Kiểm thử (9)

Nguyễn Thanh Bình
Khoa Công nghệ Thông tin
Trường Đại học Bách khoa
Đại học Đà Nẵng

Nội dung

- Giới thiệu về kiểm thử
- Kiểm thử trong tiến trình phát triển
- Kiểm thử hộp đen
- Kiểm thử hộp trắng

Kiểm thử là gì ?

- IEEE: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó
- Myers: Kiểm thử là tiến trình thực thi chương trình với mục đích tìm thấy lỗi (The art of software testing)

3

Kiểm thử là gì ?

- Kiểm thử ≠ Gở rối (debug)
  - Kiểm thử
    - nhằm phát hiện lỗi
  - Gở rối
    - xác định bản chất lỗi và định vị lỗi trong chương trình
    - tiến hành sửa lỗi

4

Các khái niệm

- Một **sai sót** (error) là một sự nhầm lẫn hay một sự hiểu sai trong quá trình phát triển phần mềm của người phát triển
- Một **lỗi** (fault, defect) xuất hiện trong phần mềm như là kết quả của một sai sót
- Một **hỏng hóc** (failure) là kết quả của một lỗi xuất hiện làm cho chương trình không hoạt động được hay hoạt động nhưng cho kết quả không như mong đợi

5

Các khái niệm

- Dữ liệu thử (test data)
  - dữ liệu vào cần cung cấp cho phần mềm trong khi thực thi
- Kịch bản kiểm thử (test scenario)
  - các bước thực hiện khi khi kiểm thử
- Phán xét kiểm thử (test oracle)
  - đánh giá kết quả của kiểm thử
    - tự động: chương trình
    - thủ công: con người

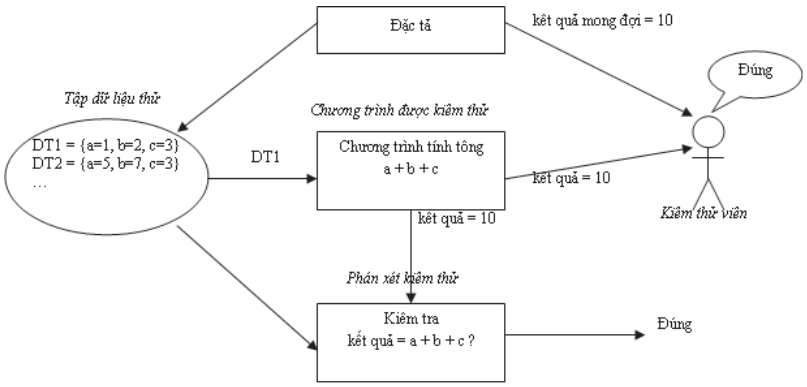
6

Các khái niệm

- Kiểm thử viên (tester)
  - người thực hiện kiểm thử
- Ca kiểm thử (test case)
  - tập dữ liệu thử
  - điều kiện thực thi
  - kết quả mong đợi

7

Các khái niệm



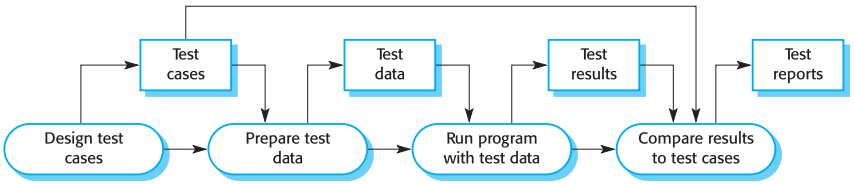
8

Tiến trình kiểm thử

- Kiểm thử thường bao gồm các bước
  - thiết kế các ca kiểm thử
  - bước tạo dữ liệu thử
    - kiểm thử với tất cả các dữ liệu vào là cần thiết
      - không thể kiểm thử "vét cạn"
    - chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào
      - dựa trên các tiêu chuẩn chọn dữ liệu thử
  - bước thực thi chương trình trên dữ liệu thử
    - cung cấp dữ liệu thử
    - thực thi
    - ghi nhận kết quả
  - bước quan sát kết quả kiểm thử
    - thực hiện trong khi hoặc sau khi thực thi
    - so sánh kết quả nhận được và kết quả mong đợi

