Lecture 7.1

Internal and External Views of Testing

INTRODUCTION

- Any engineered product (and most other things) can be tested in one of two ways:

✓Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function. → ***Black-Box Testing***

✓Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh," that is, internal operations are performed according to specifications and all internal components have been adequately exercised. → ***White-Box Testing***

BASIS PATH TESTING

- Basis path testing is a white-box testing technique first proposed by Tom McCabe.

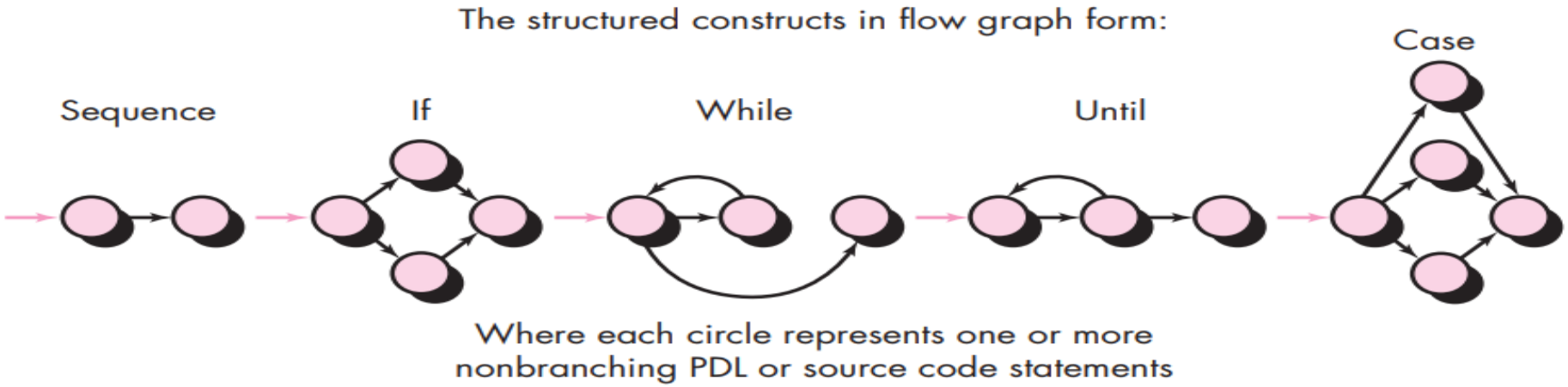
- The basis path method enables the test-case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths.

- Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

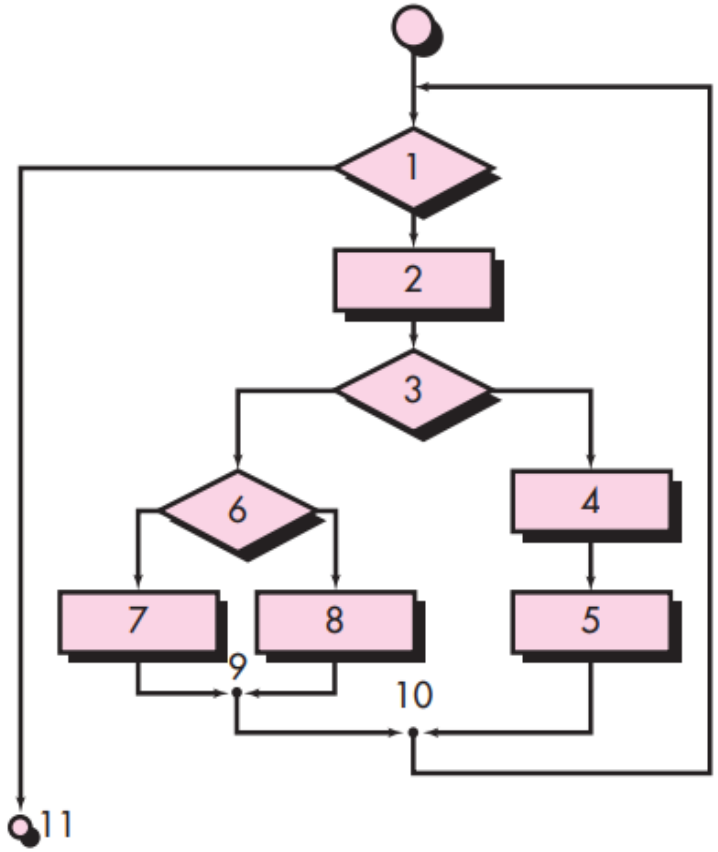
FLOW GRAPH NOTATION

- The flow graph depicts logical control flow using the notation. Each structured construct has a corresponding flow graph symbol.

