SLIIT

Discover Your Future

Software Metrics

Software Quality

- How can software quality be measured?
- Software quality metrics:
  - Code Quality
  - Reliability
  - Performance
  - Usability
  - Correctness
  - Maintainability
  - Integrity
  - Security

Cyclomatic Complexity (CC)

- Measures the number of linearly **independent paths** in a program.

V(G) = d + 1

V(G) = e – n + 2

Vg = No of decision statements in each method of the class + No of methods in the class

Approaches to Measure Cyclomatic Complexity

- Explain some approaches that can be used to measure the Cyclomatic Complexity of a program?

Output of RSM

```
ark:bvk>rsm -Tf Results.java

Function: Results.OutResults
Complexity      Param 1      Return 4      Cyclo Vg 4      Total      9
LOC 10          eLOC 9       lLOC 4        Comment 0       Lines      10
Function Points              FP(LOC) 0.2   FP(eLOC) 0.2    FP(lLOC)   0.1

                    ~~ Project Functional Analysis ~~

Total Functions .....:    1     Total Physical Lines .....:    10
Total LOC .....:          10    Total Function Pts LOC.....:   0.2
Total eLOC .....:         9     Total Function Pts eLOC .....: 0.2
Total lLOC.....:          4     Total Function Pts lLOC.....:  0.1
Total Cyclomatic Comp.....:   4     Total Interface Comp .....:    5
Total Parameters .....:       1     Total Return Points .....:     4
Total Comment Lines .....:    0     Total Blank Lines .....:       0
```

Calculating Cyclomatic Complexity of a Byte Code

- Calculate the cyclomatic complexity of the following decompiled byte code:

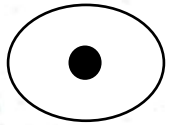
```
Method void D0(boolean, java.lang.String)
   0 iload_0
   1 ifeq 11
   4 getstatic #10 <Field java.io.PrintStream out>
   7 aload_1
   8 invokevirtual #16 <Method void println(java.lang.String)>
  11 return
```

Calculating Cyclomatic Complexity of a Byte Code

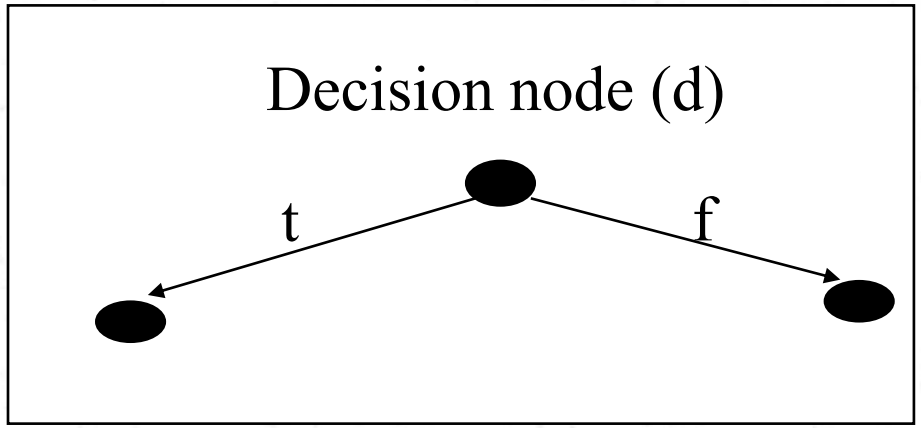
- Calculate the cyclomatic complexity of the following decompiled byte code?

