ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

THUẬT TOÁN ỨNG DỤNG

NP-Đầy đủ

ONE LOVE. ONE FUTURE.

NỘI DUNG

- **Giới thiệu**
- Lớp bài toán P, NP, NPC
- Bài toán quyết định và Bài toán tối ưu
- Phép quy dẫn
- Chứng minh NP-đầy đủ
- Các hướng giải bài toán NP-khó

ĐỘ khó của bài toán

- Đánh giá độ khó của bài toán là ước lượng thời gian tính toán của thuật toán tốt nhất trong số tất cả các thuật toán giải bài toán, kể cả các thuật toán đã biết và các thuật toán còn chưa biết.
- Hai cách tiếp cận:
  - Cách 1: Tìm cách đưa ra đánh giá cận dưới độ phức tạp của bài toán (Khóa học phân tích độ phức tạp thuật toán cơ bản và nâng cao)
  - Cách 2: Tập trung vào chỉ ra mức độ khó của bài toán có thể so sánh với bất kỳ bài toán khó hiện biết (Lý thuyết NP-đầy đủ, NP-khó)
- Việc đánh giá độ phức tạp tính toán của bài toán giữ vai trò rất quan trọng trong đính hướng thiết kế thuật toán để giải bài toán đặt ra.

Lịch sử

- Từ những năm 1960, Steve Cook và Dick Karp đã đưa ra khái niệm, một ***thuật toán hiệu quả*** là thuật toán có thời gian tính toán phải là đa thức, $O(n^c)$ với c là hằng số và n là kích thước bài toán.
- Các nhà khoa học máy tính cũng nhận thấy rằng: Tồn tại nhiều lớp bài toán, việc tìm ra các thuật toán hiệu quả như vậy rất khó khăn, thậm chí là không biết nó có tồn tại hay không/
- Vì vậy, Cook và Karp cùng một số nhà khoa học máy tính khác đã đưa ra định nghĩa lớp bài toán NP-đầy đủ, mà cho đến hiện nay giới khoa học vẫn tin rằng không tồn tại thuật toán hiệu quả cho lớp bài toán này.

NỘI DUNG

- Giới thiệu
- **Lớp bài toán P, NP, NPC**
- Bài toán quyết định và Bài toán tối ưu
- Phép quy dẫn
- Chứng minh NP-đầy đủ
- Các hướng giải bài toán NP-khó

Khái niệm bài toán P và NP

- **P** là lớp các bài toán ***giải được*** trong thời gian đa thức.
  - Bài toán kiểm tra tính liên thông của đồ thị có thể giải được nhờ thuật toán với thời gian tính toán $O(n^2)$, với n là số lượng đỉnh của đồ thị. Vì vậy, bài toán này thuộc lớp **P.**
  - Bài toán nhân dãy ma trận có thể giải được nhờ thuật toán quy hoạch động với thời gian $O(n^3)$, với n là kích thước ma trận. Vì vậy, bài toán này thuộc lớp **P.**
- **NP** là lớp các bài toán ***kiểm chứng được*** trong thời gian đa thức.
  - ***Kiểm chứng được*** là chỉ ra được thuật toán chạy trong thời gian đa thức kiểm chứng tính chính xác của một bộ kết quả đầu ra với một bộ dữ liệu đầu vào tương ứng.
  - Ví dụ: Bài toán tìm chu trình Hamilton, việc kiểm tra dãy đỉnh $v_1, v_2, \dots, v_n, v_1$ có là chu trình Hamilton của đồ thị đã cho hay không có thể thực hiện trong thời gian đa thức. Vì vậy, bài toán này thuộc lớp **NP**.

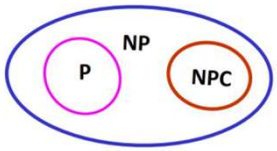
Lớp bài toán NP-đầy đủ (NPC)

- **NP-đầy đủ** là lớp bài toán con của **NP** và khó không kém bất cứ bài toán nào trong **NP**.
  - Đây là lớp bài toán khó nhất trong lớp **NP**.
- Vì sao không nói "Giải được trong thời gian hàm mũ" hay "Giải được không trong thời gian đa thức"?
  - Không giải được khi kích thước bài toán lớn
- $O(n^{100})$ và $O(2^n)$
  - Thuật toán $O(n^{100})$ vẫn là thuật toán đa thức, tuy nhiên thời gian tính toán này làm cho thuật toán không có tính ứng dụng thực tế.
  - Các thuật toán chạy trong thời gian đa thức áp dụng trong thực tế có thời gian tính toán nhỏ hơn rất nhiều so với $0(n^{100})$.
  - Thường khi một thuật toán đa thức được tìm ra cho một bài toán nào đó sẽ có thêm nhiều thuật toán hiệu quả mới sẽ được tìm ra cho bài toán.

