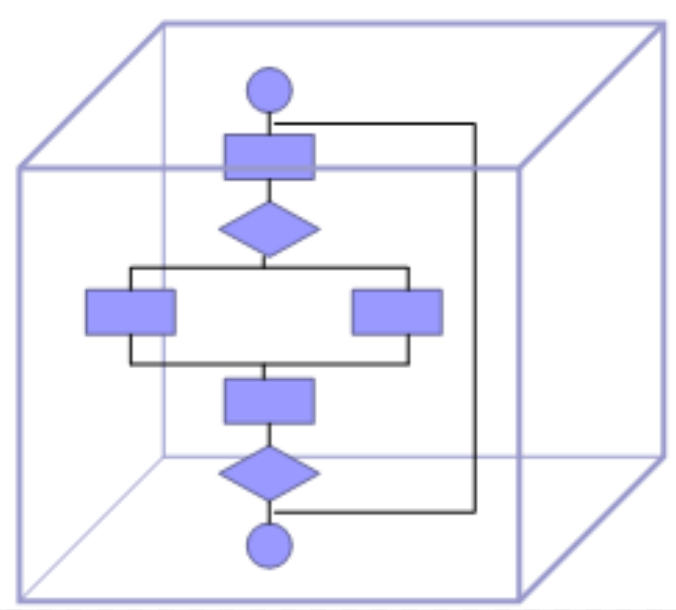
White-box testing

- Là phương pháp kiểm thử dựa trên cấu trúc logic của PM

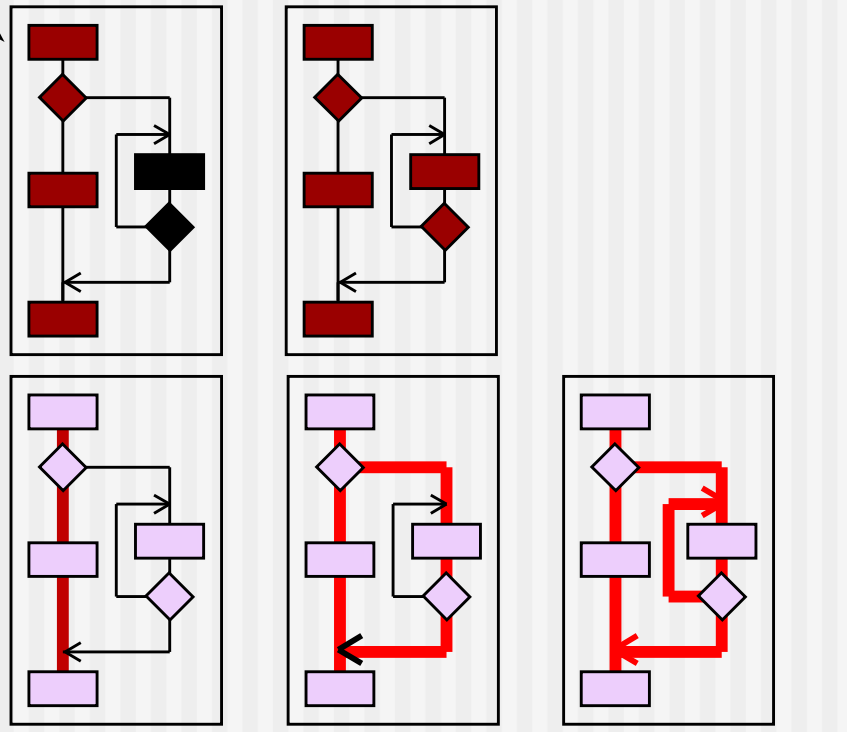


White-box testing

- Control Flow Testing
  - Statement testing
  - Branch / Decision testing
  - Branch condition testing
  - Branch condition combination testing
- Data flow testing

Kĩ thuật độ bao phủ cấu trúc (Coverage techniques)

Cái gì được bao phủ?



Các thành phần cấu trúc khác nhau

Tăng độ bao phủ

Kĩ thuật độ bao phủ cấu trúc (Coverage techniques)

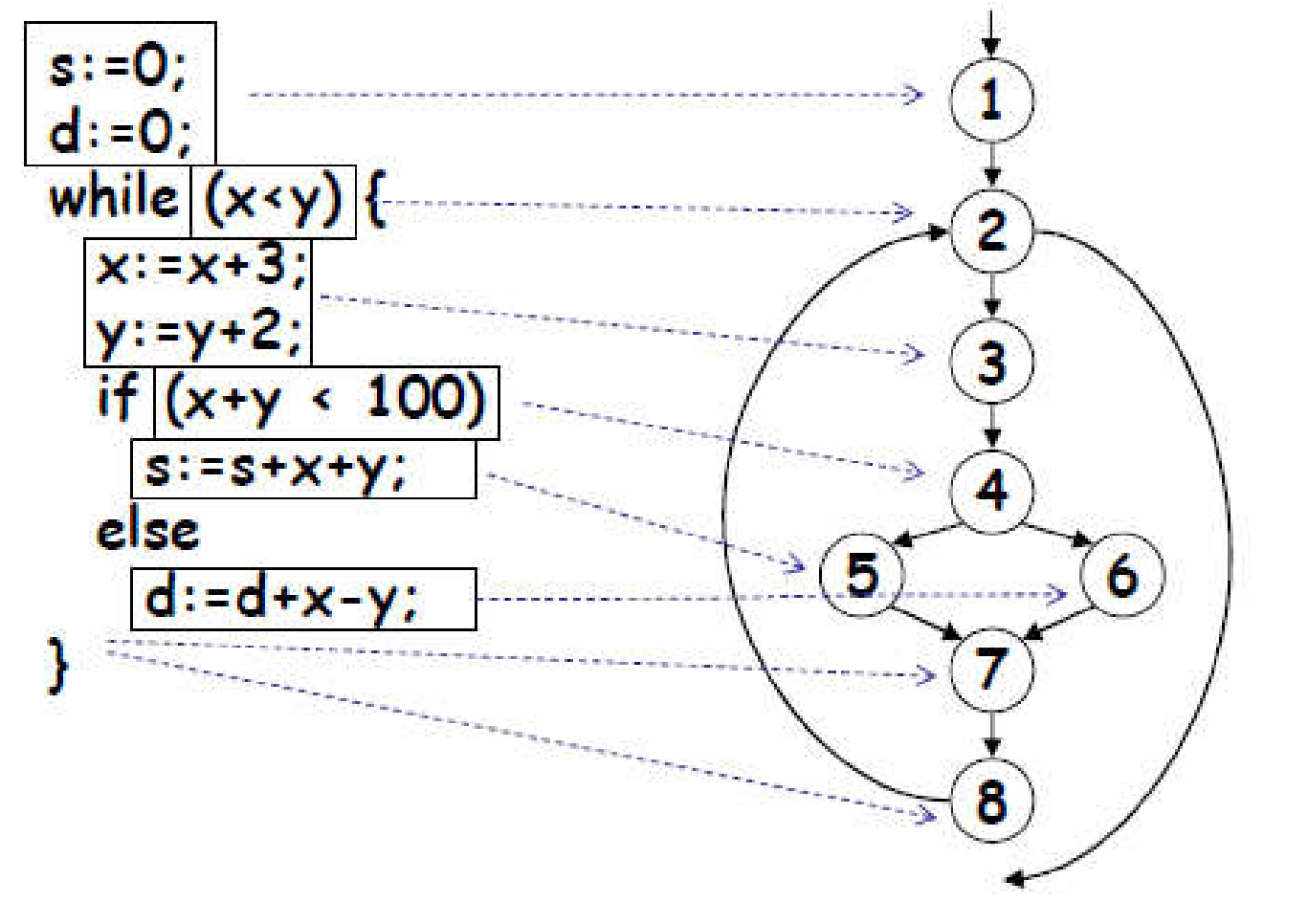
- Đánh giá độ bao phủ của các thành phần cấu trúc
- Thiết kế các test bổ sung → có thể đạt được mức bao phủ 100%

