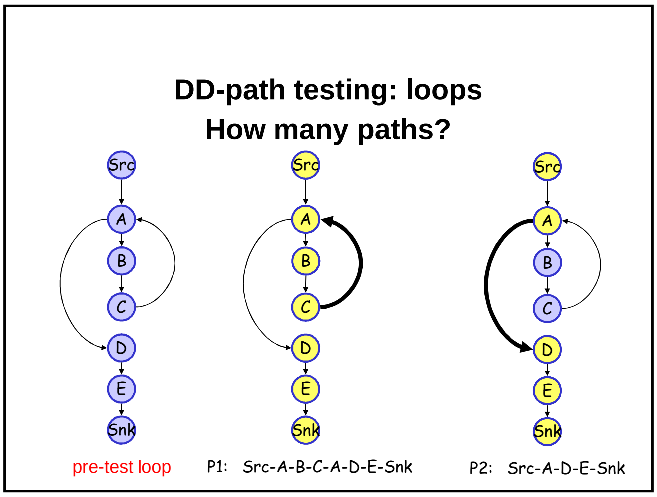


DD-path testing: loops

Concatenated loop

Nested loop

Both loops are post-test loops

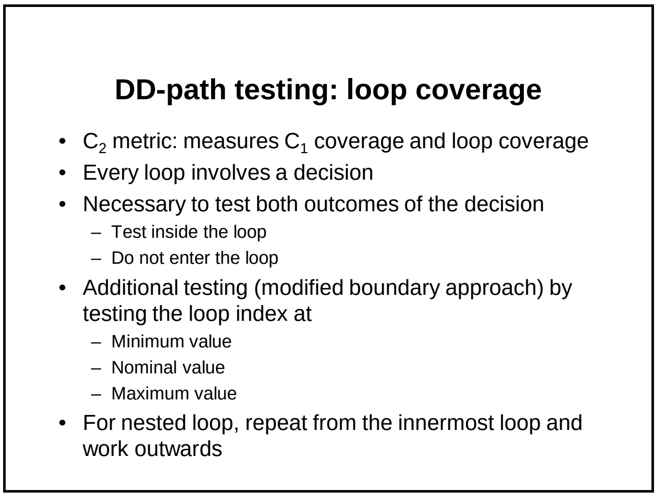


DD-path testing: loops
How many paths?

pre-test loop

P1: Src-A-B-C-A-D-E-Snk

P2: Src-A-D-E-Snk



DD-path testing: loop coverage

- C_2 metric: measures C_1 coverage and loop coverage
- Every loop involves a decision
- Necessary to test both outcomes of the decision
  - Test inside the loop
  - Do not enter the loop
- Additional testing (modified boundary approach) by testing the loop index at
  - Minimum value
  - Nominal value
  - Maximum value
- For nested loop, repeat from the innermost loop and work outwards

Path Testing - conclusions

- Based on code – complementary to functional methods
- Provides useful metrics, especially valuable for discovering redundancy in the number of test cases
- Metrics also useful for software testing quality assurance

- Cumbersome to use
- Does not make distinction between the feasible and infeasible paths

Next lecture

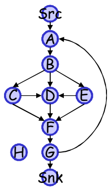
Data Flow Testing

Further reading

- Additional material on the web
  http://www.cs.bham.ac.uk/~exc/Teaching/STesting
  - Using the Cyclomatic Complexity Metric
  - BCS Standard for Software Component Testing

Homework

- Re-write the Triangle program segment 7-14 so that the compound conditions are replaced by nested if-then-else statements. Compare the cyclomatic complexity of the new program with that of the existing version.
- Use the whiteBox.exe program (the Course resource page) to experiment with various sets of test cases to determine DD-path coverage for the Triangle problem and the NextDate problem
- Draw a CFG for two nested pre-test loops and specify all the possible paths through the graph.



- Specify all the directed paths between nodes C and Snk for the graph on the left.