KIỂM THỬ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM

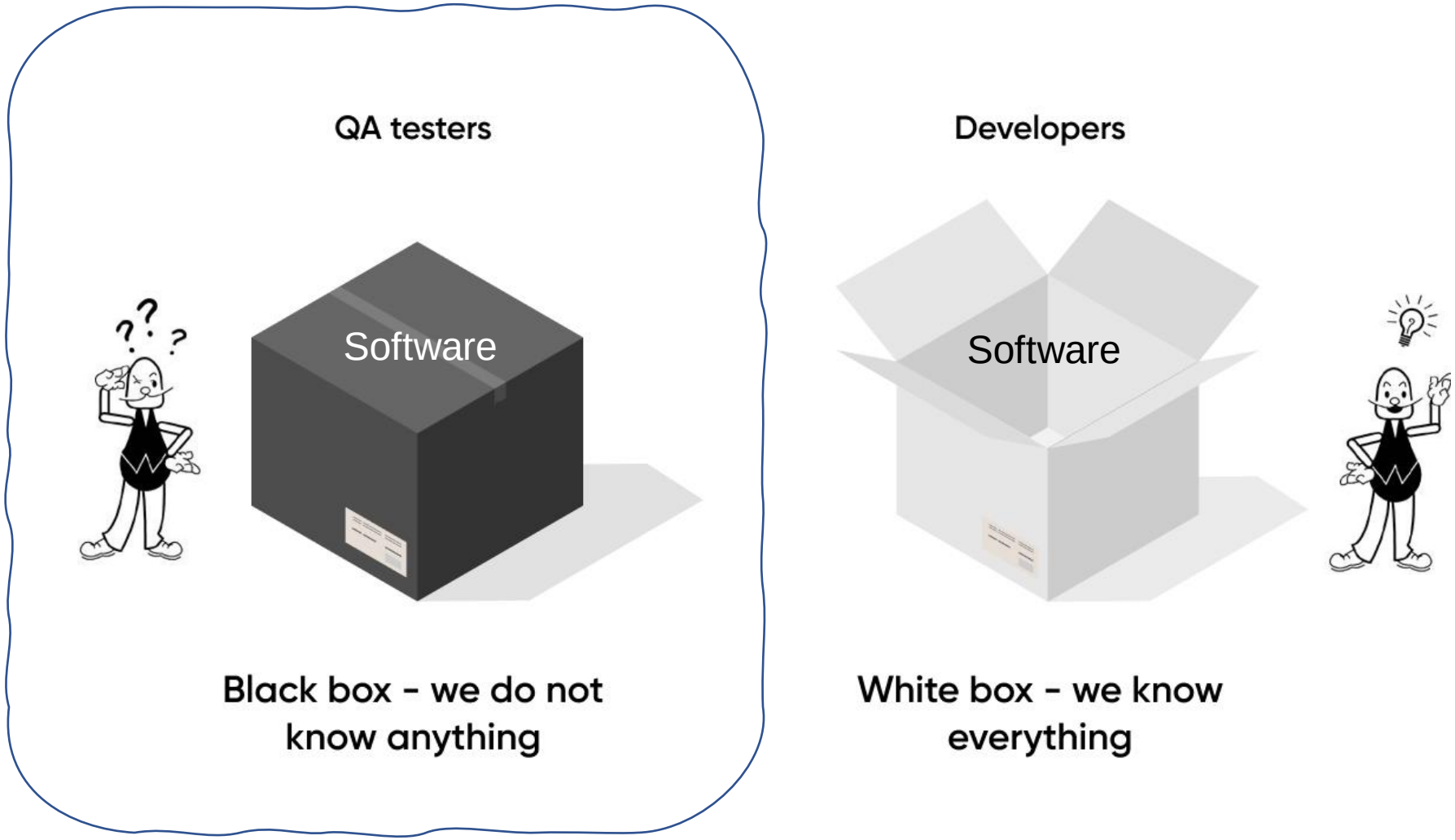
INT3117

Bài giảng 03: Black-box testing

Hôm nay…

- Các kỹ thuật sinh test black-box (black-box testing)
  - Kiểm thử vét cạn (Exhaustive testing)
  - Kiểm thử ngẫu nhiên (Random testing)
  - Kiểm thử ngẫu nhiên thích ứng (Adaptive random testing)
  - Phân lớp tương đương (Equivalence class partitioning)
  - Phân tích giá trị biên (Boundary value analysis)
  - Bảng quyết định (Dicision table – ***tự học***)
- Tài liệu tham khảo
  - Giáo trình Kiểm thử Phần mềm (PNH, TAH & DVH, 2014)
  - Introduction to Software Testing (PA & JO, 2016)

