



KIỂM THỬ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

INT3117

Bài giảng 04: White-box testing (I)

Câu hỏi (10')

- Áp dụng kỹ thuật kiểm thử phân lớp tương đương và phân tích giá trị biên đơn giản cho hàm sau đây:

Record createRecord(Person p)

- Class Person bao gồm các thuộc tính sau:
 - name: String
 - age: Integer
 - gender: String (“male”, “female”, “other”)
 - employeeStatus: Boolean (true/false)

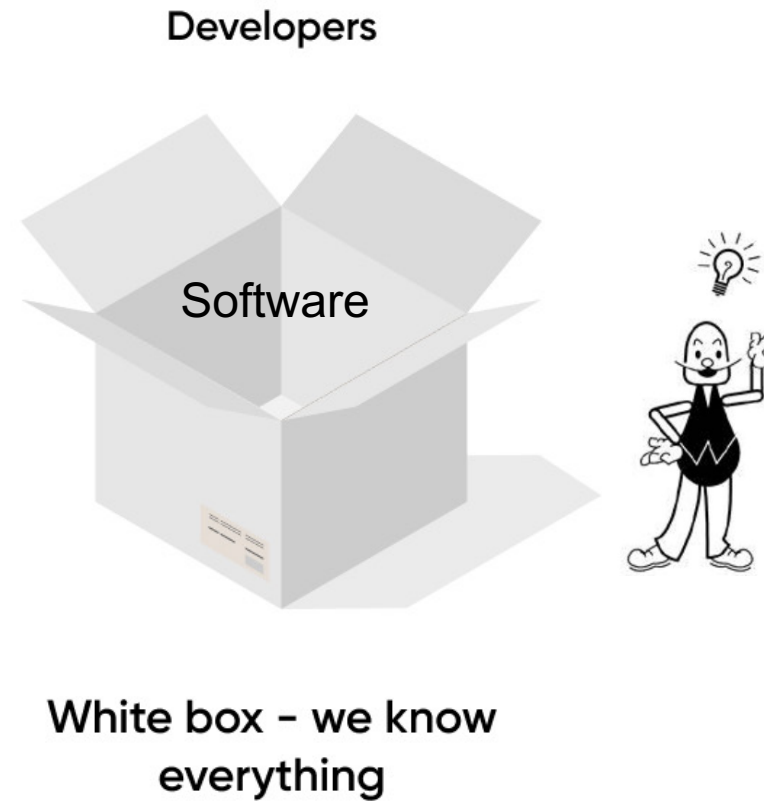
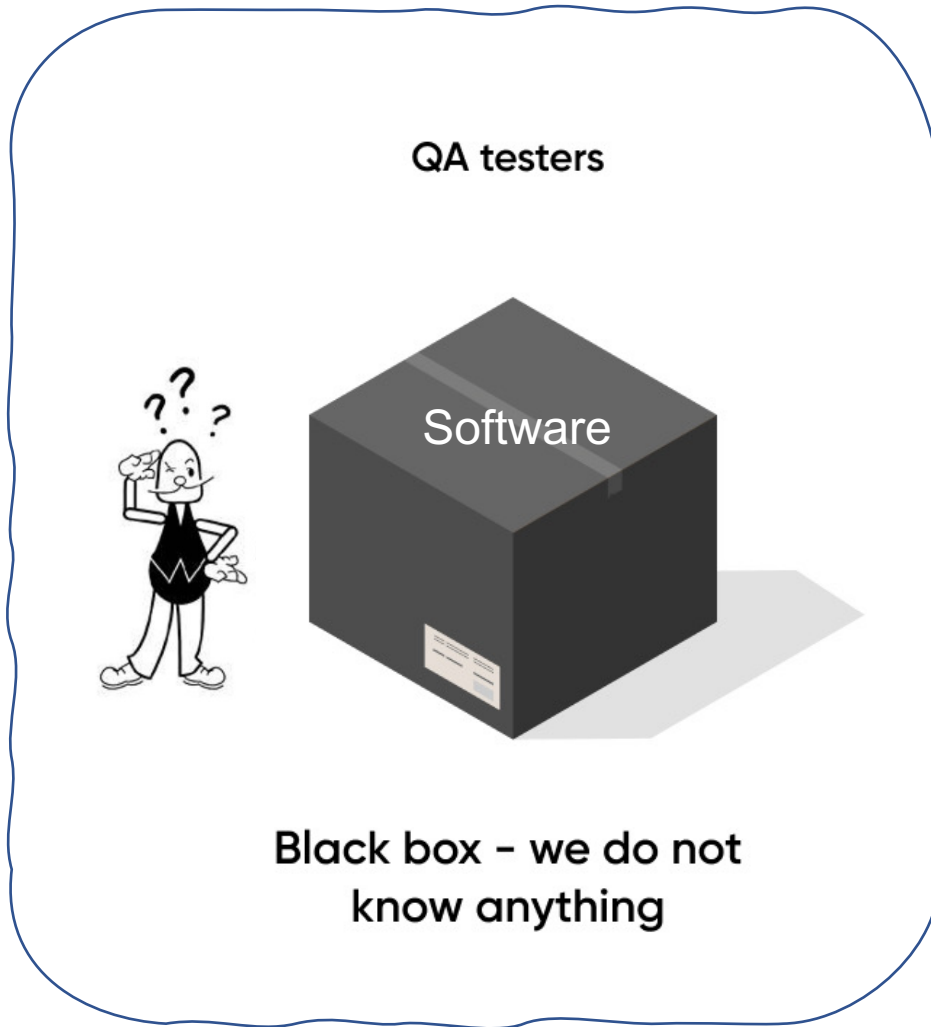
Chia nhóm & chọn công cụ kiểm thử tự động