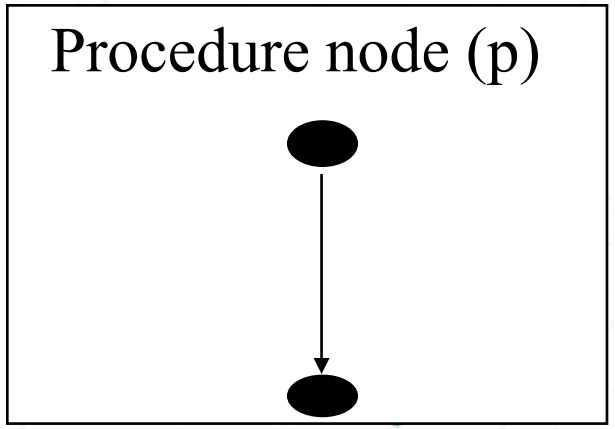
```
Method void D1(boolean, java.lang.String, java.lang.String)
 0  iload_0
 1  ifeq 14
 4  getstatic #2 <Field java.io.PrintStream out>
 7  aload_1
 8  invokevirtual #3 <Method void println(java.lang.String)>
11 goto 21
14 getstatic #2 <Field java.io.PrintStream out>
17 aload_2
18 invokevirtual #3 <Method void println(java.lang.String)>
21 return
```

Control Flow Graph Rules

1. To represent a start or a stop node use the notation 
2. To represent an intermediary node use the notation ●
3. Start node, stop node, decision nodes, and true/false paths have to be labeled.
4. Edges should always indicate the directions.

Intermediary Nodes in a Control Flow graph

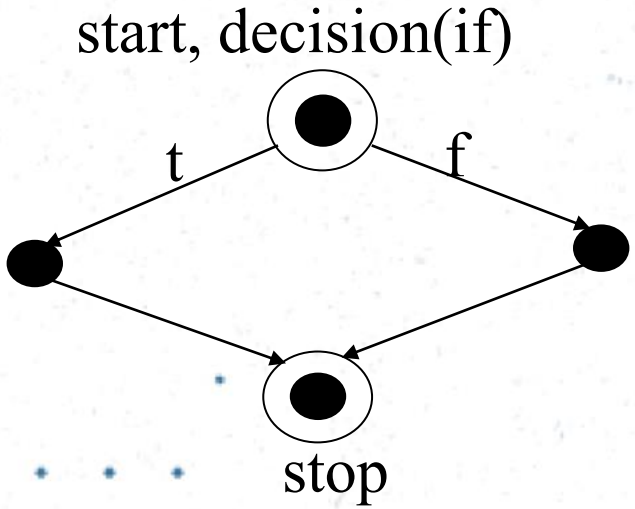




Control Flow Graph Rules – Rule Five

5. Along with a start node, procedure and decision nodes can also be represented.

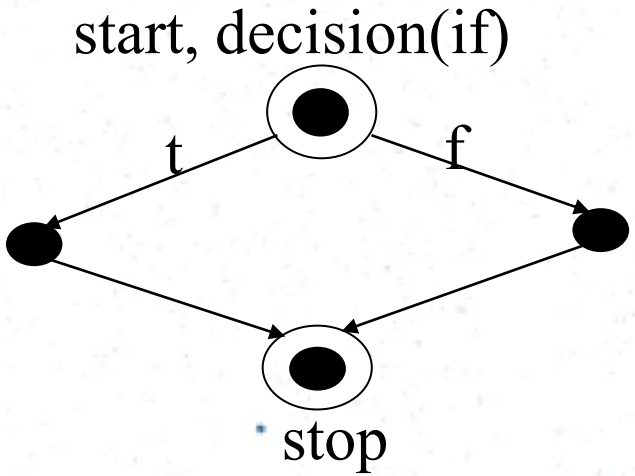
```
int p;
if(p <10)
    System.out.println("Value of p is less than 10");
else
    System.out.println("Value of p is a grater than or equal to 10");
```



Control Flow Graph Rules – Rule Six

6. A procedure node represent one or more non-decisional statements.

```
int p;
if(p <10)
    System.out.println("Current value of p is " + p);
    p = p + 1;
else
    System.out.println("Value of p is a grater than or equal to 10");
```

