Chung Qui Chỉ Tại Cantor

Ngô Quang Hưng
Computer Science and Engineering Department,
State University of New York at Buffalo,
Amherst, NY 14260, USA.
E-mail: hungngo@cse.buffalo.edu.

Ngày 24 tháng 3 năm 2005

Tóm tắt nội dung

Trong bài này, tôi sẽ duyệt qua các ý tưởng lớn và lịch sử xoay quanh một trong những bài toán quan trọng nhất của thế kỷ 21: bài toán "P chọi NP" (P versus NP). Câu trả lời đáng giá 1 triệu USD [20] và danh tiếng đi vào lịch sử khoa học. Cũng như bài toán Fermat lớn, bản thân câu trả lời cho bài "P chọi NP" không hẳn quan trọng bằng các kỹ thuật, hướng nghiên cứu, ngành nghiên cứu mới mà người ta khám phá ra để đi đến câu trả lời.

Quan trọng hơn hết, tôi hy vọng bài này đóng vai trò giới thiệu và khích lệ các bạn đến với một nhánh trung tâm của khoa học máy tính: lý thuyết tính toán và sự phức tạp (computational and complexity theory). Hành trình của ta sẽ ghé qua những vấn đề căn bản nhất mang đậm tính triết học của khoa học máy tính.

1 Giải thuật và độ phức tạp

In P or not in P, that's the question!

Đọc các quyển sách về phân tích và thiết kế giải thuật (như [10]), ta thấy rất nhiều các bài toán thú vị, cực kỳ hữu dụng và đẹp, mà lại có thể giải được trong thời gian đa thức. Ví dụ: xếp thứ tự (sort) n số cần thời gian $O(n \lg n)$, tìm đường đi ngắn nhất (shortest path) trong một đồ thị $G = (V, E)$ cần khoảng chừng $O(|V| \lg |V| + |E|)$, biến đổi Fourier nhanh (FFT) cần $O(n \lg n)$, vân vân.

Các vấn đề này đều là các vấn đề then chốt và tự nhiên của các ngành khoa học kỹ thuật như mạng máy tính (bài toán tìm đường), điện/điện tử và xử lý tín hiệu (FFT), cơ sở dữ liệu (xếp thứ tự các records), vân vân. (Các ví dụ nêu ra chỉ mang tính minh họa, các vấn đề tìm đường, xếp thứ tự, FFT, ... có cực kỳ nhiều ứng dụng khác!)

Có lẽ phải giải thích một chút tôi muốn nói gì bằng chữ "tự nhiên". Có các vấn đề không tự nhiên. Đó là các vấn đề mà người ta tạo ra chỉ để chứng minh một mệnh đề nào đó hoặc giải quyết một ứng dụng nho nhỏ nào đó, mà không có ứng dụng ở chỗ nào khác. *Khi một nhánh nghiên cứu còn non trẻ, sẽ có nhiều vấn đề không tự nhiên kiểu này*. Có rất nhiều các bài báo chuyên ngành giải quyết các bài toán đi vào quên lãng ngay sau khi bài báo được đăng. Khi nào có thời gian ta sẽ nói thêm về đề tài này, bản thân tôi nghĩ là nó rất quan trọng cho những ai bắt đầu làm nghiên cứu. Trong giới hạn của bài này, ta cứ nghĩ về "vấn đề tự nhiên" như một vấn đề có nhiều ứng dụng các nơi.

Đến đây nảy ra câu hỏi thú vị: "có thể nào tất cả các vấn đề tự nhiên đều giải được trong thời gian đa thức?" Sau hơn nửa thế kỷ của nghiên cứu giải thuật, về căn bản là ta vẫn chưa trả lời được câu hỏi này. Dường như có rất nhiều bài toán tự nhiên rất khó (theo nghĩa là không có giải thuật hiệu quả để giải). Có khoảng một chục nghìn bài toán tự nhiên kiểu này [11], thách đố hơn nửa thế kỷ nghiên cứu của các

kỹ sư và khoa học gia hàng đầu. Ngược lại, cũng không ai chứng minh được bất kỳ bài nào trong số đó là không có giải thuật hiệu quả để giải.

Ta hãy xem thử một vài bài toán kiểu này:

- VERTEX COVER: một vertex cover của một đồ thị $G = (V, E)$ là một tập S các đỉnh sao cho tất cả các cạnh của G đều kề với một đỉnh trong S. Ta muốn tìm vertex cover với kích thước nhỏ nhất.
- 01-KNAPSACK: một anh ăn trộm tìm được n vật quý, vật quý thứ i nặng w_i cân và đáng giá v_i đồng ($v_i, w_i \in \mathbb{Z}^+$). Anh ta lại chỉ có thể vác tối đa là W cân. Hỏi: anh ta phải lấy những vật nào để có được tổng giá trị lớn nhất?[1]
- EUCLIDEAN TRAVELING SALESMAN (Euclidean TSP): tìm đường ngắn nhất cho một anh bán hàng đi qua n thành phố, mỗi thành phố một lần và trở về thành phố đầu tiên.

Giả sử bạn đi làm cho một công ty nào đó, xếp giao nhiệm vụ viết một chương trình tốt, chạy nhanh, để giải một bài toán loại rất khó này, bạn sẽ làm gì? Ta thử ghi ra đây vài khả năng:

1. Hỏi giáo sư dạy giải thuật (và ông ta bí)
2. Bảo với xếp là "tôi chịu thôi" (và mất việc)
3. Ném vào sọt rác 6 tháng cuộc đời tìm giải thuật hiệu quả, và sau đó quay lại khả năng 2
4. Tìm giải thuật cơ bắp (brute-force) tốn khoảng vài chục thế kỷ chạy mới xong (mất việc và mất mặt)
5. Chứng minh cho xếp thấy là bài toán xếp cho không có giải thuật hiệu quả (cận dưới thời gian chạy là một hàm mũ chẳng hạn).
    - Khả năng này rất khó xảy ra, kinh nghiệm cho thấy chứng minh những điều tương tự là cực kỳ khó. Với tất cả các bài toán loại này, cận dưới tốt nhất người ta biết là $\Omega(n)$: vô dụng.
6. Chứng minh rằng bài toán xếp cho cũng khó như chục ngàn bài toán khác chưa ai biết giải.

A ha, khả năng số 6 nghe có vẻ được, nhưng cũng có vẻ lừa phỉnh vì ta thật ra không giải quyết thẳng vào vấn đề, mà bán cái nó cho nửa thế kỷ nghiên cứu của vô vàn người khác!

Nhưng làm thế nào ta chứng minh một điều như vậy? Thế nào gọi là khó? Thế nào là cái này khó tương đương cái kia? Về mặt trực quan thì có thể hiểu như sau: một bài toán khó là bài toán không giải được trong thời gian đa thức; hai bài toán khó tương đương nhau nếu giải bài này một cách hiệu quả thì cũng giải được bài kia một cách hiệu quả và ngược lại. Còn để trả lời nghiêm túc về mặt toán học, thì ta phải xem lại định nghĩa lại khái niệm giải thuật và các thứ liên quan.

Ta sẽ phải quay lại bánh xe lịch sử. Hành trình này đưa ta đến với Cantor, Russell, Hilbert, Gödel, Church, Turing, Cook/Levin, Karp và các ý tưởng tuyệt vời của họ.

2 Georg Cantor (1845–1918)

> Tôi thấy, nhưng tôi không tin.

[Trích một bức thư Cantor gửi Dedekind năm 1878]

[1] Ta có thể thiết kế một giải thuật qui hoạch động (dynamic programming) để giải bài này trong thời gian $O(nW)$, nhưng đó không phải là thời gian đa thức.

