

Xây dựng giải thuật

Duc-Minh Vu @ Phenikaa - ORLab

Lịch sử thuật toán

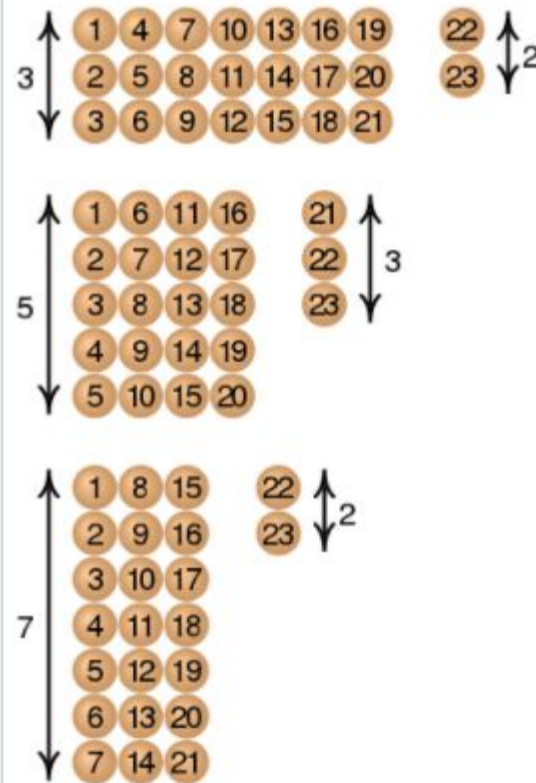
- Các thuật toán **số học** được sử dụng bởi các **nhà toán học Babylon** cổ đại vào khoảng 2500 TCN và các **nhà toán học Ai Cập** vào khoảng 1550 TCN.^[10]
- Các **nhà toán học Hy Lạp** sau đó đã sử dụng các thuật toán trong **sàng Eratosthenes** để tìm số nguyên tố,^[11] và **thuật toán Euclide** để tìm ước chung lớn nhất của hai số.^[12]
- Các **nhà toán học Ả Rập** như **al-Kindi** vào thế kỷ thứ 9 đã sử dụng **các** thuật toán **mật mã** để **phá mã**, dựa trên **phân tích tần số**.^[13]
- Từ *thuật toán* (*algorithm*) từ bắt nguồn từ nhà toán học thế kỷ thứ 9 **Muhammad ibn Mūsā al-Khwārizmī**, tên ông được Latinh hóa thành *Algoritmi*.^[14]


Châu Á

- [Định lý số dư Trung Quốc – Wikipedia tiếng Việt](#)
- Được biết đến đầu tiên bởi nhà toán học Trung Quốc Sun-tzu *Sun-tzu Suan-ching* từ thế kỉ thứ 3.

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \dots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

trong đó m_1, m_2, \dots, m_k đôi một nguyên tố cùng nhau.



Sun-tzu's original formulation: 

$x \equiv 2 \pmod{3} \equiv 3 \pmod{5} \equiv 2 \pmod{7}$
with the solution $x = 23 + 105k$,
with k an integer

Toán vs Thuật toán

Giải phương trình $x^2 + 2x + 2 = 0$.

Do $x^2 + 2x + 2 = (x + 1)^2 + 1 > 0 \forall x$ nên phương trình vô nghiệm.

$$\text{Đặt } \Delta = b^2 - 4ac$$

- Nếu $\Delta < 0$ thì phương trình vô nghiệm.
- Nếu $\Delta = 0$ thì phương trình có nghiệm kép $x_1 = x_2 = -\frac{b}{2a}$
- Nếu $\Delta > 0$ thì phương trình bậc 2 có hai nghiệm x_1, x_2

- [Phương trình bậc hai – Wikipedia tiếng Việt](#)

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a} = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Thuật toán là gì

- Tập hợp hữu hạn các thao tác được định nghĩa rõ ràng nhằm giải quyết một bài toán cụ thể nào đó.

