Aimee Valladares

CEN 4072

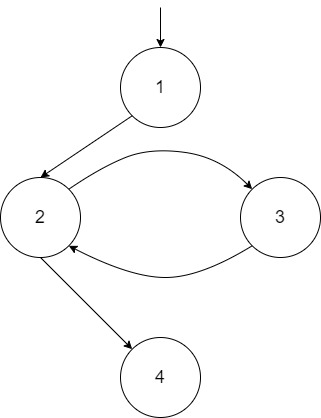
2 March 2022

# Assignment #4

Q1) (25 points) Answer questions a-d for the graph defined by the following sets:

* N = {1, 2, 3, 4}
* N0 = {1}
* Nf = {4}
* E = {(1; 2); (2; 3); (3; 2); (2; 4)}

1. Draw the graph



1. If possible, list test paths that achieve Node Coverage, but not Edge Coverage. If not possible, explain why not.

It is not possible to list the test paths that achieve Node Coverage but *not* Edge Coverage because all the edges that must be navigated through to cover all the nodes.

1. If possible, list test paths that achieve Edge Coverage, but not Edge Pair Coverage. If not possible, explain why not.

Edge Pair Table:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 4 |
| 1 | 2 | 3 |
| 2 | 3 | 2 |
| 3 | 2 | 3 |
| 3 | 2 | 4 |

The test path **1 2 3 2 4** achieves Edge Coverage without achieving Edge Pair Coverage due to not containing the edge pair (3 2 3).

1. List test paths that achieve Edge Pair Coverage.

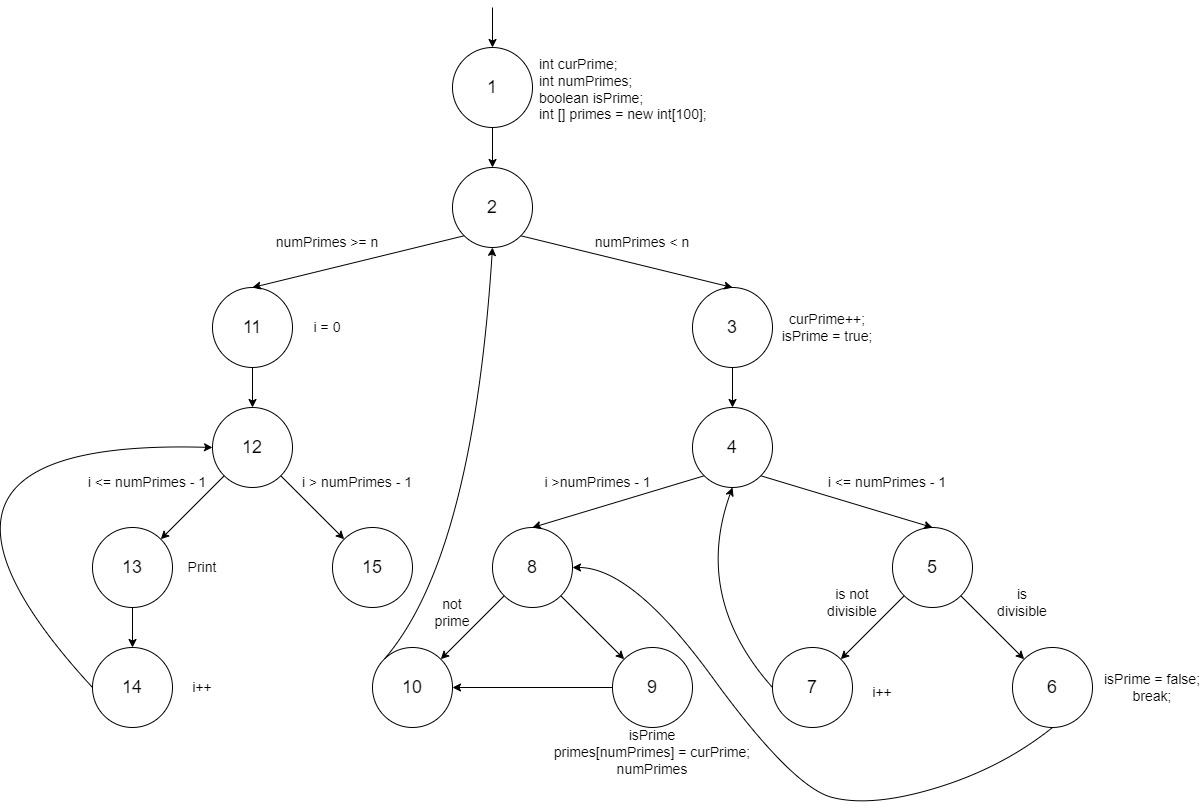
Edge Pair Table:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 4 |
| 1 | 2 | 3 |
| 2 | 3 | 2 |
| 3 | 2 | 3 |
| 3 | 2 | 4 |

The test paths that achieve Edge Pair Coverage are **1 2 3 2 3 2 4** and **1 2 4.**

Q2) (25 points) Use the method printPrimes() for questions a-e below. A compilable version is available on the book website in the file PrintPrimes.java. A line-numbered version suitable for this exercise is available on the book website in the file PrintPrimes.num.

1. Draw the control flow graph for the printPrimes() method.



1. Consider test cases t1 = (n = 3) and t2 = (n = 5). Although this tour the same prime paths in printPrimes(), they do not necessarily find the same faults. Design a simple fault that t2 would be more likely to discover than t1 would.