Georg Cantor (1845–1918)

David Hilbert (1862–1943)

Trở lại với hành trình lịch sử của ta: trong các thập niên cuối cùng cùng của thế kỷ 19, nhà toán học vĩ đại Georg Cantor (Georg Ferdinand Ludwig Philipp Cantor, 1845-1918) đề ra một trong những tư tưởng táo bạo nhất trong lịch sử toán học, khuấy đảo toàn bộ thế giới toán học lúc bấy giờ, làm cho các nhà toán học xuất sắc nhất thế giới thời kỳ đó như Henri Poincaré (1854–1912) và David Hilbert (1862–1943) phải nhảy vào "tham chiến". Cá nhân tôi có cảm tình đặc biệt với Cantor, và cho rằng ông là một trong vài nhà toán học có tư tưởng độc đáo nhất mọi thời đại. Chữ độc đáo không đủ để diễn tả điều tôi muốn nói về Cantor. Tiếng Anh họ dùng chữ original, mang nghĩa là nguyên gốc, độc đáo, đầu tiên, ...

Lý thuyết tập hợp của Cantor không chỉ làm lung lay nền tảng toán học, đặt ra các vấn đề mấu chốt của triết học và logic, là tiền thân của lý thuyết tập hợp hiện đại, mà còn mang một trong những tư tưởng tiên phong của lý thuyết tính toán hiện đại. Sinh viên học máy tính nên biết và đọc về lý thuyết Cantor.

Lý thuyết tập hợp Cantor có hai đối tượng chính: các tập hợp, và lực lượng (cardinality) của các tập hợp. Tập $\{a, b, c\}$ có lực lượng là 3. Ta dùng $|S|$ để chỉ lực lượng của tập S. Có tất cả $2^3 = 8$ tập con của tập $\{a, b, c\}$. Nói chung, nếu S là tập hữu hạn thì có tất cả $2^{|S|}$ tập con của tập S. Ta dùng 2^S để chỉ tập tất cả các tập con của S. Nếu S hữu hạn, thì rõ ràng $|2^S| = 2^{|S|} > |S|$.

Dĩ nhiên những điều này người ta đã biết từ lâu. Cantor bắt đầu lý thuyết của ông bằng một câu hỏi cực kỳ khó chịu: thế nếu S là tập vô hạn thì thế nào? Quan hệ $|S| < |2^S|$ có còn đúng nữa không? Thế nào là lực lượng của tập vô hạn?

Hai tập $\{a, b, c\}$ và $\{x, y, z\}$ có lực lượng bằng nhau. Điều này có thể hiểu theo hai cách: (1) đếm các phần tử trong từng tập, và cho kết quả là 3, thế là chúng bằng nhau. Nhưng đối với Cantor, lý do chúng bằng nhau sâu sắc hơn nhiều: (2) chúng có lực lượng bằng nhau là vì ta có thể ghép cặp một-một giữa chúng: $(a, y), (b, x), (c, z)$. Dĩ nhiên có nhiều song ánh (bijection - hoặc ánh xạ một-một) giữa hai tập này ($3! = 6$ song ánh), nhưng miễn có một song ánh là chúng bằng nhau. Theo dòng lý luận này, nếu có một đơn ánh (injection) từ tập A vào tập B thì $|A| \leq |B|$. Ví dụ: từ $\{a, b\}$ vào $\{x, y, z\}$ thì có đơn ánh $\{(a, z), (b, x)\}$, cho nên $|\{a, b\}| \leq |\{x, y, z\}|$. Nếu có đơn ánh từ A vào B, nhưng không có song ánh từ A vào B, thì $|A| < |B|$.

Để làm ví dụ, ta thử xét các tập hợp sau: $\mathbb{N}$ là tập các số tự nhiên, $\mathbb{E}$ là tập các số tự nhiên chẵn, và $\mathbb{R}$ là tập các số thực. Dễ thấy $|\mathbb{E}| \leq |\mathbb{N}| \leq |\mathbb{R}|$. Về mặt trực quan, ta có thể nghĩ rằng tập $\mathbb{E}$ bằng khoảng một nửa tập $\mathbb{N}$, nhưng thực ra $|\mathbb{N}| = |\mathbb{E}|$ vì ta có song ánh $f : \mathbb{N} \to \mathbb{E}$ định nghĩa bởi $f(x) = 2x$. Ví dụ

Alonzo Church (1903–1995)

Alan Turing (1912–1954)

này minh họa sự tinh tế khi ta xét các tập vô hạn. Với căn bản như trên, ta có thể chứng minh $|S| < |2^S|$ bằng một lý luận đơn giản nhưng có uy lực cực kỳ gọi là lập luận đường chéo (diagonal argument). Lý luận này cũng là nền tảng của các kết quả nổi tiếng của Gödel và Turing mà ta sẽ đề cập sau. Đại khái, để chứng minh $|S| < |2^S|$ (dù S là vô hạn) ta có thể làm như sau:

- Dễ thấy $|S| \leq |2^S|$ (bạn nên tự tìm vài đơn ánh chứng minh điều này).
- Giả sử $|S| = |2^S|$, nghĩa là có một song ánh $f : S \to 2^S$. Gọi T là tập tất cả các phần tử x thuộc S sao cho x không thuộc $f(x)$. Gọi b là phần tử của S mà $f(b) = T$. Bây giờ ta thử hỏi: b có nằm trong T hay không? Nếu b thuộc T thì, theo định nghĩa của T, b không thuộc $f(b) = T$, vô lý. Nếu b không thuộc T, thì cũng theo định nghĩa của T, b thuộc $f(b) = T$.
- Tóm lại, $|S| < |2^S|$.

Nhà toán học nổi tiếng Paul Erdős thường ví von rằng một chứng minh tuyệt đẹp như vậy phải là một chứng minh trong quyển sách của thượng đế (xem "Proofs from the book" [3]). Không biết các bạn thế nào chứ lần đầu tiên tôi thấy chứng minh này, tôi cảm thấy ná thở vì cái đẹp! Sau này tôi mới khám phá ra rằng lúc đó, dù hiểu về mặt kỹ thuật và cảm nhận được một ít cái đẹp của ý tưởng này, tôi thật sự chưa hiểu vấn đề! Ta sẽ còn gặp lại lập luận đường chéo vài lần trên hành trình xuôi dòng lịch sử này.

Bài tập 1. *Chứng minh rằng* $|\mathbb{N}| < |\mathbb{R}|$.

Ta đã chứng minh $|S| < |2^S|$ với mọi tập S bằng lập luận đường chéo. Cantor không dừng lại ở đó, ông xét một chuỗi vô hạn các lực lượng của các tập vô hạn được tạo ra theo cách này: $|\mathbb{N}|, |2^{\mathbb{N}}|, |2^{2^{\mathbb{N}}}|, \ldots$, và ông dùng $\aleph_0, \aleph_1, \aleph_2, \ldots$, (đọc là *Aleph* không, Aleph một, Aleph hai, vân vân) để chỉ lực lượng của các tập này. Như vậy, $\aleph_0$ là lực lượng của $\mathbb{N}$, câu hỏi tự nhiên là: thế lực lượng của $\mathbb{R}$ nằm đâu trong chuỗi này? Cantor tin rằng $|\mathbb{R}| = \aleph_1$ nhưng ông không chứng minh được điều này. Đây là một trong những bài toán nổi tiếng nhất của thế kỷ 20, thường được gọi là *giả thuyết công ti num*, hay *giả thuyết liên tục* (*continuum hypothesis*).

Giả thuyết liên tục, các va chạm với Leopod Kronecker (1823 – 1891) (người không chấp nhận lý thuyết tập hợp này), cùng với vài nghịch lý đau đầu của lý thuyết của mình (xem dưới đây) làm Cantor

bị suy nhược thần kinh liên tục trong hơn hai mươi năm cuối đời mình. Ông ra đi trong một viện an dưỡng sau một cơn đau tim.

