ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

THIẾT KẾ VÀ ĐÁNH GIÁ THUẬT TOÁN

Bài 8

Lớp bài toán P, NP và NPC

(P, NP and NPC Problems)

Nguyễn Thị Hồng Minh
minhnth@gmail.com

Nội dung

1. **Giới thiệu**
2. **Bài toán “dễ” (tractable) và “khó” (intractable)**
3. **Lớp bài toán P và NP**
4. **NP-đầy đủ**
5. **Giải thuật xấp xỉ**

Giới thiệu

- Thuật toán là công cụ hữu hiệu để giải các bài toán
- Tuy nhiên, sức mạnh của thuật toán không phải là “vô hạn”
- Một số bài toán không thể giải bằng thuật toán
- Một số bài toán khác có thể giải bằng thuật toán, nhưng với độ phức tạp thời gian lớn (not in polynomial time)
- Thông thường có một tiệm cận cho việc đánh giá hiệu quả của thuật toán (sử dụng kí pháp O, Θ, Ω)

Bài toán "dễ" và "khó"

- **Độ phức tạp thuật toán**

  Chúng ta có thể xem xét hiệu quả của một thuật toán theo 2 cách:

  - Lớp hàm đánh giá độ phức tạp (theo kí pháp O lớn).
    - Ví dụ thuật toán sắp chọn là $O(n^2)$,
    - Thuật toán chuyển tháp Hà Nội là $O(2^n)$.
  - Xem xét hiệu quả của thuật toán trên cơ sở so sánh với các thuật toán khác giải cùng bài toán đó

- **Hai lớp hàm đánh giá độ phức tạp**
  - *Lớp hàm đa thức*: hàm có dạng $O(n^k)$, có thể hiểu rộng hơn theo nghĩa bị chặn trên bởi n^k *(*với hằng số *k* nào đó)
    - Ví dụ: $O(1)$, $O(\log n)$, $O(n)$, $O(n\log n)$, $O(n^2)$, $O(n^3)$
  - *Lớp hàm mũ*: Các hàm còn lại
    - Ví dụ: $\Theta(2^n)$, $\Theta(n!)$, $\Theta(n^n)$

Bài toán “dễ” và “khó”

- **Phân lớp bài toán theo độ phức tạp**

  Liệu có hay không ***một giải thuật với thời gian đa thức*** để giải một bài toán?

  - ***Có***
    - Có thuật toán để giải bài toán với thời gian thực hiện của giải thuật thuộc lớp hàm đa thức.
  - ***Không***
    - Vì mọi thuật toán để giải đều thuộc lớp hàm mũ.
    - Vì không tồn tại thuật toán để giải bài toán.
  - ***Không biết***
    - Nhưng nếu tìm được thuật toán như vậy, có thể nó sẽ cung cấp một phương thức chung để giải nhiều bài toán khác với thời gian đa thức.

Bài toán "dễ" và "khó"

- **Tính "dễ xử lí"**
  - Một thuật toán giải bài toán trong thời gian đa thức nếu thời gian để thực hiện trong trường hợp xấu nhất thuộc lớp hàm đa thức.
  - ***Bài toán "dễ" (tractable)***: bài toán có thể giải được trong thời gian đa thức. Cận trên là hàm đa thức.
    - Ví dụ: Tìm kiếm trong danh sách được sắp ($O(\log n)$); sắp xếp danh sách ($O(n\log n)$).
  - ***Bài toán "khó" (intractable)***: bài toán không thể giải trong thời gian đa thức. Cận dưới hoặc cận chặt là hàm mũ.
    - Ví dụ: Thuật toán chuyển tháp Hà Nội ($\Theta(2^n)$); liệt kê các hoán vị khác nhau của n số ($\Theta(n!)$).

Bài toán “dễ” và “khó”

- **Xử lí các bài toán “khó” (intractable)**
  - Tìm kiếm những cải thiện càng nhiều càng tốt (đối với thuật toán), hi vọng sự khả thi. Ví dụ thuật toán quay lui.
  - Giải bài toán trong một số trường hợp đơn giản, đặc biệt nào đó. Có thể những phiên bản đơn giản này sẽ có ích trong khi tránh được độ phức tạp hàm mũ.
  - Sử dụng thuật toán xác suất thời gian đa thức, giúp có câu trả lời đúng với xác suất cao nhất. Nghĩa là “thỏa hiệp” tính đúng và quan tâm tới tốc độ.
  - Với các bài toán tối ưu, sử dụng thuật toán xấp xỉ với thời gian đa thức, tuy nhiên không đảm bảo là tìm được lời giải tối ưu nhất.