KIỂM THỬ HỘP ĐEN vs. KIỂM THỬ HỘP TRẮNG



Kiểm thử hộp đen

- Xác định các chức năng kỳ vọng của phần mềm
- Tạo ra dữ liệu kiểm thử kiểm tra xem các chức năng đã xác định có được thực thi đúng bởi phần mềm
- Không quan tâm tới cách chương trình thực hiện các chức năng, chương trình được xem như hộp đen

Test như nào?

```
int sum (int a, int b) {return a + b; }
```

Kiểm thử vét cạn?

int sum (int a, int b) {return a + b; }

- Kiểm thử vét cạn kiểm tra tất cả giá trị đầu vào có thể
- Cho hàm sum(int a, int b):
  - Có 2^32 giá trị cho a và 2^32 giá trị cho b
  - 2^32 * 2^32 đầu vào
  - Tốn 1 nano giây (10^-9 giây) cho mỗi test
  - ~585 năm để hoàn thành việc kiểm thử vét cạn

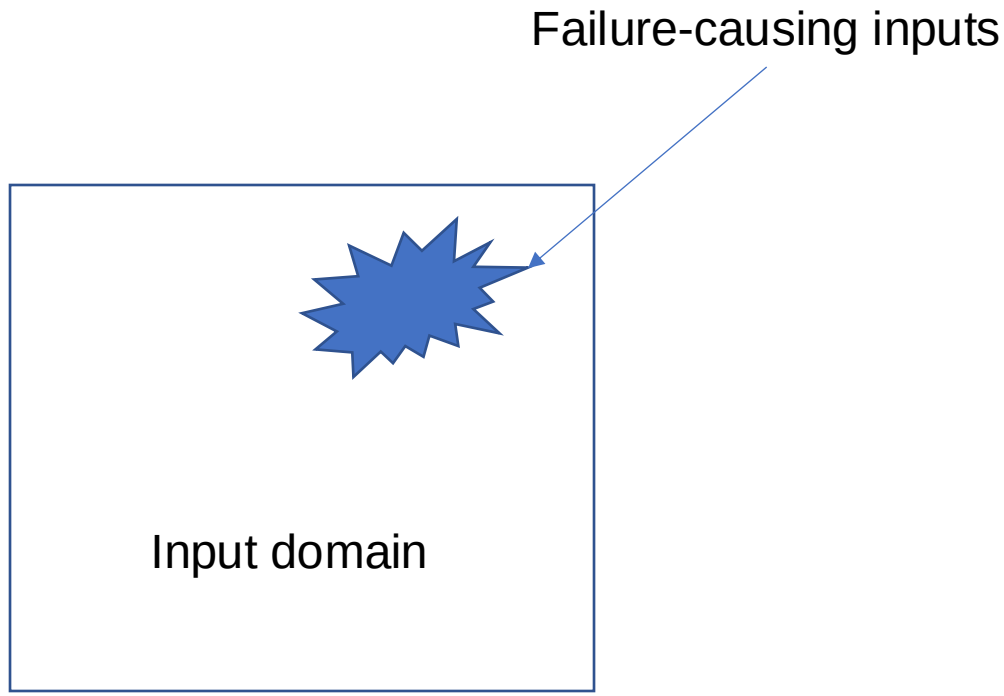
Kiểm thử ngẫu nhiên – Monkey testing

Kiểm thử ngẫu nhiên

- Sử dụng các bộ sinh dữ liệu ngẫu nhiên (khỉ – monkey) để sinh tests
- Sử dụng các mô hình xác suất để đánh giá độ tin cậy của phần mềm và bộ test
  - Cần tới số lượng tests lớn để đảm bảo xác suất lỗi của phần mềm không vượt quá ngưỡng
  - Ví dụ: cần tới 495 ca kiểm thử pass để đảm bảo chương trình có xác suất lỗi nhỏ hơn 1/100 với độ tự tin (confidence rate) là 99%

Kiểm thử ngẫu nghiên – Các khái niệm

- Miền đầu vào (input domain): tập tất cả giá trị có thể có của đầu vào
- Miền đầu vào gây lỗi (failure-causing input): các input phô bày ra lỗi của chương trình