Ta thử xem xét một nghịch lý nảy sinh từ lý thuyết Cantor, gọi là *nghịch lý Russell*. Bertrand Russell (1872-1970) xét tập hợp sau đây:

$$S = \{X | X \text{ không phải là phần tử của } X\}.$$

Ví dụ, nếu X là tập hợp các khái niệm hiểu được, thì X là phần tử của chính nó, vì rõ ràng khái niệm này có thể hiểu được. Nhưng nếu X là tập hợp các quán nhậu ở Sài Gòn, thì X không phải là phần tử của chính nó. Russell đặt câu hỏi: "thế S có phải là phần tử của chính nó không?" Dễ thấy đây là nghịch lý. Rất oái oăm là nghịch lý này có ý tưởng tương tự như lập luận đường chéo của Cantor. Trước đó, người ta cũng đã biết các nghịch lý khác có tư tưởng tương tự, nhưng vì không phát biểu bằng ngôn ngữ hình thức của toán học nên chúng bị bỏ qua, bị cho là sản phẩm của tính mù mờ của văn nói, văn viết. Ví dụ, nghịch lý kẻ nói dối có thể tả như sau: anh A nói "cái tôi đang nói là sai". Nghịch lý chàng cắt tóc thì nằm ở câu: "trong một làng nọ, có một chàng cắt tóc cắt tóc cho tất cả những ai trong làng không tự cắt tóc lấy" (câu hỏi là, thế anh ta có tự cắt không?). Vấn đề của các nghịch lý này là lối tham chiếu vào bản thân (self-reference) mà sau này Gödel cũng dùng để chứng minh định lý không toàn vẹn (incompleteness theorem) lừng danh của ông. Ta sẽ nói rõ hơn về định lý này sau.

Các nghịch lý kiểu như trên khuấy động một cuộc tranh luận lớn có một không hai trong lịch sử về nền tảng của toán học. Bạn thử tưởng tượng, nếu có một nghịch lý không giải quyết được trong bất kỳ ngành học nào, thì nền tảng logic của ngành đó có nguy cơ sụp đổ hoàn toàn! Các nhà toán học hoặc ủng hộ Cantor, hoặc chống đối hoàn toàn lý thuyết tập hợp này. David Hilbert nói: "không ai có thể đuổi chúng ta ra khỏi thiên đàng mà Cantor đã tạo cho chúng ta". Henri Poincaré bảo: "các thế hệ sau sẽ xem lý thuyết tập hợp như một thứ bệnh".

Các nhà toán học và logic học thời đó đề cử vài phương pháp để giải quyết các nghịch lý này:

- **Trường phái logic** (logicism) đề nghị dùng logic hình thức (formal logic) để làm toán, với hy vọng là logic hình thức sẽ xóa bỏ được sự mù mờ của ngôn ngữ phổ thông. Bộ sách Principia Mathematica của Alfred North Whitehead và Bertrand Russell [38] là một trong những công trình đại diện cho trường phái này. Các tác giả đã phải dùng đến hai quyển để có thể chứng minh $1 + 1 = 2$.

- **Trường phái trực quan** (intuitionism) được khởi xướng khoảng năm 1908 bởi nhà toán học Hà Lan Luitzen Egbertus Jan Brouwer (1881-1966). Trường phái này là cả một hệ thống triết học. Một trong những luận điểm cơ bản nhất của trường phái trực quan về toán học là: ta phải làm toán mang tính xây dựng (constructive mathematics). Các khái niệm như các số tự nhiên $1, 2, 3$ có thể "xây dựng" được từ trực quan con người. Khi định nghĩa một khái niệm mới, nó phải được xây dựng được bằng một số hữu hạn các bước. Như vậy, các chứng minh bằng phản chứng không thể chấp nhận được trong trường phái này, và vì thế các nghịch lý trên không mang ý nghĩa gì cả. (Xem thêm [12, 32, 33, 36].) Kể cũng thú vị là một định lý cực kỳ nổi tiếng của Brouwer trong hình học tô-pô (fixed point theorem) lại được chứng minh bằng phản chứng!

- **Trường phái hình thức** (formalism) do Hilbert khởi xướng. Về căn bản, Hilbert tin rằng các nhánh của toán học có thể được mô tả bằng một hệ thống tiên đề và một ngôn ngữ hình thức bậc nhất (first order language) với cú pháp cụ thể. Ngôn ngữ này có thể được nghiên cứu như một đối tượng toán học, và nhờ đó ta có thể trả lời chắc chắn các câu hỏi kiểu như: "có thể nào nhánh toán học này có nghịch lý không?" (Dĩ nhiên nghịch lý đó phải được phát biểu bằng ngôn ngữ hình thức ấy.) Hilbert đề nghị là hình thức hóa tất cả các nhánh của toán học, sau đó chứng minh rằng tất cả các nhánh này đều không có nghịch lý. Đây là nội dung chính của chương trình Hilbert (Hilbert's program). (Xem thêm [32].)

Hiện nay, chúng ta nói chung đều theo trường phái hình thức của Hilbert. Các ngành như hình học Euclid, lý thuyết số, đại số, vân vân đều được tiên đề hóa và hình thức hóa khá cụ thể. Các sách giáo khoa đều dạy học sinh theo kiểu này.

Ảnh hưởng của Hilbert vào sự phát triển của toán học hơn 100 năm qua là vô cùng to lớn. Khoa học máy tính cũng không ngoại lệ.

3 David Hilbert (1862–1943)

> "Nếu tôi có một số thành công nhất định trong toán học thì đó là vì tôi luôn thấy nó rất khó."

[Trích lời Hilbert nói với Harald Bohr (1887-1951)]

Năm 1900, tại đại hội các nhà toán học thế giới ở Paris (International Congress of Mathematicians), David Hilbert đề nghị 23 bài toán làm kim chỉ nam cho nghiên cứu toán học của thế kỷ 20. Các bài toán của Hilbert đã truyền cảm hứng cho biết bao thế hệ các nhà toán học, đã thiết lập các nhánh mới của toán học, thậm chí định hình cả cơ sở của lý thuyết tính toán trong khoa học máy tính hiện đại. Nếu các bạn muốn tìm hiểu thêm về các bài toán Hilbert và lịch sử xung quanh chúng thì tôi giới thiệu quyển "The Honors Class" của Benjamin Yandell [41].

Trong hành trình của chúng ta thì ba bài toán Hilbert sau đây đóng vai trò quan trọng:

- **Bài toán Hilbert số 1**: chính là continuum hypothesis như ta đã đề cập.
- **Bài toán Hilbert số 2**: có thể nào chứng minh rằng các tiên đề của số học (arithmetic) không dẫn đến các mệnh đề mâu thuẫn nhau?
- **Bài toán Hilbert số 10**: thiết kế một quá trình, một thủ tục mà nhờ đó sau một số hữu hạn các bước ta có thể biết một phương trình Diophantine (tìm nghiệm nguyên của các đa thức hệ số nguyên) có nghiệm hay không?

Một bài toán có liên quan nữa mà Hilbert đề nghị năm 1928 là bài toán quyết định (**Entscheidungsproblem** trong tiếng Đức, hay **decision problem** trong tiếng Anh). Một cách nôm na, bài toán này hỏi "có một thủ tục nào để quyết định xem một phát biểu toán học là đúng hay sai?" (dĩ nhiên là cho trước một tập các tiên đề và các luật suy luận). Gottfried Leibniz (1646-1716) đã hỏi câu hỏi này dưới một dạng hơi khác một chút. Leibniz sáng chế ra một máy tính cơ học, và ông mơ một ngày nào đó sẽ chế được một máy tính thao tác các biểu tượng toán học, và máy tính này sẽ có thể dùng để chứng minh hoặc phản chứng minh các phát biểu toán học.