Các tính chất

- Đầu vào (Input): Thuật toán nhận dữ liệu vào từ một tập nào đó.
- Đầu ra (Output): Với mỗi tập các dữ liệu đầu vào, thuật toán đưa ra các dữ liệu tương ứng với lời giải của bài toán.
- Chính xác: Các bước của thuật toán được mô tả chính xác.
- Hữu hạn: Thuật toán cần phải đưa được đầu ra sau một số hữu hạn (có thể rất lớn) bước với mọi đầu vào.
- Đơn trị: Các kết quả trung gian của từng bước thực hiện thuật toán được xác định một cách đơn trị và chỉ phụ thuộc vào đầu vào và các kết quả của các bước trước.
- Tổng quát: Thuật toán có thể áp dụng để giải mọi bài toán có dạng đã cho.

Mã giả

- Dùng để trình bày thuật toán

Một số ví dụ về thuật toán

- Tìm giá trị lớn nhất của một dãy số gồm N phần tử

1. Xác định bài toán

- Input: Số nguyên dương N và dãy N số nguyên a_1, \dots, a_N .
- Output: Giá trị lớn nhất Max của dãy số.

2. Thuật toán.

a/ Thuật toán giải bài toán này có thể được mô tả theo cách liệt kê như sau:

Bước 1. Nhập N và dãy a_1, \dots, a_N ;

Bước 2. $\text{Max} := a_1, i := 2$;

Bước 3. Nếu $i > N$ thì đưa ra giá trị Max rồi kết thúc;

Bước 4.

Bước 4.1. Nếu $a_i > \text{Max}$ thì $\text{Max} := a_i$;

Bước 4.2. $i := i + 1$ rồi quay lại bước 3;

Một số ví dụ về thuật toán

- [Thuật toán kiểm tra Số Nguyên Tố - O2 Education](#)

Giả sử cần kiểm tra số n có phải là số nguyên tố hay không thì các bước thực hiện như sau:

- **Bước 1:** Nhập vào n
- **Bước 2:** Kiểm tra nếu $n < 2$ thì kết luận n không phải là số nguyên tố
- **Bước 3:** Lặp từ 2 tới $(n-1)$, nếu trong khoảng này tồn tại số mà n chia hết thì kết luận n không phải là số nguyên tố, ngược lại n là số nguyên tố.

Thuật toán này chạy trong bao lâu ?

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  bool kiem_tra_nguyen_to(int n){
5.      bool nguyen_to = true;
6.      for(int i=2;i<n;i++)
7.          if (n%i == 0){
8.              nguyen_to = false;
9.              break;
10.         }
11.     return nguyen_to;
12. }
13. int main() {
14.     cout<<(kiem_tra_nguyen_to((int)1e9)?"Prime Number":"Composite Number")<<endl;
15.     cout<<(kiem_tra_nguyen_to((int)1e9 + 7)?"Prime Number":"Composite Number")<<endl;
16.     return 0;
17. }
```


- [THK6Ua - Online C++0x Compiler & Debugging Tool - Ideone.com](https://www.ideone.com/THK6Ua)

Ngạc nhiên chưa, nhanh đấy :D

Success #stdin #stdout 3.52s 5360KB

 stdin

Standard input is empty

 stdout

Composite Number

Prime Number

Thử với số này xem $2 \cdot (1e9) + 11$

 input  Output

Time limit exceeded #stdin #stdout 5s 5528KB

- Đợi chút, để em cải tiến thuật toán.


Thuật toán mới của em đây

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  bool kiem_tra_nguyen_to(int n){
5.      bool nguyen_to = true;
6.      for(int i=2;i<=n/2;i++)
7.          if (n%i == 0){
8.              nguyen_to = false;
9.              break;
10.         }
11.     return nguyen_to;
12. }
13. int main() {
14.     cout<<(kiem_tra_nguyen_to(2*(int)1e9 + 11)?"Prime Number":"Composite Number")<<endl;
15.     return 0;
16. }
```

Chạy rồi nhé

- 4s cũng không lâu lắm

Success #stdin #stdout 3.39s 5336KB

 stdin

Standard input is empty

 stdout


Prime Number

Chạy được với số lớn hơn 2 tỉ tí xíu chứ, 64 bit chẳng hạn?