Độ bao phủ 100% không có nghĩa là 100% được test

Kiểm thử luồng điều khiển (Control Flow Testing)

- Bước 1:
  - Xây dựng đồ thị luồng điều khiển từ source code
  - Đồ thị bao gồm nút, cạnh
- Bước 2:
  - Thiết kế các Test cases để bao phủ hết các thành phần của đồ thị

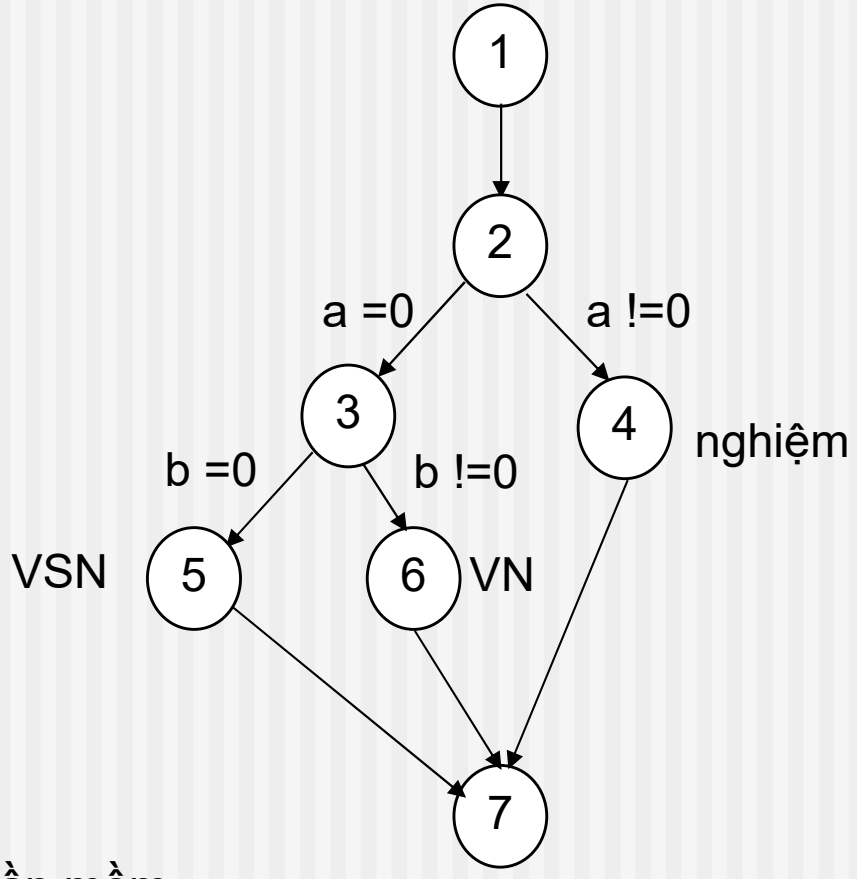
Ví dụ 1:



Ví dụ 2:

Đồ thị luồng điều khiển của đoạn chương trình giải phương trình bậc nhất

```
1.  if(a == 0)
2.      if(b == 0)
3.          cout<<"Vo so nghiem";
4.      else
5.          cout<<"Vo nghiem";
6.  else
7.      cout<<"x = "<<-b/a;
```



Độ phức tạp “Cyclomatic”

- $V(G) = E - N + 2$
- $V(G) = P + 1$
- $V(G) = R$

Trong đó:

- E: số cạnh
- N: số nút
- P: số nút điều kiện
- R: số miền

Vẽ đồ thị luồng điều khiển

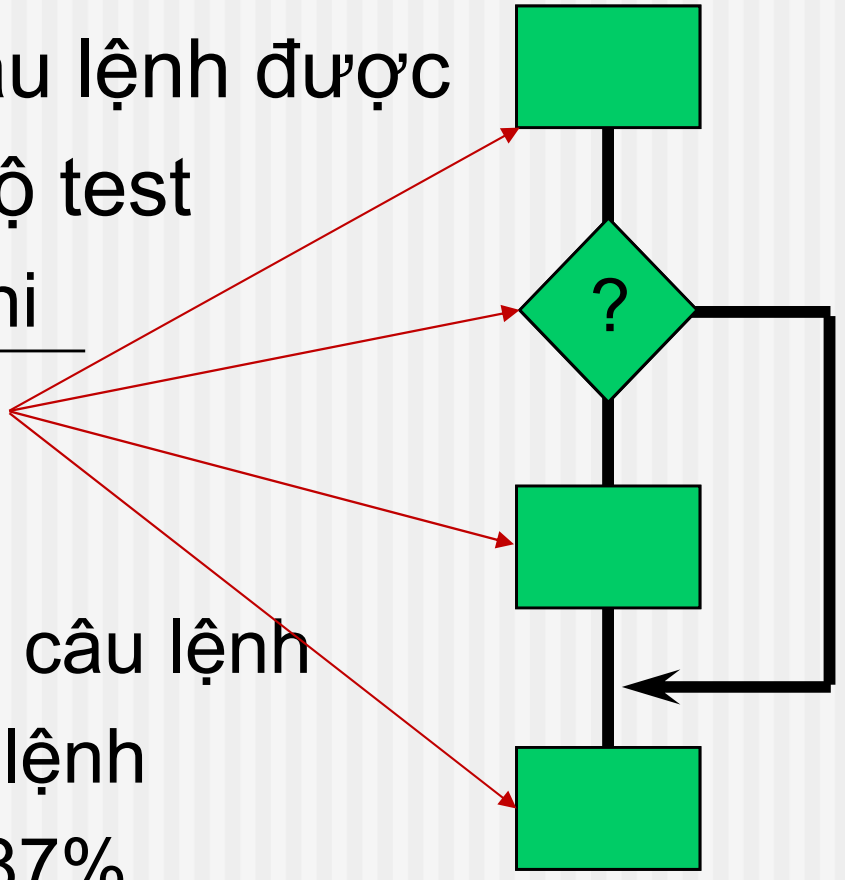
```
1.  void prime(int n){
2.       cout<<"Cac so nguyen to nho hon n la:“<<endl;
3.      for(int i = 2; i < n;i++){
4.            int flag = 1;
5.             for(int j = 2; j <= sqrt(i);j++)
6.                  if(i%j==0)
7.                       { flag = 0; break;}
8.             if(flag == 1){
9.                  cout<<i<<“\t”;
10.              }
11.     }
12.     }
```

