Testing, debugging & verification: Testing 2

Atze van der Ploeg

Who from GU want some store credit??

Admin stuff:

- Sign up for the google group!
- Make sure you are registered!
- Exercise session this afternoon! Bring laptops!

Previously on TDV

Terminology : Bug, Specification, Failure, Correct, Testing, Debugging, (Formal) Verification, Supplier, Client, Precondition, Postcondition, Unit test, Test suite, Oracle

When do we have enough tests?

```
public static boolean and(boolean a, boolean b)
```

Requires: Nothing (Input is well typed)

Ensures: The output is true if and only if both inputs are true and false otherwise.

Tests: ?

When do we have enough tests?

```
public static boolean and(boolean a, boolean b)
```

Requires: Nothing (Input is well typed)

Ensures: The output is true if and only if both inputs are true and false otherwise.

Tests:
and(false, false) == false
and(true, false) == false
and(false, true) == false
and(true, true) == true

When do we have enough tests?

```
public static boolean and(boolean a, boolean b)
```

Requires: Nothing (Input is well typed)

Ensures: The output is true if and only if both inputs are true and false otherwise.

Tests:
and(false, false) == false
and(true, false) == false
and(false, true) == false
and(true, true) == true

When do we have enough tests?

public static int[] sort(int[] arr)

When do we have enough tests?

public static int[] sort(int[] arr)

Infite (or really big) number of possible inputs!
Cannot test all!

When do we have enough tests?

public static int[] sort(int[] arr)

Infite (or really big) number of possible inputs!
Cannot test all!

An answer: coverage criteria

How much of the code is *covered* by the set of tests?

Different ways to define “covered”

Example: russian multiplication

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

Statement coverage

Are all statements is executed in some test?

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

Tests for full statement coverage here?

Statement coverage

Are all statements is executed in some test?

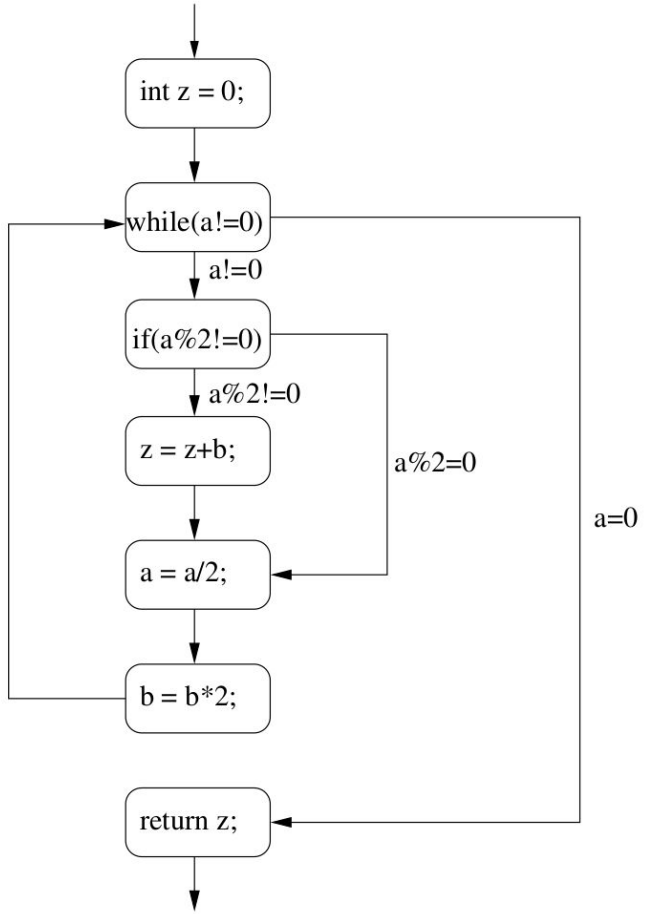
```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

Tests for full statement coverage here?

