Topic #2 – Software Basis Path Testing

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

TOPIC #2 – SOFTWARE BASIS PATH TESTING

- Flow Graph Model for White-Box Testing
- Cyclomatic Complexity
- Basis Path Testing Method
- Basis Path Testing Tips
- Basis Path Testing Coverage

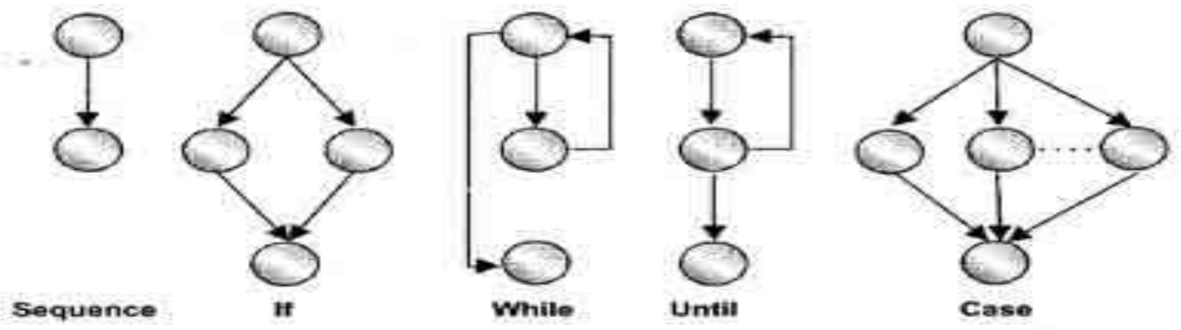
What is a program flow graph?

Definition:

--> A program flow is a graph model which is useful to present the control flows for a program. Each program flow graph consists of a set of nodes and edges (or links).

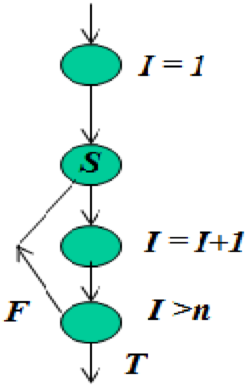
A program flow graph can be used as a test model for white-box program testing.

Typical example of program control structures



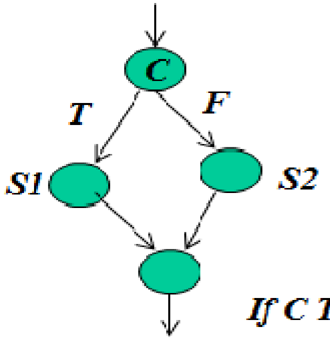
Flow Graph Notation

Typical Control Structures

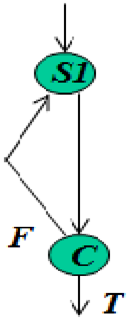


For loop:

for I = 1 to n do S;

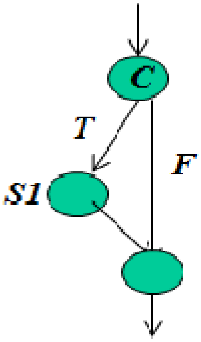


If C Then S1 else S2;



Do loop:

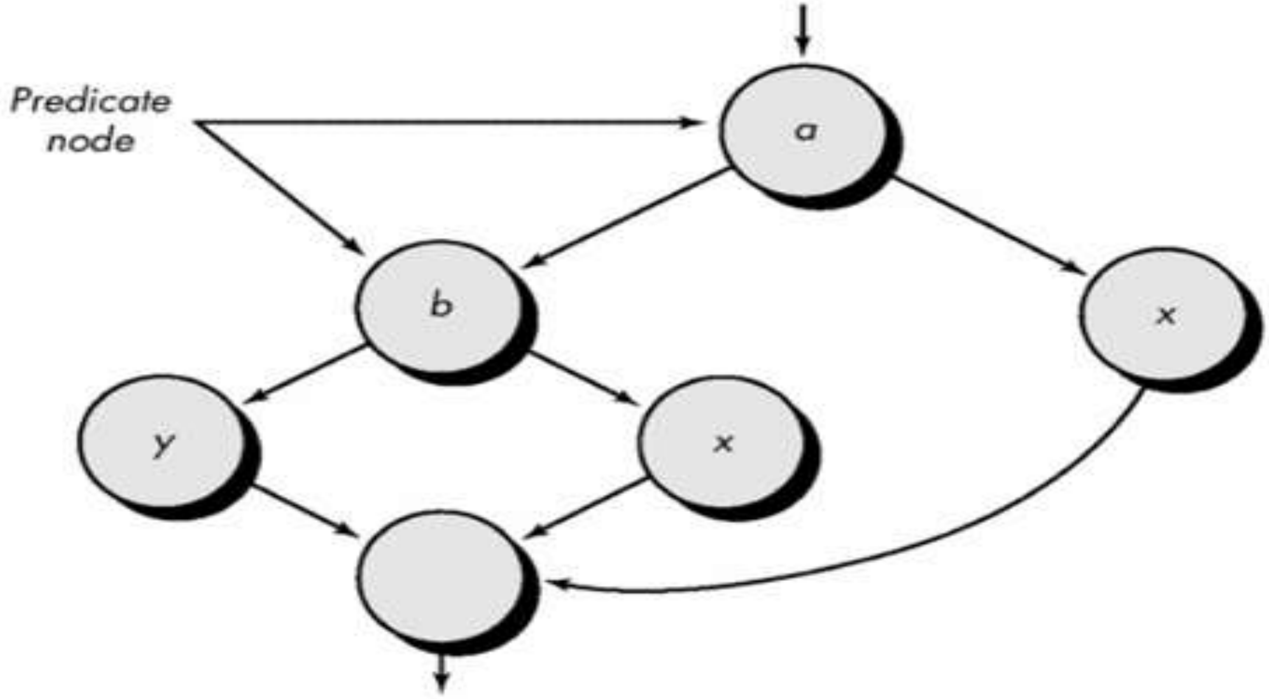
do S1 until C;



