Software Testing and Reliability

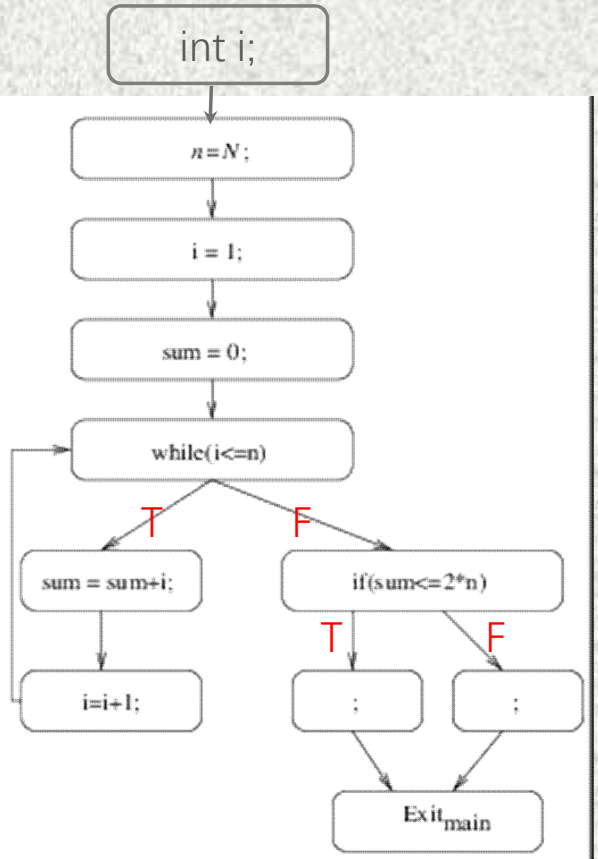
Xiaoyuan Xie 谢晓园
xxie@whu.edu.cn
计算机学院E301

Lecture 5

White-box Testing (control-flow coverage)

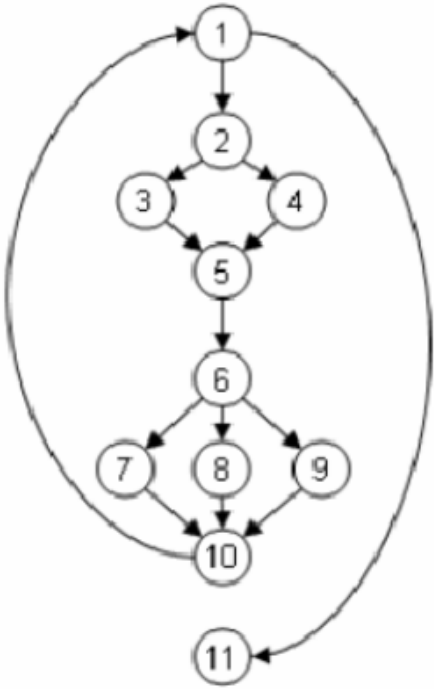
Control flow graph (example)

```
int n, sum;
main(){
  int i;
  n = N;
  i = 1;
  sum = 0;
  while ( i<=n ){
    sum = sum + i;
    i = i + 1;
  }
  if ( sum < 2*n ){
   ERROR: ;
  } else {
   ;
  }
}
```

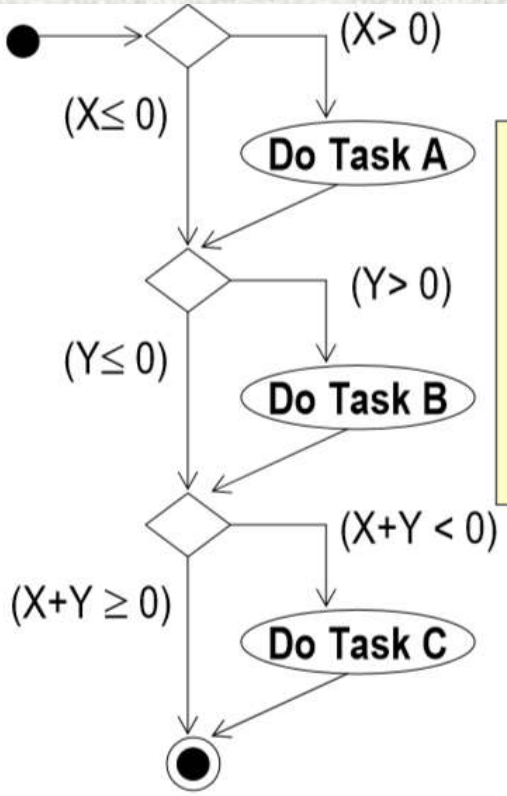


Control flow graph (example)

