Chương 3:
Các Kỹ thuật kiểm thử phần mềm

Khoa Công nghệ thông tin

Trường Đại học Nguyễn Tất Thành

Nội dung

1/. Black Box Testing

2/. White Box Testing

3/. Grey Box Testing

4/. Các công cụ kiểm thử

5/. Kết hợp các kỹ thuật trong kiểm thử

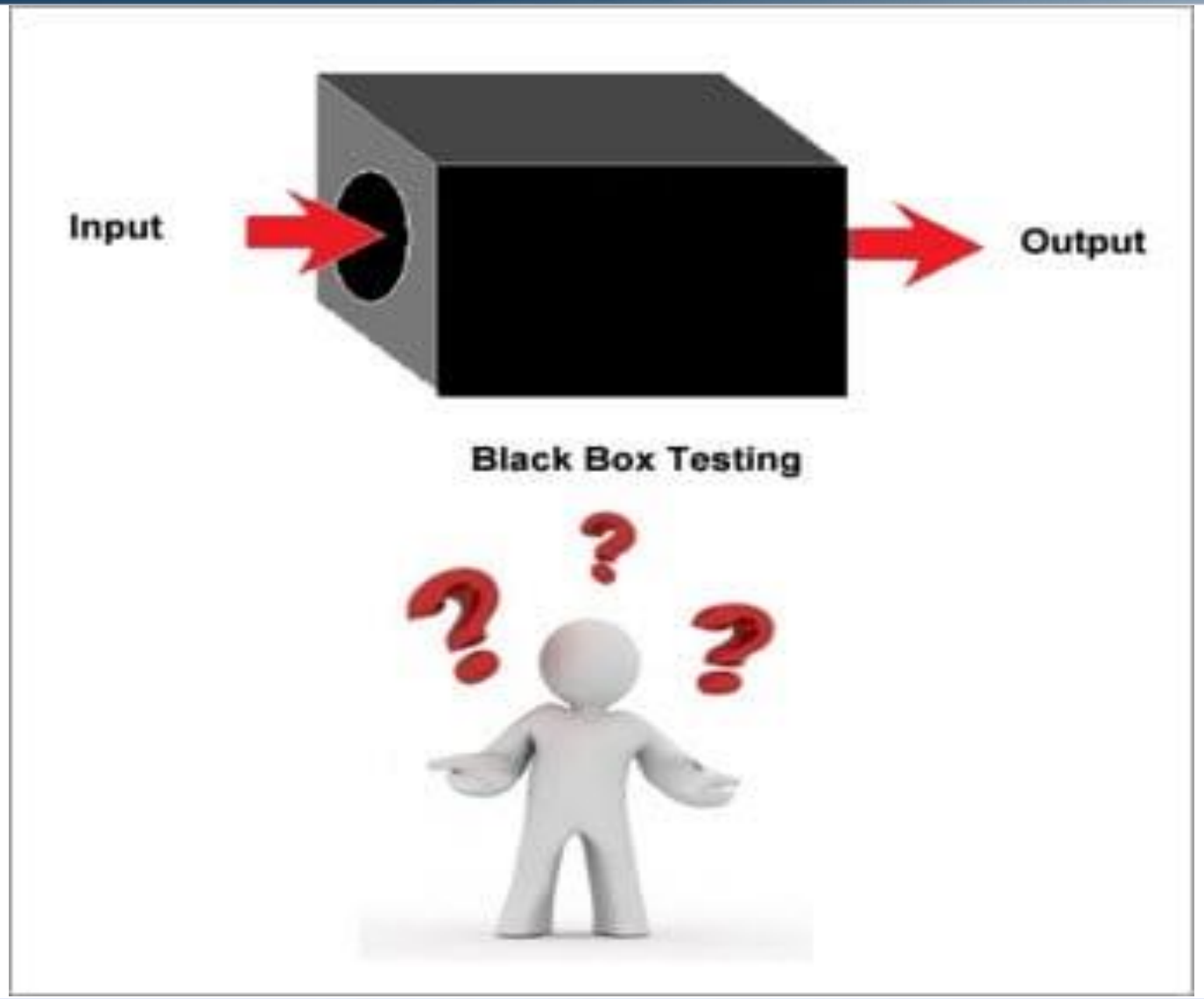
1. Black box Testing

1.1/. Giới thiệu:

Kiểm thử hộp đen còn được gọi là kiểm tra hộp mờ, hộp kín, kiểm tra dựa trên đặc điểm kỹ thuật hoặc bằng mắt.

Đây là phương pháp Kiểm thử phần mềm, phân tích chức năng của phần mềm / ứng dụng mà không biết nhiều về cấu trúc / thiết kế bên trong của mục đang được kiểm tra, phương pháp này thực hiện so sánh giá trị đầu vào với giá trị đầu ra để có nhận định đúng sai.

1. Black box Testing



1. Black box Testing

1.2/. Các kiểu kiểm thử Black box: Có 2 loại sau:

1.2.1/. Functional Testing:

Loại này liên quan đến các yêu cầu chức năng hoặc thông số kỹ thuật của một ứng dụng. Ở đây, các hành động hoặc chức năng khác nhau của hệ thống đang được kiểm tra bằng cách cung cấp các giá trị đầu vào và so sánh các giá trị đầu ra thực tế (với đầu ra dự kiến).

1. Black box Testing

Ví dụ: khi chúng ta kiểm tra danh sách thả xuống (Dropdown List), chúng ta click vào nó và xác minh rằng nó sẽ được mở ra và tất cả các giá trị dự kiến sẽ hiển thị trong danh sách.

1. Black box Testing

1.2.2/. Non-Functional Testing:

- Ngoài các chức năng của các yêu cầu, còn có một số khía cạnh phi chức năng cũng được yêu cầu kiểm thử để cải thiện chất lượng và hiệu suất của ứng dụng.
- Một số loại kiểm thử phi chức năng chính bao gồm:
  - Usability Testing (kiểm thử khả năng sử dụng); Load Testing (kiểm thử tải); Performance Testing (kiểm thử năng suất); Compatibility Testing (kiểm thử tương thích); Stress Testing (kiểm thử áp lực); Scalability Testing (kiểm thử khả năng mở rộng).

1. Black box Testing

1.3/. Các kỹ thuật kiểm thử hộp đen (Black box testing):

Bao gồm các kỹ thuật sau:

- Equivalence Class Partitioning
- Boundary Value Analysis
- Decision Table Testing
- Cause Effect Graph
- Error Guessing
- Comparison Testing
- State Transition Testing

1. Black box Testing

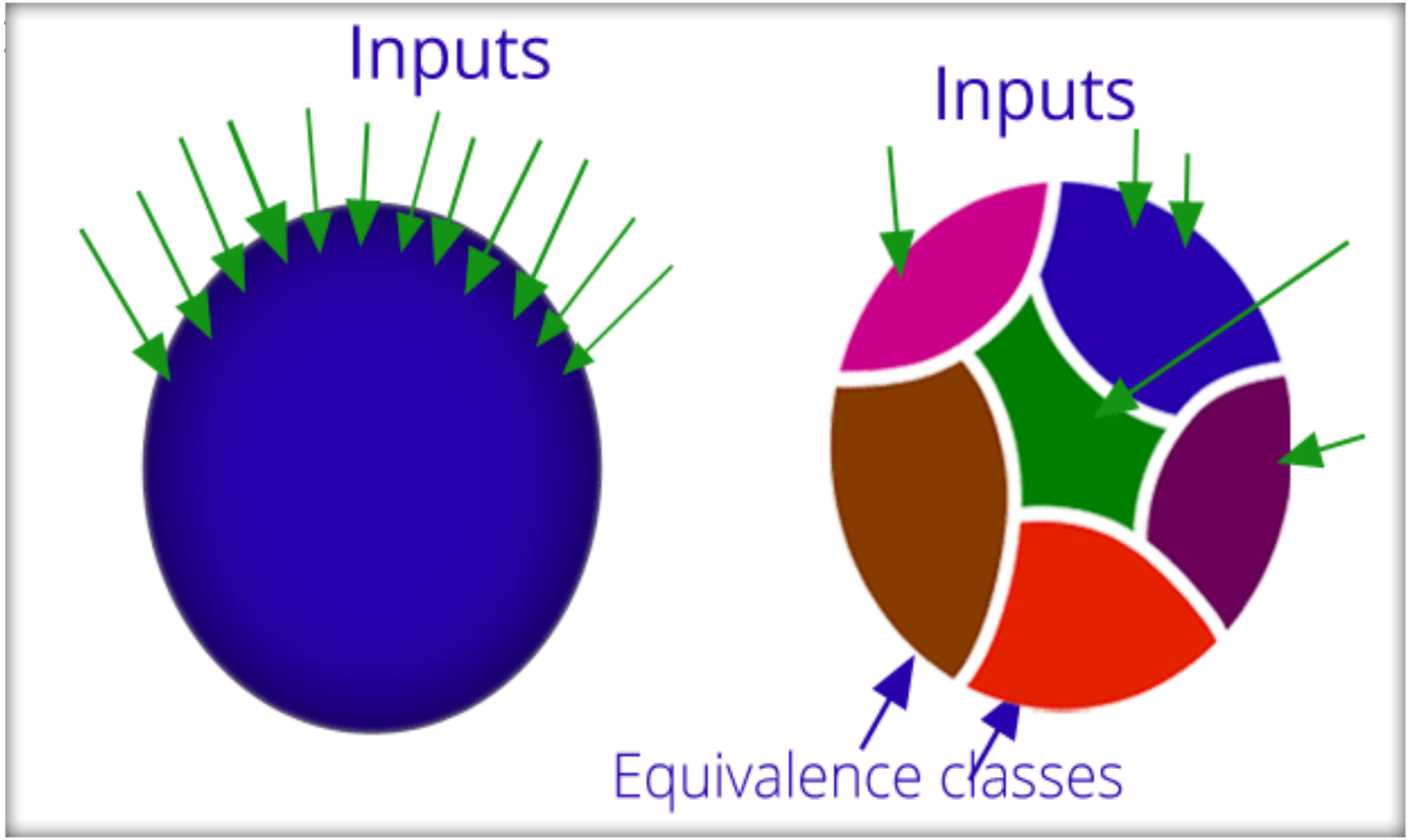
1.3.1/. Kỹ thuật kiểm thử Equivalence Class Partitioning (phân vùng tương đương):

Kỹ thuật này còn được gọi là phân vùng lớp tương đương (ECP). Trong kỹ thuật này, các giá trị đầu vào cho hệ thống / ứng dụng được chia thành các lớp khác nhau dựa trên sự giống nhau của nó trong kết quả.

Do đó, thay vì sử dụng từng giá trị đầu vào, giờ đây chúng ta có thể sử dụng bất kỳ một giá trị nào từ lớp để kiểm tra kết quả. Bằng cách này, chúng ta có thể duy trì phạm vi kiểm tra trong để giảm bớt đi các giá trị kiểm thử và quan trọng nhất là thời gian kiểm thử.

1. Black box Testing

1.3.2/. Kỹ thuật kiểm thử Equivalence Class Partitioning (phân vùng tương đương)



1. Black box Testing

Kiểm thử phân vùng tương đương yếu:

- Kiểm thử phân vùng tương đương yếu chỉ lấy các phần tử đại diện ít nhất một lần. Kiểm thử chọn một biến từ mỗi lớp tương đương. Vậy số Test case tối thiểu sẽ bằng số vùng / lớp có nhiều tập con nhất.
- **Định nghĩa khác:** Kiểm thử phân vùng tương đương yếu chỉ yêu cầu mỗi vùng / lớp tương đương có ít nhất một phần tử xuất hiện trong một Test case nào đó.

1. Black box Testing

Vd1: Xét chương trình P có ba biến đầu vào: a, b và c với các miền xác định là A, B, and C.

Phân hoạch của các miền này giả sử là:

- A = {a1 U a2 U a3} - phân vùng A có 3 giá trị a1, a2, a3.
- B = {b1 U b2 U b3 U b4) - phân vùng B có 4 giá trị b1, b2, b3, b4.
- C = {c1 U c2} - phân vùng C có 2 giá trị c1, c2.

1. Black box Testing

- **Vd1:** Tập các Test case bên sử dụng một giá trị từ mỗi lớp tương đương. Để kiểm thử chúng ta chỉ cần để ý đến các cột, xét cột **A** ta thấy nó xuất hiện cả 3 giá trị a1, a2, a3 tập con của **A**, như vậy là đủ.
- Tương tự cho cột B, C

Test case	A	B	C
1	a1	b1	c1
2	a2	b2	c2
3	a3	b3	c1
4	a1	b4	c2

1. Black box Testing

Kiểm thử phân vùng tương đương mạnh:

- Kiểm thử phân vùng tương đương mạnh sẽ kết hợp các tổ hợp có thể của các vùng / lớp tương đương.
- Kiểm thử phân vùng tương đương mạnh dựa trên tích Đề-các của các tập con. Nó tạo ra nhiều Test case cho bất kỳ tương tác nào giữa các giá trị đại diện.

1. Black box Testing

Vd2: Các Test case phân vùng tương đương mạnh cho bài toán ở ví dụ 1

- Với ví dụ 1 ở trên ta sẽ có tổ hợp $(3 * 4 * 2) = 24$ bộ giá trị tương ứng với 24 Test case như bảng ở Slide sau.
- (số phần tử của A = 3; số phần tử của B = 4; số phần tử của C = 2)

Sinh viên thảo luận.

1. Black box Testing

Test case	A	B	C
1	a1	b1	c1
2	a1	b1	c2
3	a1	b2	c1
4	a1	b2	c2

Test case	A	B	C
5	a1	b3	c1
6	a1	b3	c2
7	a1	b4	c1
8	a1	b4	c2

1. Black box Testing

Test case	A	B	C
9	a2	b1	c1
10	a2	b1	c2
11	a2	b2	c1
12	a2	b2	c2

Test case	A	B	C
13	a2	B3	c1
14	a2	b3	c2
15	a2	b4	c1
16	a2	b4	c2

1. Black box Testing

Test case	A	B	C
17	a3	b1	c1
18	a3	b1	c2
19	a3	b2	c1
20	a3	b2	c2

Test case	A	B	C
21	a3	b3	c1
22	a3	b3	c2
23	a3	b4	c1
24	a3	b4	c2

1. Black box Testing

Vd 3:

Có Textbox Age chấp nhận các giá trị số từ 10-60.

Ta thấy có 3 lớp giá trị là:

- Các số <=17
- Các số >=61
- Các số từ 18 - 61

Equivalence Class Partitioning (ECP)

AGE [Enter Age] * Accepts value from 18 to 60

Equivalence Class Partitioning		
Invalid	Valid	Invalid
<=17	18-60	>=61

1. Black box Testing

- Các số <=17 (lớp không hợp lệ _ Invalid)
- Các số >=61 (lớp không hợp lệ _ Invalid)
- Các số từ 18 – 61 (lớp hợp lệ)

Một lớp hợp lệ sẽ là bất cứ số nào từ 18 đến 60.

Do đó, chúng ta đã giảm các trường hợp kiểm thử xuống chỉ còn 3 trường hợp kiểm thử dựa trên các lớp được hình thành (bao gồm tất cả các khả năng). Vì vậy, kiểm thử với bất kỳ giá trị nào từ mỗi bộ của 3 lớp là đủ để kiểm tra kịch bản trên.

1. Black box Testing

Vd 4: Chương trình tính tiền thưởng cuối năm cho nhân viên, nó phụ thuộc vào thời gian làm việc với công ty:

- Làm việc nhiều hơn 3 năm = 50% bonus
- Làm việc nhiều hơn 5 năm = 80% bonus
- Làm việc nhiều hơn 8 năm = 100% bonus

Sinh viên thảo luận.

1. Black box Testing

Test case theo phân vùng tương đương đơn giản:

	Vùng / lớp tương đương	Giá trị kiểm thử	Kết quả mong đợi
Thời gian làm việc của nhân viên tính bằng năm (X)	0 <= X <= 3	2	0%
	3 < X <= 5	4	50%
	5 < X <= 8	7	80%
	X > 8	12	100%

1. Black box Testing

Test case theo phân vùng tương đương hợp lệ và không hợp lệ:

	Vùng / lớp tương đương	Giá trị kiểm thử	Kết quả mong đợi
Thời gian làm việc của nhân viên tính bằng năm (X)	0 <= X <= 3	2	0%
	3 < X <= 5	4	50%
	5 < X <= 8	7	80%
	X > 8	12	100%
	X < 0	-1	Từ chối
	X > 70	75	Từ chối
	Na#	Abc	Từ chối
	Blank		Từ chối

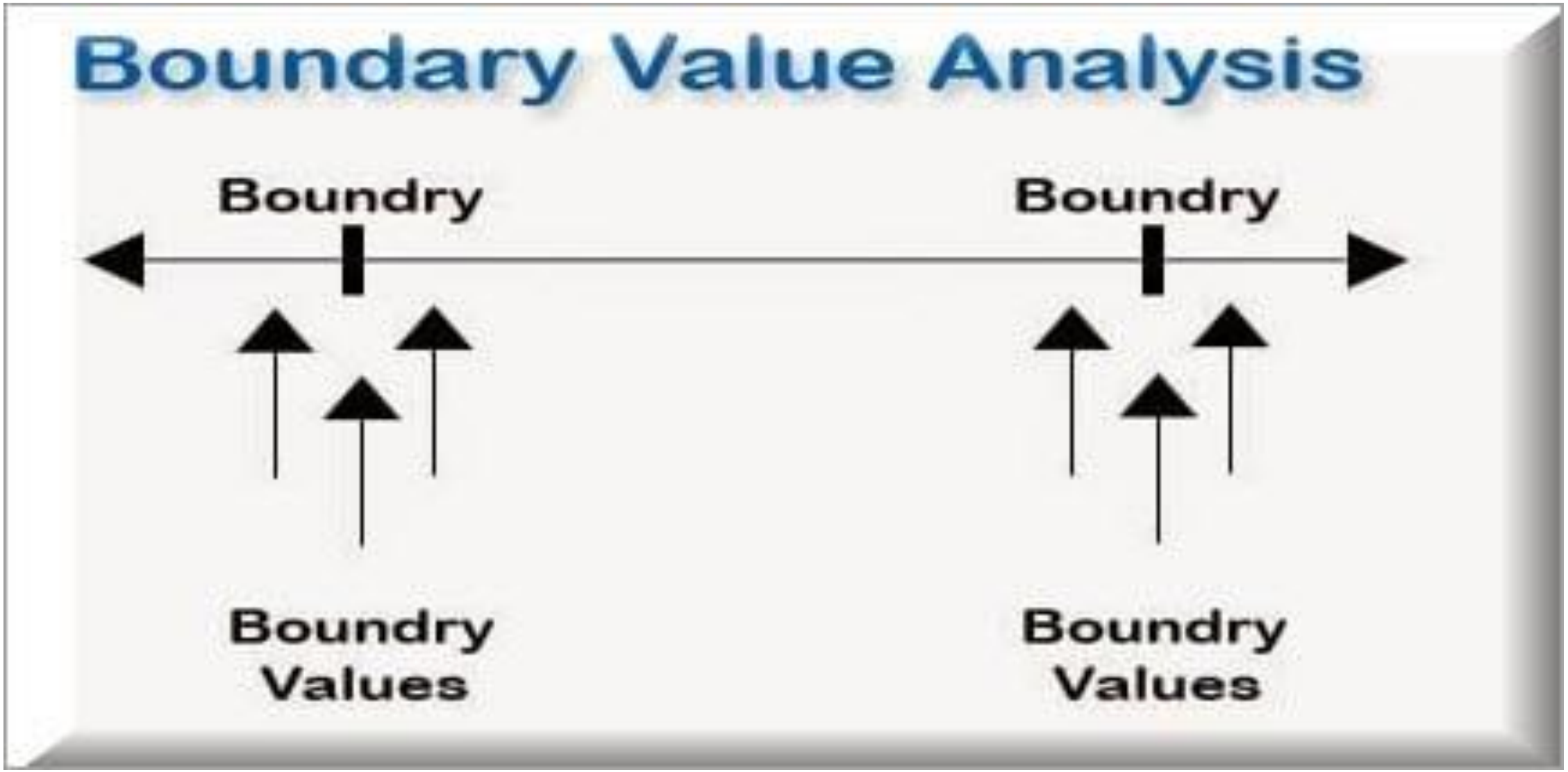
1. Black box Testing

1.3.3/. Kỹ thuật kiểm thử Boundary Value Analysis (phân tích giá trị biên):

Chúng ta tập trung vào các giá trị tại các ranh giới / biên vì người ta thấy rằng nhiều ứng dụng có phần lớn các vấn đề trên các ranh giới / biên.