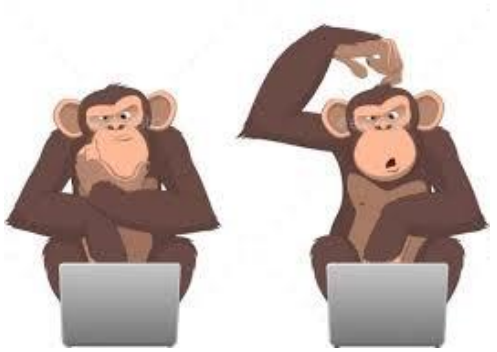


Sinh test ngẫu nhiên

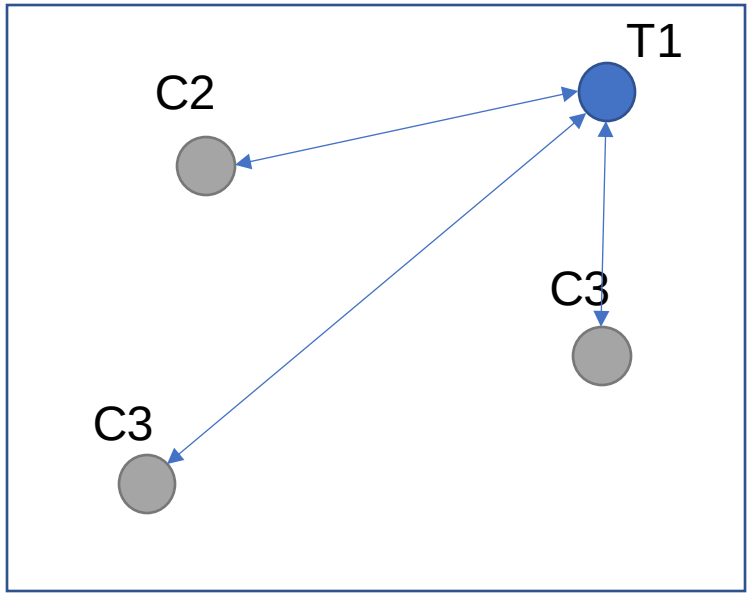
- Lựa chọn các giá trị cho mỗi đầu vào một cách ngẫu nhiên và độc lập với các đầu vào khác
- Điểm mạnh:
  - ???
- Điểm yếu:
  - ???

Làm sao để cải thiện kỹ thuật kiểm ngẫu nhiên?

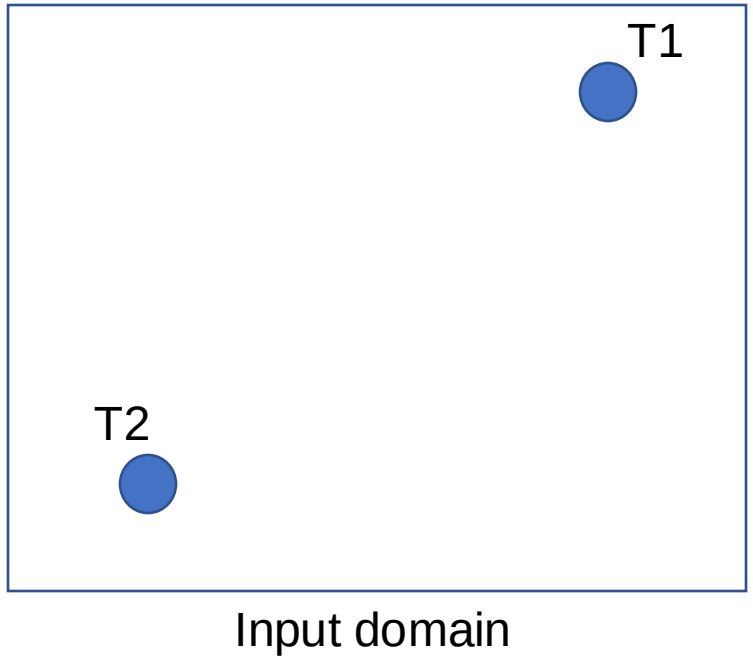
- Phân tán các tests ngẫu nhiên sẽ tăng khả năng phát hiện ra lỗi của bộ test
- Đạt được bằng kỹ thuật kiểm thử ngẫu nhiên thích ứng



Kiểm thử ngẫu nhiên thích ứng



Kiểm thử ngẫu nhiên thích ứng



Lựa chọn dữ liệu kiểm thử thế nào?

Với một bộ test T, và bộ test ứng viên được sinh ngẫu nhiên C,

lựa chọn ứng viên nào trong C để thêm vào T?

- Khoảng cách từ mỗi ứng viên c tới T là khoảng cách c tới test gần c nhất trong T.
- Lựa chọn ứng viên có khoảng cách xa nhất tới T
- Thêm ứng viên được lựa chọn vào T, và bỏ đi trong C

Khoảng cách tính ra sao?