9

Tiến trình kiểm thử



10

Khó khăn của kiểm thử

- Liên quan đến tiến trình phát triển
  - gồm nhiều giai đoạn phát triển
    - cái ra của một giai đoạn là cái vào của giai đoạn khác
    - mất mát thông tin
- Về mặt con người
  - thiếu đào tạo
  - ít chú trọng vai trò kiểm thử
- Về mặt kỹ thuật
  - không tồn tại thuật toán tổng quát có thể chứng minh sự đúng đắn hoàn toàn của bất kỳ một chương trình nào

11

Tại sao kiểm thử

- Hợp thức hóa (validation)
  - chỉ ra rằng sản phẩm đáp ứng được *yêu cầu người sử dụng*
- Xác minh (verification)
  - chỉ ra rằng sản phẩm thỏa mãn *đặc tả yêu cầu*
- Phân biệt hợp thức hóa và xác minh
  - “Verification: Are we building the product right ?”
  - “Validation: Are we building the right product ?”

12

Kiểm thử trong tiến trình phát triển

- Các kỹ thuật kiểm thử
  - kỹ thuật kiểm thử tĩnh (static testing)
  - kỹ thuật kiểm thử động (dynamic testing)
    - kiểm thử hộp đen (black-box testing)
      - kỹ thuật kiểm thử chức năng (functional testing)
    - kiểm thử hộp trắng (white-box testing)
      - kỹ thuật kiểm thử cấu trúc (structural testing)
- Các hoạt động kiểm thử/chiến lược kiểm thử
  - kiểm thử đơn vị (unit testing)
  - kiểm thử tích hợp (integration testing)
  - kiểm thử hợp thức hóa (validation testing)
  - kiểm thử hồi quy (regression testing)

13

Kiểm thử trong tiến trình phát triển

- Kiểm thử đơn vị (unit testing)
  - kiểm thử mỗi đơn vị phần mềm (mô-đun)
  - sử dụng kỹ thuật kiểm thử hộp đen
  - dữ liệu thử đươc tạo ra dựa trên tài liệu thiết kế
  - có thể sử dụng cả kiểm thử hộp trắng và kiểm thử tĩnh
    - phần mềm yêu cầu chất lượng cao
  - thường được thực hiện trên phần cứng phát triển phần mềm

14

Kiểm thử trong tiến trình phát triển

- Kiểm thử tích hợp (integration testing)
  - sau khi đã thực hiện kiểm thử đơn vị
  - ghép nối các đơn vị/thành phần phần mềm
  - kiểm thử sự ghép nối, trao đổi dữ liệu giữa các đơn vị/thành phần
  - sử dụng kỹ thuật kiểm thử hộp đen
  - một số trường hợp, sử dụng kỹ thuật kiểm thử hộp trắng
    - chi phí cao, khó khăn
  - dữ liệu thử được tạo ra dựa trên thiết kế tổng thể

15

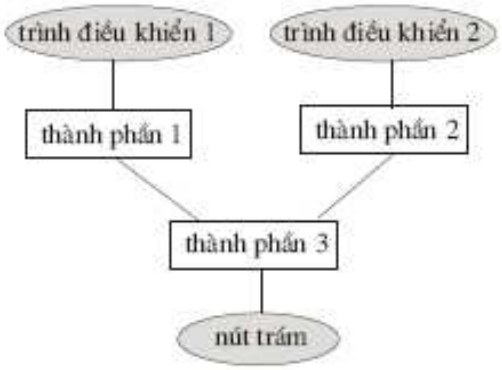
Kiểm thử trong tiến trình phát triển

- Kiểm thử tích hợp (2)
  - cần xây dựng thêm
    - *nút trám* (stub): các thành phần khác mô phỏng các thành phần phần mềm chưa được tích hợp
    - *trình điều khiển* (driver): các thành phần tạo ra các dữ liệu vào cho một vài các thành phần phần mềm trong tập hợp đang được kiểm thử

16

Kiểm thử trong tiến trình phát triển

- Kiểm thử tích hợp (3)



