White box testing- basis path testing

Madheswari.K

AP/CSE

White box testing

- *White-box testing, sometimes called glass-box testing, is a test-case design philosophy* that uses the control structure described as part of component-level design to derive test cases.
- Using white-box testing methods, you can derive test cases that

(1) guarantee that all independent paths within a module have been exercised at least once,

(2) exercise all logical decisions on their true and false sides,

(3) Execute all loops at their boundaries and within their operational bounds, and

(4) Exercise internal data structures to ensure their validity.

Basis path testing

- *Basis path testing is a white-box testing technique first proposed by Tom McCabe*
- The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths.
- Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

Basis path testing contd…

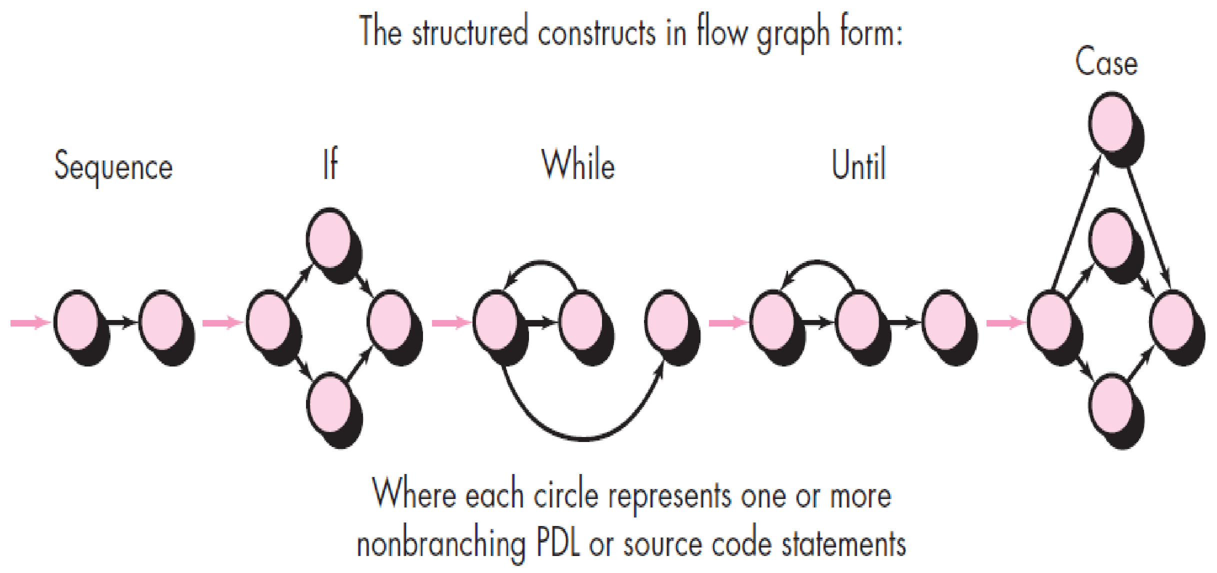
- Flow graph Notation
- Independent Program Paths
- Deriving test cases
- Graph matrices

Flow graph Notation

- a simple notation for the representation of control flow, called a *flow graph (or program graph)*
- The flow graph depicts logical control flow using the notations.

FIGURE 18.1

Flow graph notation

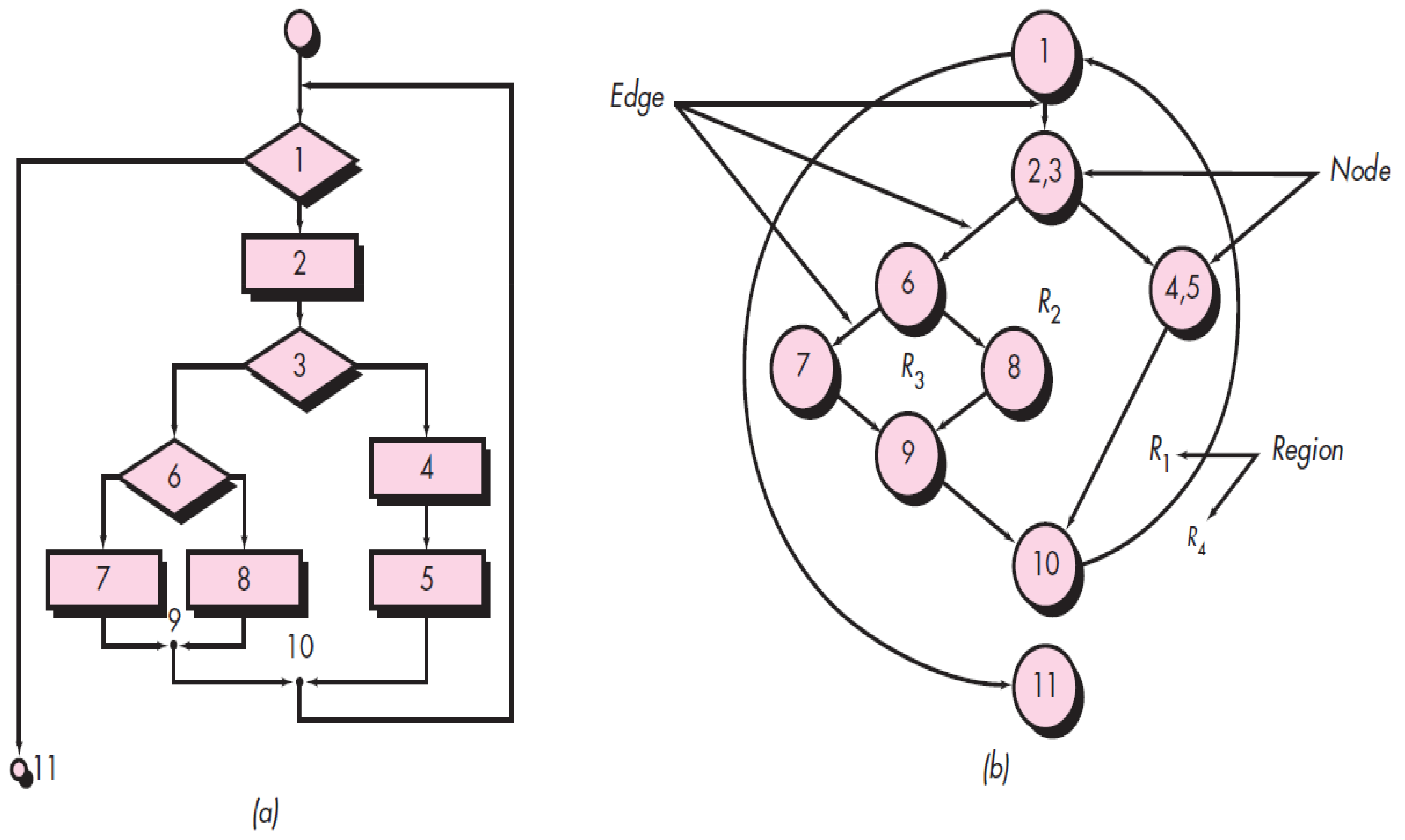

The structured constructs in flow graph form:
Sequence
If
While
Until
Case
Where each circle represents one or more nonbranching PDL or source code statements

Flow graph Notation

- Figure 18.2b maps the flowchart into a corresponding flow graph (assuming that no compound conditions are contained in the decision diamonds of the flowchart).
- Referring to Figure 18.2b, each circle, called a *flow graph node, represents one or more procedural* statements
- The arrows on the flow graph, called *edges or links, represent flow of control* and are analogous to flowchart arrows
- Areas bounded by edges and nodes are called *regions. When counting regions, we include the area outside the graph as a region.*

Flow graph Notation

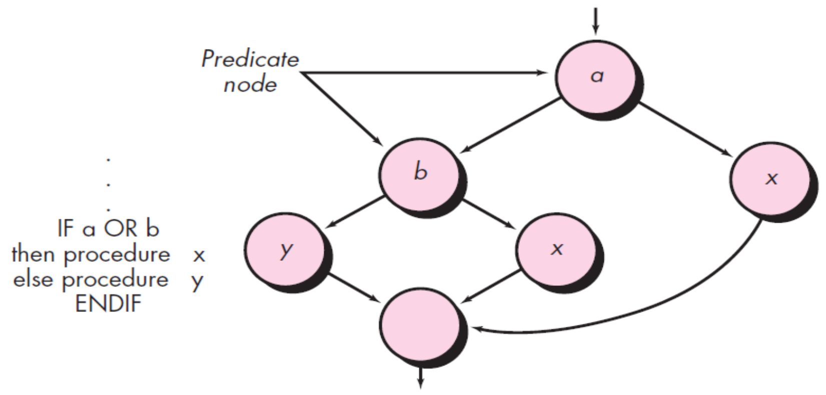
FIGURE 18.2 (a) Flowchart and (b) flow graph



Flow graph Notation

FIGURE 18.3

Compound logic



- A compound condition occurs when one or more Boolean operators (logical OR, AND, NAND, NOR) is present in a conditional statement.
- Note that a separate node is created for each of the conditions *a and b in the statement IF a OR b.*
- *Each node that* contains a condition is called a *predicate node and is characterized by two or more* edges emanating from it.

Independent Program paths

<u>Independent path</u>

- An *independent path is any path through the program that introduces at least one* new set of processing statements or a new condition.
- When stated in terms of a flow graph, an independent path must move along at least one edge that has not been traversed before the path is defined.

<u>Example: set of independent path</u>

- Path 1: 1-11
- Path 2: 1-2-3-4-5-10-1-11
- Path 3: 1-2-3-6-8-9-10-1-11
- Path 4: 1-2-3-6-7-9-10-1-11

<u>Not a basis set: example</u>

1-2-3-4-5-10-1-2-3-6-8-9-10-1-11 (not a independent path , it does not traverse any new edges)

Independent Program paths

<u>Basis set</u>

- Paths 1 through 4 constitute a *basis set for the flow graph in Figure 18.2b. That is,* if you can design tests to force execution of these paths (a basis set), every statement in the program will have been guaranteed to be executed at least one time and every condition will have been executed on its true and false sides

<u>Example: set of independent path</u>

- Path 1: 1-11
- Path 2: 1-2-3-4-5-10-1-11
- Path 3: 1-2-3-6-8-9-10-1-11
- Path 4: 1-2-3-6-7-9-10-1-11

Independent Program paths

Cyclomatic complexity

- *Cyclomatic complexity is a software metric that provides* a quantitative measure of the logical complexity of a program.
- When used in the context of the basis path testing method, the value computed for cyclomatic complexity defines the number of independent paths in the basis set of a program and provides you with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.

Independent Program paths

<u>**How do I calculate Cyclomatic complexity**</u>

- Complexity is computed in one of three ways:

1. The number of regions of the flow graph corresponds to the cyclomatic complexity.

2. Cyclomatic complexity *V(G) for a flow graph G is defined as*

$V(G) = E - N + 2$

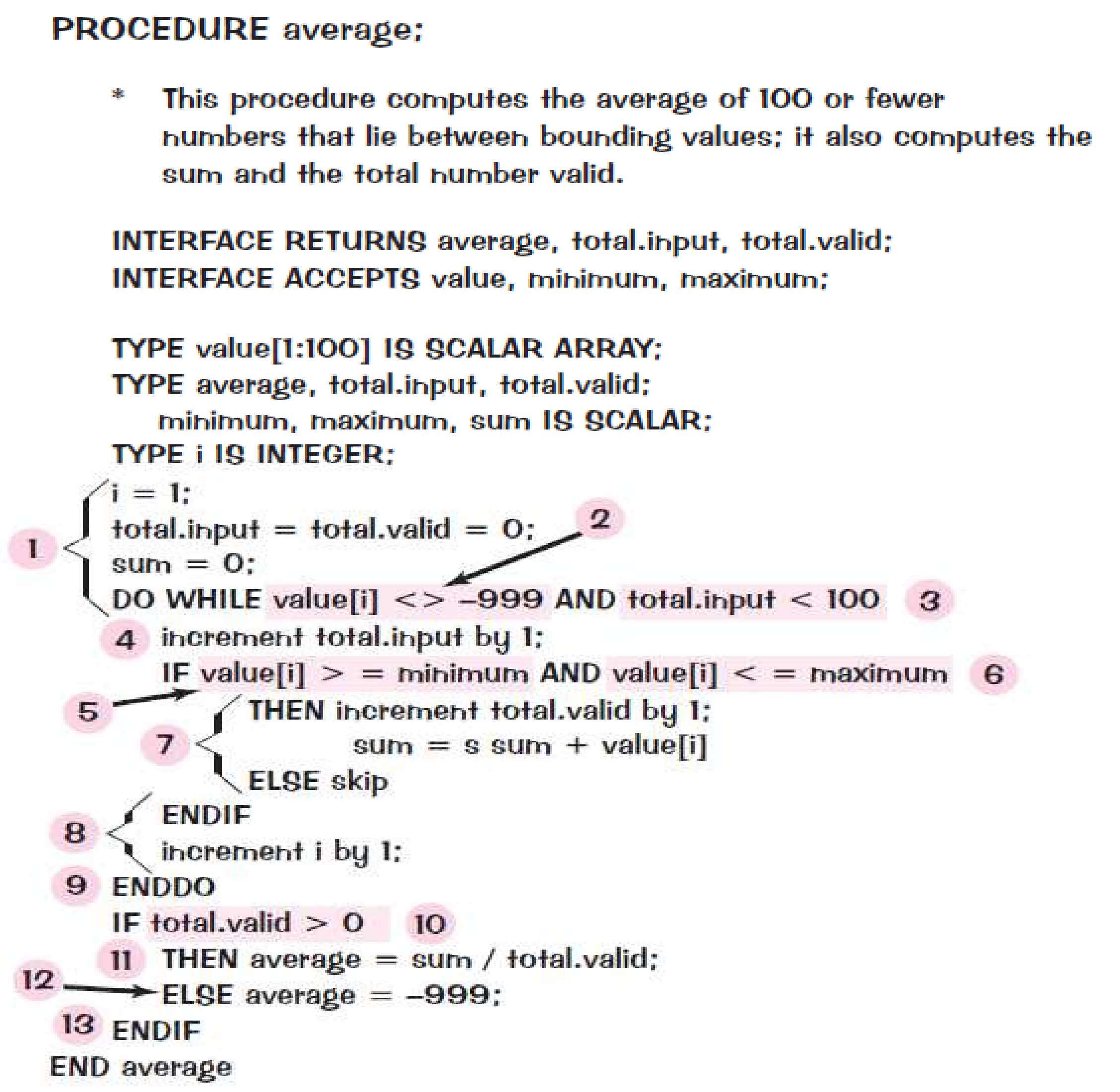
where *E is the number of flow graph edges and N is the number of flow* graph nodes.

