**BTVN kiểm thử dòng dữ liệu**

**Câu 1:** Quy trình kiểm thử dòng dữ liệu động:

Kiểm thử dòng dữ liệu động (Dynamic Data Flow Testing) là phương pháp kiểm thử được thực hiện trên chương trình đang chạy, nhằm phát hiện các lỗi liên quan đến quá trình khởi tạo, gán giá trị và sử dụng biến trong suốt vòng đời thực thi. Phương pháp này giúp đảm bảo tính đúng đắn của dữ liệu, cũng như sự hợp lý của các phép toán và điều kiện trong chương trình.

Quy trình tổng quát của kiểm thử dòng dữ liệu động thường bao gồm các bước sau:

Bước 1. Xây dựng đồ thị luồng điều khiển (Control Flow Graph – CFG)

- Từ mã nguồn của chương trình, tiến hành biểu diễn cấu trúc điều khiển bằng đồ thị CFG.

- Mỗi đỉnh (node) trong đồ thị tương ứng với một câu lệnh hoặc khối lệnh, còn cạnh (edge) thể hiện luồng điều khiển có thể xảy ra giữa các câu lệnh. - Việc xây dựng CFG là nền tảng để xác định các mối quan hệ định nghĩa – sử dụng (def-use) của các biến.

Bước 2. Lựa chọn tiêu chí kiểm thử luồng dữ liệu- Tùy theo mục tiêu và mức độ bao phủ mong muốn, người kiểm thử lựa chọn tiêu chí phù hợp, chẳng hạn như:

+ All-defs coverage

+ All-c-uses coverage

+ All-p-uses coverage

+ All-uses coverage

+ All-du-paths coverage

-> Mỗi tiêu chí quy định phạm vi cần kiểm tra giữa các điểm định nghĩa (def) và sử dụng (use) của biến.

Bước 3. Xác định các đường đi trên CFG thoả mãn tiêu chí kiểm thử đã chọn

- Dựa trên CFG, xác định các đường đi def-clear — tức là những đường đi mà trên đó biến không bị định nghĩa lại trước khi được sử dụng.

- Từ đó, liệt kê các cặp def-use (du-pairs) tương ứng và chọn ra những đường đi cần kiểm thử theo tiêu chí đã chọn.

Bước 4. Sinh các ca kiểm thử tương ứng

- Từ các đường đi đã xác định, thiết kế các bộ dữ liệu đầu vào sao cho khi chương trình được chạy, các đường đi def-clear và du-pair được thực hiện.

- Mục tiêu là đảm bảo mỗi biến được gán và sử dụng trong ngữ cảnh đúng đắn, giúp phát hiện các lỗi như:

+ Biến sử dụng trước khi khởi tạo giá trị,

+ Biến được gán giá trị nhưng không bao giờ sử dụng,

+ Biến bị gán lại mà giá trị cũ chưa được dùng.

=> Kiểm thử dòng dữ liệu động là một bước quan trọng nhằm đánh giá tính hợp lệ của luồng dữ liệu trong chương trình. Việc tuân thủ quy trình từ xây dựng CFG đến sinh ca kiểm thử giúp phát hiện sớm các lỗi tiềm ẩn liên quan đến biến, góp phần nâng cao chất lượng và độ tin cậy của phần mềm.

**Câu 2:**

1. Đồ thị dòng điều khiển (CFG)

A diagram of a flowchart

AI-generated content may be incorrect.

2. Xác định các du-pairs cho biến X và Y

Các du-pairs của X:

(1, 3): Đường đi <1, 2(T), 3>

(1, 4): Đường đi <1, 2(T), 3(T), 4>

(1, 6): Đường đi <1, 2(F), 6>

(5, 3): Đường đi <5, 2(T), 3>

(5, 4): Đường đi <5, 2(T), 3(T), 4>

(5, 6): Đường đi <5, 2(F), 6>

Các du-pairs của Y:

(1, 2): Đường đi <1, 2>

(1, 4): Đường đi <1, 2(T), 3(T), 4>

(1, 6): Đường đi <1, 2(F), 6>

(4, 2): Đường đi <4, 2>

(4, 4): Đường đi lặp <4, 2(T), 3(T), 4>

(4, 6): Đường đi <4, 2(F), 6>

3. Sinh đường đi và các ca kiểm thử với độ đo all-use

Ca kiểm thử 1: Không vào vòng lặp

Mục tiêu: Bao phủ các du-pair kết thúc ở nút 6 mà không đi qua vòng lặp.

