

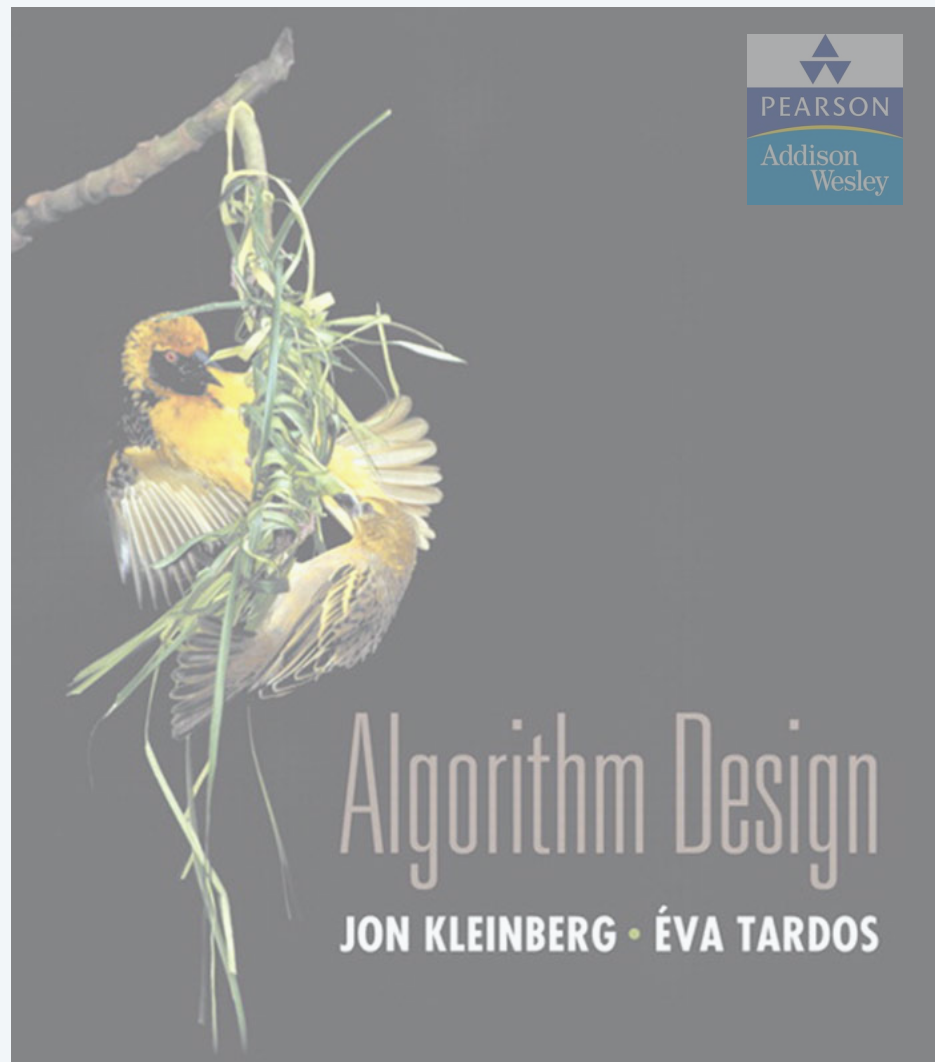
ĐỘ PHỨC TẠP TÍNH TOÁN

- ▶ *computational tractability*
- ▶ *bậc tăng tiệm cận*
- ▶ *khảo sát một số thời gian chạy thông dụng*
- ▶ *thời gian đa thức yếu, mạnh và giả đa thức*

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson–Addison Wesley

<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>



SECTION 2.1

ĐỘ PHỨC TẠP TÍNH TOÁN

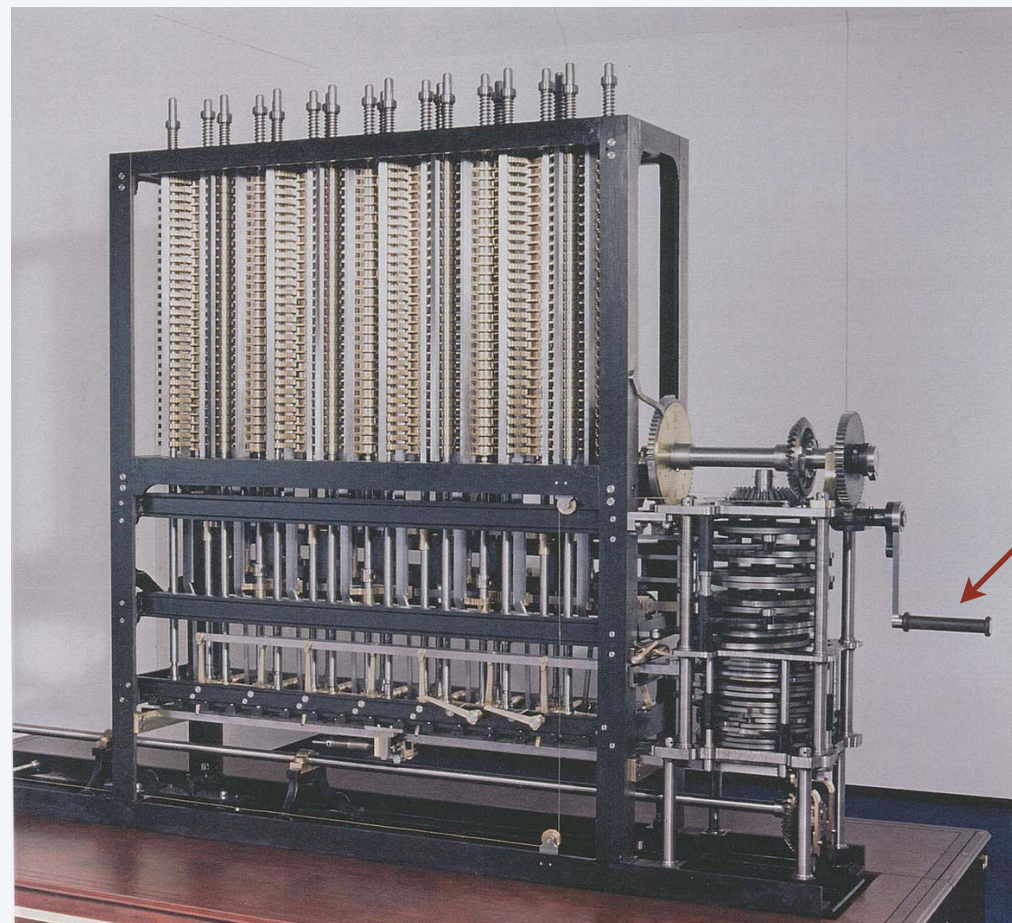
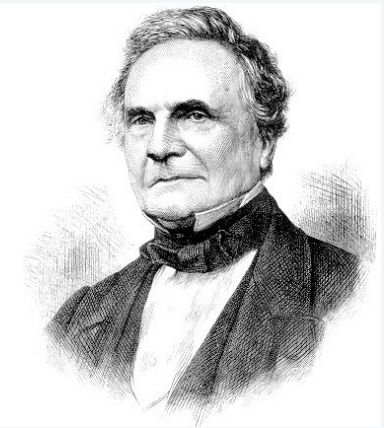
- ▶ *computational tractability*
- ▶ *bậc tăng tiệm cận*
- ▶ *khảo sát một số thời gian chạy thông dụng*
- ▶ *thời gian đa thức yếu, mạnh và giả đa thức*