Bài toán số 1 và số 2 liên quan đến nền tảng logic của lý thuyết tập hợp và của toán học nói chung. Lời giải cho các bài toán này có ý nghĩ to lớn về triết học, đặc biệt là bài số 2. Bài toán số 10 và entscheidungsproblem lại có ý nghĩa tuyệt đối quan trọng trong sự hình thành khái niệm giải thuật và lý thuyết tính toán. Các "thủ tục" tính toán mà Hilbert đề cập đều là các giải thuật, mặc dù lúc đó ông không dùng ngôn ngữ này. Để giải các bài toán này, ta cần định nghĩa rõ ràng thế nào là một giải thuật. Quá trình đi tìm định nghĩa giải thuật đã dẫn đến các mô hình máy tính hiện đại như máy Turing và phép tính lambda (lambda calculus) mà chúng ta sẽ thảo luận sau.

4 Kurt Gödel (1906–1978)

> "Tôi đã khám phá ra một khả năng logic và hợp pháp mà hợp chủng quốc Hoa Kỳ có thể biến thành một nước độc tài."

Kurt Gödel (1906 – 1978)

John von Neumann (1903 – 1957)

[Trích lời Gödel nói với Oskar Morgenstern năm 1952]

Đầu thế kỷ 20, Hilbert dành rất nhiều thời gian phát triển "chương trình Hilbert" như đã đề cập. Tại đại hội các nhà toán học thế giới năm 1928 ở Bologna, Hilbert tuyên bố là Wilhelm Ackermann (1896-1962) và John von Neumann (1903-1957) đã chứng minh được tính nhất quán (consistent – nghĩa là không dẫn đến các mệnh đề mâu thuẫn nhau) của lý thuyết số. Dĩ nhiên nếu tuyên bố này là đúng, thì chương trình Hilbert đã tiến được một bước dài. Hilbert còn tuyên bố là Ackermann gần kết thúc được cả chứng minh rằng số học của các số thực cũng nhất quán. Ông liệt kê bốn bài toán để hoàn tất chứng minh này của Ackermann. Đến năm 1931 thì cả bốn vấn đề này cùng với bài toán số 2 của Hilbert đều được giải quyết thỏa đáng. *Kurt Gödel phủ định tất cả* [15], ông phá tan tành chương trình Hilbert. Năm ấy Gödel mới có 25 tuổi.

Bài tập 2. *Chúng ta đều biết là các dữ liệu trong máy tính đều được lưu trữ theo dạng nhị phân. Mã ASCII gán* 8 *bits nhị phân (*0 *hoặc* 1*) cho mỗi chữ cái và chỉ cần cách này ta có thể mã hóa các văn tự tiếng Anh. Thế có thể nào mã hóa các văn tự tiếng Anh với chỉ một ký tự* 1 *thôi hay không? (Thay vì có cả* 0 *lẫn* 1*.)*

Gödel có ba định lý rất nổi tiếng: (1) định lý toàn vẹn (completeness theorem), (2) định lý không toàn vẹn thứ nhất (first incompleteness theorem), và (3) định lý không toàn vẹn thứ hai (second incompleness theorem).

Để giới thiệu về định lý toàn vẹn, hãy xét một bộ tiên đề của một trường (field) số, như trường số thực, trường số hữu tỉ, trường số phức, vân vân. Với một bộ tiên đề như vậy, có nhiều cấu trúc toán học thỏa mãn bộ này (các cấu trúc thực, phức, hữu tỉ, vectors, ...). Nếu có một mệnh đề nào đó có thể suy ra được từ hệ tiên đề này (ví dụ "nếu $a + a = a$, thì $a = 0$"), thì mệnh đề này đúng cho tất cả các cấu trúc toán trên hệ tiên đề này. Tính chất này gọi là tính hợp lý (soundness) của hệ thống. Ngược lại, nếu một mệnh đề nào đó đúng cho tất cả các cấu trúc toán học trên hệ tiên đề, thì có chắc rằng ta sẽ chứng minh được mệnh đề đó không? Câu trả lời là có, và đó chính là nội dung của định lý toàn vẹn Gödel. Theo một nghĩa nào đó, định lý này ủng hộ chương trình Hilbert.

Định lý không toàn vẹn thứ nhất đại khái nói rằng: nếu ta lấy hệ tiên đề cho số học các số tự nhiên (bao gồm cộng trừ nhân chia), thì sẽ có một phát biểu (bằng ngôn ngữ logic) không thể chứng minh

được là đúng hay sai. Điều cực kỳ thú vị là Gödel dùng "nghịch lý kẻ nói dối" để xây dựng một phát biểu trong hệ logic này. Phát biểu đó nói rằng: "mệnh đề này không chứng minh được", sau đó dùng lập luận đường chéo của Cantor để hoàn tất chứng minh.

Ta thử nghĩ thêm một chút về ý nghĩa to lớn của định lý này. Thứ nhất, nó hủy tan chương trình Hilbert. Không cần biết hệ tiên đề tốt đến đâu (kể cả khi có một số vô hạn – nhưng đếm được – các tiên đề), luôn luôn có các mệnh đề "độc lập" với hệ thống. Ta có thể lấy mệnh đề mới này làm một tiên đề và mở rộng hệ thống. Như vậy, toán học là tuyệt đối vô hạn! Bạn có thể tưởng tượng một cái "cây tri thức", bắt đầu bằng một số tiên đề và phân nhánh ra nhờ suy luận logic. Các tiên đề có thể là các nguyên tắc sống (không giết người, cướp của) chẳng hạn, và ta sống theo luật logic cùng các nguyên tắc của mình, ai cũng hy vọng là có thể làm thế được. Không may là, sẽ luôn có các tình huống trong cuộc sống nằm ngoài hệ thống giá trị của bản thân ta. Như vậy, định lý này của Gödel theo nghĩa nào đó đã dạy cho chúng ta phải sống với một tâm hồn mở, không thể theo một nguyên tắc bất di bất dịch nào đó. Ít nhất là nếu ta sống có logic.

Định lý không toàn vẹn thứ hai của Gödel phá hủy hoàn toàn những gì còn lại của chương trình Hilbert. Định lý này nói rằng không thể chứng minh rằng số học có tính nhất quán. Chỉ số học đã "phức tạp" đến thế, ta không có hy vọng gì vào chương trình chứng minh tính nhất quán của toàn bộ toán học.

Sự độc lập (independence) của một mệnh đề trong hệ thống (như mệnh đề Gödel dựng nên để chứng minh định lý không toàn vẹn thứ nhất) là một ý tưởng rất quan trọng trong lịch sử toán học. Tiên đề số 5 của Euclid là một ví dụ kinh điển. Bỏ nó vào thì ta có hình học Euclid, thay đổi nó một chút theo vài kiểu khác nhau là ta có vài loại hình học phi Euclid với các công trình của Carl Friedrich Gauss, Nikolai Ivanovich Lobachevsky, János Bolyai, và Bernhard Riemann. Giả thuyết liên tục (bài toán số 1 của Hilbert) được Paul Cohen chứng minh tính độc lập năm 1963. Câu hỏi "$\mathbf{P}$ chọi $\mathbf{NP}$" của chúng ta cũng có khả năng là nó độc lập với hệ thống suy luận hiện hành, ta sẽ quay lại khả năng này sau.

Các nghiên cứu về các hệ thống chứnh minh và các tiên đề có giá trị to lớn trong khoa học máy tính. Nhiều nhà khoa học máy tính nổi tiếng (như Edsger Wybe Dijkstra) đã bỏ nhiều năm nghiên cứu để làm thế nào "chứng minh" được là một chương trình máy tính chạy đúng với dự định thiết kế (specification), mà theo một nghĩa nhất định trong ngữ cảnh của chúng ta là các tiên đề! Một công trình khác của Gödel về "độ dài" của các chứng minh [17] đã nuôi mầm cho các định lý "tăng tốc" (speed-up theorems) của lý thuyết tính toán hiện đại.

John von Neumann từng nói là Kurt Gödel là nhà logic học lỗi lạc nhất thế giới sau Aristotle. Ta không thể không đồng ý với Neumann. (Bản thân Neumann cũng là một thiên tài tuyệt đỉnh có ảnh hưởng cực lớn đến khoa học máy tính mà ta sẽ "gặp" ông vào dịp khác.)