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  bool kiem_tra_nguyen_to(long long n){
5.      bool nguyen_to = true;
6.      for(int i=2;i<=n/2;i++)
7.          if (n%i == 0){
8.              nguyen_to = false;
9.              break;
10.         }
11.     return nguyen_to;
12. }
13. int main() {
14.     cout<<(kiem_tra_nguyen_to(2*(long long)1e9 + 11)?"Prime Number":"Composite Number")<<endl;
15.     return 0;
16. }
```

Time limit exceed - Chạy quá thời gian cho phép

Time limit exceeded #stdin #stdout 5s 5516KB

 stdin

Standard input is empty

 stdout

Composite Number

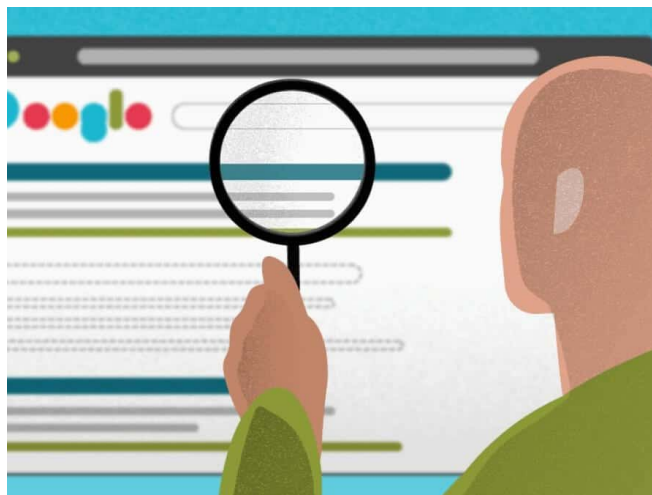
- Không phải lỗi tại em, do máy tính chạy chậm thôi ...

Ok, hơi bực mình chút

- Quay lại với kiểu int 32 bit.
- Ok, chạy cũng nhanh nhưng nếu tôi muốn kiểm tra tầm 10000 số ngẫu nhiên lên tới 9 chữ số thì sao.
- Tôi không muốn đợi khoảng $3 \cdot 10^4(s) \sim 8$ tiếng để có được hết kết quả.

Ok, đợi em đi học bổ túc về số học cơ bản đã

•



Định lý: nếu n là hợp số thì n có một ước nguyên tố nhỏ hơn \sqrt{n} .

Code mới đây anh

```
1.  #include <iostream>
2.  #include <cmath>
3.  using namespace std;
4.
5.  bool kiem_tra_nguyen_to(int n){
6.      bool nguyen_to = true;
7.      for(int i=2;i<sqrt(n);i++)
8.          if (n%i == 0){
9.              nguyen_to = false;
10.             break;
11.         }
12.      return nguyen_to;
13.  }
14.  int main() {
15.      for(int i=0; i<1e4;i++)
16.          cout<<kiem_tra_nguyen_to((1e9 + 7))<<endl;
17.      return 0;
18.  }
```

- [THK6Ua - Online C++0x Compiler & Debugging Tool - Ideone.com](https://www.ideone.com/THK6Ua)

Rất nhanh

```
Success #stdin #stdout 0.84s 5660KB
```

Số nguyên tố lớn nhất mà con người tìm được

- Số nguyên tố lớn nhất được biết đến hiện nay là một số nguyên tố Mersenne, có dạng $2^{82\,589\,933} - 1$. Nó gồm 24.862.048 chữ số khi viết trong hệ thập phân.
 - Tìm thấy qua một máy tính do Patrick Laroche của dự án Great Internet Mersenne Prime Search (**GIMPS** - Tìm kiếm số nguyên tố Mersenne khổng lồ trên internet) vào tháng 12/2018.
 - [Great Internet Mersenne Prime Search - PrimeNet](#)
 - Ba lập trình viên khác kiểm tra độc lập tính chính xác của kết quả.
- Họ làm như nào để tìm ra một số lớn như vậy
 - [GIMPS - The Math - PrimeNet \(mersenne.org\)](#)
 - [Mersenne Prime Discovery - \$2^{82589933}-1\$ is Prime!](#)
 - Nghĩa là các thuật toán mạnh hơn, hiệu quả hơn