Tractable problem

A problem that can be solved in theory (e.g. given large but finite resources, especially time), but for which in practice any solution takes too many resources to be useful, is known as an **intractable** problem. Conversely, a problem that can be solved in practice is called a **tractable** problem, literally "a problem that can be handled".

A strikingly modern thought

“ As soon as an Analytic Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will arise—By what course of calculation can these results be arrived at by the machine in the shortest time? ” — Charles Babbage (1864)

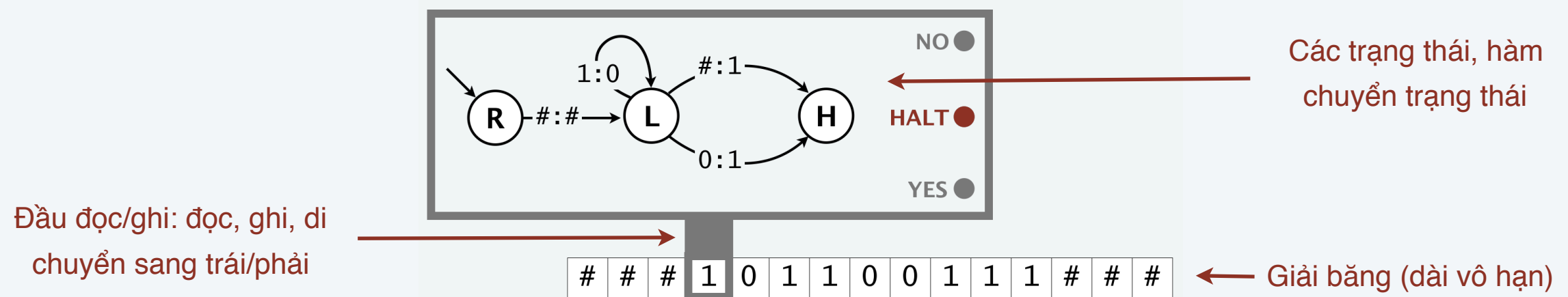


how many times do you have to turn the crank?

Analytic Engine

Mô hình tính toán: Máy Turing (Turing machines)

Máy Turing đơn định (deterministic Turing machine). Mô hình đơn giản và lý tưởng.



Các máy Turing được xây dựng dành cho các thí nghiệm tưởng tượng để tìm hiểu về các giới hạn của việc tính toán trên máy (không dành cho việc trực tiếp chế tạo ra máy tính).

Thời gian chạy. Số bước.

Bộ nhớ. Số lượng ô trên giải bằng được sử dụng.

Nhược điểm. No random access of memory.

- Máy Turing một giải bằng cần $\geq n^2$ bước để kiểm tra n -bit palindromes.
- Dễ dàng kiểm tra palindromes trong $\leq cn$ bước trên máy tính thật.

Các mô hình tính toán (computation models)

Đọc thêm mục 1.4 (Computational Models) trong sách

John Savage, *Models of Computation: Exploring the Power of Computing*,
Addison-Wesley, 1998

Link download: [http://cs.brown.edu/people/jsavage/book/pdfs/
ModelsOfComputation.pdf](http://cs.brown.edu/people/jsavage/book/pdfs/ModelsOfComputation.pdf)

Vét cạn (brute force)

Vét cạn. Với nhiều bài toán tồn tại các giải thuật tìm kiếm vét cạn kiểm tra mọi nghiệm có thể có.

- Thông thường cần 2^n bước (hoặc nhiều hơn) cho đầu vào có kích thước n .
- Đó là điều không thể chấp nhận trong thực tế.



Ví dụ. Vấn đề ghép cặp ổn định (hôn nhân bền vững): kiểm tra tất cả $n!$ cách ghép cặp và so sánh độ ổn định.

Thời gian chạy đa thức (polynomial running time)

Desirable scaling property. Khi kích thước đầu vào tăng gấp đôi, thuật toán nên chậm đi tối đa C lần với C là một hằng số.

Định nghĩa. Một thuật toán được gọi là có **thời gian đa thức (poly-time)** nếu tính chất trên được thỏa mãn, hay chính xác hơn

Tồn tại các hằng số $c > 0$ và $d > 0$ sao cho với mọi đầu vào (input) có kích thước n , thời gian chạy (running time) của thuật toán bị chặn bởi cn^d các bước tính toán nguyên thủy (primitive computational steps).



von Neumann
(1953)



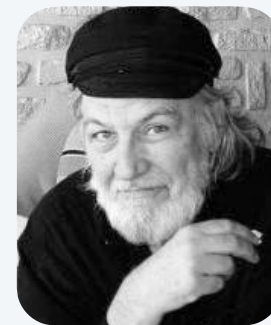
Nash
(1955)



Gödel
(1956)



Cobham
(1964)



Edmonds
(1965)



Rabin
(1966)

Thời gian chạy đa thức


Ta nói một thuật toán là **hữu hiệu (efficient)** nếu nó có thời gian chạy đa thức.