5 Alonzo Church (1903–1995)

> "Không. Hồi đó khoa Triết Học không có ai quan tâm đến mấy thứ ấy."
>
> [Alonzo Church trả lời khi được hỏi có quan hệ gì giữa khoa Toán và khoa Triết ở Princeton thời cuối 1920, đầu 1930.]

Bản chất của bài toán Hilbert số 10 nằm ở các câu hỏi: "các hàm nào có thể tính toán được (computable)?", và "thế nào là một giải thuật tính một hàm, hay nói cách khác: tính toán là gì?" Từ đầu thế kỷ 20, các nhà logic học đã nhận ra rằng một số rất lớn các hàm toán học đều có thể được định nghĩa bằng phương pháp đệ qui (và vài luật đơn giản khác). Tuy vậy, có một số hàm không định nghĩa được theo cách đệ qui sơ đẳng này (primitive recursive), ví dụ như hàm Ackermann vì nó tăng siêu nhanh. Thế là các nhà logic học tập trung đi tìm các hệ thống tạo hàm mạnh hơn, với mục tiêu tối hậu là hệ thống ấy có thể tạo được tất cả các hàm tính toán được.

Alonzo Church (1903-1995) sáng tạo ra phép tính lambda (lambda calculus, hoặc λ-calculus) cho mục tiêu này [5, 6]. Church và Gödel thảo luận về phép tính lambda ở đại học tổng hợp Princeton. Lúc

đầu Gödel không thích món phép tính này lắm vì Gödel nghĩ hệ thống của Church thiếu động cơ triết học. Church thách thức Gödel sáng tạo ra hệ thống tạo hàm khác mạnh hơn giải tích lambda. Vài tháng sau Gödel nghĩ ra cách cải tiến một ý tưởng của Jacques Herbrand (1908-1931) [19] để tạo nên các hàm đệ qui Herbrand-Gödel [16], hay gọi ngắn gọi là các hàm đệ qui (recursive functions). Hóa ra là hai hệ thống này tương đương nhau, nhờ vào các nghiên cứu của Stephen C. Kleene (1909-1994) [22], một học trò của Church. Với kết quả này, Church tin rằng các hàm tính toán được đều là các hàm đệ qui. Niềm tin này thường được gọi là **luận đề Church** (Church thesis).

Một giải thuật thật ra cũng là một hàm số (hay ánh xạ), ví dụ như giải thuật xếp thứ tự chuyển một dãy số thành dãy có thứ tự. Khi nói đến "tính toán được", ta thực sự muốn nói là "có giải thuật để tính". Vì thế, luận đề Church cũng có thể được phát biểu là: "cái gì tính được bằng một giải thuật thì tính được bằng phép tính lambda". Chú ý rằng "luận đề" có ý nghĩa như một định nghĩa, một tiên đề, không cần phải chứng minh. Vấn đề là ta có tin rằng phép tính lambda nắm bắt được mấu chốt của khái niệm "giải thuật" hay không mà thôi. Năm 1936 [6], Church dùng luận đề này để giải quyết entscheidungsproblem của Hilbert. Ông thiết kế hai biểu thức trong phép tính lambda mà sự tương đương của chúng không thể tính được bằng một hàm đệ qui. Nói cách khác, có các phát biểu toán học không thể chứng minh được một cách cơ bắp (bằng giải thuật).

Trong khoa học máy tính thi phép tính lambda và các hàm đệ qui có ảnh hưởng sâu sắc đến ngành trí tuệ nhân tạo. Các ngôn ngữ hàm (functional languages) thuộc họ Lisp (như Scheme và Common Lisp) đều dựa trên phép tính lambda và định nghĩa đệ qui.

6 Alan Turing (1912–1954)

> "Tầm mắt của chúng ta chẳng xa gì lắm, mà ta đã thấy bao nhiêu việc để làm."

[Alan Turing - 1950]

Việc ai đó có tin là Church đã giải được entscheidungsproblem hay không phụ thuộc vào việc người đó có tin rằng giải tích lambda thật sự bao gồm những gì tính toán được hay không. Hồi đó Gödel và nhiều người khác chưa thật sự tin là các hàm tính được đều tính được bằng giải tích lambda. Turing thay đổi niềm tin này. Khoảng năm 1935, 1936, Turing cũng tìm được một phương pháp để trả lời entscheidungsproblem [34], không biết rằng Church cũng đang nghiên cứu vấn đề này gần xong. Cuối cùng Church làm xong trước Turing khoảng 6 tháng – có một transcript một cuộc phỏng vấn Church ở đại học UCLA có đề cập đến điều này [7]. Tuy vậy, phương pháp của Church và Turing khác nhau hoàn toàn và chúng có giá trị độc lập.

Phương pháp của Turing có thể được tóm tắt như sau. Entscheidungsproblem đại khái nói: "tìm một giải thuật để quyết định xem một phát biểu logic có thể chứng minh được hay không".

Trước hết, Turing sáng tạo ra một mô hình "máy" để nắm bắt khái niệm giải thuật. Cái máy của ông — mà ta sẽ gọi là **máy Turing** (Turing Machine) từ giờ trở đi — được thiết kế để bắt chước quá trình tính toán một cách cơ học của con người (ví dụ như trẻ nhỏ có thể cộng và nhân hai số trên giấy mà không hiểu tại sao làm thế thì đúng).

Sau đó, ông thuyết phục chúng ta là máy Turing có thể tính tất cả các hàm tính toán được bằng cách thiết kế cái máy của ông cho nó tính rất nhiều hàm khác nhau, rồi chứng minh rằng tất cả các hàm tính được bằng giải tích lambda hay các hàm đệ qui Herbrand-Gödel đều có thể tính được bằng máy Turing. Dịp khác tôi sẽ viết chi tiết về máy Turing. Để cảm nhận được máy Turing mạnh như thế nào, ta nên biết rằng tất cả các chương trình máy tính hiện đại đều tương đương với một máy Turing nào đó. Mỗi máy Turing tương đương với một chương trình viết bằng ngôn ngữ lập trình yêu thích nhất của bạn. Một chương trình chơi cờ vua viết bằng Java là một máy Turing, chương trình Kazaa để download nhạc/phim là một máy Turing, chương trình mô phỏng big-bang là một máy Turing.

Cuối cùng, ông thiết kế một bài toán mà không máy Turing nào có thể tính được, gọi là **bài toán dừng** (halting problem). Có lẽ đến đây bạn cũng có thể đoán được là để chứng minh bài toán dừng không quyết định được bằng máy Turing thì ông đã dùng lập luận đường chéo của Cantor!

Turing cũng có luận đề tương tự như luận đề của Church mà Kleene gọi chung là **luận đề Church-Turing**. Trong ngôn ngữ hiện đại luận đề Church-Turing có thể phát biểu như sau:

Luận đề Church-Turing

> "máy Turing nắm bắt chính xác khái niệm giải thuật, hàm nào tính được bằng một giải thuật thì tính được bằng máy Turing và ngược lại".

Máy Turing hiện nay là công cụ phổ biến nhất để nghiên cứu độ phức tạp tính toán. Người ta đã sáng chế ra nhiều mô hình tính toán khác như máy truy cập ngẫu nhiên (Random Access Machine – RAM) của von Neumann, hệ thống Post, giải thuật Markov, logic tổ hợp (combinatory logic), phép tính lambda, các hàm đệ qui Herbrand-Gödel, máy thanh ghi vô hạn (unlimited register machine), máy tính xử lý song song (parallel computers), máy tính lượng tử (quantum computer), vân vân, nhưng không có mô hình nào mạnh hơn máy Turing về khả năng tính toán. Chú ý rằng ta vẫn chưa định nghĩa thật rõ ràng thế nào là "khả năng tính toán", "hàm tính được". Bạn có thể hiểu nôm na sự tính toán là một thủ tục rõ ràng, không có chỗ nào mơ hồ, có thể về nguyên tắc làm được bởi một người bình thường nếu có đủ thời gian và giấy nháp.