```
russianMultiplication(1,0) == 0
```

Control flow graph

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

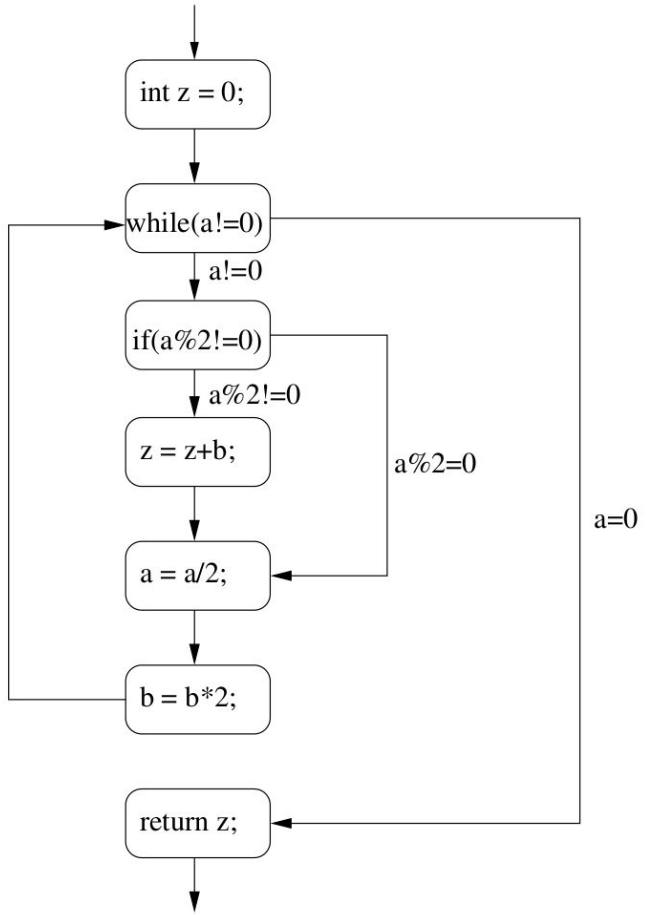


Make *graph* from code:

- *Node* = statement or while/if/for start
- *Edge* from a to b iff next execution step after a can b
- *Label* on edge = condition which should hold to travese edge (or no condition)

Control flow graph

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

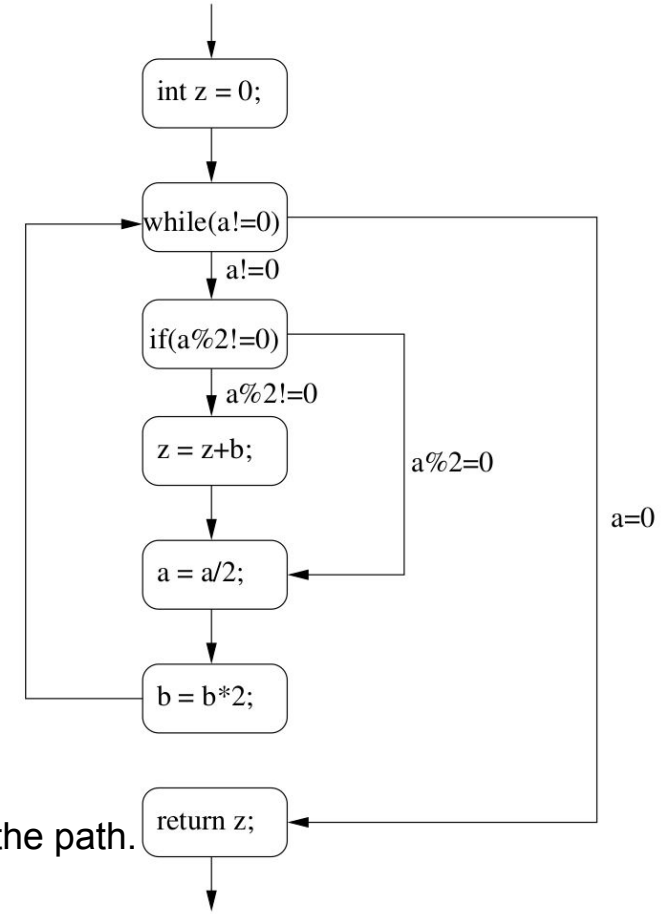


Make *graph* from code:

- *Node* = statement or while/if/for start
- *Edge* from a to b iff next execution step after a can be b
- *Label* on edge = condition which should hold to travese edge (or no condition)

Execution path

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

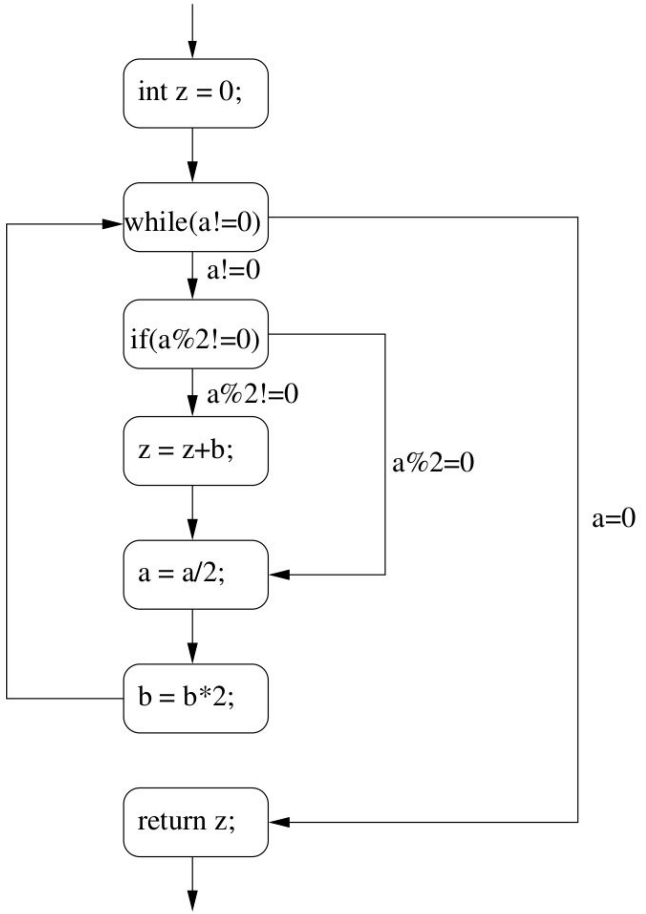


A *path* in a graph is a sequence of nodes,
such that there is an edge between any 2 subsequent nodes in the path.
(Can be infinite.)

Execution path is a path through a control flow graph

Execution path - example

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

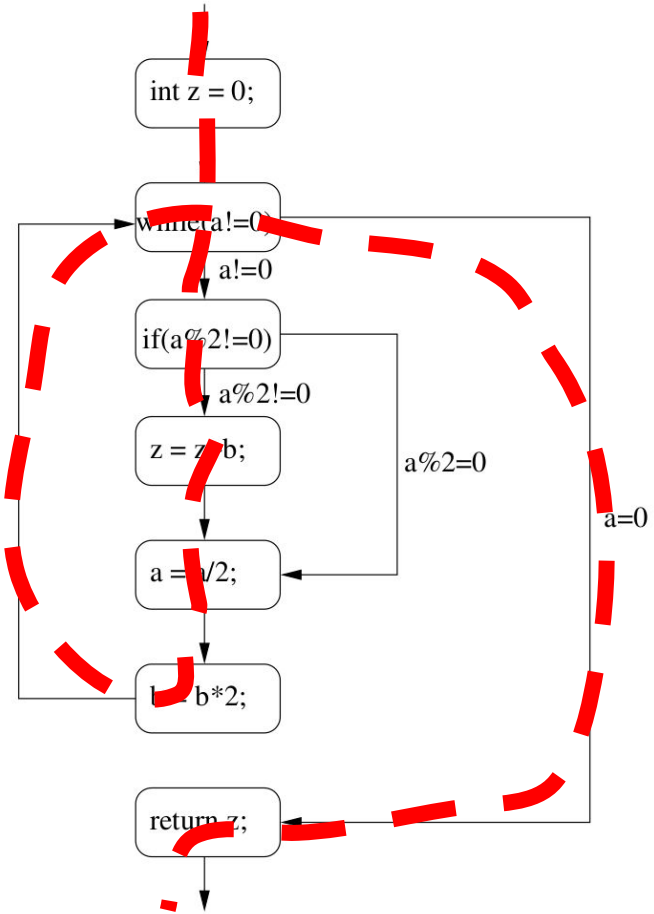


Each test case has an execution path.

`russianMultiplication(1,0) == 0`

Execution path - example

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```



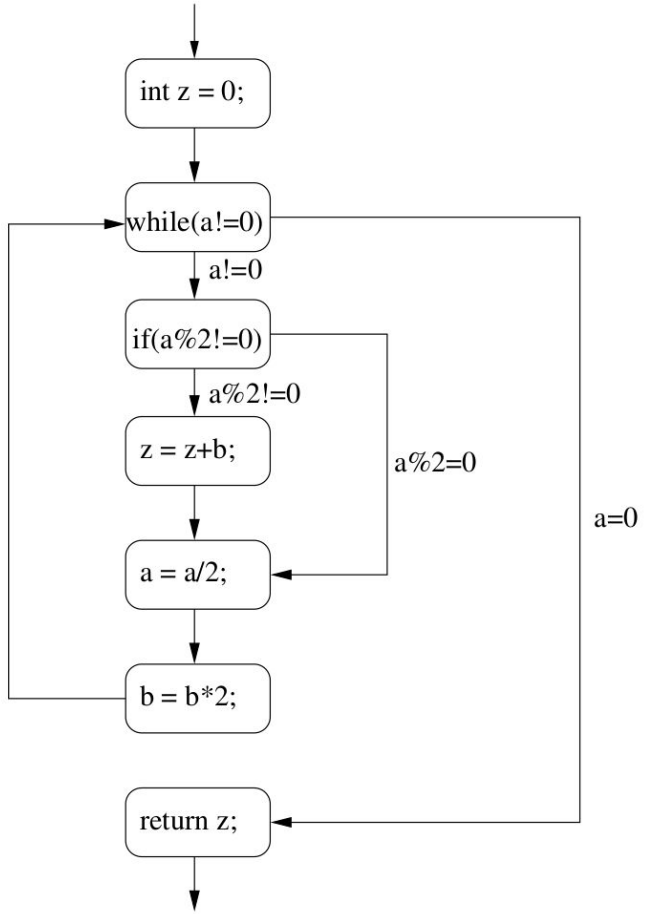
Each test case has an execution path.