- Khoảng cách ciữa 2 điểm p và q trên hệ trục tọa độ:
  - Euclidean Distance

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

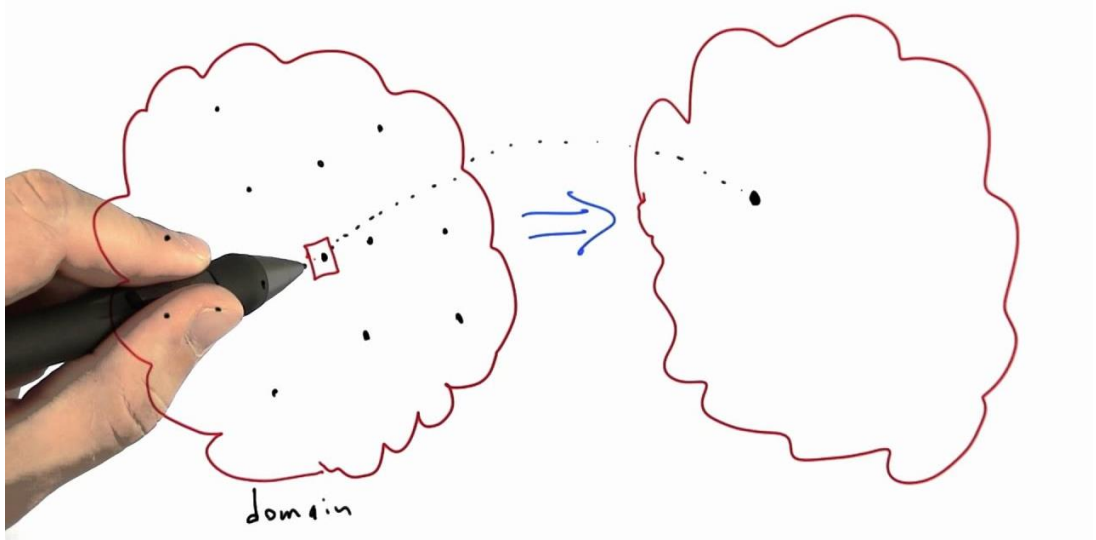
- Khoảng cách giữa 2 chuỗi (string) a và b:
  - Levenshtein Distance: số thao tác tối thiểu để biến a thành b

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}\left(\operatorname{tail}(a), b\right) \\ \operatorname{lev}\left(a, \operatorname{tail}(b)\right) \\ \operatorname{lev}\left(\operatorname{tail}(a), \operatorname{tail}(b)\right) \end{cases} & \text{otherwise}, \end{cases}$$

Bài tập 01

- Sinh ngẫu nhiên 5 test cases cho hàm login: **String** login(**String** username, **String** password)
- Sử dụng kỹ thuật kiểm thử ngẫu nhiên thích ứng để chọn 3 tests để thực thi

Phân lớp tương đương



- Miền đầu vào thường rất lớn, Không thể thử được tất cả giá trị (như kiểm thử vét cạn)
- Lựa chọn một số lượng tương đối nhỏ test cases để chạy
- Lựa chọn các test cases nào?

Lựa chọn các test cases nào?

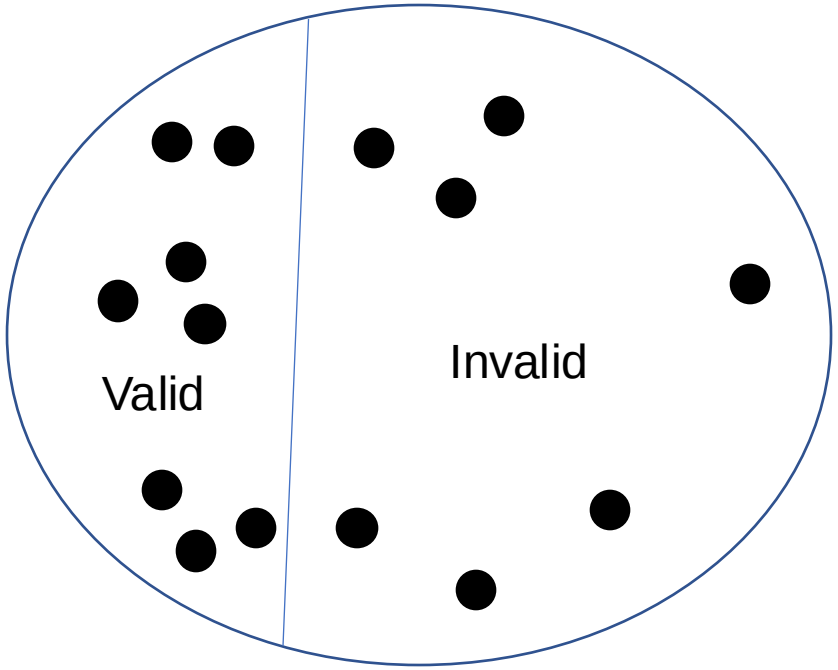
Giả sử có một bộ tất cả tests có thể có

- Phân các bộ test thành các “lớp tương đương”
- Mỗi lớp chứa một tập các test cases tương đương
- Hai tests được coi là tương đương nếu chúng ta kỳ vọng chương trình xử lý theo cùng một kiểu