Đóng góp của Turing vào khoa học máy tính không dừng ở đó. Năm 1950, bài báo "máy tính và sự thông minh" (Computing Machinary and Intelligence, [35]) của ông là một trong những bài báo đặt nền móng cho ngành trí tuệ nhân tạo. Phép thử Turing (Turing test) vẫn là một thách thức khổng lồ cho cả ngành khoa học máy tính.

Tầm nhìn của Turing, không xa lắm đối với ông, nhưng đã vượt qua cả thời đại chúng ta.

7 Jack Edmonds, Hartmanis và Stearns, Manuel Blum

> "Các bài toán có và không có các giải thuật tốt để giải đều rất đáng quan tâm ... Tôi giả định (conjecture) rằng không có giải thuật tốt nào để giải bài toán người bán hàng (traveling salesman). Các lý do của tôi cũng giống như lý do cho các giả định toán học khác: (1) chuyện đó có khả năng xảy ra, và (2) tôi không biết."

[Jack Edmonds, 1966]

Ta đã nói đến bài toán dừng của Turing và bài toán trong giải tích lambda của Church, các bài toán không có giải thuật nào giải được. Đây là một trong những kết quả cực kỳ quan trọng của khoa học máy tính lý thuyết, nhất là về mặt triết học. Khả năng của máy tính đúng là có giới hạn! Khẳng định này dường như có vẻ tầm thường, ai chẳng nghĩ là khả năng của một cái máy chỉ có một giới hạn nhất định nào đó.

Không hẳn như thế. Thứ nhất, bài toán dừng là một bài toán rất cụ thể, vạch một ranh giới cụ thể cho giới hạn của máy tính. Kỹ thuật chứng minh này sẽ được dùng đi dùng lại cho các bài toán không quyết định được (undecidable problems) khác. Thứ hai, khi biết cái gì làm cho máy Turing có giới hạn, việc tìm một mô hình mạnh hơn mô hình máy Turing sẽ có đường đi rõ ràng hơn. Thứ ba, kể cả vũ trụ cũng có giới hạn, cho nên một giới hạn về mặt lý thuyết không hẳn là liên quan gì đến giới hạn thực tế. Bằng chứng là sự bùng nổ của khoa học máy tính và các công nghệ đi kèm trong vài chục năm qua và vài trăm, có lẽ vài ngàn năm tới. Điều này cho thấy ta cần phải nghiên cứu kỹ hơn bên trong cái giới hạn lý thuyết đó: nghiên cứu các bài toán có giải thuật để giải, và phân lớp chúng ra, xem giới hạn thực tế của máy tính là đến đâu.

Đầu những năm 60, các kỹ sư và khoa học gia bắt đầu chú ý rằng có các bài toán mà giải thuật cho chúng đòi hỏi rất nhiều thời gian chạy. Thời đó, thời gian chạy máy khá đắt tiền, tài nguyên tính toán thì ít ỏi. Người ta mới nghĩ đến việc nghiên cứu giải thuật bằng tổng số bước nó cần để giải một bài toán nhất định (Micheal O. Rabin [27], Juris Hartmanis và Richard E. Stearns [18], Manuel Blum [4]). Ở Nga, năm 1959 Yablonski cũng đã có hơi hướng đề cập đến độ phức tạp giải thuật [39, 40] Trước đó, Kurt Gödel đã chú ý đến vấn đề này trong một lá thư gửi Jon von Neumann khi Neumann đang dưỡng bệnh ung thư (xem thêm bài của M. Sipser [31] có bản sao của lá thư này).

Ngành nghiên cứu giải thuật và độ phức tạp của chúng thật sự bắt đầu với Jack Edmonds và bài báo năm 1965 của ông với tựa đề "đường đi, cây, và hoa" [13]. Đóng góp chính của bài báo là một giải thuật thời gian đa thức cho bài toán maximum matching. Bài báo đó của Edmonds, và hai bài khác của Alan Cobham [8] và von Neumann [37] bắt đầu định nghĩa **lớp P** là lớp các bài toán có thể giải được trong thời gian đa thức.

Bài toán nào nằm trong $\mathbf{P}$ được xem là bài toán "dễ", còn lại là các bài toán khó. Lớp $\mathbf{P}$ chỉ có thể được định nghĩa chính xác bằng toán học nếu ta có một mô hình toán học cho khái niệm giải thuật. Dĩ nhiên mô hình này không gì khác hơn là máy Turing.

Máy Turing còn có một dạng bất định, gọi là **máy Turing bất định** (non-deterministic Turing machine, ký hiệu là NTM). Máy Turing bình thường còn được gọi là **máy Turing xác định** (deterministic Turing machine, hay DTM). Máy NTM cũng tương tự như DTM, chỉ khác ở chỗ máy NTM có thể chọn bước thực thi kế tiếp tùy hỉ trong một tập cho trước các lệnh. Ta sẽ định nghĩa rõ ràng hơn trong phần tới.

Về mặt trực quan thì rõ ràng NTM có vẻ mạnh hơn DTM. Tuy vậy, chứng minh điều này không dễ dàng chút nào. Lớp các bài toán giải được bằng NTM trong thời gian đa thức được gọi là lớp $\mathbf{NP}$. Câu hỏi rằng liệu NTM có thật sự mạnh hơn DTM hay không chính là câu hỏi trung tâm của khoa học máy tính hiện nay. Nói cách khác: "$\mathbf{P}$ có bằng $\mathbf{NP}$ không?"

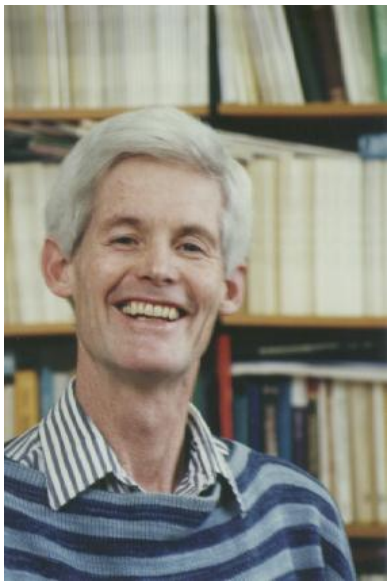
Trong cùng tinh thần của các bài toán Hilbert, viện toán Clay (Clay Mathematics Institute) đã treo giải thưởng một triệu đô la cho ai giải được một trong 7 bài toán chưa có lời giải của thế kỷ 20, một trong 7 bài toán này chính là câu hỏi $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$. (Một bài khác chính là bài toán số 8 của Hilbert: giả thuyết Riemann).

8 Stephen Cook, Leonid Levin, và Richard Karp

Năm 1971, Stephen Cook [9] chứng minh rằng có các vấn đề trong lớp $\mathbf{NP}$ mà tất cả các vấn đề khác trong $\mathbf{NP}$ có thể *chuyển giản* (reduced) về được trong thời gian đa thức. Phép chuyển giản này có thể được mô tả đại khái như sau: nếu vấn đề A có thể chuyển giản được về vấn đề B trong thời gian đa thức, thì nếu ta giải được B trong thời gian đa thức ta sẽ giải được A trong thời gian đa thức. Theo nghĩa thời gian đa thức thì vấn đề B khó hơn hoặc bằng A. Vì thế, các vấn đề mà Cook đề cập là các vấn đề khó nhất trong lớp $\mathbf{NP}$. Các vấn đề này gọi là các vấn đề $\mathbf{NP}$-*đầy đủ*. Cook chứng minh rằng SAT, 3-SAT, và SUBGRAPH ISOMORPHISM là $\mathbf{NP}$-đầy đủ. Độc lập với Cook, Leonid Levin cũng có một bài báo năm 1973 [25] với kết quả tương tự.