Bao phủ câu lệnh (Statement coverage)

- Tỉ lệ phần trăm các câu lệnh được thực thi bởi bộ test

$$= \frac{\text{Số câu lệnh thực thi}}{\text{Tổng số câu lệnh}}$$

- Ví dụ:
  - Chương trình có 100 câu lệnh
  - tests thực thi 87 câu lệnh
  - Bao phủ câu lệnh = 87%



Ví dụ 1:

```
1  cin>>a;
2  if (a > 6)
3      b = a;
4  cout<<b;
```

Test case	Input	Expected output
1	7	7

Số câu lệnh

Với 1 test case này thì tất cả 4 câu lệnh đều được thực thi → đạt độ bao phủ dòng lệnh 100%

Ví dụ 2:

```
Prints (int a, int b) {
    int result = a+ b;
    If (result> 0)
        Print ("Positive", result)
    Else
        Print ("Negative", result)
    }
```

Ví dụ 2:

- Với A = 2, B = 5 → phủ 5/7 câu lệnh (71%)

```
1  Prints (int a, int b) {
2  int result = a+ b;
3  If (result> 0)
4      Print ("Positive", result)
5  Else
6      Print ("Negative", result)
7  }
```

Ví dụ 2:

- Với A = 3, B = -5 → phủ 6/7 câu lệnh (85%)

```
1  Prints (int a, int b) {
2    int result = a+ b;
3    If (result> 0)
4        Print ("Positive", result)
5    Else
6        Print ("Negative", result)
7    }
```

Ví dụ 2:

Tối thiểu 2 test-case để phủ 100% câu lệnh

```
Prints (int a, int b) {
    int result = a+ b;
    If (result> 0)
        Print ("Positive", result)
    Else
        Print ("Negative", result)
    }
```

Bài tập 1

Đoạn chương trình giải phương trình bậc nhất ax+b =0

```
1.  {
2.  if(a == 0)
3.      if(b == 0)
4.          cout<<”Vo so nghiem”;
5.      else
6.          cout<<”Vo nghiem”;
7.  else
8.      cout<<”x =”<<-b/a;
9.  }
```

1. *Có bao nhiêu câu lệnh?*
2. *Test case (a=0,b=8) bao phủ bao nhiêu % câu lệnh ?*
3. *Cần tối thiểu bao nhiêu test case để bao phủ 100% các câu lệnh ?*

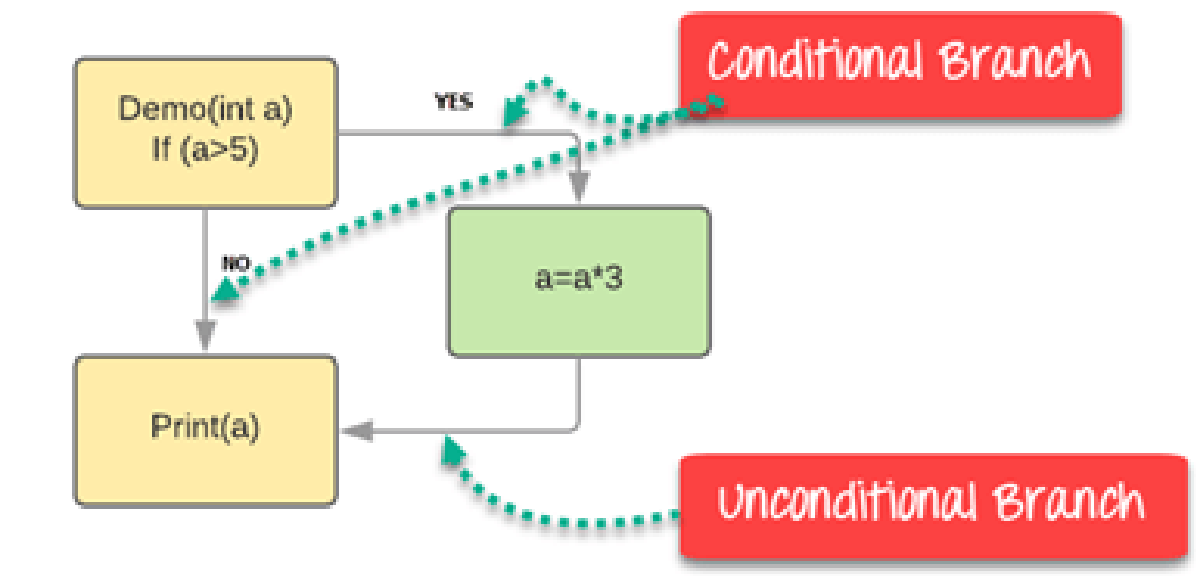
Bài tập 2

```
1.  void func(int A, int B, int X )
2.  {
3.      if  (A > 1  && B == 0 )
4.          X = X / A;
5.      if ( A == 2  ||  X > 1)
6.          X = X + 1;
7.  }
```

1. *Có bao nhiêu câu lệnh ?*
2. *Test case (A=2,B=1,X=3) bao phủ bao nhiêu % câu lệnh ?*
3. *Test case (A=3,B=0,X=0) bao phủ bao nhiêu % câu lệnh ?*
4. *Tối thiểu bao nhiêu test case để bao phủ 100% các câu lệnh ?*