`russianMultiplication(1,0) == 0`

Note all nodes are visited, so statement coverage

Branch coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

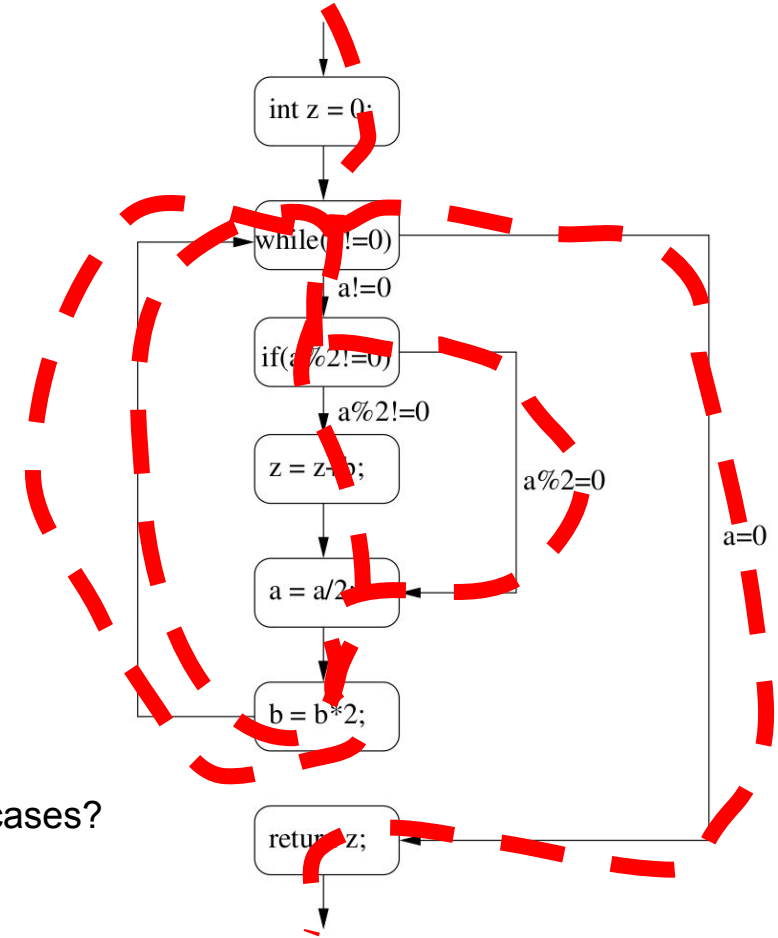


Branch coverage = are all *edges* is taken in some test case?

Test cases for full branch coverage here?

Branch coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

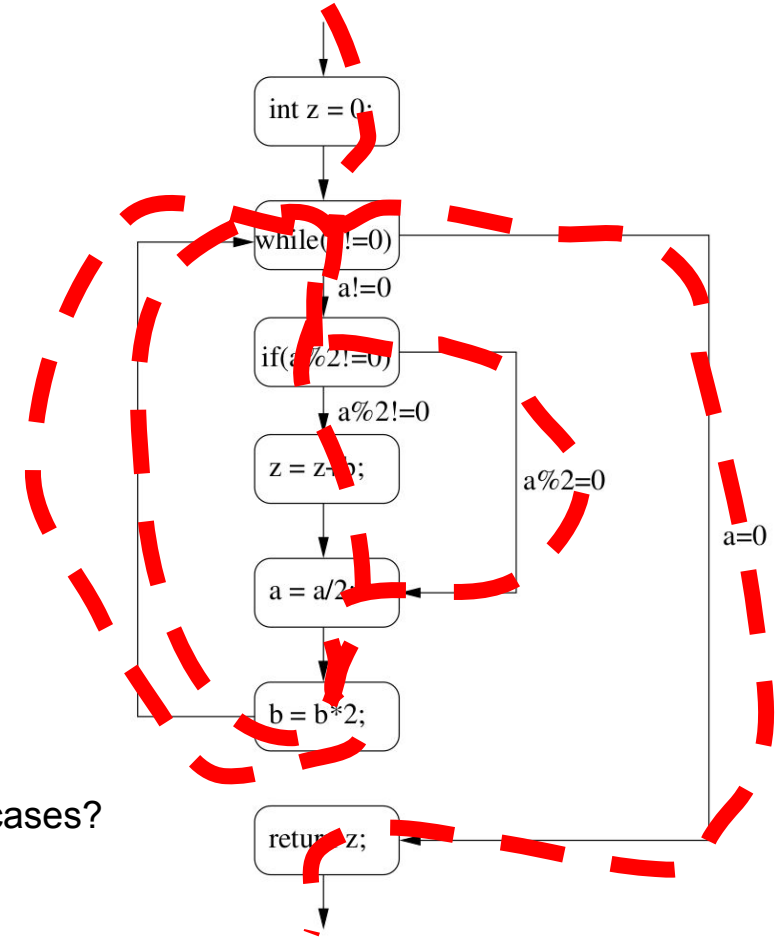


Branch coverage = which % of *edges* is taken in all test cases?

`russianMultiplication(2,0) == 0`

Branch coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```



Branch coverage = which % of *edges* is taken in all test cases?

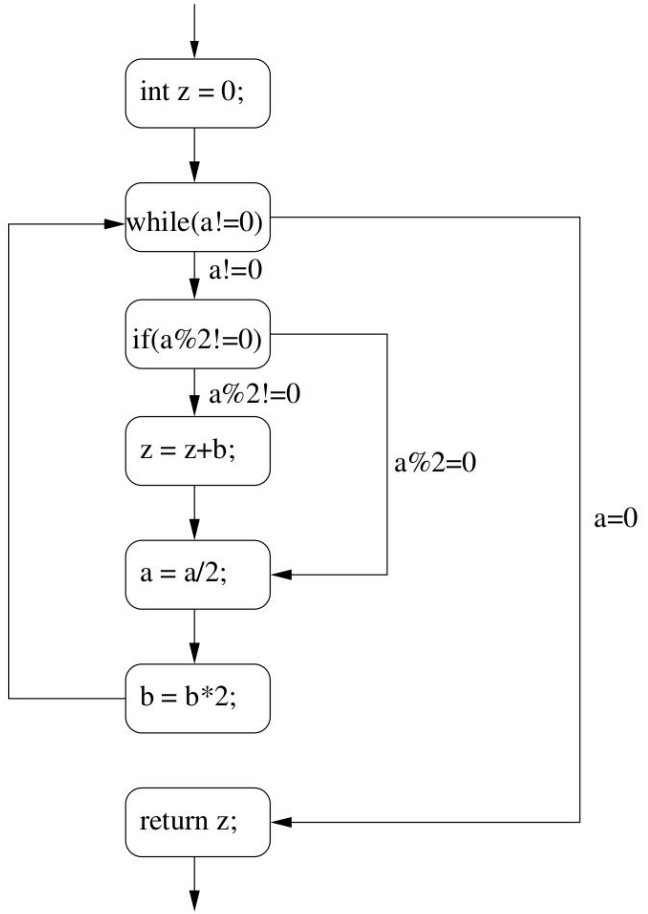
`russianMultiplication(2,0) == 0`