Trong thực tế.

- Nhiều thuật toán có thời gian đa thức được phát triển có cả hằng số lẫn số mũ nhỏ.
- Tốt hơn hẳn mức hàm mũ của thuật toán vét cạn, đạt được nhờ tận dụng những cấu trúc đặc biệt của bài toán.

Ngoại lệ. Một số thuật toán thời gian đa thức có hằng số hoặc số mũ siêu lớn, tức là chỉ mang ý nghĩa về mặt lý thuyết.

Q. Bạn thích cái nào hơn: $20 n^{120}$ hay $n^{1 + 0.02 \ln n}$?



Map graphs in polynomial time

Mikkel Thorup*
Department of Computer Science, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
mthorup@diku.dk

Abstract

Chen, Grigni, and Papadimitriou (WADS'97 and STOC'98) have introduced a modified notion of planarity, where two faces are considered adjacent if they share at least one point. The corresponding abstract graphs are called map graphs. Chen et al. raised the question of whether map graphs can be recognized in polynomial time. They showed that the decision problem is in NP and presented a polynomial time algorithm for the special case where we allow at most 4 faces to intersect in any point — if only 3 are allowed to intersect in a point, we get the usual planar graphs.

Chen et al. conjectured that map graphs can be recognized in polynomial time, and in this paper, their conjecture is settled affirmatively.

Tại sao điều đó quan trọng

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

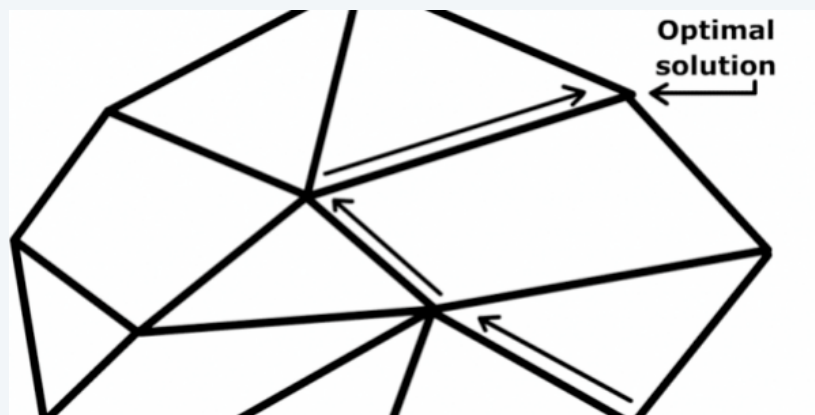
	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Đánh giá theo trường hợp xấu nhất (worst-case analysis)

Trường hợp xấu nhất. Thời gian chạy xét cho **mọi đầu vào** có kích thước n .

- Đánh giá thời gian chạy (cho trường hợp) xấu nhất.
- Nhìn chung phản ánh được hiệu quả của thuật toán trong thực tế.
- Cách đánh giá này tương đối ngặt, nhưng cũng khó tìm phương pháp thích hợp thay thế.

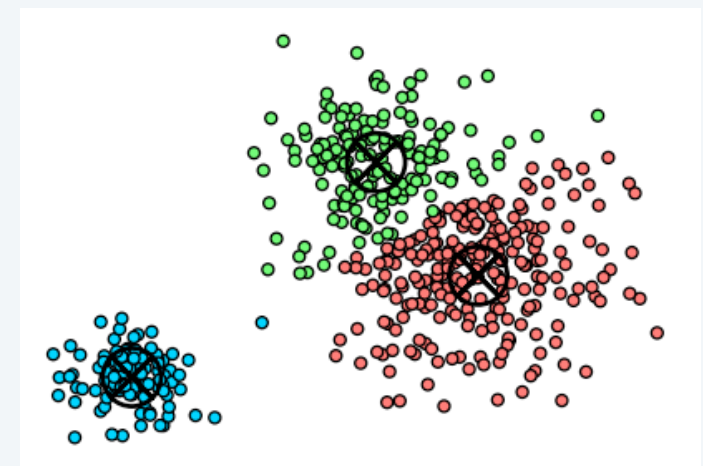
Ngoại lệ. Một số thuật toán có thời gian mũ vẫn được sử dụng rộng rãi trong thực tế vì các ví dụ xấu nhất thường không xuất hiện.



