CSE403 ● Software engineering ● sp12

Week 5-6				
Monday	Tuesday	Wednesday	Thursday	Friday
• Testing I • No reading	• Group meetings	• Midterm	• No Section	• Testing II • Progress report due • Readings out
• Testing III • Readings due	• Group meetings	• TBA	• Section • ZFR due	• ZFR demos

White box: slides and a demo

Today: white box testing
(≡ clear box ≡ transparent box ≡ glass box)

- Goals
  - Ensure test suite covers (executes) all of the program
  - Measure quality of test suite with % coverage
- Assumption
  - High coverage → few mistakes in the program
  - Assumes test produce expected output
- Focus: features not described by specification
  - Control-flow details
  - Performance optimizations

White-box motivation

- Some pieces of code are hard to test fully without knowing the code

```
boolean[] primeTable = new boolean[CACHE_SIZE];
boolean isPrime(int x) {
  if (x>CACHE_SIZE) {
    for (int i=2; i<x/2; i++) {
      if (x%i==0) return false;
    }
    return true;
    } else {
      return primeTable[x];
  }
}
```

- Important transition around **`x = CACHE_SIZE`** that would be hard to guess at since **`CACHE_SIZE`** is hidden from the interface

White Box Testing: Advantages

- Finds an important class of boundaries
  - Yields useful test cases
- Consider **`CACHE_SIZE`** in **`isPrime`** example
  - Need to check numbers on each side of **`CACHE_SIZE`**
    - **`CACHE_SIZE-1`**, **`CACHE_SIZE`**, **`CACHE_SIZE+1`**
  - If **`CACHE_SIZE`** is mutable, may need to test with different **`CACHE_SIZE`**s
- Disadvantages?
  - Tests may have same bugs as implementation
  - What's a statement?

What is full coverage?

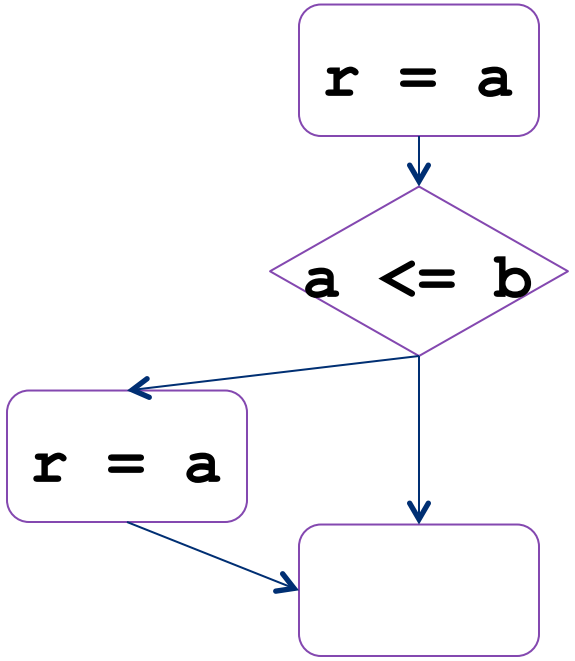
```
static int min (int a, int b) {
    int r = a;
    if (a <= b) {
       r = a;
    }
    return r;
}
```

- Consider any test with **a ≤ b** (e.g., **min(1,2)**)
  - It executes every instruction
  - It misses the bug
- Statement coverage is not enough

Edge coverage

- Another approach is to use a control flow graph (CFG) representation of a program
  - Essentially, a flowchart
- Then ensure that the suite covers all edges in the CFG

```
static int min (int a, int b) {
    int r = a;
    if (a <= b) {
       r = a;
    }
    return r;
}
```