To visit all edges, we need to visit all nodes, so branch coverage implies statement coverage

Path coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

Path coverage = Are all *paths* taken in some test case?

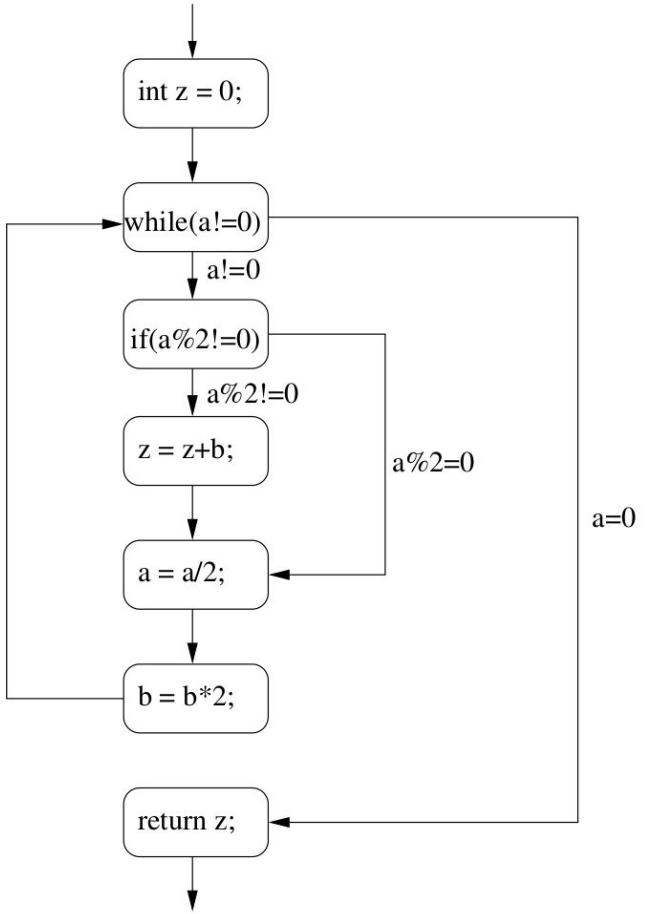


Path coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

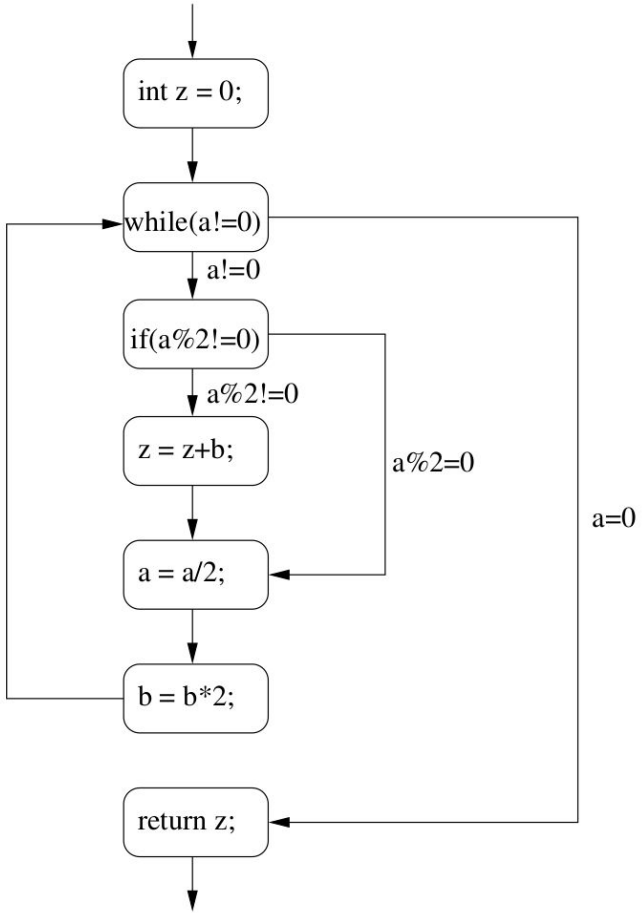
Path coverage = Are all *paths* taken in some test case?

Usually not realistic: > 2^31 paths here!



Mini quiz: Coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```

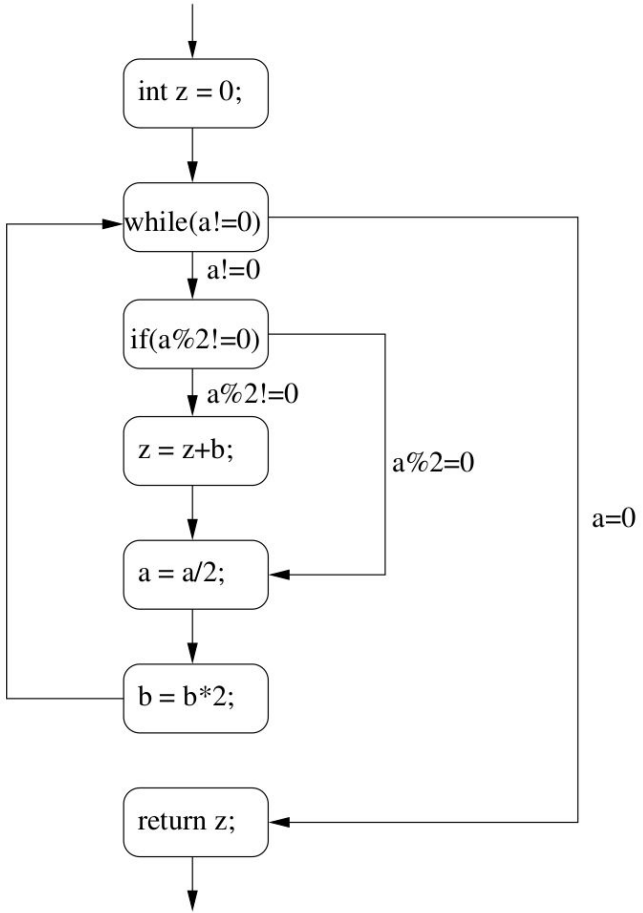


Do the following individual inputs achieve full statement coverage, branch coverage, path coverage?