Mối quan hệ giữa P, NP và NPC

- Chắc chắn
  - $\boldsymbol{P \subseteq NP}$
  - $\boldsymbol{NPC \subseteq NP}$
- Vấn đề mở:
  - $\boldsymbol{P = NP}$ hoặc $\boldsymbol{P \subset NP}$ hoặc $\boldsymbol{P \neq NP}$
  - $\boldsymbol{NPC = NP}$ hoặc $\boldsymbol{NPC \subset NP}$ hoặc $\boldsymbol{NPC \neq NP}$
- Câu hỏi $\boldsymbol{P = NP}$ hay không?
  - Một trong những vấn đề học búa nhất và là trung tâm của lý thuyết tính toán.
  - Cho đến hiện nay, câu hỏi này vẫn là chưa có câu trả lời chính xác.

Mối quan hệ P, NP và NPC – Quan điểm (Chưa chứng minh được)



- Quan điểm của rất nhiều nhà nghiên cứu khoa học máy tính, và chưa chứng minh được như sau:

$$\boldsymbol{P \in NP, NPC \in NP, P \cap NPC = \emptyset}$$

Tại sao phải nghiên cứu lý thuyết NP-đầy đủ

- Nếu một bài toán chứng minh được là thuộc lớp **NP-đầy đủ** ➔ Chứng minh được độ khó của bài toán ~ Khó nhất trong lớp bài toán **NP**.
- Nếu một bài toán chứng minh được là thuộc **NP-đầy đủ** ➔ Không lãng phí thời gian để tìm ra thuật toán hiệu quả. Thay vào đó, tập trung vào thiết kế thuật toán gần đúng hoặc heuristic/metaheuristic cho bài toán, hoặc tìm lời giải hiệu quả cho một số trường hợp đặc biệt của bài toán.
- Chú ý: Một số bài toán rất dễ phát biểu, ai cũng có thể giải thử, nhưng thực tế nó rất khó vì thuộc lớp **NP-đầy đủ**.
  - Ví dụ: Có thể chia một tập các số nguyên thành 2 tập con không giao nhau sao cho tổng các phần tử trong 2 tập con bằng nhau.

NỘI DUNG

- Giới thiệu
- Lớp bài toán P, NP, NPC
- **Bài toán quyết định và Bài toán tối ưu**
- Phép quy dẫn
- Chứng minh NP-đầy đủ
- Các hướng giải bài toán NP-khó

Bài toán quyết định và Bài toán tối ưu

- ***Bài toán tối ưu (PO)*** yêu cầu tìm ra một lời giải thỏa mãn một tập ràng buộc và cực đại (max) hoặc cực tiểu (min) một hàm mục tiêu nào đó.
  - Ví dụ: Bài toán tìm đường đi ngắn nhất trên đồ thị: Cho *G*, *u*, *v*, hãy tìm một đường đi từ *u* tới *v* đi qua ít cạnh nhất trên *G*.
- ***Bài toán quyết định (PD)*** chỉ trả lời YES hoặc NO.
  - Trong bài toán quyết định, một số bộ dữ liệu đầu vào sẽ có câu trả lời YES; ngược lại một số bộ dữ liệu đầu vào sẽ có câu trả lời NO.
  - Bộ dữ liệu có câu trả lời YES, được gọi là bộ dữ liệu YES; ngược lại gọi là bộ dữ liệu NO.
  - Ví dụ: Tồn tại hay không một đường đi từ *u* tới *v* trên đồ thị *G* đi qua tối đa *k* cạnh?

Bài toán quyết định và Bài toán tối ưu

- Lớp ***NP-đầy đủ*** chỉ bao gồm các ***bài toán quyết định***.
  - Nếu một bài toán tối ưu kiểm chứng được tính tối ưu trong thời gian đa thức thì cũng có thể giải được bằng một thuật toán hiệu quả (có độ thức tạp tính toán đa thức).
  - Việc quy dẫn giữa các bài toán quyết định thường dễ dàng hơn so với việc quy dẫn giữa các bài toán tối ưu.