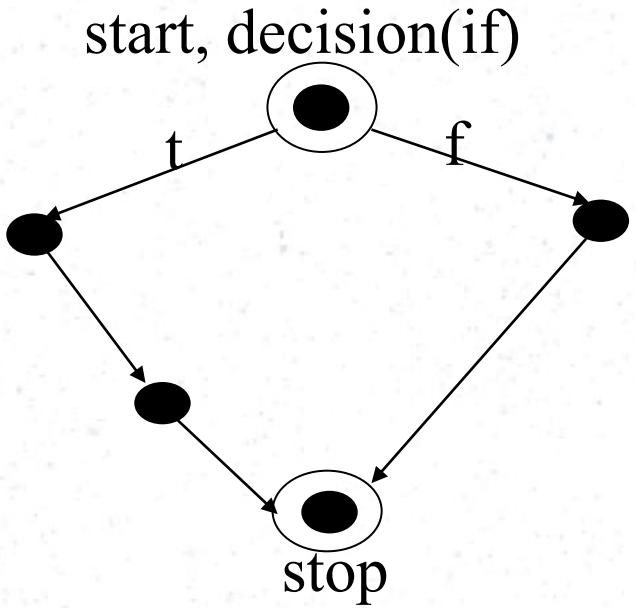


V(G) = e – n + 2

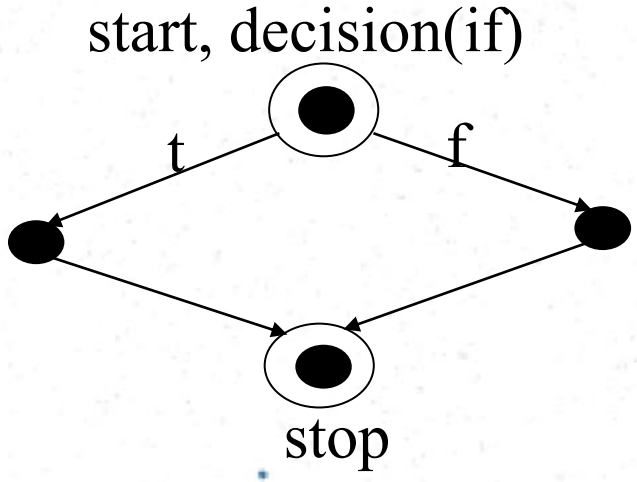
= 4 – 4 + 2

= 2

Do procedure nodes have an impact on the CC value??



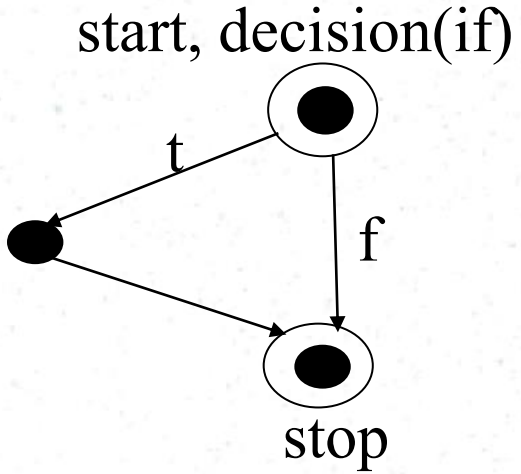
$$V(G) = e - n + 2$$
$$= 5 - 5 + 2$$
$$= 2$$



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

If-then Implementation

1)	public static void D0 (boolean a, String x){
	if(a)
	System.out.println("x");
	}