```
a = 3, b = 3
a = 0, b = 2
a = 4, b = 1
```

Mini quiz: Coverage

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```



Do the following individual inputs achieve full statement coverage, branch coverage, path coverage?

a = 3, b = 3 SC

a = 0, b = 2 None

a = 4, b = 1 SC and BC

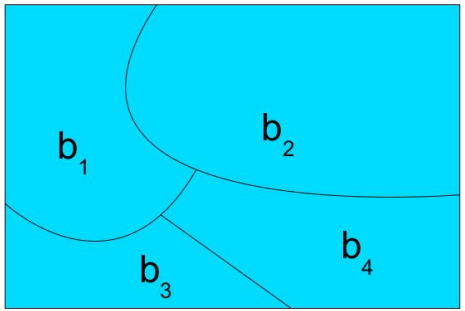
How to find good unit tests?

An anwer: *Input space partioning*

Devide input space into regions with for which the program acts “similar”.

Take an some inputs from *each region*

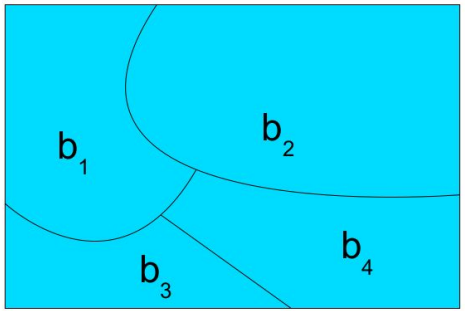
More math training!



More math training!

A partition of set s is a set of sets $\{b_1, \ldots, b_n\}$ such that:

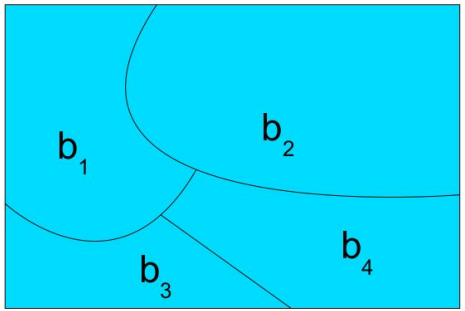
- $\forall i \,.b_i \neq \emptyset$
- $b_1 \cup b_2 \cup \cdots \cup b_n = s$
- $\forall i\, j.\; i \neq j \rightarrow b_i \cap b_j = \emptyset$

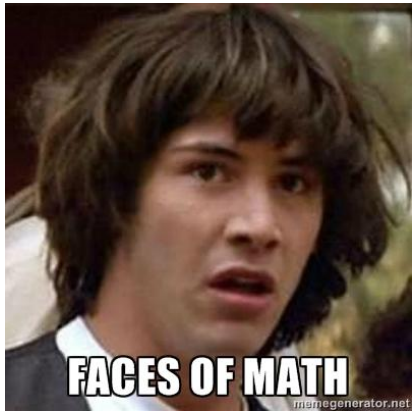


More math training!

A partition of set s is a set of sets $\{b_1, \ldots, b_n\}$ such that:

- $\forall i \,.\, b_i \neq \emptyset$
- $b_1 \cup b_2 \cup \cdots \cup b_n = s$
- $\forall i\, j.\; i \neq j \rightarrow b_i \cap b_j = \emptyset$





Example: Partitions of natural numbers

{prime numbers, non-prime numbers}

{even,odd}

{ {x | x % 3 == 0} , {x | x % 3 == 1}, {x | x % 3 == 2}}

Example: Partitions of arrays

Is this a partition of int[]?

{ {null}, {sorted in ascending order}, {sorted in descending order}, {unsorted}}

Example: Partitions of arrays

Is this a partition of int[]?

{ {null}, {sorted in non-decreasing order}, {sorted in descreasing order}, {not sorted}}

Let’s try