Bài toán quyết định của một bài toán tối ưu

- Xét bài toán tối ưu:

$$(PO): \max\{f(x): x \in D\}$$

- Bài toán quyết định/Dạng quyết định (PD) tương ứng với bài toán tối ưu là:

$$(PD) \text{“}Cho\ giá\ trị\ k, hỏi\ có\ tồn\ tại\ u \in D\ sao\ cho\ f(u) \geq k\ ?\text{”}$$

  - Lời giải của bài toán tối ưu dẫn ra trực tiếp đáp án của bài toán quyết định tương ứng.
  - Nếu bài toán quyết định tương ứng với một bài toán tối ưu có thể giải được hiệu quả (tồn tại thuật toán chạy trong thời gian đa thức trả lời chính xác) thì bài toán tối ưu đó cũng giải được hiệu quả.,

Mối liên hệ giữa Bài toán tối ưu và Bài toán quyết định tương ứng

- Giả sử hàm f nhận giá trị nguyên trên miền giá trị D; α_0 và β_0 là cận dưới và cận trên của giá trị hàm f trên D.
- Giả sử A là thuật toán hiệu quả giải bài toán quyết định tương ứng với độ phức tạp $0(n^c)$
- Thuật toán tìm kiếm nhị phân giải bài toán tối ưu. Tại mỗi bước lặp k $(k = 1, 2, \dots)$
  - Tính $\theta_k = (\alpha_k + \beta_k)/2$
  - Nếu thuật toán A tìm được $x_k \in D$ thỏa mãn $f(x_k) > \theta_k$ thì $\alpha_{k+1} = \theta_k, \beta_{k+1} = \beta_k$
  - Ngược lại, cập nhật $\alpha_{k+1} = \alpha_k, \beta_{k+1} = \theta_k$
  - Chuyển tiếp sang bước lặp $k + 1$ nếu $\beta_{k+1} - \alpha_k \geq \epsilon$
- ***Thuật toán tìm kiếm nhị phân trên có thời gian tính toán*** $\boldsymbol{O(log(\beta_0 - \alpha_0) \times n^c)}$

Ví dụ: Bài toán tìm tập độc lập cực đại

- ***Bài toán tối ưu***: Cho đồ thị vô hướng $G = (V, E)$, một tập con các đỉnh của đồ thị mà hai đỉnh bất kỳ trong nó là không kề nhau trên đồ thị được gọi là tập độc lập của đồ thị. Tìm tập độc lập với lực lượng lớn nhất.
- ***Bài toán quyết định***: Cho số nguyên dương k, hỏi đồ thị G có chứa hay không tập độc lập với lực lượng tối thiểu k.
- Thuật toán tìm kiếm nhị phân, $\alpha_0 = 1, \beta_0 = n$ trong đó n là số lượng đỉnh của đồ thị G. Nếu bài toán quyết định giải được bằng một thuật toán hiệu quả, thì bài toán tối ưu có thể giải được không quá $1 + \log(n)$ lần áp dụng thuật toán hiệu quả cho bài toán quyết định tương ứng.

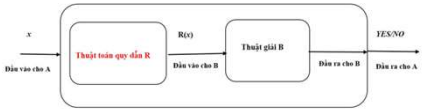
NỘI DUNG

- Giới thiệu
- Lớp bài toán P, NP, NPC
- Bài toán quyết định và Bài toán tối ưu
- **Phép quy dẫn**
- Chứng minh NP-đầy đủ
- Các hướng giải bài toán NP-khó

Phép quy dẫn

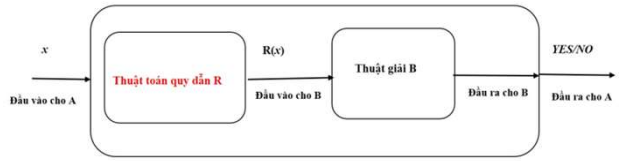
- Giả sử **A** và **B** là 2 bài toán quyết định. Ta nói, bài toán **A** có thể **quy dẫn** trong thời gian đa thức về bài toán **B** nếu tồn tại thuật toán ***R*** chạy trong thời gian đa thức cho phép biến đổi bộ dữ liệu đầu vào ***x*** của **A** thành bộ dữ liệu đầu vào ***R(x)*** của **B** sao cho ***x*** là bộ dữ liệu ***YES*** của **A** khi và chỉ khi ***R(x)*** là bộ dữ liệu ***YES*** của **B**.
- Phép quy dẫn trong thời gian đa thức được viết tắt là Phép quy dẫn.

Phép quy dẫn



- Kí hiệu **A** quy dẫn về **B: A** ≺ **B**
- Nếu tồn tại thuật toán đa thức giải **B** thì **A** cũng giải được trong thời gian đa thức.
- Ta có thể nói, **A** không khó hơn **B** hay **B** không dễ hơn **A.** Nếu bài toán **A** là khó (ví dụ, thuộc lớp NP-đầy đủ) thì bài toán **B** cũng khó.
- Phép quy dẫn là cách dễ dàng và thường áp dụng để chứng minh bài toán là khó.