17

Kiểm thử trong tiến trình phát triển

- Kiểm thử tích hợp (4)
  - chiến lược *từ trên xuống* (top-down)
    - kiểm thử tích hợp các thành phần chính trước, sau đó thêm vào các thành phần được gọi trực tiếp bởi các thành phần vừa kiểm thử
    - cho phép xác định sớm các lỗi về kiến trúc
    - các bộ dữ liệu thử có thể được tái sử dụng cho các bước tiếp theo
    - tuy nhiên chiến lược này đòi hỏi phải xây dựng nhiều nút trám
  - chiến lược *từ dưới lên* (bottom-up)
    - kiểm thử các thành phần không gọi các thành phần khác, sau đó thêm vào các thành phần gọi các thành phần vừa kiểm thử
    - ít sử dụng các nút trám
    - nhưng lại xác định lỗi trễ hơn

18

Kiểm thử trong tiến trình phát triển

- Kiểm thử hợp thức hóa (validation testing)
  - còn gọi là kiểm thử hệ thống (system testing)
  - thực hiện sau khi kiểm thử tích hợp kết thúc
  - chứng minh phần mềm thực hiện đúng mong đợi của người sử dụng
  - dựa vào yêu cầu người sử dụng
  - chỉ sử dụng kỹ thuật kiểm thử hộp đen
  - nên thực hiện trong môi trường mà phần mềm sẽ được sử dụng

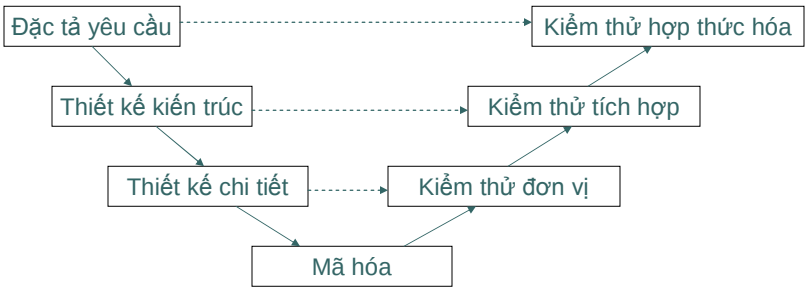
19

Kiểm thử trong tiến trình phát triển

- Kiểm thử hồi quy (regression testing)
  - phần mềm sau khi đưa vào sử dụng, có thể có các chỉnh sửa
    - có thể phát sinh lỗi mới
  - cần kiểm thử lại: kiểm thử hồi quy
  - thường tái sử dụng các bộ dữ liệu thử đã sử dụng trong các giai đoạn trước

20

Kiểm thử trong mô hình V



21

Các kỹ thuật kiểm thử

- kỹ thuật kiểm thử tĩnh (static testing)
  - **không thực thi** chương trình
- kỹ thuật kiểm thử động (dynamic testing)
  - kiểm thử hộp đen (black-box testing)
    - kỹ thuật kiểm thử chức năng (functional testing)
  - kiểm thử hộp trắng (white-box testing)
    - kỹ thuật kiểm thử cấu trúc (structural testing)

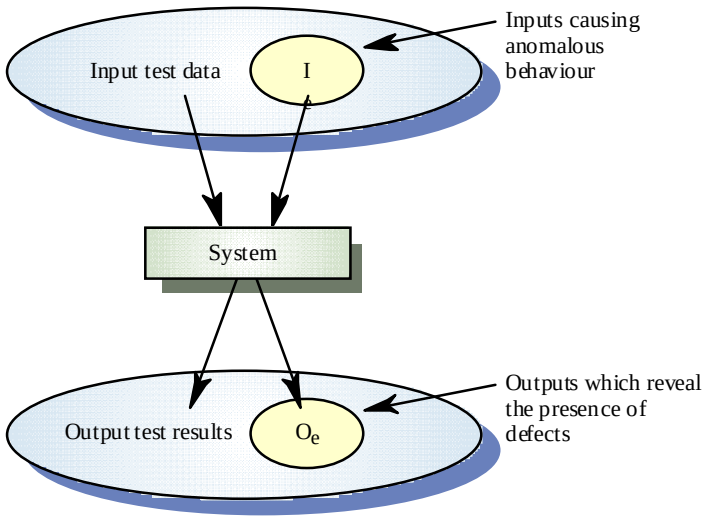
22

Kiểm thử tĩnh

- Thanh tra mã nguồn (code inspection)
- Chứng minh hình thức
- Thực thi hình thức (symbolic execution)
- Đánh giá độ phức tạp
  - McCabe
  - Nejmeh