simplex algorithm



Linux grep



k-means algorithm

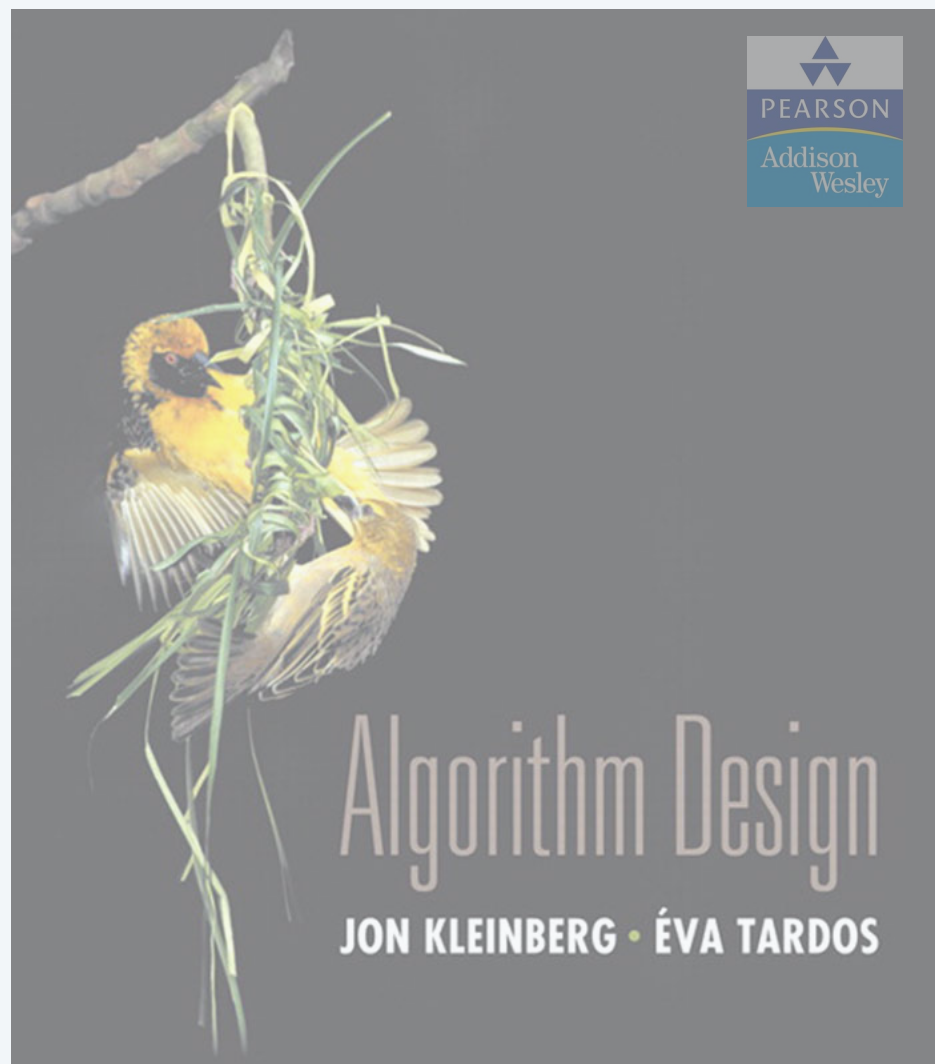
Các phương pháp đánh giá khác

Xác suất (probabilistic). Thời gian chạy **kỳ vọng (expected)** running time) của một thuật toán **ngẫu nhiên (randomized algorithm)**.



Ví dụ. thời gian chạy kỳ vọng của thuật toán quicksort cho n phần tử là $\sim 2n \ln n$.

Các ví dụ khác. Amortized analysis, average-case analysis, smoothed analysis, competitive analysis, ...



SECTION 2.2

ĐỘ PHỨC TẠP TÍNH TOÁN

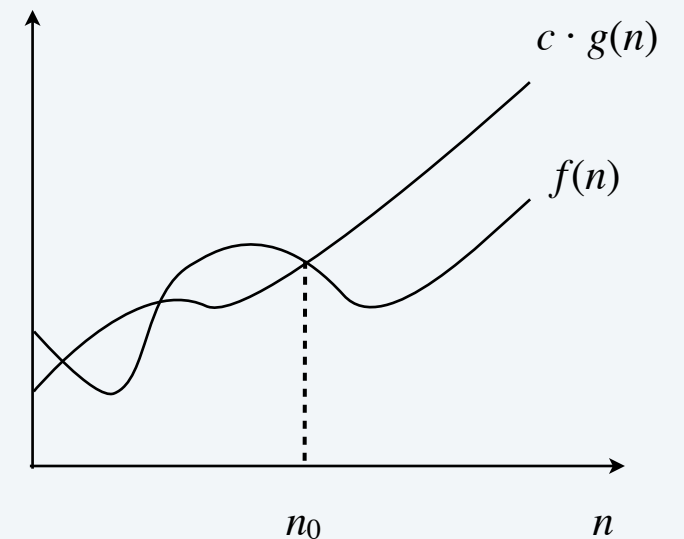
- ▶ *computational tractability*
- ▶ *bậc tăng tiệm cận (asymptotic order of growth)*
- ▶ *khảo sát một số thời gian chạy thông dụng*
- ▶ *thời gian đa thức yếu, mạnh và giả đa thức*

Kí hiệu O lớn (big O notation)

Cận trên. $f(n)$ là $O(g(n))$ nếu tồn tại các hằng số $c > 0$ và $n_0 \geq 0$ sao cho $0 \leq f(n) \leq c \cdot g(n)$ với mọi $n \geq n_0$.

Ví dụ. $f(n) = 32n^2 + 17n + 1$.

- $f(n)$ là $O(n^2)$. ← choose $c = 50, n_0 = 1$
- $f(n)$ không phải là $O(n)$ hay $O(n \log n)$.





Let $f(n) = 3n^2 + 17n \log_2 n + 1000$. Which of the following are true?

- A. $f(n)$ is $O(n^2)$.
- B. $f(n)$ is $O(n^3)$.
- C. Both A and B.
- D. Neither A nor B.

Kí hiệu O lớn: tính chất

Phản xạ. f là $O(f)$.

Hằng số. Nếu f là $O(g)$ và $c > 0$ thì cf là $O(g)$.

Tích. Nếu f_1 là $O(g_1)$ và f_2 là $O(g_2)$ thì $f_1 f_2$ là $O(g_1 g_2)$.

Chứng minh. Bài tập về nhà

Tổng. Nếu f_1 là $O(g_1)$ và f_2 là $O(g_2)$ thì $f_1 + f_2$ là $O(\max \{g_1, g_2\})$.

Bắc cầu. Nếu f là $O(g)$ và g là $O(h)$ thì f là $O(h)$.

Ví dụ. $f(n) = 5n^3 + 3n^2 + n + 1234$ is $O(n^3)$.



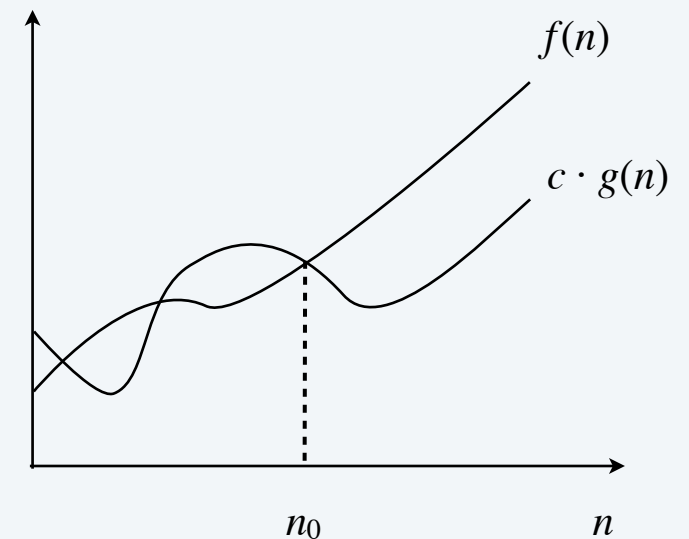
bỏ qua các bậc thấp

Kí hiệu Omega lớn

Cận dưới. $f(n)$ là $\Omega(g(n))$ nếu tồn tại các hằng số $c > 0$ và $n_0 \geq 0$ sao cho $f(n) \geq c \cdot g(n) \geq 0$ với mọi $n \geq n_0$.