Phép quy dẫn



- Tính chất bắc cầu: **C** ≺ **A** và **A** ≺ **B** thì **C** ≺ **B**

NỘI DUNG

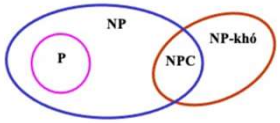
- Giới thiệu
- Lớp bài toán P, NP, NPC
- Bài toán quyết định và Bài toán tối ưu
- Phép quy dẫn
- **Chứng minh NP-đầy đủ, NP-khó**
- Các hướng giải bài toán NP-khó

NP-đầy đủ và NP-khó

- **Định nghĩa**: Một bài toán quyết định **A** được gọi là ***NP-đầy*** đủ khi
  1) **A** là một bài toán trong **NP**
  2) Mọi bài toán trong **NP** đều có thể quy dẫn về **A**
- Một bài toán chỉ thỏa mãn điều kiện (2) thì bài toán này được gọi là **NP-khó**.
- **Nhận xét**:
  - Khái niệm về bài toán khó trong lớp **NP** được xây dựng trên cơ sở phép quy dẫn;
  - Nếu tất cả các bài toán trong **NP** có thể quy dẫn về một bài toán **A**, thì **A** khó không kém bất cứ bài toán nào trong số chúng;
  - Chỉ với điều kiện (2), nếu tồn tại thuật toán đa thức để giải **A** thì sẽ kéo theo sự tồn tại thuật toán đa thức để giải mội bài toán trong **NP**.

Mối quan hệ giữa NP-khó và NP-đầy đủ

- Quan điểm của rất nhiều nhà nghiên cứu khoa học máy tính, và chưa chứng minh được như sau:

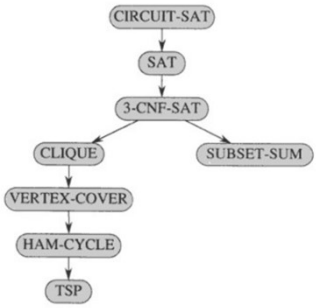


- Việc chứng minh bài toán **B** là NP-đầy đủ sử dụng phép quy dẫn là tìm ra một bài toán **A** thuộc NP-đầy đủ và phép quy dẫn A về B.
- **Bổ đề**: Nếu **B** thuộc NP-đầy đủ, **A** thuộc **NP**, và $\boldsymbol{A} \prec \boldsymbol{B}$ thì **A** là NP-đầy đủ.

Sơ đồ chứng minh bài toán L là NP-đầy đủ sử dụng phép quy dẫn

1. Chứng minh **L** thuộc NP (*Thường khá dễ để chứng minh*)
2. Chọn bài toán **L'** là NP-đầy đủ đã biết (*Thường dựa trên kinh nghiệm, và không dễ dàng*)
3. Xây dựng thuật toán đa thức *R* để biến đổi bộ dữ liệu của **L'** thành bộ dữ liệu **L** (*Thường không dễ dàng để chỉ ra*)
4. Chứng minh, *x* là bộ dữ liệu YES của **L'** khi và chỉ khi bộ dữ liệu *R(x)* tương ứng cũng là bộ YES của **L**.

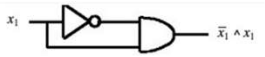
Cấu trúc cây quy dẫn các bài toán NP-đầy đủ cơ bản



- ***CIRCUIT-SAT*** (Circuit Satisfiability) là bài toán đầu tiên được chứng minh là NP-đầy đủ.
- ***CIRCUIT-SAT***: Cho một mạch logic với đầu vào gồm n giá trị, hỏi có tồn tại một đầu vào của mạch để đầu ra của mạch là TRUE, hay mạch luôn đưa ra FALSE.

CIRCUIT-SAT

- **CIRCUIT-SAT:** Cho một mạch logic với đầu vào gồm n giá trị, hỏi có tồn tại một đầu vào của mạch để đầu ra của mạch là TRUE, hay mạch luôn đưa ra FALSE.
- Ví dụ:

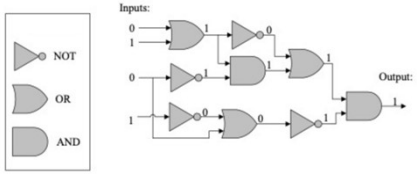


- Mạch không thể bật được với mọi giá trị của $x_1 \in \{0, 1\}$

- Mạch nhận giá trị TRUE với $(x_1, x_2) \in \{(0,1), (1,0), (1,1)\}$

CIRCUIT-SAT

- Định lý: **CIRCUIT-SAT** là NP-đầy đủ (Cook, 1971)
- Chứng minh:
  - **Bước 1**: **CIRCUIT-SAT** thuộc NP. Cho một đầu vào, dễ dàng tính được đầu ra tương ứng sau thời gian đa thức với số lượng biến nhờ sử dụng thuật toán duyệt đồ thị.