Bao phủ nhánh (Decision/Branch coverage)

```
Demo(int a)
{
      If (a> 5)
         a=a*3
      Print (a)
}
```

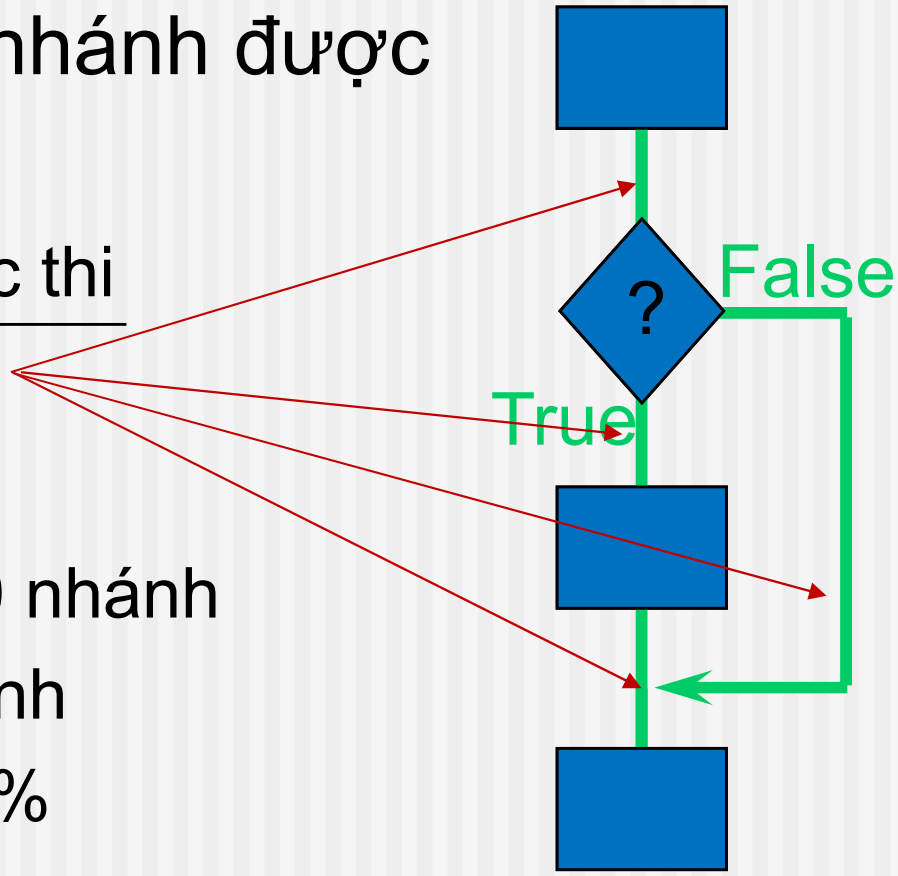


Bao phủ nhánh (Decision/Branch coverage)

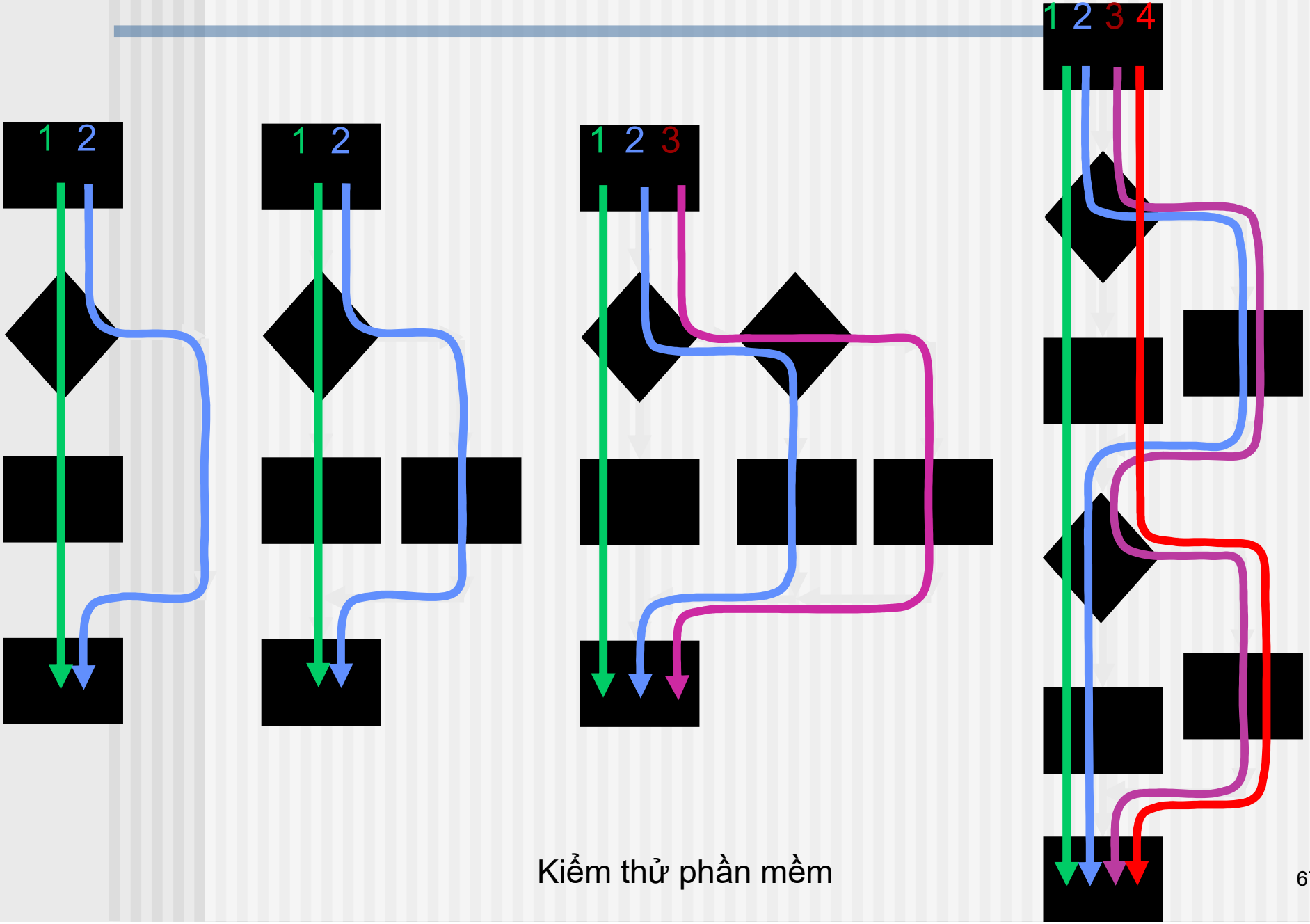
- Tỉ lệ phần trăm các nhánh được thực thi bởi bộ test

