Chương 7
Vấn đề NP-đầy đủ

1. Giải thuật thời gian đa thức tất định và không tất định
2. Vấn đề NP-đầy đủ
3. Định lý Cook
4. Một số bài toán NP-đầy đủ
5. Một số kỹ thuật để đối phó với những bài toán NP-đầy đủ

Tồn tại hay không tồn tại giải thuật hữu hiệu

- **Đối với nhiều bài toán chúng ta có những giải thuật hữu hiệu để giải.**
- **Tuy nhiên, có rất nhiều bài toán khác không có giải thuật hữu hiệu để giải.**
- **Và đối với một lớp khá lớn của những bài toán như vậy, chúng ta không thể nói có tồn tại giải thuật hữu hiệu để giải nó hay không.**

Những bài toán khó và những bài toán dễ

- **Nhiều nghiên cứu đã được thực hiện và có những cơ chế để phân loại những bài toán mới là "khó bằng" một số bài toán cũ đã biết.**
- **Tuy nhiên, đôi khi ranh giới giữa những bài toán "khó" và những bài toán "dễ" là khá tế nhị..**

Thí dụ:

- **Dễ: Có tồn tại một lối đi từ x đến y mà trong số nhỏ hơn M?**
- ***KHó*: Có tồn tại một lối đi từ x đến y mà trọng số ≥M?**

Bài toán 1 - BFS – thời gian tuyến tính

Bài toán 2 – thời gian hàm mũ

1. Giải thuật thời gian đa thức tất định và không tất định

P: Tập hợp tất cả những bài toán có thể giải được bằng những giải thuật tất định trong thời gian đa thức.

"Tất định" (*Deterministic*) : khi giải thuật đang làm gì, cũng chỉ có một việc duy nhất có thể được thực hiện kế tiếp. (whatever the algorithm is doing, there is only one thing it could do next).

Thí dụ: Xếp thứ tự bằng phương pháp chèn thuộc lớp P vì có độ phức tạp đa thức O(N^2)

Tính không tất định

Một cách để mở rộng quyền năng của máy tính là cho nó có năng lực làm việc không tất định (non-determinism).

***Không tất định*: khi một giải thuật gặp một sự lựa chọn giữa nhiều khả năng, nó có quyền năng "tiên đóan" để biết chọn một khả năng thích đáng.**

Giải thuật không tất định (nondeterministic algorithm)

Thí dụ: Cho A là một mảng số nguyên. Một giải thuật không tất định NSORT(A, n) sắp thứ tự các số theo thứ tự tăng và xuất chúng ra theo thứ tự này.

Thí dụ về một giải thuật không tất định

```
// An array B is used as temporary array.
procedure NSORT(A,n)
 // sort n positive integers //
begin
   for i:= 1 to n do B[i]:= 0;
  // guessing stage
   for i:= 1 to n do
   begin
      j := choice(1:n);
      if B[j] <> 0 then failure
      else B[j]:= A[i]
    end
// verification stage
   for i:= 1 to n-1 do
     if B[i] > B[i-1] then failure;
   print(B);
   success
end;
```

Hàm *choice(1:n)* có khả năng xác định một vị trí đúng trong tầm trị từ 1 đến *n*.

Thí dụ về một giải thuật không tất định (tt.)

Sự phân giải một giải thuật không tất định có thể được thực hiện bằng một sự song song hóa không hạn chế (*unbounded parallelism*).

Mỗi lần có bước lựa chọn phải thực hiện, giải thuật tạo ra nhiều bản sao của chính nó (*copies of itself*). Mỗi bản sao được thực hiện cho khả năng lựa chọn. Như vậy nhiều khả năng được thực hiện cùng một lúc.

- Bản sao đầu tiên kết thúc thành công thì làm kết thúc tất cả các quá trình tính tóan khác.

- Nếu một bản sao kết thúc thất bại thì chỉ bản sao ấy kết thúc mà thôi.

Giải thuật không tất định (tt.)

Thật ra, một máy tính không tất định không tạo ra những bản sao của giải thuật một khi phải thực hiện một lựa chọn.

Mà, nó có quyền năng chọn một yếu tố "đúng" từ một tập những khả năng lựa chọn mỗi phải thực hiện một lựa chọn.

Một yếu tố "đúng" được định nghĩa như là một chuỗi ngắn nhất của những lựa chọn (*shortest sequence of choices)* mà dẫn đến sự kết thúc thành công.

Trong trường hợp không tồn tại một chuỗi những lựa chọn mà dẫn đến sự kết thúc thành công ta giả định rằng giải thuật dừng và in ra thông báo *"tính toán không thành công"*.

Giải thuật không tất định (tt.)

Ghi chú:

- **Các thông báo *success* và *failure* là tương đương với phát biểu *stop* trong một giải thuật tất định.**
- **Độ phức tạp tính toán của NSORT là O(n).**

NP: tập hợp tất cả những bài toán mà có thể được giải bằng giải thuật không tất định trong thời gian đa thức.

Thí dụ : Bài toán có tồn tại lối đi dài nhất từ đỉnh *x* đến đỉnh *y* là thuộc lớp NP.

Bài toán thỏa mãn mạch logic (circuit satisfiability problem)

Cho một công thức logic có dạng

(x1 + x3 + x5)*(x1+ ~x2 + x4)*(~x3 + x4 +x5)*

(x2 + ~x3 + x5)

với các biến *xi* là các biến logic (*true* or *false*), "+" diễn tả OR, "*" diễn tả AND, và ~ diễn tả NOT.

Bài toán CSP là xác định xem có tồn tại một phép gán các trị logic vào các biến logic sao cho toàn công thức trở thành *true*.

CSP cũng là một bài toán NP.

Ghi chú: Lớp P là một tập con của lớp NP.

2. Vấn đề NP-đầy đủ

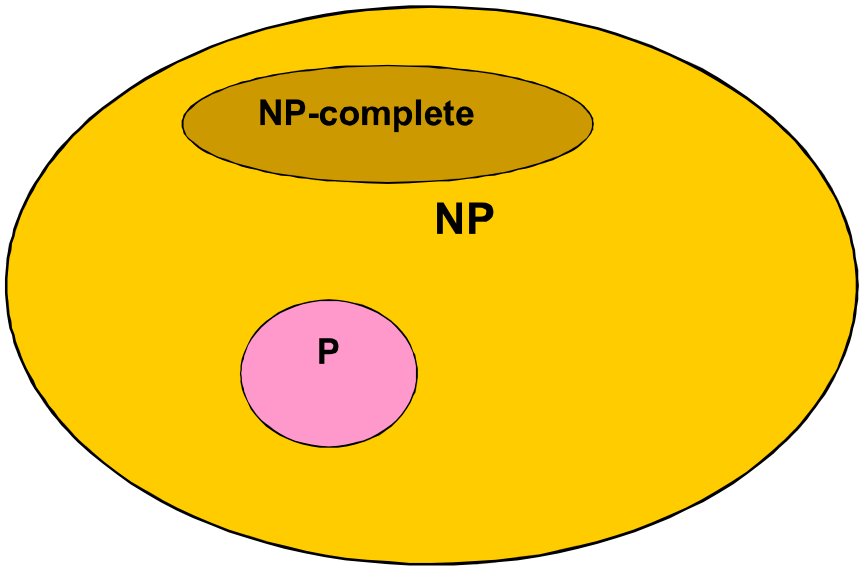
Có một danh sách những bài toán mà đã biết là thuộc về lớp NP nhưng không rõ có thể thuộc về lớp P hay không. (*Tức là, ta giải chúng dễ dàng trên một máy không tất định nhưng chưa ai có thể tìm thấy một giải thuật hữu hiệu chạy trên máy tính thông thường để giải bất kỳ một bài toán nào của chúng*).

Những bài toán NP này lại có thêm một tính chất nữa là:

"Nếu bất kỳ một trong những bài toán này có thể giải được trong thời gian đa thức thì tất cả những bài toán thuộc lớp NP cũng sẽ được giải trong thời gian đa thức trên một máy tất định."

Những bài toán như vậy được gọi là những bài toán NP-đầy đủ (*NP-complete*)

Hình 6.1



Tính khả thu giảm đa thức (Polynomial reducibility)

- **Lớp NP-đầy đủ là lớp con của những bài toán khó nhất trong lớp NP.**
- **Công cụ chính để chứng minh một bài toán thuộc loại NP-đầy đủ là ý tưởng về tính khả thu giảm đa thức (*polynomial reducibility*).**
- **Bất cứ giải thuật nào giải được bài toán mới thuộc loại NP có thể được dùng để giải một bài toán NP-đầy đủ nào đó đã biết bằng cách sau:**

  biến thể một thể hiện bất kỳ của bài toán NP-đầy đủ đã biết thành một thể hiện của bài toán mới, giải bài toán này bằng giải thuật đã có để tìm ra một lời giải, rồi biến thể lời giải này trở về thành một lời giải của bài toán NP-đầy đủ đã biết.

Tính khả thu giảm đa thức (tt.)

Để chứng minh một bài toán thuộc loại NP là NP-đầy đủ, ta chỉ cần chứng tỏ rằng một bài toán NP-đầy đủ đã biết nào đó thì khả thu giảm đa thức về bài toán mới ấy.

Định nghĩa: (*Thu giảm về*). Ta bảo bài toán L1 *thu giảm về* (*reduces to*) bài toán L2, ký hiệu là L1 α L2 nếu bất kỳ giải thuật nào giải được L2 thì cũng có thể được dùng để giải L1.

Tính khả thu giảm đa thức (tt.)

Để chứng minh một bài toán mới L là NP-đầy đủ, chúng ta cần chứng minh:

1. **Bài toán L thuộc lớp NP**
2. **Một bài toán NP-đầy đủ đã biết thu giảm về L.**

<u>Thí dụ:</u> Cho hai bài toán

- **Bài toán người thương gia du hành (TSP): cho một tập các thành phố và khoảng cách giữa mỗi cặp thành phố, tìm một lộ trình đi qua tất cả mọi thành phố sao cho tổng khoảng cách của lộ trình nhỏ hơn M.**
- **Bài toán chu trình Hamilton (HCP): Cho một đồ thị, tìm một chu trình đơn mà đi qua tất cả mọi đỉnh.**

Tính khả thu giảm đa thức (tt.)

Giả sử ta biết HCP là *NP-đầy đủ* và muốn xác định xem TSP cũng là NP-đầy đủ hay không. Bất kỳ giải thuật nào có thể được dùng để giải bài toán TSP cũng có thể được dùng để giải bài toán HCP, thông qua sự thu giảm sau:

Cho một thể hiện của bài toán HCP (một đồ thị), hãy tạo ra một thể hiện của bài toán TSP tương ứng như sau:

- **tạo ra các thành phố của bài toán TSP bằng cách dùng tập đỉnh trong đồ thị;**
- **về khoảng cách giữa hai cặp thành phố ta gán giá trị 1 nếu có tồn tại một cạnh giữa hai đỉnh tương ứng trong đồ thị và giá trị 2 nếu không có cạnh.**

Rồi thì dùng giải thuật giải TSP để tìm một lộ trình ≤ N (N là số đỉnh trong đồ thị).

Tính khả thu giảm đa thức (tt.)

Nghĩa là HCP *thu giảm về* TSP, như vậy tính chất NP-đầy đủ của HCP hàm ý tính chất tính chất NP-đầy đủ của TSP.

Sự thu giảm HCP về TSP là đơn giản vì hai bài toán có những nét tương tự nhau.

Sự thu giảm thời gian đa thức có thể sẽ rất phức tạp khi chúng ta liên kết những bài toán mà tương đối khác nhau.

Thí dụ: Có thể thu giảm bài toán thoả mãn mạch logic (CSP) về bài toán HCP.

3. Định lý Cook

Nhưng: Bài toán nào là bài toán NP-đầy đủ đầu tiên?

S.A. Cook (1971) đã đề xuất được một chứng minh trực tiếp đầu tiên rằng bài toán thỏa mãn mạch logic (CSP) là bài toán NP-đầy đủ.

"*Nếu tồn tại một giải thuật thời gian đa thức để giải bài toán thỏa mãn mạch logic thì tất cả mọi bài toán trong lớp NP có thể được giải trong thời gian đa thức.*"

Chứng minh của Cook rất phức tạp nhưng chủ yếu dựa vào máy Turing (Turing machine) tổng quát.

4. Một số bài toán NP-đầy đủ

Hàng nghìn bài toán khác nhau được biết là NP-đầy đủ. Danh sách này bắt đầu bằng bài toán thoả mãn mạch logic, bài toán người thương gia du hành (TSP) và bài toán chu trình Hamilton.

Một vài bài toán khác như sau:

- ***Bài toán phân hoạch số*: Cho một tập những số nguyên, có thể phân hoạch chúng thành hai tập con mà có tổng trị số bằng nhau?**
- ***Bài toán qui hoạch nguyên*: Cho một bài toán qui hoạch tuyến tính, liệu có tồn tại một lời giải toàn số nguyên?**

- ***Xếp lịch công việc trên đa bộ xử lý (multiprocessor scheduling)*: Cho một kỳ hạn (deadline) và một tập các công tác có chiều dài thời gian khác nhau phải được thực thi trên hai bộ xử lý. Vấn đề là có thể sắp xếp để thực thi tất cả những công tác đó sao cho thỏa mãn kỳ hạn không?**

- ***Bài toán phủ đỉnh (VERTEX COVER)*: Cho một đồ thị và một số nguyên N, có thể kiếm được một tập nhỏ hơn N đỉnh mà chạm hết mọi cạnh trong đồ thị ?**

- ***Bài toán xếp thùng* (*BIN PACKING*): cho *n* món đồ mà phải đặt vào trong các thùng có sức chứa bằng nhau *L*. Món đồ *i* đòi hỏi l_i đơn vị sức chứa của thùng. Mục đích là xác định số thùng ít nhất cần để chứa tất cả *n* món đồ đó.**

P ≠ NP ?

Những bài toán nêu trên và nhiều bài toán liên quan có những ứng dụng thực tế quan trọng.

Sự kiện không có những giải thuật tốt được tìm thấy cho bất kỳ bài toán nào trong số những bài toán nêu trên là một bằng chứng mạnh mẽ rằng P ≠ NP.

Dù cho P có khác NP hay không, một sự kiện thực tế là *chúng ta không có những giải thuật đảm bảo có thể giải bất kỳ một bài toán NP-đầy đủ nào một cách hữu hiệu.*

Một số kỹ thuật để đối phó với những bài toán NP-đầy đủ

1. Dùng "giải thuật xấp xỉ "(approximation algorithm) để tìm lời giải xấp xỉ tối ưu (near-optimal).

2. Dựa vào hiệu năng của trường hợp trung bình để phát triển một giải thuật mà tìm ra lời giải trong một số trường hợp nào đó, mặc dù không làm việc được trong mọi trường hợp.

3. Sử dụng những giải thuật có độ phức tạp hàm mũ nhưng hữu hiệu, ví dụ như giải thuật quay lui.

4. Đưa heuristic vào giải thuật để tăng thêm hiệu quả của giải thuật.

5. Sử dụng metaheuristic.

Heuristic và meta heuristic

- **Heuristic là tri thức về bài toán cụ thể được sử dụng để dẫn dắt quá trình tìm ra lời giải của giải thuật. Nhờ sự thêm vào các heuristic mà giải thuật trở nên hữu hiệu hơn.**
- **Meta heuristic là loại heuristic tổng quát có thể áp dụng cho nhiều lớp bài tóan.**
- **Gần đây meta heuristic là một lãnh vực nghiên cứu phát triển mạnh mẽ, với sự ra đời của nhiều meta heuristic như:**
  - **giải thuật di truyền (genetic algorithm)**
  - **giải thuật mô phỏng luyện kim (simulated annealing)**
  - **tìm kiếm tabu (Tabu search)**

  v.v….

Đóng góp của vấn đề NP-đầy đủ

Có nhiều bài toán NP-đầy đủ trong các lãnh vực

- **giải tích số (numerical analysis),**
- **sắp thứ tự và tìm kiếm,**
- **xử lý dòng ký tự (string processing),**
- **Mô hình hóa hình học (geometry modeling)**
- **xử lý đồ thị (graph processing).**

Sự đóng góp quan trọng nhất của lý thuyết về NP-đầy đủ là: ***nó cung cấp một cơ chế để xác định một bài toán mới trong các lãnh vực trên là "dễ" hay "khó".***

Bốn lớp bài toán phân theo độ khó

- ***Những bài toán bất khả quyết (Undecidable problems):*** **Đây là những bài toán chưa hề có giải thuật để giải.**

  Thí dụ: Bài toán quyết định xem một chương trình có dừng trên một máy Turing.

- ***Những bài toán khó giải (intractable)*** **: đây là những bài toán mà không tồn tại giải thuật thời gian đa thức để giải chúng. Chỉ tồn tại giải thuật thời gian hàm mũ để giải chúng.**
- ***Những bài toán NP-đầy đủ***

  Những bài toán NP-đầy đủ là một lớp con đặc biệt của lớp bài toán NP.
- ***Những bài toán P*****.**