Graph Coverage for Source Code

Idaho State University | Computer Science

Isaac Griffith

CS 4422 and CS 5522
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

At the end of Today's Lecture you will be able to:

- Develop CFGs for given source code
- Connect concepts from graph coverage to CFGs

ROAR

Inspiration

"You must be a constructive schizophrenic. Be clear about the difference between your role as a programmer and as a tester. The tester in you must be suspicious, uncompromising, hostile, and compulsively obsessed with destroying, utterly destroying, the programmer's software. The tester in you is your Mr. Hyde – your Incredible Hulk. He must exercise what Gruenberger calls 'low cunning." – Boris Beizer

Overview

- A Common application of graph criteria is to program **source**
- **Graph**: Usually the control flow graph (CFG)
- **Node coverage**: Execute every statement
- **Edge coverage**: Execute every branch
- **Loops**: Looping structures such as for loops, while loops, etc.
- **Data flow coverage**: Augment the CFG
  - defs are statements that assign values to variables
  - uses are statements that use variables

Control Flow Graphs

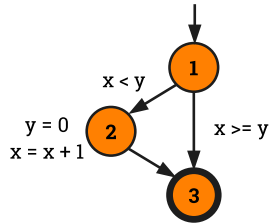
- A **CFG** models all executions of a method by describing control structures
- **Nodes**: Statements or sequences of statements (basic blocks)
- **Edges**: Transfers of control
- **Basic Block**: A sequence of statements such that if the first statement is executed, all statements will be (no branches)
- CFGs are sometimes annotated with extra information
  - branch predicates
  - defs
  - uses
- Rules for translating statements into graphs ...

Idaho State University
Computer Science

CFG: The If Statement

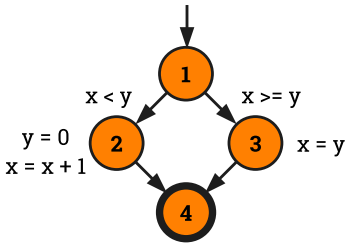
Basic If

```
if (x < y) {
  y = 0;
  x = x + 1;
}
```



If/Else

```
if (x < y) {
  y = 0;
  x = x + 1;
} else {
  x = y;
}
```

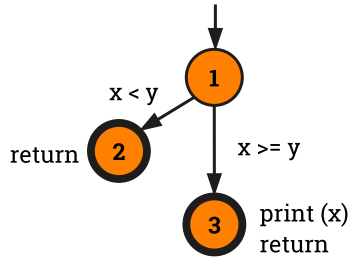


ROAR

CFG: The If-Return Statement

If with Return

```
if (x < y) {
  return;
}
print(x);
return;
```



No edge from node 2 to 3. The return nodes must be distinct.

Loops

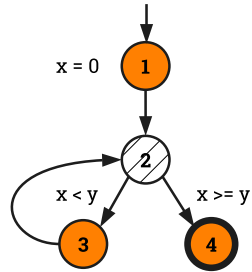
- Loops require “extra” nodes to be added
- Nodes that **do not** represent statements or basic blocks

Idaho State University
Computer Science

CFG: while and for Loos

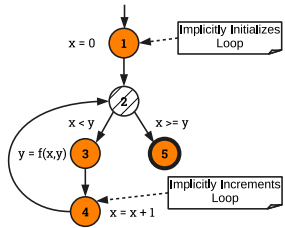
While Loop

```
x = 0;
while (x < y) {
  y = f(x, y);
  x = x + 1;
}
return(x);
```



For Loop

```
for (x = 0; x < y; x++) {
  y = f(x, y);
}
return(x);
```

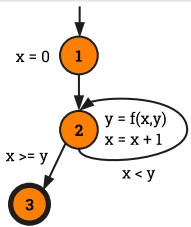


ROAR

CFG: do Loop, break and continue

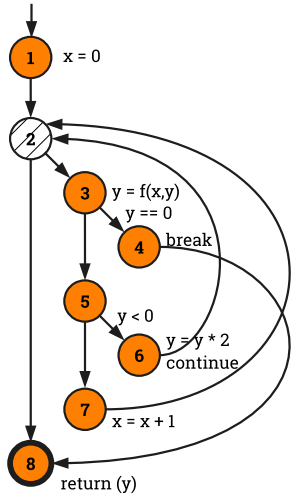
Do Loop

```
x = 0;
do {
  y = f(x, y);
  x = x + 1;
} while (x < y);
return(y);
```



While with Break/Cont

```
x = 0;
while(x < y) {
  y = f(x,y);
  if (y == 0) {
    break;
  } else if (y < 0) {
    y = y * 2;
    continue;
  }
  x = x + 1;
}
return(y);
```

