Toán học rời rạc dành cho các nhân viên kiểm thử

Trần Văn Mạnh, Bùi Hoàng Khánh
Lớp K16-CNPM3, trường ĐHCN, Đại học quốc gia Hà Nội

Tóm tắt nội dung

Toán rời rạc là một lĩnh vực quan trọng trong tin học, để hiểu nó một cách thấu đáo vả sâu sắc quả thật không đơn giản. Bài viết này chỉ tập trung vào nghiên cứu toán rời rạc dành cho các nhân viên kiểm thử phần mềm. Các nhân viên kiểm thử phần mềm coi Toán rời rạc như một công cụ giúp học có được tư duy logic chặt chẽ hơn trong quá trình kiểm thử phần mềm để tránh bỏ xót những lỗi không đáng tiếc. Trong tâm của bài viết này đi vào nhiên cứu các khái niệm cơ bản như lý thuyết tập hợp, các hàm, các mối quan hệ, logic mệnh đề, lý thuyết xác suất.
Phần lý thuyết tập hợp chúng tôi trình bày các khái niệm như định nghĩa tập hợp, tập rỗng, biểu diễn tập hợp bởi biểu đồ Venn. Các phép toán trên tập hợp, mối quan hệ giữa các tập hợp. Phân hoạch một tập hợp thành các tập con và các phép toán đồng nhất thức giữa các tập hợp.
Phần các hàm chung tôi trình bày các khái niệm liên quan đến miền và phạm vi. Các loại hàm và luật hợp thành hàm. Phần các quan hệ trình bày các quan hệ liên quan đến tập hợp và các quan hệ liên quan đến một tập đơn. Phần logic mệnh đề trình bày các khái niệm các toán tử logic, biểu thức logic và các quan hệ tương đương. Phần lý thuyết xác suất trình bầy một số khái niệm như định nghĩa và một số cách sử dụng lý thuyết xắc suất trong vấn đề kiểm tra bộ test case có đảm bảo hết được các chức năng cần thiết khi kiểm thử một phần mềm.
Nói chung trong mỗi phần chúng tối đều đưa nhưng mình họa ví dụ cụ thể việc ứng dụng cho nó trong kiểm chứng phần mềm như thế nào. Phần cuối chúng tôi trình bày vị thế của toán rời rạc trong các kỹ thuật kiểm thử phần mềm. Phần này chỉ muốn làm rõ thêm là toán rời rạc được ứng dụng rất nhiều trong các kỹ thuật kiểm thử chức năng. Chúng tôi hy vọng sẽ phần nào chuyển tải được những gì mình đã nghiên cứu và tìm hiểu đến bạn đọc.

Lời chú của các học viên Trần Văn Mạnh, Bùi Hoàng Khánh:
Những nội dung trình bày trong bài viết này phần lớn dựa trên chương 3, cuốn sách "Software testing - a Craftman's approach, second edition" của tiến sỹ Paul C.Jorgensen, Department of Computer science and Information Systems, Grand Valley State University
Khả năng của chúng tôi là có hạn và chưa đủ để nghiên cứu một đề tài lớn như trong sách này. Do vậy chúng tôi cố gắng đọc hiểu, tham khảo các tải liệu, và truyền tải lại nội dung dưới dạng một bài báo. Tuy nhiên bản quyền nội dung vẫn hoàn toàn là của tác giả Tiến sỹ Paul C.Jorgensen.

1 Giới thiệu chung

Từ trước đến nay khi nói đến IT người ta thường nghĩ ngay đến những chuyên gia về bảo mật dữ liệu, chuyên gia quản trị dự án và chuyên gia phần mềm. Ít ai nhắc đến chuyên gia đảm bảm bảo chất lượng phần mềm. Một phần cũng do người trong giới IT cũng như ngoại đạo chưa đánh giá được đúng bản chất của kiểm thử và đảm bảo chất lượng phần mềm. Phần nữa là dường như những năm trước đây ngay ở các trường đại học cũng chưa thật coi trọng vấn đề này khi đưa vào giáo trình giảng dạy ở các bậc đại học cũng như sau đại học. Tuy nhiên những năm gần đây, do tốc độ phát triển như vũ bão của ngành công nghệ thông tin. Các phần mềm

hiện nay lớn hơn rất nhiều về khối lượng cũng như độ phức tạp về mặt nghiệp vụ. Cho nên vấn đề đảm bảo phần mềm chạy đúng và không có lỗi cũng như chất lượng tốt đã ngày càng trở lên quan trọng. Và trong công nghệ phần mềm bộ phận quản lý chất lượng phần mềm trở lên hết sức quan trọng, một trong những đội ngũ của họ là những nhân viên kiểm thử. Họ là những người làm nhiệm vụ để phần mềm luôn luôn chạy đúng và tối ưu ở mức chấp nhận được. Vậy căn cứ vào đâu, tiêu chí nào và nền tảng vững chắc gì để có thể một trở thành một nhân viên kiểm thử chuyên nghiệp. Đó chính là họ cần có nền tảng toán học vững chắc, đặc biệt là toán rời rạc. Nó sẽ giúp ích rất nhiều cho công việc sau này của họ. Vậy sẽ có người tự hỏi, toán rời rạc giúp ích gì cho các nhiên viên kiểm thử. Để trả lời câu hỏi này chúng ta sẽ nghiên cứu từng bước một về các khái niệm như lý thuyết tập hợp, các hàm, các mối quan hệ, logic mệnh đề, lý thuyết xác suất, mỗi trong số đó sẽ được thảo luận ở đây.