Bài tập 3. *Cho một đồ thị vô hướng $G = (V, E)$. Một* phủ đỉnh *(vertex cover) của G là một tập con S các đỉnh sao cho các cạnh của G đều có một đầu trong S. Một* tập độc lập *(independent set) của G là một tập con I các đỉnh sao cho không có cạnh nào nối hai đỉnh trong I.*

1. *Chứng minh rằng S là một phủ đỉnh nếu và chỉ nếu $V - S$ là một tập độc lập.*
2. *Chứng minh rằng ta có thể tìm một tập độc lập với lực lượng lớn nhất trong thời gian đa thức nếu và chỉ nếu ta tìm được một phủ đỉnh nhỏ nhất trong thời gian đa thức. Như vậy, bài toán phủ đỉnh có thể được chuyển giản về bài toán tập độc lập và ngược lại.*



Stephen Cook

Leonid Levin

Năm 1972, Richard Karp [21] chứng minh rằng 20 vấn đề tự nhiên khác cũng **NP**-đầy đủ, như VERTEX COVER, EUCLIDEAN TSP, CLIQUE, INDEPENDENT SET, ... Năm 1972, Donald Knuth [23, 24] góp phần định ra các thuật ngữ của ngành này như ta thấy ngày nay, vì lúc đó ông đang viết phần 4 của bộ sách kinh điển "nghệ thuật lập trình máy tính". Quyển sách của Garey và Johnson [14] có thảo luận sơ qua về quá trình thay đổi thuật ngữ trong thời gian đó.

Hầu hết những nhà khoa học ta vừa nhắc đến đều được giải Turing (giải thưởng tương đương với giải Nobel cho khoa học máy tính): Rabin, Hartmanis và Stearns, Blum, Cook, Karp, và Knuth.

9 Tình trạng hiện tại của câu hỏi P chọi NP

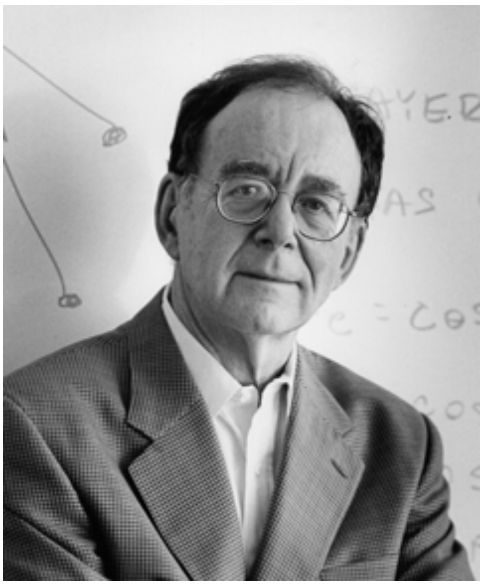
Có bốn khả năng trả lời câu hỏi **P** chọi **NP**: (a) $\mathbf{P} = \mathbf{NP}$, (b) **P** chọi **NP** không quyết định được, nghĩa là độc lập với hệ thống, (c) $\mathbf{P} = \mathbf{NP}$ hoặc **P** khác **NP** tùy theo ta chọn hệ tiên đề nào (tương tự như continuum hypothesis), và (d) $\mathbf{P} \neq \mathbf{NP}$. (Bài báo của Micheal Sipser [31] là tham khảo tốt cho tình trạng hiện tại của câu hỏi này.)

Nếu (a) đúng thì cả chục nghìn vấn đề **NP**-đầy đủ có giải thuật đa thức để giải, mật mã khóa công cộng (public key cryptography) là vô vọng, vân vân. Các giao thức (protocol) bảo mật trên Internet hiện nay đều ít nhiều dựa vào mật mã khóa công cộng, nếu $\mathbf{P} = \mathbf{NP}$ thì các giao dịch tiền tệ trên Internet đều cực kỳ bất ổn. Nếu $\mathbf{P} = \mathbf{NP}$ thì hàng nghìn hàng vạn các nhà khoa học, toán học, kỹ sư máy tính trong hơn năm chục năm qua đã bỏ qua một ý tưởng căn bản nào đó có thể giải một trong số cả chục nghìn bài toán **NP**-đầy đủ. Chuyện này quá hy hữu. Vì thế, khả năng (a) rất khó xảy ra.

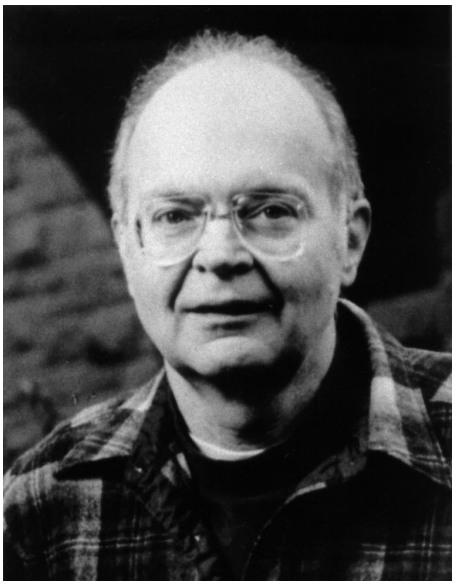
Các khả năng (b) và (c) liên quan đến trạng thái logic của vấn đề (thay vì tính thuật toán của nó). Scott Aaronson [1] có một bài báo tổng quan rất hay, thảo luận các khả năng này. Kết luận chung là các khả năng này rất khó xảy ra. Một tham khảo thú vị là nghiên cứu của Razborov và Rudich [28] năm 1993, trong đó các tác giả đưa bằng chứng cho thấy: "có lẽ lý do mà chứng minh $\mathbf{P} \neq \mathbf{NP}$ rất khó là tại vì ... $\mathbf{P} \neq \mathbf{NP}$!" Ta sẽ thảo luận kỹ hơn về kết quả "oái oăm" này vào dịp khác.

Như vậy, khả năng cuối cùng là $\mathbf{P} \neq \mathbf{NP}$, và đa số khoa học gia đều tin vào khả năng này. Thế ta phải làm gì nếu **P** thật sự khác **NP**? Ta thử liệt kê vài hướng đi hiện nay:

- Có lẽ mô hình máy Turing không phải là mô hình tính toán mạnh nhất. Điều này có nghĩa là ta



Richard Karp



Donald Knuth

phải thiết kế các máy tính theo kiểu khác, dựa trên một nguyên tắc hoàn toàn khác với nguyên tắc thiết kế hiện tại (với bộ vi xử lý, bộ nhớ, bus, thiết bị ngoại vi, vân vân). Đã có vài đề cử, trong đó quan trọng nhất là tính toán lượng tử (quantum computing [26]), và tính toán sinh học (biological computing [2]). Công trình đột phá của Peter Shor [29, 30] cho thấy ta có thể phân tích một số nguyên ra thừa số (FACTORIZATION) trong thời gian đa thức với một máy tính lượng tử. Bài toán này là một trong số rất ít các bài toán trong $\mathbf{NP}$ mà ta không biết là nó ở trong $\mathbf{P}$ hay là $\mathbf{NP}$-đầy đủ. Ngoài bài báo của Peter Shor, thật sự vẫn chưa có bằng chứng nào ủng hộ khả năng là ta có thể giải các bài toán $\mathbf{NP}$-đầy đủ trong thời gian đa thức với máy tính lượng tử. Tình hình với máy tính sinh học còn tệ hơn.

- Trong hướng đi thứ hai, ta chấp nhận rằng các bài toán $\mathbf{NP}$-đầy đủ không thể giải được nhanh bằng các máy tính vật lý (bất kể mô hình vật lý nào). Đến đây thì có vài nhánh nghiên cứu khác:
  - Tìm cách hiệu quả nhất (dù là trong thời gian không đa thức) để giải các bài toán này. Các nghiên cứu về quy hoạch nguyên (integer programming) là một ví dụ tốt.
  - Trong thời gian đa thức, tìm lời giải càng gần tối ưu càng tốt. Đây là hướng đi của các giải thuật xấp xỉ (approximation algorithms), một hướng rất quan trọng trong khoa học máy tính hiện đại.
  - Một hướng khác cũng quan trọng không kém là dùng các giải thuật ngẫu nhiên hóa (randomized algorithms). Ta có thể tìm lời giải tối ưu trong thời gian kỳ vọng là đa thức (expected polynomial time), hoặc tìm lời giải với giá kỳ vọng (expected cost) gần tối ưu trong thời gian đa thức.

Tất cả các khả năng và hướng đi trên đưa chúng ta đến các phương trời mới của khoa học máy tính, toán học và vật lý, là các bước tiến quan trọng cho quá trình tìm hiểu vũ trụ và các quy luật hoạt động của tự nhiên (trong đó tính toán là một quá trình căn bản).

Người mở một trong những cánh cửa quan trọng nhất chính là Cantor!

Tài liệu

[1] S. AARONSON, *Is P versus NP formally independent?*, Bulletin of the European Association for Theoretical Computer Science, 81 (2003).

[2] L. M. ADLEMAN, *Computing with DNA*, Scientific American, 279 (1998), pp. 54–61.

[3] M. AIGNER AND G. M. ZIEGLER, *Proofs from The Book*, Springer-Verlag, Berlin, second ed., 2001. Including illustrations by Karl H. Hofmann.

[4] M. BLUM, *A machine-independent theory of the complexity of recursive functions.*, J. Assoc. Comput. Mach., 14 (1967), pp. 322–336.

[5] A. CHURCH, *A set of postulates for the foundation of logic*, Ann. of Math. (2), 33 (1932), pp. 346–366.

[6] A. CHURCH, *An unsolvable problem in elementary number theory*, American Journal of Mathematics, 58 (1936), pp. 345–363.

[7] ———, *Alonzo church is interviewed by william aspray on 17 may 1984 at the university of california at los angeles.*, 1984. The Princeton Mathematics Community in the 1930s, Transcript Number 5 (PMC5), http://libweb.princeton.edu/libraries/firestone/rbsc/finding_aids/mathoral/pmc05.htm.

[8] A. COBHAM, *The intrinsic computational difficulty of functions*, in Logic, Methodology and Philos. Sci. (Proc. 1964 Internat. Congr.), North-Holland, Amsterdam, 1965, pp. 24–30.

[9] S. A. COOK, *The complexity of theorem-proving procedures*, in Conference Record of the Third Annual ACM Symposium on the Theory of Computing, 1971, pp. 151–158.

[10] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to algorithms*, MIT Press, Cambridge, MA, second ed., 2001.

[11] P. CRESCENZI AND V. K. (EDS.), *A compendium of NP-optimization problems*. http://www.nada.kth.se/

[12] M. DUMMETT, *Elements of intuitionism*, vol. 39 of Oxford Logic Guides, The Clarendon Press Oxford University Press, New York, second ed., 2000.

[13] J. EDMONDS, *Paths, trees, and flowers*, Canad. J. Math., 17 (1965), pp. 449–467.

[14] M. R. GAREY AND D. S. JOHNSON, *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[15] K. GÖDEL, *Über formal unentscheidbare sätze der principia mathematica und verwandter systeme I*, Monatshefte für Mathematik und Physik, 38 (1931), pp. 173–198. English translation, "On formally undecidable propositions of Principia mathematica and related systems I.", in van Heijenoort, J. (ed.) (1967) From Frege to Gödel: A Source Book in Mathematical Logic 1879–1931. Harvard University Press. pp 592–618.

[16] ———, *On undecidable propositions of formal mathematical systems*, 1934. Lecture notes taken by Kleene and Rosser at the Institute for Advanced Study. Reprinted in Davis, M. (ed.) 1965. *The Undecidable*. New York: Raven.

[17] ———, *Über die länge von beweisen*, Ergebnisse eines Mathematischen Kolloquiums, 7 (1936), pp. 23–24.

[18] J. HARTMANIS AND R. E. STEARNS, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc., 117 (1965), pp. 285–306.

[19] J. HERBRAND, *Sur la non-contradiction de l'arithmetique*, Journal fur die reine und angewandte Mathematik, 166 (1932), pp. 1–8.

[20] C. M. INSTITUTE. `http://www.claymath.org/millennium/`.

[21] R. M. KARP, *Reducibility among combinatorial problems*, in Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), Plenum, New York, 1972, pp. 85–103.

[22] S. KLEENE, *Lambda-definability and recursiveness*, Duke Mathematical Journal, 2 (1936), pp. 340–353.

[23] D. KNUTH, *Postscript about np-hard problems*, SIGACT News, 6 (1974), pp. 15–16.

[24] ———, *A terminological proposal*, SIGACT News, 6 (1974), pp. 12–18.

[25] L. LEVIN, *Universal search problems (in russian)*, Problemy Peredachi Informatsii, 9 (1973), pp. 265–266. English translation in Trakhtenbrot, B. A.: A survey of Russian approaches to Perebor (brute-force search) algorithms. Annals of the History of Computing, 6 (1984), pp 384–400.

[26] M. A. NIELSEN AND I. L. CHUANG, *Quantum computation and quantum information*, Cambridge University Press, Cambridge, 2000.

[27] M. O. RABIN, *Degree of difficulty of computing a function and a partial ordering of recursive sets*, 1960. Tech Rep. No. 2.

[28] A. A. RAZBOROV AND S. RUDICH, *Natural proofs*, J. Comput. System Sci., 55 (1997), pp. 24–35. 26th Annual ACM Symposium on the Theory of Computing (STOC '94) (Montreal, PQ, 1994).

[29] P. W. SHOR, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.

[30] ———, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Rev., 41 (1999), pp. 303–332 (electronic).

[31] M. SIPSER, *The history and status of the p versus np question*, in STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, ACM Press, 1992, pp. 603–618.

[32] E. SNAPPER, *The three crises in mathematics: logicism, intuitionism and formalism*, Math. Mag., 52 (1979), pp. 207–216.

[33] R. TIESZEN, ed., *Perspectives on intuitionism*, Canadian Society for History and Philosophy of Mathematics, Winnipeg, MB, 1998. Philos. Math. (3) **6** (1998), no. 2.

[34] A. TURING, *On computable numbers with an application to the entscheidungsproblem*, Proc. London Math. Soc., 42 (1936), pp. 230–265. "A correction" ibid, 43, 544-546.

[35] A. M. TURING, *Computing machinery and intelligence*, Mind, 59 (1950), pp. 433–460.

[36] D. VAN DALEN, *The development of Brouwer's intuitionism*, in Proof theory (Roskilde, 1997), vol. 292 of Synthese Lib., Kluwer Acad. Publ., Dordrecht, 2000, pp. 117–152.

[37] J. VON NEUMANN, *A certain zero-sum two-person game equivalent to the optimal assignment problem*, in Contributions to the theory of games, vol. 2, Annals of Mathematics Studies, no. 28, Princeton University Press, Princeton, N. J., 1953, pp. 5–12.

[38] A. N. WHITEHEAD AND B. RUSSELL, *Principia mathematica to *56*, Cambridge Mathematical Library, Cambridge University Press, Cambridge, 1997. Reprint of the second (1927) edition.

[39] S. YABLONSKI, *The algorithmic difficulties of synthesizing minimal switching circuits*, Problemy Kibernetiki 2, (1959), pp. 75–121. Fizmatgiz, Moscow, English translation in Problems of Cybernetics II.

[40] ———, *On the impossibility of eliminating brute force search in solving some problems of circuit theory*, Doklady, AN SSSR 124 (1959), pp. 44–47.

[41] B. H. YANDELL, *The honors class*, A K Peters Ltd., Natick, MA, 2002. Hilbert's problems and their solvers.