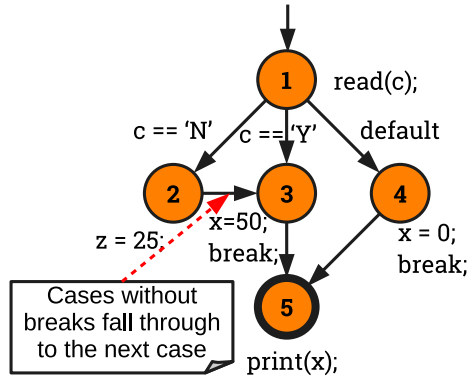


Idaho State University
Computer Science

CFG: The case (switch) Structure

Switch

```
read(c);
switch (c) {
  case 'N':
    z = 25;
  case 'Y':
    x = 50;
    break;
  default:
    x = 0;
    break;
}
print(x);
```

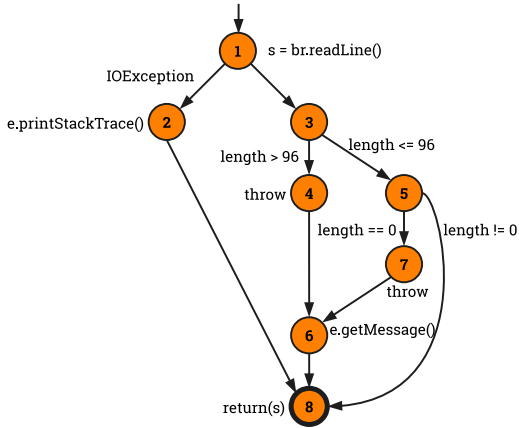


ROAR

CFG: Exceptions (try-catch)

Exception Handling

```
try {
  s = br.readLine();
  if (s.length() > 96)
    throw new Exception("too long");
  if (s.length() == 0)
    throw new Exception("too short");
} catch (IOException e) {
  e.printStackTrace();
} catch (Exception e) {
  e.getMessage();
}
return(s);
```



Idaho State University
Computer Science

Example Control Flow - Stats

Draw the graph and label the edges

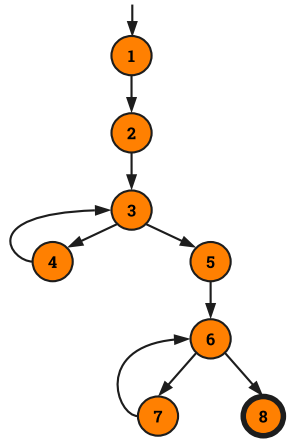
```
public static void computeStats (int[] numbers) {
     int length = numbers.length;
     double med, var, sd, mean, sum, varsum;

     sum = 0.0;
     for (int i = 0; i < length; i++) {
          sum += numbers[i];
     }
     med   = numbers[length / 2];
     mean = sum / (double) length;

     varsum = 0.0;
     for (int i = 0; i < length; i++) {
          varsum = varsum + ((numbers[i] - mean) * (numbers[i] - mean));
     }
     var = varsum / (length - 1);
     sd  = Math.sqrt (var);

     System.out.println("length:              " + length);
     System.out.println("mean:                " + mean);
     System.out.println("median:              " + med);
     System.out.println("variance:            " + var);
     System.out.println("standard deviation: " + sd);
}
```


ROAR

Idaho State University
Computer Science

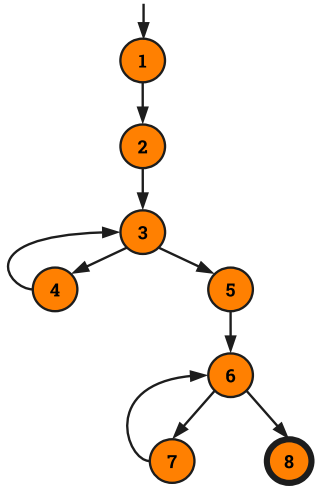
Example Control Flow - Stats

```
public static void computeStats (int[] numbers) {
     int length = numbers.length;
     double med, var, sd, mean, sum, varsum;

     sum = 0.0;
     for (int i = 0; i < length; i++) {
          sum += numbers[i];
     }
     med   = numbers[length / 2];
     mean = sum / (double) length;

     varsum = 0.0;
     for (int i = 0; i < length; i++) {
          varsum = varsum + ((numbers[i] - mean) * (numbers[i] - mean));
     }
     var = varsum / (length - 1);
     sd  = Math.sqrt (var);

     System.out.println("length:              " + length);
     System.out.println("mean:                " + mean);
     System.out.println("median:              " + med);
     System.out.println("variance:            " + var);
     System.out.println("standard deviation: " + sd);
}
```