2 Lý thuyết tập hợp

Thật là boăn khoăn khi chấp nhận rằng không tồn tại một định nghĩa cụ thể nào về tập hợp. Điều này thật sự là một phiền toái bởi vì lý thuyết tập hợp là trọng tâm của 2 chương toán học này. Theo đó, lý thuyết toán học đã đưa ra một sư phân loại quan trọng, lý thuyết tập hợp sơ khai và lý thuyết tập hợp tiên đề (axiomatic). Trong tập hợp sơ khai (naive), một tập hợp được coi như một số hạng nguyên thủy (Primitive term), nó giống như điểm và điểm dòng và dòng là các khái niệm nguyên thủy trong hình học. Có một số từ đồng nghĩa với tập hợp như: bộ sưu tập, nhóm, chum. Bạn đã hiểu ý ở đây. Một điều quan trọng về tập hợp là nó cho phép chúng ta quy một vài thứ thành một nhóm hay toàn bộ . Ví dụ, chúng ta muốn quy một tập hợp các tháng, mà có chính xác 30 ngày. (Chúng ta cần nhớ tập hợp này khi chúng ta kiểm thử chức năng NextDate trong chương 2). Trong lý thuyết tập hợp chúng ta viết:

$$\text{M1} = \{\text{April, June, September, November}\}$$

và chúng ta đọc ký hiệu này là "M1 là một tập hợp gồm các phần tử là các tháng April, June, September, November."

2.1 Các thành phần của tập hợp

Các đối tượng trong một tập hợp được gọi là các thành phần của một tập hợp, và mối quan hệ này được miêu tả bằng biểu tượng "thuộc", do đó chúng ta có thể viết April thuộc M1. Khi một đối tượng nào đó không là thành phần của tập hợp, chúng ta sử dụng biểu tượng "không thuộc", chúng ta có thể nói December không thuộc M1.

2.2 Định nghĩa tập hợp

Một tập hợp được định nghĩa theo ba cách: bằng cách liệt kê các phần tử của nó, bằng cách đưa ra một quy tắc quyết định hoặc bằng cách xây dựng tập hợp từ các tập hợp khác. Phương án liệt kê các phần tử được sử dụng trong trường hợp các tập hợp chỉ có ít phần tử cũng như trong các tập hợp mà các phần tử của nó tuân theo một kiểu mẫu rõ ràng. Chúng ta có thể xác định một tập hợp các năm trong chương trình Nextdate như sau:

$$\text{Y} = \{1812, 1813, 1814, \ldots, 2011, 2012\}$$

Khi định nghĩa một tập hợp bằng cách liệt kê các phần tử của nó, trật tự của các phần tử sẽ không quan trọng. Chúng ta hiểu lý do tại sao khi chúng ta thảo luận về đẳng thức tập hợp (set equality). Phương pháp quy tắc quyết định (Decision rule approach) thì phức tạp hơn, và sự phức tạp này mang lại cả lợi ích và bất lợi. Chúng ta có thể định nghĩa các năm cho NextDate như sau:

$$\text{Y} = \{\text{year} : 1812 \leq year \leq 2012\}$$

Chúng ta đọc "Y là tập hợp các năm sao cho các năm từ 1812 đến 2012". Khi một quy tắc quyết định được dùng để định nghĩa một tập hợp, quy tắc đó phải rõ ràng. Do đó, khi được đưa gia bất cứ giá trị hợp lý nào của năm, chúng ta có thể quyết định liệu năm đó có thể thuộc tập hợp Y hay không. Lợi ích của phương pháp định nghĩa tập hợp bằng các quy tắc quyết định là yêu cầu về sự rõ ràng của nó tạo ra tính rõ ràng. Các Testers có kinh nghiệm đã gặp phải rất nhiều yêu cầu không thể được kiểm thử. Trong nhiều lần, lý do về các yêu cầu không được kiểm thử dẫn đến một quy tắc quyết định rõ ràng. Trong chương trình hình tam giác của chúng ta là một ví dụ, giả sử chúng ta định nghĩa một tập hợp:

$$\text{N} = \{\text{t: là một tam giác có các cạnh gần bằng nhau}\}$$

Chúng ta có thể nói rằng, tam giác với các cạnh (500, 500, 501) là 1 phần tử của N , nhưng chúng ta sẽ làm thế nào với những tam giác có các cạnh là (50, 50, 51) hay (5, 5, 6)?

Điểm lợi thứ hai của phương pháp định nghĩa tập hợp bằng các quy tắc quyết đinh là chúng ta có thể gặp khó khăn, ví dụ chúng ta có thể quan tâm đến tập hợp:

$$\text{S} = \{\text{Bán hàng} : 15\% \text{ tỉ lệ hoa hồng áp dụng tới việc bán hàng}\}$$

Chúng ta không dễ dàng viết ra các phần tử của tập hợp này, nhưng được cho một giá trị đặc biệt cho bán hàng, chúng ta có thể dễ dàng áp dụng quy tắc quyết định. Bất lợi chính của các quy tắc quyết định là ở chỗ chúng có thể trở nên phức tạp một cách hợp lý, đặc biệt khi chúng ta được trình bày các biểu thức với các phép toán tồn tại và với mọi cho tất cả. Nếu mọi người hiểu các ký hiệu này, chính xác là hữu ích: rất thường xuyên, các khách hàng bị căng thẳng bởi các báo cáo với các lượng tử hóa. Một vấn đề thứ hai với nguyên tắc quyết định đã làm với chính tài liệu tham khảo. Điều này thật thú vị , nhưng nó thật sự ứng dụng rất ít cho testers. Vấn đề phát sinh khi một quy tắc quyết định liên quan đến bản thân, mà là một chu kỳ (circularity). Ví dụ Barber của Seville (trong truyện cổ tích hay ngụ ngôn gì đó) "là người đàn ông cạo tất cả những người không cạo mình."