→ Chỉ cần test 1 trong 2 trường hợp → Giảm thiểu việc thực thi

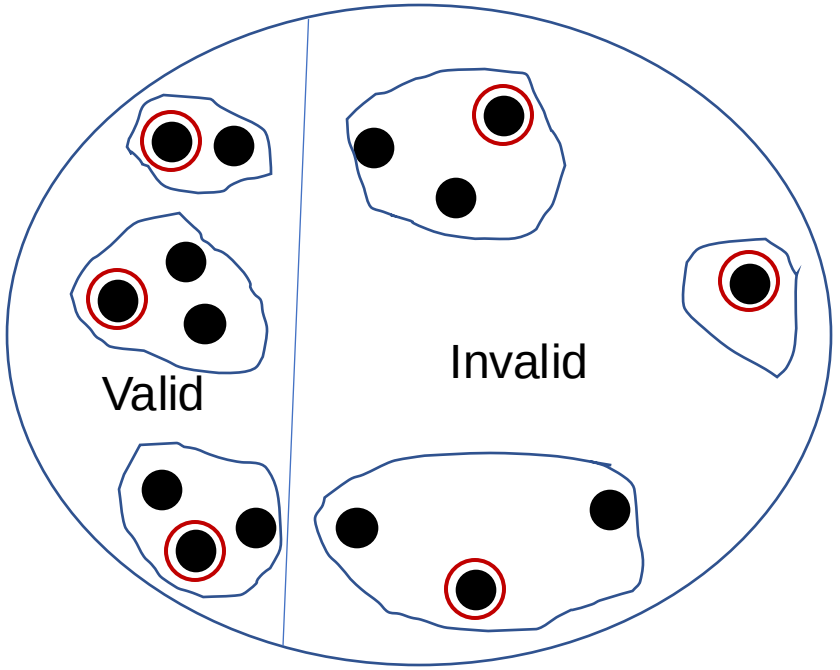
Phân lớp tương đương

- Mức 1: Test cases hợp lệ (valid) và test cases không hợp lệ (invalid)

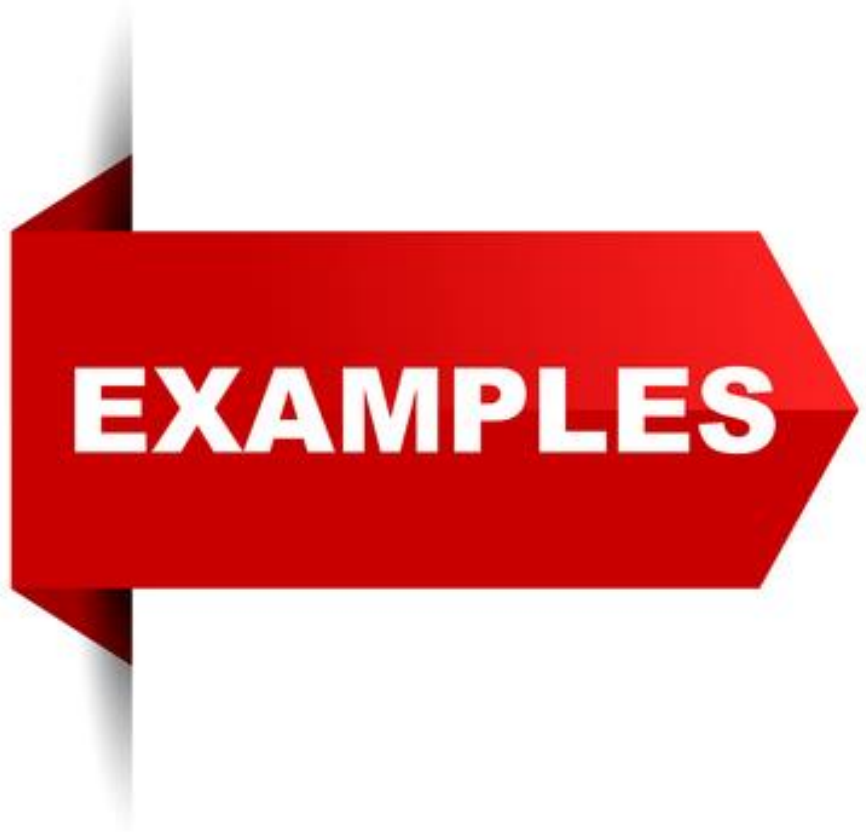


Phân lớp tương đương

- Phân chia các test cases hợp lệ và không hợp lệ thành các lớp tương đương
- Tạo 1 test case cho ít nhất 1 giá trị trong mỗi lớp tương đương
- Có thể dùng nhiều khái niệm “tương đương” ở mỗi chương trình