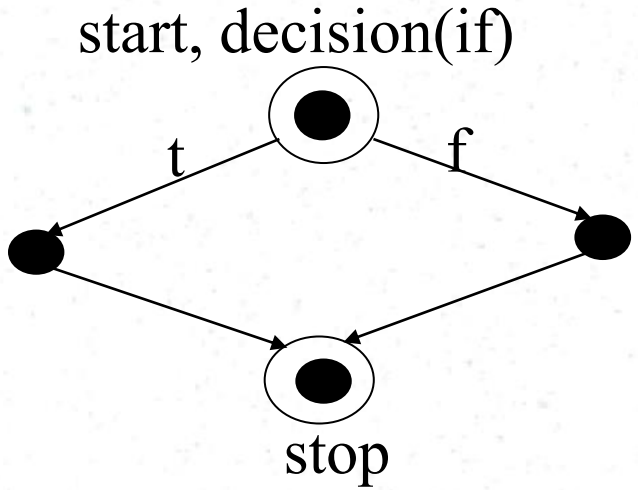
V(G) = e – n + 2

= 3 – 3 + 2

= 2

If-then-else Implementation

2)	public static void D1 (boolean a, String x, String y) {
	if(a)
	System.out.println("x");
	else
	System.out.println("y");
	}



V(G) = e – n + 2

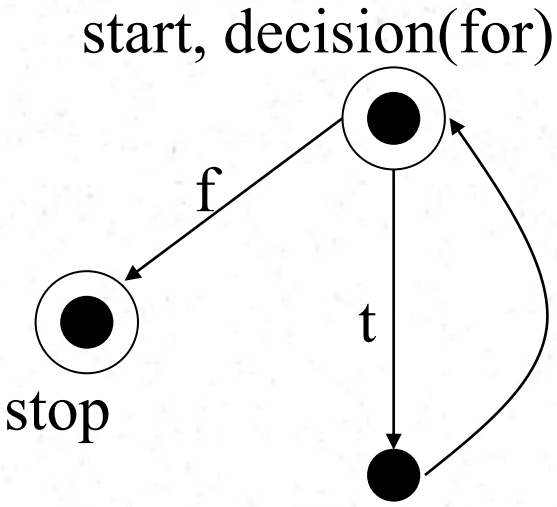
= 4 – 4 + 2

= 2

For Implementation

3)

```
public static void D3(int m, String x) {
for(int i=0; i<m; i++)
    System.out.println("x");
}
```