2.3 Tập rỗng

Tập hợp rỗng, được ký hiệu là Ø, chiếm một vị trí đặc biệt trong lý thuyết tập hợp. Tập hợp rỗng không chứa phần tử nào. Về điểm này, các nhà toán học đã sẽ lạc đề khi chứng minh các cơ sở lập luận về các tập hợp rỗng:

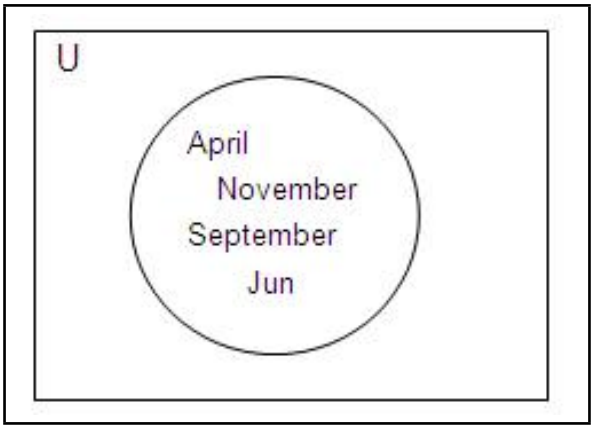
- Tập hợp rỗng là duy nhất, điều đó có nghĩa là, sẽ không có hai tập hợp rỗng.
- $\text{Ø}, \{\text{Ø}\}, \{\{\text{Ø}\}\}$, là những tập hợp khác nhau.

Rất hữu ích để chú ý rằng, khi một tập hợp được định nghĩa theo một nguyên tắc quyết định điều đó luôn luôn sai, thì tập hợp đó là tập hợp rỗng. Ví dụ:

$$\text{Ø} = \{\text{year} : 2012 \leq year \leq 1812\}$$

2.4 Các biểu đồ Venn

Tập hợp thường được mô tả bằng các biểu đồ Venn, như trong chương 1, khi chúng ta thảo luẩn về các tập hợp các hành vi cụ thể và các hành vi được lập trình. Trong một biểu đồ Venn, một tập hợp được mô tả là các điểm hình tròn, trong của một vòng tròn chứa các phần tử của một tập hợp. Sau đó, chúng ta có thể vẽ tập hợp M1 gồm các tháng có 30 ngày (hình 1).



Hình 1: Biểu đồ Venn của tập hợp gồm những tháng 30 ngày

Các biểu đồ Venn cho ta biết có nhiều mối quan hệ tập hợp khác nhau một cách trực quan, nhưng một số có vẻ cầu kỳ được đặt ra. Các tập hợp vô hạn khác hữu hạn (Infinite and finite sets) như thế nào? Cả hai loại này có thể được biểu diễn trong biểu đồ Venn, trong trường hợp các tập hợp hữu hạn, chúng ta giả định rằng mỗi điểm bên trong vòng tròn tương ứng với một phần tử. Chúng ta không cần phải lo lắng về điều này, nhưng sẽ thật hữu ích khi chúng ta biết được các giới hạn. Đôi khi, chúng ta có thể nhận thấy thật là tốt khi chúng ta gán nhãn các phần tử cụ thể.

Một điểm kết dính khác cần phải quan tâm là tập hợp rỗng. Chúng ta biểu diễn thế nào khi một tập hợp hay là một phần của tập hợp là rỗng? Câu trả lời chung chúng ta đánh bóng các khu vực trống, nhưng điều này lại mâu thuẫn với cách dung khác khi chúng ta dùng cách đánh bóng để biểu diễn khu vực quan tâm. Cách tốt nhất là cung cấp một ghi chú để phân loại các ý nghĩa của khu vực đánh bóng. Sẽ thật hữu ích khi chúng ta coi tất cả các loại tập hợp trong một thảo luận là các tập con của tập hợp lớn hơn, được gọi là tập vũ trụ (universe of discourse).

Chúng ta đã làm điều này trong chương 1 khi chúng ta chọn một tập hợp tất cả các hành vi chương trình được coi như universe of discourse. Universe of discourse thường được dự đoán từ các tập hợp đã cho. Ví dụ trong hình 1, hầu hết mọi người sẽ lấy tập vũ trụ là một tập hợp tất cả các tháng trong một năm. Những testers nên hiểu rằng cái tập vũ trụ mà được giả định thường là những nguyên nhân gây nhầm lẫn. Do vậy, chúng sẽ gây ra sự khó hiểu trong giao tiếp giữa khách hàng và các lập trình viên.

2.5 Các phép toán tập hợp

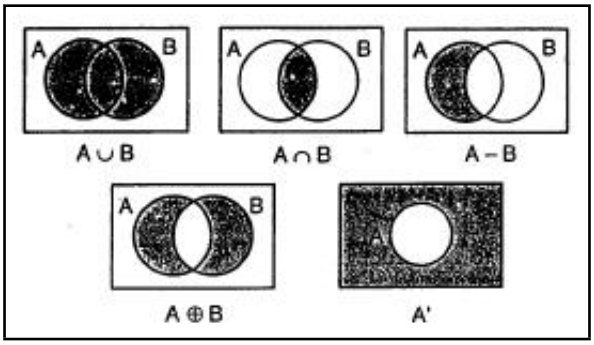
Phần lớn sức mạnh của lý thuyết tập hợp là dựa trên các phép toán về tập hợp: phép hợp, phép giao và phép lấy phần bù. Các phép toán khác thường được sử dụng là: phần bù tương đối (relative complement), hiệu đối xứng (sysmetric difference) và tích đề các. Mỗi một phép toán này sẽ được định nghĩa ở phần tiếp theo. Trong mỗi định nghĩa này, chúng tôi bắt đầu với các tập hợp A và B, thuộc tập vũ trụ U. Các định nghĩa này sử dụng các liên kết logic từ các phép toán mệnh đề như và $(\wedge)$,hoặc$(\vee)$,hoặc loại trừ$(\oplus)$,và phủ định$(\neg)$.

Định nghĩa 1. *Cho hai tập A và B, hợp của hai tập A và B là tập* $A \cup B = \{x : x \in A \vee x \in B\}$.
Giao của hai tập hợp là tập hợp $A \cap B = \{x : x \in A \wedge x \in B\}$.
Phần bù của tập A là tập $A' = \{x : x \notin A\}$.
Hiệu của hai tập A và B là tập $A - B = \{x : x \in A \wedge x \notin B\}$

Hiệu đối xứng của A và B là tập $A \oplus B = \{x : x \in A \oplus x \in B\}$.