Ex. $f(n) = 32n^2 + 17n + 1$.

- $f(n)$ là $\Omega(n^2)$ lẫn $\Omega(n)$.
- $f(n)$ không phải $\Omega(n^3)$.





Which is an equivalent definition of big Omega notation?

- A. $f(n)$ is $\Omega(g(n))$ iff $g(n)$ is $O(f(n))$.
- B. $f(n)$ is $\Omega(g(n))$ iff there exist constants $c > 0$ such that $f(n) \geq c \cdot g(n) \geq 0$ for infinitely many n .
- C. Both A and B.
- D. Neither A nor B.



Which is an equivalent definition of big Omega notation?

- A.** $f(n)$ is $\Omega(g(n))$ iff $g(n)$ is $O(f(n))$.
- B.** $f(n)$ is $\Omega(g(n))$ iff there exist constants $c > 0$ such that $f(n) \geq c \cdot g(n) \geq 0$ for infinitely many n . ← why not use this definition instead?
- C.** Both A and B.
- D.** Neither A nor B.

$f(n)$ is $\Omega(g(n))$ if there exist constants $c_1 > 0$ and $n_0 \geq 0$ such that $f(n) \geq c_1 \cdot g(n) \geq 0$ for all $n \geq n_0$

$g(n)$ is $O(f(n))$ if there exist constants $c_2 > 0$ and $n_0 \geq 0$ such that $0 \leq g(n) \leq c_2 \cdot f(n)$ for all $n \geq n_0$

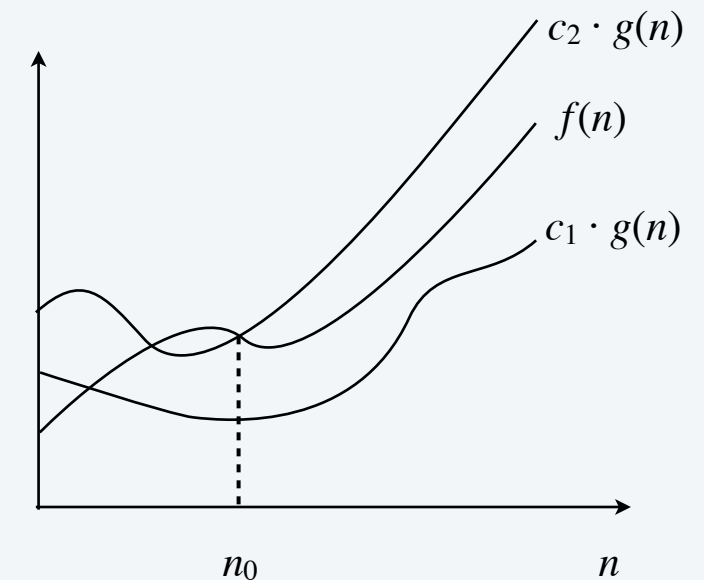
$$c_1 = 1 / c_2$$

Kí hiệu Theta lớn

Cận chặt. $f(n)$ là $\Theta(g(n))$ nếu tồn tại các hằng số $c_1 > 0$, $c_2 > 0$, và $n_0 \geq 0$ sao cho $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ với mọi $n \geq n_0$.

Ví dụ. $f(n) = 32n^2 + 17n + 1$.

- $f(n)$ là $\Theta(n^2)$. ← chọn $c_1 = 32, c_2 = 50, n_0 = 1$
- $f(n)$ không phải là $\Theta(n)$ hay $\Theta(n^3)$.





Which is an equivalent definition of big Theta notation?

- A. $f(n)$ is $\Theta(g(n))$ iff $f(n)$ is both $O(g(n))$ and $\Omega(g(n))$.
- B. $f(n)$ is $\Theta(g(n))$ iff $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$ for some constant $0 < c < \infty$.
- C. Both A and B.
- D. Neither A nor B.

Cận tiệm cận (asymptotic bounds) và giới hạn

Mệnh đề. Nếu $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c$ với hằng số $0 < c < \infty$ thì $f(n)$ là $\Theta(g(n))$.

Mệnh đề. Nếu $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ thì $f(n)$ là $O(g(n))$ nhưng không phải $\Omega(g(n))$.

Mệnh đề. Nếu $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$ thì $f(n)$ là $\Omega(g(n))$ nhưng không phải $O(g(n))$.

Chứng minh. Tự chứng minh


Cận tiệm cận của một số hàm thông dụng

Đa thức. Cho $f(n) = a_0 + a_1 n + \dots + a_d n^d$ với $a_d > 0$. Ta có $f(n)$ là $\Theta(n^d)$.

Chứng minh.

$$\lim_{n \rightarrow \infty} \frac{a_0 + a_1 n + \dots + a_d n^d}{n^d} = a_d > 0$$

Hàm lôga. $\log_a n$ là $\Theta(\log_b n)$ với mọi $a > 1$ và $b > 1$.

Chứng minh. $\frac{\log_a n}{\log_b n} = \frac{1}{\log_b a}$  cơ sở của hàm loga không quan trọng

Hàm lôga và đa thức. $\log_a n$ là $O(n^d)$ với mọi $a > 1$ và $d > 0$.

Chứng minh.

$$\lim_{n \rightarrow \infty} \frac{\log_a n}{n^d} = 0$$

Hàm mũ và đa thức. n^d là $O(r^n)$ với mọi $r > 1$ và $d > 0$.

Chứng minh.

$$\lim_{n \rightarrow \infty} \frac{n^d}{r^n} = 0$$

Giai thừa. $n!$ là $2^{\Theta(n \log n)}$.