- Đăng ký thông qua lớp trưởng, lớp trưởng tổng hợp và gửi lại cho GV
- Tiêu chí đánh giá dự án:
 - Mục đích của công cụ (kiểm thử khía cạnh nào của phần mềm) (20)
 - Ý tưởng của công cụ (20)
 - Cách thức xây dựng/hoạt động của công cụ (30)
 - Điểm mạnh/yếu của công cụ với các đối thủ (10)
 - Demo (Video) (20)
 - Điểm cho nhóm = điểm chung * Số thành viên
 - Thành viên nhóm tự đánh giá điểm của nhau sau khi nhận điểm từ GV

Chính sách chuyển điểm

- 5 – 10/100 điểm sẽ được chuyển từ nhóm A sang nhóm B khi:
 - **A** trình bày một ý **X**, B không đồng tình với **A** về **X**
 - **B** chỉ ra được bằng chứng về **X** là không đúng, trong khi **A** không có bằng chứng về **X** là đúng
 - Lời khuyên: Nên tìm hiểu kỹ càng và có backup cho các nhận định và đánh giá về công cụ đang tìm hiểu để tối thiểu khả năng bị trừ điểm
- Mỗi nhóm thường sẽ có 1-n nhóm tìm hiểu công cụ đối thủ/tương đương
 - Lời khuyên: Nên tìm hiểu các công cụ đối thủ để tối đa hóa khả năng có thêm điểm
- Mục tiêu của chính sách chuyển điểm nhằm khuyến khích các nhóm có trách nhiệm hơn với các nhận định được đưa ra khi trình bày

Nội dung trước

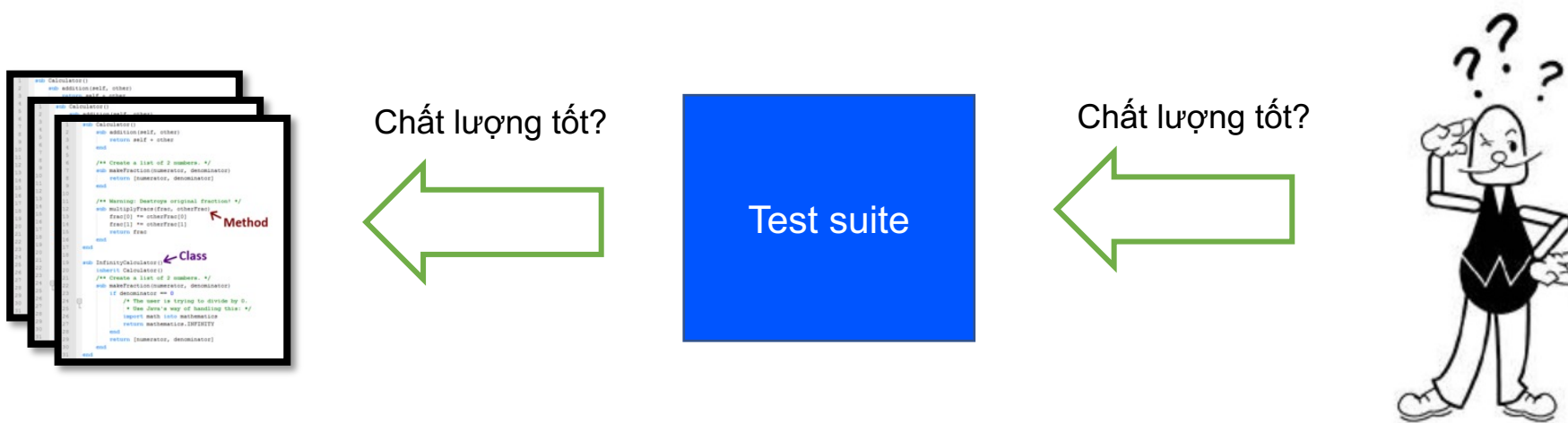


Hôm nay...

- Các kỹ thuật sinh test white-box (white-box testing)
 - Dòng điều khiển (Control-flow)
 - Phủ dòng lệnh (Statement coverage)
 - Phủ nhánh (Branch coverage)
 - Phủ đường (Path coverage)
 - Công cụ tính độ phủ: EclEmma/ OpenClover (Tự tìm hiểu)

Độ phủ mã nguồn (code coverage)

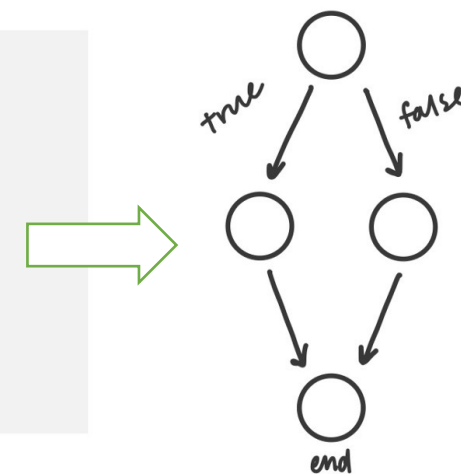
- Đánh giá chất lượng bộ test như thế nào?
- Bộ test phủ/chạy càng nhiều mã nguồn có thể có chất lượng tốt hơn
- Định hướng việc sinh test



Tính độ phủ mã nguồn như thế nào?