23

Kiểm thử hộp đen



24

Kiểm thử hộp đen

- Chỉ cần dựa vào đặc tả chương trình
  - Xây dựng dữ liệu thử trước khi mã hóa/lập trình
- Thường phát hiện các lỗi đặc tả yêu cầu, thiết kế
- Dễ dàng thực hiện
- Chi phí thấp

25

Kiểm thử hộp đen

- Kiểm thử giá trị biên (boundary value analysis)
- Kiểm thử lớp tương đương (equivalence class testing)
- Kiểm thử ngẫu nhiên (random testing)
- Đồ thị nhân-quả (cause-efect graph)
- Kiểm thử cú pháp

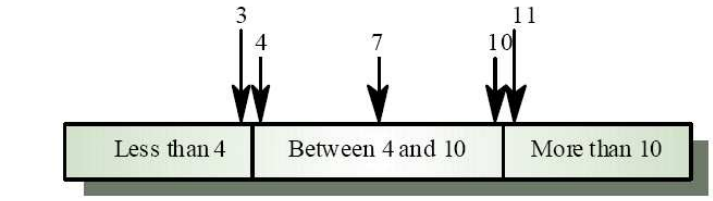
26

Kiểm thử giá trị biên

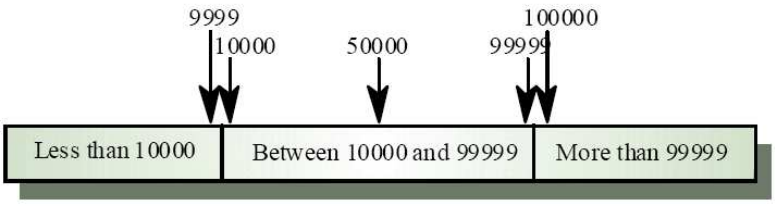
- Cơ sở
  - lỗi thường xuất hiện gần các giá trị biên của miền dữ liệu
- Tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử
- Nguyên tắc: kiểm thử các dữ liệu vào gồm
  - giá trị nhỏ nhất
  - giá trị gần kề lớn hơn giá trị nhỏ nhất
  - giá trị bình thường
  - giá trị gần kề nhỏ hơn giá trị lớn nhất
  - giá trị lớn nhất

27

Kiểm thử giá trị biên



Number of input values



Input values

28

Kiểm thử giá trị biên

- Nguyên tắc chọn dữ liệu thử
  - Nếu *dữ liệu vào thuộc một khoảng*, chọn
    - 2 giá trị biên
    - 4 giá trị = giá trị biên ± sai số nhỏ nhất
  - Nếu *giá trị vào thuộc danh sách các giá trị*, chọn
    - phần tử thứ nhất, phần tử thứ hai, phần tử kế cuối và phần tử cuối
  - Nếu dữ liệu vào là *điều kiện ràng buộc số giá trị*, chọn
    - số giá trị tối thiểu, số giá trị tối đa và một số các số giá trị không hợp lệ
  - Tự vận dụng khả năng và thực tế để chọn các giá trị biên cần kiểm thử

29

Kiểm thử giá trị biên

- Ví dụ (1)
  - Chương trình nhận vào ba số thực, kiểm tra ba số thực có là độ dài ba cạnh một tam giác. Nếu là độ dài ba cạnh của một tam giác, thì kiểm tra xem đó là tam giác thường, cân, đều cũng như kiểm tra đó là tam giác nhọn, vuông hay tù.