```
findElement (int[] arr, int elem)
```

Partitions:

```
r = {{ empty array}, {non-empty arrays}}

q = {{null}, {non-null arrays}}

s = {{null}, {x | x.length == 0}, {x | x.length == 1},

{x | x.length > 1}}
```

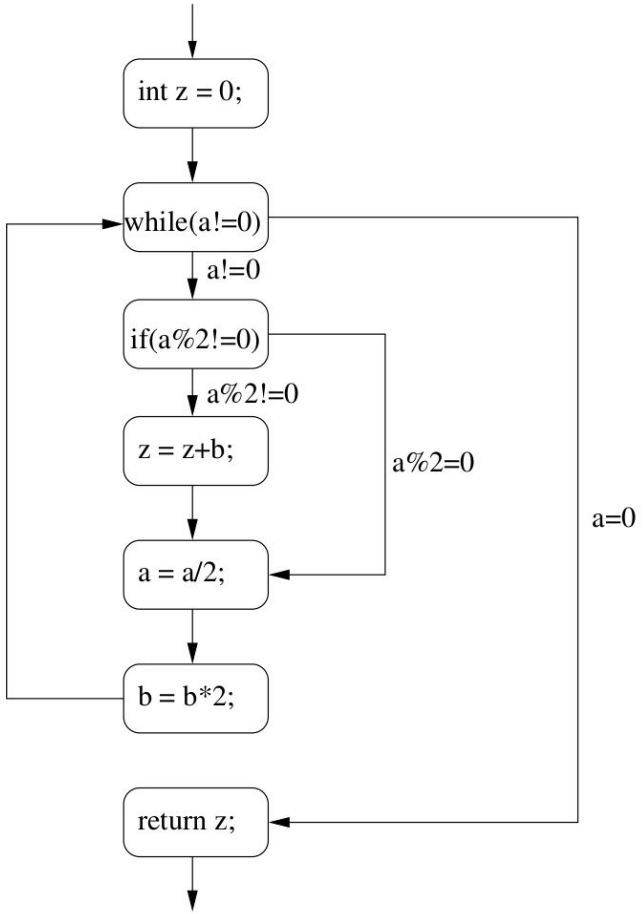
r is a sub-partition of q

Strategy for finding good inputs

1. Devide input space into regions with for which the program acts “similar”.
2. Choose inputs from each regions, especially from *borders*

Example:

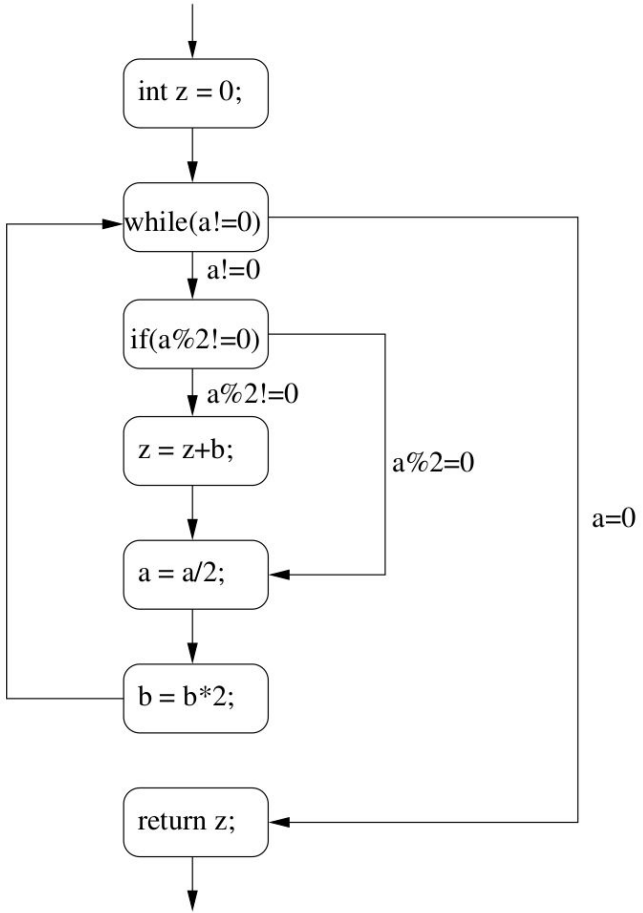
```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```



Input partition?

Example:

```
int russianMultiplication(int a, int b){
  int z = 0;
  while(a != 0){
    if(a%2 != 0){
      z = z+b;
    }
    a = a/2;
    b = b*2;
  }
  return z;
}
```



For example:
{ {(0,0}, {(x,y) | x > 0, y > 0} , {(x,y) | x < 0, y < 0}, {(x,y) | x < 0, y > 0} , {(x,y) | x < 0, y > 0}}

More coverage critaria

Logic coverage:

Based on boolean (sub-)expressions in the program.

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

More coverage critaria

Logic coverage:

Based on boolean (sub-)expressions in the program.

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Seems familiar?

More coverage critaria

Logic coverage:

Based on boolean (sub-)expressions in the program.

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Seems familiar?

decision coverage <-> branch coverage...

More coverage critaria

Logic coverage:

Based on the boolean (sub-)expressions in the program.

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Seems familiar?

Full decision coverage <-> full branch coverage...

Note: Many languages have *implicit* decisions! Type-checks, null checks etc.

Decision coverage - example

Program contains a bit :

```
if ( a < c || b > c) {
   ...
}
```

Need one test where this evals to true, one where false.