Đường đi: 1 -> 2(F) -> 6

Input: X = 5, Y = 0 (Để Y > 0 là sai ngay từ đầu)

Các du-pair được bao phủ: (1,6) cho X; (1,2) và (1,6) cho Y.

Ca kiểm thử 2: Lặp 1 lần, X > 0

Mục tiêu: Bao phủ các du-pair từ def ban đầu (nút 1) đến các use trong vòng lặp.

Đường đi: 1 -> 2(T) -> 3(T) -> 4 -> 2(F) -> 6

Input: X = 10, Y = 5

Lần 1: Y=5 > 0 (True), X=10 > 0 (True). Y được gán lại Y = 5 - 10 = -5.

Lần 2: Y=-5 > 0 (False). Thoát lặp.

Các du-pair được bao phủ: (1,3) và (1,4) cho X; (1,4), (4,2) và (4,6) cho Y.

Ca kiểm thử 3: Lặp nhiều lần và có thay đổi giá trị X

Mục tiêu: Bao phủ các du-pair bắt nguồn từ def ở nút 5 và các du-pair có tính lặp lại.

Đường đi: 1 -> 2(T) -> 3(F) -> 5 -> 2(T) -> 3(T) -> 4 -> 2(F) -> 6

Input:

Lần 1 (nút 1): X = -1, Y = 10

Lần 2 (nút 5): X = 4

Y=10 > 0 (True), X=-1 > 0 (False) -> đi vào nút 5, nhập X mới là 4.

Y=10 > 0 (True), X=4 > 0 (True) -> Y được gán lại Y = 10 - 4 = 6.

Y=6 > 0 (True), X=4 > 0 (True) -> Y được gán lại Y = 6 - 4 = 2.

Y=2 > 0 (True), X=4 > 0 (True) -> Y được gán lại Y = 2 - 4 = -2.

Y=-2 > 0 (False) -> Thoát lặp.

Các du-pair được bao phủ: (5,3), (5,4), (5,6) cho X; (4,4) cho Y.

Với 3 ca kiểm thử trên, chúng ta đã đi qua tất cả các du-pairs đã liệt kê, do đó đã đạt được 100% độ phủ All-Uses.

**Câu 3:**

Liệt kê các câu lệnh def, c-use, và p-use:

1. Biến n:

def:

int calFactorial(int n)

p-use:

while (i <= n)

2. Biến result:

def:

int result = 1;

result = result \* i;

c-use:

result = result \* i;

return result;

3. Biến i:

def:

int i = 1;

i++;

c-use:

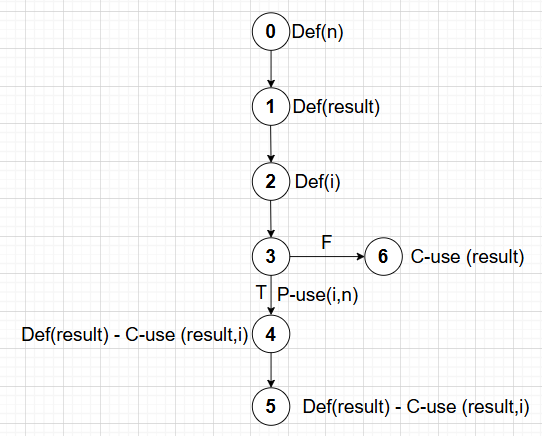
result = result \* i;

i++;

p-use:

while (i <= n)

Đồ thị dòng dữ liệu của hàm là:



**Câu 4:**

1. Các Def-clear-path của biến x và y là:

Biến x: def(x) = {0, 3}

Các Def-clear-path của biến x tại nút 0:

<0, 1>

<0, 1, 2>

<0, 1, 3>

<0, 1, 2, 4>

<0, 1, 2, 4, 5>

<0, 1, 2, 4, 6>

Các Def-clear-path của biến x tại nút 3:

<3, 4>

<3, 4, 5>

<3, 4, 6>

<3, 4, 5, 6>

Biến y: def(y) = {0, 2, 5}

Các Def-clear-path của biến y tại nút 0:

<0, 1>

<0, 1, 3>

<0, 1, 3, 4>

<0, 1, 3, 4, 6>

<0, 1, 3, 4, 5>

Các Def-clear-path của biến y tại nút 2:

<2, 4>

<2, 4, 5>

<2, 4, 6>

Các Def-clear-path của biến y tại nút 5:

<5, 6>

2. Tất cả các du-paths của các biến x và y là:

Đối với biến x:

def(x) = {0, 3}

use(x) = {1(p-use), 3(c-use), 4(p-use), 5(c-use)}

Các du-paths:

Từ Def tại 0:

<0, 1> (đến p-use tại 1)

<0, 1, 3> (đến c-use tại 3)

<0, 1, 2, 4> (đến p-use tại 4)

<0, 1, 2, 4, 5> (đến c-use tại 5)

Từ Def tại 3:

<3, 4> (đến p-use tại 4)

<3, 4, 5> (đến c-use tại 5)

Đối với biến y:

def(y) = {0, 2, 5}

use(y) = {1(p-use), 4(p-use), 6(c-use)}

Các du-paths:

Từ Def tại 0:

<0, 1> (đến p-use tại 1)

<0, 1, 3, 4> (đến p-use tại 4)

<0, 1, 3, 4, 6> (đến c-use tại 6)

<0, 1, 3, 4, 5, 6> (đến c-use tại 6)

Từ Def tại 2:

<2, 4> (đến p-use tại 4)

<2, 4, 6> (đến c-use tại 6)

<2, 4, 5, 6> (đến c-use tại 6)

Từ Def tại 5:

<5, 6> (đến c-use tại 6)

3. Xác định tất cả các cho All-p-uses/Some-c-uses và All-c-uses/Some-p-uses

All-p-uses/Some-c-uses:

Biến x:

def(x) = {0, 3}

p-use(x) = {1, 4}

|  |  |  |
| --- | --- | --- |
| Cặp Du-pair (def, p-use) | Def-clear path | Đường đi hoàn chỉnh (Complete path) |
| (0, 1) | <0, 1> | <0, 1, 2, 4, 6> |
| (0, 4) | <0, 1, 2, 4> | <0, 1, 2, 4, 6> |
| (3, 4) | <3, 4> | <0, 1, 3, 4, 6> |

Biến y:

def(y) = {0, 2, 5}

p-use(y) = {1, 4}

c-use(y) = {6}

|  |  |  |  |
| --- | --- | --- | --- |
| Cặp Du-pair (def, use) | Loại Use | Def-clear path | Đường đi hoàn chỉnh (Complete path) |
| (0, 1) | p-use | <0, 1> | <0, 1, 3, 4, 6> |
| (0, 4) | p-use | <0, 1, 3, 4> | <0, 1, 3, 4, 6> |
| (2, 4) | p-use | <2, 4> | <0, 1, 2, 4, 5, 6> |
| (5, 6) | c-use | <5, 6> | <0, 1, 2, 4, 5, 6> |

All-c-uses/Some-p-uses:

Biến x:

def(x) = {0, 3}

c-use(x) = {3, 5}

|  |  |  |
| --- | --- | --- |
| Cặp Du-pair (def, c-use) | Def-clear path | Đường đi hoàn chỉnh (Complete path) |
| (0, 3) | <0, 1, 3> | <0, 1, 3, 4, 6> |
| (0, 5) | <0, 1, 2, 4, 5> | <0, 1, 2, 4, 5, 6> |
| (3, 5) | <3, 4, 5> | <0, 1, 3, 4, 5, 6> |

Biến y:

def(y) = {0, 2, 5}

c-use(y) = {6}

p-use(y) = {1, 4}

|  |  |  |  |
| --- | --- | --- | --- |
| Cặp Du-pair (def, use) | Loại Use | Def-clear path | Đường đi hoàn chỉnh (Complete path) |
| (0, 6) | c-use | <0, 1, 2, 4, 6> | <0, 1, 2, 4, 6> |
|  |  | <0, 1, 3, 4, 6> | <0, 1, 3, 4, 6> |
| (2, 6) | c-use | <2, 4, 6> | <0, 1, 2, 4, 6> |
|  |  | <2, 4, 5, 6> | <0, 1, 2, 4, 5, 6> |
| (5, 6) | c-use | <5, 6> | <0, 1, 2, 4, 5, 6> |

4. Đường đi (0 - 1 - 3 - 4 - 5 - 6) là khả thi

Chứng minh bằng ví dụ:

Bước 1 (Nút 0): Giả sử chương trình bắt đầu với x = 1, y = 3.

Bước 2 (Cạnh 1 -> 3): Kiểm tra điều kiện x + y = 4. Ta có 1 + 3 = 4. Điều kiện ĐÚNG. Chương trình đi theo nhánh này.