30

Kiểm thử giá trị biên

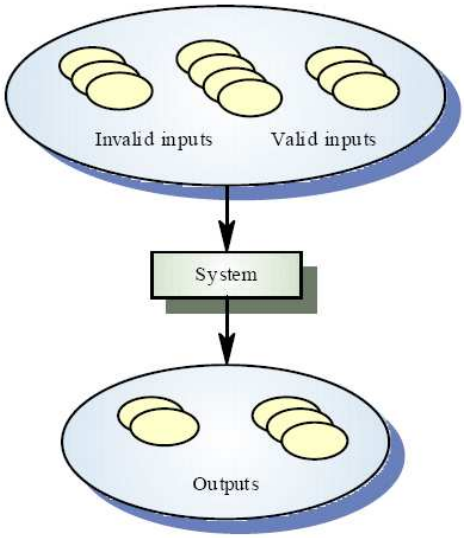
- Ví dụ (2)
  - Dữ liệu thử

1, 1, 2	Không là tam giác
0, 0, 0	Chỉ một điểm
4, 0, 3	Một cạnh bằng không
1, 2, 3.00001	Gần là một tam giác
0.001, 0.001, 0.001	Tam giác rất nhỏ
99999, 99999, 99999	Tam giác rất lớn
3.00001, 3, 3	Tam giác gần đều
2.99999, 3, 4	Tam giác gần cân
3, 4, 5.00001	Tam giác giác gần vuông
3, 4, 5, 6	Bốn giá trị
3	Chỉ một giá trị
	Dữ liệu vào rỗng
-3, -3, 5	Giá trị âm

31

Kiểm thử lớp tương đương

- Ý tưởng
  - phân hoạch miền dữ liệu vào thành các lớp các dữ liệu có quan hệ với nhau
  - mỗi lớp dùng để kiểm thử một chức năng, gọi là *lớp tương đương*



32

Kiểm thử lớp tương đương

- Ba bước
  - đối với mỗi dữ liệu vào, **xác định các lớp tương đương** từ miền dữ liệu vào
  - **chọn dữ liệu đại diện** cho mỗi lớp tương đương
  - **kết hợp các dữ liệu** thử bởi tích Đề-các để tại ra bộ dữ liệu kiểm thử

33

Kiểm thử lớp tương đương

- Nguyên tắc phân hoạch các lớp tương đương
  - Nếu *dữ liệu vào thuộc một khoảng*, xây dựng
    - 1 lớp các giá trị lớn hơn
    - 1 lớp các giá trị nhỏ hơn
    - n lớp các giá trị hợp lệ
  - Nếu *dữ liệu là tập hợp các giá trị*, xây dựng
    - 1 lớp với tập rỗng
    - 1 lớp quá nhiều các giá trị
    - n lớp hợp lệ
  - Nếu dữ liệu vào là *điều kiện ràng buộc*, xây dựng
    - 1 lớp với ràng buộc được thỏa mãn
    - 1 lớp với ràng buộc không được thỏa mãn

34

Kiểm thử lớp tương đương

- Ví dụ
  - Bài toán tam giác

	Nhọn	Vuông	Tù
Thường	6,5,3	5,6,10	3,4,5
Cân	6,1,6	7,4,4	√2,2,√2
Đều	4,4,4	không thể	không thể

Không là tam giác -1,2,8

35

Bài tập

- Kiểm thử giá trị biên
  - Viết một chương trình thống kê phân tích một tệp chứa tên và điểm của sinh viên trong một năm học. Tệp này chứa nhiều nhất 100 trường. Mỗi trường chứa tên của mỗi sinh viên (20 ký tự), giới tính (1 ký tự) và điểm của 5 môn học (từ 0 đến 10). Mục đích chương trình:
    - tính diểm trung bình mỗi sinh viên
    - tính điểm trung bình chung (theo giới tính et theo môn học)
    - tính số sinh viên lên lớp (điểm trung bình trên 5)
  - Xây dựng dữ liệu thử cho chương trình trên bởi kiểm thử giá trị biên

36

Bài tập

- Kiểm thử lớp tương đương
  - Viết chương trình dịch, trong đó có câu lệnh FOR, đặc tả câu lệnh FOR như sau: “Lệnh FOR chỉ chấp nhận **một tham số duy nhất** là biến đếm. Tên biến không được sử dụng quá hai ký tự khác rỗng. Sau ký hiệu = là cận dưới và cận trên của biến đếm. Các cận trên và cận dưới là các số nguyên dương và được đặt giữa từ khóa TO”.
  - Xây dựng dữ liệu thử để kiểm thử câu lệnh FOR theo kỹ thuật kiểm thử lớp tương đương

37

Kiểm thử hộp trắng

- Dựa vào mã nguồn/cấu trúc chương trình
  - Xây dựng dữ liệu thử sau khi mã hóa/lập trình
- Thường phát hiện các lỗi lập trình
- Khó thực hiện
- Chi phí cao