V(G) = e – n + 2

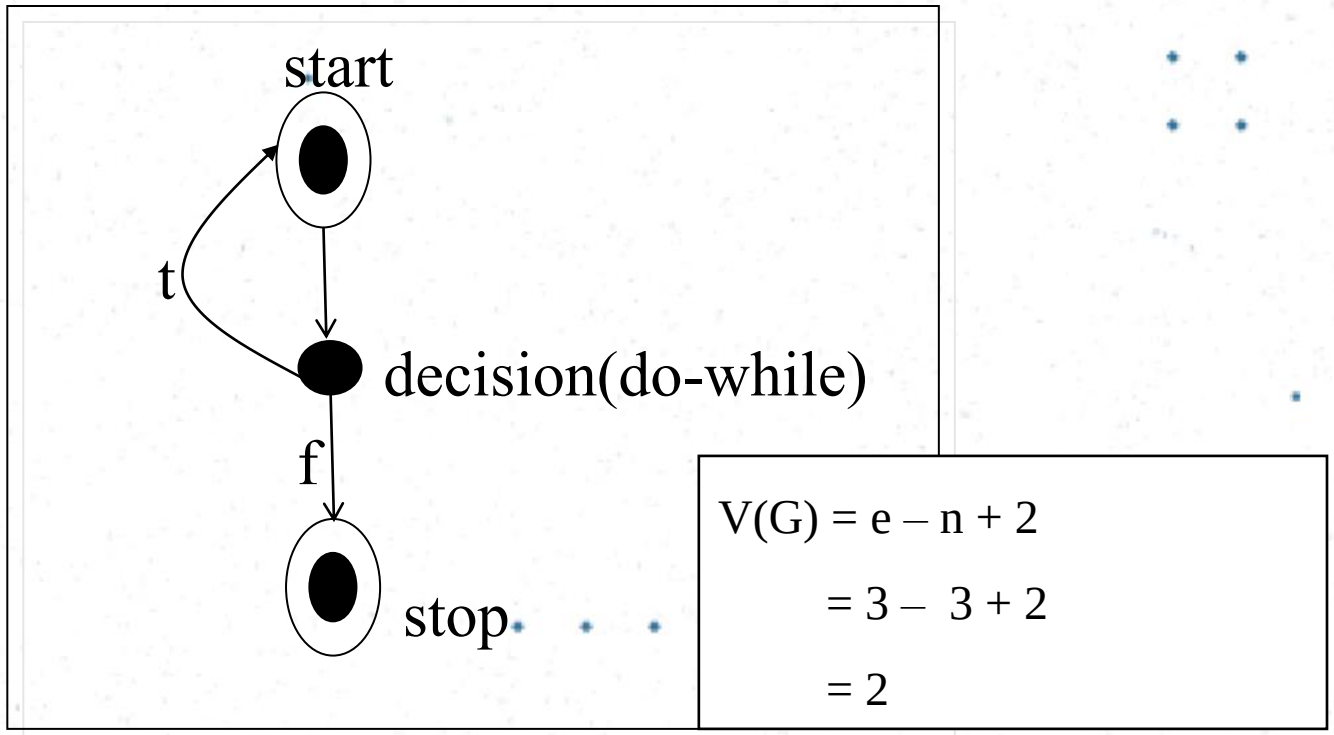
= 3 – 3 + 2

= 2

Do-while Implementation

4)

```
public static void D3(int a, String x) {
do
  {
        System.out.println("x");
        a++;
   } while (a <10)
}
```

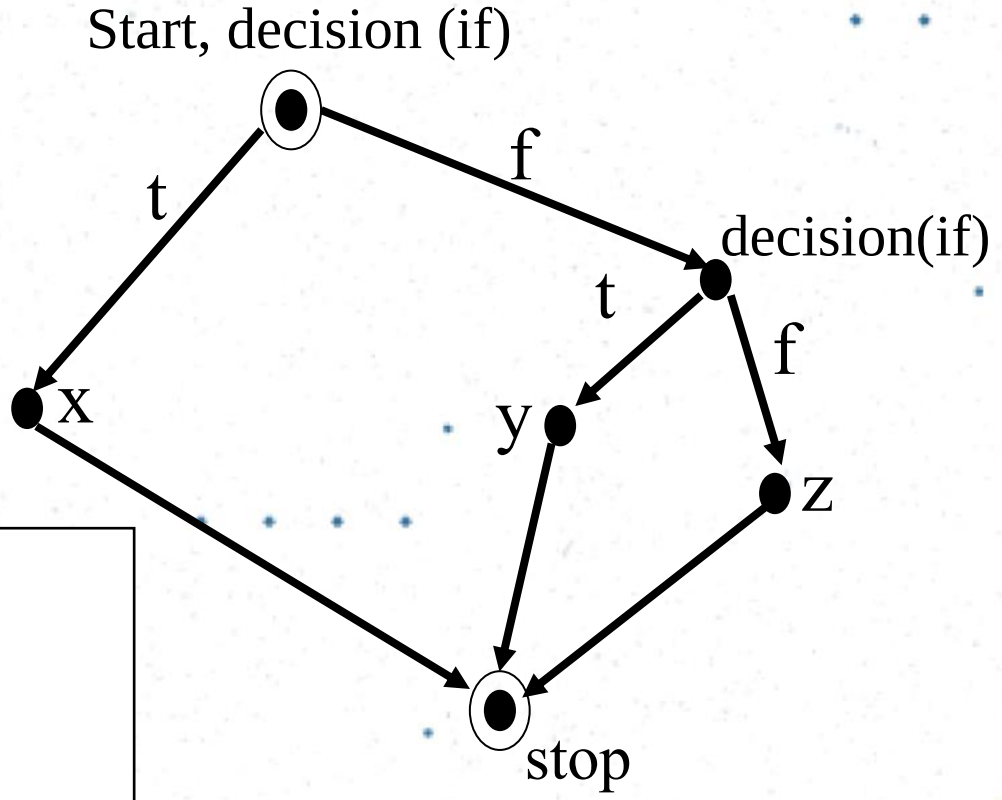


Question & Answer

- Draw the control flow graph for the following code segment and calculate the cyclomatic complexity.