Biểu đồ Venn biểu diễn các phép toán trên ở trong hình 2. Biểu đồ Venn rất hữu ích khi biểu diễn trực quan để mô tả mỗi quan hệ giữa các ca kiểm thử và giữa các đối tượng được kiểm thử. Nhìn trong biểu đồ Venn hình 2, chúng ta có thể biết rằng:

$$A \oplus B = (A \cup B) - (A \cap B)$$



Hình 2: Biểu đồ Venn biểu diễn các phép toán tập hợp

Đây là một trường hợp, mà chúng ta có thể dùng các logic mệnh đề để chứng minh. Biểu đồ Venn được sử dụng ở bất kỳ nơi nào trong việc phát triển phần mềm: cùng với đồ thị có hướng, chúng là cơ sở của các ký hiệu trong biểu đồ trạng thái (StateCharts), cái mà là những kỹ thuật chuyên ngành chặt nhất được hỗ trợ bởi kỹ thuật CASE, biểu đồ trạng thái cũng là các ký hiệu điều khiển được dùng cho UML, Ngôn ngữ mô hình hóa toàn cầu (The Universal Modeling Language) của công ty Rational Corp và nhóm quản lý các đối tượng (Object Management Group) Phép tích đề các của hai tâp tập hợp thì phức tạp hơn; nó phụ thuộc vào các hiệu của các cặp có thứ tự, là những tập hợp có hai phần tử mà trong đó các thứ tự của các phần tử này là rất quan trọng. Chúng ta ký hiệu các cặp có thứ tự và không có thứ tự như sau:

$$(a, b) = (b, a) \text{ nhưng } \langle a, b\rangle \neq \langle b, a\rangle$$

Sự phân biệt này là quan trọng khi tìm hiểu chương 4; như chúng ta đã biết, sự khác nhau căn bản giữa đồ thị thông thường và đồ thị có hướng chính xác là sự khác nhau giữa các cặp không có thự tự và có thứ tự.

Định nghĩa 2. *Tích đề các của hai tập hợp A và B là một tập hợp* $A \times B = \{\langle x, y : x \in A \wedge x \in B\}$.

Biểu đồ Venn không biểu diễn được phép tích đề các, do đó chúng ta sẽ xem một ví dụ đơn giản. Phép tích đề các của các tập hợp $A = \{1, 2\}, B = \{a, b\}$ là một tập hợp:

$$A \times B = \{\langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle, \langle 2, b\rangle\}$$

Phép tích đề các có một mối liên hệ trực giác với số học. Bản số của một tập hợp A là số các phần tử trong tập hợp A và được ký hiệu là $|A|$ (một số tác giả khác gọi lại Card(A)).Cho tập A và B, $|A \times B| = |A| \times |B|$.

Khi chúng ta nghiên cứu phần kiểm thử chức năng trong chương 5, chúng sẽ sử dụng tích đề các để mô tả các ca kiểm thử cho các chương trình với một vài biến đầu vào. Các thuộc tính bội của một tích đề các có nghĩa là chức năng kiểm thử tạo ra vô số các trường hợp kiểm thử khác.

2.6 Mối quan hệ giữa các tập hợp

Chúng ta sử dụng các phép toán tập hợp để xây dựng một số các tập hợp mới từ những tập hợp đã tồn tại. Khi chúng ta làm, chúng ta thường muốn tìm hiểu về một quan hệ giữa các tập hợp mới và các tập hợp cũ (ban đầu) như thế nào. Cho hai tập hợp A và B, chúng ta sẽ định nghĩa ba mối quan hệ tập hợp cơ bản:

Định nghĩa 3. *A là tập con của B, được viết là* $A \subseteq B$, *khi và chỉ khi* $a \in A \Rightarrow a \in B$ *A là tập con thực sự của B, viết là* $A \subset B$, *khi và chỉ khi* $A \subseteq B \wedge B - A \neq \emptyset$
A và B là các tập hợp bằng nhau, viết là $A = B$, *khi và chỉ khi* $A \subseteq B \wedge B \subseteq A$

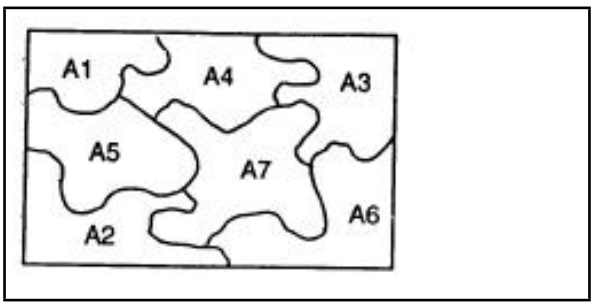
Trong tiếng anh hàng ngày, tập hợp A là một tập hợp con của tập hợp B nếu mỗi phần tử của tập hợp A cũng là phần tử của tập B. Để A trở thành một tập con thật sự của B, A phải là một tập con của B và phải có một số phần tử trong B nhưng không phải là phần tử của A. Cuối cùng thì các tập hợp A và B gọi là bằng nhau nếu mỗi tập hợp này là tập con của tập hợp khác.

2.7 Các cách phân hoạch tập hợp

Một cách phân hoạch tập hợp là một việc đặc biệt quan trọng cho các kiểm thử. Các phân chia sẽ có một vài tương tự như trong cuộc sống hàng ngày; chung ta có thể dụng sự phận chia để chia tách một vùng của văn phòng lớn thành các văn phòng riêng biệt; chúng ta cũng phải đối mặt với sự phân chia về chính trị khi một quốc gia được chia thành nhiều các đơn vị hành chính (như các thành phố, tỉnh,...). Trong cả hai ví dụ này, cần chú ý rằng tính chất của "sự phần chia" là để chia toàn bộ thành các phần nhỏ hơn, cái mà mọi thứ ở trong một vài phần, và không có cái nào bị loại bỏ ra ngoài. Cụ thể là:

Định nghĩa 4. *Cho một tập B và một tập các tập con A1, A2, ..., An = B, các tập con này là phân hoạch (cách phân chia) của tập B nếu và chỉ nếu* $A1 \cup A2 \cup \cdots \cup An = B$ *và* $i \neq j \Rightarrow Ai \cap Aj = \emptyset; i, j = \overline{1, n}$.