If C Then S1;

Program Flow Graph Example

```
.
.
.
IF a OR b
then procedure   x
else procedure   y
ENDIF
```



Cyclomatic Complexity

What is Cyclomatic complexity?

Cyclomatic complexity is a software metric (developed by Thomas J. McCabe, Sr. in 1976) It is used to indicate the complexity of a program.

It is a quantitative measure of the complexity of programming instructions. It directly measures the number of linearly independent paths through a program's source code.

Cyclomatic complexity is computed using the control flow graph of a program.

One testing method, called Basis path testing (proposed by McCabe).

It is useful to test each linearly independent path through the program.

Cyclomatic Complexity

Cyclomatic complexity is computed using the control flow graph of a program.

Let M(G) represents the cyclomatic complexity of a program flow graph G.
N stands for the node set in G, and E stands for the edge set in G.
P stands for the predicate node set in G.

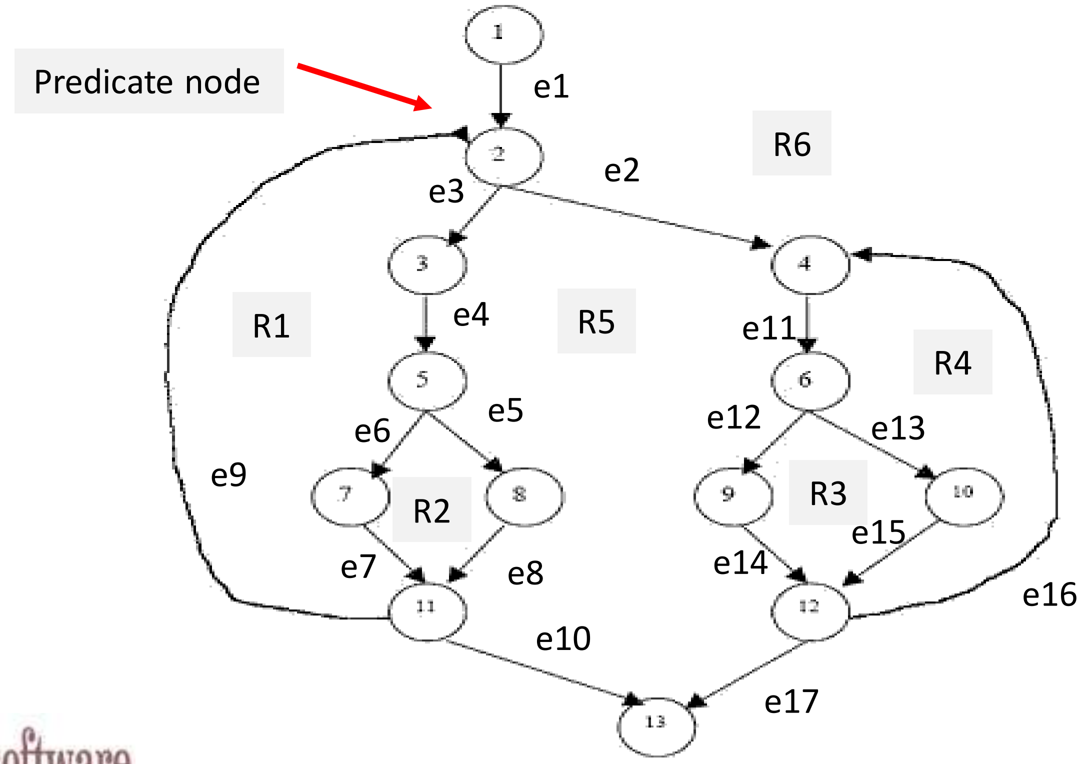
Three ways to compute cyclomatic complexity for a program:

#1: M(G) = No. of regions in G

#2: $M(G) = |E| - |N| +2$
Where |E| is the number of edges and |N| is the number of nodes.