Ranh giới có nghĩa là các giá trị gần giới hạn ở các biên, mà hành vi của hệ thống sẽ thay đổi. Trong phân tích giá trị biên có xem xét cả các giá trị đầu vào hợp lệ và đầu vào không hợp lệ khi kiểm thử, nhằm xác minh các vấn đề / lỗi.

1. Black box Testing



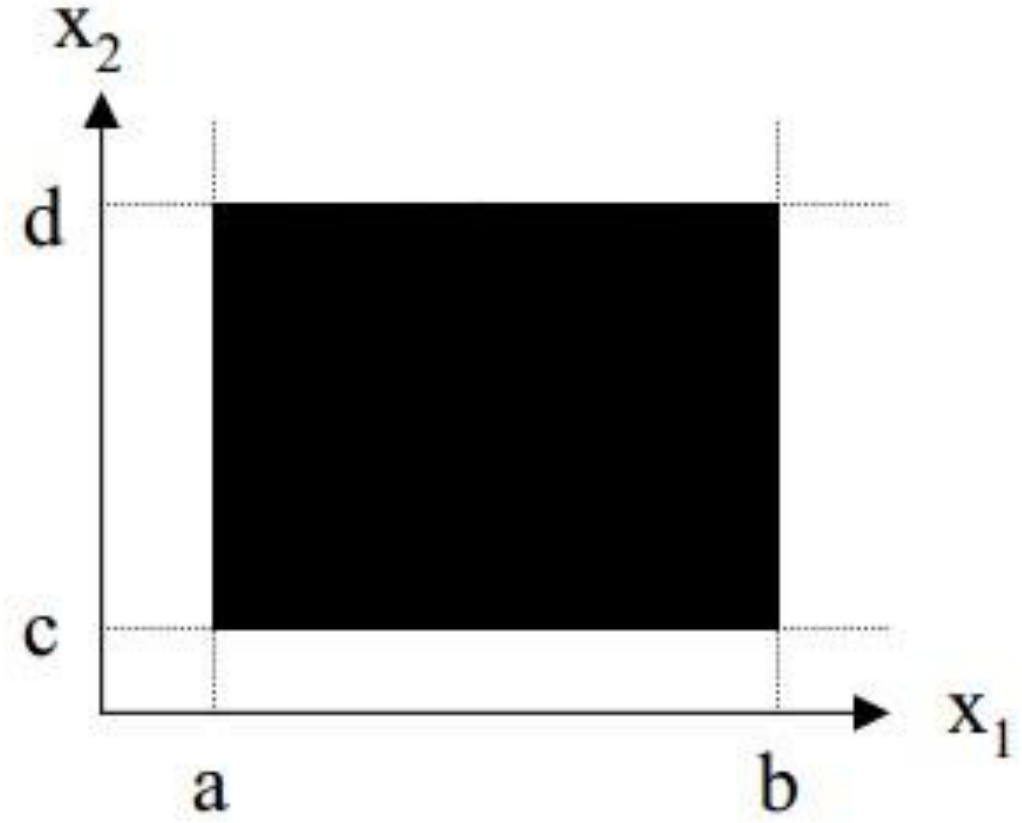
1. Black box Testing

Giả sử y = f(x1,x2) với x1,x2,y ∈ N là một hàm toán học của một chương trình. Khi đó thông thường x1 và x2 có miền xác định thể hiện bằng các biến.

Vd: $a \leq x1 \leq b$ và $c \leq x2 \leq d$

trong đó a,b,c,d là các giá trị biên.

Hình thể hiện miền xác định của hai biến



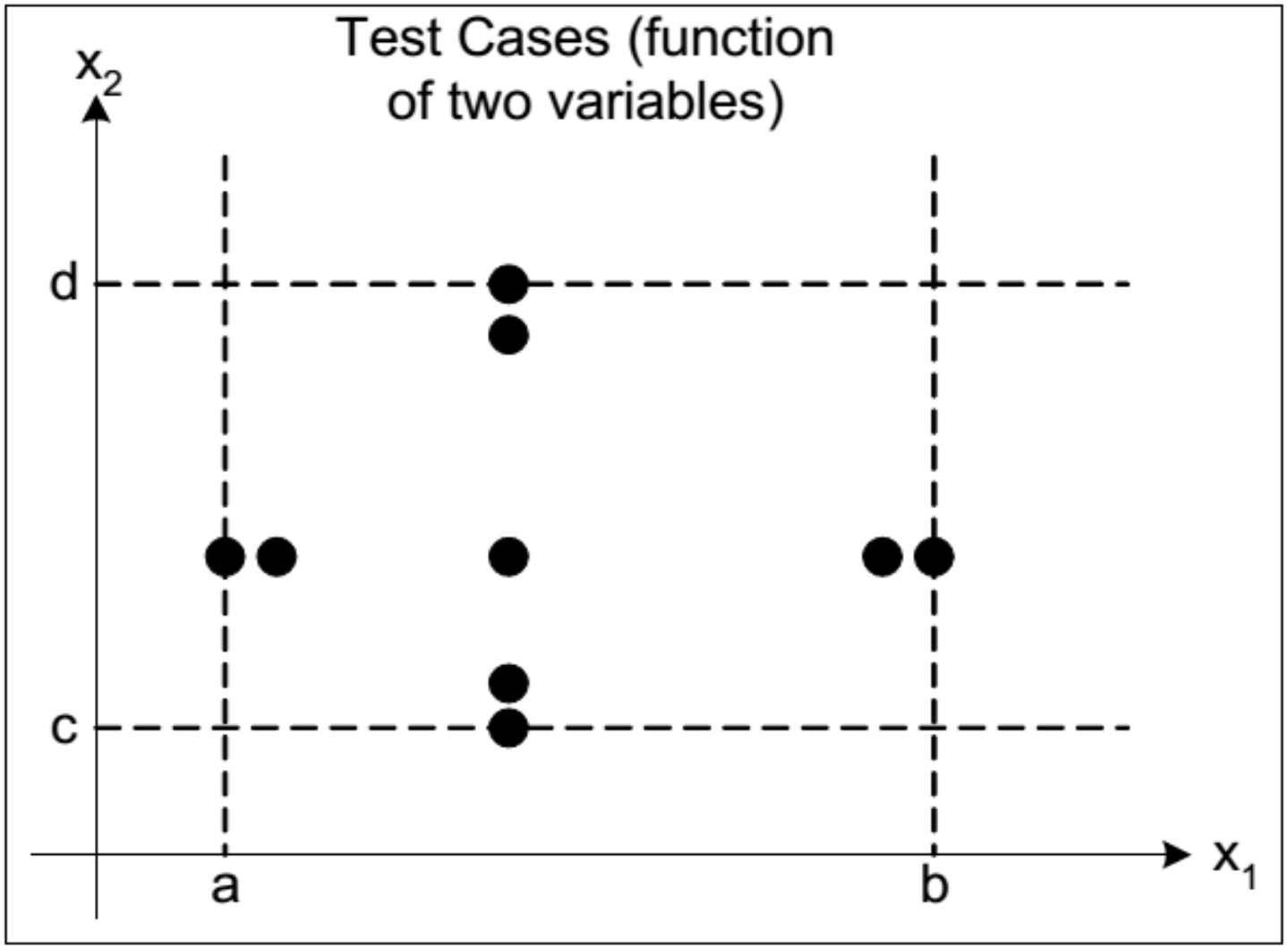
1. Black box Testing

Với dữ liệu được nhập vào, do lỗi lập trình hoặc lỗi đặc tả làm các biểu thức điều kiện không chính xác. Chẳng hạn đúng ra phải là dấu <= nhưng người lập trình hoặc đặc tả lại chỉ viết < hoặc ngược lại. Kiểm thử với các giá trị biên giúp chúng ta phát hiện các lỗi này.

Để tăng khả năng phát hiện lỗi, kiểm thử giá trị biên thường xác định 05 giá trị: Max (cực đại), Min (cực tiểu), Min+, Max- (các giá trị cận biên), Nom (giá trị ở giữa miền xác định đại diện cho giá trị thông thường).

(Min, Min+, Nom, Max-, Max)

1. Black box Testing



Sinh viên thảo luận, giải thích hình bên

1. Black box Testing

Kiểm thử giá trị biên mạnh:

Kiểm thử giá trị biên mạnh là mở rộng của kiểm thử giá trị biên bằng việc ở bên ngoài miền xác định. Ngoài 5 giá trị biên đã nêu ở phần trước, chúng ta sẽ lấy thêm các giá trị cận biên ở ngoài miền xác định là (Max+ và Min−).

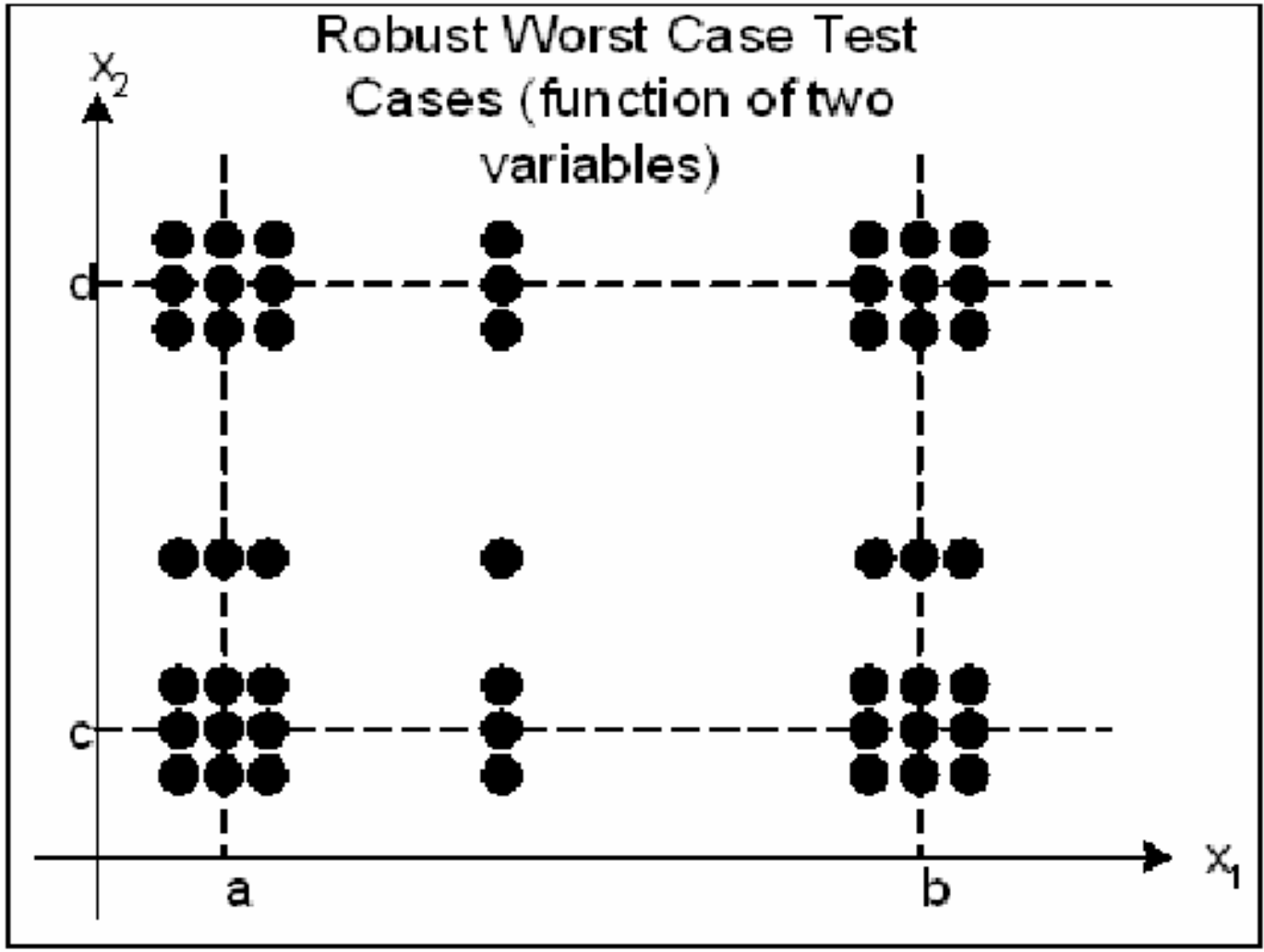
Ta được 7 giá trị cần kiểm thử

(Min-, Min, Min-, Nom, Max-, Max, Max+)

1. Black box Testing

Kiểm thử giá trị: biên tổ hợp: là mở rộng của kiểm thử giá trị biên mạnh bằng việc tổ hợp các giá trị biên và cận biên cho tất cả các trường hợp kiểm thử của các biến.

Sinh viên xem hình và thảo luận.



1. Black box Testing

Vd 5:Bài toán Tam giác (Triangle)

- Chúng ta xem việc áp dụng kiểm thử giá trị biên với bài toán Triangle. Bài toán yêu cầu nhập 3 cạnh của 1 tam giác, thực hiện kiểm thử các trường hợp giá trị, biết cận dưới của các cạnh tam giác là 1 và cận trên là 200.
- Ta có (4n + 1) = 12 + 1 = 13 Test case với T/h kiểm thử giá trị biên bình thường:

  (Min = 1; Min + = 2; Nom = 100; Max - = 199; Max = 200)
- Bài toán phải thoả 6 điều kiện sau:
- C1. $1 \leq a \leq 200$ và C2. $1 \leq b \leq 200$ và C3. $1 \leq c \leq 200$.
- C4. $a < b + c$ và C5. $b < a + c$ và C6. $c < a + b$.

1. Black box Testing

Vd 5:

Bài toán Tam giác

T/h Kiểm thử giá trị biên thường

Sinh viên thảo luận.

Boundary Value Analysis Test Cases

Case	a	b	c	Expected Output
1	100	100	1	Isosceles
2	100	100	2	Isosceles
3	100	100	100	Equilateral
4	100	100	199	Isosceles
5	100	100	200	Not a Triangle
6	100	1	100	Isosceles
7	100	2	100	Isosceles
8	100	199	100	Isosceles
9	100	200	100	Not a Triangle
10	1	100	100	Isosceles
11	2	100	100	Isosceles
12	199	100	100	Isosceles
13	200	100	100	Not a Triangle

1. Black box Testing

Vd 5:

Bài toán Tam giác

T/h Kiểm thử giá trị biên tổ hợp.

60 bộ giá trị kiểm thử trong 125 Test case.

Sinh viên thảo luận.

Worst Case Test Cases (60 of 125)

Case	a	b	c	Expected Output	Case	a	b	c	Expected Output
1	1	1	1	Equilateral	31	2	2	1	Isosceles
2	1	1	2	Not a Triangle	32	2	2	2	Equilateral
3	1	1	100	Not a Triangle	33	2	2	100	Not a Triangle
4	1	1	199	Not a Triangle	34	2	2	199	Not a Triangle
5	1	1	200	Not a Triangle	35	2	2	200	Not a Triangle
6	1	2	1	Not a Triangle	36	2	100	1	Not a Triangle
7	1	2	2	Isosceles	37	2	100	2	Not a Triangle
8	1	2	100	Not a Triangle	38	2	100	100	Isosceles
9	1	2	199	Not a Triangle	39	2	100	199	Not a Triangle
10	1	2	200	Not a Triangle	40	2	100	200	Not a Triangle
11	1	100	1	Not a Triangle	41	2	199	1	Not a Triangle
12	1	100	2	Not a Triangle	42	2	199	2	Not a Triangle
13	1	100	100	Isosceles	43	2	199	100	Not a Triangle
14	1	100	199	Not a Triangle	44	2	199	199	Isosceles
15	1	100	200	Not a Triangle	45	2	199	200	Scalene
16	1	199	1	Not a Triangle	46	2	200	1	Not a Triangle
17	1	199	2	Not a Triangle	47	2	200	2	Not a Triangle
18	1	199	100	Not a Triangle	48	2	200	100	Not a Triangle
19	1	199	199	Isosceles	49	2	200	199	Scalene
20	1	199	200	Not a Triangle	50	2	200	200	Isosceles
21	1	200	1	Not a Triangle	51	100	1	1	Not a Triangle
22	1	200	2	Not a Triangle	52	100	1	2	Not a Triangle
23	1	200	100	Not a Triangle	53	100	1	100	Isosceles
24	1	200	199	Not a Triangle	54	100	1	199	Not a Triangle
25	1	200	200	Isosceles	55	100	1	200	Not a Triangle
26	2	1	1	Not a Triangle	56	100	2	1	Not a Triangle
27	2	1	2	Isosceles	57	100	2	2	Not a Triangle
28	2	1	100	Not a Triangle	58	100	2	100	Isosceles
29	2	1	199	Not a Triangle	59	100	2	199	Not a Triangle
30	2	1	200	Not a Triangle	60	100	2	200	Not a Triangle

1. Black box Testing

1.3.4/. Kỹ thuật kiểm thử Decision Table:

Kiểm thử dựa trên bảng quyết định là phương pháp chính xác nhất trong các kỹ thuật kiểm thử chức năng.