Bởi vì một phân hoạch là một tập của các tập con, chúng ta thường gọi là các các tập con riêng biệt như là các phần tử của sự phân hoạch. Hai phần của định nghĩa này là rất quan trọng đối với kiểm thử. Phần thứ nhất đảm bảo rằng mọi phần tử của B là trong một số tập con, trong khi phần thứ hai đảm bảo rằng không có phần tử nào của B thuộc hai tập con.



Hình 3: Biểu đồ Venn của một phân hoạch

Một trò chơi ghép hình là một vị dụ tốt về sự phân hoạch, trong thực tế, các biểu đồ Venn của các phân hoạch được được vẽ giống như các trò chơi ghép hình, như hình 3.

Sự phân hoạch rất hữu ích cho các testers bởi vì hai thuộc tính trên (định nghĩa) đảm bảo hai yếu tố: sự hoàn chỉnh và không dư thừa. Khi chúng ta nghiên cứu kiểm thử chức năng,

chúng ta sẽ nhận thấy rằng những điểm yếu rõ ràng của nó gây ra sự thiếu hụt và sự dư thừa: một số thứ thì không được kiểm thử, trong khi những thứ khác lại được kiểm thử đi kiểm thử lại. Một trong những khó khăn của kiểm thử chức năng tập trung vào việc tìm kiếm một phân hoạch phù hợp. Ví dụ như trong chương trình tam giác (Triangle Program), tập vũ trụ là tập hợp tất cả các bộ ba số nguyên dương. (Chú ý rằng đây thực sự là một tích đề các của tập các số nguyên dương được lặp lại ba lần). Chúng ta có thể phân hoạch tập vũ trụ theo hai cách:

1. Thành tam giác và phi tam giác
2. Thành tam giác đều, tam giác cân, tam giác không cân, tam giác vuông và phi tam giác

Mới đầu các phân hoạch này dường như là được, nhưng sẽ có một vấn đề với các phân hoạch cuối cùng. Các tập hợp của tam giác cân và tam giác vuông sẽ không bị tách rời (tam giác với các cạnh 3, 4, 5 là một tam giác vuông, đồng thời cũng là tam giác không cân.)

2.8 Đồng nhất thức tập hợp

Các phép toán và các mối liên hệ tập hợp, khi được kết hợp với nhau sẽ tạo ra một lớp quan trọng gồm các đồng nhất thức được sử dụng để đơn giản hóa bằng phương pháp đại số các biểu thức tập hợp phức tạp. Các sinh viên toán học thường phải bắt đầu từ tất cả điệu này, chúng ta sẽ chỉ liệt kê và thường xuyên sử dụng chúng như ở hình 4.

Tên	Biểu thức	
Quy luật đồng nhất	$A \cup \varnothing = A$	$A \cap U = A$
Quy luật thống trị	$A \cup U = U$	$A \cap \varnothing = \varnothing$
Quy luật bất biến	$A \cup A = A$	$A \cap A = A$
Quy luật bù	$(A')' = A$	
Quy luật giao hoán	$A \cup B = B \cup A$	$A \cap B = B \cap A$
Quy luật kết hợp	$A \cup (B \cup C) = (A \cup B) \cup C$	$A \cap (B \cap C) = (A \cap B) \cap C$
Quy luật phân bổ	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Quy luật DeMorgan	$(A \cup B)' = A' \cap B'$	$(A \cap B)' = A' \cup B'$

Hình 4: Các đồng nhất thức tập hợp

3 Các hàm

Các hàm là một khái niệm trọng tâm của phát triển phần mềm và kiểm thử. Ví dụ, mô hình phân tích hàm toàn bộ hoàn toàn sử dụng các khái niệm toán học của một hàm. Chúng ta sẽ làm rõ khái niệm này ở đây bởi vì tất cả kiểm thử hàm (kiểm thử chức năng) dựa trên khái niệm này.

Nói một cách dân dã, một hàm kết hợp các phần tử của các tập hợp. Ví dụ, trong các chương trình Nextdate. Hàm của một ngày đã cho là ngày của ngày tiếp theo, và trong vấn đề tam giác, hàm của ba số nguyên đã cho là loại tam giác được tạo bởi các cạnh với các độ dài đã cho. Trong vấn đề tiền hoa hồng, hoa hồng của người bán là một hàm của việc bán hàng, nó lần lượt là một hàm của số lượng khóa, lượng cổ phần và số thùng được bán. Các hàm trong hệ thống ATM phức tạp hơn nhiều, không ngạc nhiên rằng, điều này sẽ làm tăng sự phức tạp

cho quá trình kiểm thử. Bất cứ một chương trình nào có thể được coi là một hàm, các đầu vào là các miền (domain) và các sản phẩm là một loạt các hàm.

Định nghĩa 5. *Cho 2 tập hợp A và B , một hàm f là một tập con của* $A \times B$.
Cho $a_i, a_j \in A, b_i, b_j \in B$, *và* $f(a_i) = b_i, f(a_j) = b_j, b_i \neq b_j \Rightarrow a_i \neq a_j$.

Các định nghĩa chính thức như trên khá ngắn gọn, do vậy chúng ta hãy xem xét kỹ hơn. Các đầu vào đối với hàm f là các phần tử của tập hợp A, và các đầu ra của f là các phần tử của B. Những định nghĩa chỉ ra rằng hàm được xem xét kỹ lưỡng ở chỗ một phần tử trong tập hợp A chưa bao giờ kết hợp với hơn một phần tử của B. Nếu điều này có thể xảy ra, chúng ta đã từng kiểm thử một hàm như thế nào?. Đây là một ví dụ không tất định.

3.1 Miền và phạm vi

Trong khái niệm vừa ra, tập hợp A là một miền của hàm f, và tập hợp B là phạm vi. Bởi vì đầu vào và đầu ra có một thứ tự "tự nhiên", thật là đơn giản để nói rằng một hàm thực sự là một tập hợp gồm các cặp có thứ tự cái mà phần tử đầu tiên là từ miền và phần tử thứ hai là từ phạm vi. Đây là hai ký hiệu cho hàm: $f \subseteq A \times B$.
Chúng ta không đặt bất kỳ hạn chế nào trên các tập A và B trong định nghĩa này. Chúng ta có thể có A = B, vả cả A hoặc B có thể là một tích đề các của các tập hợp khác.

3.2 Các loại hàm

Các hàm sẽ được mô tả kỹ lưỡng hơn bởi phép ánh xạ. Trong định nghĩa dưới đây, chúng ta bắt đầu với một f : A $\rightarrow B$, và chúng ta định nghĩa một tập hợp:
f$(A) = \{b_i \in B : b_i = f(a_i)$ với một vài $a_i \in A$.

Tập hợp này đôi khi được gọi là hình ảnh của A dưới hàm f.

Định nghĩa 6. *f là một hàm từ A đến B khi và chỉ khi f*$(A) = B$.
f là một hàm từ A vào B khi và chỉ khi $f(A) \subset B$*(chú ý các tập con thích hợp ở đây).*
f là một hàm ánh xạ một một từ A đến B khi và chỉ khi, cho mọi:
$a_i, a_j \in A, a_i \neq a_j \Rightarrow f(a_i) \neq f(a_j)$.
f là một hàm ánh xạ từ nhiều đến một từ A đến B khi và chỉ khi, tồn tại $a_i, a_j \in A, a_i \neq a_j$ *mà* $f(a_i) = f(a_j)$.

Trở lại tiếng Anh thông dụng, nếu f là một hàm từ A đến (onto) B, chúng ta biết rằng mỗi phần tử của tập B sẽ được kết hợ với vài 1 số phần tử của A. Nếu hàm từ A vào (into) B, chúng ta biếts rằng sẽ có ít nhất một phần tử của B mà không được kết hợp với một phần tử của A. Các hàm ánh xạ một-một đàm bảo một dạng đơn nhất: Các phần tử domain riêng biệt sẽ chưa bao giờ được ánh xạ tới các phần tử trong cùng một dãy. Nếu một hàm không phải là đơn ánh (ánh xạ một một), nó sẽ là ánh xạ từ nhiều đến một (many-to-one), đó là nhiều hơn một phần tử domain có thể được ánh xạ đến phần tử trong cùng một dãy. Trong những thuật ngữ này, các yêu cầu về "đối xử tốt" sẽ ngăn cản các hàm trở thành ánh xạ một đến nhiều (one-to-many). Các tester quên với các dữ liệu có tính chất liên quan sẽ nhận ra rằng tất cả các khả năng (Một đến một, một đến nhiều, nhiều đến một, và nhiều đến nhiều) được cho phép cho các mối quan hệ. Trở lại với các ví dụ kiểm thử của chúng ta, giả sử chúng ta lấy A, B, C là các tập hợp các ngày trong chương trình NextDate, nơi mà :

A $= \{date : 1\ January\ 1812 \leq date \leq 31\ December\ 2012\}$
$B = \{date : 2\ January\ 1812 \leq date \leq 1\ January\ 2013\}$

$C = A \cup B$

Bây giờ, NextDate : $A \rightarrow B$ là một ánh xạ một-một, lên hàm(onto function), và NextDate: $A \rightarrow C$ là một ánh xạ một-một , vào trong hàm (into function). Ý nghĩa của NextDate không là nhiều tới một, nhưng rất dễ thấy các vấn đề ở hình tam giác có thể là nhiều tới một. Khi một hàm là một tới một và đến (onto), như NextDate $A \rightarrow B$ ở trước, mỗi phần tử của miền tương ứng với đúng một phần tử của của dãy (range); ngược lại, mỗi phần tử của dãy tương ứng với đúng một phần tử của miền. Khi điều này xảy ra. Nó luôn luôn có thể tìm thấy một hàm ngược (có thể nhìn thấy vấn đề YesterDate trong chương 2) đó là một tới một từ dãy trở về miền. Tất cả điều này là quan trọng để kiểm thử. Định nghĩa vào (into) khác biệt với đến (onto) có ý nghĩa cho tên miền và phạm vi kiểm tra dựa trên kiêm thử chức năng, và các hàm một tới một được yêu cầu kiểm thử nhiều hơn so với các hàm nhiều tới một.

3.3 Hợp thành hàm

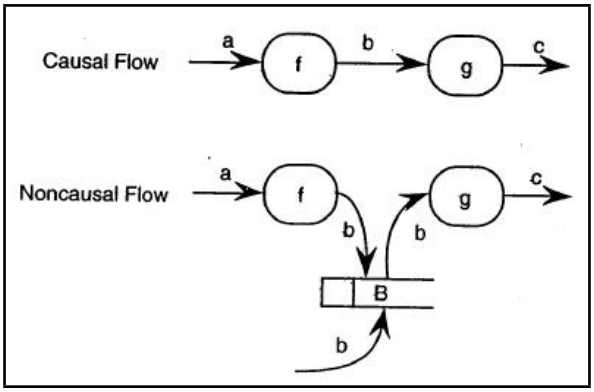
Giả sử chúng ta có các tập hợp và các hàm như vậy sau:
$f : A \rightarrow B, g : B \rightarrow C, h : C \rightarrow D$
Khi điều này xảy ra, chúng ta có tạo ra các hàm. Để làm điều này, chúng ta hãy liên hệ đến các phần tử của domain và phạm vi các tập hợp a $\in A, b \in B, c \in C, d \in D$, và giả sử rằng $f(a) = b, f(b) = c, f(c) = d$. Bây giờ sự hợp thành của các hàm g và f là:
$h \circ g \circ f(a) = h(g(f(a))) = h(g(b)) = h(c) = d.$