P và NP

- **Hai kiểu của bài toán cơ bản**
  - ***Bài toán tối ưu***: tìm lời giải làm sao cho một hàm mục tiêu đạt giá trị (nào đó/*max/min)*.
  - ***Bài toán quyết định***: bài toán có 1 trong hai phương án trả lời, YES hoặc NO (hoặc ***true*** hoặc ***false***, hoặc 0 hoặc 1).
  - Nhiều bài toán có thể có cả hai kiểu tối ưu và quyết định.
    - Ví dụ: Bài toán người du lịch; Xếp balo; Tô màu đồ thị…
  - Với nhiều bài toán không phải quyết định, có thể có phát biểu dạng quyết định
    - Ví dụ: Bài toán TSP tối ưu, thêm tham số k, được bài toán: Có hay không đường đi có chi phí *k*.

P và NP

- **Độ phức tạp lớp P**
  - ***Định nghĩa***: Lớp P (*Polinomial*) là tập các bài toán quyết định có thể giải bằng thuật toán với độ phức tạp đa thức.
  - Phát biểu khác: Một bài toán thuộc lớp P nếu
    - Nó là bài toán quyết định
    - Tồn tại thuật toán giải bài toán với độ phức tạp $O(n^k)$, trong đó n là kích thước dữ liệu vào, k là hằng số.
  - Như vậy, P là tập các bài toán quyết định "dễ" (tractable decision problems).

P và NP

- **Thuật toán không đơn định**
  - Lớp độ phức tạp thuật toán thứ hai được gọi là NP (Non-deterministically Polynomial - Không đơn định đa thức).
  - ***Thuật toán không đơn định (non-deterministic algorithm)***: Thuật toán giải quyết một trường hợp *I* cụ thể của một bài toán quyết định thông qua hai bước:
    - Bước đề xuất (*Guessing stage*): Một chuỗi *S* được đề xuất như nghiệm của bài toán (Một số nghiệm đề xuất này có thể được sinh là đồng thời tại cùng một thời điểm - parallel).
    - Bước kiểm chứng (*Verification stage*): Một thuật toán xác định kiểm tra xem *S* có phải là nghiệm của trường hợp *I* của bài toán không (kết quả sẽ là *YES* nếu *S* là nghiệm của *I* hoặc *NO* trong trường hợp ngược lại).

P và NP

- **Độ phức tạp lớp NP**
  - Định nghĩa không hình thức: NP là lớp các bài toán quyết định mà có thể dễ dàng kiểm tra lời giải đã được đề xuất.
  - Định nghĩa hình thức: NP là lớp các bài toán quyết định được giải bằng các thuật toán không đơn định với thời gian đa thức.
  - Hình ảnh bài toán NP
    - Có những vấn đề rất khó tìm ra lời giải, nhưng lại dễ thẩm tra kết quả.
    - Ví dụ: tìm các nhân tử nguyên tố của 3717421 – Phức tạp

P và NP

- **Độ phức tạp lớp NP**
  - Định nghĩa không hình thức: NP là lớp các bài toán quyết định mà có thể dễ dàng kiểm tra lời giải đã được đề xuất.
  - Định nghĩa hình thức: NP là lớp các bài toán quyết định được giải bằng các thuật toán không đơn định với thời gian đa thức.
  - Hình ảnh bài toán NP
    - Có những vấn đề rất khó tìm ra lời giải, nhưng lại dễ thẩm tra kết quả.
    - Ví dụ: tìm các nhân tử nguyên tố của 3717421 – Phức tạp

P và NP

- **Độ phức tạp lớp NP**
  - Định nghĩa không hình thức: NP là lớp các bài toán quyết định mà có thể dễ dàng kiểm tra lời giải đã được đề xuất.
  - Định nghĩa hình thức: NP là lớp các bài toán quyết định được giải bằng các thuật toán không đơn định với thời gian đa thức.

  - Hình ảnh bài toán NP
    - Có những vấn đề rất khó tìm ra lời giải, nhưng lại dễ thẩm tra kết quả.
    - Ví dụ: tìm các nhân tử nguyên tố của 3717421 – Phức tạp
    - Nhưng dễ kiểm tra rằng 3607 x 3803 = 13717421

P và NP

- **Độ phức tạp lớp NP**
  - **Ví dụ 1:** Thuật toán không đơn định tìm chu trình Hamiltonian
    - *Guessing stage*: sinh dãy $v_1\ v_2\ \ldots\ v_n$, với mỗi v_i là một cạnh của đồ thị $G=(V,E)$, $(n = |V|)$.
    - *Verification stage*: kiểm tra xem $v_1\ v_2\ \ldots\ v_n$ có là một đường đi không, nghĩa là $(v_i,v_{i+1})\in E$,

    Dễ thấy bước kiểm chứng có độ phức tạp thời gian là $O(n)$

P và NP

- **Độ phức tạp lớp NP**
  - **Ví dụ 2:** Thuật toán không đơn định tô màu đồ thị
    - *Guessing stage*: sinh một dãy *s* các kí tự $c_1\ c_2\ \ldots\ c_q$, là dãy giả thiết các màu được tô cho đồ thị.
    - *Verification stage*: kiểm tra xem mỗi màu c_i có tô được cho cạnh v_i không.

P và NP

- **Độ phức tạp lớp NP**
  - **Ví dụ 3:** Bài toán thỏa được dạng chuẩn hội (*CNF Satisfiability*)
    - Bài toán: Cho một công thức logic biểu diễn ở dạng chuẩn hội (conjunctive normal form-CNF), có thể tìm dãy giá trị true/false của các biến để CNF là đúng?
    - Đây là bài toán thuộc lớp *NP*.
    - Thuật toán không đơn định:
      - Sinh dãy giá trị true/false của các biến.
      - Kiểm tra với dãy được sinh thì CNF có đúng không.
    - Ví dụ:

$$(A \vee \neg B \vee \neg C) \wedge (\neg A \vee B) \wedge (\neg B \vee D \vee E) \wedge (F \vee \neg D)$$

P và NP

- **Độ phức tạp lớp NP**
  - **Ví dụ 3:** Bài toán thỏa được dạng chuẩn hội (CNF Satisfiability)
    - Ví dụ:

$$(A \vee \neg B \vee \neg C) \wedge (\neg A \vee B) \wedge (\neg B \vee D \vee E) \wedge (F \vee \neg D)$$

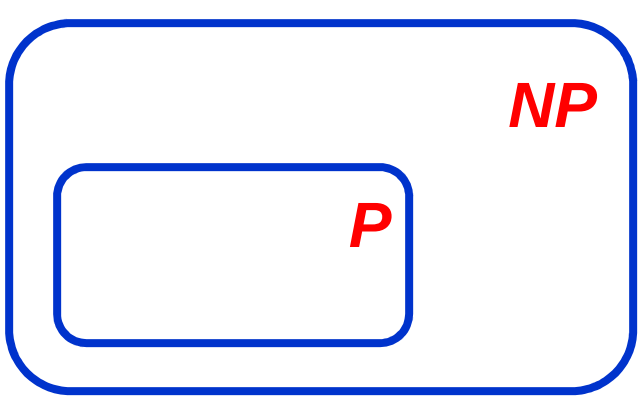
	A	B	C	D	E	CNF
1	0	1	1	0	1	0
2	1	0	0	0	0	1
3	1	1	0	0	0	0
4	. . .					

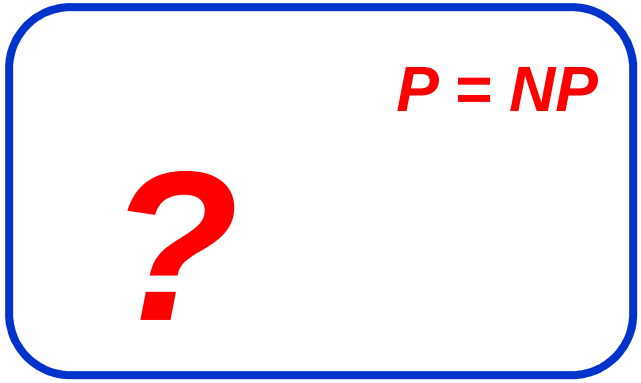
P và NP

- **P và NP**
  - $P \subseteq NP$
    - Mọi bài toán quyết định được giải bằng thuật toán với thời gian đa thức cũng có thể giải được bằng thuật toán không đơn định thời gian đa thức.
  - Để chứng minh điều này, quan sát thấy bất kì thuật toán đơn định nào đều có thể được sử dụng làm thuật toán để kiểm chứng (verification stage) của một thuật toán không đơn định.
    - Nếu $I \in P$ và A là thuật toán đơn định thời gian đa thức để giải I, chúng ta có thể nhận được một thuật toán không đơn định thời gian đa thức giải I bằng việc sử dụng A như thuật toán kiểm chứng mà bỏ qua bước đề xuất.

P và NP

- **P và NP**
  - *P = NP* - MỘT CÂU HỎI THÁCH THỨC?
    - Các bài toán thuộc lớp NP có thể giải trong thời gian đa thức?
    - Sơ đồ quan hệ:





P và NP

- **P và NP**
  - P = NP? - MỘT CÂU HỎI THÁCH THỨC?
    - Đây là một trong những thách thức thế kỉ của lĩnh vực Toán-Tin. Nếu giải được vấn đề này (nghĩa là khẳng định P=NP hoặc P#NP) bạn có thể đạt được phần thưởng 1 TRIỆU đô la của hội Toán học Mỹ (claymath)!

      http://www.claymath.org/millennium/

    - Tại sao bài toán lại quan trọng như vậy?
      - *"Nếu P=NP, Một mặt, điều này sẽ giải quyết được rất nhiều vấn đề tin học ứng dụng trong công nghiệp; nhưng mặt khác lại sẽ phá hủy sự bảo mật của toàn bộ các giao dịch tài chính thực hiện qua Internet"*.
      - Mọi "ngân hàng" và các hệ thống bảo mật đều hoảng sợ trước vấn đề lôgic nhỏ bé và cơ bản này!

NP- Complete

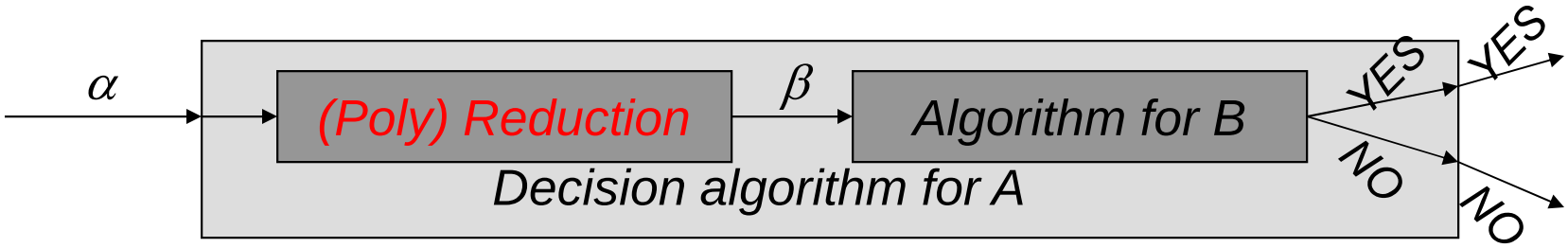
- **NP-**Complete (NP đầy đủ)
  - Tính *NP* đầy đủ (NP-Completelly) của bài toán: bài toán quyết định khó nhất trong lớp NP.
  - Nếu có thuật toán với thời gian đa thức cho một bài toán NPC thì sẽ có thuật toán với thời gian đa thức cho mỗi bài toán NP.
  - Ví dụ: Hamiltonian cycle, Traveling salesman, Knapsack, Bin packing, Graph coloring, CNF Satisfiability.

NP- Complete

- **Bài toán NPC**
  - Hình thức: Một bài toán quyết định D thuộc lớp NPC nếu:
    - Nó thuộc lớp NP.
    - Mỗi bài toán trong NP là thu giảm đa thức (*polynomially reducible*) tới D.
  - Lớp các bài toán *NP*-đầy đủ gọi là *NPC*.

NP- Complete

- **Bài toán NPC**
  - Phép thu giảm đa thức
    - Nếu α, β là một trường hợp của bài toán A, B tương ứng. Phép thu giảm đa thức từ A tới (về B) là phép chuyển có tính chất sau:
      - Thời gian chuyển là đa thức
      - Câu trả lời là như nhau ở hai trường hợp của bài toán (câu trả lời của α là YES nếu và chỉ nếu câu trả lời của β là YES).



NP- Complete

- **Bài toán NPC**
  - Chứng minh bài toán thuộc lớp NPC:
    - Chứng minh bài toán thuộc về lớp *NP*.
    - Chỉ là rằng có một bài toán NPC đã biết có thể được biến đổi (*transform*) về bài toán đó bằng phép thu giảm đa thức
  - Định lí Cook (1971): Bài toán NPC đầu tiên là bài toán thỏa được CNF.

NP- Complete

- **Quan hệ P, NP, NPC**
  - P $\subseteq$ NP (Sure)
  - NPC $\subseteq$ NP (sure)
  - P = NP (or P $\subset$ NP, or P $\neq$ NP) ???
  - NPC = NP (or NPC $\subset$ NP, or NPC $\neq$ NP) ???
  - P $\neq$ NP: one of the deepest, most perplexing open research problems in (theoretical) computer science since 1971.

NP- Complete

- **Một số tranh luận về P, NP, NPC**
  - Chưa tìm được một thuật toán đa thức cho bất kì bài toán NPC nào (mặc dù có khá nhiều bài toán NPC)
  - Chưa có chứng minh nào chỉ ra rằng rằng không tồn tại thuật toán đa thức cho bất kì bài toán NPC (mặc dù đã rất cố gắng).
  - Hầu hết các nhà khoa học về lí thuyết tính toán đều ***tin rằng*** NPC là intractable (tức là khó và $P \neq NP$).
  - Thực tế???

NP- Complete

- **Một số kĩ thuật đối phó với bài toán NPC**
  - *Xấp xỉ*: Thay vì tìm nghiệm tối ưu, mà tìm lời giải gần nhất với sự tối ưu.
  - *Ngẫu nhiên:* Sử dụng phép ngẫu nhiên để làm thuật toán chạy nhanh hơn, chấp nhận một số trường hợp thất bại với xác suất nhỏ.
  - *Hạn chế*: Hạn chế cấu trúc của dữ liệu đầu vào để thuật toán chạy nhanh hơn.
  - *Tham số hóa*: Thuật toán có thể chạy nhanh hơn nếu một số tham số nào đó của dữ liệu đầu vào được cố định (are fixed).
  - *Heuristic*: Thuật toán có thể thực hiện tốt chấp nhận được trong nhiều trường hợp theo kinh nghiệm nào đó. Nhưng có thể không chứng minh được cả hai tính chất chạy nhanh và cho ra kết quả chính xác.

Thuật toán xấp xỉ

- **Khái niệm**
  - Lớp bài toán quyết định khó trong NP là *NP*-đầy đủ (NPC).
  - Dạng tối ưu của các bài toán này là N-khó (*NP*-hard). Hiện chưa có thuật toán thời gian đa thức để giải các bài toán này.
  - Sẽ làm gì để giải những bài toán như vậy?
  - Nhắc lại: thuật toán thử sai dạng nhánh cận cũng không đảm bảo giải quyết được những bài toán như vậy.
  - Các tiếp cận khác: giải xấp xỉ, hi vọng thời gian sẽ nhanh hơn.

Thuật toán xấp xỉ

- **Mục tiêu:**
  - Đảm bảo thời gian chạy thuật toán là đa thức.
  - Đảm bảo tìm được nghiệm đủ tốt (gần với nghiệm tối ưu).
- **Khó khăn:**
  - Cần chứng minh nghiệm tìm được đủ tốt, tức là gần với nghiệm tối ưu nhưng lại không biết nghiệm tối ưu!

Thuật toán xấp xỉ

- **Hệ số chính xác**
  - Nếu sử dụng giải thuật xấp xỉ, chúng ta có thể muốn biết độ chính xác của giải thuật đó?
  - Sai số quan hệ (Relative error): càng nhỏ càng tốt $\rightarrow$ 0):

$$re(s_a) = \frac{f(s_a) - f(s^*)}{f(s^*)}$$

trong đó: s_* là một nghiệm chính xác

s_a là nghiệm xấp xỉ.

  - Hệ số chính xác (accurate ratio): càng lớn càng tốt $\rightarrow$ 1

$$ar(s_a) = \frac{f(s_a)}{f(s^*)}$$

Thuật toán xấp xỉ

- **Heuristic**
  - Tại sao cần thuật toán xấp xỉ:
    - Đôi khi một lời giải đủ tốt là đủ.
    - Trong thực tế có thể dữ liệu vào không thật chính xác.
  - Thuật toán xấp xỉ dựa trên một số cảm nhận, kinh nghiệm (heuristic) nào đó của bài toán.
  - Một heuristic là một qui tắc được rút ra từ kinh nghiệm hơn là một khẳng định được chứng minh bằng toán học.

Thuật toán xấp xỉ

- Thiết kế thuật toán xấp xỉ cho một số bài toán
  - Thuật toán phụ thuộc vào bài toán được giải, mỗi bài toán có cách xây dựng thuật toán khác nhau
  - Ví dụ:
    - Approximation Agorithms for TSP
    - Approximation Agorithms for Knapsack Problem
    - Approximation Agorithms for Vertex Cover Problem
    - …

    http://www.cs.iupui.edu/~xkzou/teaching/CS580/ (chapter 35)