Bảng quyết định là phương pháp hiệu quả để mô tả các tình huống, hành động sẽ xảy ra khi một số điều kiện thỏa mãn.

Kỹ thuật kiểm thử dựa trên bảng quyết định yêu cầu chúng ta xem xét các điều kiện của bộ giá trị của đầu vào từ đó có quyết định cụ thể.

1. Black box Testing

Cấu trúc bảng quyết định (Decision Table):

Cấu trúc bảng quyết định chia thành 03 phần chính (như trong hình):

- **Các dòng điều kiện**: chứa biểu thức điều kiện, có thể có nhiều hình thức khác. Mỗi điều kiện được xem như 1 biến / 1 quan hệ / 1 mệnh đề, giá trị của điều kiện có thể là Y/N, T/F (bảng quyết định logic) hoặc giá trị khác (bảng quyết định mở rộng)

1. Black box Testing

- **Các dòng hành động ứng với điều kiện có các giá trị** : Y/N hoặc T/F (có xảy ra hay không ?) hoặc X (xảy ra / không xảy ra) hoặc bỏ trống hoặc ‘-‘ (không quan tâm),...
- **Các cột là các qui tắc (rule)**: là sự kết hợp các giá trị của biến ứng với điều kiện, các qui tắc này được giải thích như các Test case.

1. Black box Testing

- Khi lập bảng quyết định, người ta thường tìm các điều kiện có thể xảy ra, để xét các tổ hợp của chúng rồi từ đó ta sẽ xác định được các Test case tương ứng cho các điều kiện được thỏa mãn. Các hành động xảy ra chính là kết quả mong đợi của Test case đó.
- Bảng quyết định với các giá trị điều kiện chỉ là T/ F (hoặc Y / N), và – được gọi là **bảng quyết định lôgic**. Chúng ta có thể mở rộng các giá trị này bằng các tập giá trị khác, ví dụ 1, 2, 3, 4,… khi đó chúng ta có **bảng quyết định mở rộng**.

1. Black box Testing

Bảng quyết định

	Rule 1 R1	Rule 2 R2	Rule 2 R2	Rule 3 R3	Rule 4 R4
Điều kiện A _CA	T	T	F	F	T
Điều kiện B_CB	F	F	F	T	F
Hành động 1_A1	T	F	T	T	F
Hành động 2_A2	F	F	T	F	T

1. Black box Testing

Bảng quyết định mở rộng

	Rule 1 R1	Rule 2 R2	Rule 2 R2	Rule 3 R3	Rule 4 R4
Điều kiện A _CA	T	T	F	F	T
Điều kiện B_CB	F	F	F	T	F
Hành động 1_A1	X	_	X	Y	Y
Hành động 2_A2	X	Y	Z	Z	Z

1. Black box Testing

Vd 6: bảng quyết định xử lý sự cố máy tính

Conditions Điều kiện	Printer không in	Y	Y	Y	Y	N	N	N	N
	Đèn đỏ nhấp nháy	Y	Y	N	N	Y	Y	N	N
	Printer không được nhận	Y	N	Y	N	Y	N	Y	N
Actions Hành động	Kiểm tra cable nguồn			X					
	Kiểm tra cable printer-computer	X		X					
	Bảo đảm Driver máy in được cài đặt	X		X		X		X	
	Kiểm tra / thay hộp mực	X	X			X	X		
	Kiểm tra có bị kẹt giấy		X		X				

1. Black box Testing

Tóm tắt:

- Các điều kiện được hiểu là yếu tố đầu vào (Input) hoặc các lớp tương đương của đầu vào (n là số điều kiện).
- Các hành động là kết quả đầu ra (Output) hoặc chức năng xử lý chính của bảng quyết định.
- Các qui tắc là các cột ứng với các giá trị xử lý (là tổ hợp / bộ giá trị được kết hợp) của điều kiện. Nếu các điều kiện chỉ có giá trị là T/F (nhị phân) thì ta có 2^n cột qui tắc; trường hợp điều kiện có giá trị từ tập hợp > hơn 2 thì bảng quyết định trở thành bảng quyết định mở rộng với nhiều hơn 2^n cột.

1. Black box Testing

Bảng quyết định dùng cho trường hợp nào?

Bảng quyết định thích hợp nhất cho các chương trình / phần mềm có:

- Nhiều việc ra quyết định, có mối quan hệ giữa các biến đầu vào, có các tính toán liên quan đến các tập hợp con của các biến đầu vào, có mối quan hệ nhân quả giữa đầu vào và đầu ra, có logic tính toán phức tạp.
- Bảng quyết định không có quy mô tăng lên lớn; bảng quyết định có thể được tinh chế gọn lại.

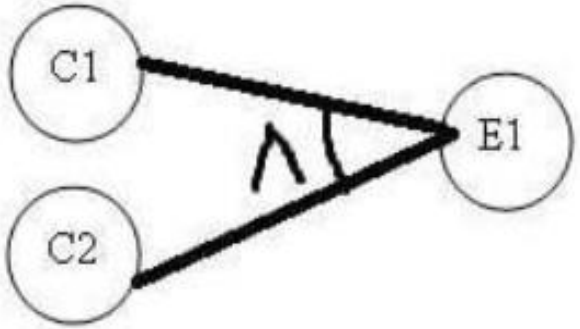
1. Black box Testing

1.3.5/. Kỹ thuật kiểm thử đồ thị nhân quả (CEG_ Cause Effect Graph):

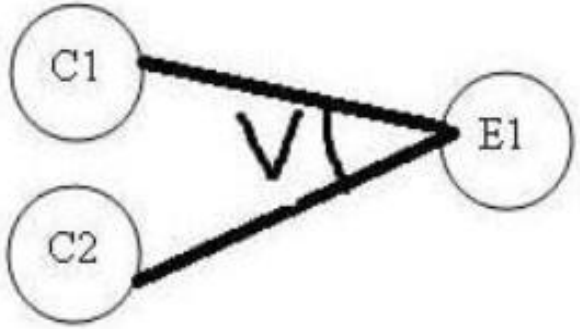
- Đồ thị nhân quả là một kỹ thuật cơ bản của phần cứng nhưng tương thích với phần mềm. Kỹ thuật kiểm thử dựa trên CEG là kỹ thuật Black box, nó thể hiện mối quan hệ nguyên nhân _ kết quả cho thấy quyết định xử lý. CEG là kỹ thuật dùng hỗ trợ thêm cho công việc phát triển bảng quyết định.
- Đồ thị nhân quả bao gồm :
  - Các nút : là các nguyên nhân và kết quả (input và output).
  - Các cạnh : là đường nối giữa các nút thể hiện mối quan hệ.
  - Các phép toán : $\wedge$ (and), $\vee$ (or), ~ (not)

1. Black box Testing

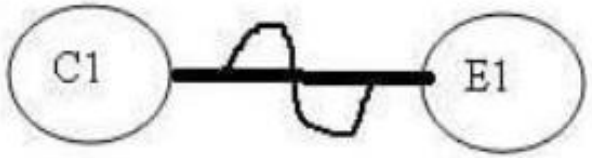
AND – For effect E1 to be true, both the causes C1 and C2 should be true



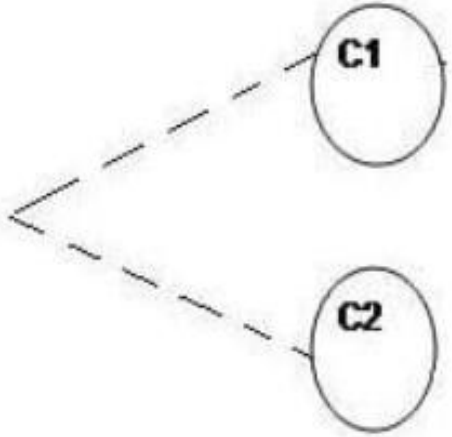
OR – For effect E1 to be true, either of causes C1 OR C2 should be true



NOT – For Effect E1 to be True, Cause C1 should be false



MUTUALLY EXCLUSIVE – When only one of the causes will hold true.



1. Black box Testing

Vd 7: Máy in tin nhắn là phần mềm đọc hai ký tự và tùy thuộc vào giá trị của chúng, tin nhắn được in:

- Nếu ký tự đầu tin là ‘A’ hoặc ‘B’ và ký tự thứ 2 là số thì tập tin sẽ được cập nhật.
- Nếu ký tự đầu tiên không chính xác (không là ‘A’ hay ‘B’), thông điệp X phải được in.
- Nếu ký tự thứ hai không chính xác (không là số), thông báo Y phải được in.

1. Black box Testing

Giải:

- **Các nguyên nhân (Causes) của bài toán:**

  C1 – ký tự đầu là A

  C2 – ký tự đầu là B

  C3 – ký tự thứ hai là số

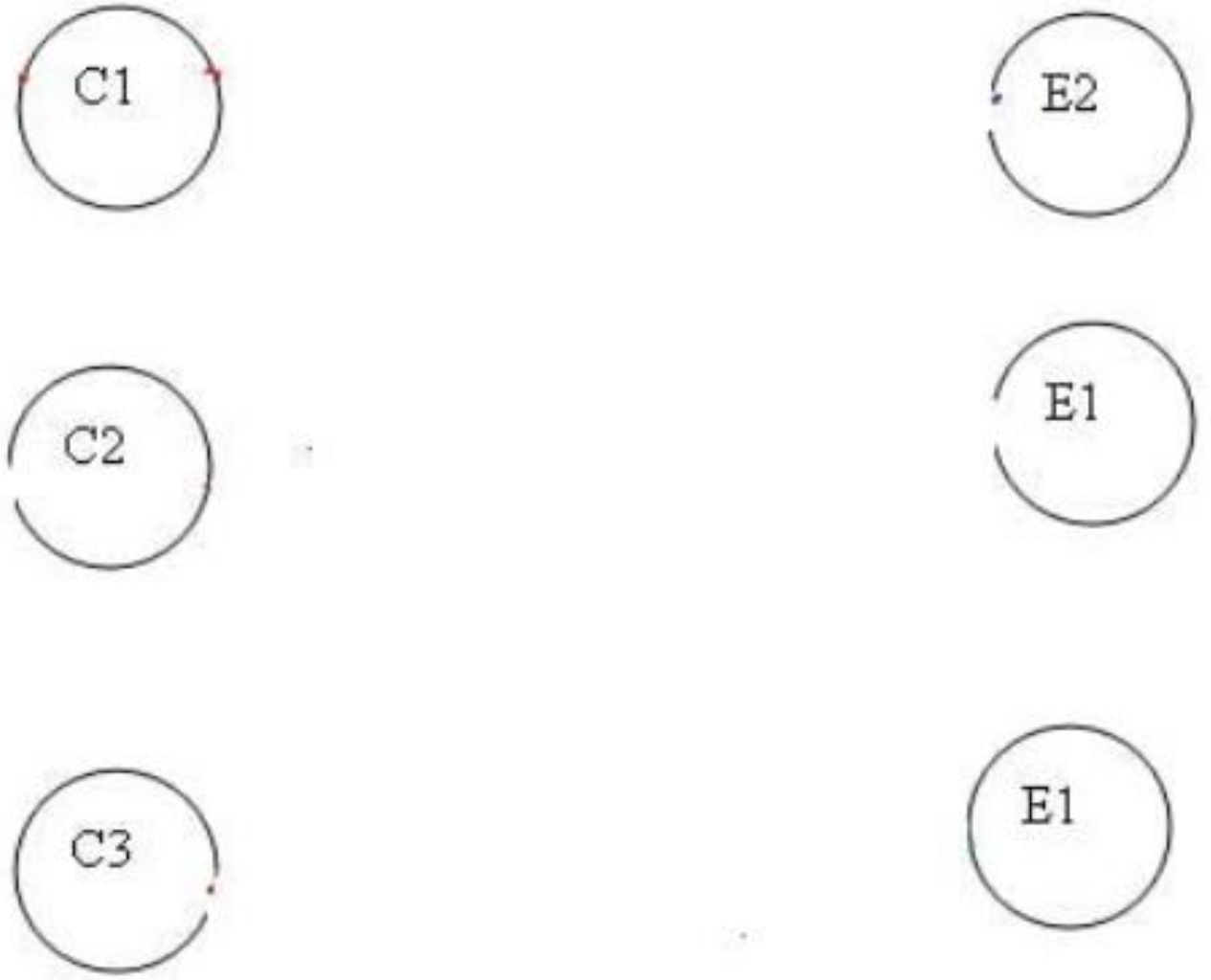
- **Các kết quả (Results) của bài toán:**

  E1 – Cập nhật file

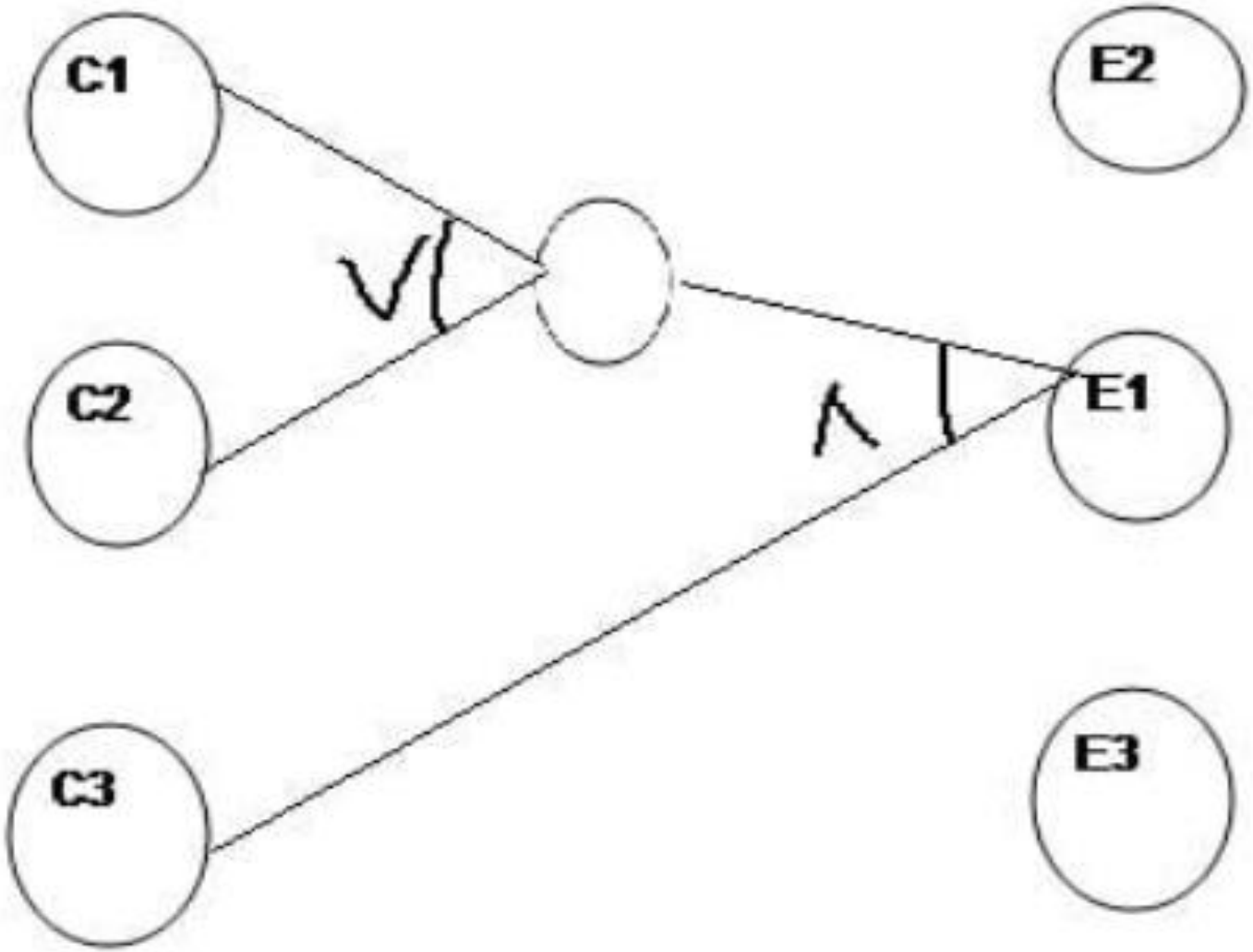
  E2 – In tin nhắn X

  E3 – In tin nhắn Y.