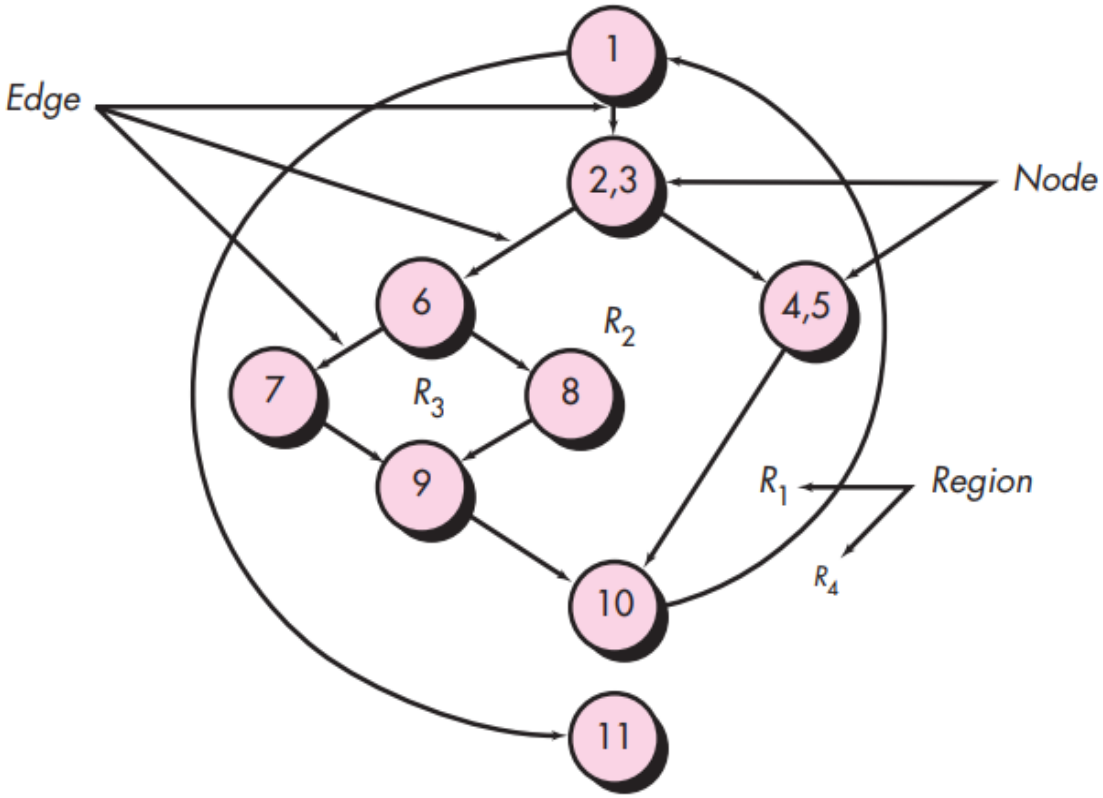
- The basis path method can be conducted without the use of flow graphs. However, they serve as a useful notation for understanding control flow and illustrating the approach.



(a) FLOWCHART and (b) FLOW GRAPH



(a)



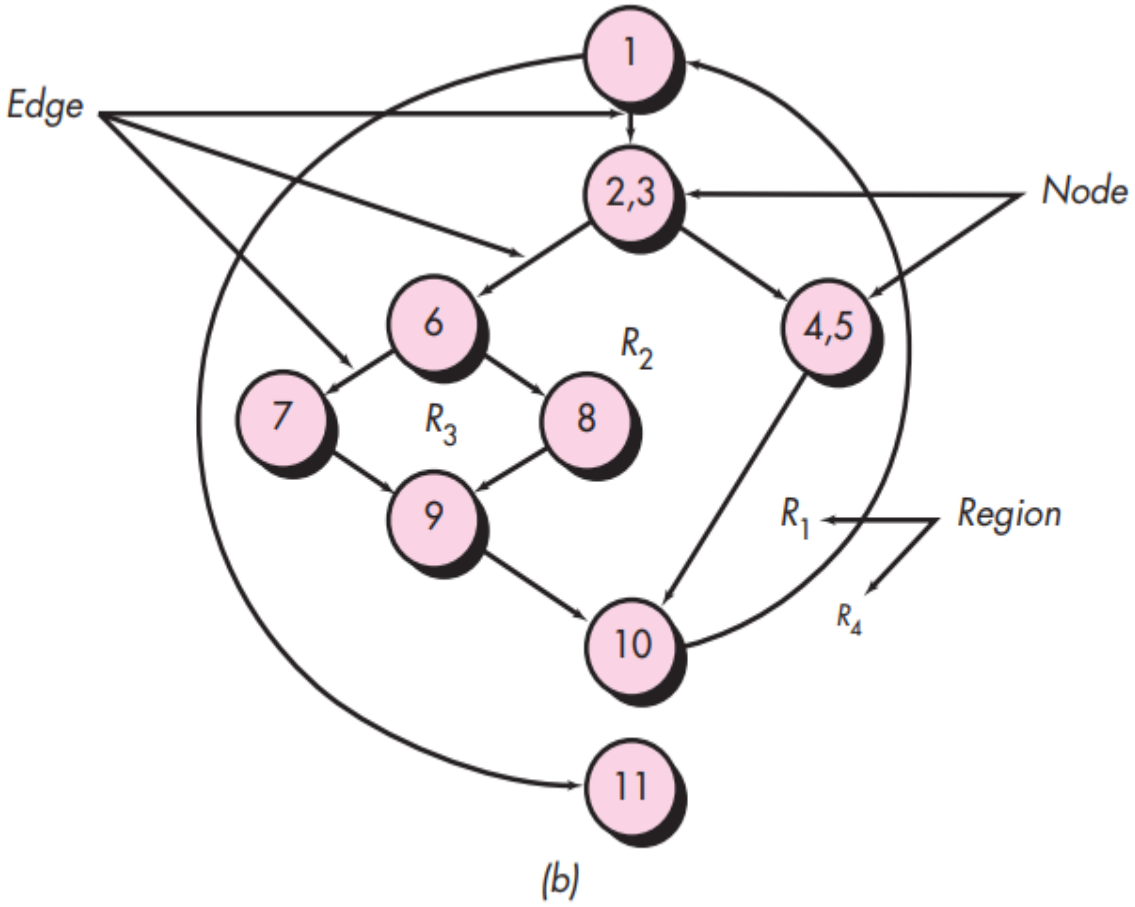
(b)

INDEPENDENT PROGRAM PATHS

- 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.

INDEPENDENT PROGRAM PATHS



❑A set of independent paths for the flow graph illustrated in Figure (b)

- 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

- Note that each new path introduces a new edge. The path
- 1-2-3-4-5-10-1-2-3-6-8-9-10-1-11
- is not considered to be an independent path because it is simply a combination of already specified paths and does not traverse any new edges.

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.

- Cyclomatic complexity has a foundation in graph theory and provides you with an extremely useful software metric.

CYCLOMATIC COMPLEXITY

- 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

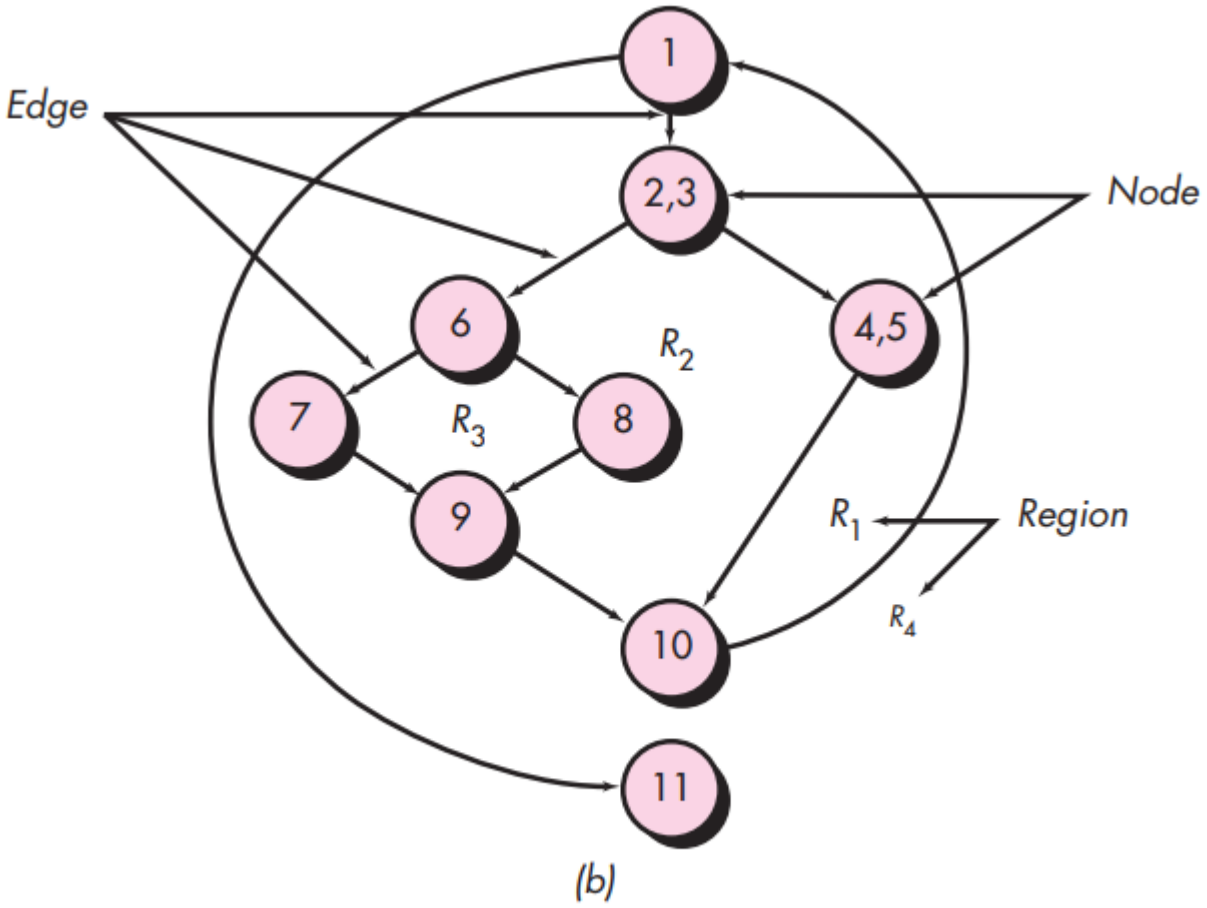
where E is the number of flow graph edges and N is the number of flow graph nodes.

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.

CYCLOMATIC COMPLEXITY



- The cyclomatic complexity can be computed for Figure (b) using each of the algorithms just noted:

  - 1. The flow graph has four regions.
  - 2. V(G) = 11 edges – 9 nodes + 2 = 4
  - 3. V(G) = 3 predicates nodes + 1 = 4

- Therefore, the cyclomatic complexity of the flow graph in Figure (b) is 4

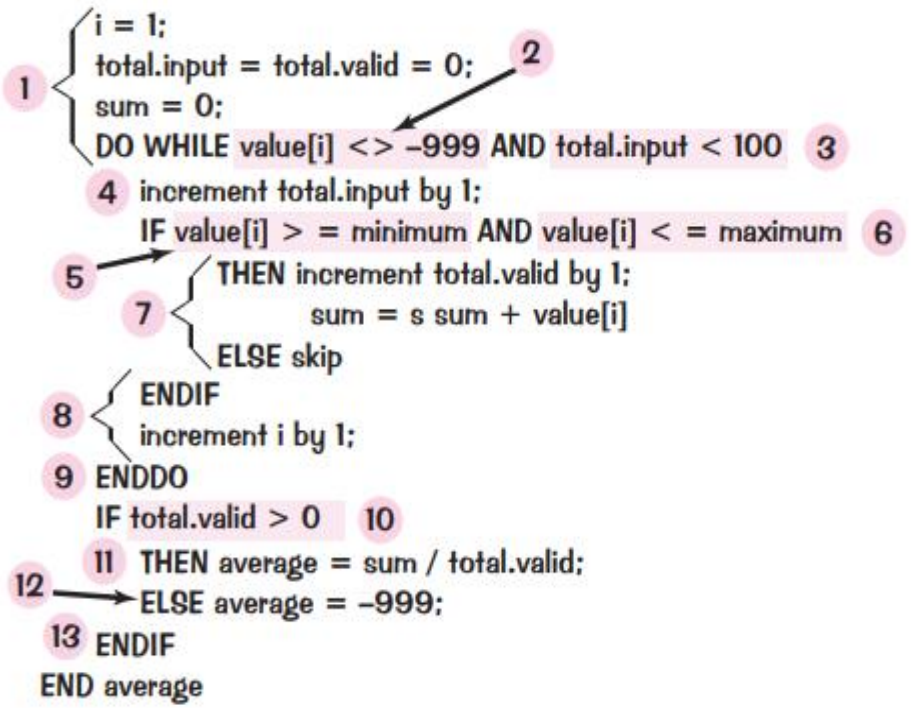
CYCLOMATIC COMPLEXITY

- The basis path testing method can be applied to a procedural design or to source code.

- The procedure average, depicted in the following segment, will be used as an example to illustrate each step in the test-case design method.

- Note that in the following segment, although an extremely simple algorithm, contains compound conditions and loops.

CYCLOMATIC COMPLEXITY

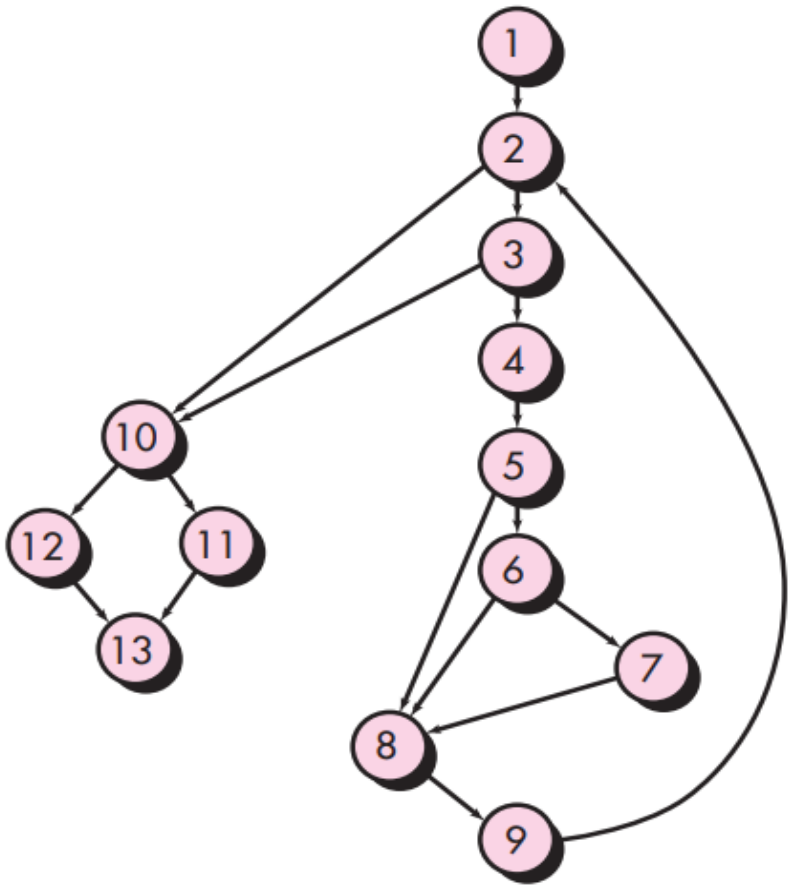


Algorithmic Segment

- Determine the cyclomatic complexity of the algorithmic segment:

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

CYCLOMATIC COMPLEXITY



Flow Graph for the algorithmic segment

- All possible independent paths for the following algorithmic segment after drawing the flow graph:

- we expect to specify six 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-. . .

- The ellipsis (. . .) following paths 4, 5, and 6 indicates that any path through the remainder of the control structure is acceptable. Here, nodes 2, 3, 5, 6, and 10 are predicate nodes

The End