38

Các kỹ thuật kiểm thử hộp trắng

- Kiểm thử dựa trên đồ thị luồng điều khiển
- Kiểm thử dựa trên đồ thị luồng dữ liệu
- Kiểm thử đột biến (mutation testing)

39

Đồ thị luồng điều khiển

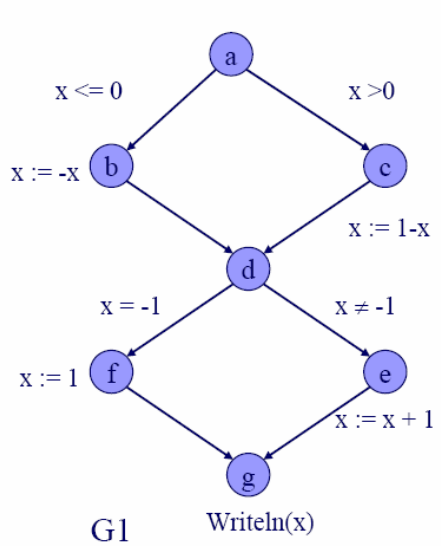
- Đồ thị luồng điều khiển (Control Flow Graph - ĐTLĐK) là đồ thị có hướng, biểu diễn một chương trình
  - đỉnh: biểu diễn lệnh tuần tự hay khối lệnh
  - cung: biểu diễn các rẽ nhánh
  - một đỉnh vào và một đỉnh ra được thêm vào để biểu diễn điểm vào và ra của chương trình
- Lộ trình (path) trong ĐTLĐK
  - xuất phát từ đỉnh vào đi qua các đỉnh và cung trong đồ thị và kết thúc tại đỉnh ra

40

Đồ thị luồng điều khiển

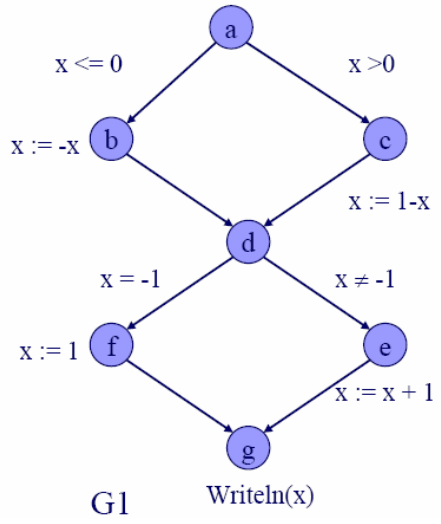
- Ví dụ 1

```
if x <= 0 then
        x := -x
else
        x := 1 -x;
if x = -1 then
        x=1
else
        x := x+1;
writeln(x);
```



Đồ thị luồng điều khiển

- Ví dụ 1 (2)
  - Có 4 lộ trình
    - [a, b, d, f, g]
    - [a, b, d, e, g]
    - [a, c, d, f, g]
    - [a, c, d, e, g]



Đồ thị luồng điều khiển

- Ví dụ 1 (3)
  - Đồ thị G1 có thể biểu diễn dạng biểu thức chính quy:

    G1 = abdfg + abdeg + acdfg + acdeg

  - Hay đơn giản:

    G1 = a(bdf + bde + bdf + bde)g

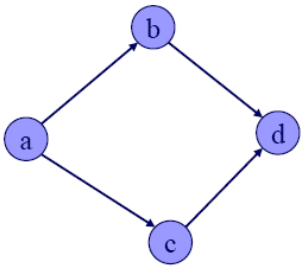
    G1 = a(b + c)d(e + f)g

43

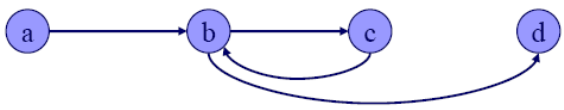
Đồ thị luồng điều khiển

- Biểu diễn các cấu trúc

Cấu trúc tuần tự: ab



Cấu trúc rẽ nhánh: b(a + d)c



Cấu trúc lặp: ab(cb)*d

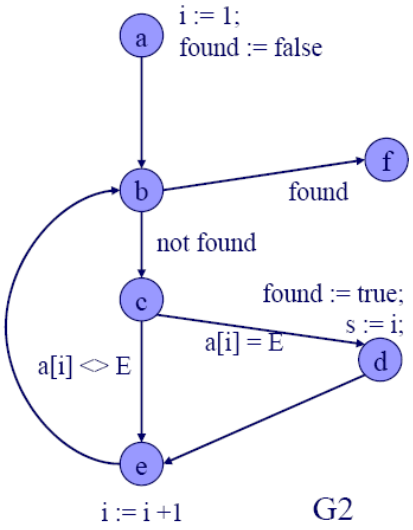
44

Đồ thị luồng điều khiển

- Ví dụ 2

```
i := 1;
found:= false;
while (not found) do
begin
if (a[i] = E) then
   begin
      found:= true;
      s := i;
   end;
   i := i + 1;
end;
```

G2 = ab(c(ε + d)eb)*f



Đồ thị luồng điều khiển

- Bài tập 1
  - Vẽ đồ thị luồng điều khiển
  - Xây dựng biểu thức chính quy biểu diễn đồ thị

```
if n <= 0 then
            n := 1-n
end;
if (n mod 2) = 0 then
            n := n / 2
else
            n := 3*n + 1
end ;
write(n);
```

Đồ thị luồng điều khiển

- Bài tập 2
  - Vẽ đồ thị luồng điều khiển
  - Xây dựng biểu thức chính quy biểu diễn đồ thị

```
read(i);
s := 0;
while(i <= 3) do
begin
        if a[i] > 0 thens := s + a[i];
        i := i + 1;
end
```

47

Kiểm thử dựa trên ĐTLĐK

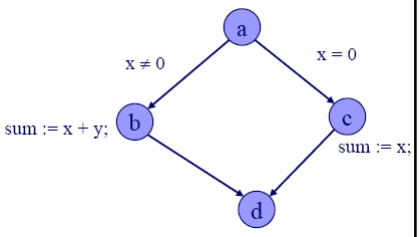
- Các tiêu chuẩn bao phủ
  - Phủ tất cả các đỉnh/lệnh
  - Phủ tất cả các cung
  - Phủ tất cả các quyết định
  - Phủ tất cả các đường đi

48

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các đỉnh/lệnh

- Cho phép phủ tất cả các đỉnh/lệnh
  - mỗi lệnh được thực thi ít nhất một lần
  - tiêu chuẩn tối thiểu

```
functionsum(x,y : integer) : integer;
begin
  if (x = 0) then
          sum := x
  else
          sum := x + y
end;
```



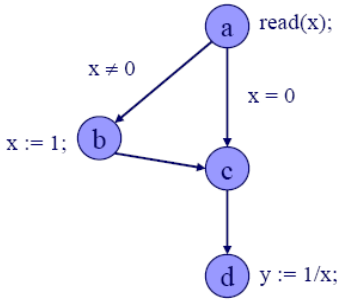
Khi thực thi lộ trình acd sẽ phát hiện lỗi

49

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các đỉnh/lệnh

- Hạn chế của tiêu chuẩn

```
read(x);
…
if (x <> 0) then x := 1;
…
y := 1/x;
```



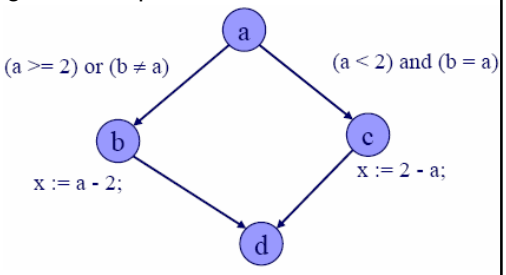
Phủ tất cả các đỉnh không phát hiện được phát hiện lỗi

50

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các cung

- Phủ tất cả các cung ít nhất một lần
  - phủ tất các giá trị đúng sai của một biểu thức lô-gíc
  - phủ tất cả các cung kéo theo phủ tất cả các đỉnh

```
if ((a < 2) and (b = a))
then
        x := 2 -a
else
        x := a -2
```



Dữ liệu thử DT1 = {a=b=1} và DT2 = {a=b=3} thỏa mãn phủ tất cả các cung, nhưng không phủ tất cả các quyết định, chẳng hạn DT3 = {a=3, b=2}