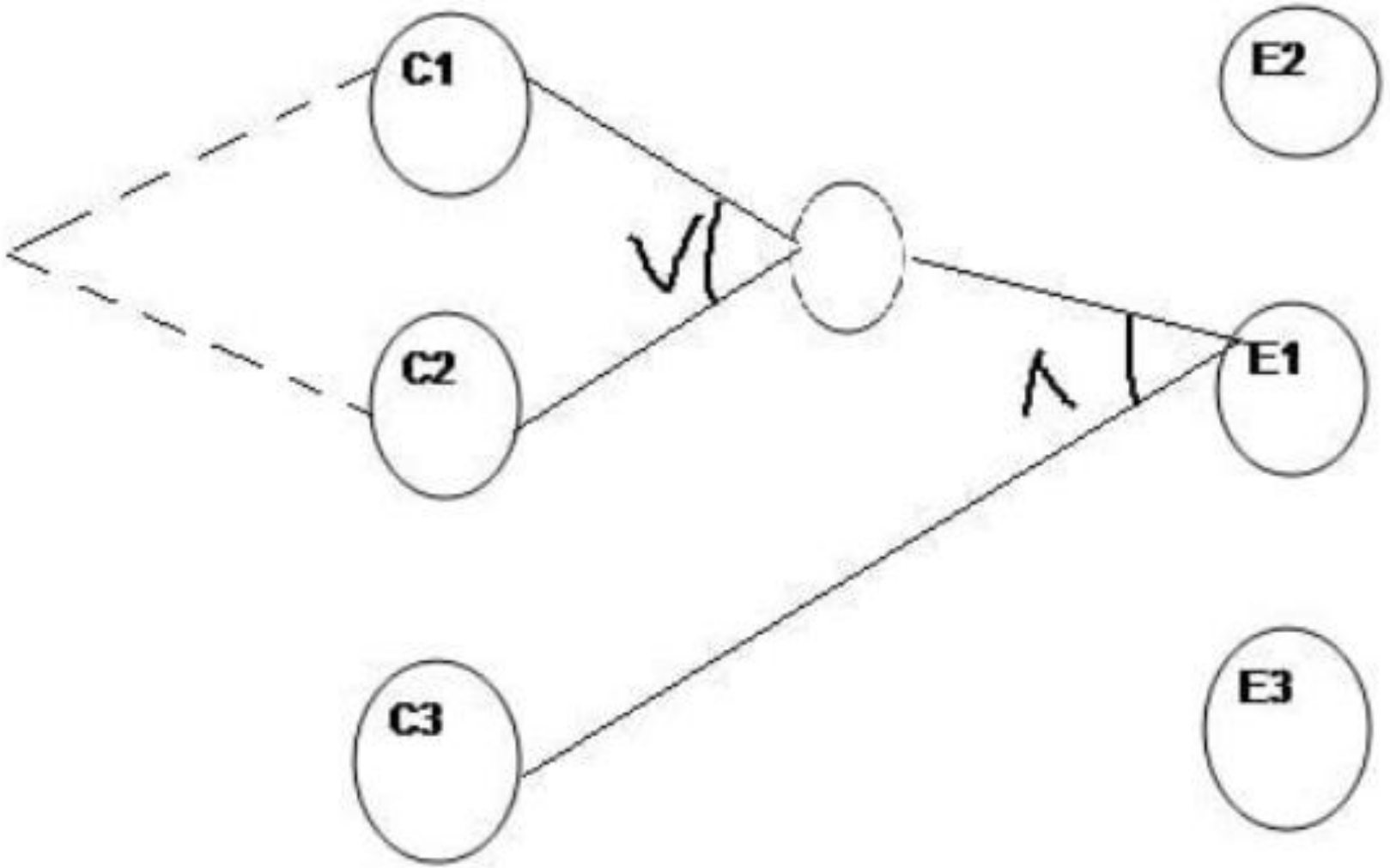
1. Black box Testing



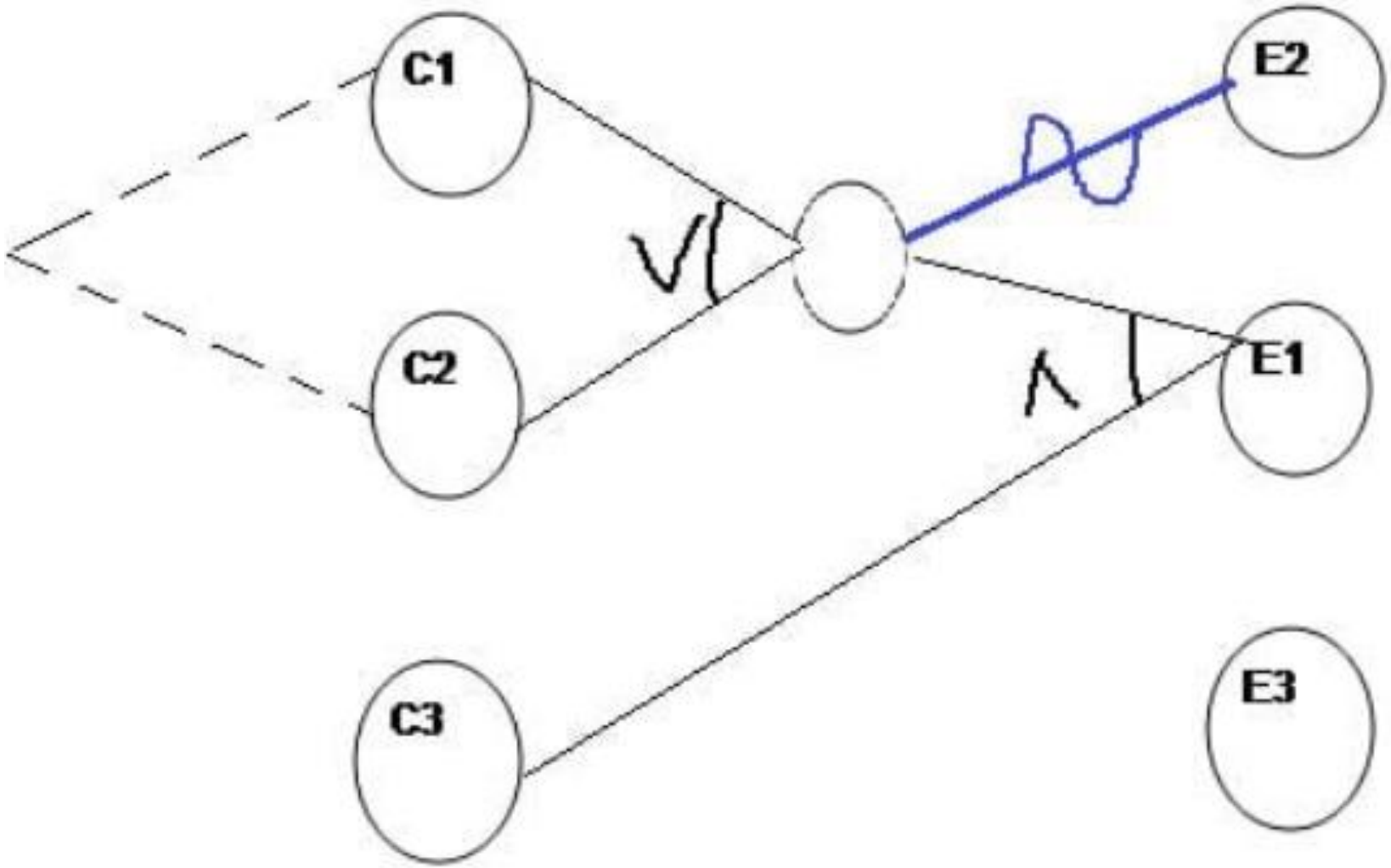
1. Black box Testing



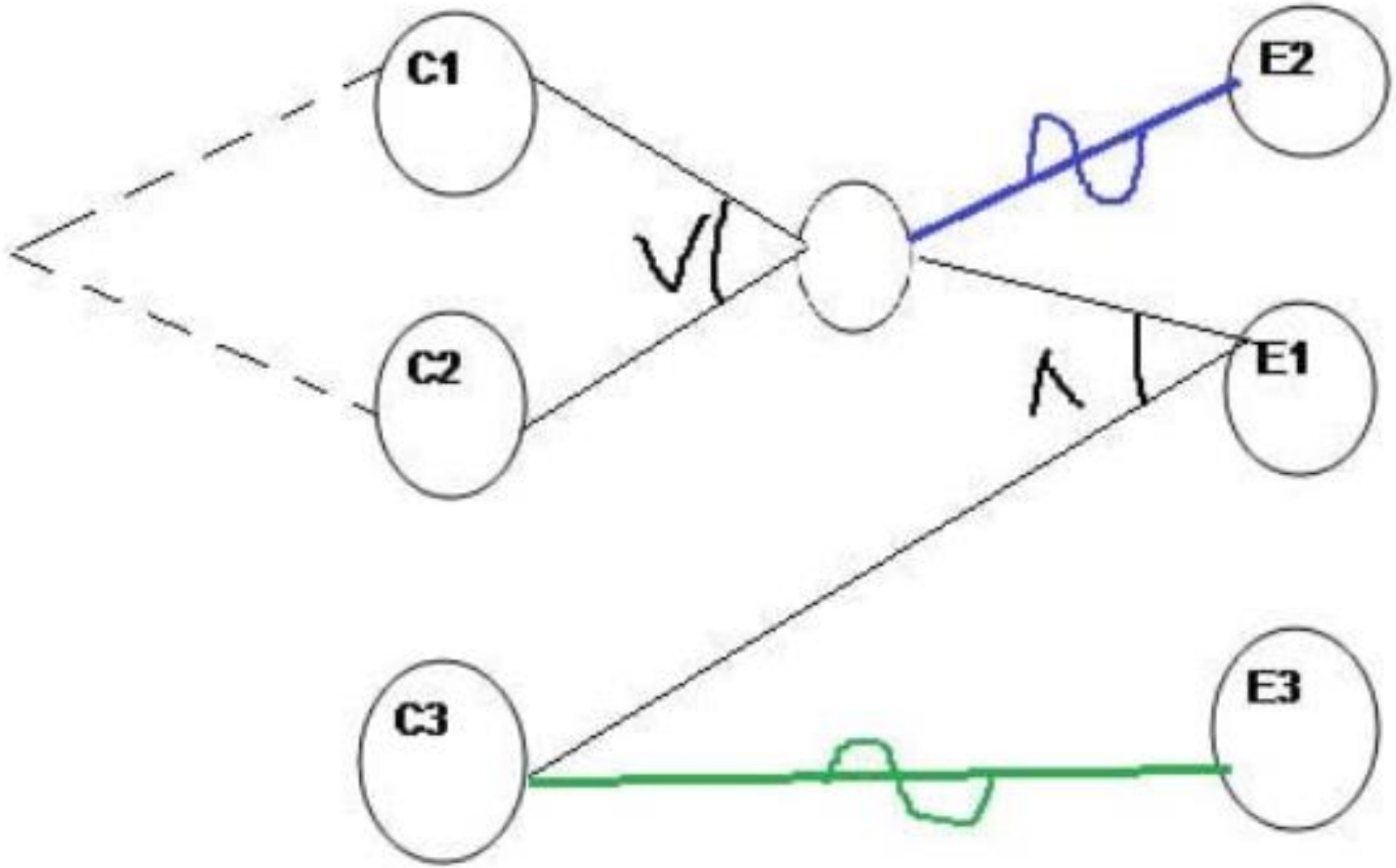
1. Black box Testing



1. Black box Testing



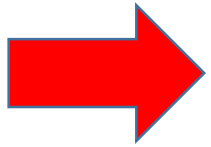
1. Black box Testing



1. Black box Testing

Tạo bảng quyết định ban đầu, E1 là đúng với (C1 ∨ C2) ∧ C3

Actions
C1
C2
C3
E1
E2
E3



Actions	
C1	
C2	
C3	
E1	1
E2	
E3	

1. Black box Testing

E1 là đúng, với 2 điều kiện: (C1 ∧ C3) là đúng, (C2 ∧ C3) là đúng

Actions	
C1	
C2	
C3	
E1	1
E2	
E3	

Actions		
C1	1	
C2		1
C3	1	1
E1	1	1
E2		
E3		

1. Black box Testing

E2 là đúng, với cả C1, C2 là sai.

Actions				
C1	1		0	
C2		1		0
C3	1	1	0	1
E1	1	1		
E2			1	1
E3				

1. Black box Testing

E3 là đúng, C3 sẽ là sai.

Actions						
C1	1		0		1	
C2		1		0		1
C3	1	1	0	1		
E1	1	1				
E2			1	1		
E3					1	1

1. Black box Testing

Kết quả ta có bảng quyết định sau với các Test case.

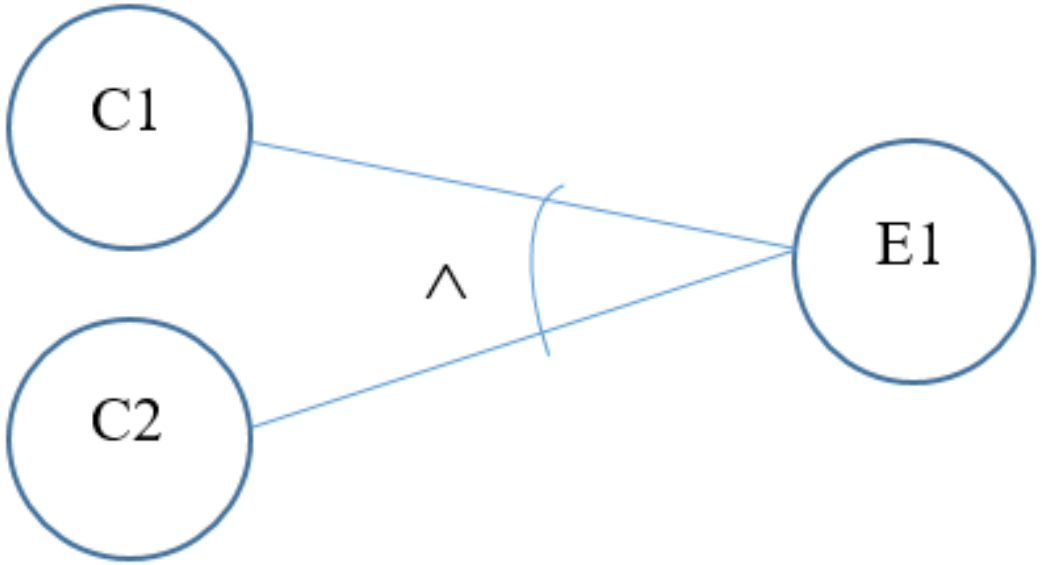
Actions	TC1	TC2	TC3	TC4	TC5	TC6
C1	1	0	0	0	1	0
C2	0	1	0	0	0	1
C3	1	1	0	1	0	0
E1	1	1	0	0	0	0
E2	0	0	1	1	0	0
E3	0	0	0	0	1	1

1. Black box Testing

Vd 8: Có các nguyên nhân (Cause) và kết quả (Effect) như sau:

- C1: Tài khoản không bị khoá
- C2: số dư > số tiền rút
- C3: số dư <= 0
- C4: Tài khoản bị khoá
- E1: chấp nhận giao dịch
- E2: từ chối giao dịch

1. Black box Testing



Sinh viên thảo luận và tự vẽ đồ thị CEG cho Vd 7 (Bài tập)

1. Black box Testing

1.3.6/. Kỹ thuật kiểm thử State Transition : được sử dụng khi hệ thống được mô tả bằng máy trạng thái. Hiểu một cách đơn giản là hệ thống có thể ở một số lượng giới hạn các trạng thái và phiên làm việc từ một trạng thái này đến một trạng thái kia được qui định bởi một rule của máy trạng thái. Đây mà mô hình mà ta sẽ dựa vào đó làm cơ sở để kiểm thử. Bất kỳ hệ thống nào mà ta có những output khác nhau từ những input giống nhau và việc có được output còn phụ thuộc vào những trạng thái xảy ra trước đó và một hệ thống như vậy sẽ được thể hiện bởi một State diagram (sơ đồ trạng thái).

1. Black box Testing

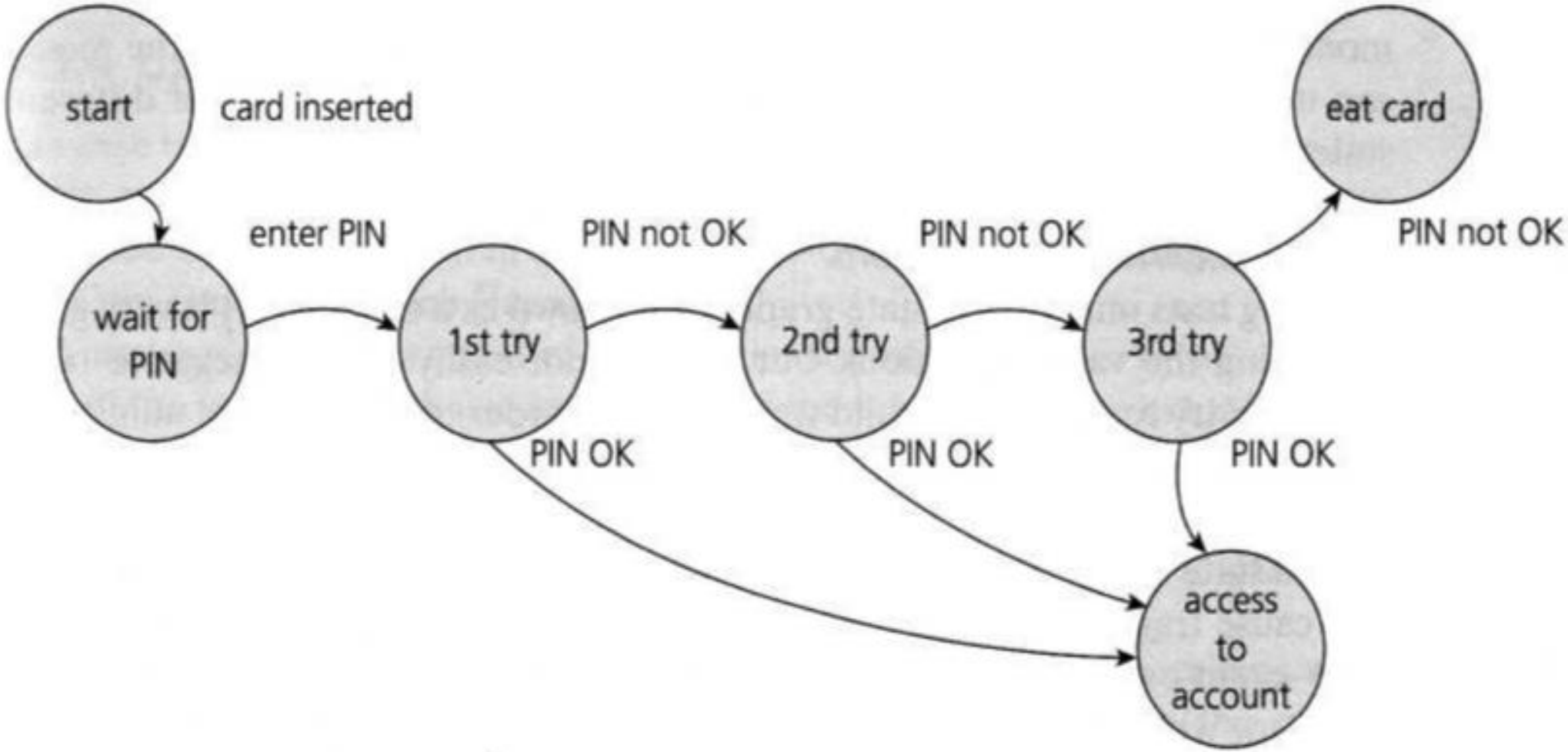
1.3.7/. Kỹ thuật kiểm thử State Transition :

Kiểm thử chuyển đổi trạng thái là một kỹ thuật được sử dụng để kiểm tra các trạng thái khác nhau của hệ thống. Trạng thái của hệ thống thay đổi tùy theo điều kiện hoặc sự kiện. Các sự kiện kích hoạt các trạng thái trở thành kịch bản và người kiểm tra cần kiểm thử chúng.

Một sơ đồ chuyển trạng thái có hệ thống cung cấp một cái nhìn rõ ràng về các thay đổi trạng thái nhưng nó có hiệu quả đối với các ứng dụng đơn giản hơn. Các dự án phức tạp hơn có thể dẫn đến các sơ đồ chuyển tiếp phức tạp hơn do đó làm cho nó kém hiệu quả hơn.