Node	Statement
(1)	`while(x<100){`
(2)	`  if (a[x] % 2 == 0) {`
(3)	`      parity =  0;`
	`   }`
	`   else {`
(4)	`      parity =  1;`
(5)	`   }`
(6)	`   switch(parity){`
	`      case 0:`
(7)	`        println( "a[" + i + "] is even");`
	`      case 1:`
(8)	`        println( "a[" + i + "] is odd");`
	`      default:`
(9)	`        println( "Unexpected error");`
	`    }`
(10)	`   x++;`
	`}`
(11)	`p = true;`



Basic Concepts (continued)



Path 1 (X> 0) (Y>0)(X+Y < 0)	**Infeasible**
Path 2(X> 0) (Y>0)(X+Y ≥ 0)	Feasible
Path 3(X> 0) (Y≤0)(X+Y < 0)	Feasible
Path 4(X> 0) (Y≤0)(X+Y ≥ 0)	Feasible
Path 5(X≤ 0) (Y>0)(X+Y < 0)	Feasible
Path 6(X≤ 0) (Y>0)(X+Y ≥ 0)	Feasible
Path 7(X≤ 0) (Y≤0)(X+Y < 0)	Feasible
Path 8(X≤ 0) (Y≤0)(X+Y ≥ 0)	Feasible

Coverage Testing

- Intuition?
  - -- If a certain part of the program is not covered, we cannot know whether there are faults inside that part.
- Coverage criteria
  - -- Control-flow coverage
  - -- Data-flow coverage

Procedure of coverage testing

1. Given a coverage criterion, list out all the items to be covered

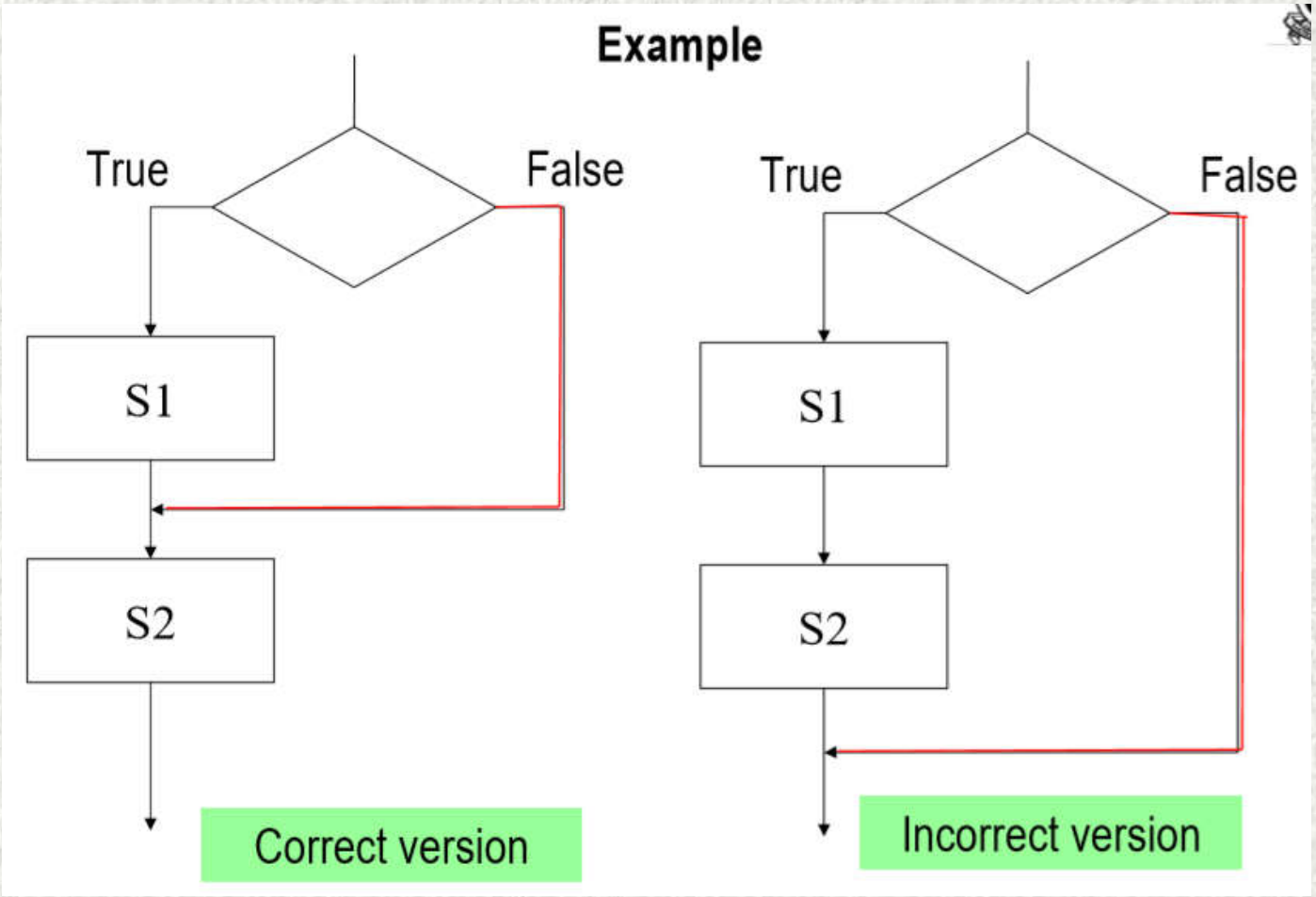
2. Find test cases to execute each item at least once