A fault that t2 would more likely discover than t1 is setting the value for primes to 4 (I.e. int [] primes = new int [100]; --> int [] primes = new int [4]) because it will create a boundary fault and overflow when n = 5.

1. For printPrimes(), find a test case such that the corresponding test path visits the edge that connects the beginning of the while statement to the for statement that appears after the while loop, without going through the body of the while loop.

The test case that visits the edge connecting the beginning while statement to the last for statement is between nodes 2 and 11 which would occur if n is equal to 1 or 0 as it completely passes the right-hand side.

1. List the test requirements for Node Coverage and Edge Coverage.

The following test requirements achieve both Node Coverage and Edge Coverage are (1, 2), (2, 3), (2, 11), (3, 4), (4, 5), (4, 8), (5, 6), (5, 7), (6, 8), (7, 4), (8, 9), (8, 10), (9, 10), (10, 2), (11, 12), (12, 13), (12, 15), (13, 14), and (14, 12).

1. List test paths that achieve Node Coverage but not Edge Coverage on the graph.

The following test requirements achieve Node Coverage *but not* Edge Coverage are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15.

Q3) (25 points) Use the following program fragment for questions a-d below.

public static void f1 (int x, int y)

{

if (x < y) { f2 (y); } else { f3 (y); };

}

public static void f2 (int a)

{

if (a % 2 == 0) { f3 (2\*a); };

}

public static void f3 (int b)

{

if (b > 0) { f4(); } else { f5(); };

}

public static void f4() {...f6()...}

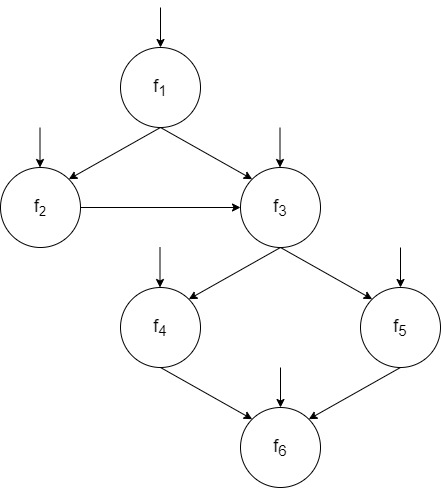
public static void f5() {...f6()...}

public static void f6() {...}

Use the following test inputs:

* t1 = f1 (0, 0)
* t2 = f1 (1, 1)
* t3 = f1 (0, 1)
* t4 = f1 (3, 2)
* t5 = f1 (3, 4)

1. Draw the call graph for this program fragment.



1. Give the path in the graph followed by each test.

t1 = f1 (0, 0) ---------> [f1, f3, f5, f6]

t2 = f1 (1, 1) ----------> [f1, f3, f4, f6]

t3 = f1 (0, 1) ----------> [f1, f2]

t4 = f1 (3, 2) ----------> [f1, f3, f4, f6]

t5 = f1 (3, 4) ----------> [f1, f2, f3, f4, f6]

1. Find a minimal test set that achieves Node Coverage.

A minimal test set that achieves Node Coverage is {t1, t2, t3}.

1. Find a minimal test set that achieves Edge Coverage.

A minimal test that achieves Edge Coverage is {t1, t5}.

Q4) (25 points) For the following questions a-c, consider the FSM that models a (simplified) programmable thermostat. Suppose the variables that define the state and the methods that transition between states are:

partOfDay : {Wake, Sleep}

temp : {Low, High}

// Initially “Wake” at “Low temperature”

// Effects: Advance to next part of day

public void advance();

// Effects: Make current temp higher, if possible

public void up();

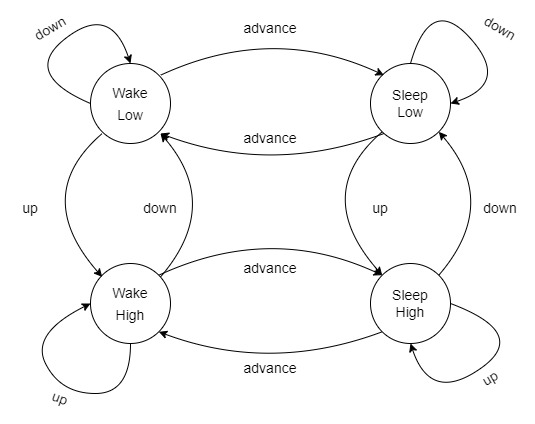
// Effects: Make current temp lower, if possible

public void down();

1. How many states are there?

There are 4 states: **Wake Low, Wake High, Sleep Low, Sleep High**

1. Draw and label the states (with variable values) and transitions (with method names). Notice that all the methods are total.



1. A test case is simply a sequence of method calls. Provide a test set that satisfies Edge Coverage on your graph.

Wake Low *--down-->* Wake Low –*advance-->* Sleep Low –*down-->* Sleep Low –*up-->* Sleep High – *up-->* Sleep High –*down-->* Sleep Low –*advance-->* Wake Low –*up-->* Wake High –up--> Wake High –advance --> Sleep High –advance--> Wake High –down--> Wake Low

The test case would be {Wake Low, Wake Low, Sleep Low, Sleep Low, Sleep High, Sleep High, Sleep Low, Wake Low, Wake High, Wake High. Sleep High, Wake High, Wake Low}