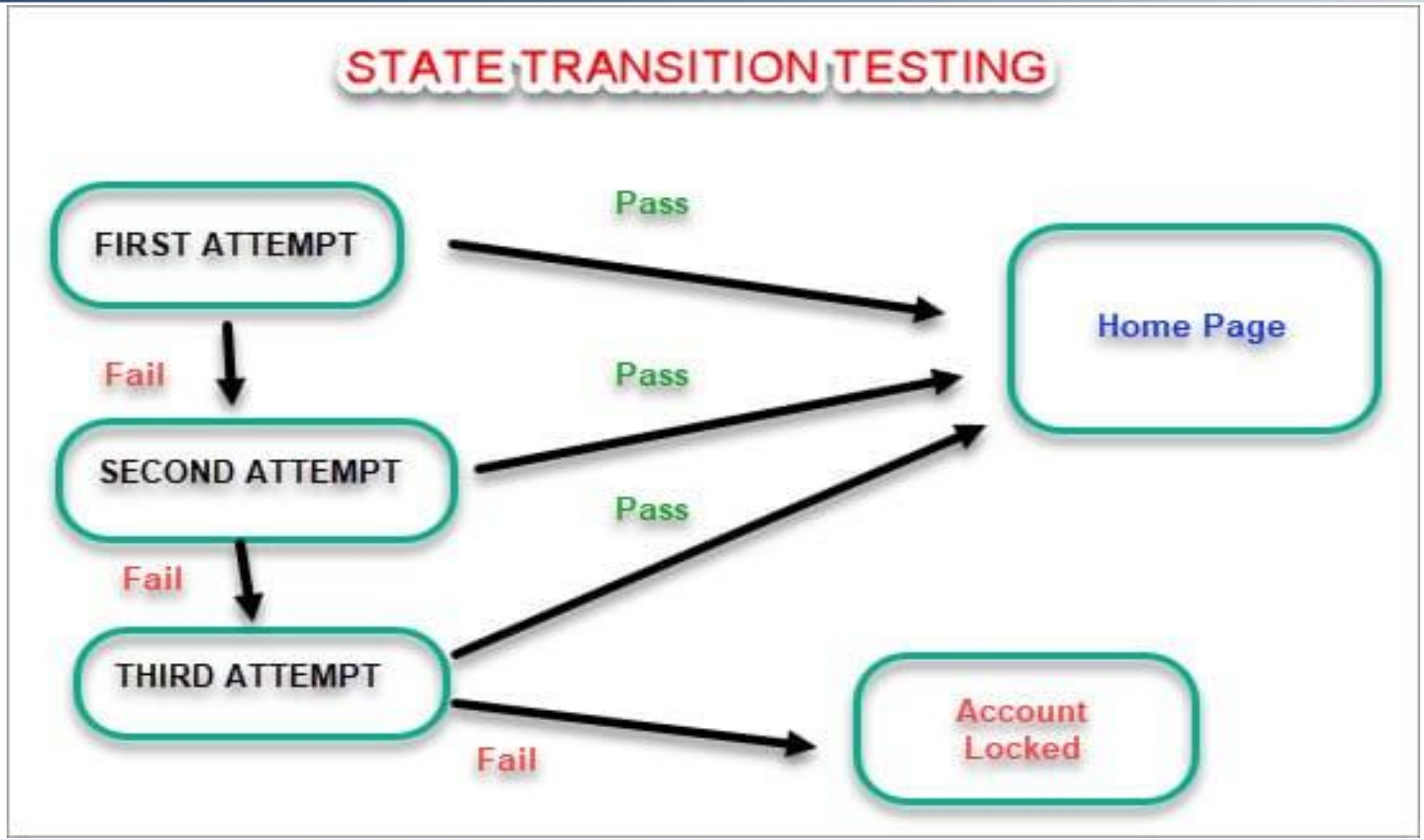
1. Black box Testing

Ví dụ:
Hệ thống rút tiền tại cây ATM



1. Black box Testing

Ví dụ:
Kiểm thử chuyển trạng thái cho hệ thống đăng nhập, nếu đăng nhập sai 3 lần sẽ bị khóa tài khoản.



1. Black box Testing

1.3.8/. Kỹ thuật kiểm thử Error Guessing : Trong kỹ thuật này, người kiểm thử có thể sử dụng kinh nghiệm của mình đã trải nghiệm để đoán các khu vực dễ bị lỗi. Nhiều lỗi có thể được tìm thấy bằng cách đoán lỗi trong đó hầu hết các nhà phát triển thường mắc lỗi. Một số lỗi phổ biến thường gặp:

- Chia cho số không.
- Xử lý các giá trị null trong các trường văn bản.
- Chấp nhận nút Submit mà không có giá trị nào nhập.
- Tải lên tập tin mà không cần đính kèm.
- Tải lên tập tin với kích thước nhiều hơn kích thước giới hạn.
- Tham số không hợp lệ.

1. Black box Testing

Ví dụ về kiểm thử đoán lỗi: Giả sử có một yêu cầu rằng số điện thoại di động phải là số và không ít hơn 10 số và PM ứng dụng cần đáp ứng yêu cầu này. Sau đây là kỹ thuật đoán lỗi:

- Các số điện thoại di động không để trống, có chứa khoảng trống?
- Có bất kỳ ký tự nào (không phải số) được nhập không?
- Có ít hơn hoặc nhiều hơn10 chữ số được nhập?

1. Black box Testing

1.3.9/. Kỹ thuật kiểm thử Comparison Testing :

Các phiên bản độc lập khác nhau của cùng một phần mềm (hoặc các phần mềm tương tự) được sử dụng để so sánh với nhau để kiểm thử so sánh theo phương pháp này. Đối tượng kiểm thử có thể là bất cứ thứ gì như:

- Một ứng dụng Web
- Ứng dụng ERP
- Ứng dụng CRM
- Một Module của một ứng dụng yêu cầu xác thực dữ liệu sau khi hoàn thành giao dịch, v.v.

1. Black box Testing

Các giai đoạn kiểm thử so sánh: có thể được thực hiện trong hai giai đoạn riêng biệt:

1) So sánh sp phần mềm với các tiêu chuẩn hoặc điểm chuẩn đã biết.
2) So sánh sp phần mềm với các tính năng cụ thể của các phần mềm hiện có khác (phần mềm tương tự).

Ví dụ: nếu ứng dụng TM điện tử được kiểm thử, chúng ta biết rằng có nhiều ứng dụng TM điện tử , ứng dụng nào cũng có các chức năng xử lý đặt hàng, nắm thông tin khách hàng, giỏ hàng và các vấn đề khác …

1. Black box Testing

Giai đoạn kiểm thử 1: chúng ta có thể kiểm tra chức năng của ứng dụng TMĐT theo các tiêu chuẩn và chức năng đã biết như hiện có trên thị trường tại thời điểm kiểm thử như: liệt kê, tìm kiếm, chọn lựa, chăm sóc khách hàng, thống kê,...

Giai đoạn kiểm thử 2: chúng ta có thể so sánh các tính năng của ứng dụng TMĐT so với các tính năng của các sản phẩm TMĐT khác trên thị trường (Lazada, Sendo,..).

2. White box Testing

2.1/. Giới thiệu: **Kiểm t**hử White box gọi là kiểm thử hộp trắng nhằm kiểm tra mã nguồn của phần mềm, gồm có:

- Xác minh các lỗ hổng thiếu sót, khiếm khuyết trong các mã nguồn.
- Kiểm tra các đường dẫn (Path) bị hỏng hoặc không đầy đủ trong mã nguồn.
- Kiểm tra dòng chảy của cấu trúc đề cập đến trong tài liệu đặc tả.

2. White box Testing

- Kiểm tra xem có các dead code (mã chết) trong mã nguồn hay không? (Sv thảo luận về dead code)
- Kiểm tra các kết quả đầu ra có như mong đợi?
- Xác minh từng dòng hoặc phần của các mục trong mã nguồn & bao phủ các phân nhánh xử lý.
- Kiểm tra các vòng lặp, các điều kiện trong các mã nguồn có thực hiện đúng không?

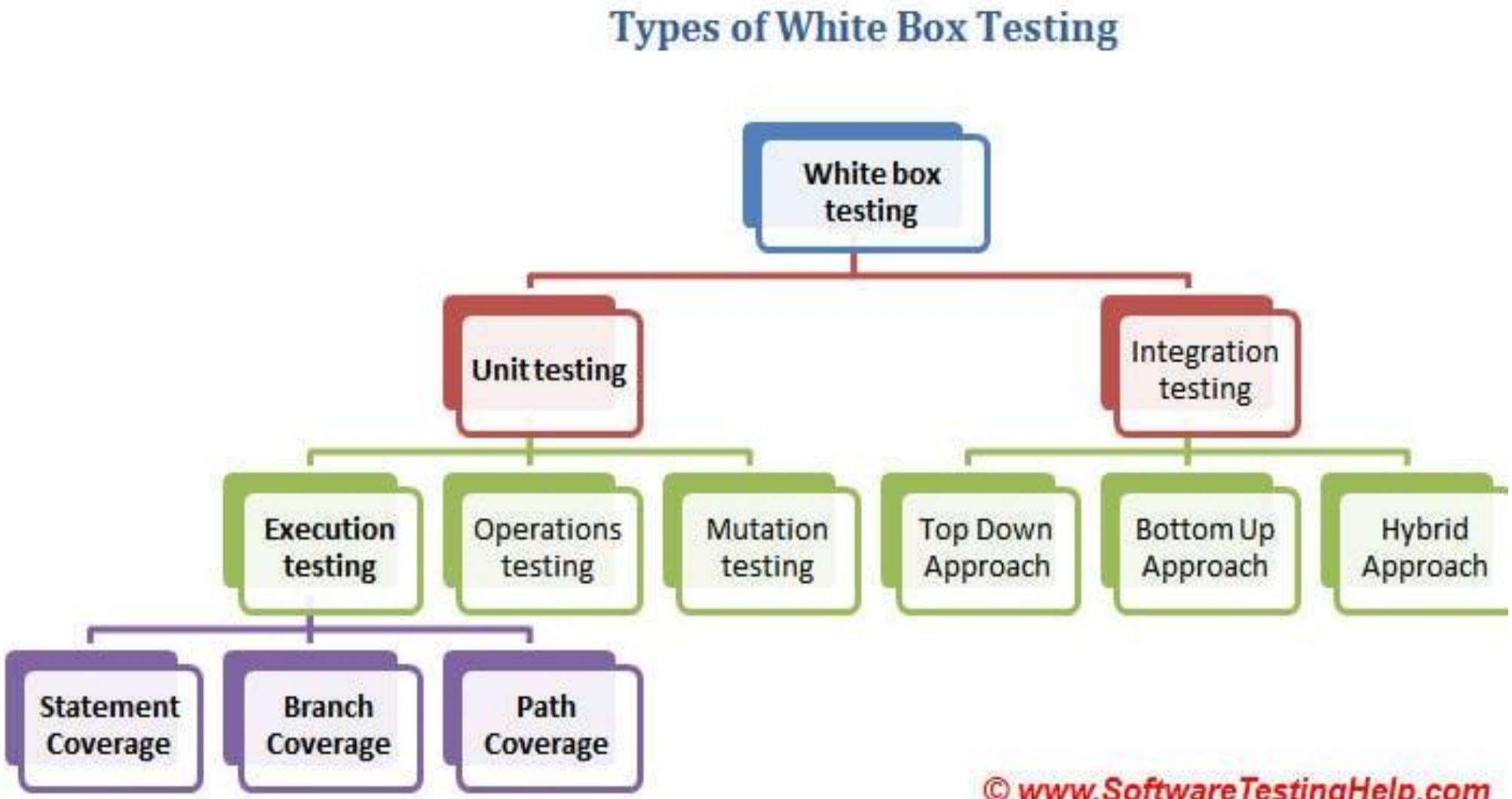
2. White box Testing

Có cần đến kỹ năng ngôn ngữ lập trình không ?

- Chúng ta cần viết các Test case để đảm bảo phạm vi bao phủ hoàn toàn của logic chương trình.
- Đối với điều này, chúng ta cần biết rõ về chương trình, tức là chúng ta nên biết đặc tả kỹ thuật và mã lệnh sẽ được kiểm tra. Kiến thức về ngôn ngữ lập trình và logic là cần thiết cho loại kiểm thử này.

Các loại kiểm thử hộp trắng: sinh viên thảo luận (Slide kế)

2. White box Testing



2. White box Testing

2.2/. Dead code: là các dòng lệnh có thể không bao giờ thực hiện hoặc gây ra kết thúc chương trình không mong muốn.

```
public void howToDoInJava_method1() {
    System.out.println("how to do");
    return;
    System.out.println("in java");
}

public void howToDoInJava_method2() {
    System.out.println("how to do");
    if (true) {
        return;
    }
    System.out.println("in java");
}
```

Add control flow to example:

```
x = y + 1;
y = 2 * z;
if (d)  x = y+z;
z = 1;
z = x;
```

Is 'x = y+1' dead code? Is 'z = 1' dead code?

2. White box Testing

2.3/. Tạo đồ thị CFG _Control Flow Graph: gồm các nút và cạnh.

- Có 03 loại nút:
  - Nút lệnh: mô tả nút vào, ra, tuần tự.
  - Nút điều kiện: nút mô tả điều kiện cho 1 nhánh.
  - Nút hỗ trợ: nút kết nối như IF, …
- Cạnh: biểu diễn các luồng điều khiển.

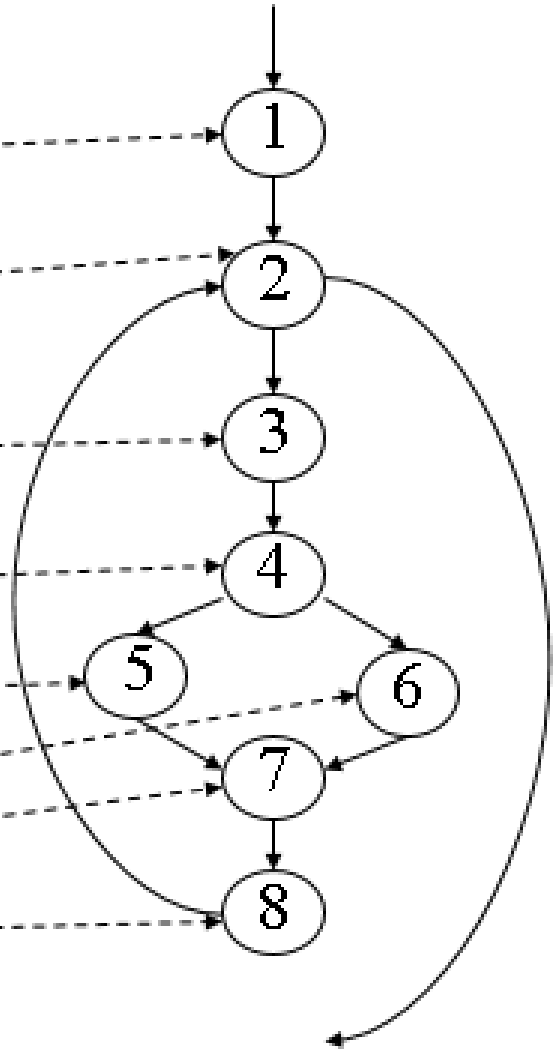
Rất dễ khi tạo đồ thị, nó tương tự như vẽ lưu đồ chương trình.

2. White box Testing

Ví dụ:

Sinh viên thảo luận

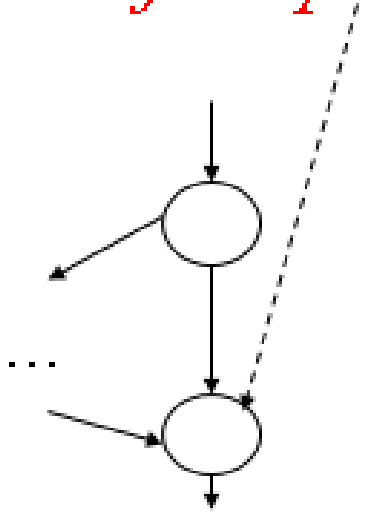
```
s:=0;
d:=0;
while (x<y) {
    x:=x+3;
    y:=y+2;
    if (x+y < 100)
        s:=s+x+y;
    else
        d:=d+x-y;
}
```



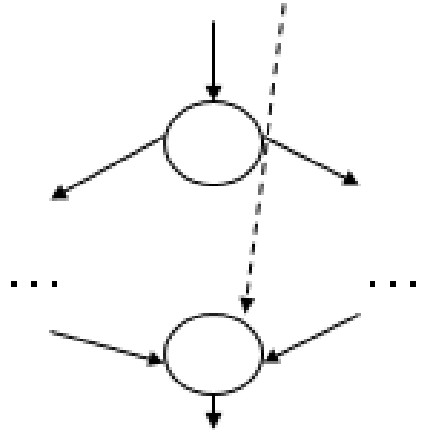
2. White box Testing

+ Biểu diễn IF-THEN, IF-THEN-ELSE, SWITCH:

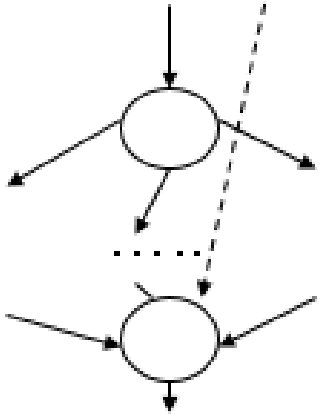
```
if (c)
   then
   // join point
```



```
if (c)
   then
else
   // join point
```



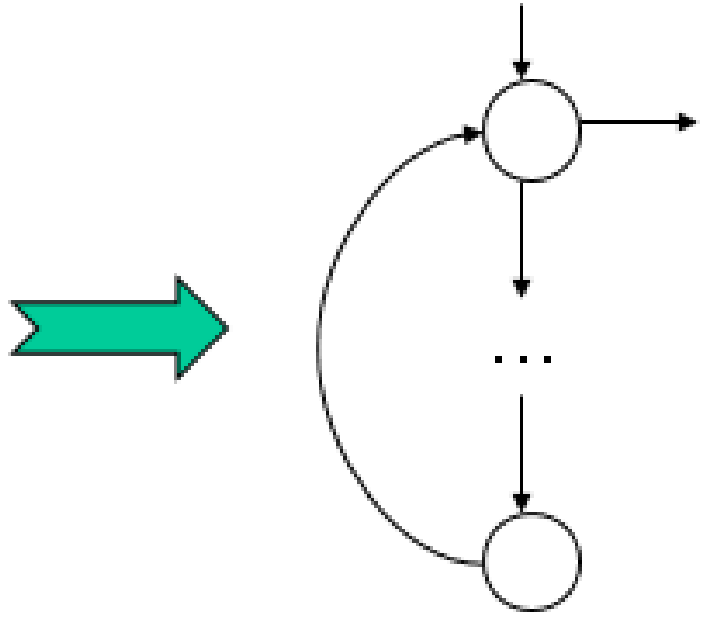
```
switch (c)
   case 1:
   case 2:
   // join point
```



2. White box Testing

+ Biểu diễn vòng lặp (Loop):

```
while (c)
{
        …
}
```



Chú ý: Các vòng lặp khác như FOR, DO - WHILE được biểu diễn tương tự.

2. White box Testing

+ Tạo các khối lệnh (Block statement):

- Để đơn giản hóa các mã nguồn lớn có nhiều lệnh, người ta thường tạo CFG bằng các khối lệnh thay vì cho từng lệnh.

- Khối lệnh là tập hợp các lệnh liên tiếp tuần tự, không có phân nhánh (trừ ở cuối), không chứa vòng lặp.