5)

```
void composite (boolean a, boolean b, String x, String y, String z)
{
    if (a)
      System.out.println(x);
    else {
      if (b)
        System.out.println(y);
      else
        System.out.println(z);
      }
}
```



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

$= 7 - 6 + 2$

$= 3$

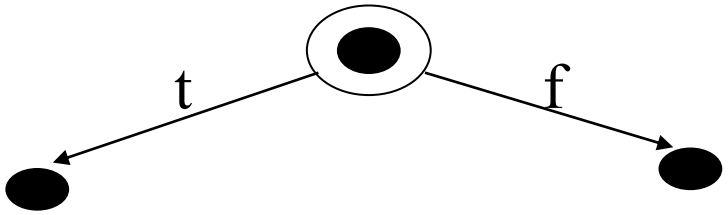
Question??

- **From the V(G) = e – n + 2 equation, drive that V(G) = d + 1.**

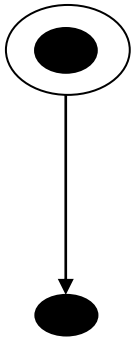
- Nodes in a control flow graph:
  - Decision nodes (d)
  - Procedure nodes (p)
  - Start node
  - Stop node

Total number of nodes in a control flow graph = d + p + 1

Decision node (d)



Procedure node (p)



Total edges in a control flow graph = 2d + 1p

Derivation

e = p + 2d

Where

e = number of edges

p = procedure nodes

d = decision nodes

n = d + p + 1

Where

n = number of nodes

$$
\begin{aligned}
V(G) &= e - n + 2 \\
&= (p + 2d) - (d + p + 1) + 2 \\
&= d + 1
\end{aligned}
$$

Cyclomatic Complexity of a Class

Total Cyclomatic Complexity for a class (V_g)	=	Sum of the cyclomatic complexity of each method

$$V_g = \sum_{i=1}^{n} V(G_i)$$

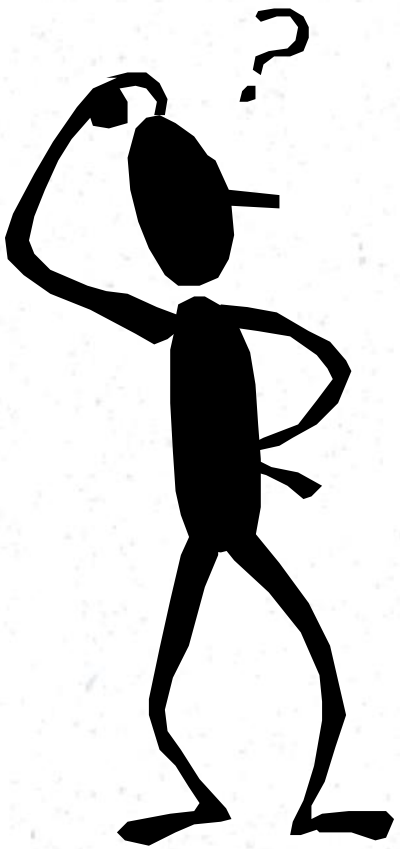
$$V_g = \sum_{i=1}^{n} (d_i + 1)$$

$$V_g = n + d_i$$

Where:

- n = Number of methods in the class
- G_i = Flow graph for method **i**
- d_i = Number of decisions in method **i**

Programs with Compound Statements



- You can't expect the same cyclomatic complexity from all the approaches.
- The CC value obtained from the class file can be higher than CC obtained from the source file.

Question??

- Draw the control flow graph for the following code segment and calculate the cyclomatic complexity.

6)

```
public static void main (String[ ] args) {
    int i = 0;
    switch (i) {
            case 1: System.out.println("its 1");
                    break;
            case 2: System.out.println("its 2");
                    break;
            case 3: System.out.println("its 3");
                    break;
            default: System.out.println("its none");
                    break;
      }
  }
```