Bước 3 (Nút 3): Tại đây, biến x được định nghĩa lại (Def(x)). Giá trị của y không đổi (y vẫn là 3). Giả sử câu lệnh tại đây là x = x \* 5. Vậy giá trị mới của x sẽ là 1 \* 5 = 5.

Bước 4 (Cạnh 4 -> 5): Chương trình đi đến nút 4 với x = 5 và y = 3. Kiểm tra điều kiện x² + y² > 17. Ta có 5² + 3² = 25 + 9 = 34. Vì 34 > 17, điều kiện ĐÚNG. Chương trình đi theo nhánh này.

Vì ta đã tìm được một bộ giá trị (x=1, y=3) và một phép toán (x = x \* 5) làm cho chương trình thực thi đúng theo đường đi 0-1-3-4-5-6, nên đường đi này là khả thi.

5. Không có mối quan hệ def-use (3, 3) vì:

Việc "sử dụng" và "định nghĩa" tại nút 3 không đề cập đến cùng một giá trị của x. use là giá trị cũ đi vào, def là giá trị mới đi ra.

Không có đường đi (path) nào từ điểm "sau khi định nghĩa" tại nút 3 quay trở lại điểm "trước khi sử dụng" tại chính nút 3. Dữ liệu luôn chảy về phía trước trong đồ thị.

Vì vậy, mặc dù Def(x) và C-use(x) cùng xuất hiện tại nút 3, chúng thuộc về các cặp def-use khác nhau: use tại 3 liên kết với def tại 0, và def tại 3 liên kết với các use trong tương lai (ví dụ như tại nút 4 và 5).

**Câu 5:**

1. Xây dựng CFG cho hàm UCLN với đồ thị C2

A diagram of a flowchart

AI-generated content may be incorrect.

2. Đường đi và ca kiểm thử với độ đo C2 (Phủ nhánh)

Path 1: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(F) -> 10(T) -> 11(T) -> 12 -> 10(F) -> 14

Test case 1: UCLN(6,3)

Path 2: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(F) -> 10(T) -> 11(F) -> 13 -> 10(F) -> 14

Test case 2: UCLN(3,6)

Path 3: 1 -> 2(T) -> 3 -> 4(T) -> 5 -> 6(F) -> 8(F) -> 10(T) -> 11(F) -> 13 -> 10(F) -> 14

Test case 3: UCLN(-5,-10)

Path 4: 1 -> 2(F) -> 4(F) -> 6(T) -> 7

Test case 4: UCLN(0,5)

Path 5: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(T) -> 9

Test case 5: UCLN(5,0)

3. Đường đi và ca kiểm thử với độ đo all-def coverage

- Biến m: def(m) = {1, 3, 9, 12}; c-use(m) = {3, 12, 13, 14}; p-use(m) = {2, 6, 10, 11}

- Biến n: def(n) = {1, 5, 7, 13}; c-use(m) = {5, 7, 12, 13}; p-use(m) = {4, 8, 10, 11}

Ca kiểm thử 1: UCLN(m = 6, n = 3)

Mục đích: Bao phủ def(m, 12) và def(m, 1).

Đường đi: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(F) -> 10(T) -> 11(T) -> 12 -> 10(F) -> 14

Ca kiểm thử 2: UCLN(m = 3, n = 6)

Mục đích: Bao phủ def(n, 13).

Đường đi: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(F) -> 10(T) -> 11(F) -> 13 -> 10(F) -> 14

Ca kiểm thử 3: UCLN(m = -5, n = 5)

Mục đích: Bao phủ def(m, 3) và def(n, 1).

Đường đi: 1 -> 2(T) -> 3 -> 4(F) -> 6(F) -> 8(F) -> 10(F) -> 14

Ca kiểm thử 4: UCLN(m = 5, n = -5)

Mục đích: Bao phủ def(n, 5).

Đường đi: 1 -> 2(F) -> 4(T) -> 5 -> 6(F) -> 8(F) -> 10(F) -> 14

Ca kiểm thử 5: UCLN(m = 0, n = 10)

Mục đích: Bao phủ def(n, 7).

Đường đi: 1 -> 2(F) -> 4(F) -> 6(T) -> 7

Ca kiểm thử 6: UCLN(m = 5, n = 0)

Mục đích: Bao phủ def(m, 9).

Đường đi: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(T) -> 9

Ca kiểm thử 7: UCLN(m = 7, n = 7)

Mục đích: Bao phủ def(m, 14).