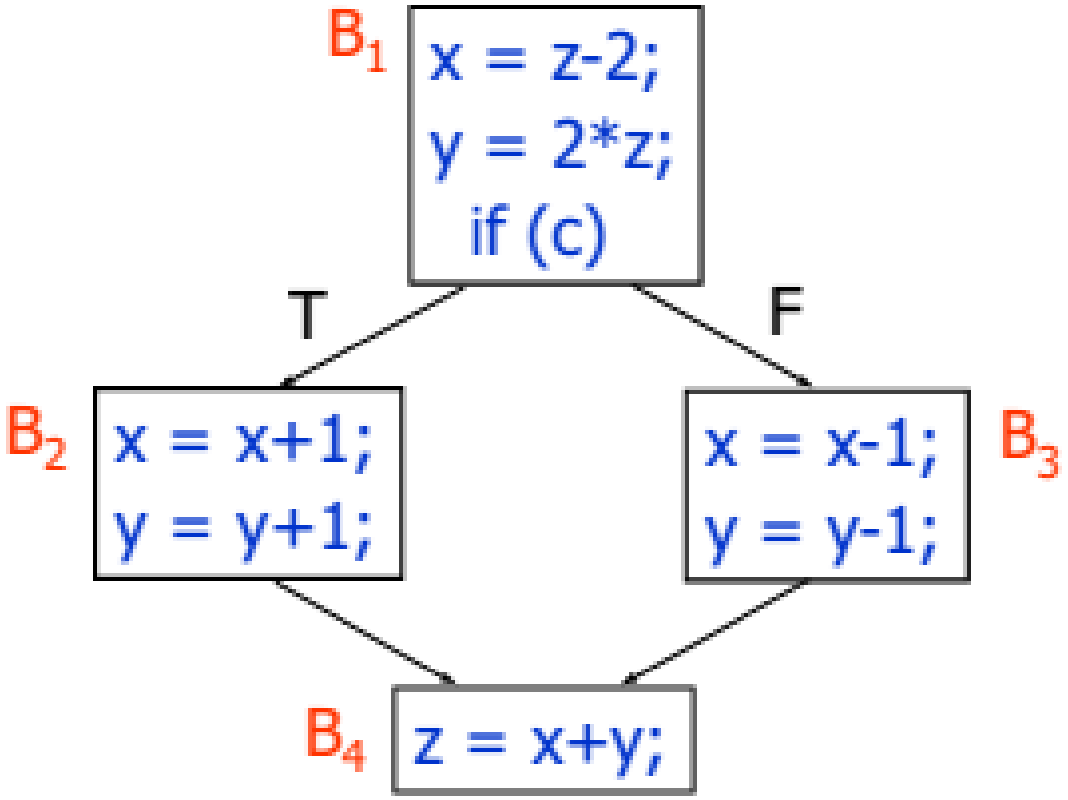
2. White box Testing

Ví dụ:

Program

```
x = z-2 ;
y = 2*z;
if (c)  {
    x = x+1;
    y = y+1;
}
else {
    x = x-1;
    y = y-1;
}
z = x+y;
```

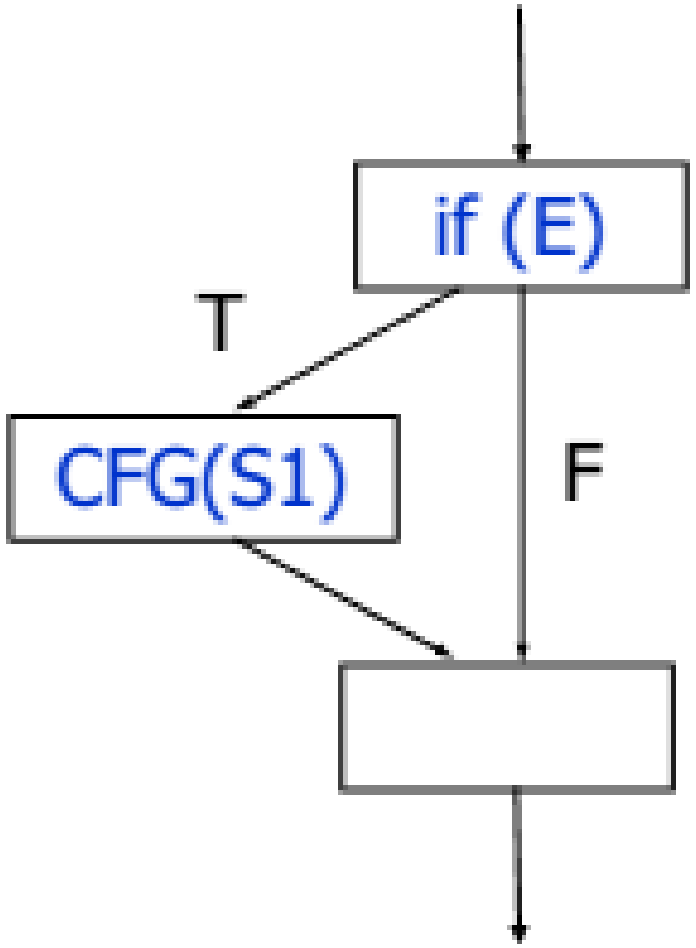
Control Flow Graph



2. White box Testing

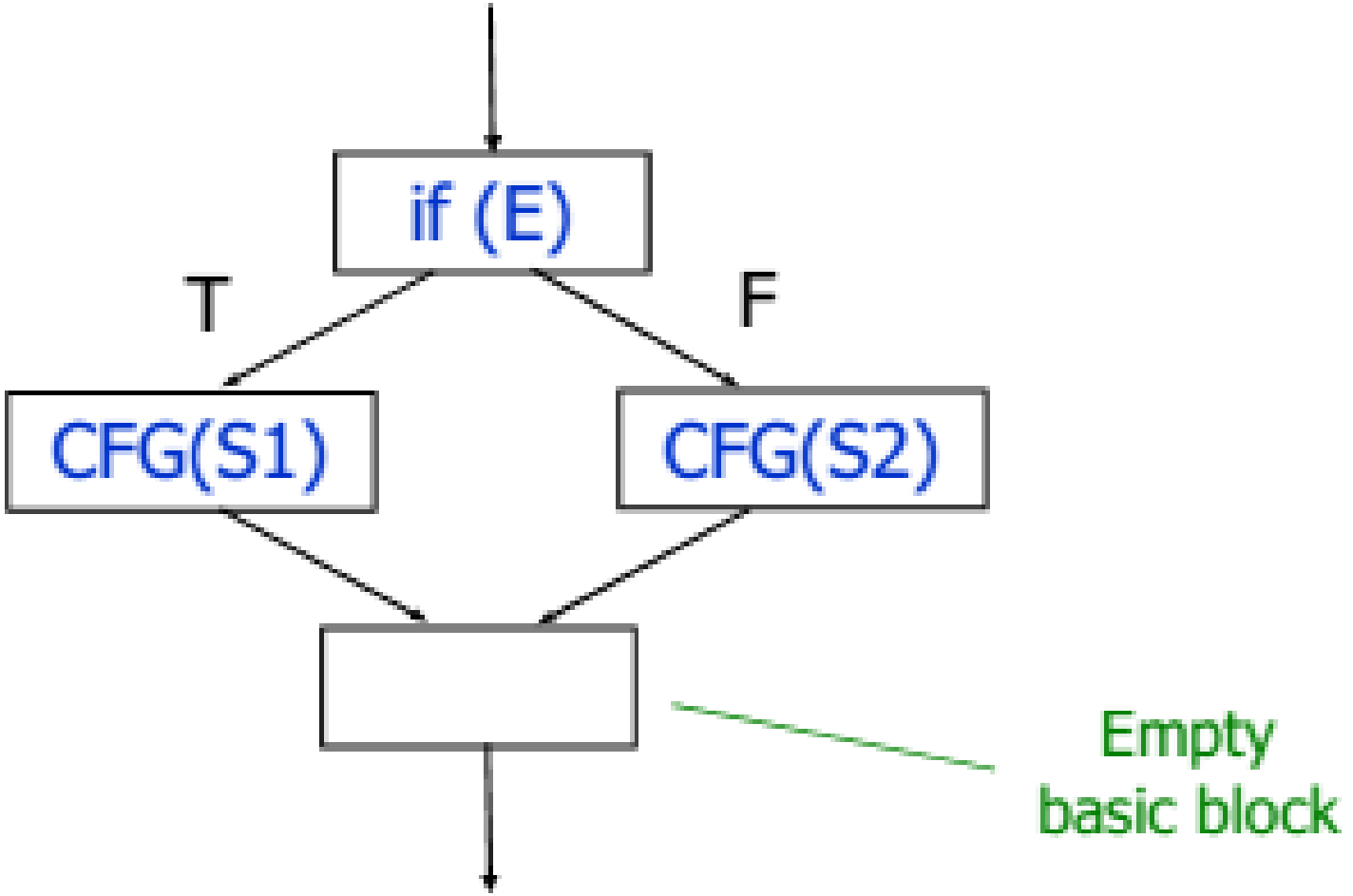
Ví dụ:

CFG(if (E) S)



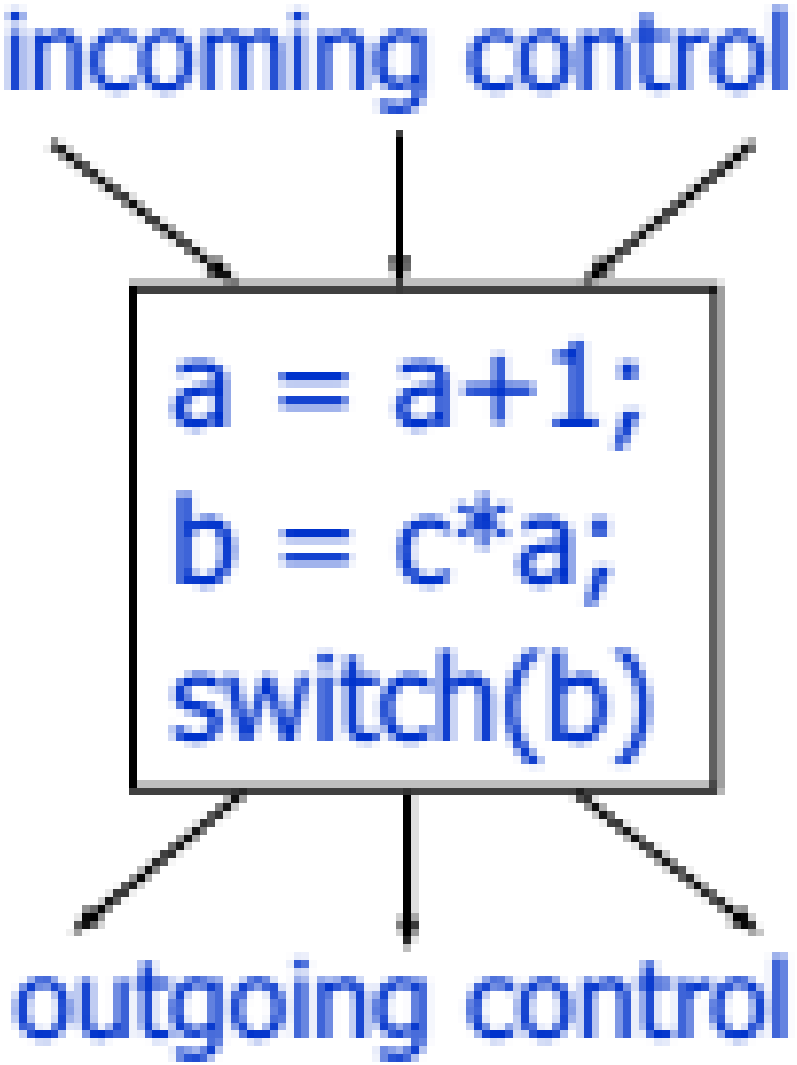
2. White box Testing

Ví dụ: CFG (if (E) S1 else S2)



2. White box Testing

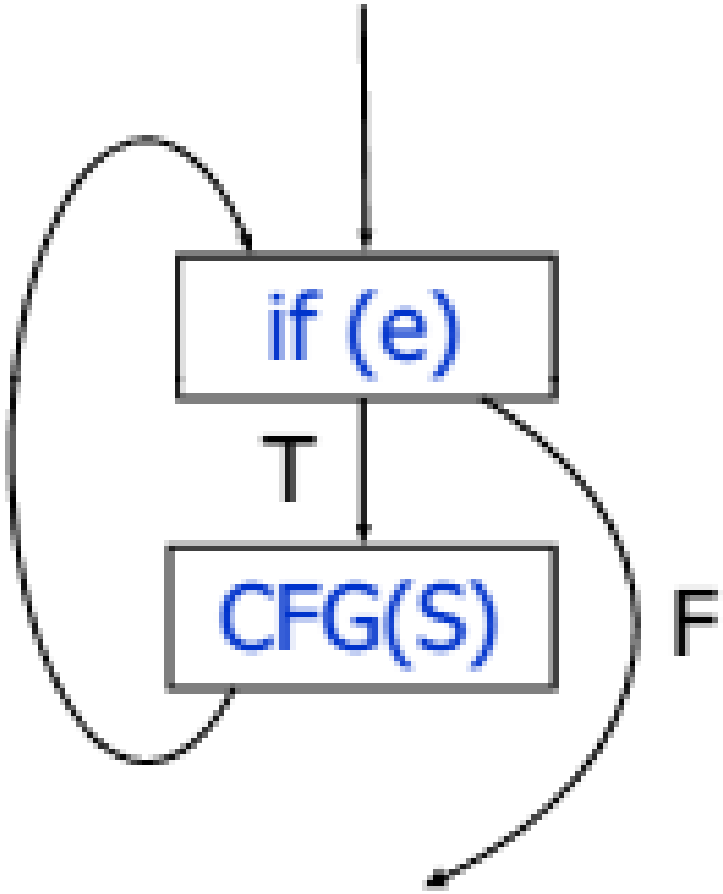
Ví dụ:



2. White box Testing

Ví dụ:

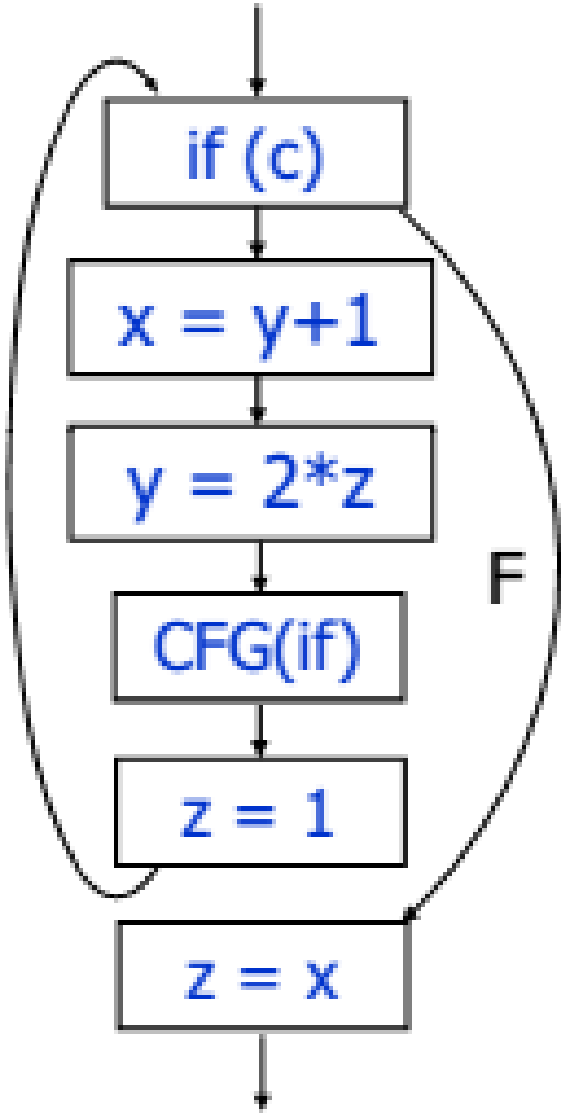
CFG for: while (e) S



2. White box Testing

Ví dụ:

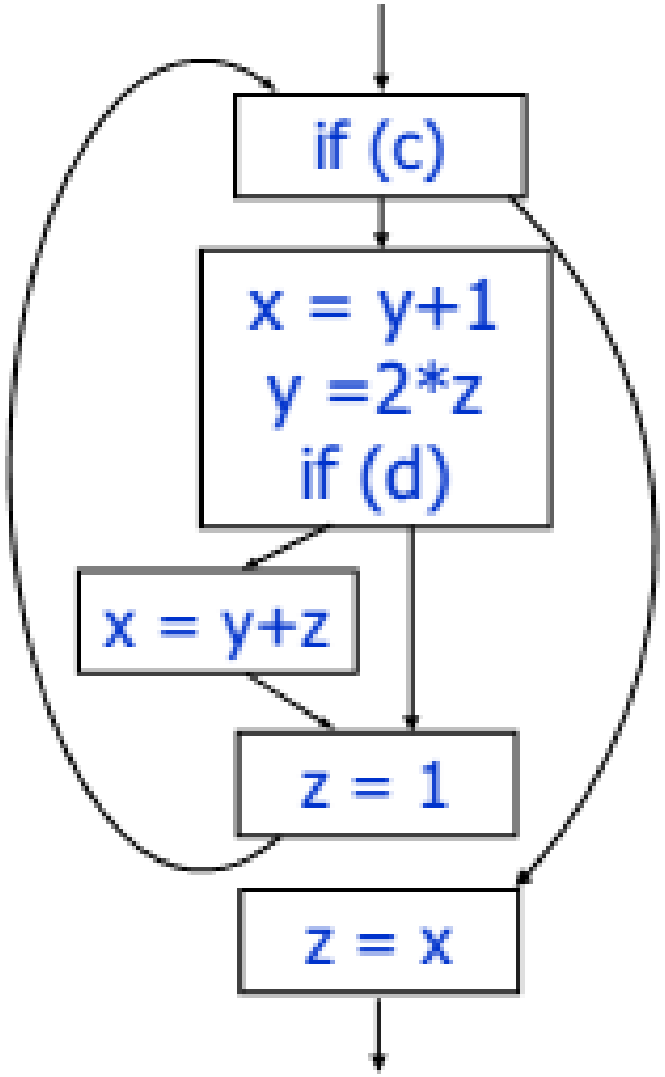
```
while (c) {
        x = y + 1;
        y = 2 * z;
        if (d)  x = y+z;
        z = 1;
}
z = x;
```



2. White box Testing

Ví dụ:

```
while (c) {
        x = y + 1;
        y = 2 * z;
        if (d)  x = y+z;
        z = 1;
}
z = x;
```



2. White box Testing

2.4/. Các kỹ thuật kiểm thử hộp trắng:

Có 03 kỹ thuật kiểm thử White box sau:

- Statement Coverage (phủ lệnh)
- Condition Coverage / Branch Coverage (phủ điều kiện / nhánh)
- Path Coverage (phủ đường đi)

2. White box Testing

A/. Statement Coverage:

- Lệnh là các dòng mã hoặc chỉ thị hướng dẫn cho máy tính hiểu và hành động phù hợp. Một lệnh sẽ trở thành một lệnh thực thi khi nó được biên soạn và chuyển đổi thành mã đối tượng và thực hiện các hành động khi chương trình ở chế độ chạy.
- Do đó '**Phủ lệnh**', cho thấy đó là kỹ thuật / phương pháp chứng thực rằng mỗi dòng lệnh được thực hiện ít nhất một lần.

2. White box Testing

B/. Condition Coverage (Branch Coverage):

- Lệnh If trong NNLT có điều kiện và rẽ nhánh: đúng và sai, hoặc các lệnh chuyển hướng điều khiển trong mã nguồn như switch(), goto, …
- Trong phạm vi phủ điều kiện (Condition Coverage _ còn gọi là phủ quyết định), chúng ta xác nhận rằng mỗi điều kiện được thực hiện ít nhất một lần.

2. White box Testing

B/. Condition Coverage: (tiếp theo)

Trong trường hợp của một lệnh **IF**, sẽ có hai điều kiện kiểm thử:

- Một để xác nhận nhánh đúng và
- Cái khác để xác nhận nhánh sai.

Do đó về lý thuyết, Phủ điều kiện là một phương pháp kiểm thử mà khi thực hiện đảm bảo rằng từng điều kiện của lệnh (có chứa quyết định) sẽ được thi hành.

2. White box Testing

C/. Path Coverage:

Phủ đường đi là kiểm tra tất cả các đường đi của chương trình. Đây là một kỹ thuật toàn diện, đảm bảo rằng tất cả các đường đi của chương trình được đi qua ít nhất một lần. Phủ đường đi thậm chí còn mạnh mẽ hơn phủ điều kiện. Kỹ thuật này rất hữu ích để kiểm thử các chương trình phức tạp.

Hãy lấy một ví dụ đơn giản để hiểu tất cả các kỹ thuật kiểm thử hộp trắng.

2. White box Testing

Xem xét đoạn Mã giả (pseudo code**) sau:**

1	INPUT A & B
2	C = A + B
3	IF C>100
4	PRINT “IT’S DONE”

2. White box Testing

* Đối với Statement Coverage: ta chỉ có một trường hợp kiểm thử để kiểm tra tất cả các dòng lệnh.

- Xét Test Case_01 với (A = 40 và B = 70), ta thấy tất cả các dòng mã sẽ được thực thi
- Nếu ta xét trường hợp kiểm thử Test Case_02 với (A = 33 và B = 45) sẽ ra sao?

Bởi vì Statement Coverage sẽ chỉ bao trùm bên phía True tức là chỉ có một trường hợp thử nghiệm vậy sẽ không đủ để kiểm tra nó. Là một Tester, chúng ta cần phải xem xét cả trường hợp False.

2. White box Testing

Bài tập: Xem xét đoạn Mã giả (pseudo code) **khác như sau:**

1	INPUT A & B
2	C = A + B
3	IF C>100
4	PRINT “IT’S DONE”
5	ELSE
6	PRINT “IT’S PENDING”

Sinh viên kiểm thử bằng kỹ thuật Condition, Path Testing.

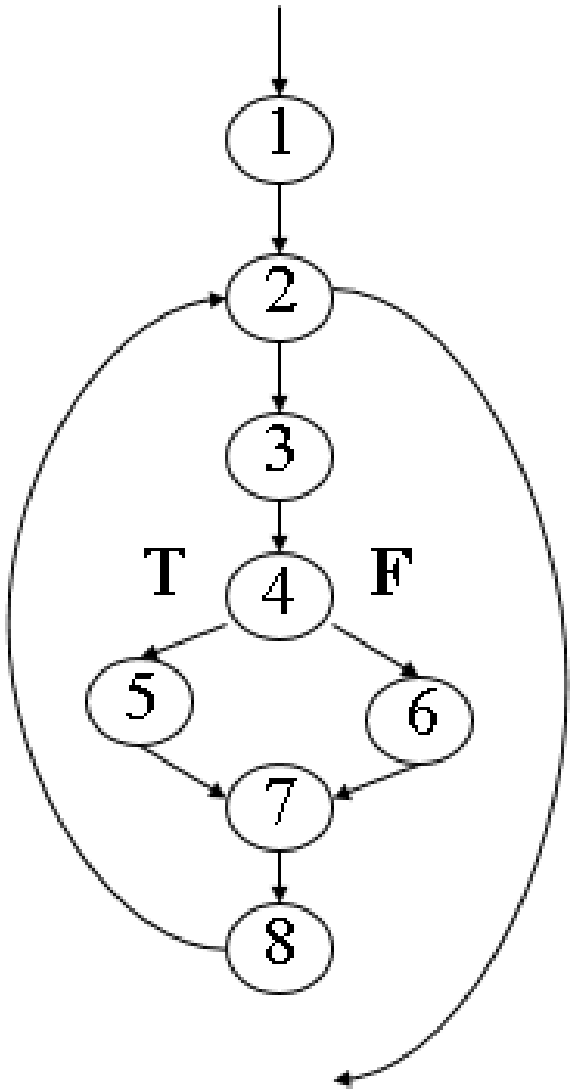
2. White box Testing

Bài tập: Xem xét hình sau:

Giả sử, chúng ta viết và thực hiện 2 Test case:

- Test case #1: 1-2-exit (không bao giờ vào vòng lặp)
- Test case #2: 1-2-3-4-5-7-8-2-3-4-5-7-8-2-exit (vòng lặp thực hiện 2 lập và cả 2 đi theo nhánh T (true))

Chúng ta đã phủ 100% lệnh chưa ? (Sv trả lời)



2. White box Testing

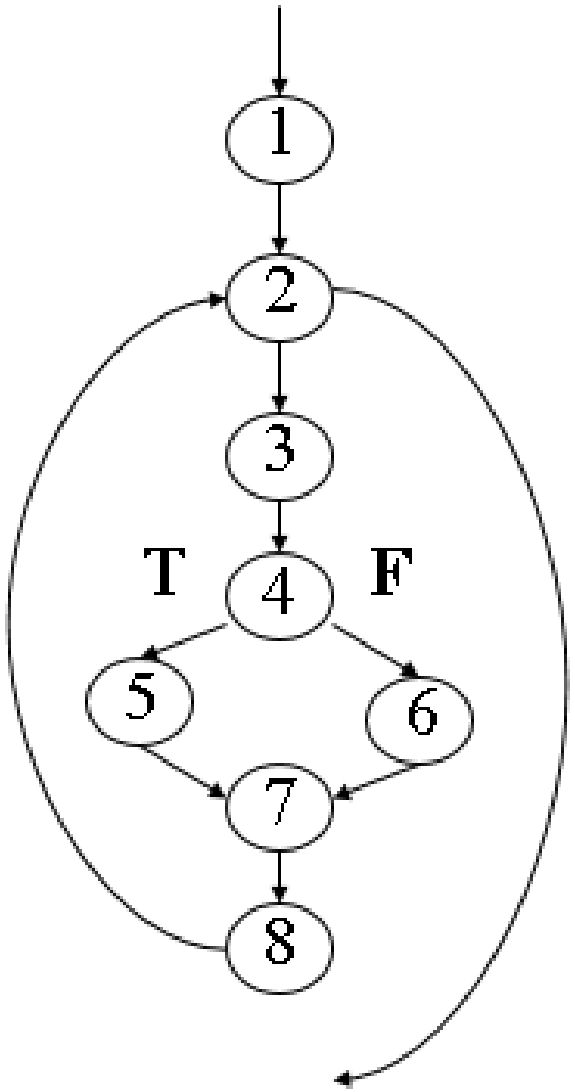
Bài tập: Sử dụng lại bài tập ở Slide trước:

Dùng kỹ thuật Condition Coverage, chúng ta thực hiện 2 Test case:

- Test case #1: nhánh 1: 1-2-exit
- Test case #2: nhánh 2: 1-2-3-4-5-7-8-2-3-4-5-7-8-2-exit

Chúng ta đã phủ được tất cả các nhánh chưa ?

(Sv trả lời)

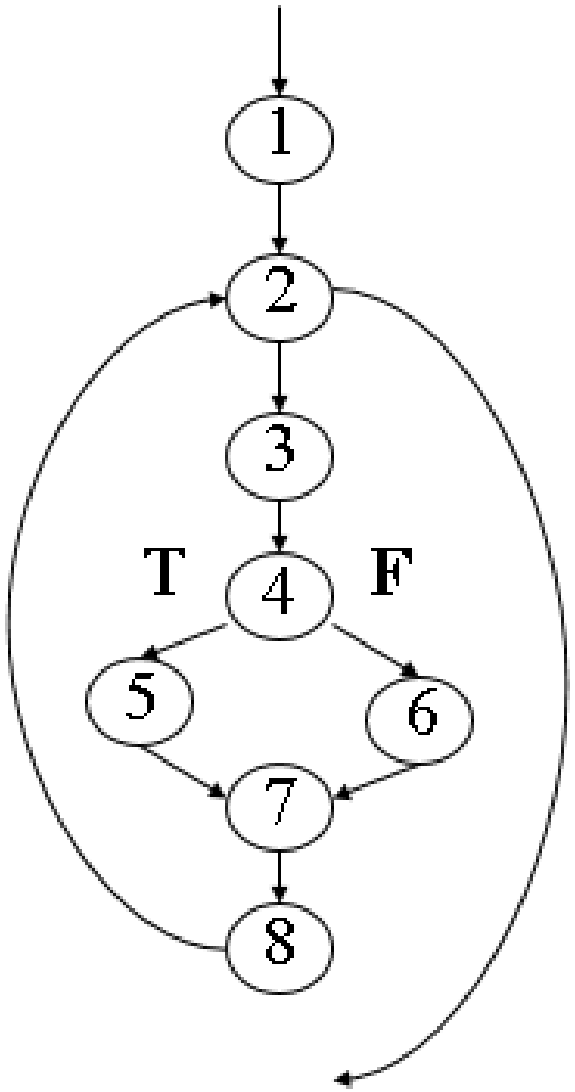


2. White box Testing

Bài tập: Sử dụng lại bài tập ở Slide trước:

Dùng kỹ thuật Path Coverage, chúng ta thực hiện 2 Test case:

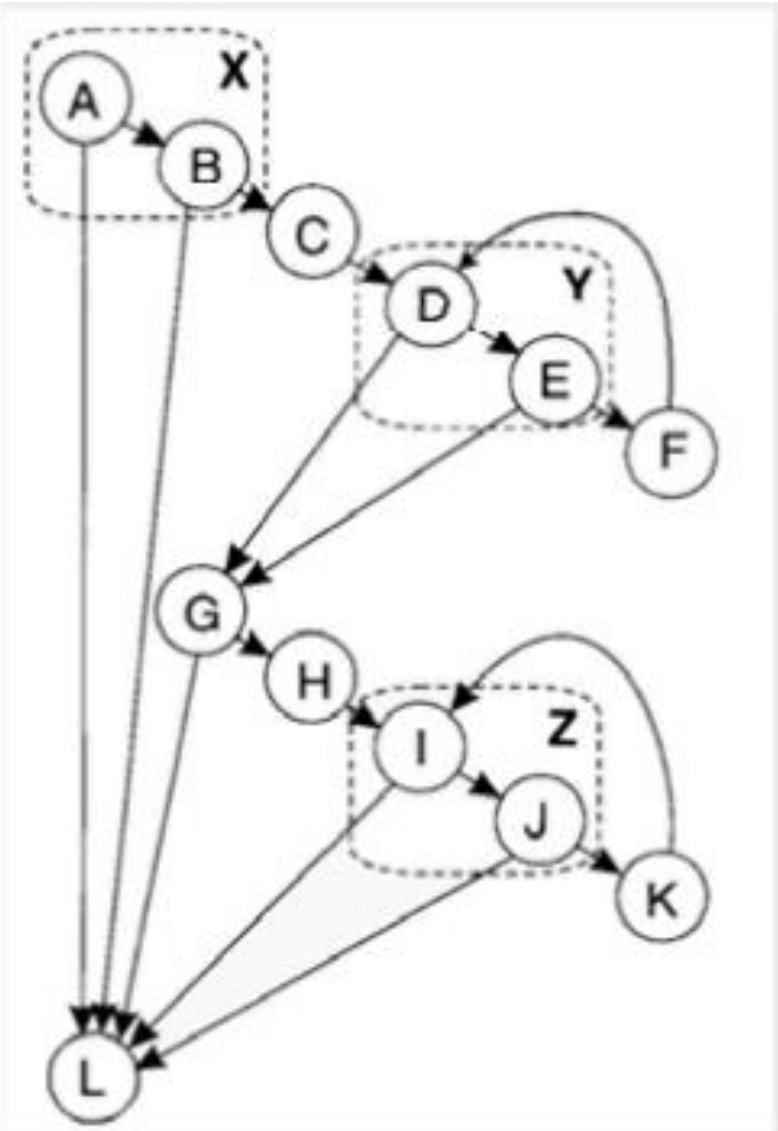
- Đường đi 1: 1-2-exit (không thực hiện vòng lặp)
- Đường đi 2: Ta muốn bao gồm cạnh 2-3, vậy phải chọn 1-2-3-4-5-7-8-2-exit



Đường đi thứ 3 phải chọn thế nào? (Sv trả lời)

2. White box Testing

Bài tập:



AL
ABL
ABCDGL
ABCDEGL
ABC(DEF)*DGL
ABC(DEF)*DEGL
ABCDGHIL
ABCDGHIJL
ABCDGH(IJK)*IL
ABC(DEF)*DEGH(IJK)*IJL

2. White box Testing

Kinh nghiệm:

- Làm thế nào chúng ta có thể chọn các đường đi bao phủ 100% các nhánh?

- Có thể không có con đường nào đi từ Start → End, vì vậy cần phải chọn nhiều con đường đi khác nhau để bao phủ tất cả các nhánh, các nút.

2. White box Testing

2.5/. Kiểm thử Vòng lặp:

Mục tiêu của Kiểm tra vòng lặp:

- Để khắc phục / phát hiện sự cố lặp lại vòng lặp vô hạn.
- Để biết hiệu suất chương trình.
- Để xác định các vấn đề khởi tạo vòng lặp.
- Để xác định các biến chưa được khởi tạo.

2. White box Testing

2.5/. Kiểm thử Vòng lặp:

- Condition coverage, Path coverage có thể không đủ để kiểm tra thực hiện các vòng lặp.
  - Nghĩa là có 2 kịch bản: Một là vòng lặp không thực hiện lần nào; Một là vòng lặp thực hiện ít nhất 1 lần.
- Thường xảy ra nhiều lỗi trong vòng lặp với điều kiện / giá trị tại biên vậy cần có sự kết hợp.
- Kiểm thử vòng lặp tập trung vào kiểm chứng tính hợp lệ của các vòng lặp số lần typical (kinh nghiệm).

2. White box Testing

Giả sử **min** là số lần lặp tối thiểu, **max** là số lần lặp đối đa của vòng lặp.

Cần viết các Test case sau: (5 test case)

- 1 Test case thực hiện lặp **min** lần,
- 1 Test case thực hiện lặp **min** + **1** lần.
- 1 Test case thực hiện 1 số lần lặp tiêu biểu (typical).
- 1 Test case thực hiện lặp **max** **-1** lần,
- 1 Test case thực hiện lặp **max** lần. **(tiếp ở đây)**

2. White box Testing

Với mỗi vòng lặp, chúng ta sẽ muốn kiểm thử 07 trường hợp sau

1. Vượt qua vòng lặp (không lặp lần nào)
2. Lặp 1 lần
3. Lặp 2 lần
4. **Lặp 1 số lần** ("typical" number)
5. Lặp (Max-1) lần ← **Có thể không thực hiện được**
6. Lặp (Max) lần ← **Có thể không thực hiện được**
7. Lặp (Max + 1) lần ← **Có thể không thực hiện được**

2. White box Testing

2.6/. Kiểm thử Vòng lặp lồng nhau:

Đối với vòng lặp lồng nhau, cho mỗi lần lặp của vòng lặp bên ngoài (Outer loop), chúng ta có một số lần lặp của vòng lặp bên trong (Inner loop):

- Thực hiện kiểm thử từ trong ra ngoài (bắt đầu từ vòng lặp trong nhất)
- Vòng lặp trong có m lần lặp,Vòng lặp ngoài có n lần lặp:

(0 x n

Như vậy ện kiểm thử, là q

Kết luận: nếu có nhiều vòng lặp lồng nhau và số lần thực hiện nhiều thì nên chọn phương pháp kiểm thử kết hợp để đạt sự hiệu quả hơn.

2. White box Testing

Ví dụ: Nhìn vào whilevòng lặp vô hạn sau trong Java. Nó gây ra lỗi thời gian biên dịch cho câu lệnh bên dưới nó.

```
while(true)
{
        System.out.println("inside while");
}
System.out.println("while terminated");
//Unreachable statement - compiler-error.
```

2. White box Testing

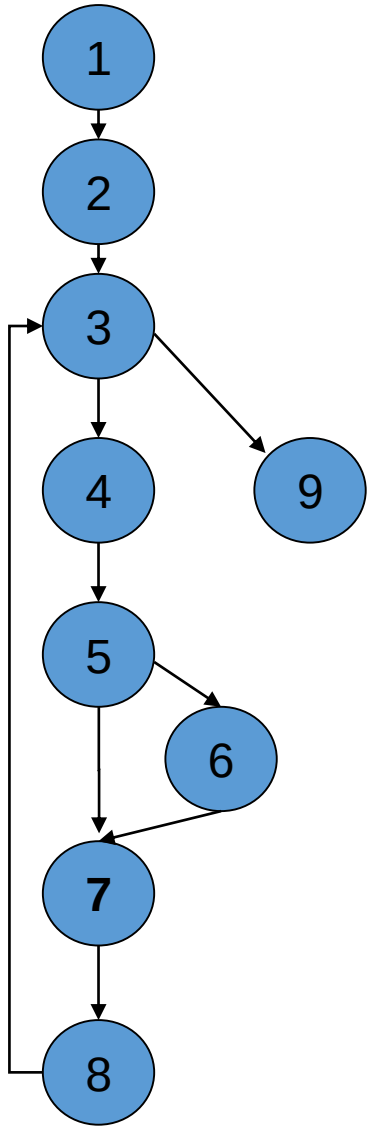
Ví dụ: Tuy nhiên, cùng một vòng lặp vô hạn sau đây hoạt động tốt và không gây ra bất kỳ lỗi nào (thay thế điều kiện = biến Boolean).

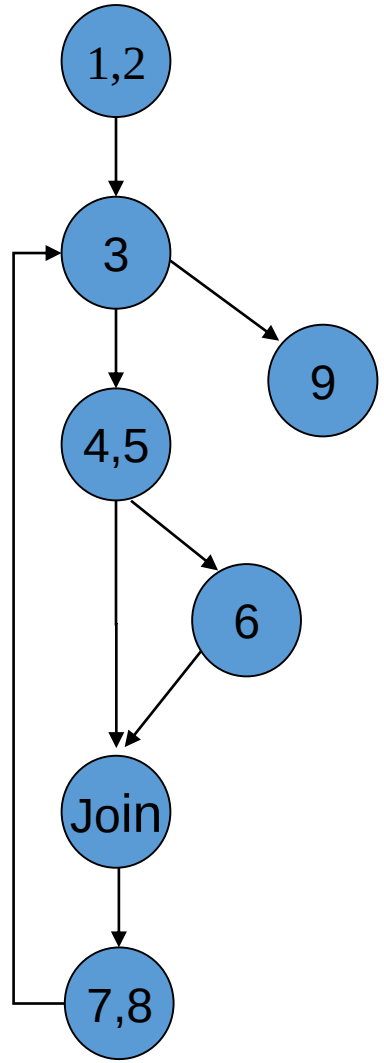
```
boolean b=true;
while(b)
{
        System.out.println("inside while");
}
System.out.println("while terminated"); //No error here.
```

2. White box Testing

Bài tập: Tạo CFG cho source code sau

```
1. read (result);
2. read (x,k)
3. while result < 0 {
4.   ptr ← false
5.   if x > k then
6.         ptr ← true
7.    x ← x + 1
8.    result ← result + 1
}
9. print result
```

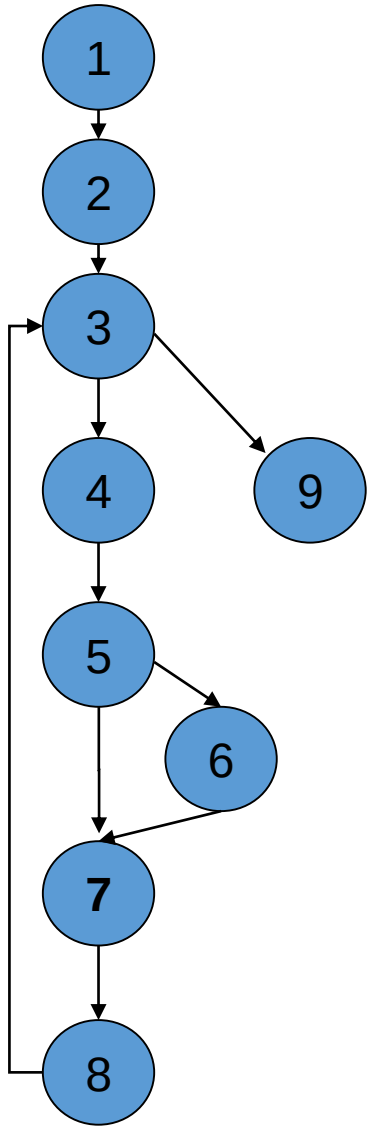


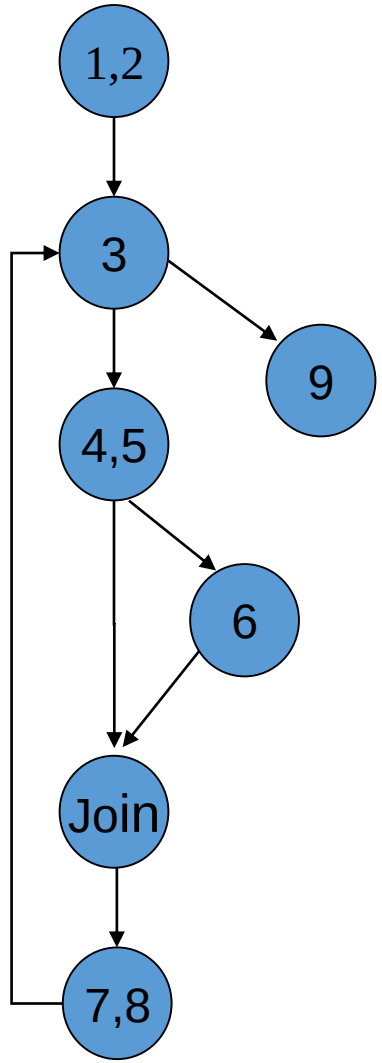


2. White box Testing

Bài tập: Tạo CFG cho source code sau

```
1. read (result);
2. read (x,k)
3. while result < 0 {
4.   ptr ← false
5.   if x > k then
6.         ptr ← true
7.    x ← x + 1
8.    result ← result + 1
}
9. print result
```





2. White box Testing

Bài tập: Tạo CFG cho source code sau

EX 1

```
1.  if (a < b) then
2.       while (a < n)
3.            a ← a + 1;
4.   else
5.       while (b < n)
6.            b ← b + 1;
```

EX 2

```
1. read (a, b);
2. if (a ≠ 0 && b ≠ 0) then
{
3.     c ← a + b;
4.  if c > 10 then
5.         c ←  max;
6.  else c ←  min; }
7. else  print 'Done';
```

EX 3

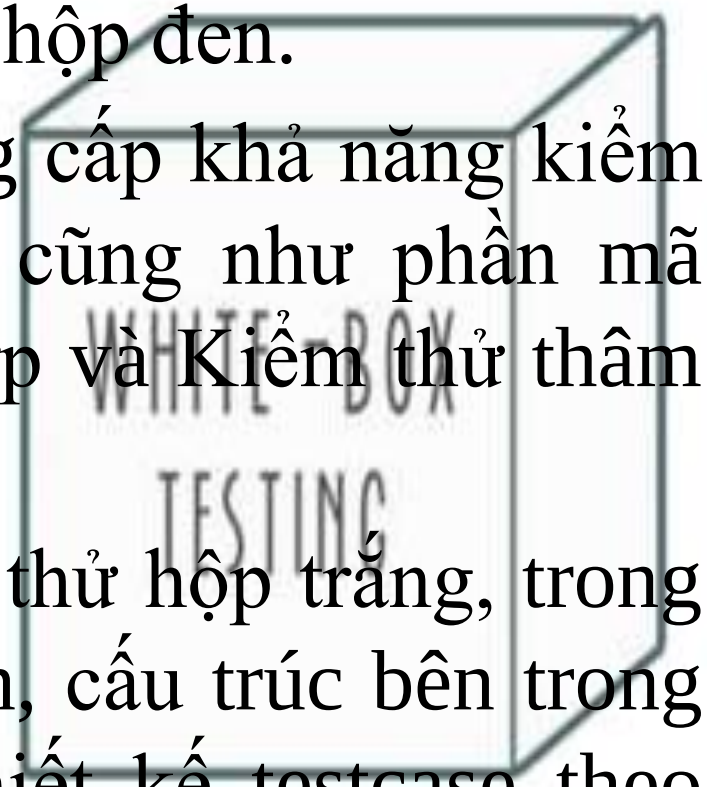
```
1. read (a, b);
2. if (a ≠ 0 || b ≠ 0) then
{
3.     c ← a + b;
4.     while( c < 100)
5.           c ← a + b; }
6.  c ← a * b;
```

3. Grey box Testing (Gray box Testing)

Kiểm thử hộp xám là phương pháp kiểm thử phần mềm, có sự kết hợp của cả phương pháp kiểm tra hộp trắng và kiểm thử hộp đen.

Trong kỹ thuật phần mềm, kiểm thử hộp xám cung cấp khả năng kiểm tra [illegible] một ứng dụng, lớp trình bày cũng như phần mã ng[illegible] hữu ích trong Kiểm thử tích hợp và Kiểm thử thâm nh[illegible]ting và Penetration Testing).

Là [illegible]ữa kiểm thử hộp đen và kiểm thử hộp trắng, trong đó [illegible]ụng các kiến thức về thuật toán, cấu trúc bên trong chương trình, ... như của hộp trắng nhưng để thiết kế testcase theo hướng người sử dụng hoặc có testcase như của hộp đen.



3. Grey box Testing

Các Yêu cầu kiểm thử hộp xám:

Để thực hiện kiểm tra hộp xám, mà người kiểm thử không có quyền truy cập vào mã nguồn. Một bài kiểm thử được thiết kế dựa trên kiến thức về thuật toán, kiến trúc, trạng thái bên trong hoặc các mô tả cao cấp khác về hành vi của chương trình.

Để thực hiện Kiểm tra hộp xám:

- Test case cần dựa vào yêu cầu và nội dung Source Code (biết và hiểu Code của PM)
- Thực hiện Test trên giao diện của chương trình (yêu cầu PM chạy được mới test được, không can thiệp vào code).

3. Grey box Testing

Các Test case để kiểm thử hộp xám có thể bao gồm và liên quan đến: Giao diện, bảo mật, cơ sở dữ liệu, trình duyệt, hệ điều hành, …

3. Grey box Testing

Các Kỹ thuậ

A/. Kiểm thử

- Đây là loại l tại trong hệ thống. Và các dữ liệu đầu vào cho sách để đảm bảo rằn
- Tham khảo

Table II: Requirements Traceability Matrix

	P_w	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9
R1	2	X	X							
R2	2		X							
R3	2	X								
R4	2		X	X						
R5	2				X					
R6	1		X				X			
R7	1		X					X		
R8	1		X							X
R9	1	X			X	X				
R10	1		X						X	
SCORE		5	10	2	3	1	1	1	1	1

3. Grey box Testing

Các Kỹ thuật kiểm thử hộp xám: (tiếp theo)

B/. Kiểm thử hồi quy (Regression Testing)

- Khi thêm 1 chức năng mới (có 1 sửa đổi / 1 cập nhật) cho phần mềm, để đảm bảo chức năng mới này không làm ảnh hưởng đến các chức năng khác trong hệ thống, chúng ta cần phải cân nhắc việc kiểm tra lặp lại những Test case đã từng kiểm thử rồi. Lúc này việc kiểm thử lặp lại được gọi là kiểm thử hồi quy.

3. Grey box Testing

C/. Kiểm thử mẫu (Pattern Testing)

- Kỹ thuật này được thực hiện trên lịch sử dữ liệu của các lỗi hệ thống trước đó. Không giống như thử nghiệm hộp đen, kiểm thử hộp xám đào sâu vào trong mã nguồn và xác định lý do tại sao sự cố xảy ra.

D/. Kiểm thử mảng trực giao (Orthogonal Array Testing or OAT)

- Đây như là một kỹ thuật thống kê để tạo ra hoán vị các đầu vào, tạo ra các test case có phạm vi kiểm thử tối thiểu để giảm công sức của con người trong giai đoạn lập kế hoạch kiểm thử và thiết kế Test case.

3. Grey box Testing

Các bước thực hiện Kiểm thử hộp xám:

- **Step 1**: Identify inputs
- **Step 2**: Identify the outputs
- **Step 3**: Identify the major paths
- **Step 4**: Identify Subfunctions
- **Step 5**: Develop inputs for Subfunctions
- **Step 6**: Develop outputs for Subfunctions
- **Step 7**: Execute test case for Subfunctions
- **Step 8**: Verify the correct result for Subfunctions
- **Step 9**: Repeat steps 4 & 8 for other Subfunctions
- **Step 10**: Repeat steps 7 & 8 for other Subfunctions

3. Grey box Testing

Các bước thực hiện Kiểm thử hộp xám:

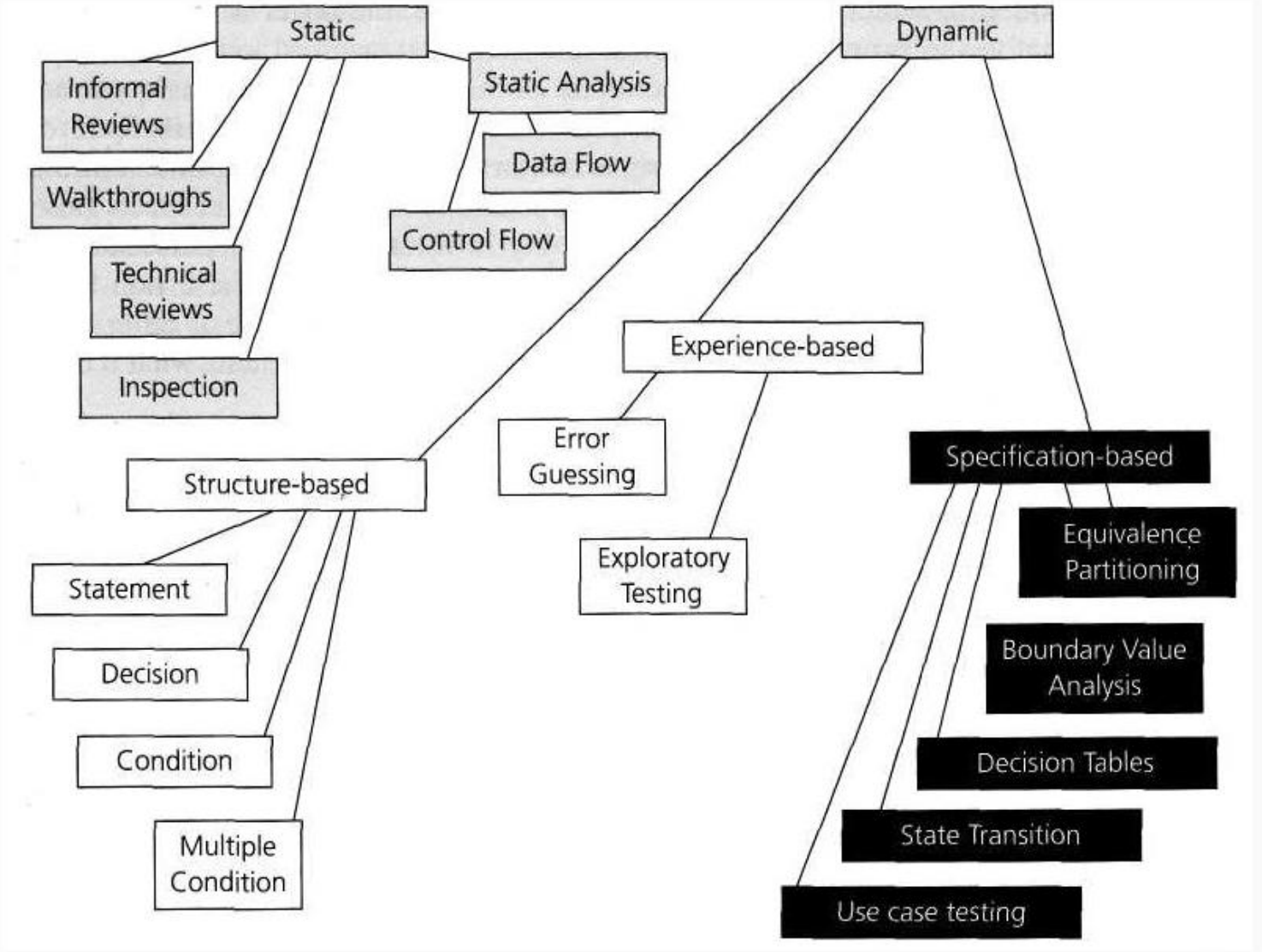
- **Step 1**: Identify inputs
- **Step 2**: Identify the outputs
- **Step 3**: Identify the major paths
- **Step 4**: Identify Subfunctions
- **Step 5**: Develop inputs for Subfunctions
- **Step 6**: Develop outputs for Subfunctions
- **Step 7**: Execute test case for Subfunctions
- **Step 8**: Verify the correct result for Subfunctions
- **Step 9**: Repeat steps 4 & 8 for other Subfunctions
- **Step 10**: Repeat steps 7 & 8 for other Subfunctions

4. Các công cụ kiểm thử

- Sinh viên tìm hiểu và giới thiệu 1 số công cụ kiểm thử.

- Phần này là chuyên đề tự học của Sinh viên. Sinh viên tự chọn / được phân công nghiên cứu một phần mềm là công cụ kiểm thử và trình bày trước lớp (hoặc làm đồ án môn học Chuyên đề KTPM 2).
- **Thực hành**: sử dụng Nunit, Junit hoặc tự viết chương trình Test các Unit (phương thức) của bản thân / người khác.

5. Kết hợp các Kỹ thuật kiểm thử



5. Kết hợp các Kỹ thuật kiểm thử

- Sinh viên xem hình để tìm hiểu. Phần này giới thiệu cho Sv kỹ thuật kiểm thử Use case rất cần thiết trong phân tích chức năng chủa Hệ thống PM.

5. Kết hợp các Kỹ thuật kiểm thử

- **Kiểm thử Use case (Use case testing)**: là một kỹ thuật xác định các test case mô tả từ đầu đến cuối hành vi của hệ thống từ góc nhìn của người sử dụng. Use case mô tả sự tương tác đặc trưng giữa người dùng bên ngoài (Actor) và hệ thống. Mỗi Use case sẽ mô tả cách thức người dùng tương tác với hệ thống để đạt được mục tiêu nào đó. Ngoài ra, Use case cũng xác định trình tự các bước mô tả mọi tương tác giữa người dùng và hệ thống.
- Mô tả hoạt động của Use case thì người ta thường dùng Workflow hoặc mô hình activity, mô hình Use case, …

5. Kết hợp các Kỹ thuật kiểm thử

Các thành phần của sơ đồ Use case:

- **Tác nhân (Actor)**: Người dùng hoặc đối tượng nào đó tương tác với hệ thống.
- **Brief description**: Mô tả ngắn gọn giải thích các trường hợp
- **Precondition**: Là các điều kiện được thỏa mãn trước khi bắt đầu thực hiện.

5. Kết hợp các Kỹ thuật kiểm thử

Các thành phần của sơ đồ Use case: (tiếp theo)

- **Basic flow / Normal flow**: là những dòng cơ bản trong hệ thống. Đó là dòng giao dịch được thực hiện bởi người dùng để hoàn thành mục đích của họ. Khi người dùng tương tác với hệ thống, vì đó là workflow bình thường nên sẽ không có bất kỳ lỗi nào xảy ra và người dùng sẽ nhận được đầu ra như mong đợi.
- **Alternate flow**: Ngoài workflow thông thường, hệ thống cũng có thể có workflow thay thế. Đây là tương tác ít phổ biến hơn được thực hiện bởi người dùng với hệ thống.

5. Kết hợp các Kỹ thuật kiểm thử

Các thành phần của sơ đồ Use case: (tiếp theo)

- **Exception flow**: Là các dòng lỗi / sai / ngăn cản người dùng đạt được mục đích của họ
- **Post conditions**: Các điều kiện cần được kiểm tra sau khi hoàn thành.

Các Tester sẽ duyệt trên sơ đồ bằng kỹ thuật Walkthrough là Review lại các Use case (kể cả các mở rộng của nó), để tìm các lỗi trong sơ đồ so với tài liệu đặc tả phần mềm (SRS), phát hiện các thiếu sót, sai, không hợp lý (phần này có học trong môn Phân tích thiết kế HTTT).

Q/A