Condition coverage

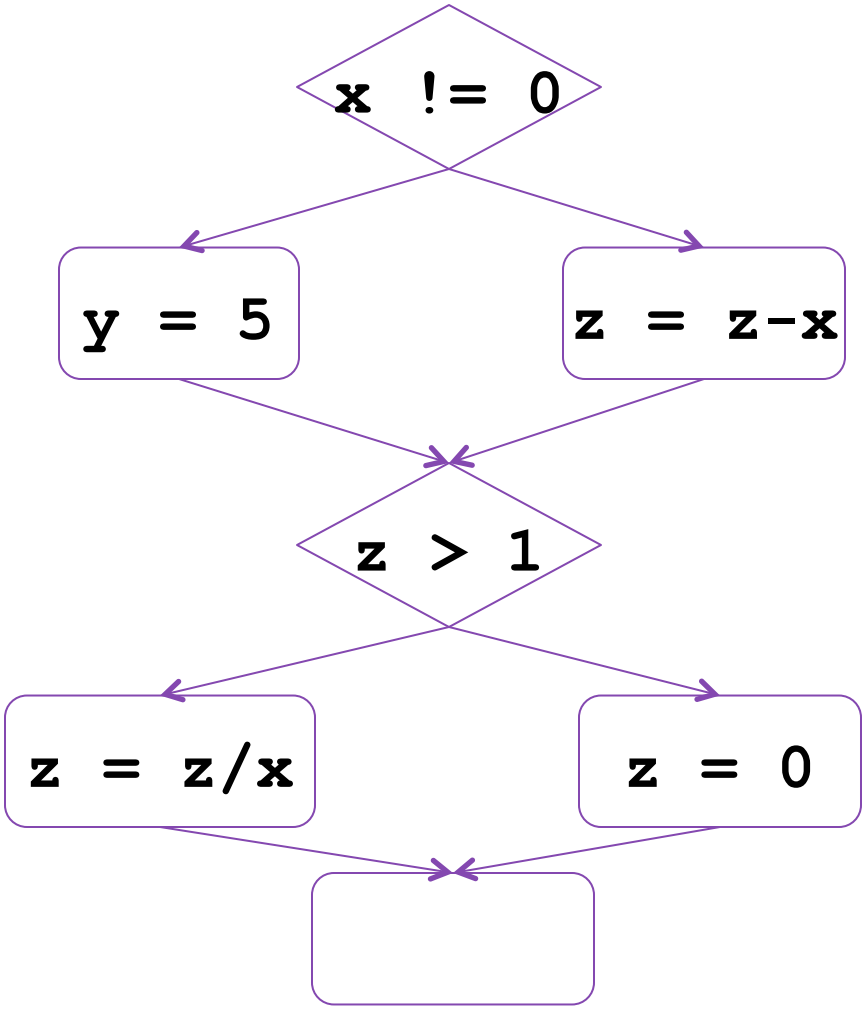
- Complex conditions can confound edge coverage
  - `if (p != NULL) and (p->left < p->right)` …
- Is this a single conditional statement in the CFG?
- How are short-circuit conditionals handled?
  - `andthen`, `orelse`

Path coverage

- Edge coverage is in some sense very static
- Edges can be covered without covering paths (sequences of edges)
  - These better model the actual execution

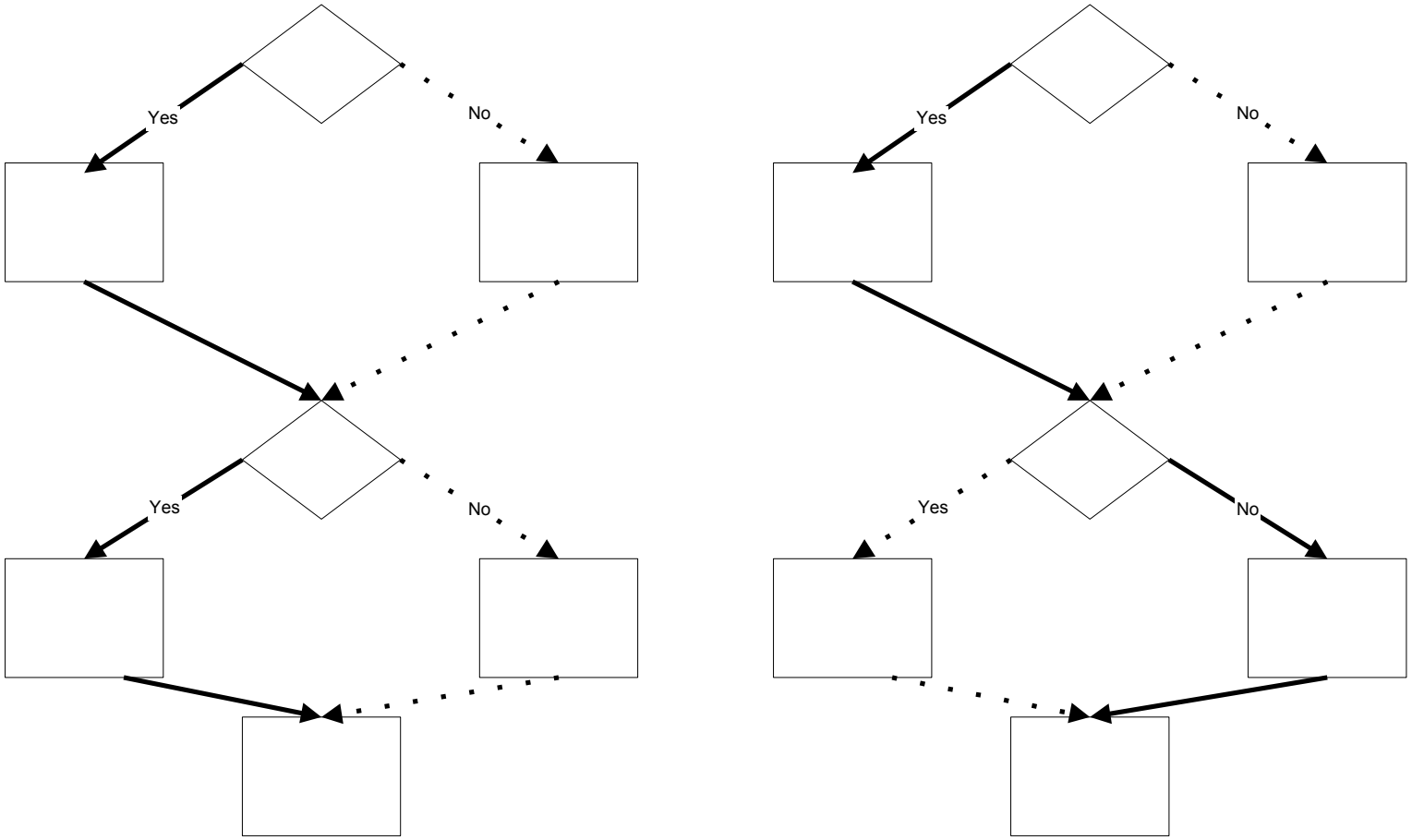
T1: {x=1,z=2}

T2: {x=0,z=-2}



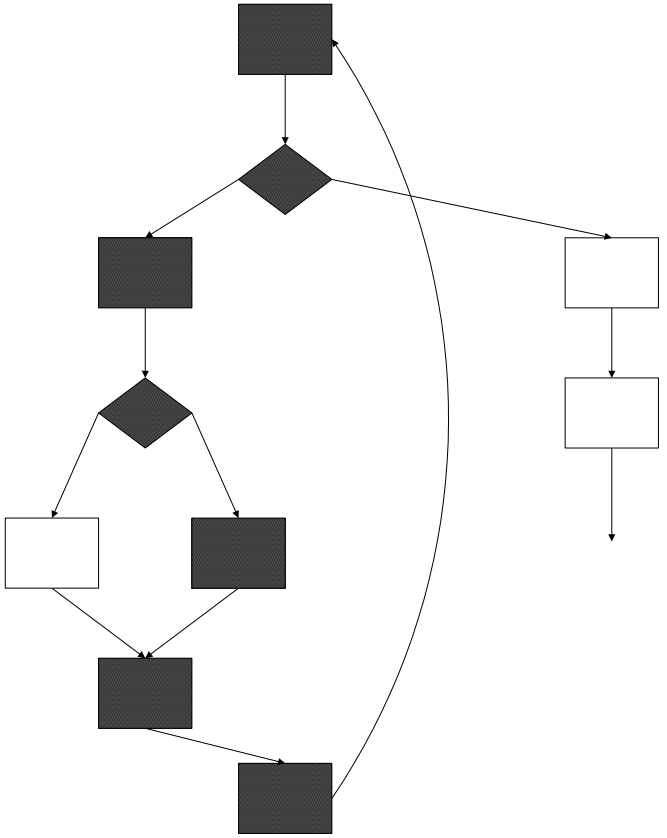

```
if (x != 0){
   y = 5
} else
   z = z-x
}
if (z > 1) {
   z = z/x
} else {
    z = 0
}
```

Example



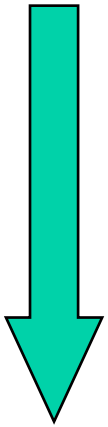
Path coverage and loops

- In general, we can't bound the number of times a loop executes
- So there are an unbounded number of paths in general
- Often in practice
  - Clear boundary conditions
  - 10



Varieties of coverage

- Statement coverage
- Branch coverage
- Decision coverage
- Loop coverage
- Condition/Decision coverage
- Path coverage



increasing number of test cases

Limitations of coverage

- 100% coverage is not always a reasonable target
- 100% may be unattainable (dead code)
- High cost to approach the limit
- Coverage is just a heuristic
- Oracles – “does it do the right thing?” – are often neglected in the face of coverage
- High-coverage can be a counter-intuitive indicator of poor code

How to increase coverage?

- Another limitation is that, beyond simple examples, it is often hard to figure out how to increase coverage – that is, what tests should be added to cover a particular statement or edge or path or such?
- How might you do this?

CSE403 ● Software engineering ● sp12

Week 5-6				
Monday	Tuesday	Wednesday	Thursday	Friday
• Testing I • No reading	• Group meetings	• Midterm	• No Section	• Testing II • Progress report due • Readings out
• Testing III • Readings due	• Group meetings	• TBA	• Section • ZFR due	• ZFR demos