Đường đi: 1 -> 2(F) -> 4(F) -> 6(F) -> 8(F) -> 10(F) -> 14

Bộ 7 ca kiểm thử này đảm bảo rằng mọi điểm gán giá trị (định nghĩa) cho cả hai biến m và n đều được thực thi và giá trị đó được sử dụng ở một bước tiếp theo.

**Câu 6:**

CFG cho chương trình của tôi với đồ thị C2

A diagram of a program

AI-generated content may be incorrect.

Đường đi và ca kiểm thử với độ đo all-uses

1. Biến gia\_ve

Defs (Điểm gán giá trị):

Node 1: int gia\_ve = 100000;

Node 11: gia\_ve = 90000; (Khi là sinh viên)

Node 13: gia\_ve = 80000; (Khi tuổi được giảm giá)

Uses (Điểm sử dụng):

Node 14 (c-use): System.out.println("..." + gia\_ve);

2. Biến tuoi

Defs:

Node 2: int tuoi = scanner.nextInt();

Uses:

Node 4 (p-use): if (tuoi < 0)

Node 12 (p-use): if (tuoi < 12 || tuoi >= 60)

3. Biến isStudent

Defs:

Node 8: isStudent = scanner.nextBoolean();

Uses:

Node 10 (p-use): if (isStudent)

Các cặp Def-Use (du-pairs):

Với gia\_ve:

(1, 14): Giá trị gốc (100000) được dùng để in ra.

(11, 14): Giá trị sau khi giảm cho sinh viên (90000) được dùng để in ra.

(13, 14): Giá trị sau khi giảm theo tuổi (80000) được dùng để in ra.

Với tuoi:

(2, 4): Tuổi nhập vào được dùng trong điều kiện tuoi < 0.

(2, 12): Tuổi nhập vào được dùng trong điều kiện tuoi < 12 || tuoi >= 60.

Với isStudent:

(8, 10): Trạng thái sinh viên được dùng trong điều kiện if (isStudent).

Ca kiểm thử để bao phủ All-Uses:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Mục tiêu | Đầu vào (tuoi, isStudent) | Đường đi (Path) | Kết quả mong đợi | Các du-pair được bao phủ |
| TC1 | Kiểm tra giá vé gốc, không giảm giá. | tuoi = 25  isStudent = false | 1-2-3-4(F)-6-7-8-10(F)-12(F)-14 | "Giá vé bạn phải trả là: 100000" | (gia\_ve: 1, 14), (tuoi: 2, 12), (isStudent: 8, 10) |
| TC2 | Kiểm tra giảm giá chỉ cho sinh viên. | tuoi = 20  isStudent = true | 1-2-3-4(F)-6-7-8-10(T)-11-12(F)-14 | "Giá vé bạn phải trả là: 90000" | (gia\_ve: 11, 14), (tuoi: 2, 12), (isStudent: 8, 10) |
| TC3 | Kiểm tra giảm giá tuổi (trẻ em), ghi đè giảm giá sinh viên. | tuoi = 10  isStudent = true | 1-2-3-4(F)-6-7-8-10(T)-11-12(T)-13-14 | "Giá vé bạn phải trả là: 80000" | (gia\_ve: 13, 14), (tuoi: 2, 12), (isStudent: 8, 10) |
| TC4 | Kiểm tra tuổi không hợp lệ. | tuoi = -5 | 1-2-3-4(T)-5 | "Input không hợp lệ" | (tuoi: 2, 4) |
| TC5 | Kiểm tra giảm giá tuổi (người lớn tuổi). | tuoi = 65  isStudent = false | 1-2-3-4(F)-6-7-8-10(F)-12(T)-13-14 | "Giá vé bạn phải trả là: 80000" | (gia\_ve: 13, 14), (tuoi: 2, 12), (isStudent: 8, 10) |

Bộ 5 ca kiểm thử trên đã thực thi các đường đi bao phủ tất cả 6 cặp def-use đã được xác định.

TC1 đã bao phủ cặp (gia\_ve: 1, 14).

TC2 đã bao phủ cặp (gia\_ve: 11, 14).

TC3 đã bao phủ cặp (gia\_ve: 13, 14). (TC5 cũng bao phủ cặp này nhưng cho trường hợp khác).

TC4 đã bao phủ cặp (tuoi: 2, 4).

Tất cả các ca kiểm thử từ TC1, TC2, TC3, và TC5 đều đi qua các cặp (tuoi: 2, 12) và (isStudent: 8, 10).

=> Bộ ca kiểm thử này đã thỏa mãn tiêu chí độ phủ All-Uses.