51

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các quyết định

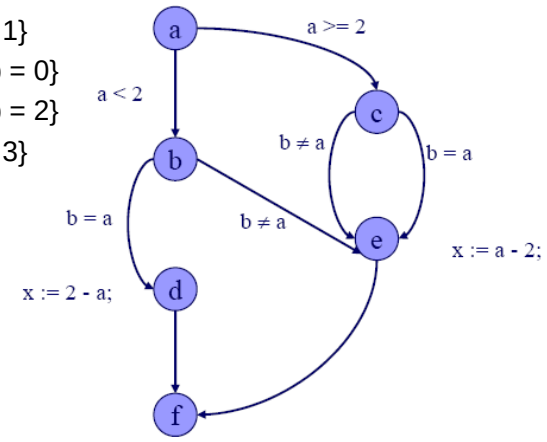
- Phủ tất cả các quyết định được thỏa mãn khi:
  - tiêu chuẩn phủ tất cả các cung được thỏa mãn và
  - mỗi biểu thức con của biểu thức điều kiện được thử với tất cả các giá trị có thể

- Nếu (a AND b)
  - a = b = true
  - a = b = false
  - a = true, b = false
  - a = false, b = true

52

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các quyết định

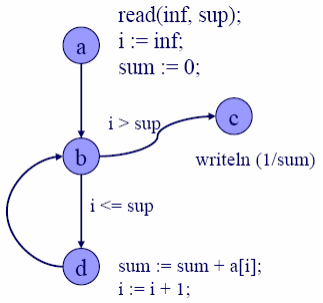
- Dữ liệu thử
  - DT1 = {a = b = 1}
  - DT2 = {a = 1, b = 0}
  - DT3 = {a = 3, b = 2}
  - DT4 = {a = b = 3}



Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các quyết định

- Hạn chế
  - Không phát hiện lỗi trường hợp không thực thi vòng lặp

```
read(inf, sup);
i := inf;
sum:= 0;
while(i <= sup) do
begin
        sum:= sum+ a[i];
        i := i + 1;
end;
writeln(1/sum);
```



Dữ liệu thử DT1 = {a[1]=50, a[2]=60, a[3]=80, inf=1, sup=3} phủ tất cả các cung/quyết định, nhưng không phát hiện lỗi

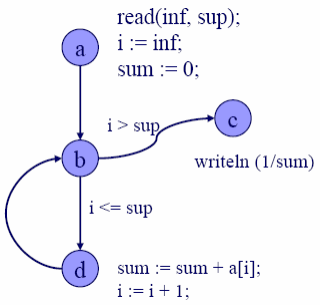
Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các lộ trình

- Mỗi lộ trình phải được thực thi ít nhất một lần
- Gặp khó khăn khi số lần lặp vô hạn
  - Chỉ thực hiện một số lần lặp nhất định
  - Hoặc chỉ thực hiện hai loại lộ trình
    - các lộ trình vượt qua vòng lặp nhưng không lặp
    - các lộ trình chỉ lặp n lần (chẳng hạn n = 1)

55

Kiểm thử dựa trên ĐTLĐK
Phủ tất cả các lộ trình

- Dữ liệu thử
  - DT1 = {a[1]=50, a[2]=60, a[3]=80, inf=1, sup=3}
  - DT2 = {a[1]=50, a[2]=60, a[3]=80, inf=3, sup=2}



56

Kiểm thử dựa trên ĐTLĐK
Bài tập

- Xây dựng dữ liệu thử thỏa mãn các tiêu chuẩn
  - phủ tất cả các đỉnh
  - phủ tất cả các cung
  - phủ tất cả các lộ trình

```
if n ≤ 0 then
        n := 1-n
end;
if (n mod 2) = 0
then
        n := n / 2
else
        n := 3*n + 1
end ;
write(n);
```

57

Kiểm thử dựa trên ĐTLĐK
Bài tập

- Xây dựng dữ liệu thử thỏa mãn các tiêu chuẩn phủ tất cả các lộ trình

```
function goodstring(var count : integer) : boolean;
var ch : char;
begin
  goodstring := false;
  count := 0;
  read(ch);
  if ch = 'a' then
    begin
        read(ch)
        while(ch = 'b') or (ch = 'c') do begin
          count := count + 1;
          read(ch);
        end;
        if ch = 'x' then goodstring = true;
    end;
end;
```

58