Chứng minh. Công thức Stirling $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$

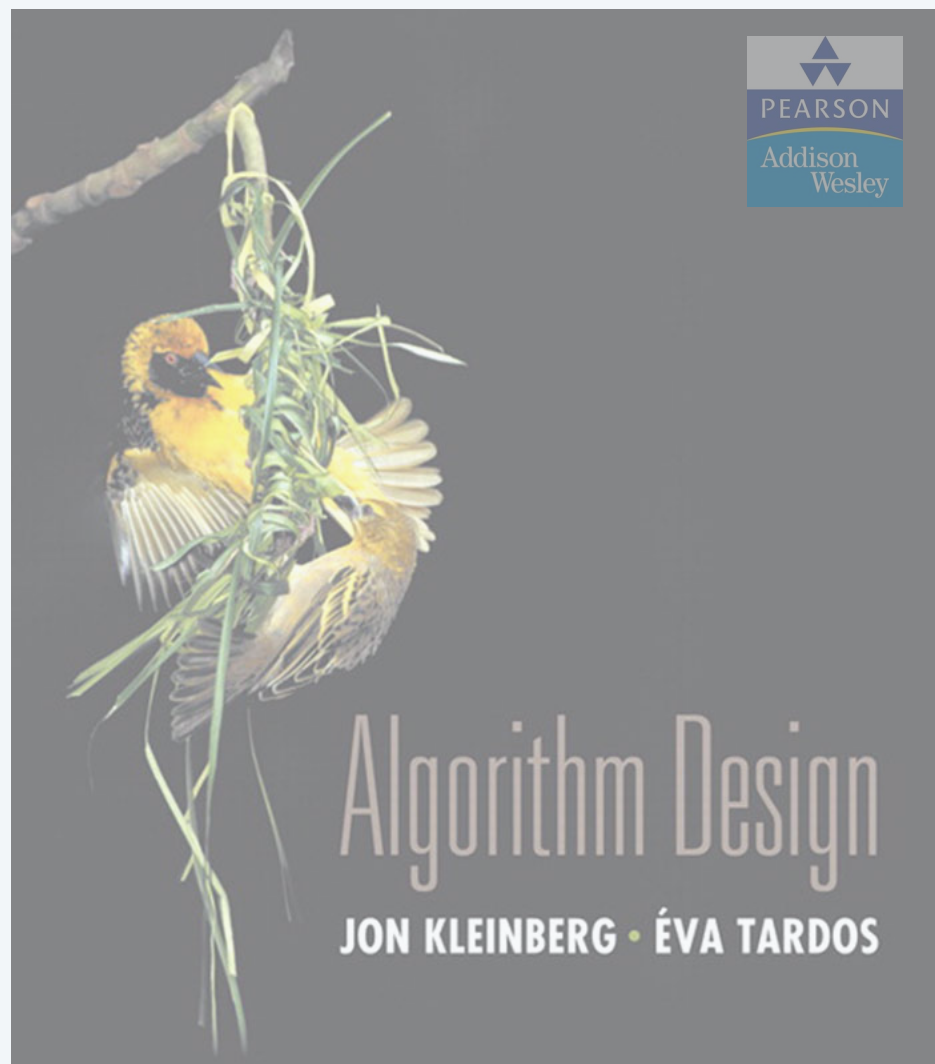
Công thức O lớn với nhiều biến

Cận trên. $f(m, n)$ là $O(g(m, n))$ nếu tồn tại các hằng số $c > 0$, $m_0 \geq 0$, và $n_0 \geq 0$ sao cho $f(m, n) \leq c \cdot g(m, n)$ với mọi $n \geq n_0$ và $m \geq m_0$.

Ví dụ. $f(m, n) = 32mn^2 + 17mn + 32n^3$.

- $f(m, n)$ là $O(mn^2 + n^3)$ và $O(mn^3)$.
- $f(m, n)$ không phải là $O(n^3)$ hay $O(mn^2)$.

Áp dụng. BFS (breadth-first search) có thời gian chạy $O(m + n)$ trên đồ thị có hướng với n đỉnh và m cạnh.



SECTION 2.4

ĐỘ PHỨC TẠP TÍNH TOÁN

- ▶ *computational tractability*
- ▶ *bậc tăng tiệm cận*
- ▶ *khảo sát một số thời gian chạy thông dụng*
- ▶ *thời gian đa thức yếu, mạnh và giả đa thức*

Thời gian hằng (constant time)

Thời gian hằng. Thời gian chạy là $O(1)$.

Ví dụ.

← bị chặn bởi một hằng số
không phụ thuộc vào kích thước đầu vào n

- Chia nhánh có điều kiện (if).
- Các phép toán số học/logic.
- Khai báo/khởi tạo một biến.
- Theo một liên kết trong một danh sách liên kết (linked list).
- Truy cập phần tử thứ i trong một mảng.
- So sánh/đổi hai phần tử trong một mảng.
- ...

Thời gian tuyến tính (linear time)

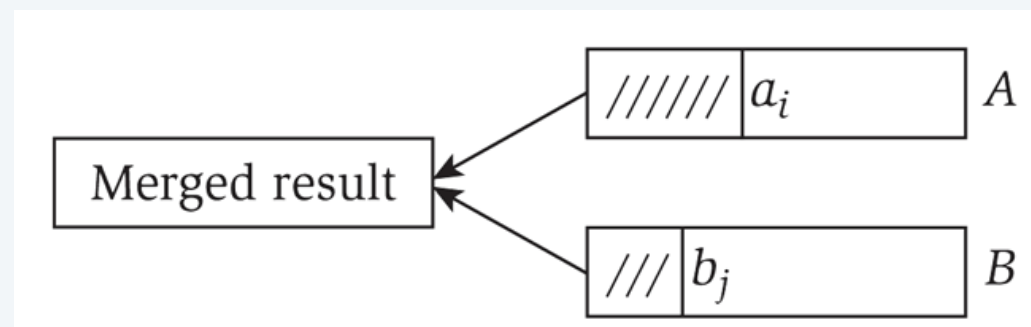
Thời gian tuyến tính. Thời gian chạy là $O(n)$.

Hợp nhất hai danh sách xếp thứ tự. Gộp hai danh sách được xếp thứ tự

$$A = a_1, a_2, \dots, a_n \text{ và } B = b_1, b_2, \dots, b_n$$

thành một danh sách được xếp thứ tự.