3. Cyclomatic complexity *V(G) for a flow graph G is also defined as*

$V(G) = P + 1$

where *P is the number of predicate nodes contained in the flow graph G.*

Deriving Test Cases

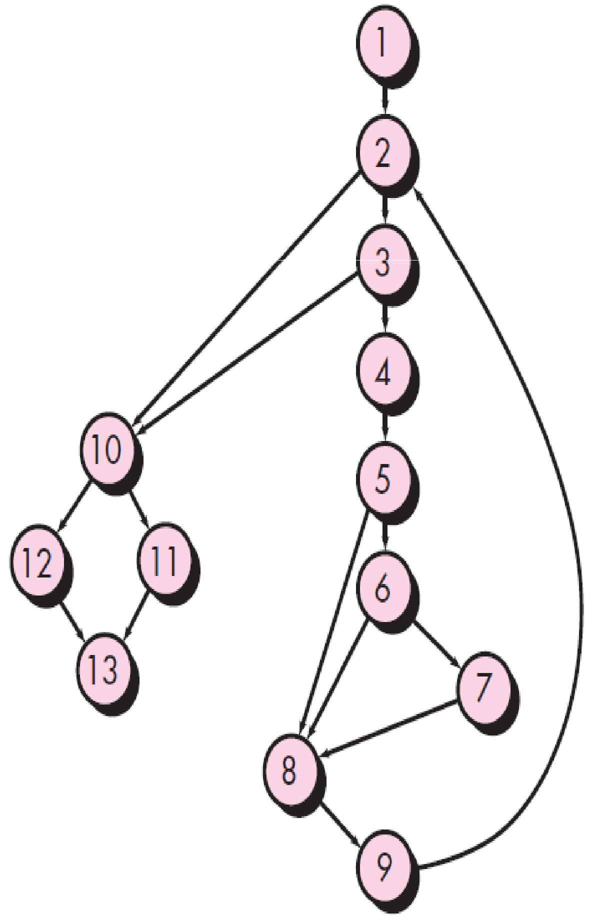


```
PROCEDURE average;

        *   This procedure computes the average of 100 or fewer
            numbers that lie between bounding values; it also computes the
            sum and the total number valid.

        INTERFACE RETURNS average, total.input, total.valid;
        INTERFACE ACCEPTS value, minimum, maximum;

        TYPE value[1:100] IS SCALAR ARRAY;
        TYPE average, total.input, total.valid;
            minimum, maximum, sum IS SCALAR;
        TYPE i IS INTEGER;
        i = 1;
   1    total.input = total.valid = 0;         2
        sum = 0;
        DO WHILE value[i] <> –999 AND total.input < 100     3
    4       increment total.input by 1;
            IF value[i] > = minimum AND value[i] < = maximum     6
    5           THEN increment total.valid by 1;
        7                sum = s sum + value[i]
                ELSE skip
    8       ENDIF
            increment i by 1;
    9   ENDDO
        IF total.valid > 0     10
    11      THEN average = sum / total.valid;
   12       ELSE average = –999;
    13  ENDIF
END average
```

Deriving Test Cases

FIGURE 18.5

Flow graph for the procedure *average*



Deriving Test Cases

- The following steps can be applied to derive the basis set:

1. **Using the design or code as a foundation, draw a corresponding flow graph.**
2. **Determine the cyclomatic complexity of the resultant flow graph.**

- *V(G) = 6 regions*
- *V(G) =17 edges - 13 nodes + 2 = 6*
- *V(G) =5 predicate nodes+ 1 = 6*

Deriving Test Cases

3. Determine a basis set of linearly independent paths.

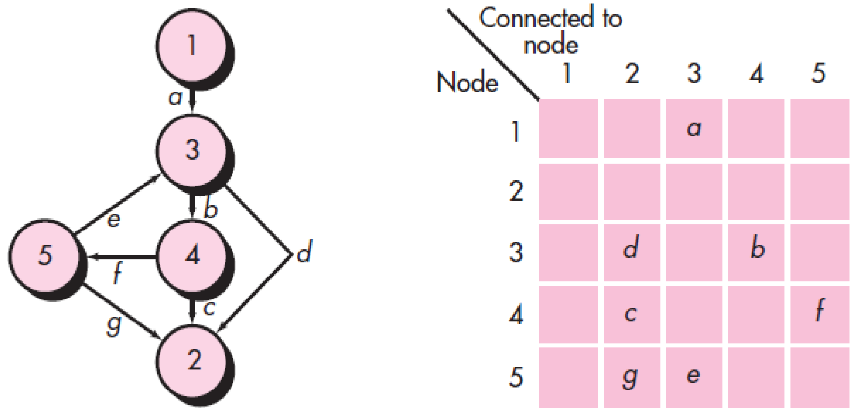
- Path 1: 1-2-10-11-13
- Path 2: 1-2-10-12-13
- Path 3: 1-2-3-10-11-13
- Path 4: 1-2-3-4-5-8-9-2-. . .
- Path 5: 1-2-3-4-5-6-8-9-2-. . .
- Path 6: 1-2-3-4-5-6-7-8-9-2-. . .

4. Prepare test cases that will force execution of each path in the basis set.

Graph Matrices

- A data structure, called a *graph matrix, can be quite* useful for developing a software tool that assists in basis path testing.
- A graph matrix is a square matrix whose size (i.e., number of rows and columns) is equal to the number of nodes on the flow graph. Each row and column corresponds to an identified node, and matrix entries correspond to connections (an edge) between nodes.

FIGURE 18.6
Graph matrix



Node \ Connected to node	1	2	3	4	5
1			*a*		
2					
3		*d*		*b*	
4		*c*			*f*
5		*g*	*e*		

Flow graph Graph matrix