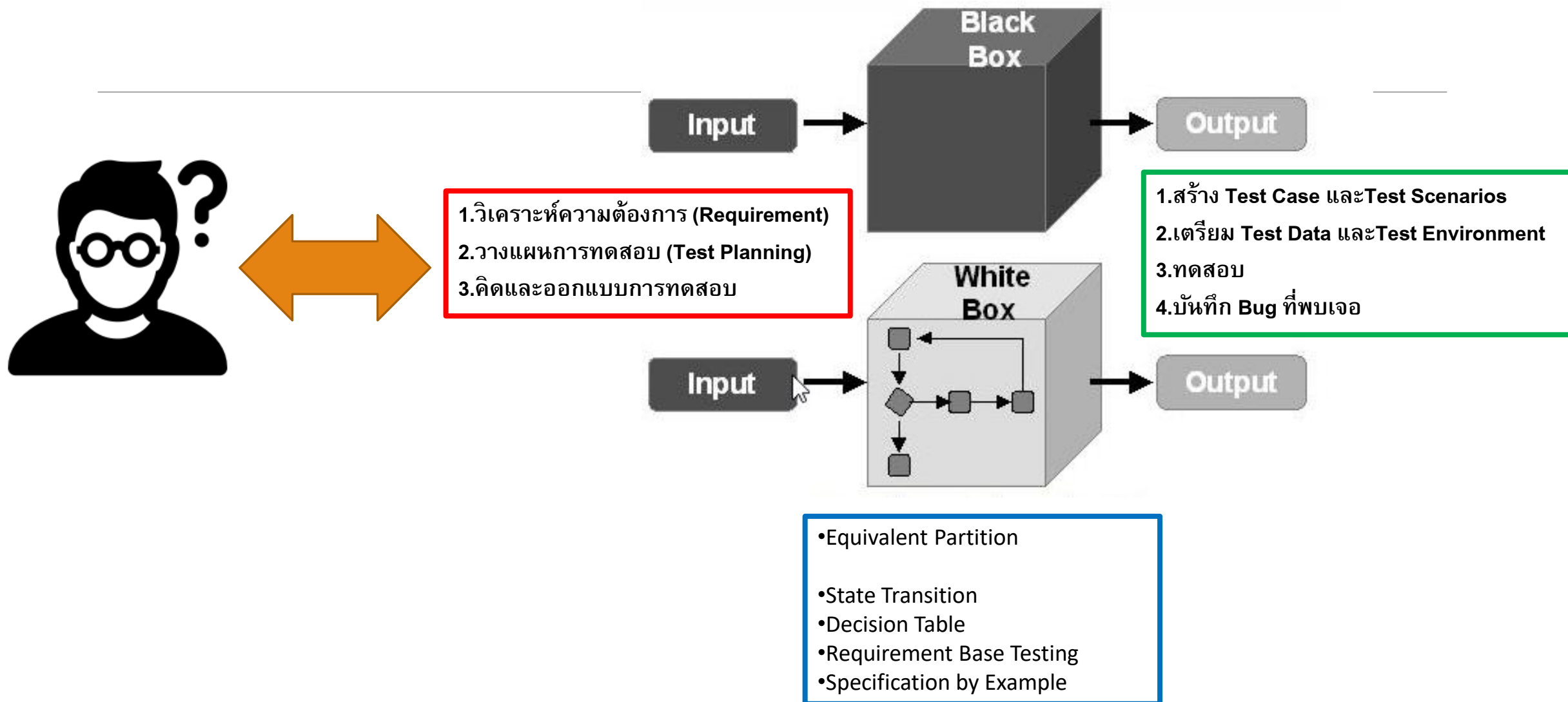
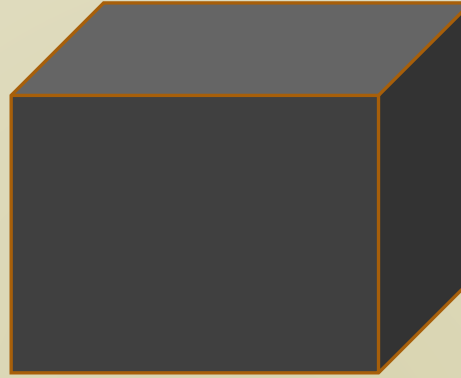