- Biểu diễn chương trình thành một đồ thị:
 - Thường là đồ thị dòng điều khiển, Control Flow Graph (CFG)
 - Phủ đỉnh (node coverage): Thực thi toàn bộ dòng lệnh
 - Phủ cạnh (edge coverage): Thực thi toàn bộ nhánh

```
num = 10
if num % 2 == 0:
    print("even")
else:
    print("odd")
```

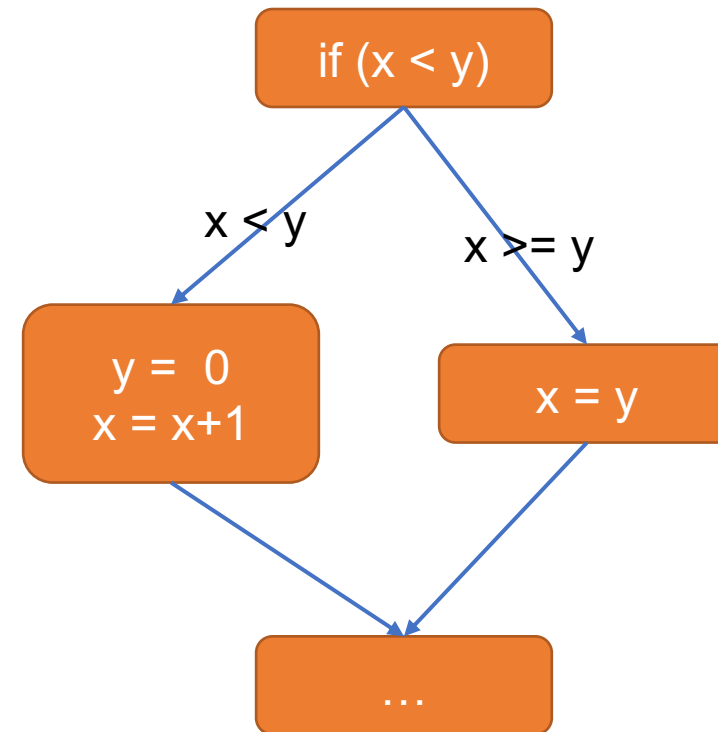


Đồ thị dòng điều khiển (CFG)

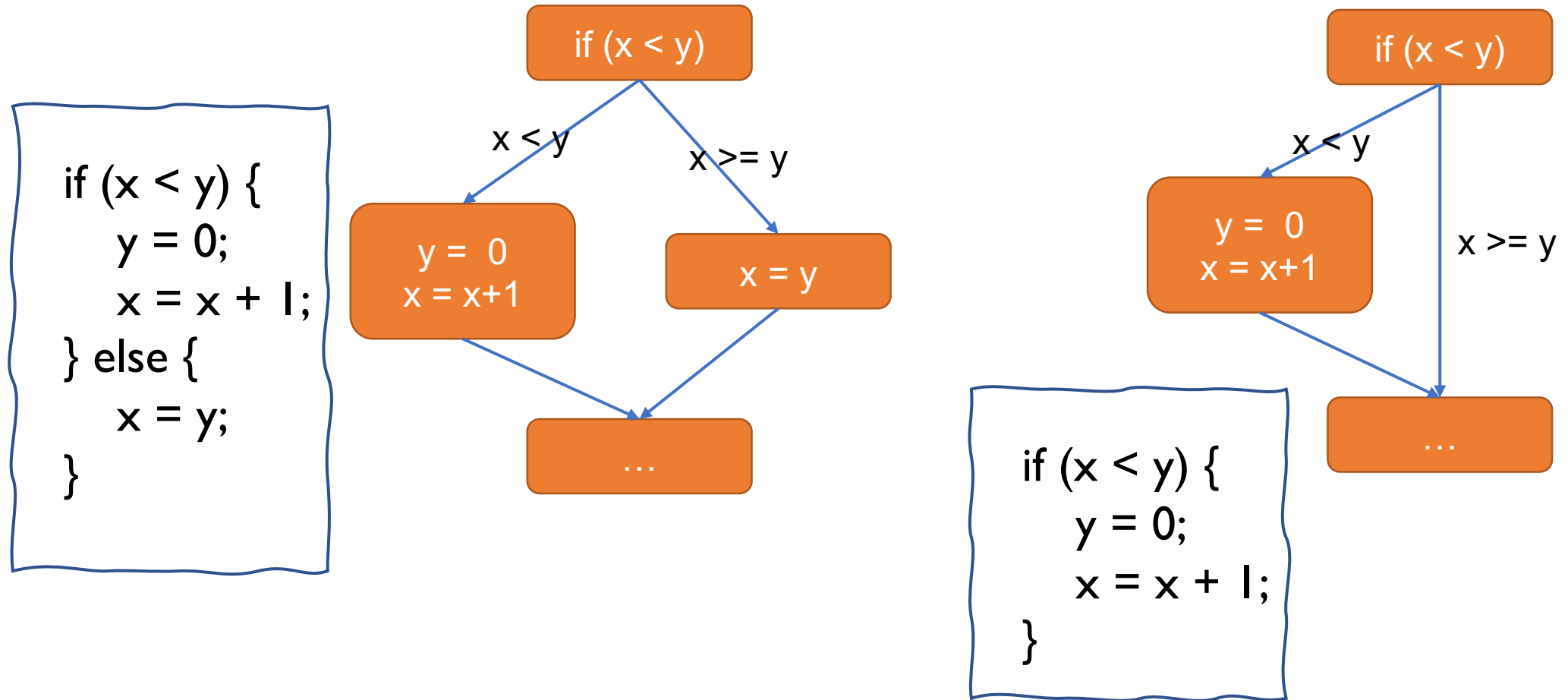
- Đồ thị dòng điều khiển (CFG) mô hình tất cả việc thực thi của mã nguồn bằng việc mô tả cấu trúc điều khiển:
 - Đỉnh (node): Một chuỗi dòng lệnh block (có thể chỉ bao gồm 1 dòng lệnh)
 - Cạnh (edge): Chuyển đổi điều khiển

CFG: Dòng lệnh if

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

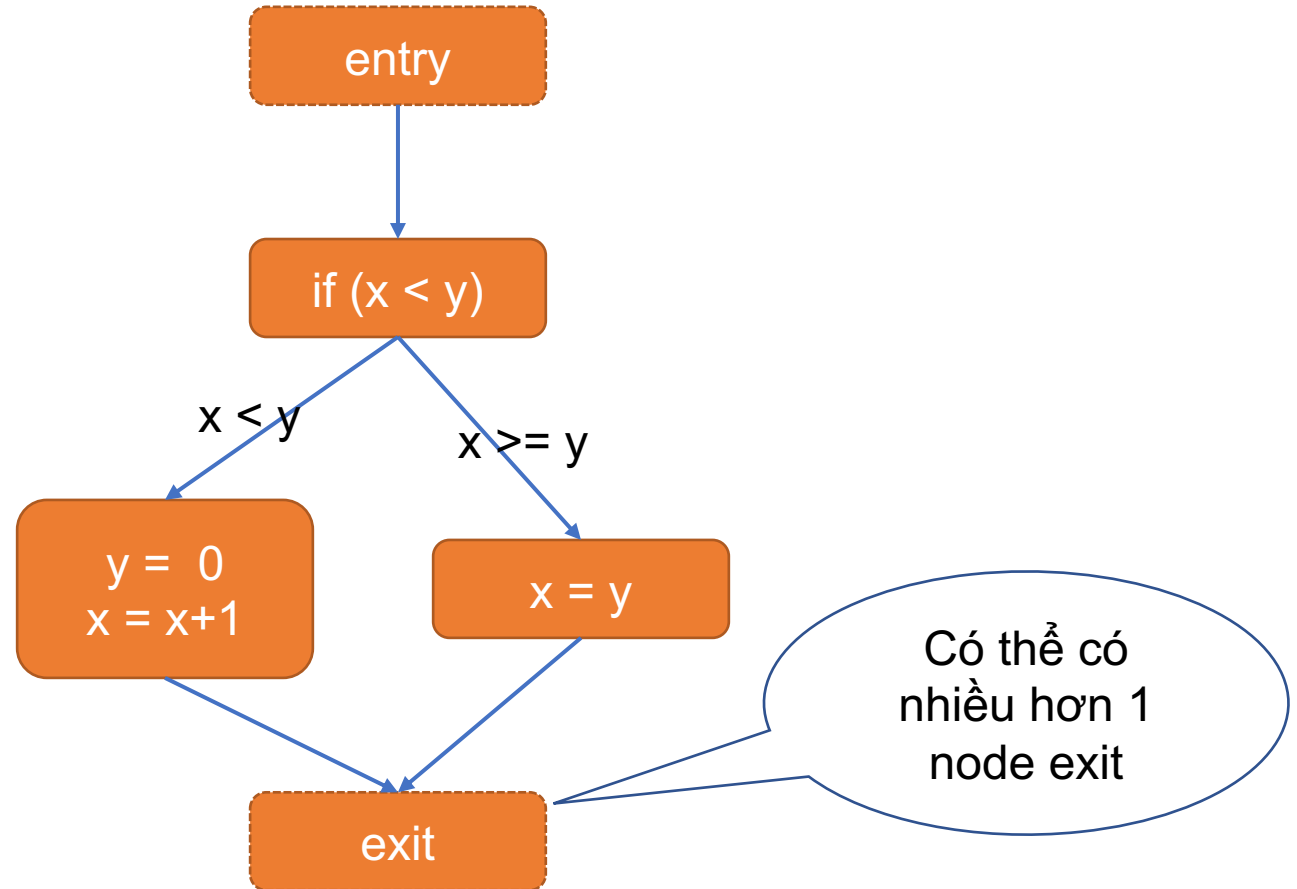


CFG: Dòng lệnh if



CFG: Dòng lệnh if

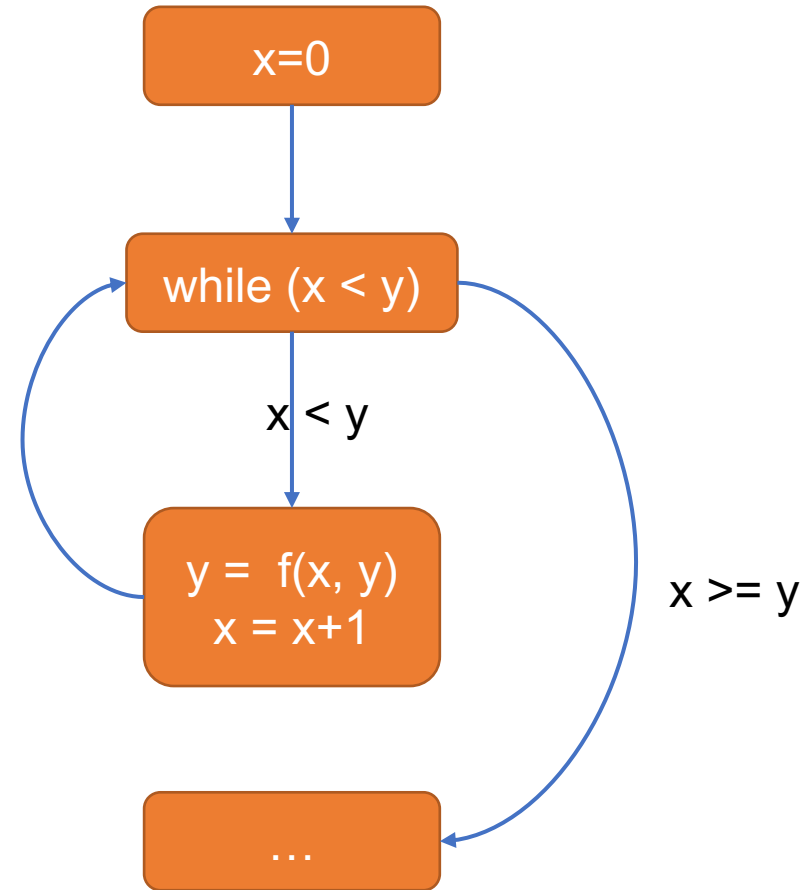
```
if (x < y) {  
    y = 0;  
    x = x + 1;  
} else {  
    x = y;  
}  
return;
```