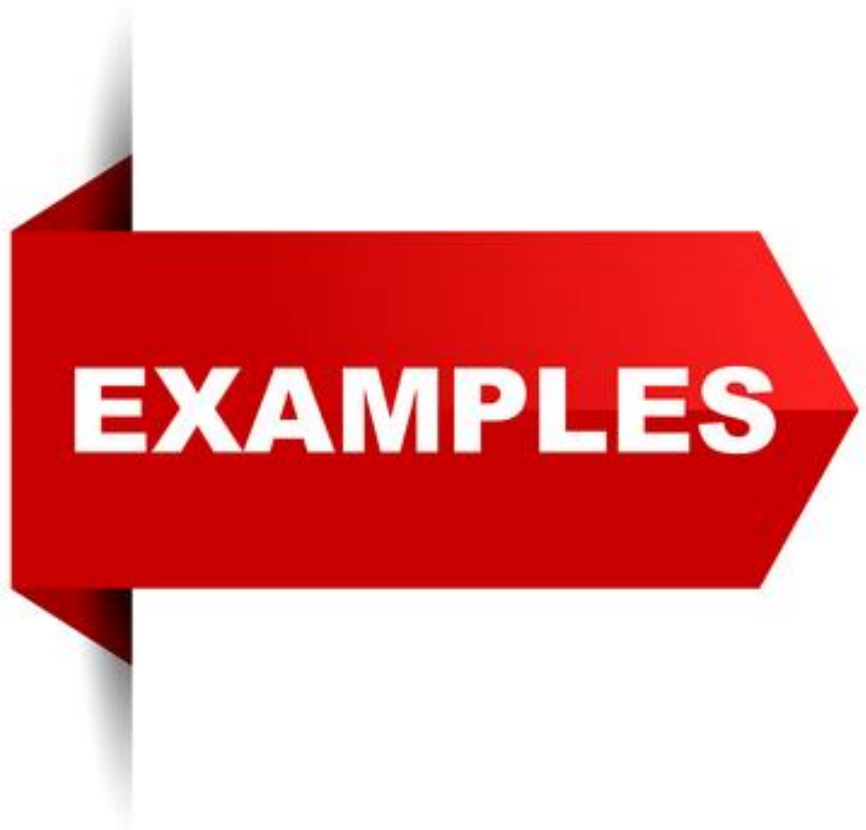


Phân hoạch tương đương – Ví dụ



- int add(int n1, int n2, int n3, …):
  - Khái niệm tương đương 1: phân loại tests bằng dấu của input (dương, âm, 0, tất cả)
  - Khái niệm tương đương 2: phân loại theo độ lớn (số lớn, nhỏ, cả 2)
  - Khái niệm tương đương khác????

Phân hoạch tương đương – Ví dụ



- String fetch (String url):
  - Khái niệm tương đương 1: phân loại theo protocol (http, https, ftp, file, …)
  - Khái niệm tương đương 2: phân loại theo định dạng của file (html, gif, jpeg, text, …)
  - Khái niệm tương đương khác????

Các bước thực hiện

1. Xác định input
2. Chia input thành các “lớp tương đương”
3. **Lựa chọn giá trị đại diện ở các lớp**
4. Tạo các bộ input và tests tương ứng
5. Thực thi

Một số loại kiểm thử phân lớp tương đương

- Kiểm thử lớp tương đương yếu
- Kiểm thử lớp tương đương mạnh
- Kiểm thử lớp tương đương đơn giản

- Tự học, chương 5.3, Giáo trình kiểm thử phần mềm

Bài tập 02

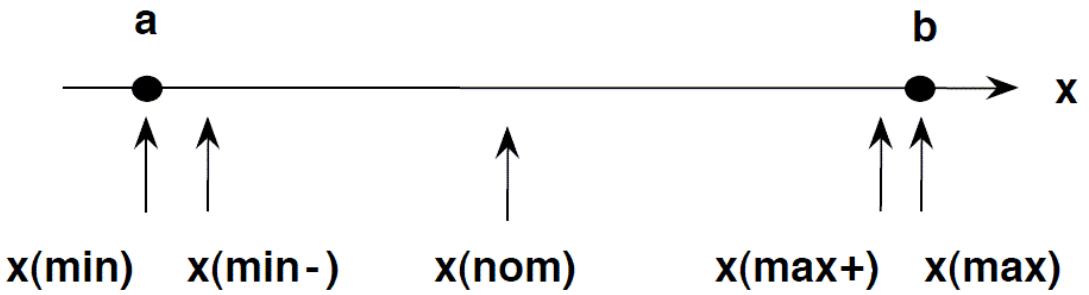
Input	Lớp tương đương hợp lệ	Lớp tương đương không hợp lệ
Một số nguyên N: -99 <= N <=99	?	?
Số điện thoại: Mã vùng: [201, 299] Tiền tố: (100, 999) Hậu tố: 4 chữ số bất kì	?	?

Phân hoach tương đương – Ưu & nhược điểm

- Điểm mạnh: ???
- Điểm yếu: ???