Testing Analysis



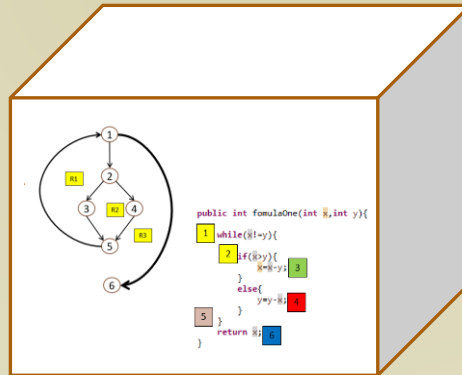
Test coverage case conceptual

Input



Out put

Input



Out put

Testing Type	Specification	General Scope	Opacity	Who generally does it?
Unit	Low-Level Design Actual Code Structure	Small unit of code no larger than a class	White Box	Programmer who wrote code
Integration	Low-Level Design High-Level Design	Multiple classes	White Box Black Box	Programmers who wrote code
Functional	High Level Design	Whole product	Black Box	Independent tester
System	Requirements Analysis	Whole product in representative environments	Black Box	Independent tester
Acceptance	Requirements Analysis	Whole product in customer's environment	Black Box	Customer
Beta	Ad hoc	Whole product in customer's	Black box	Customer

White Box Testing

White Box

เรียกอีกอย่างว่า “Structural หรือ Clear Box”

จะพิจารณากลไกภายในของระบบเป็นหลัก

ตรวจสอบกลไกและ ลอจิก รวมถึงโครงสร้างของโคด

สามารถแบ่งได้ 2 ชนิด

1. การทดสอบแบบกระแสควบคุม (Control Flow Testing)

- เน้นไปที่กลไกควบคุมการทำงานหลัก

2. การทดสอบกระแสข้อมูล (Data Flow Testing)

- เน้นไปที่การนิยามข้อมูลและการใช้งานของตัวแปร รวมถึงการประมวลผลที่มีความเกี่ยวเนื่องกัน

White Box

White box : 1.Control Flow Testing

เน้นการทดสอบโครงสร้างภายในของโปรแกรมเป็นหลัก

ทดสอบทุกๆความเป็นไปได้ของอัลกอริทึมในโปรแกรม

Control Flow แบ่งได้ 2 ทฤษฎีหลักๆดังนี้

1.1การวิเคราะห์ความครอบคลุมโค้ด (**Code Coverage Analysis**) คือ การเปรียบเทียบการทำงานของ โปรแกรม กับ โค้ด

1.2.การทดสอบเส้นทางการประมวลผลหลัก (**Basis Path Testing**) คือ วิธีการทดสอบที่ใช้ในการวัดความซับซ้อนทางลอจิกของการออกแบบ

White Box

White box : 1.Control Flow Testing → 1.1 Code Coverage Analysis

เป็นการทดสอบเพื่อวัดผลทางอ้อม

การทดสอบแบบนี้จะช่วยระบุเส้นทางภายในโปรแกรมที่ยังไม่ได้มีการทดสอบ

Code Coverage Analysis สามารถแบ่งได้ 3 ลำดับ

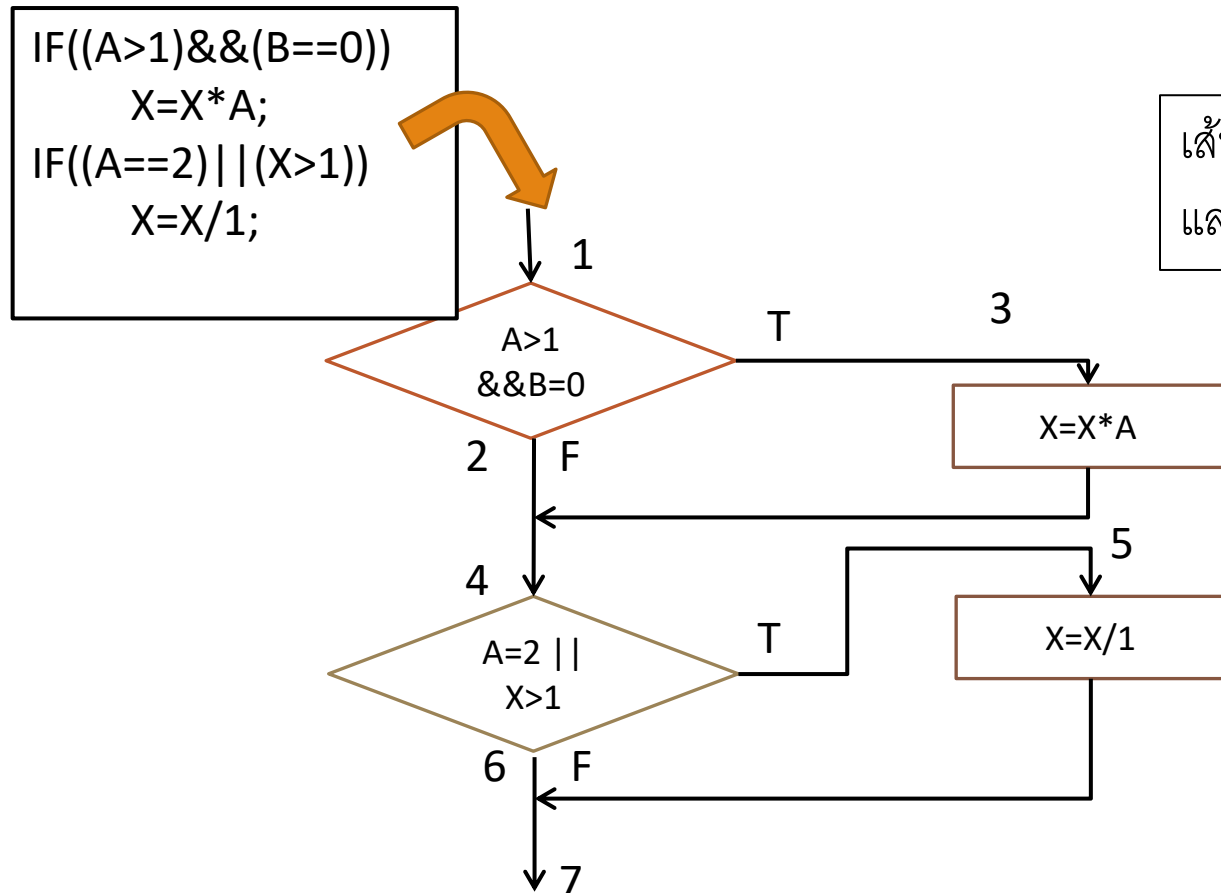
1.1.1ระดับคำสั่ง (Statement) : โค้ดแต่ละบรรทัดจะถูกประมวลผลอย่างน้อย 1 ครั้ง

1.1.2ระดับการตัดสินใจ (Branch) : ตรวจสอบความเที่ยงตรงในการตัดสินใจของโค้ด โดยการตัดสินใจต้องไม่ไปในทิศทางที่ผิด

1.1.3ระดับเส้นทางประมวลผล (Path) : เพื่อให้แน่ใจว่าทุกๆเส้นทางภายในโปรแกรม จะมีการประมวลผลเกิดขึ้นอย่างน้อย 1 ครั้ง

White Box

White box : 1.Control Flow Testing → 1.1 Code Coverage Analysis → 1.1.1 Statement