$$= \frac{\text{Số nhánh được thực thi}}{\text{Tổng số nhánh}}$$

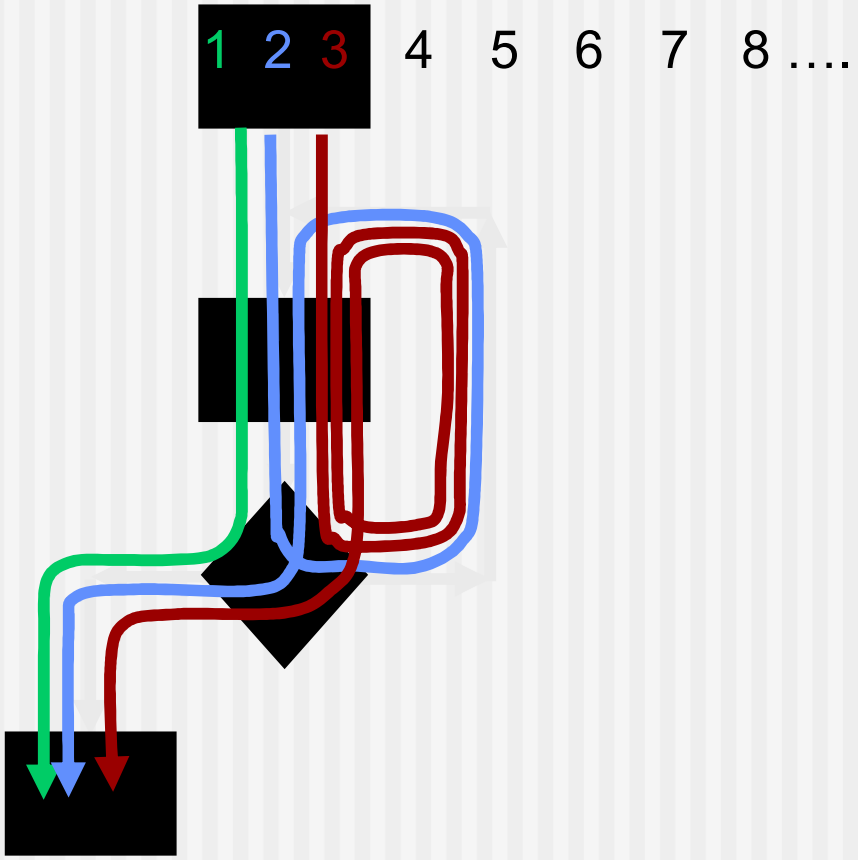
- Ví dụ:
  - Chương trình có 120 nhánh
  - tests thực thi 60 nhánh
  - Bao phủ nhánh = 50%



Tất cả các đường trong code



Các đường trong vòng lặp



Ví dụ 1:

Đoạn chương trình giải phương trình bậc nhất ax+b =0

```
1.  {
2.  if(a == 0)
3.      if(b == 0)
4.          cout<<"Vo so nghiem";
5.      else
6.          cout<<"Vo nghiem";
7.  else
8.      cout<<"x =",-b/a;
9.  }
```

1. *Có bao nhiêu nhánh ?*
2. *Test case (a=0,b=8),(a=0,b=0) bao phủ bao nhiêu % nhánh ?*
3. *Tối thiểu bao nhiêu test case để bao phủ 100% các nhánh?*

Ví dụ 2:

```
void func(int A, int B, int X )
1.  {
2.      if  ( A > 1 && B == 0 )
3.          X = X / A;
4.      if ( A == 2 || X > 1)
5.          X = X + 1;
6.  }
```

1. *Có bao nhiêu nhánh?*
2. *Test case (A=2,B=0,X=3) bao phủ bao nhiêu % nhánh ?*
3. *Tối thiểu bao nhiêu test case để bao phủ 100% các nhánh ?*

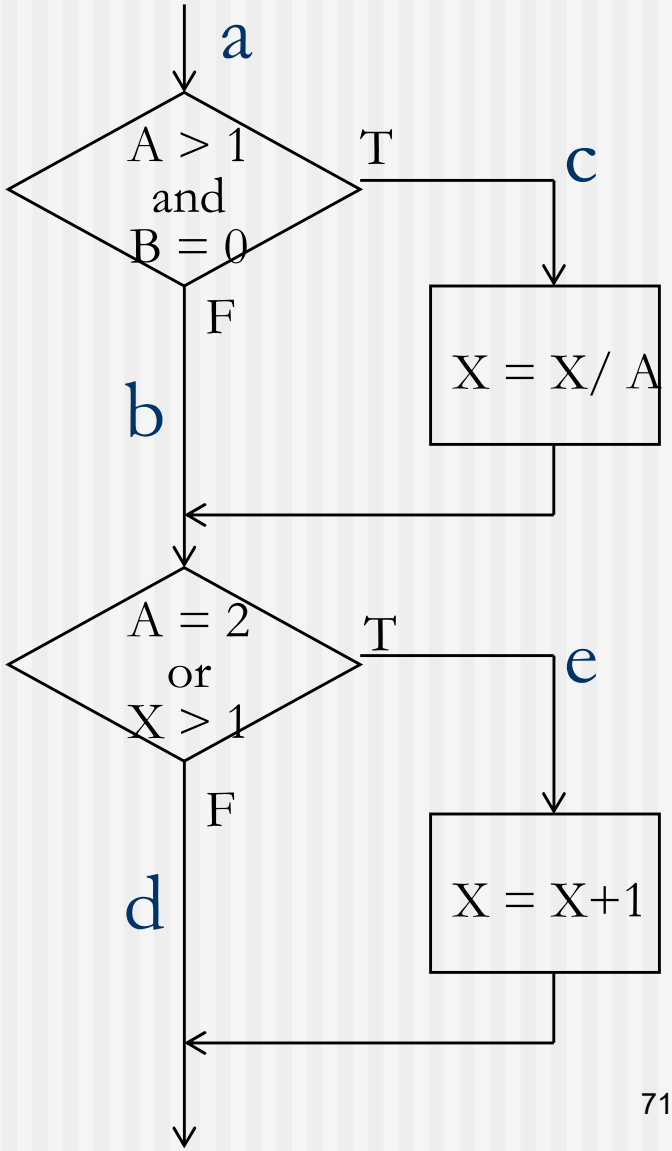
Bao phủ nhánh (điểm yếu ?)

```
void  func(int A, int B, int X )
{
if  ( A > 1 && B == 0 )
        X = X / A;
if ( A == 2  || X > 1)
        X = X + 1;
}
```

Test cases bao phủ nhánh

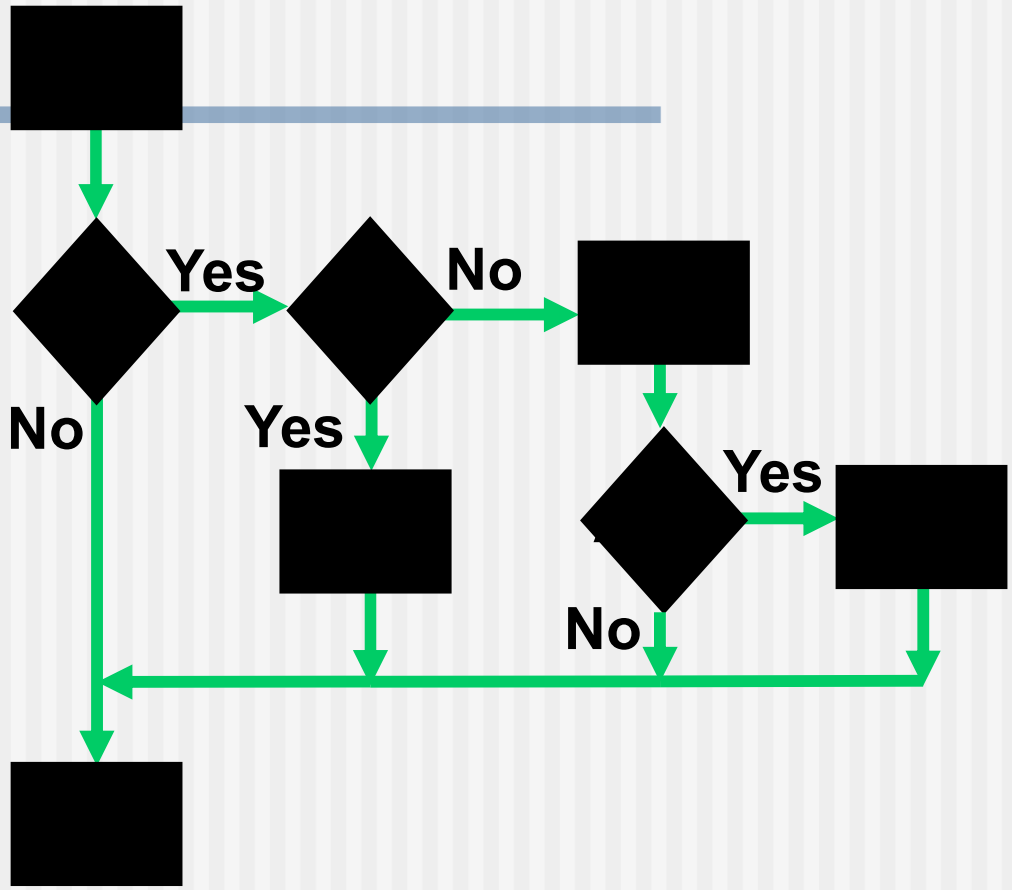
1) A = 3, B = 0, X = 3 (acd)

2) A = 2, B = 1, X = 1 (abe)



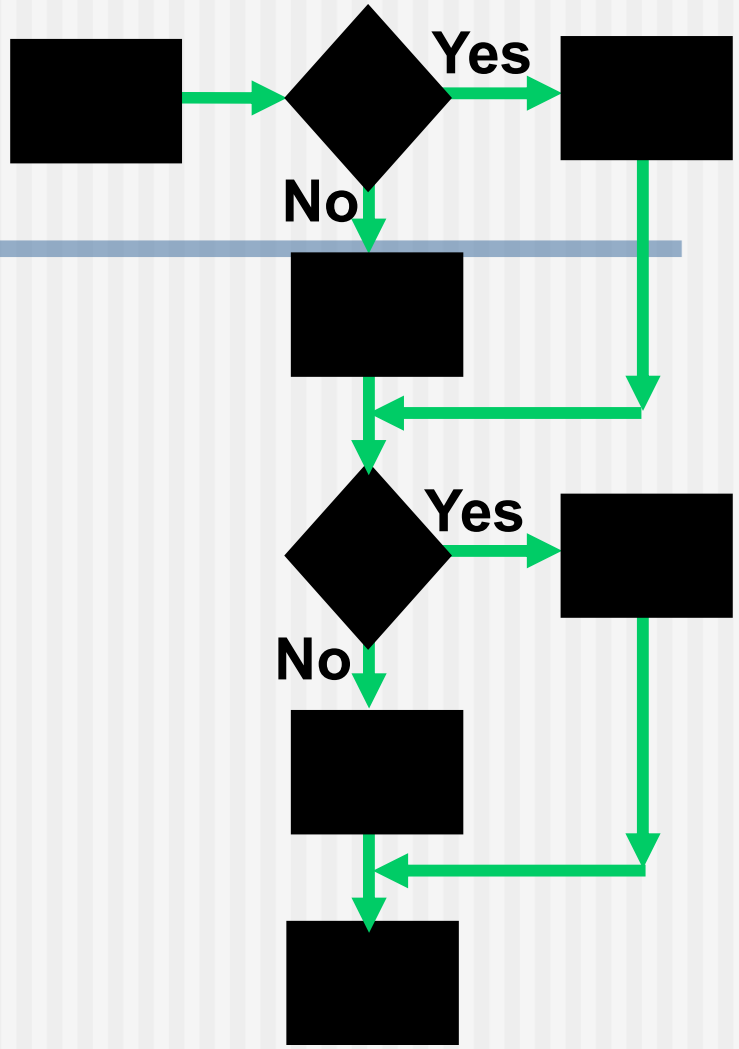
Ví dụ

```
1.  cin>>A;
2.  cin>> B;
3.  if (A > 0)
4.      if (B  == 0)
5.          cout<< “No values”;
6.      else
7.      {
8.          cout<<“B”<<endl;
9.          if (A > 21)
10.             cout<<“A”;
11.     }
```



- Độ phức tạp Cyclomatic : 4
- Số tests nhỏ nhất:
  - Statement coverage: 2
  - Branch coverage: 4

Ví dụ:



1. Read A
2. Read B
3. IF A < 0 THEN
4. Print “A negative”
5. ELSE
6. Print “A positive”
7. ENDIF
8. IF B < 0 THEN
9. Print “B negative”
10. ELSE
11. Print “B positive”
12. ENDIF

- Độ phức tạp Cyclomatic : 3
- Số tests nhỏ nhất :
  - Statement coverage: 2
  - Branch coverage: 2

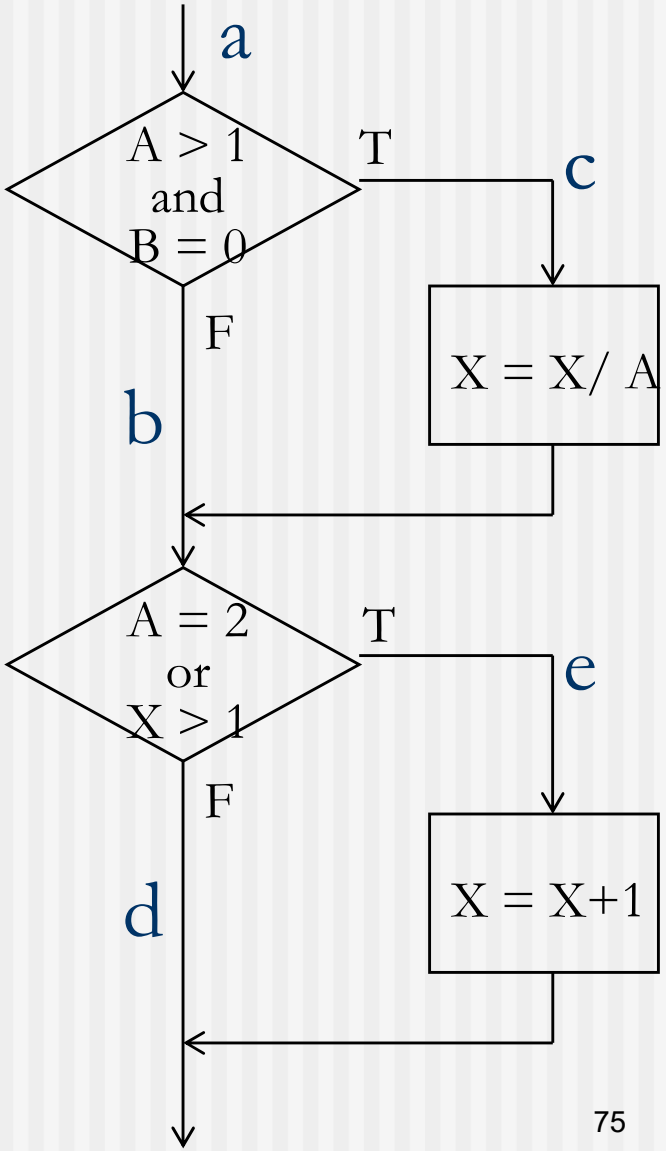
Bao phủ điều kiện (Condition coverage)

- Tỉ lệ phần trăm các điều kiện được thực thi bởi bộ test

$$= \frac{\text{Số điều kiện được thực thi}}{\text{Tổng số điều kiện}}$$

Ví dụ: Bao phủ điều kiện

- Cần thiết kế các test cases bao phủ hết các điều kiện sau:
  A>1, A<=1, B=0, B !=0
  A=2, A !=2, X >1, X<=1
- Test cases:
  1) A = 1, B = 0, X = 3 (abe)
  2) A = 2, B = 1, X = 1 (abe)
- Không thỏa mãn bao phủ nhánh



Bài tập 1

```
1.  float  A, B, C;
2.  printf("Enter three values\n");
3.  scanf("%f%f%f", &A, &B, &C);
4.  printf("\n largest value is: ");
5.  if( A>B)
6.  {
7.      if (A>C)   printf("%f\n",A);
8.      else    printf("%f\n",C);
9.  }
10. else
11. {
12.     if (C>B)   printf("%f\n",C);
13.     else    printf("%f\n",B);
14. }
```

Bài tập 2

```
1.  void PTA(int a[], int n)
2.  {
3.      int i = 0;
4.      while ((i < n) && (a[i] >= 0))
5.          i++;
6.      if (i < n)
7.          printf("Phan tu am a[%d] = %d", i, a[i]);
8.      else
9.          printf("Khong co phan tu am.");
10. }
```

Bài tập 3

```
1.  const int SCALENE = 1;
2.  const int ISOSCELES = 2;
3.  const int EQUILATERAL = 3;
4.  const int ERROR = 4;
5.  int TriangleType(int x, int y, int z){
6.        if (x<=0 || y<=0 || z<=0)
7.              return ERROR;
8.        if ((x + y <= z) || (x + z <= y) || (z + y <= x))
9.              return ERROR;
10.       if (x == y && y == z)
11.             return EQUILATERAL;
12.       if (x == y || y == z || z == x)
13.             return ISOSCELES;
14.     return SCALENE;
15. }
```

Bao phủ nhánh – điều kiện

- Bao phủ nhánh - điều kiện
  - Viết các TCs sao cho mỗi điều kiện (đơn) trong mỗi nhánh nhận được 2 giá trị (T và F) ít nhất 1 lần và mỗi nhánh được thực hiện ít nhất 1 lần
- Bao phủ tổ hợp các điều kiện
  - Viết các TCs để thực thi được tất cả các tổ hợp giá trị (T và F) của các điều kiện (đơn) trong 1 nhánh

Ví dụ:Bao phủ nhánh - điều kiện

- Các TCs cần bao phủ tất các điệu kiện

  A>1, A<=1, B=0, B !=0

  A=2, A !=2, X >1, X<=1

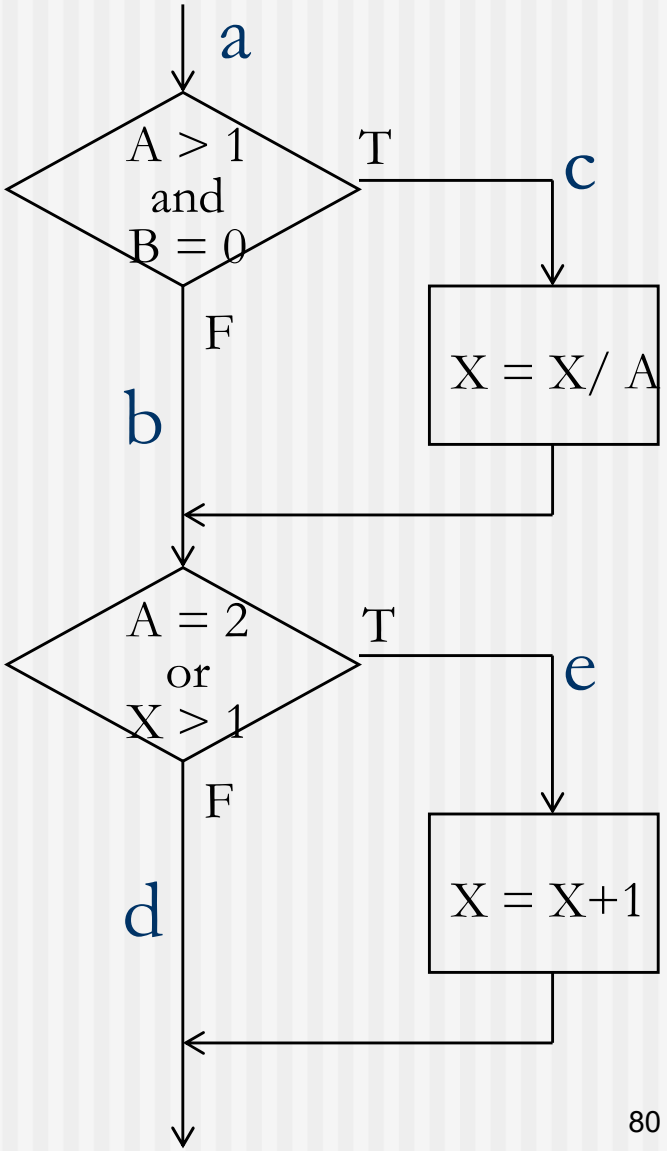
  và (A > 1 and B = 0) T, F

  (A = 2 or X > 1) T, F

- Test cases:

  1) A = 2, B = 0, X = 4 (ace)

  2) A = 1, B = 1, X = 1 (abd)



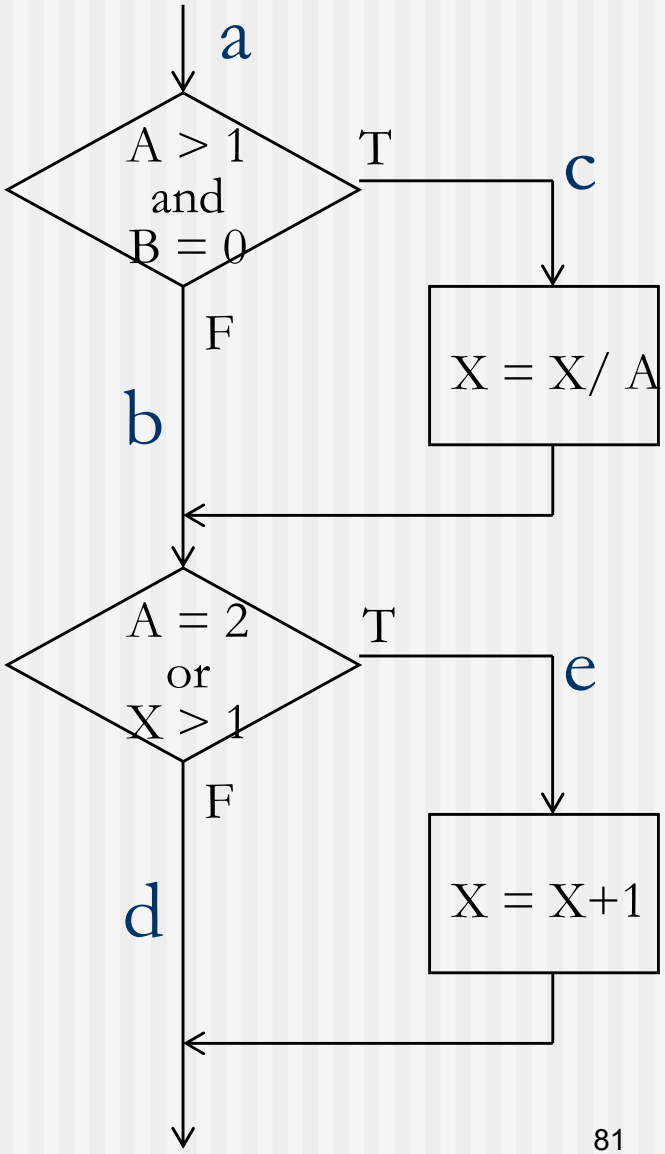
Ví dụ: Bao phủ tổ hợp điều kiện

- TCs phải bao phủ các điều kiện

1) A >1, B =0
2) A >1, B !=0
3) A<=1, B=0
4) A <=1, B!=0
5) A=2, X>1
6) A=2, X <=1
7) A!=2, X > 1
8) A !=2, X<=1

- Test cases:

1) A = 2, B = 0, X = 4 (bao phủ 1,5)
2) A = 2, B = 1, X = 1 (bao phủ 2,6)
3) A = 1, B = 0, X = 2 (bao phủ 3,7)
4) A = 1, B = 1, X = 1 (bao phủ 4,8)



Kĩ thuật kiểm thử dựa trên kinh nghiệm

- Error guessing
- Exploratory testing

Đoán lỗi(Error guessing)

- Được sử dụng sau khi áp dụng các kĩ thuật hình thức khác
- Có thể tìm ra 1 số lỗi mà các kĩ thuật khác có thể bỏ lỡ
- Không có qui tắc chung

Kiểm thử thăm dò (Exploratory testing)

- Hoạt động thiết kế test và thực thi test được thực hiện song song
- Các chú ý sẽ được ghi chép lại để báo cáo sau này
- Khía cạnh chủ chốt: learning(cách sử dụng, điểm mạnh, điểm yếu của PM)

Chọn kĩ thuật kiểm thử

- Mỗi kĩ thuật riêng lẻ chỉ hiệu quả với 1 nhóm lỗi cụ thể
- Các yếu tố ảnh hưởng đến việc chọn kĩ thuật kiểm thử:
  - Loại hệ thống
  - Tiêu chuẩn quy định
  - Yêu cầu khách hàng hoặc hợp đồng
  - Mức độ rủi ro, loại rủi ro
  - Mục tiêu kiểm thử
  - Tài liệu có sẵn

Chọn kĩ thuật kiểm thử

- Kiến thức của testers
- Thời gian và ngân sách
- Mô hình phát triển PM
- Kinh nghiệm về các loại lỗi đã được tìm ra trước đó