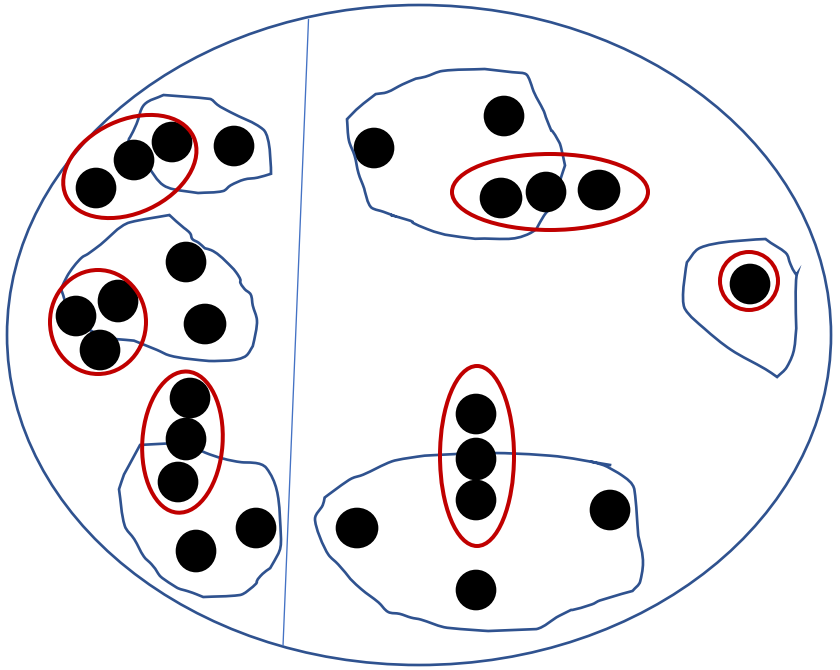
Phân tích giá trị biên

- Lỗi thường xảy ra ở các “biên” của các lớp tương đương, thay vì ở “trung tâm”
  - Kiểm tra mã vùng biển số xe
  - If (11 < areaCode && areaCode < 99) { //Bước kiểm tra sai }
  - If (11 <= areaCode && areaCode <= 99) { //Bước kiểm chính xác }
  - 11 và 99 bắt được lỗi, vị trí trung tâm như 50 có thể không bắt được lỗi
- Ngoài việc kiểm tra giá trị trung tâm, giá trị biên cũng nên được kiểm tra
  - Ngay tại vị trí biên
  - Cận biên



Phân tích giá trị biên

- Tạo ra các test cases để kiểm tra giá trị biên của các lớp tương đương



Các bước thực hiện

1. Xác định input
2. Chia input thành các “lớp tương đương”
3. **Lựa chọn giá trị bình thường, biên và cận biên**
4. Tạo các bộ input và tests tương ứng
5. Thực thi

Một số dạng kiểm thử phân tích giá trị biên

- Kiểm thử giá trị biên mạnh
- Kiểm thử giá trị biên tổ hợp
- Kiểm thử giá trị biên đặc biệt

- Tự học, chương 5.2, Giáo trình kiểm thử phần mềm

Bài tập 03

Input	Trường hợp biên
Một số nguyên N: -99 <= N <=99	?
Số điện thoại: Mã vùng: [201, 299] Tiền tố: (100, 999) Hậu tố: 4 chữ số bất kì	?

Phân tích giá trị biên – Ưu & nhược điểm

- Điểm mạnh: ???
- Điểm yếu: ???

Thảo luận

- Khi nào sử dụng kiểm thử vét cạn?
- Khi nào sử dụng kiểm thử ngẫu nhiên (thích ứng)?
- Khi nào sử dụng phân lớp tương đương và phân tích giá trị biên?

Kiểm thử bảng quyết định

- Tự học
- Chương 5.4, Giáo trình kiểm thử phần mềm

Kiểm thử tổ hợp (Combinatorial Testing)

- Chương trình thực thường có nhiều hơn 1 inputs
  - int decide (bool b1, bool b2, …, bool b10)
  - 2*2*…*2 = 2^10 test cases có thể có
- All-values: Xét tới tất cả giá trị có thể có của mỗi đầu vào:
  - ???
- All-pairs (pairwise – đôi một): Xét tới tất cả giá trị có thể có của bất kỳ tổ hợp 2 input nào:
  - b1-b2, b1-b3 ….
  - Có thể cần tới 10 tests để đạt được điều kiện all-pairs cho ví dụ trên

Kiểm thử tổ hợp – Phân tích

- Sự cố xảy ra khi:
  - pressure < 10
    - Tương tác 1-chiều (1-way) – Tìm ra lỗi bằng sử dùng all-values
  - pressure < 10 & volume > 300
    - Tương tác 2 chiều (2-way) – Tìm ra lỗi bằng sử dụng all-pairs
- Kiểm thử đôi một (pairwise) được áp dụng phổ biến trong công nghiệp
  - Một số vấn đề chỉ xảy ra bởi sự tương tác giữa các đối số đầu vào hoặc components
- Kiểm thử đôi một tìm được từ 50% - 90% lỗi

Tìm được 90% số lỗi là khá ngầu, đúng không?