เส้นทางการประมวลผลที่สมบูรณ์คือ 1-3-4-5-7 [T][T]
และกำหนด $A=2, B=0, X=4$

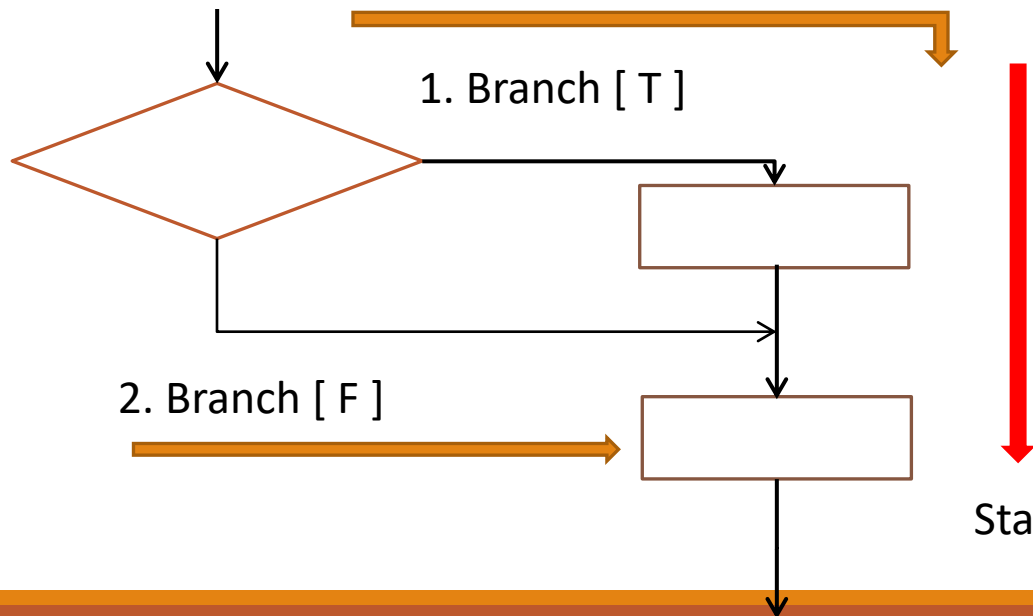
ข้อเสีย: ขาดความสามารถในการวิเคราะห์
ความสมบูรณ์ของอัลกอริทึมภายใน
โปรแกรม กรณีที่เงื่อนไข IF ไม่มี else จะ
พิจารณาแค่ เงื่อนไขที่เป็น จริงเท่านั้น

White Box

White box : 1.Control Flow Testing → 1.1 Code Coverage Analysis → 1.1.2 **Branch**

การตรวจสอบผลลัพธ์ในการตัดสินใจที่ใช้ในคำสั่งเงื่อนไข และการทำซ้ำ

แก้ไข ข้อเสียของวิธีการแบบ **Statement**



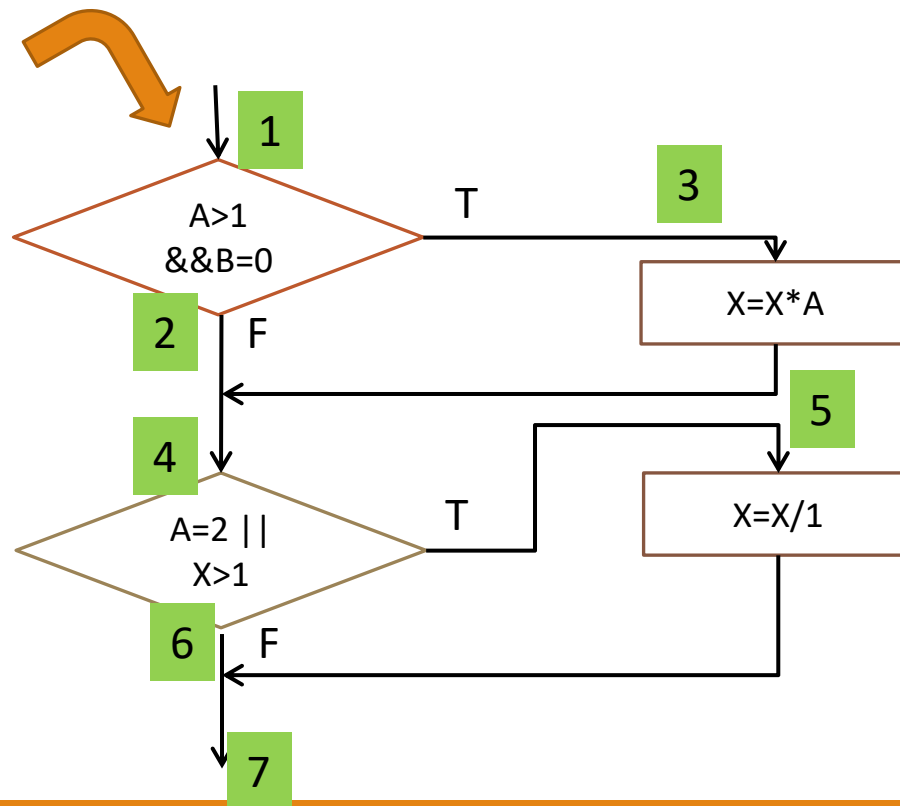
- **Branch** คือ การวัดจำนวนการทำงานของกลไกควบคุมภายในโปรแกรมจะต้องมีการทำงานในทุก ๆ ผลลัพธ์อย่างน้อย **1** ครั้ง

Statement Coverage

White Box

White box : 1.Control Flow Testing → 1.1 Code Coverage Analysis → 1.1.2 Branch

```
IF((A>1)&&(B==0))  
    X=X*A;  
IF((A==2) || (X>1))  
    X=X/1;
```



เส้นทางการประมวลผลที่สมบูรณ์คือ

1-3-4-5-7 [T][T] , A=2,B=0,X=4

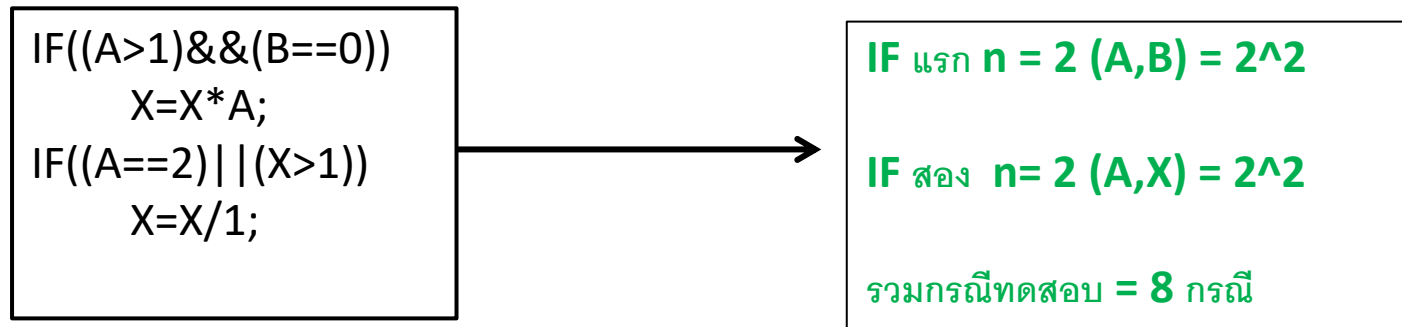
1-2-4- 6-7 [F][F] , A=1,B=0,X=1

White Box

White box :1.Control Flow Testing → 1.1Code Coverage Analysis → 1.1.2 Branch

ในกรณีที่โปรแกรมประกอบไปด้วยเงื่อนไขตั้งแต่ 2 หรือ มากกว่าขึ้นไป และมีการทำงานด้วย AND OR(Logic Operation) แต่ละเงื่อนไขจะถูกประเมินแยกจากกันจำเป็นจะต้องใช้การทดสอบแบบเงื่อนไขร่วม

(Multiple Condition) ซึ่งจำนวนการทดสอบจะหาได้จาก 2^n (n = จำนวน Input ในแต่ละเงื่อนไข)



AND

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

IF...THEN

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

NOT

A	$\sim A$
T	F
F	T

OR

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

xor

A	B	$A \text{ xor } B$
T	T	F
T	F	T
F	T	T
F	F	F

White Box

White box :1.Control Flow Testing →1.1Code Coverage Analysis→1.1.2 **Branch**

จงเขียนกรณีทดสอบว่ามีกี่จำนวน อะไรบ้าง ???

1. [A>1 เป็น T]& [B=0 เป็น T] - T

2. [A>1 เป็น T]&[B=0 เป็น F] - F

3. [A>1 เป็น F]&[B=0 เป็น T] - F

4. [A>1 เป็น F]&[B=0 เป็น F] - F

5. [A=2 เป็น T] || [X>1 เป็น T] - T

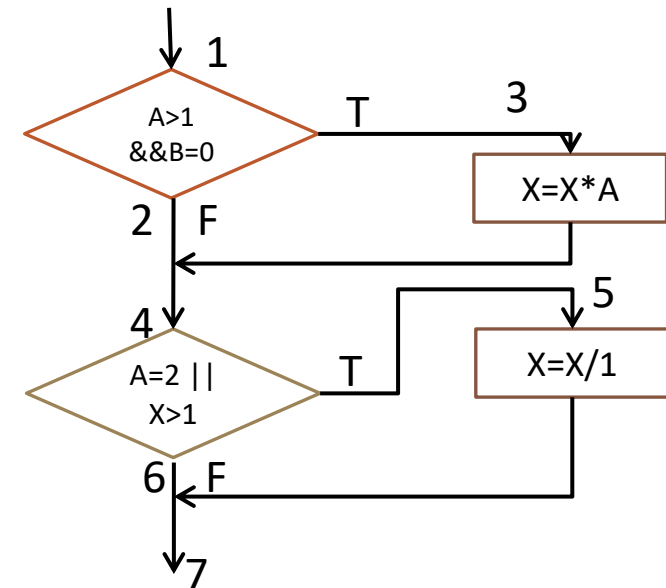
6. [A=2 เป็น T] || [X>1 เป็น F] - T

7. [A=2 เป็น F] || [X>1 เป็น T] - T

8. [A=2 เป็น F] || [X>1 เป็น F] - F

Input ที่อยู่ใน Test case ใดบ้าง

A=2, B=0, X=2 _____ [1][5] T,T
A=2, B=1, X=0 _____ [6][2] T,F
A=0, B=0, X=2 _____ [3][7] F,T
A=0, B=1, X=0 _____ [4][8] F,F

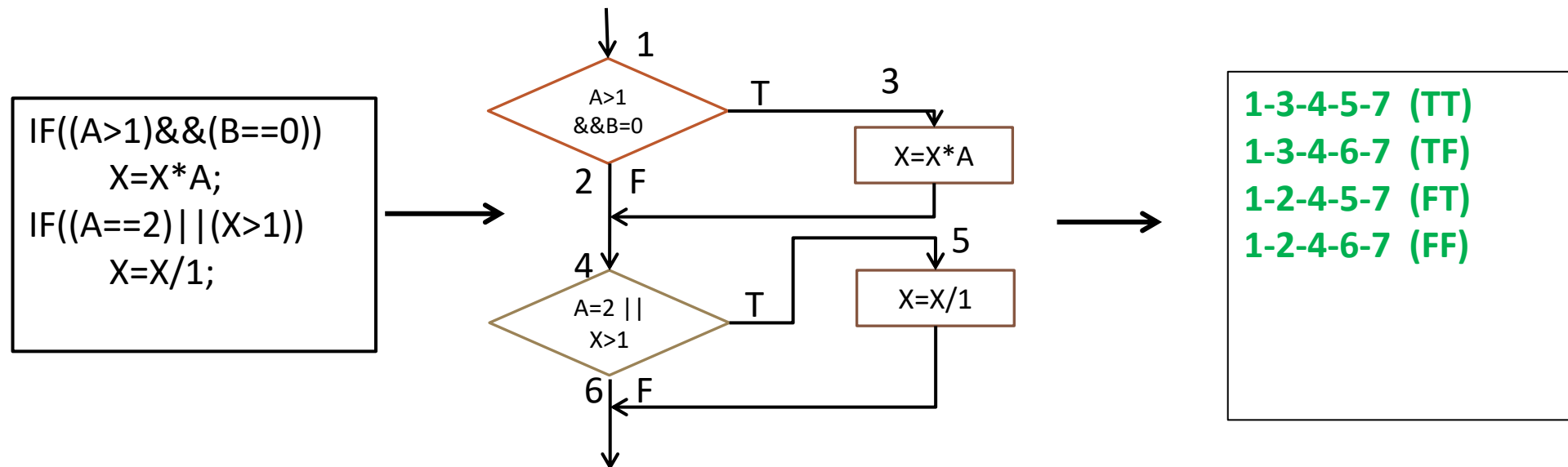


White Box

White box :1.Control Flow Testing →1.1 Code Coverage Analysis→1.1.3 Path

เป็นการทดสอบที่พิจารณาทุกๆความเป็นไปของเส้นทางการประมวลผล ตั้งแต่จุดเริ่มต้นถึงจุดสิ้นสุด

ข้อดี คือ สามารถใช้ได้ในทุกๆช่วงที่ทดสอบ



White Box

White box :1.Control Flow Testing → **1.2 Basis Path Testing**

คือ การทดสอบการประมวลผลในแต่ละคำสั่งภายในโปรแกรมอย่างน้อย 1 ครั้งเพื่อกำหนดเส้นทาง (Path) ที่แตกต่างกัน

นักทดสอบจำเป็นต้องนำเสนอการทำงานของโปรแกรมในรูปแบบของ กราฟกระแสควบคุม (Control Flow Graph)


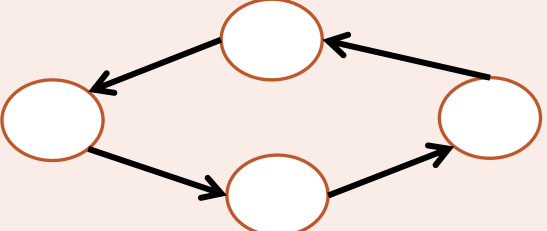
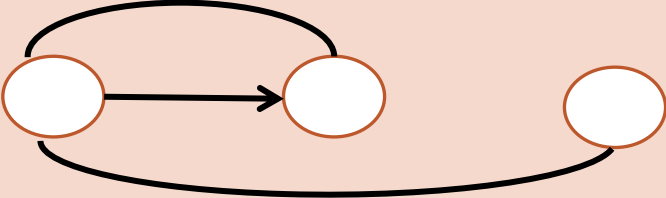
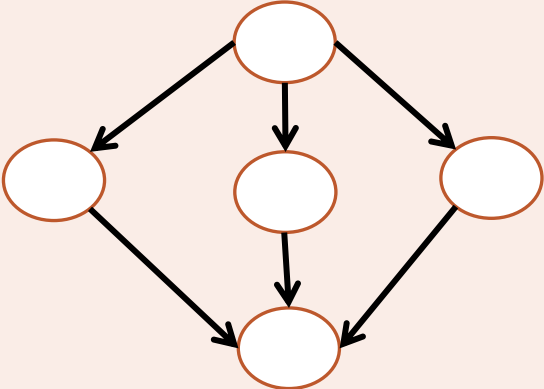
ขั้นตอนก่อนนำเสนอมีดังนี้

