

# Cyclomatic Complexity

# Cyclomatic Complexity

- ▶ Invented by **Thomas McCabe (1974)** to measure the complexity of a program's conditional logic
- ▶ Cyclomatic complexity of graph G equals #edges - #nodes + 2

$$V(G) = e - n + 2$$

- ▶ Also corresponds to the number of linearly independent paths in a program

# Converting Code to Graph

	CODE	FLOWCHART	GRAPH
(a)	<pre> if expression1 then     statement2 else     statement3 end if statement4 </pre>	<pre> graph TD     D{expr1 ?} -- T --&gt; R1[stmtm2]     R1 --&gt; R4[stmtm4]     D -- F --&gt; R2[stmtm3]     R2 --&gt; R4 </pre>	<pre> graph TD     n1((n1)) --&gt; n2((n2))     n1 --&gt; n3((n3))     n1 --&gt; n4((n4))     n2 --&gt; n3     n3 --&gt; n4     n2 --&gt; n4     n3 --&gt; n4 </pre>
(b)	<pre> switch expr1 case 1:     statement2 case 2:     statm3 case 3:     statm4 end switch statm5 </pre>	<pre> graph TD     D{expr1 ?} -- 1 --&gt; R1[stmtm2]     D -- 2 --&gt; R2[stmtm3]     D -- 3 --&gt; R3[stmtm4]     R1 --&gt; R5[stmtm5]     R2 --&gt; R5     R3 --&gt; R5 </pre>	<pre> graph TD     n1((n1)) --&gt; n2((n2))     n1 --&gt; n3((n3))     n1 --&gt; n4((n4))     n1 --&gt; n5((n5))     n2 --&gt; n3     n3 --&gt; n4     n4 --&gt; n5     n2 --&gt; n5     n3 --&gt; n5     n4 --&gt; n5 </pre>
(c)	<pre> do     statement1 while expr2 end do statement3 </pre>	<pre> graph TD     R1[stmtm1] --&gt; D{expr2 ?}     D -- T --&gt; R1     D -- F --&gt; R3[stmtm3] </pre>	<pre> graph TD     n1((n1)) --&gt; n2((n2))     n2 --&gt; n2     n2 --&gt; n3((n3))     n3 --&gt; n4((n4)) </pre>

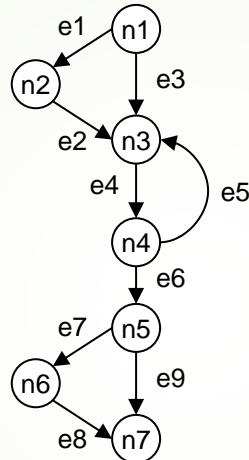
# Example Paths

```

if expression1
then
    statement2
end if

do
    statement3
    while expr4
end do

if expression5
then
    statement6
end if
statement7
  
```



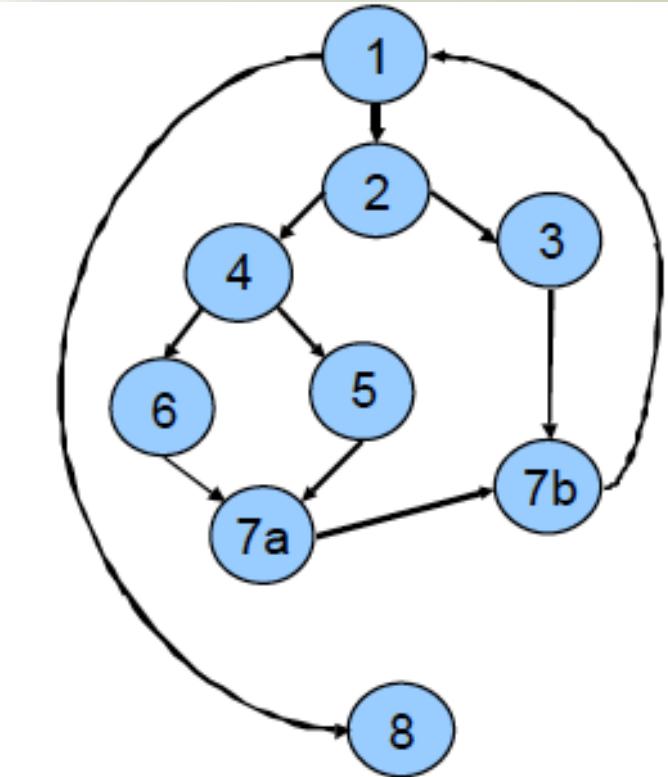
Paths:

- 
- P1 = e1, e2, e4, e6, e7, e8
  - P2 = e1, e2, e4, e5, e4, e6, e7, e8
  - P3 = e3, e4, e6, e7, e8, e10
  - P4 = e6, e7, e8, e10, e3, e4
  - P5 = e1, e2, e4, e6, e9, e10
  - P6 = e4, e5
  - P7 = e3, e4, e6, e9, e10
  - P8 = e1, e2, e4, e5, e4, e6, e9, e10

$$V(G) = e - n + 2 = 9 - 7 + 2 = 4$$

# Example 1

```
1. do while records remain  
    read record;  
2. if record field 1 = 0  
3.     then process record;  
        store in buffer;  
        increment counter;  
4. elseif record field 2 = 0  
5.     then reset record;  
6.     else process record;  
        store in file;  
7a.    endif;  
    endif;  
7b.    enddo;  
8. end;
```



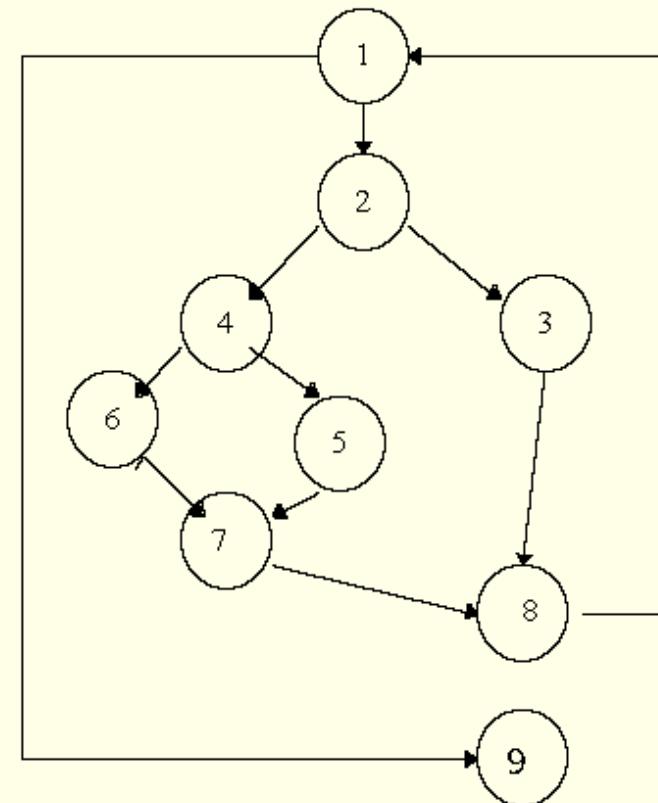
$$V = e - n + 2 = 11 - 9 + 2 = 4$$

# Example 2

```

1: WHILE NOT EOF LOOP
2:   Read Record;
3:   IF field1 equals 0 THEN
4:     Add field1 to Total
5:     Increment Counter
6:   ELSE
7:     IF field2 equals 0 THEN
8:       Print Total, Counter
9:       Reset Counter
10:    ELSE
11:      Subtract field2 from Total
12:    END IF
13:  END IF
14:  Print "End Record"
15: END LOOP
16: Print Counter

```



$$V = E - n + 2 = 11 - 9 + 2 = 4$$