Control-flow Coverage

- Statement coverage
  - -- Execute every statement at least once
    - -- For example, cover all nodes (1, 2, …, 18) in Slide 4

- Branch coverage
  - -- Execute every branch at least once
    - -- For example, cover all branches (b, d, f, h, k, q, l, m, s, t, wand r) in Slide 4
      - -- Imply coverage of all edges
  - -- **Imply statement coverage**

Control-flow Coverage (continued)



Control-flow Coverage (continued)

In this case,

- statement coverage testing **only** covers the true branch
- branch coverage testing **covers** all statements and all branches
- Branch coverage can detect the fault, but the statement coverage cannot

Control-flow Coverage (continued)

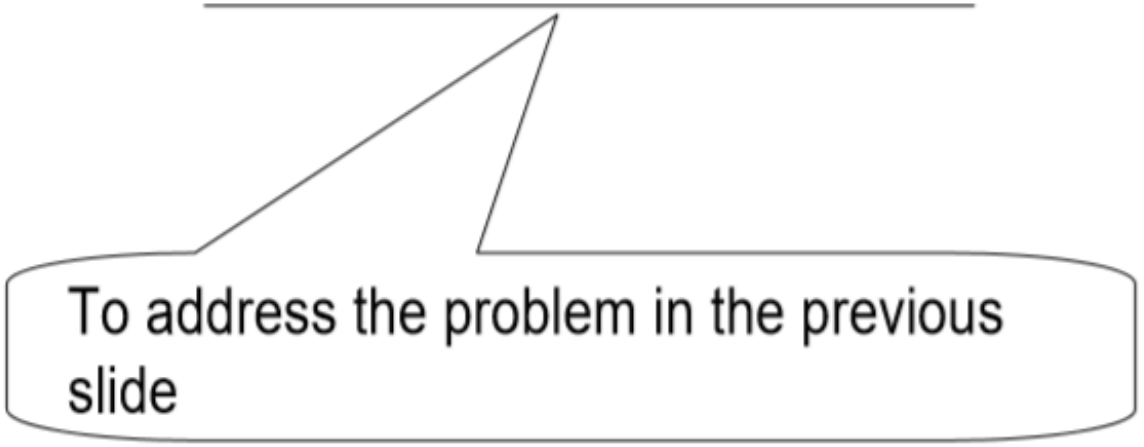
- Condition coverage
  - Execute all possible outcomes (TRUE or FALSE) of every condition in a decision at least once
  - **Do not imply branch coverage**
  - Example: WHILE (x < 10 OR x > 200) DO

Once reaching 100% condition coverage, other test cases will be ignored. As a result, the FALSE branch will not be tested

Test input value	Condition 1 x < 10	Condition 2 x > 200	Branch x < 10 OR x > 200
x < 10	TRUE √	FALSE √	TRUE
x > 200	FALSE √	TRUE √	TRUE
200 >= x >=10	FALSE	FALSE	FALSE

Control-flow Coverage (continued)

- Decision-condition coverage
  - Execute all possible outcomes of every condition in a decision at least once
  - Execute all possible outcomes of a decision at least once



Control-flow Coverage (continued)

- Multiple-condition coverage

  -- Execute all possible combinations of condition outcomes in a decision

  -- Remark: Some combinations may never be satisfied, for example, we cannot have (x < 10 = TRUE) and (x > 200 = TRUE) both condition outcomes at the same time

  --- with respect to the same variable

  -- **Imply decision-condition coverage**

Control-flow Coverage (continued)

- Path coverage
  - -- Execute every path at least once
  - -- **Imply all the coverage criteria**
  - -- Problem
    - -- When the software contains loops, it has infinite number of paths – then 100% path coverage becomes infeasible.
    - -- Solution
      - -- Group paths into equivalence classes (see the next slide)