1. สร้างกราฟกระแสควบคุม
2. คำนวณค่า **Cyclomatic complexity**
3. เลือกกลุ่มของเส้นทางการประมวลผลหลัก
4. สร้างกรณีทดสอบเพื่อตรวจสอบแต่ละเส้นทาง

White Box

White box : Control Flow Testing → Basis Path Testing → กราฟกระแสควบคุม

รูปแบบสัญลักษณ์	ความหมาย
→	เรียกว่า “Edge” ทำหน้าที่สำหรับเป็น กลไกการควบคุม
●	เรียกว่า “Node” ใช้เสนอการกระทำ ตั้งแต่หนึ่งหรือมากกว่า

รูปแบบสัญลักษณ์	เงื่อนไข
	Sequence
	If-then-else
	While
	Case

White Box

White box : Control Flow Testing → Basis Path Testing → **Cyclomatic complexity**

คือ การวัดในเชิงปริมาณความซับซ้อนทางลอจิกที่เกิดขึ้นภายในโปรแกรม

คือ การนับจำนวนของเส้นทางการทำงานที่ไม่ซ้ำกันของ Source Code

ถ้ามี คำสั่งประเภทการตัดสินใจ เช่น **IF, For, While** จะนับเส้นทางการทำงาน ชนิดละ 2 เส้นทาง

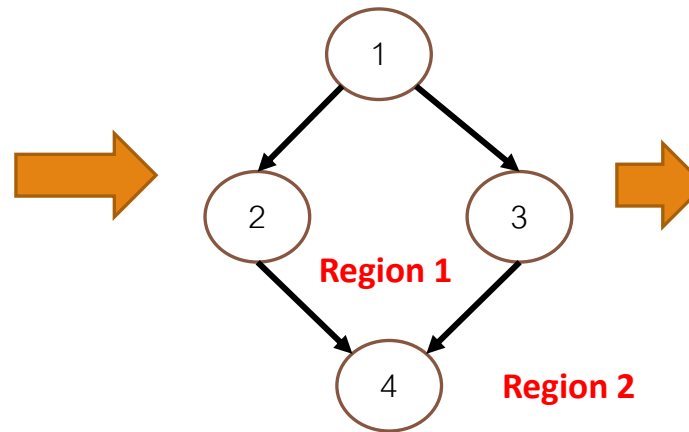
การคำนวณค่าของ **Cyclomatic Complexity**

Cyclomatic Complexity = จำนวนของ **Regions** ภายใน **Flow graph** (นับ **Region**)

White Box

White box : Control Flow Testing → Basis Path Testing → **Cyclomatic complexity**

```
public int check(int a,int b){  
    if(a>b){  
        a++;  
    }  
    else{  
        b++;  
    }  
    return a+b;  
}
```



Cyclomatic = 2 Regions

White Box

White box : Control Flow Testing → Basis Path Testing → **Cyclomatic complexity**

Cyclomatic = 2



Cyclomatic = 2 คือ จำนวนเส้นทางการทำงานที่ไม่ซ้ำกันสองเส้นทางตั้งแต่ Node แรกจนถึง Node สุดท้าย

Path 1 : 1-2-4

Path 2 : 1-3-4

White Box

White box : Control Flow Testing → Basis Path Testing → **Cyclomatic complexity**

ขั้นตอนสุดท้ายคือการเขียน Test Case

Test Id	Path	Input a	Input b	Expected Result
1	1,2,4	2	1	a=3 [a++]
2	1,3,4	1	2	b=3 [b++]

White Box

White box : Control Flow Testing → Basis Path Testing → **Cyclomatic complexity**

```
public int fomulaOne(int x,int y){  
    while(x!=y){  
        if(x>y){  
            x=x-y;  
        }  
        else{  
            y=y-x;  
        }  
    }  
    return x;  
}
```

Practice (Full Process) !!!!