Hàm hợp thành là một ứng dụng rất phổ biến trong phát triển phần mềm; nó được thừa hưởng trong quá trình định nghĩa thủ tục (procedures) và các thủ tục con (subroutines). Chúng ta có một ví dụ về vấn đề này trong chương trình tiền hoa hồng, trong đó:
$f_1(locks, stocks, barels) = sales$
$f_2(sales) = commission$

Hàng loạt các hàm được tạo ra có thể gây khó khăn cho testers, đặc biệt khi phạm vi của một hàm là một tập con thích hợp của domain của hàm "tiếp theo" trong hàng loạt các hàm. Hình 5 biểu diễn nó có thể xảy ra như thế nào trong một chương trình được định nghĩa bởi biểu đồ luồng dữ liệu. Trong causal flow (luồng nhân quả), g.f(a) (cái mà chúng ta biết kết quả sẽ là g(b), g(b) lại tạo ra c) là một quá trình khá giống như assemblyline-like. Trong luồng không nhân quả (nocausal flow), khả năng nhiều hơn một tài nguyên của các giá trị b cho kho dữ liệu B sẽ gây ra hai vấn đề cho testers. Nhiều nguồn của các giá trị b có thể gây lên những vấn đề khả năng tương thích về domain/range(phạm vi); và thậm chí nếu điều này không phải là một vấn đề, có thể sẽ có bất thường thời gian đối với các giá trị của b.

Một trường hợp đặc biệt của hợp thành có thể được sử dụng, giúp kiểm tra một cách tò mò (curious). Nhớ lại chúng ta đã thảo luận làm thế nào đơn ánh (one-to-one) vào các hàm có một hàm ngược (hàm nghịch đảo). Nó chỉ ra rằng hàm ngược này là duy nhất và được đảm bảo tồn tại.

Nếu f là một hàm đơn ánh từ A vào B, chúng ta biểu thị hàm ngược bởi f^{-1}. Nó chỉ ra rằng $a \in A$ và $b \in B, f^{-1}.f(a) = a$ và $f^{-1} \ . \ f(b) = b$. Các chương trình NextDate và YesterDate được biết đến như là một ánh xạ ngược. Có một cách giúp kiểm thử là, đối với một hàm nhất định, ánh xạ ngược của nó như là một "kiểm tra chéo", và điều này thường có thể tiến hành việc xác định các trường hợp thử nghiệm chức năng.



Hình 5: Luồng nhân quả và không nhân quả trong biểu đồ luồng dữ liệu

4 Các quan hệ

đánh tiếp vào đây nhe

5 Logic mệnh đề

đánh tiếp vào đây nhe

6 Lý thuyết xác suất

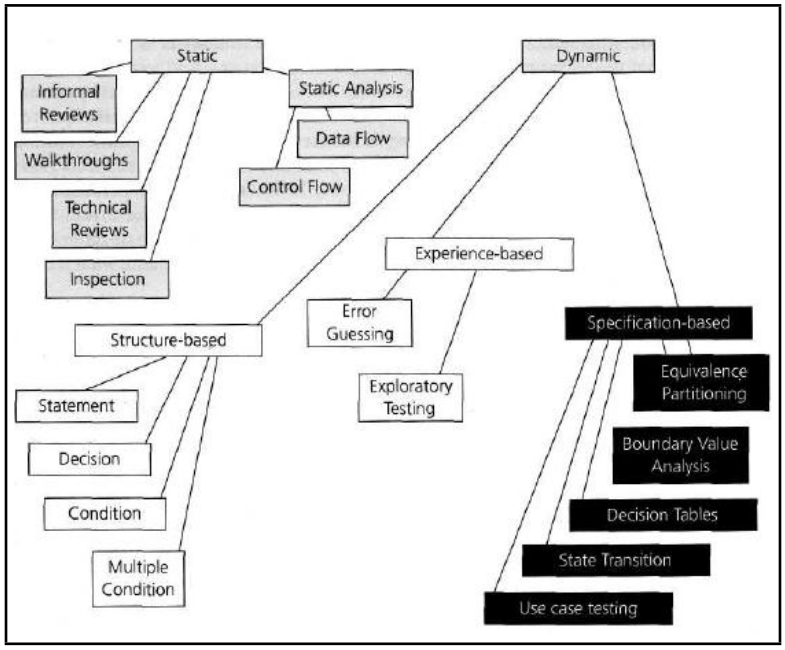
đánh tiếp vào đây nhe

7 Vị thế của toán rời rạc trong các kỹ thuật kiểm thử

Có nhiều loại kỹ thuật kiểm thử phần mềm khác nhau, mỗi một loại đều có điểm mạnh và điểm yếu của nó. Phần toán rời rạc như chúng ta đã trình bày ở trên thì chủ yếu ứng dụng nhiều cho phần kiểm thử chức năng (phần in đậm trong hình 6). Phần lý thuyết đồ thị ứng dụng nhiều trong phần kiểm thử cấu trúc. Dưới đây chúng ta sẽ tìm hiểu sơ bộ các loại đó và tìm hiểu xem vị thế của toán rời rạc sẽ nằm ở phần kiểm thử chức năng trong các kỹ thuật kiểm thử này.

7.1 Các kỹ thuật kiểm thử tĩnh (Static testing techniques)

Các kỹ thuật kiểm thử tĩnh không thực hiện kiểm tra bằng cách chạy mã (code) và thường được sử dụng trước khi kiểm thử được thực hiện trên phần mềm. Nó có thể được gọi là kỹ thuật không chạy code. Hầu hết các kỹ thuật kiểm thử tĩnh có thể được sử dụng để "kiểm thử" của bất kỳ hình thức của tài liệu bao gồm cả mã nguồn, tài liệu thiết kế và các mô hình, tài liệu kỹ thuật đặc tả chức năng và tài liệu đặc tả yêu cầu. Tuy nhiên "phân tích tĩnh" là một loại công cụ hỗ trợ của kiểm thử tĩnh mà tập trung vào kiểm thử các ngôn ngữ hình thức và do đó thường dùng để kiểm thử tĩnh mà nguồn.