CFG: Dòng lệnh while

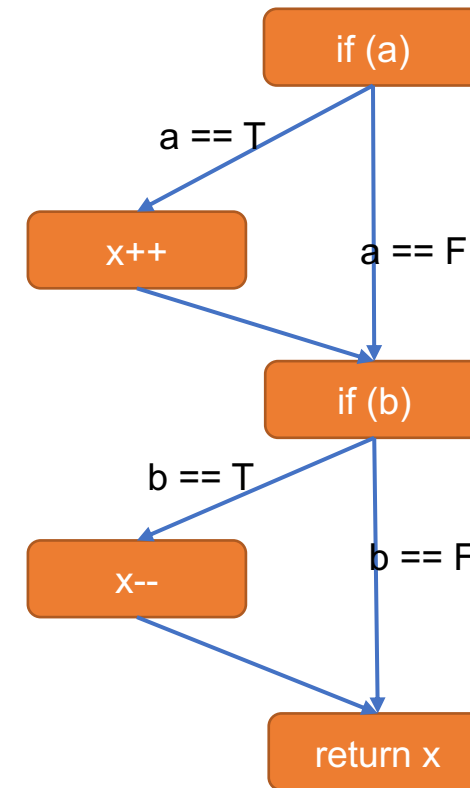
```
x=0;  
while (x < y) {  
    y = f(x, y);  
    x = x + 1;  
}  
...
```

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



Độ phủ dự trên CFG

```
public int testMe(int x, boolean a,  
boolean b){  
    if(a)  
        x++;  
    if(b)  
        x--;  
    return x;  
}
```



Độ phủ dựa trên CFG: Ví dụ

```
public class JUnitStatementCov {  
    CFGCoverageExample tester;  
    @Before  
    public void initialize() {  
        tester = new CFGCoverageExample();  
    }  
    @Test  
    public void testCase() {  
        assertEquals(1, tester.testMe(0, true, false));  
    }  
}
```



Test case này
tốt như nào????

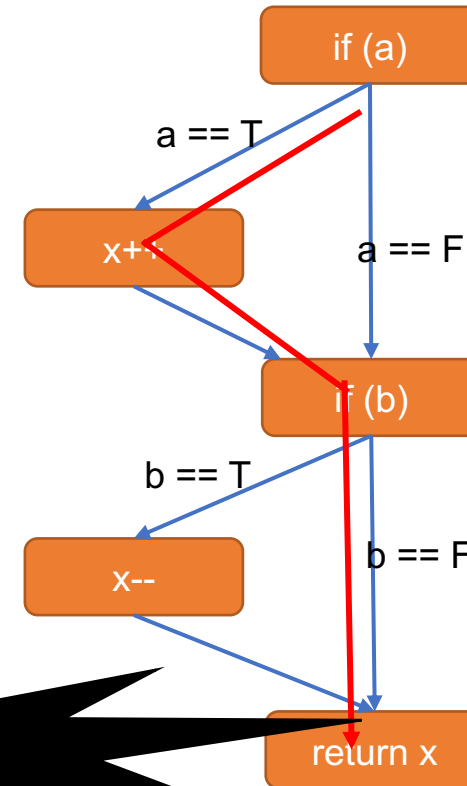
Độ phủ dự trên CFG: Phủ dòng lệnh

```
public int testMe(int x, boolean a,  
boolean b){  
    if(a)  
        x++;  
    if(b)  
        x--;  
    return x;  
}
```

tester.testMe(1, true, false)



x=1, a = T, b = F



**Phủ 4/5 (80%) dòng
lệnh**

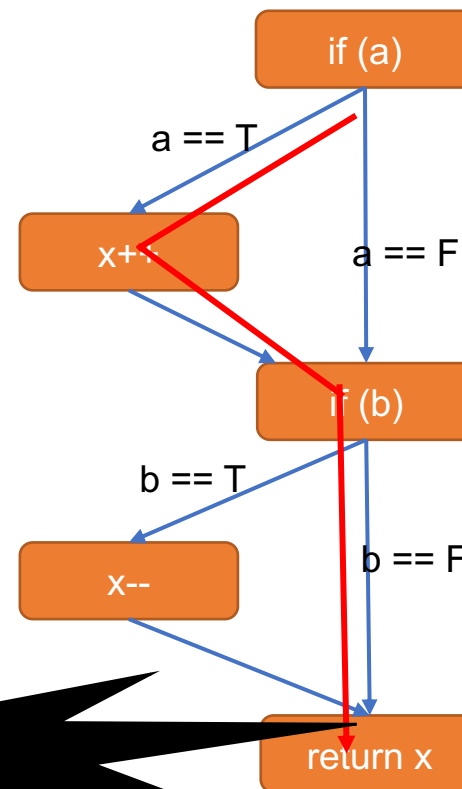
Độ phủ dựa trên CFG: Phủ nhánh

```
public int testMe(int x, boolean a,  
boolean b){  
    if(a)  
        x++;  
    if(b)  
        x--;  
    return x;  
}
```

tester.testMe(1, true, false)

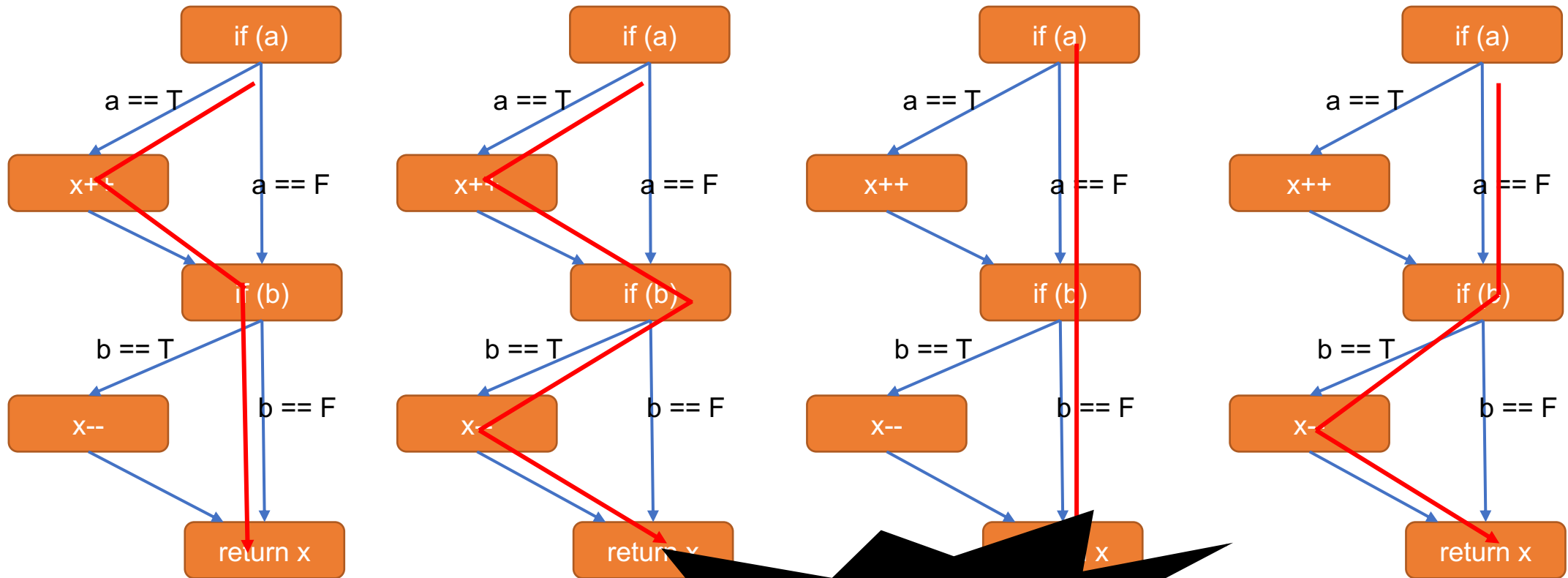


x=1, a = T, b = F



Phủ 2/4 (50%) nhánh

Độ phủ dự trên CFG: Phủ đường



Phủ 1/4 (25%) đường

Độ phủ dựa trên CFG: Ví dụ

```
public class JUnitStatementCov {  
    CFGCoverageExample tester;  
    @Before  
    public void initialize() {  
        tester = new CFGCoverageExample();  
    }  
    @Test  
    public void testCase() {  
        assertEquals(1, tester.testMe(0, true, false));  
    }  
}
```



Độ phủ dựa trên CFG: Ví dụ

```
public class JUnitStatementCov {  
    CFGCoverageExample tester;  
    @Before  
    public void initialize() {  
        tester = new CFGCoverageExample();  
    }  
    @Test  
    public void testCase() {  
        assertEquals(1, tester.testMe(0, true, false));  
    }  
}
```

Phủ dòng lệnh: 80%

Phủ nhánh: 50%

Phủ đường: 25%

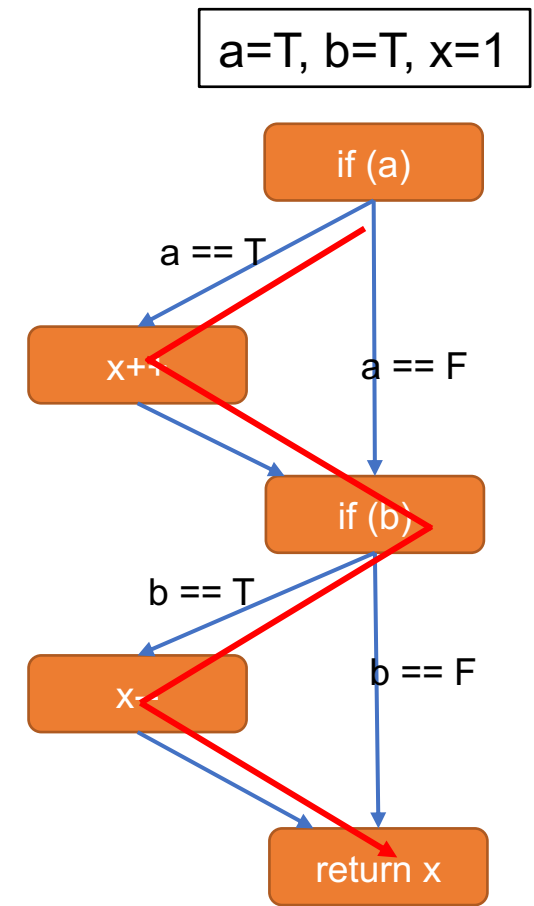
Nếu phủ nhánh là 100%, phủ dòng lệnh có đạt 100% không?

Nếu phủ đường là 100%, phủ nhánh có đạt 100% không?

Phủ dòng lệnh vs. Phủ nhánh

- Khi phủ nhánh đạt 100%, phủ dòng lệnh sẽ đạt 100%:
 - Có 2 loại dòng lệnh: ở nhánh hoặc không ở nhánh
 - Dòng lệnh ở nhánh sẽ được phủ bởi ít nhất 1 test
 - Dòng lệnh không ở nhánh thì sẽ phủ bởi bất kỳ test nào
- Khi phủ dòng lệnh đạt 100%, phủ nhánh có đạt 100%?

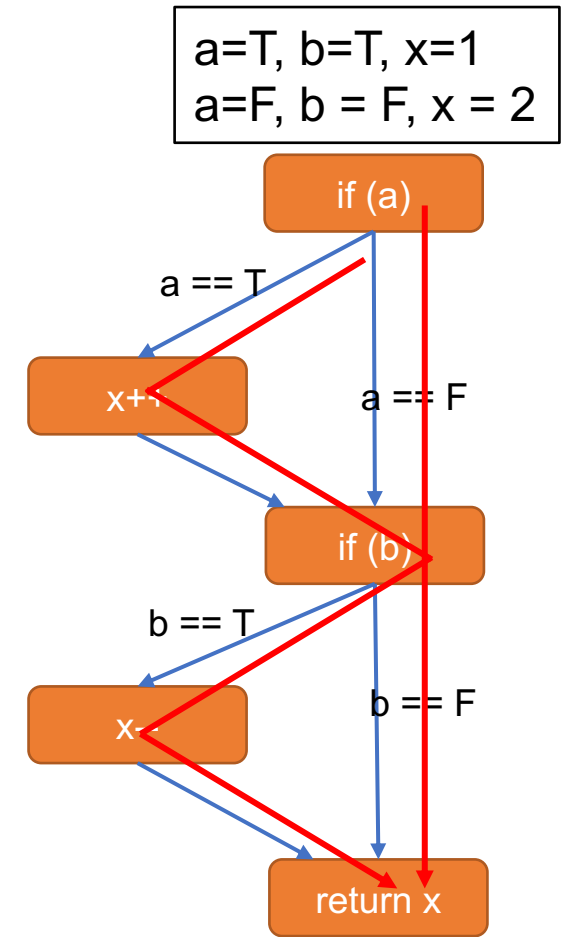
Phủ nhánh chặt hơn phủ dòng lệnh



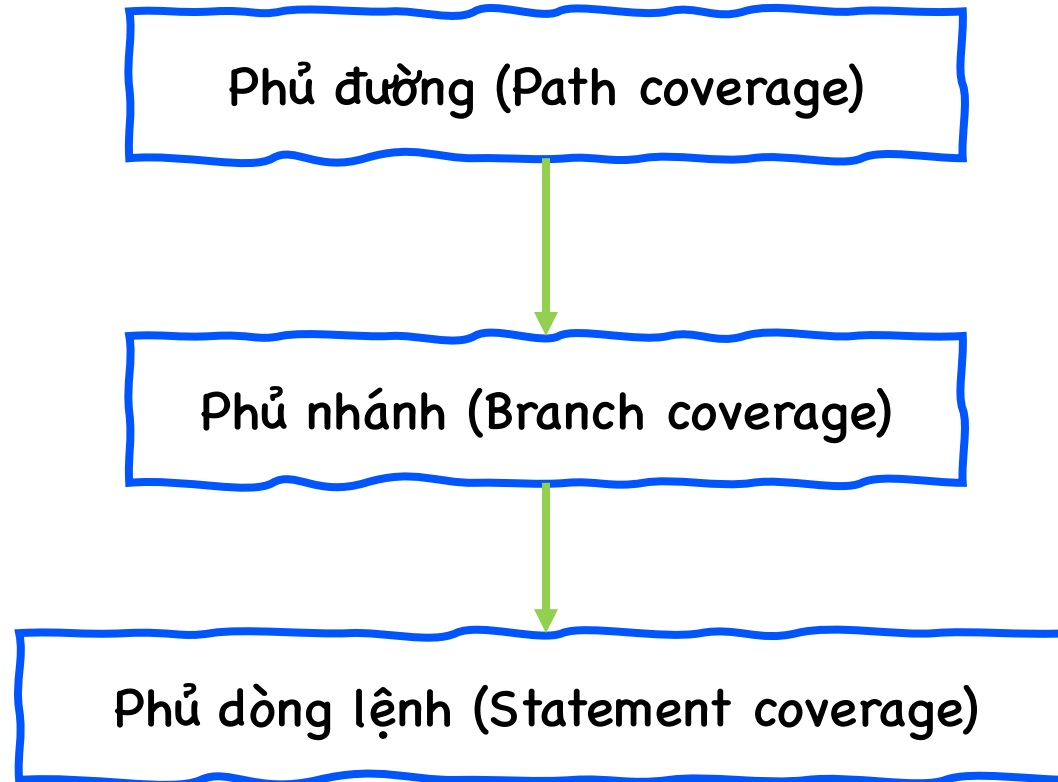
Phủ nhánh vs. Phủ đường

- Khi phủ đường đạt 100%, phủ nhánh sẽ đạt 100%:
 - Phủ đường là phủ tất cả tổ hợp của nhánh
 - Phủ nhánh 100%
- Khi nhánh đạt 100%, phủ đường có đạt 100%?