For example,

```
a = 0, c = 2, b = 1
```

```
a = 10, c = 0, b = 1
```

Condition coverage

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Condition coverage: ... Take all outcomes of ***atomic boolean sub-expressions*** that occur in a decision. (atomic = contains no boolean sub-expressions) Do all of them occur in a test?

```
if ((a < b) || D) && (m ≥ n * o) ...
```

atomic boolean subexpressions: (a < b), D, (m >= n * o)

Condition coverage

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Condition coverage: ... Take all outcomes of ***atomic boolean sub-expressions*** that occur in a decision. (atomic = contains no boolean sub-expressions) Do all of them occur in a test?

```
if ((a < b) || D) && (m ≥ n * o) ...
```

atomic boolean subexpressions: (a < b), D, (m >= n * o),

Note: Condition does not subsume decision, nor vice versa

Example: if (a || b) : ((a = true, b = false), (a = false, b = true)) has CC, but not DC

Condition Decision coverage

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Condition coverage: ... Take all outcomes of ***atomic boolean sub-expressions*** that occur in a decision. (atomic = contains no boolean sub-expressions) Do all of them occur in a test?

Condition decision coverage: Take all outcomes of all decisions and conditions in these decision. Do all outcomes occur in a test?

Condition Decision coverage

Decision coverage: Take all outcomes of boolean expressions where the program branches on (i.e. from while, if, for) Do all of them occur in a test?

Condition coverage: ... Take all outcomes of ***atomic boolean sub-expressions*** that occur in a decision. (atomic = contains no boolean sub-expressions) Do all of them occur in a test?

Condition decision coverage: Take all outcomes of all decisions and conditions in these decision. Do all outcomes occur in a test?

```
if ((a < b) || D) && (m ≥ n * o) ...
```

Independent conditions

conditions are *independent* if they cannot influence each other)

independent?

```
a || b || c
(a > b) || (a <= b)
x.length > 0 && x.contains(1)
```

Independent conditions

conditions are *independent* if they cannot influence each other)

independent?

```
a || b || c      --- maybe
(a > b) || (a <= b)  --- No
x.length > 0 && x.contains(1) -- No
```

Modified Condition Decision Coverage (MCDC)

Condition decision coverage: Take all outcomes of all decisions and conditions in these decision. Do all outcomes occur in a test?

Modified condition decision coverage:

Requirement: conditions are *independent* (do not influence each other)

For each condition *c*, in each decision *d*, there is a test such that:

- There is a test such that c == true
- There is a test such that c == false
- If the result of d when c == true is x, then the result of d when c == false must be !x.
- All other conditions in d evaluate identically in both test cases.

Modified Condition Decision Coverage (MCDC)

Requirement: conditions are *independent* (do not influence each other)

For each condition *c*, in each decision *d*, there is a test such that:

- There is a test such that c == true
- There is a test such that c == false
- If the result of d when c == true is x, then the result of d when c == false must be !x.
- All other conditions in d evaluate identically in both test cases.

Let's try: `if ((a < b) || D) && (m ≥ n * o) ...`

Modified Condition Decision Coverage (MCDC)

Requirement: conditions are *independent* (do not influence each other)

For each condition *c*, in each decision *d*, there is a test such that:

- There is a test such that c == true
- There is a test such that c == false
- If the result of d when c == true is x, then the result of d when c == false must be !x.
- All other conditions in d evaluate identically in both test cases.

Let's try: `if ((a < b) || D) && (m ≥ n * o) ...`

Modified Condition Decision Coverage (MCDC)

Requirement: conditions are *independent* (do not influence each other)

For each condition *c*, in each decision *d*, there is a test such that:

- There is a test such that c == true
- There is a test such that c == false
- If the result of d when c == true is x, then the result of d when c == false must be !x.
- All other conditions in d evaluate identically in both test cases.

Let's try: `if ((a < b) || D) && (m ≥ n * o) ...`

```
a = 1, b = 2, D = false, m = 10, n = 1, o = 1
a = 2, b = 1, D = false, m = 10, n = 1, o = 1
```

MCDC is required for some aviation software

MCDC is required in the avionics certification standard DO-178 as the criterion to test adequately Level A software (failure of which is classified as ‘Catastrophic’).

MCDC is required for some aviation software

MCDC is required in the avionics certification standard DO-178 as the criterion to test adequately Level A software (failure of which is classified as ‘Catastrophic’).

Quiz: Logical decision coverage

Condition: if(x < 1 || y > z)

Do the following satisfy, decision, condition, condition decision, MCDC?

```
[x=0, y=0, z=1] and [x=2, y=2, z=1]

[x=2, y=2, z=1] and [x=2, y=0, z=1]

[x=2, y=2, z=2], [x=0, y=0, z=1], [x=2, y=0, z=0], [x=2, y=2, z=1]
```

Quiz: Logical decision coverage

Condition: if(x < 1 || y > z)

Do the following satify, decision, condition, condition decision, MCDC?

`[x=0, y=0, z=1] and [x=2, y=2, z=1]`

CC
DC
CC,DC, DCD, MDCD

`[x=2, y=2, z=1] and [x=2, y=0, z=1]`

`[x=2, y=2, z=2], [x=0, y=0, z=1], [x=2, y=0, z=0], [x=2, y=2, z=1]`

Recap

How do we know we have enough tests?

Answer: Coverage criteria:

- Control flow based : Statement, Branch, Path
- Logic based : Decision, Condition, Decision condition, Modified CDC

How can find tests? (even without code)

Answer: Input space partioning, devide inputs “equally intresting” parts, tests from each part and borders