Equivalence classes of paths

- Two paths are considered equivalent if they differ only in the number of iterations
- Two classes
  - -- **Zero** loop traversal
  - -- **At least one** loop traversals

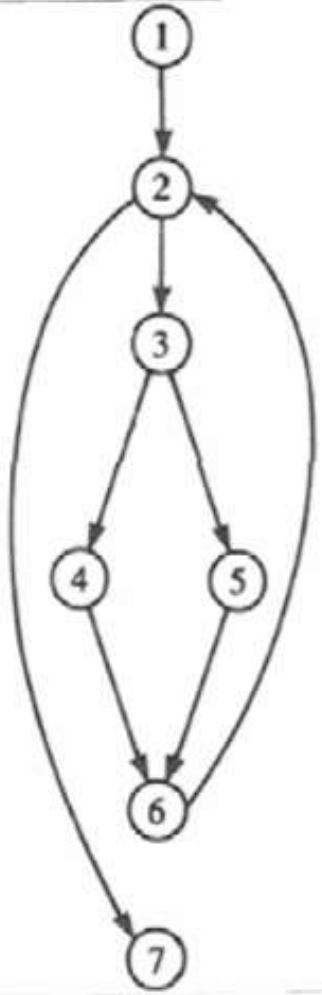
Coverage for equivalence classes of paths

Some notations

- · (dot)
  -- represents sequences
- **+** (plus)
  -- represents selections
- ***** (star)
  -- represents iterations
- **0** (zero)
  -- represents NULL

Coverage for equivalence classes of paths (continued)

```
      .
1     .
2     while (condition1) do
      begin
3        if (condition2) then
            .
4           .
         else
            .
5           .
6     end
      .
7     .
```



while loop example

$1 \cdot 2 \cdot (3 \cdot (4+5) \cdot 6 \cdot 2)^* \cdot 7$

$1 \cdot 2 \cdot ((3 \cdot (4+5) \cdot 6 \cdot 2)+0) \cdot 7$

Translate each digit into 1

$1 \cdot 1 \cdot ((1 \cdot (1+1) \cdot 1 \cdot 1)+1) \cdot 1$

3 paths

to be covered

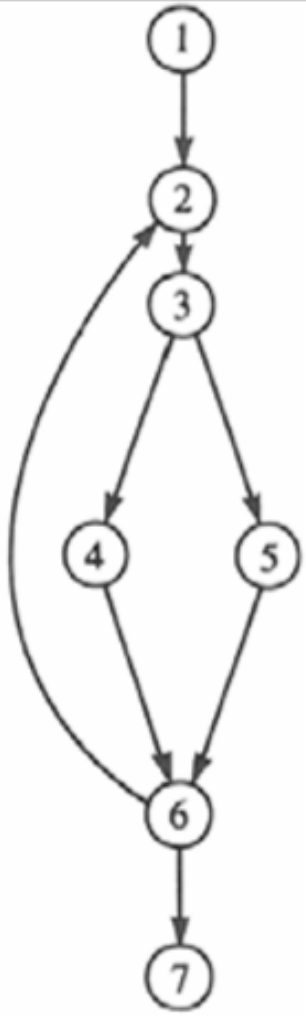
Coverage for equivalence classes of paths (continued)

3 paths for the above while loop example

1. 1–2–7
2. 1–2–3–4–6–2–7
3. 1–2–3–5–6–2–7

Coverage for equivalence classes of paths (continued)

```
      .
1     .
2     repeat
3        if (condition2) then
               .
4              .
         else
               .
5              .
6     until (condition1)
      .
7     .
```



repeat loop example

$1 \cdot 2 \cdot 3 \cdot (4+5) \cdot 6(\cdot 2 \cdot 3 \cdot (4+5) \cdot 6)^* \cdot 7$

⇩

$1 \cdot 2 \cdot 3 \cdot (4+5) \cdot 6((\cdot 2 \cdot 3 \cdot (4+5) \cdot 6)+0) \cdot 7$

⇩

$1 \cdot 1 \cdot 1 \cdot (1+1) \cdot 1((\cdot 1 \cdot 1 \cdot (1+1) \cdot 1)+1) \cdot 1$

⇩

6 paths

to be covered

Coverage for equivalence classes of paths (continued)

6 paths for the above repeat loop example

1. 1–2–3–4–6–7
2. 1–2–3–5–6–7
3. 1–2–3–4–6–2–3–4–6–7
4. 1–2–3–4–6–2–3–5–6–7
5. 1–2–3–5–6–2–3–4–6–7
6. 1–2–3–5–6–2–3–5–6–7