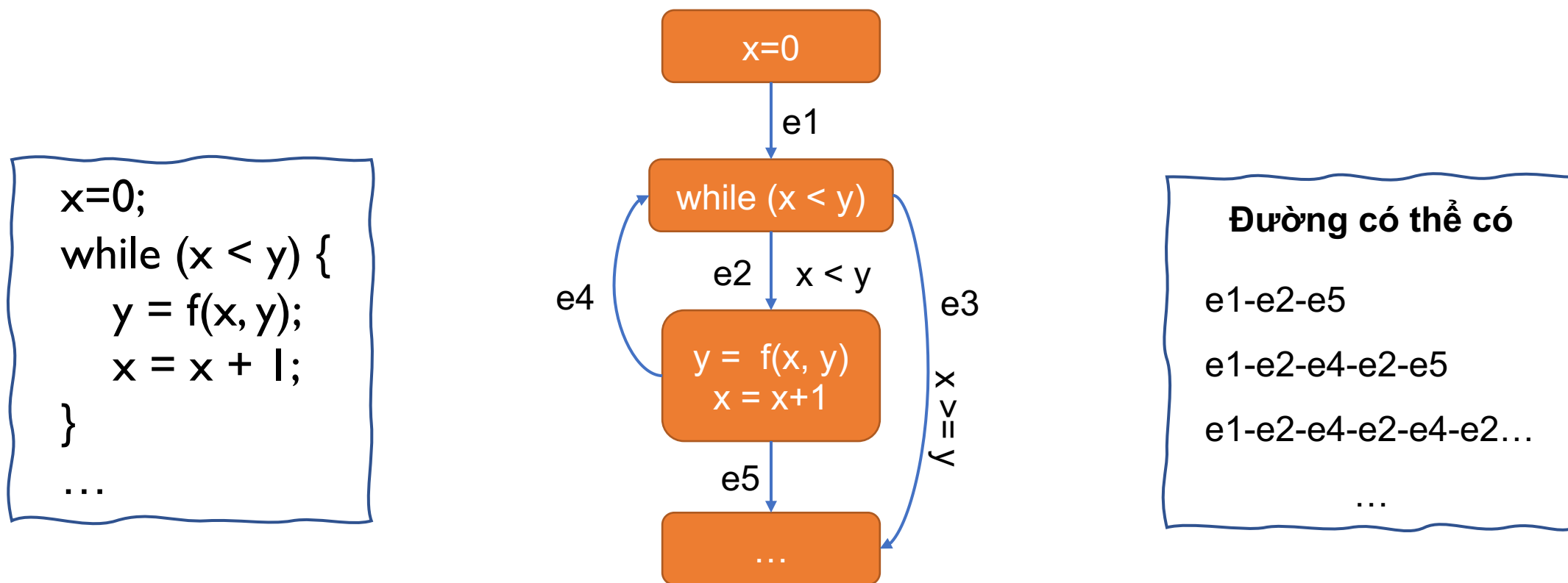
Phủ đường chặt hơn phủ dòng nhánh



Độ phủ dựa trên CFG: So sánh



Hay là luôn dùng phủ đường?



Phủ đường thường bất khả thi với chương trình thực tế!

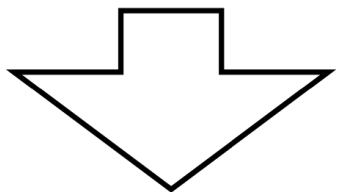
Độ phủ dựa trên CFG: Tính hiệu quả

- 65% bugs được tìm bằng kiểm thử đơn vị
- Kiểm thử đơn vị chủ yếu sử dụng kiểm thử dựa trên độ phủ CFG
- Kiểm thử dựa trên CFG chủ yếu hướng tới phủ dòng lệnh và nhánh


Độ phủ dựa trên CFG: Hạn chế

- Bất kỳ độ phủ nào 100% cũng không đảm bảo được khả năng phát hiện 100% lỗi trong chương trình

Test: assertEquals(1, sum(1,0))



```
public int sum(int x, int y){  
    return x-y; //should be x+y  
}
```



Statement coverage: 100%

Branch coverage: 100%

Path coverage: 100%

Thất bại trong việc phát hiện lỗi!



Bài tập

- Vẽ CFG cho chương trình bên
- Sinh bộ test để đảm bảo
 - Phủ 100% dòng lệnh, tính độ phủ nhánh
 - Sinh thêm tests để phủ 100% nhánh

```
1 public static int findIndex(int[] arr, int target) {  
2     int index = 0;  
3     int result = -1;  
4     while (index < arr.length) {  
5         if (arr[index] == target) {  
6             result = index;  
7             break;  
8         }  
9         if (arr[index] > target) {  
10            System.out.println("Number is greater than target");  
11        } else {  
12            System.out.println("Number is smaller than target");  
13        }  
14        index++;  
15    }  
16    return result;  
17 }
```

Buổi học sau

- Các kỹ thuật kiểm thử hộp trắng – Tập 2