ROAR

Control Flow TRs and Test Paths - EC



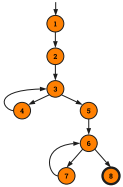
Edge Coverage

TRs

A. [1, 2] **B.** [2, 3] **C.** [3, 4]
D. [3, 5] **E.** [4, 3] **F.** [5, 6]
G. [6, 7] **H.** [6, 8] **I.** [7, 6]

Test Path: [1,2,3,4,3,5,6,7,6,8]

Control Flow TRs and Test Paths - EPC



Edge-Pair Coverage

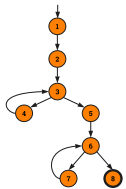
TRs
A. [1, 2, 3] **B.** [2, 3, 4] **C.** [2, 3, 5] **D.** [3, 4, 3]
E. [3, 5, 6] **F.** [4, 3, 5] **G.** [5, 6, 7] **H.** [5, 6, 8]
I. [6, 7, 6] **J.** [7, 6, 8] **K.** [4, 3, 4] **L.** [7, 6, 7]

Test Paths:
i. [1,2,3,4,3,5,6,7,6,8]
ii. [1,2,3,5,6,8]
iii. [1,2,3,4,3,4,3,5,6,7,6,7,6,8]

TP	TRs toured	sidetrips
i	~~A,B,D,E,F,G,I~~	C, H
ii	A,C,E,H	
iii	A,B,D,E,F,G,I,J,K,L	C, H

TP iii make TP i redundant. A **minimal** set of TPs is cheaper.

Control Flow TRs and Test Paths - PPC



Prime Path Coverage

TRs:
A. [3, 4, 3] **B.** [4, 3, 4] **C.** [7, 6, 7] **D.** [6, 7, 6]
E. [1, 2, 3, 4] **F.** [4, 3, 5, 6, 7] **G.** [4, 3, 5, 6, 8]
H. [4, 3, 5, 6, 8] **I.** [1, 2, 3, 5, 6 ,7] **J.** [1, 2, 3, 5, 6, 8]

Test Paths:
i. [1,2,3,4,3,5,6,7,6,8]
ii. [1,2,3,4,3,4,3,5,6,7,6,7,6,8]
iii. [1,2,3,4,3,5,6,8]
iv. [1,2,3,5,6,7,6,8]
v. [1,2,3,5,6,8]

TP	TRs toured	sidetrips
i	~~A,D,E,F,G~~	H, I, J
ii	A,B,C,D,E,F,G	H, I, J
iii	A,F,H	J
iv	D,E,F,I	J
v	J	

Data Flow Coverage for Source

- **def**: a location where a value is stored into **memory**
  - x appears on the **left side** of an assignment ($x = 44;$)
  - x is an **actual parameter** in a call and the method **changes** its value
  - x is a **formal parameter** of a method (implicit def when method starts)
  - x is an **input** to a program
- **use**: a location where variable's value is **accessed**
  - x appears on the **right side** of an assignment
  - x appears in a conditional **test**
  - x is an **actual parameter** to a method
  - x is an **output** of the program
  - x is an output of a method in a **return** statement
- If a def and a use appear on the **same node**, then it is only a DU-pair if the def occurs **after** the use and the node is in a loop.

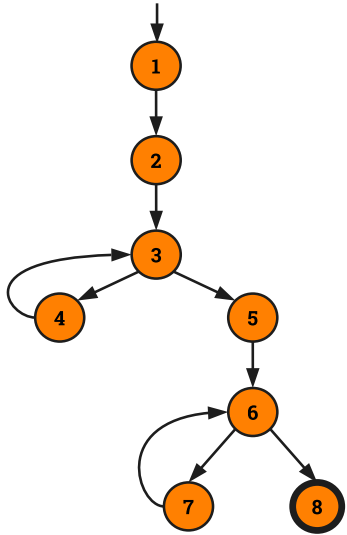
Idaho State University
Computer Science

Example Data Flow - Stats

```
public static void computeStats (int[] numbers) {
     int length = numbers.length;
     double med, var, sd, mean, sum, varsum;

     sum = 0.0;
     for (int i = 0; i < length; i++) {
          sum += numbers[i];
     }
     med   = numbers[length / 2];
     mean = sum / (double) length;

     varsum = 0.0;
     for (int i = 0; i < length; i++) {
          varsum = varsum + ((numbers[i] - mean) * (numbers[i] - mean));
     }
     var = varsum / (length - 1);
     sd  = Math.sqrt (var);

     System.out.println("length:              " + length);
     System.out.println("mean:                " + mean);
     System.out.println("median:              " + med);
     System.out.println("variance:            " + var);
     System.out.println("standard deviation: " + sd);
}
```