#3: $M(G) = |P| + 1$
Where |P| is the number of predicate nodes in the flow graph G.

Cyclomatic Complexity Computation

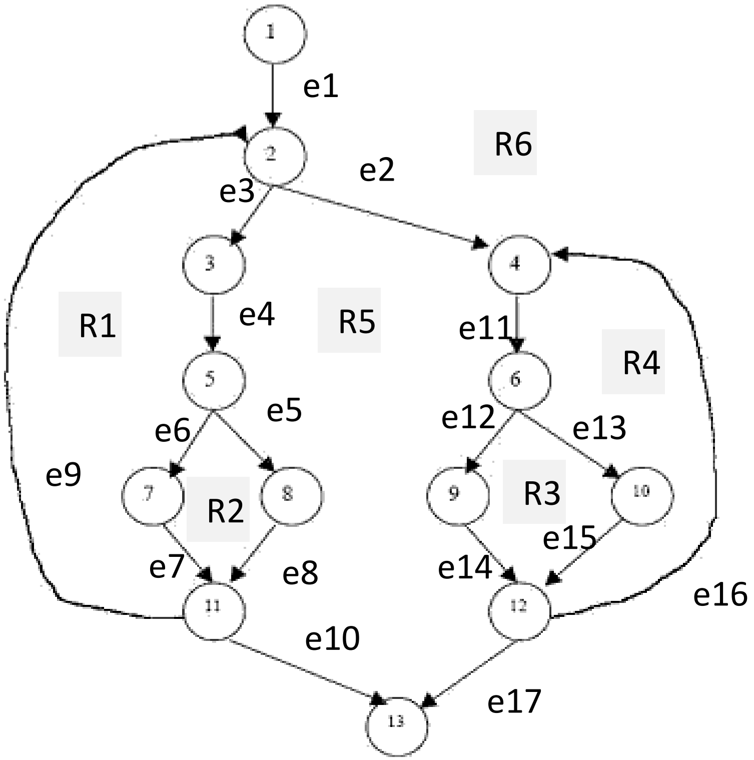

Predicate node

M(G) = 6 regions

M(G) = |E| - |N| +2
= 17 – 13 + 2 = 6

M(G) = |P| + 1
= 5 + 1 =6

Basis Path Testing Method



Step 1 : Draw a corresponding flow graph based on program codes

Step 2: Compute the cyclomatic complexity

Step 3: Determine a minimum basis set of linearly independent paths.

For example,

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

Step 4: Prepare a test case for each path in the set.

Step 5: Run the test cases and check their results

Basis Path Testing Tips

Simple tips to form your basis path set:

1. Add your basis path incrementally. (one by one)
2. Check the redundant path whenever you add one path,
3. Make sure that new path has at least one new node or new link comparing with the rest paths in the set.
4. Make sure the total no. of basis paths in the set is equal to your cyclomatic complexity.

Simple tips to form your basis path test set:

1. Make sure each basis path is executable based on inputs.
2. Make sure to find the expected outputs based on your inputs for each test case.

Compute Cyclomatic Complexity Using Graph Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1-1=0
2	0	0	1	1	0	0	0	0	0	0	0	0	0	2-1=1
3	0	0	0	0	1	0	0	0	0	0	0	0	0	1-1=0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	1-1=0
5	0	0	0	0	0	0	1	1	0	0	0	0	0	2-1=1
6	0	0	0	0	0	0	0	0	1	1	0	0	0	2-1=1
7	0	0	0	0	0	0	0	0	0	0	1	0	0	1-1=0
8	0	0	0	0	0	0	0	0	0	0	1	0	0	1-1=0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	1-1=0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	1-1=0
11	0	1	0	0	0	0	0	0	0	0	0	0	1	2-1=1
12	0	0	0	1	0	0	0	0	0	0	0	0	1	2-1=1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	

|P|=5

|P| + 1 = 6

Basis Path Testing Coverage

Following the basis path testing method, we can achieve the following White-box program test coverage for each program.

1. **Source code node coverage:**

- For each node in a program flow graph, there will be at least one basis path test case exercise it.

2. ***Control link coverage***:

- For each edge in a program flow graph, there will be at least one basis path test case cover it.

3. ***Basis path coverage***:

- For each basis path in the basis path set, there will be at least one basis test case covering it.

4. Predicate node coverage:

- For each node, there will be at least one basis path test case covering it.