Hình 6: Các kỹ thuật kiểm thử

7.2 Các kỹ thuật dựa trên tài liệu đặc tả kỹ thuật (black-box)

Việc đầu tiên của kỹ thuật kiểm thử động chúng ta sẽ xem xét là dựa trên tài liệu đặc tả kỹ thuật. Đây cũng được gọi là "black-box" hoặc kiểm tra đầu vào/đầu ra bởi vì chúng ta xem xét các phần mềm như là như là một hộp đen (black-box) với đầu vào và đầu ra, nhưng họ không có kiến thức về hệ thống hoặc các thành phần cấu trúc bên trong hộp (box). Về bản chất, kiểm thử đang tập trung vào phần mềm làm gì, và không tập trung vào nó làm thế nào. Chú ý rằng định nghĩa này đề cập đến cả hai kiểm thử chức năng và không chức năng. Kiểm thử chức năng là có liên quan tới hệ thống này làm những gì, nó gồm tính năng hoặc chức năng. Kiểm thử phi chức năng liên quan đến hệ thống làm điều gì đó tốt như thế nào, hơn là hệ thống làm gì. Các khía cạnh phi chức năng (còn gọi là đặc tính chất lượng hoặc các thuộc tính chất lượng) bao gồm hiệu năng, khả năng sử dụng, tính khả chuyển, bảo trì, v...v. Các kỹ thuật kiểm thử phi chức năng ít thủ tục và ít chính thức chính thức hơn so với các loại khác như các bài kiểm thử thực tế có nhiều sự phụ thuộc vào loại hệ thống, nó làm gì và có sẵn cho các bài kiểm thử nguồn tài nguyên. Kiểm thử phi chức năng là một phần của nghiên cứu và nó cũng được bao gồm trong [1]. Có những kỹ thuật để phát sinh các kiểm thử phi chức năng [Gilb, 1988], [Các tiêu chuẩn kiểm thử], nhưn họ cũng không bao quát hết được ở mức cơ sở. Phân loại các kiểm thử hộp đen và hộp trắng được đề cập trong một số sách về kiểm thử, bao gồm [Beizer, 1990] và [Copeland, 2003].

7.3 Các kỹ thuật dựa trên cấu trúc (white-box)

Các kỹ thuật kiểm thử dựa trên cấu truc (kỹ thuật này cũng năng động hơn là kỹ thuật tĩnh) sử dụng cấu trúc bên trong của phần mềm để lấy các ca thử nghiệm. Chúng thường được gọi là "hộp trắng" (ngụ ý bạn có thể nhìn thấy hệ thống) kể họ yêu cầu các hiểu biết làm thế nào phần mềm được thực hiện, đó là phần mềm làm việc như thế nào. Ví dụ, một cấu trúc kỹ thuật có thể được quan tâm thực hiện các vòng lặp trong phần mềm. Các trường hợp thử nghiệm khác nhau có thể được bắt nguồn để thực hiện vòng lặp một lần, hai lần, và rất nhiều

lần. Điều này có thể được thực hiện bất kể chức năng của phần mềm.

7.4 Các kỹ thuật dựa trên kinh nghiệm

Trong các kỹ thuật dựa trên kinh nghiệm, hiểu biết của mọi người, kỹ năng và kiến thức nền tảng là những đóng góp chính cho các điều kiện kiểm thử và các ca kiểm thử. Những kinh nghiệm của cả hiểu biết về kỹ thuật và nghiệp vụ là rất quan trọng, vì chúng mang lại những quan điểm khác nhau để phân tích kiểm thử và quá trình thiết kế. Do kinh nghiệm trước đó với hệ thống tương tự, họ có thể có cái nhìn vào những gì có thể không đúng, những gì là hữu ích cho kiểm thử.

Các kỹ thuật dựa trên tài liệu đặc tả thích hợp ở các cấp thử nghiệm (thành phần kiểm tra thông qua để người sử dụng kiểm thử) trong đó tài liệu đặc tả tồn tại. Khi thực hiện hệ thống hoặc kiểm thử chấp nhận (acceptance testing), các tài liệu đặc tả yêu cầu hoặc đặc tả chức năng có thể là cơ sở cho các kiểm thử. Khi thực hiện thành phần hoặc kiểm thử tích hợp, một tài liệu thiết kế hoặc tài liệu đặc tả ở thức thấp là cơ sở cho các kiểm thử. Kỹ thuật dựa trên cấu trúc cũng có thể được sử dụng ở mọi cấp độ thử nghiệm. Các lập trình viên sử dụng các kỹ thuật dựa trên cấu trúc trong thử nghiệm thành phần và thử nghiệm tích hợp các thành phần. Kỹ thuật dựa trên cấu trúc cũng được sử dụng trong hệ thống và kiểm thử chấp nhận.

Kỹ thuật dựa trên kinh nghiệm được sử dụng để bổ sung cho các kỹ thuật dựa trên tài liệu đặc tả và kỹ thuật dựa trên cấu trúc và cũng được sử dụng khi không có tài liệu đặc tả , hoặc các tài liệu đặc điểm nhưng không còn phù hợp hoặc đã lỗi thời. Điều này có thể là loại duy nhất của kỹ thuật được sử dụng cho các hệ thống rủi ro thấp, nhưng cách tiếp cận này có thể đặc biệt hữu ích dưới áp lực thời gian lớn - trong thực tế đây là một trong những yếu tố hàng đầu để kiểm thử thăm dò/khám phá.

Tài liệu

[1] Dorothy Graham, Erik van Veenendaal, Isabel Evans, and Rex Black, Foundations of software testing, ISTQB certification.

[2] Lewis, T. and Evangelist, Fat server vs fat clients: the transition from client-server to distributed computing, American programmer, Vol. 7 No. 11, November 1994, pp.2-9

[3] Milner, Robin, Element of interaction (1993 Turing Award Lecture) Communications of the ACM, Vol. 36, No. 1, January 1993.