Thuật toán $O(n)$.



$i \leftarrow 1; j \leftarrow 1.$

WHILE (both lists are nonempty)

IF ($a_i \leq b_j$) append a_i to output list and increment i .

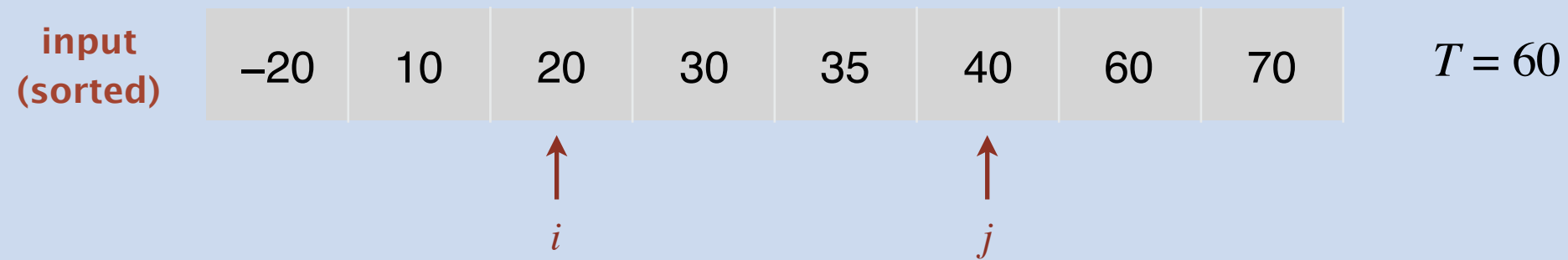
ELSE append b_j to output list and increment j .

Append remaining elements from nonempty list to output list.

TARGET SUM



TARGET-SUM. Cho một mảng được sắp thứ tự có n số nguyên khác nhau và một số nguyên T . Hãy tìm hai phần tử có tổng là T ?



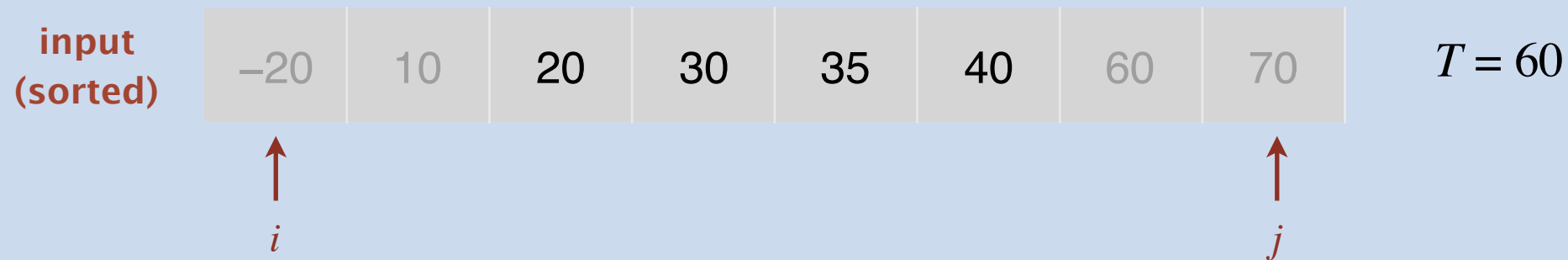
TARGET SUM



TARGET-SUM. Cho một mảng được sắp thứ tự có n số nguyên khác nhau và một số nguyên T . Hãy tìm hai phần tử có tổng là T ?

Thuật toán $O(n^2)$. Thử tất cả các cặp.

Thuật toán $O(n)$. Tận dụng cấu trúc sắp thứ tự.



Thời gian lôga (logarithmic time)

Thời gian lôga. Thời gian chạy là $O(\log n)$.

Tìm kiếm trong mảng được sắp thứ tự. Cho mảng A được sắp thứ tự gồm n số nguyên khác nhau và một số nguyên x . Hãy tìm chỉ số của x trong mảng.

Thuật toán $O(\log n)$. Tìm kiếm nhị phân.

- Sau k bước của vòng lặp WHILE, $(hi - lo + 1) \leq n / 2^k \Rightarrow k \leq 1 + \log_2 n$.

```
lo ← 1; hi ← n.
```

```
WHILE (lo ≤ hi)
```

```
    mid ←  $\lfloor (lo + hi) / 2 \rfloor$ .
```

```
    IF      ( $x < A[mid]$ ) hi ← mid - 1.
```

```
    ELSE IF ( $x > A[mid]$ ) lo ← mid + 1.
```

```
    ELSE RETURN mid.
```

```
RETURN -1.
```



Thời gian linearithmic

Thời gian linearithmic. Thời gian chạy là $O(n \log n)$.

Sắp xếp. Cho một mảng gồm n phần tử, hãy sắp xếp chúng theo thứ tự tăng dần.

Thuật toán $O(n \log n)$. Mergesort.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E
E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E
E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E
E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L
E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

LARGEST EMPTY INTERVAL



LARGEST-EMPTY-INTERVAL. Cho n mốc thời gian x_1, \dots, x_n tại đó có file gửi đến máy chủ. Khoảng thời gian dài nhất không có file nào xuất hiện là bao nhiêu?

LARGEST EMPTY INTERVAL



LARGEST-EMPTY-INTERVAL. Cho n mốc thời gian x_1, \dots, x_n tại đó có file gửi đến máy chủ. Khoảng thời gian dài nhất không có file nào xuất hiện là bao nhiêu?

Thuật toán $O(n \log n)$.

- Sắp xếp mảng a .
- Chạy theo thứ tự đó xác định khoảng chênh lệch lớn nhất giữa hai mốc thời gian liên tiếp.

Thời gian bậc hai (quadratic time)

Thời gian bậc hai. Thời gian chạy là $O(n^2)$.

Cặp điểm gần nhất. Cho 1 danh sách gồm n điểm trên mặt phẳng $(x_1, y_1), \dots, (x_n, y_n)$, tìm cặp điểm gần nhau nhất.