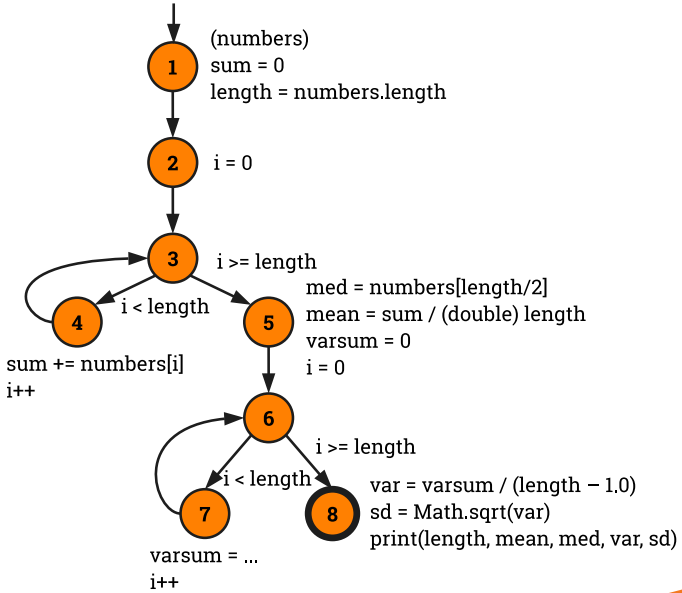
ROAR

Control Flow Graph for Stats



Idaho State University
Computer Science

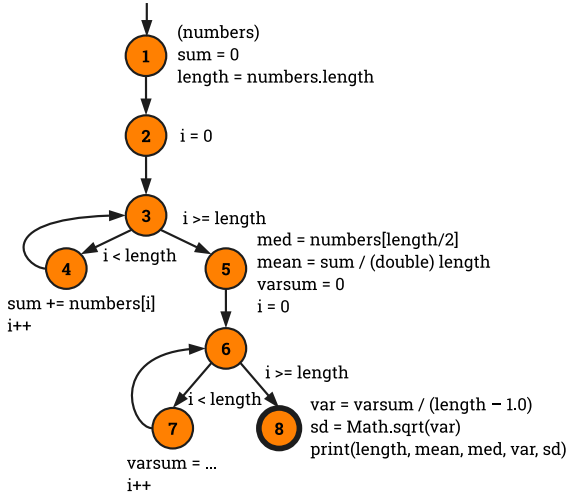
Control Flow Graph for Stats



ROAR

Idaho State University
Computer Science

CFG for Stats - With Defs & Uses

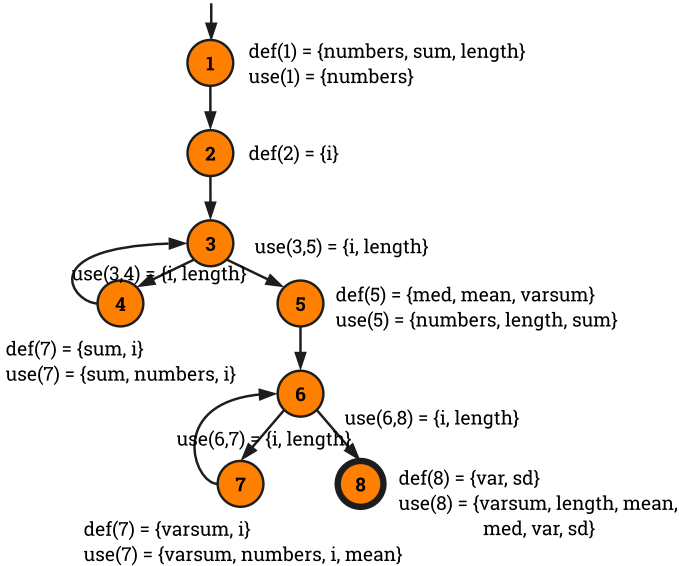


Turn the annotations into defs and use sets…

ROAR

Idaho State University
Computer Science

CFG for Stats - With Defs & Uses



ROAR

Idaho State University
Computer Science

Defs and Uses Tables for Stats

Node	Def	Use
1	{numbers, sum, length}	{numbers}
2	{i}	
3		
4	{sum, i}	{numbers, i, sum}
5	{med, mean, varsum, i}	{numbers, length, sum}
6		
7	{varsum, i}	{varsum, numbers, i, mean}
8	{var, sd}	{varsum, length, var, mean, med, var, sd}

Edge	Use
(1, 2)	
(2, 3)	
(3, 4)	{i, length}
(4, 3)	
(3, 5)	{i, length}
(5, 6)	
(6, 7)	{i, length}
(7, 6)	
(6, 8)	{i, length}

ROAR

Idaho State University
Computer Science

DU Pairs for Stats

variable	DU Pairs
numbers	(1,4) (1,5) (1,7)
length	(1,5) (1.8) (1, (3,4)) (1, (3,5)) (1, (6,7)) (1, (6,8))
med	(5,8)
var	**(8,8)** def comes before use, not a DU pair
sd	**(8,8)** def comes before use, not a DU pair
mean	(5,7) (5,8)
sum	(1,4) (1,5) **(4,4)** (4,5) def after use in loop -> DU pair
varsum	(5,7) (5,8) **(7,7)** (7,8) no def-clear path
i	(2,4) (2, (3,4)) (2, (3,5)) ~~(2,7) (2, (6,7)) (2, (6,8))~~ (4,4) (4, (3,4)) (4, (3,5)) ~~(4,7) (4, (6,7)) (4, (6,8))~~ (5,7) (5, (6,7)) (5, (6,8)) **(7,7)** (7, (6,7)) (7, (6,8)) no def-clear path

ROAR

Idaho State University
Computer Science

DU Paths for Stats

variable	DU Pairs	DU Paths
numbers	(1,4)	[1,2,3,4]
	(1,5)	[1,2,3,5]
	(1,7)	[1,2,3,5,6,7]
length	(1,5)	[1,2,3,5]
	(1,8)	[1,2,3,5,6,8]
	(1, (3,4))	[1,2,3,4]
	(1, (3,5))	[1,2,3,4]
	(1, (6,7))	[1,2,3,4,5,7]
	(1, (6,8))	[1,2,3,5,6,8]
med	(5,8)	[5,6,8]
var	(8,8)	no path needed
sd	(8,8)	no path needed
sum	(1,4)	[1,2,3,4]
	(1,5)	[1,2,3,4]
	(4,4)	[4,3,4]
	(4,5)	[4,3,5]

variable	DU Pairs	DU Paths
mean	(5,7)	[5,6,7]
	(5,8)	[5,6,8]
varsum	(5,7)	[5,6,7]
	(5,8)	[5,6,8]
	(7,7)	[7,6,7]
	(7,8)	[7,6,8]
i	(2,4)	[2,3,4]
	(2, (3,4))	[2,3,4]
	(2, (3,5))	[2,3,5]
	(4,4)	[4,3,4]
	(4, (3,4))	[4,3,4]
	(4, (3,5))	[4,3,5]
	(5, 7)	[5,6,7]
	(5, (6,7))	[5,6,7]
	(5, (6,8))	[5,6,8]
	(7,7)	[7,6,7]
	(7, (6,7))	[7,6,7]
	(7, (6,8))	[7,6,8]

ROAR

Idaho State University
Computer Science

DU Paths for Stats–No Duplicates

There are 38 DU paths for Stats, but only 12 unique

[1,2,3,4] [1,2,3,5]	[4,3,4] [4,3,5]
[1,2,3,5,6,7] [1,2,3,5,6,8]	[5,6,7] [5,6,8]
[2,3,4] [2,3,5]	[7,6,7] [7,6,8]

6 require at least one iteration of a loop

4 Expect a loop not to be “entered”

2 require at least two iterations of a loop

ROAR

Idaho State University
Computer Science

Test Cases and Test Paths

Test Case: numbers = (44); length = 1

- **Test Path**: [1, 2, 3, 4, 3, 5, 6, 7, 6, 8]
- **Additional DU Paths covered (no sidetrips)**
  - [1,2,3,4] [2,3,4] [4,3,5] [5,6,7] [7,6,8]
  - The five stars that require at least one iteration of a loop

Test Case: numbers = (2,10,15); length = 3

- **Test Path**: [1,2,3,4,3,4,3,4,3,5,6,7,6,7,6,7,6,8]
- **DU Paths covered (no sidetrips)**
  - [4, 3, 4] [7, 6, 7]
  - The two stars that require at least two iterations of a loop

- Other DU paths require arrays with length 0 to skip loops
- But the method fails with index out of bounds exception…
  - `med = numbers[length/2];`

ROAR

Summary

- Applying the graph test criteria to **control flow graphs** is relatively straightforward
  - Most of the developmental **research** work was done with CFGs
- A few **subtle decisions** must be made to translate control structures into the graph
- Some tools will assign each statement to a **unique node**
  - These slides and the book uses **basic blocks**
  - Coverage is the same, although the **bookkeeping** will differ

Are there any questions?