Chào mừng lên máy bay!

Thôi nào, đừng có lo!

Kỹ sư của chúng tôi đã tìm ra và

xử lý đươc **90%** lỗi rồi!

Kiểm thử tổ hợp – Phân tích

- Sự cố xảy ra khi:
  - pressure < 10
    - Tương tác 1-chiều (1-way) – Tìm ra lỗi bằng sử dùng all-values
  - pressure < 10 & volume > 300
    - Tương tác 2 chiều (2-way) – Tìm ra lỗi bằng sử dụng all-pairs
  - pressure < 10 & volume > 300 & velocity = 5
    - Tương tác 3 chiều (3-way)
  - Sự cố phức tạp nhất từng được báo cáo yêu cầu tương tác 4 chiều

Các công tắc điều khiển máy bay Boing 737

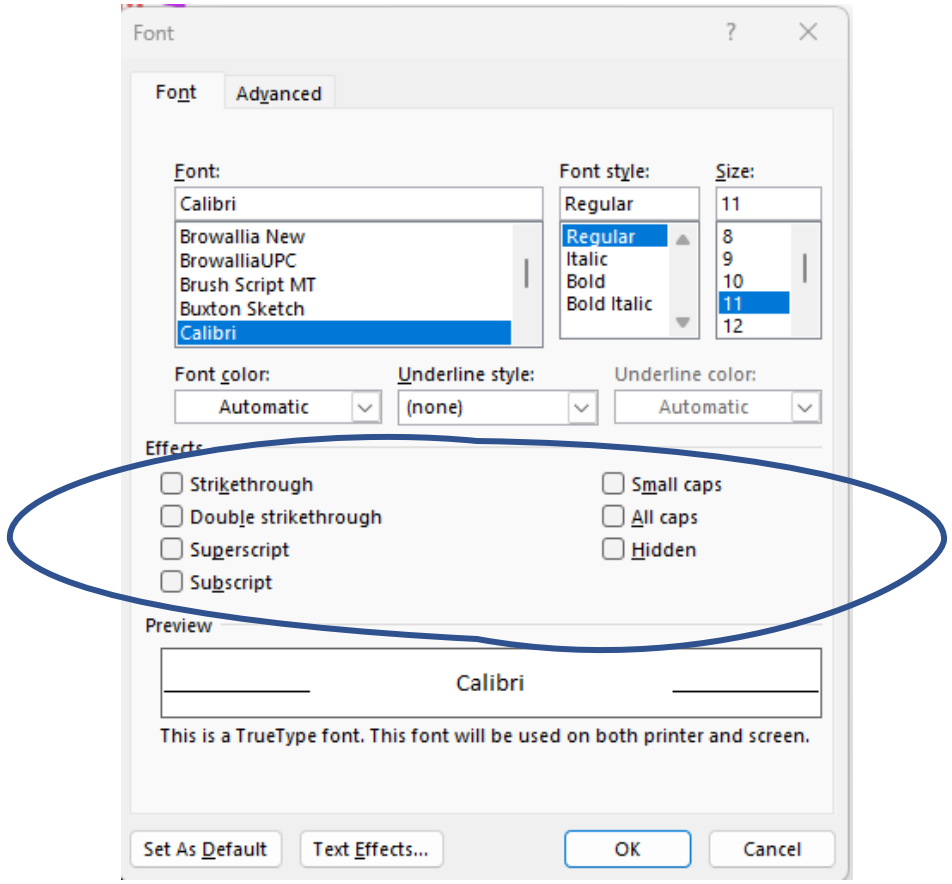
Các công tắc điều khiển máy bay Boing 737

- 34 công tắc → 2^34 = 1.7*10^10 inputs/tests

Không có sự cố nào liên quan tới sự tương tác của nhiều hơn 3 công tắc

- 34 công tắc → 2^34 = 1.7*10^10 inputs/tests
- Chỉ với tương tác 3 chiều: chỉ cần 33 tests
- Với tương tác 4 chiều: chỉ cần 85 tests

Ví dụ đơn giản



Cần test bao nhiêu trường hợp?

- 10 hiệu ứng, on/off cho mỗi hiệu ứng
- Tất cả tổ hợp: 20^10 = 1024 tests
- Kiểm thử tổ hợp tương tác 3 chiều thì cần bao nhiêu tests????

Buổi học sau

- Các kỹ thuật kiểm thử hộp trắng