Thuật toán $O(n^2)$.

```
min  $\leftarrow \infty$ .  
FOR  $i = 1$  TO  $n$   
  FOR  $j = i + 1$  TO  $n$   
     $d \leftarrow (x_i - x_j)^2 + (y_i - y_j)^2$ .  
    IF ( $d < min$ )  
       $min \leftarrow d$ .
```

Ghi chú. Có vẻ như $\Omega(n^2)$ là không tránh khỏi, thực tế không phải vậy. [xem §5.4]

Thời gian bậc ba (cubic time)

Thời gian bậc ba. Thời gian chạy là $O(n^3)$.

3-SUM. Cho một mảng n số nguyên khác nhau, tìm ba số có tổng bằng 0.

Thuật toán $O(n^3)$. Xét các bộ ba số (với $i < j < k$).

```
FOR  $i = 1$  TO  $n$ 
  FOR  $j = i + 1$  TO  $n$ 
    FOR  $k = j + 1$  TO  $n$ 
      IF  $(a_i + a_j + a_k = 0)$ 
        RETURN  $(a_i, a_j, a_k)$ .
```

Ghi chú. Có thuật toán tốt hơn với thời gian $O(n^2)$. [slide tiếp theo]



3-SUM. Cho một mảng n số nguyên khác nhau, tìm ba số có tổng bằng 0.

Thuật toán $O(n^3)$. Xét các bộ ba.

Thuật toán $O(n^2)$.



3-SUM. Cho một mảng n số nguyên khác nhau, tìm ba số có tổng bằng 0.

Thuật toán $O(n^3)$. Xét các bộ ba.

Thuật toán $O(n^2)$.

- Sắp xếp mảng a .
- Với mỗi số nguyên a_i : giải TARGET-SUM cho mảng chứa mọi phần tử trừ a_i với tổng $T = -a_i$.

Thuật toán tốt nhất đã biết. $O(n^2 / (\log n / \log \log n))$.

Phỏng đoán. Không tồn tại thuật toán $O(n^{2-\epsilon})$ với bất kì $\epsilon > 0$ nào.

Thời gian đa thức (polynomial time)

Thời gian đa thức. Thời gian chạy là $O(n^k)$ cho hằng số $k > 0$ nào đó.

Tập độc lập (independent set) có kích thước k . Cho một đồ thị vô hướng, tìm k đỉnh sao cho giữa hai đỉnh bất kì không có cạnh nào.

 k là hằng số

Thuật toán $O(n^k)$. Xét tất cả tập con có k đỉnh.


FOREACH subset S of k nodes:

Check whether S is an independent set.

IF (S is an independent set)

RETURN S .

- Kiểm tra xem S có phải là tập độc lập kích thước k không mất thời gian $O(k^2)$.
- Số tập con có k phần tử là $\binom{n}{k} = \frac{n(n-1)(n-2) \times \cdots \times (n-k+1)}{k(k-1)(k-2) \times \cdots \times 1} \leq \frac{n^k}{k!}$
- $O(k^2 n^k / k!) = O(n^k)$.

 thời gian đa thức với $k = 17$,
nhưng nó không thực tiễn

Thời gian mũ (exponential time)

Thời gian mũ. Thời gian chạy là $O(2^{n^k})$ với hằng số $k > 0$ nào đó.

Tập độc lập. Cho đồ thị vô hướng, tìm tập độc lập có lực lượng lớn nhất.

Thuật toán $O(n^2 2^n)$. Xét tất cả các tập con.

$S^* \leftarrow \emptyset.$

FOREACH subset S of nodes:

Check whether S is an independent set.

IF (S is an independent set and $|S| > |S^*|$)
)

$S^* \leftarrow S.$

RETURN $S^*.$



Which is an equivalent definition of exponential time?

- A. $O(2^n)$
- B. $O(2^{cn})$ for some constant $c > 0$.
- C. Both A and B.
- D. Neither A nor B.

ĐỘ PHỨC TẠP TÍNH TOÁN

- ▶ *computational tractability*
- ▶ *bậc tăng tiệm cận*
- ▶ *khảo sát một số thời gian chạy thông dụng*
- ▶ *thời gian đa thức yếu, mạnh và giả đa thức*

Weakly, strongly, and pseudo polynomial time

Độ dài mã hóa (encoding length). Là số bit cần để biểu diễn trên máy tính (trong hệ nhị phân).

Độ dài mã hóa của một số tự nhiên n là $\log n$ (ngầm hiểu là $\log_2 n$).

Thời gian đa thức yếu (weakly polynomial time) = thời gian đa thức. Kích thước đầu vào bao gồm cả độ dài mã hóa.

Thời gian đa thức mạnh (strongly polynomial time). Thời gian đa thức nhưng không phụ thuộc vào độ dài mã hóa.

Thời gian giả đa thức (pseudo-polynomial time). Thời gian chạy là đa thức với giá trị của đầu vào.

Đọc chi tiết hơn, ví dụ, tại <https://disopt.epfl.ch/webdav/site/disopt/shared/IntPoints2009/algorithms.pdf>

Ví dụ

Thuật toán **sắp xếp** (merge sort, quick sort), tìm **cây bao trùm** (Prim, Kruskal), tìm **đường đi ngắn nhất** (Dijkstra, Bellman-Ford), ..., có thời gian đa thức mạnh. Số bước của các thuật toán sắp xếp không phụ thuộc vào các giá trị của các số hay trọng số của các cạnh.

Thuật toán điểm trong (interior point method) giải bài toán **quy hoạch tuyến tính** có thời gian đa thức (hay đa thức yếu - để phân biệt với đa thức mạnh).

Kiểm tra số nguyên tố. Kiểm tra xem một số tự nhiên k có phải số nguyên tố không.

Thuật toán 1. Kiểm tra các số nguyên trong khoảng $[2, k^{1/2}]$ xem k có chia hết cho số nào không. Thời gian chạy của nó là $O(k^{1/2})$, tức là không phải đa thức theo độ dài mã hóa $\log k$. Thuật toán này có thời gian giả đa thức.

Vào năm 2002, **Agrawal–Kayal–Saxena** đề xuất thuật toán có thời gian $(\log k)^{12}$, tức là thời gian đa thức (yếu).