CIRCUIT-SAT

- Định lý: **CIRCUIT-SAT** là NP-đầy đủ (Cook, 1971)
- Chứng minh:
  - **Bước 2**: Mọi bài toán trong NP đều có thể quy dẫn được về **CIRCUIT-SAT**. Ý tưởng như sau:
    - Mọi bài toán trong NP đều có thể tính được nhờ một mạch logic
    - Mạch logic này có số thành phần giới hạn bởi đa thức theo số lượng biến và vì vậy cũng tính được trong thời gian đa thức.

Ví dụ quy dẫn HAM-CYCLE $\prec$ TSP

- Bài toán HAM-CYCLE: Hỏi đồ thị vô hướng G = (V, E) có chứa chu trình Hamilton hay không?
- Bài toán TSP:
  - *Bài toán tối ưu*: Tìm chu trình đi qua tất cả các đỉnh của đồ thị vô hướng G = (V, E, c) sao cho độ dài chu trình là ngắn nhất.
  - *Bài toán quyết định*: Tồn tại hay không một chu trình đi qua tất cả các đỉnh của đồ thị vô hướng G = (V, E, c) sao cho độ dài chu trình nhiều nhất k.
- Chứng minh bài toán quyết định TSP là NP-đầy đủ
  - Bài toán quyết định TSP thuộc lớp NP: Thủ tục kiểm tra lời giải có phải là chu trình chứa tất cả các đỉnh của đồ thị và độ dài không vượt quá k có thời gian tính toán đa thức.

Ví dụ quy dẫn HAM-CYCLE $\prec$ TSP

- Chứng minh bài toán quyết định TSP là NP-đầy đủ
  - Bài toán quyết định TSP thuộc lớp NP: Thủ tục kiểm tra lời giải có phải là chu trình chứa tất cả các đỉnh của đồ thị và độ dài không vượt quá k có thời gian tính toán đa thức.
  - Chỉ ra phép quy dẫn: Với mỗi bộ dữ liệu đầu vào của bài toán Ham-cycle $G = (V, E)$, ta xây dựng một bộ dữ liệu đầu vào của bài toán quyết định TSP, $G' = (V, E, c)$ và $k = 0$, với:
    - $E' = \{(u, v) | u, v \in V\}$
    - Hàm chi phí c được thiết kế như sau:
      - $c(u, v) = 1$ nếu $(u, v) \in E$
      - $c(u, v) = 0$ nếu $\notin E$
  - Ta thấy, nếu H là lời giải của bộ dữ liệu Ham-cycle thì H cũng là một lời giải của bộ dữ liệu của bài toán quyết định TSP. Ngược lại, nếu H' là một lời giải của bài toán quyết định TSP, thì H' cũng là một lời giải của bộ dữ liệu Ham-cycle.

NỘI DUNG

- Giới thiệu
- Lớp bài toán P, NP, NPC
- Bài toán quyết định và Bài toán tối ưu
- Phép quy dẫn
- Chứng minh NP-đầy đủ, NP-khó
- **Các hướng giải bài toán NP-đầy đủ, NP-khó**

Cách tiếp cận giải bài toán NP-đầy đủ và NP-khó

- Cách tiếp cận chính xác (exact/complete/determistic)
  - Tìm kiếm quy lui (Backtracking)
  - Nhánh cận (Branch-and-Bound)
  - Chia để trị (Divide-and-Conquer)
  - Quy hoạch động (Dynamic Programming)
  - Quy hoạch nguyên tuyến tính (Integer Programming)
  - Quy hoạch ràng buộc (Constraint Programming)
  - Sinh cột (Column Generation/ Branch-and-Price)
  - Kết hợp nhiều phương pháp

Cách tiếp cận giải bài toán NP-đầy đủ và NP-khó

- Gần đúng (Incomplete)
  - Tham lam (Greedy Algorithm)
  - Thuật toán xấp xỉ (Approximation Algorithm)
  - Các thuật toán Heuristics/Metaheuristics
    - Local search, Multistage heuristic
    - Tabu Search, Genetic Algorithm, Simulated Annealing, Practical Swarm, Ant Colony Optimization
  - Học tăng cường (Reinforcement Learning)
  - Học sâu (Deep Learning)

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !