**Project Euler**

**M Ụ C L Ụ C**

[Problem 1. Sum 3](#_Toc450637387)

[Problem 2. Even Fibonacci numbers 4](#_Toc450637388)

[Problem 3. Largest prime factor 6](#_Toc450637389)

[Problem 4. Largest palindrome product 8](#_Toc450637390)

[Problem 5. Smallest multiple 12](#_Toc450637391)

[Problem 6. Sum square difference 15](#_Toc450637392)

[Problem 7. 10001st prime 17](#_Toc450637393)

[Problem 8. Largest product in a series 20](#_Toc450637394)

[Problem 9. Special Pythagorean triplet 25](#_Toc450637395)

[Problem 10. Summation of primes 27](#_Toc450637396)

[Problem 11. Largest product in a grid 30](#_Toc450637397)

[Problem 12. Highly divisible triangular number 36](#_Toc450637398)

[Problem 13. Large sum 44](#_Toc450637399)

[Problem 14. Longest Collatz sequence 53](#_Toc450637400)

[Problem 15. Lattice paths 59](#_Toc450637401)

[Problem 16. Power digit sum 62](#_Toc450637402)

[Problem 17. Number letter counts 65](#_Toc450637403)

[Problem 18. Maximum sum in the triangle 72](#_Toc450637404)

[Problem 19. How many Sundays on the first of a month ? 75](#_Toc450637405)

[Problem 20. Factorial digit sum 84](#_Toc450637406)

[Problem 21. Amicable numbers 87](#_Toc450637407)

[Problem 22. Names scores 89](#_Toc450637408)

[Problem 23. Non-abundant sums 95](#_Toc450637409)

[Problem 24. Lexicographic permutations 99](#_Toc450637410)

[Problem 25. 1000-digit Fibonacci number 106](#_Toc450637411)

[Problem 26. Reciprocal cycles 108](#_Toc450637412)

[Problem 27. Quadratic primes 111](#_Toc450637413)

[Problem 28. Number spiral diagonals 116](#_Toc450637414)

[Problem 29. Distinct powers 119](#_Toc450637415)

[Problem 30. Digit fifth powers 122](#_Toc450637416)

## Problem 1. Sum

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

**Bài 1. Tổng**

Nếu liệt kê toàn bộ các số tự nhiên dưới 10 và là bội của 3 hoặc 5 thì ta thu được 3, 5, 6 và 9. Tổng của các số này là 23.

Hãy tìm tổng của toàn bộ các bội số dưới 1000 của 3 và 5.

***Hiểu đề***

Với hai số nguyên dương a và b, nếu a chia hết cho b thì ta gọi *a là bội số của b và b là ước số của a.* Yêu cầu của bài là tìm tổng của toàn bộ các số nguyên dương dưới 1000 chia hết cho 3 hoặc 5. Chú ý rằng số 15 chia hết đồng thời cho 3 và 5, như vậy 15 cũng là một ứng viên.

***Thuật toán***

Gọi sum là tổng cần tìm, ta có sum = T3+T5 – T15, trong đó

* T3 là tổng các số ≤ 999 và là bội của 3:

T3 = 3+6+…+3a = 3(1+2+…+a); a = (999 / 3) = 333;

* T5 là tổng các số ≤ 999 và là bội của 5:

T5 = 5+10+…+5b = 5(1+2+…+b); b = (999 / 5) = 199;

* T15 là tổng các số ≤ 999 và là bội của 3×5 = 15:

T15 = 15+30+…+15c = 15(1+2+…+c); c = (999 / 15) = 66.

*Giải thích*

Vì các bội của 15 đồng thời là bội của 3 và 5, như vậy mỗi bội số của 3 và mỗi bội số của 5 sẽ được tính 2 lần nên ta phải có phép trừ T15.

Ta lại biết công thức Gauss tính tổng của n số nguyên dương đầu tiên:

Vậy

sum = T3+T5 − T15 = 3\*G(999 / 3) + 5\*G(999 / 5) – 15\*G(999 / 15).

## Problem 2. [Even Fibonacci numbers](https://projecteuler.net/problem=2)

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

**Bài 2. Các số Fibonacci chẵn**

Mỗi số trong dãy Fibonacci là tổng của hai số sát trước số đó. Xuất phát từ hai số 1 và 2, mười số đầu tiên của dãy Fibonacci sẽ là:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Hãy tính tổng các số chẵn dưới bốn triệu trong dãy Fibonacci.

***Chú ý***

Bạn đừng nhầm lẫn số chẵn với số hạng có chỉ số chẵn. Trong dãy trên, nếu ta kí hiệu c1, c2,... là các số hạng của dãy Fibonacci, ta có

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | c10 |
| 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 |

các số hạng c2, c4, c6, c8 và c10 là những số hạng mang chỉ số chẵn, trong khi các số hạng c2 = 2, c5 = 8, c8 = 34 là những số chẵn.

## Problem 3. [Largest prime factor](https://projecteuler.net/problem=3)

The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143 ?

**Bài 3. Thừa số nguyên tố lớn nhất**

Số 13195 có các thừa số nguyên tố 5, 7, 13 và 29.

Hãy xác định thừa số nguyên tố lớn nhất của số 600851475143 ?

Hiểu đề

Cho hai số tự nhiên a, b > 1, nếu a chia hết cho *b,* tức là *a* = *kb*, thì *b* được gọi là *ước số của a.* Trong tích a = kb ta gọi k và b là *thừa số* hoặc nhân tử. Số nguyên tố là số nguyên dương có đúng hai ước, nói cách khác số tự nhiên p > 1 được gọi là *nguyên tố* nếu *p* chỉ chia hết cho 1 và chính nó. Lưu ý rằng số 1 không phải là số nguyên tố, Euclid gọi 1 là số *đặc biệt.* Như vậy tập số nguyên dương được chia thành ba tập con là

* Tập số đặc biệt chỉ gồm số 1, có duy nhất một ước;
* Tập các số nguyên tố, có đúng hai ước: 2, 3, 5, ...;
* Tập các *hợp số*, có trên hai ước: 4, 6, 8, ....

Trong tập các số nguyên tố thì 2 là số nguyên tố chẵn duy nhất. 2 cũng đồng thời là số nguyên tố nhỏ nhất. Các số nguyên tố còn lại đều là những số lẻ.

Thuật toán

Thuật toán dưới đây giải bài 3 với n tổng quát trong khoảng 2..60851475143. Thuật toán dựa trên ý tưởng của thuật toán phân tích số n ra thừa số nguyên tố với cơ sở lý thuyết là *Định lý cơ bản của số học*: *Mọi số tự nhiên n > 1 đều được viết dưới dạng tích duy nhất của các thừa số nguyên tố theo trật tự từ nhỏ đến lớn.*

Ta đã biết *n* là số nguyên tố khi và chỉ khi *n* không có ước nào trong khoảng 2 đến. Vậy khi xác định xem số tự nhiên *n* là nguyên tố hay không chỉ cần duyệt các thừa số *p* trong giới hạn trên 2 đến.

Chú ý rằng khi kết thúc quá trình duyệt, nếu *n* > 1 tức là *n* là số nguyên tố, đồng thời là ước nguyên tố cuối cùng của chính nó, ta phải ghi ước đó ra. Hãy chạy thử với *n* = 2\*17 = 34.

Ngoài ra, do 2 là số nguyên tố chẵn duy nhất do đó, sau khi xét *n* với *p* =2 ta chỉ cần xét các số *p* lẻ.

## ****Problem 4.**** [Largest palindrome product](https://projecteuler.net/problem=4)

A palindormic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009, which is 91 times 99 . Find the largest palindrome made from the product of two 3-digit numbers.

**Bài 4. Tích đối xứng lớn nhất**

Số đối xứng (còn được gọi là *số gánh*)là số đọc xuôi hay ngược đều như nhau. Trong các số có dạng tích của hai số hai chữ số thì 9009 là số lớn nhất, cụ thể là 9009 = 91 × 99.

Hãy tìm số đối xứng lớn nhất đồng thời là tích của hai số ba chữ số.

Thuật toán

*Phương án E4A*

Các số tự nhiên 3 chữ số nằm trong khoảng 100..999. Với mỗi cặp số a và b từ 999 ngược đến 100, ta tìm tích c = ab rồi kiểm tra c có phải là số gánh (hay số đốii xứng, tức là c viết ngược trùng với c) thì ghi nhận kết quả max.

Ta cải tiến thuật toán bằng các kĩ thuật sau:

K1. Hạn chế phép nhân là phép toán tốn kém hơn các phép cộng và trừ: thay phép nhân bằng các phép cộng hoặc/và trừ.

Giả sử tại bước i ta đã tính được c = ab. Do ta duyệt ngược, nên đến bước i+1 tiếp theo ta cần tính a(b-1). Ta có a(b-1) = ab – a = c − a. Vậy ở bước này ta thay phép tính c = a(b-1) bằng lệnh c = c − a. Tương tự, nếu c = ab thì khi cần tính (a−1)b ta thay bằng lệnh c = c − b.

Muốn kiểm tra số c có đối xứng không ta gọi hàm Inv(c) để cho ra số lật c’ của c. Thí dụ, Inv(1234) = 4321. Như vậy c đối xứng khi và chỉ khi Inv(c) = c.

*Phương án E3B* (nguồn: MathBlog)

Ta tự sinh ra các số đối xứng s bằng cách lấy số x ghép với số lật của nó là x’. Thí dụ, x = 102 cho ta x’ = 201 và do đó s = xx’ = 102201. Giả sử s là số đối xứng, ta kiểm tra xem s có là tích của 2 số 3 chữ số hay không, tức là xem có tồn tại 2 số a, b thỏa các điều kiện sau:

* (i) s = a\*b;
* (ii) len(a) = 3; len(b) = 3, trong đó len(x) là số chữ số của x

Muốn có điều kiện (ii) ta chỉ cần xét các số a và b trong khoảng 999 giảm đến 100. Ta kiểm tra điều kiện (i). Với mỗi số a trong khoảng 999..100 ta tính thương b = s/a. Nếu b là số 3 chữ số, tức là b nằm trong khoảng 999..100 và a\*b = s thì a và b sẽ là ứng viên. Lưu ý rằng phép chia nguyên b = s/a có thể không cho ta a\*b = s. Cách này sẽ giảm 2 vòng lặp (cho a và b) xuống còn 1 vòng lặp (cho a, còn b thì được tính theo s và a). Số đối xứng lớn nhất được sinh ra là s = xx’ = 999999 với x = 999. Mặt khác khi a = b = 999 (là các số 3 chữ số lớn nhất) thì a\*b = 998001 không phải là số đối xứng. Vậy số đối xứng lớn nhất chỉ có thể là 997799 đựơc tạo bởi x = 997. Ta sẽ xuất phát từ số này. Cận trên của a là 999. Vậy cận dưới của a là bao nhiêu? Vì ta đòi hỏi a\*b = s nên a phải chọn sao cho b cũng là số 3 chữ số. Do a và b có thể hoán đổi vị trí cho nhau nên ta có thể đặt điều kiện a ≥ b. Từ đây suy ra cận dưới của a chính là .

Phương án này chạy nhanh gấp đôi E4A.

Hàm Inv(x) cho ra số lật x’ của số x. Ta lại muốn hàm này có thêm chức năng là tạo ra số đối xứng xx’ từ số x nên ta sẽ sử dụng 2 tham biến, cụ thể là

Inv(x, a) sẽ lật x và ghép vào cuối a.

Thí dụ,

Inv(456, 123) = 123654

Inv(456, 0) = 654

Inv(456) = 654

Inv(123,123) = 123321

Ta mặc định, khi không viết tham biến a thì a = 0.

## Problem 5. [Smallest multiple](https://projecteuler.net/problem=5)

2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

What is the smallest positive number that is evenly divisible (divisible with no remainder) by all of the numbers from 1 to 20?

**Bài 5. Bội nhỏ nhất**

2520 là số nhỏ nhất chia hết đồng thời cho các số từ 1 đến 10. Số nhỏ nhất chia hết đồng thời cho các số từ 1 đến 20 là số nào?

Đáp án: 232792560

Thuật toán

Bội chung nhỏ nhất của các số là số nhỏ nhất chia hết đồng thời cho các số đó. Trong các sách toán, Bcnn của các số a, b, ... thường được kí hiệu là [a, b, ...].

Đáp số = [1, 2, …, 20].

Bcnn của các số thỏa các tính chất sau:

* [a, b] = a\*(b / (a,b)): Bcnn của a và b là tích của a, b chia cho Ucln của chúng.
* [a, b] = [b, a]
* [a, 1] = a
* [a] = a.
* [a, b, c] = [ [a,b], c ].

Ucln của các số thỏa các tính chất sau:

* (a, b): Ucln của a và b ước số lớn nhất của chúng.
* (a, b) = (b, a).
* (a, 0) = a;
* (a) = a;
* (a, b) = (a mod b, b): Ucln của 2 số không thay đổi nếu ta thay một trong 2 số bằng số dư của số đó chia cho số thứ hai.
* (a, b, c) = ( (a, b), c ).

Ta dùng thuật toán Euclid để tính Ucln của hai số.

Có thể cải tiến chút ít nhờ nhận xét sau:

Nếu a là ước số của b thì (a, b) = a và [a, b] = b. Thí dụ,

(2, 20) = 2; [2, 10] = 20; (3, 15) = 3; [3, 15] = 15.

Vậy để tính Bcnn của dãy số 1..20 ta có thể giữ lại số x và gạch đi các ước của x. Ta nhận được 10 phần tử như sau:

int a[11] = {1, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};

Khai báo trên cho ta mảng a gồm 11 phần tử có số hiệu từ 0 đến 10.

Xem hàm E5B.

## Problem 6. [Sum square difference](https://projecteuler.net/problem=6)

**The sum of the squares of the first ten natural numbers is,**

**12 + 22 + … + 102 = 385**

**The square of the sum of the first ten natural numbers is,**

**(1 + 2 + … + 10)2 = 552 = 3025**

**Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is 3025 – 385 = 2640.**

**Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.**

**Bài 6. Hiệu các bình phương**

**Tổng bình phương của 10 số tự nhiên đầu tiên là**

**T(10) = 12 + 22 + … + 102 = 385**

**Bình phương của tổng của 10 số tự nhiên đầu tiên là**

**B(10) = (1 + 2 + … + 10)2 = 552 = 3025**

Hiệu B(10) – T(10) = 3025 – 385 = 2640.

Hãy tính B(100) – T(100).

Thuật toán

Hai phương án

E6A: Tính thẳng

Với i = 1 .. 100 tính đồng thời tổng các bình phương t và tổng b.

Cuối cùng lấy hiệu b\*b – t.

E6B: Dùng công thức

Tổng n số tự nhiên đầu tiên (*Gauss*)

Tổng bình phương của n số tự nhiên đầu tiên (*Chương trình Toán phổ thông, hoặc xem trang web của Ken Ward*)

**Bình luận**

Bạn còn có thể biến đổi tiếp hai công thức trên để tính theo phương án thứ ba E6C như sau:

## Problem 7. [10001st prime](https://projecteuler.net/problem=7)

**By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13.**

**What is the 10001st prime number?**

**Bài 7. Số nguyên tố thứ mười ngàn lẻ một**

**Khi liệt kê sáu số nguyên tố đầu tiên: 2, 3, 5, 7, 11, và 13, chúng ta thấy số nguyên tố thứ sáu là 13.**

**Số nguyên tố thứ 10001 là số nào ?**

**Đáp án: 104743**

Thuật toán

Ta sinh dần các số nguyên tố và ghi vào mảng a gồm 10002 phần tử. Ta nên khởi trị a với 10 số nguyên tố đầu tiên là những số rất dễ được nhận biết:

int a[MN] = { 0, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };

**Tiếp đến ta dựa vào định lí sau:**

**Định lí. *Số nguyên p > 1 là nguyên tố khi và chỉ khi p không có ước nguyên tố nào trong khoảng 2...***

**Giả sử ta đã sinh được k số nguyên tố a[1], a[2], ..., a[k]. Giả sử trong dãy k số nguyên tố này ta có số a[i] đầu tiên thỏa tính chất a[i]2 > a[k]. Khi đó, để kiểm tra các số p trong khoảng từ a[k] + 1 đến a[i]2 có là số nguyên tố hay không, theo định lí trên ta chỉ cần xét xem p có ước nào trong số a[1] đến a[k] hay không. Nếu có thì p là hợp số, nếu không thì p là số nguyên tố.**

**Ta vận dụng thêm vài nhận xét sau đây:**

**Nếu p > 3 và p lẻ thì ta có thể kiểm tra các số nguyên tố trong khoảng *3...***

**Với k > 1 thì a[k] lẻ do đó chỉ cần xét các số lẻ p trong khoảng a[k]+2 đến a[i]2 – 2 vì bản thân a[i]2 là hợp số lẻ. Cũng vì p lẻ nên ta cũng không cần xét p có ước nguyên tố a[1] = 2 hay không.**

**Giả sử ta có dãy a với 10 số nguyên tố như trên:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **...** |
| **a[i]** | **2** | **3** | **5** | **7** | **11** | **13** | **17** | **19** | **23** | **29** | **?** |  |
| ***n = 4***  ***có thể kiểm tra các số nguyên tố từ***  ***29+2 đến a[4]\*a[4] = 49*** | | | | | | | | | | | | |

**Ta thấy a[4] = 7, 7\*7 = 49. Vậy để tìm các số nguyên tố tiếp theo, sau số nguyên tố thứ 10 là a[10] = 29 ta chỉ cần duyệt p từ 31 đến 49-2 = 47 (vì bản thân 49 là số chính phương nên không phải nguyên tố). Duyệt như trên bạn sẽ thu được đến số nguyên tố thứ 15 là 47.**

**a[11] = 31; a[12] = 37; a[13] = 41; a[14] = 43; a[15] = 47**

**Tiếp đến, bạn chọn n = 5 để xét a[5]\*a[5] = 11\*11 = 121 và kiểm tra các số trong khoảng từ a[15]+2 = 47+2 = 49 đến 121-2 = 119 để chọn ra các số nguyên tố rồi ghi vào mảng a.**

**Để kiểm tra số p trong khoảng trên có là nguyên tố không ta dựa vào định lí trên nên chỉ cần xét các số nguyên tố (có sẵn trong a) từ a[2] .. a[n]. Thí dụ muốn xem các số p, 49 ≤ p ≤ 119 có nguyên tố hay không ta chỉ cần xem p có chia hết cho a[2] = 3, a[3] = 5 và a[5] = 7 là đủ.**

## Problem 8. [Largest product in a series](https://projecteuler.net/problem=8)

The four adjacent digits in the 1000-digit number that have the greatest product are 9 × 9 × 8 × 9 = 5832.

73167176531330624919225119674426574742355349194934  
96983520312774506326239578318016984801869478851843  
85861560789112949495459501737958331952853208805511  
12540698747158523863050715693290963295227443043557  
66896648950445244523161731856403098711121722383113  
62229893423380308135336276614282806444486645238749  
30358907296290491560440772390713810515859307960866  
70172427121883998797908792274921901699720888093776  
65727333001053367881220235421809751254540594752243  
52584907711670556013604839586446706324415722155397  
53697817977846174064955149290862569321978468622482  
83972241375657056057490261407972968652414535100474  
821663704844031**9989**0008895243450658541227588666881  
16427171479924442928230863465674813919123162824586  
17866458359124566529476545682848912883142607690042  
24219022671055626321111109370544217506941658960408  
07198403850962455444362981230987879927244284909188  
84580156166097919133875499200524063689912560717606  
05886116467109405077541002256983155200055935729725  
71636269561882670428252483600823257530420752963450

Find the thirteen adjacent digits in the 1000-digit number that have the greatest product. What is the value of this product?

**Bài 8**

Bảng trên là số tự nhiên 1000 chữ số được viết trên 20 dòng, mỗi dòng 50 chữ số. Trong bảng, có thể tìm được 4 chữ số liên tiếp (**in đậm**) có tích lớn nhất là 9989, 9 × 9 × 8 × 9 = 5832.

Hãy tìm 13 chữ số liên tiếp có tích lớn nhất. Tích này bằng bao nhiêu?

Thuật toán

Ta giả thiết dữ liệu vào được ghi trong mảng string hai chiều ss, gồm 20 dòng, mỗi dòng 20 chữ số. Việc chuyển từ mảng hai chiều ss sang mảng string một chiều s là đơn giản.

Ta sẽ viết thuật toán tổng quát E8(k) tìm tích lớn nhất của k chữ số liền kề trong dãy chữ số s. Kết quả của bài toán khi đó sẽ là E8(13).

Ý tưởng đầu tiên là vận dụng kĩ thuật cửa sổ trượt. Với mỗi vị trí i trong dãy s ta gọi hàm Product(i, i+k-1) để tính tích của k chữ số liền kề trong dãy s, kể từ chữ số thứ i đến chữ số i+k-1 và ghi nhận giá trị max của hàm này.

Điều đáng quan tâm là nếu trong đoạn i..i+k-1 có chứa chữ số 0 thì tích sẽ bằng 0, do đó, nếu biết trước các vị trí chứa số 0 trong s thì chắc chắn là ta có thể bỏ qua việc tìm tích các chữ số của các đoạn có chứa chữ số 0. Thủ tục FixZero đảm nhận nhiệm vụ duyệt trước s và ghi nhận các vị trí i thỏa điều kiện s[i] = ‘0’ vào mảng zero: zero[j] sẽ cho biết vị trí của chữ số 0 thứ j trong s. Khi đó, nếu i là vị trí xuất hiện chữ số 0 và j là vị trí xuất hiện chữ số 0 tiếp theo và j − i − 1 ≥ k thì trong đoạn s[i+1..j−1] sẽ chứa toàn chữ số khác 0, và ta có thể duyệt đoạn này để tìm tích max của các đoạn con chứa đúng k chữ số. Đó là nhiệm vụ của hàm Segment(i+1, j – 1), j – i – 1 ≥ k.

Chú thích

Pchar là kiểu string có chiều dài có thể vượt quá 255 của Free Pascal.

## Problem 9. [Special Pythagorean triplet](https://projecteuler.net/problem=9)

A Pythagorean triplet is a set of three positive integes (a, b, c), for which,

a2 + b2 = c2.

For example, 32 + 42 = 52 (9 + 16 = 25).

There exists exactly one Pythagorean triplet for which a + b + c = 1000. Find the product abc.

**Bài 9. Bộ ba Pythagor đặc biệt**

Bộ ba Pythagor là tập ba số nguyên dương (a, b, c) thỏa điều kiện

a2 + b2 = c2.

Thí dụ, bộ ba Pythagor (3, 4, 5) cho ta 32 + 42 = 52 (9 + 16 = 25).

Tồn tại duy nhất một bộ ba Pythagor (*a, b, c*) thỏa điều kiện: a + b + c = 1000.

Hãy tìm tích abc.

Thuật toán

Nhận xét 1. Trong bộ ba Pythagor (a, b, c) không có 2 số bằng nhau. Thật vậy, giả sử a2 + b2 = c2. Khi đó, nếu a = b thì là số vô tỷ, không phải số nguyên. Nếu a = c thì b = 0, nếu b = c thì a = 0, đều vi phạm tính dương. Từ nhận xét này ta có thể quy định 0 < a < b < c.

Nhận xét 2. Nếu (a,b,c) là một bộ ba Pythagor thì với mọi số nguyên dương k ≥ 1 (ka, kb, kc) cũng là một bộ ba Pythagor. Thật vậy, ta có (ka)2 + (kb)2 = k2(a2+b2) = (kc)2. Từ nhận xét này ta qui định chỉ xét những bộ ba Pythagor (a, b, c) nguyên tố cùng nhau, tức là ucln(a, b, c) = 1. Ta gọi chúng là những *bộ ba Pythagor nguyên thủy.*

Nhận xét 3. Nếu tổng a+b+c = k thì nhận xét 1 cho ta 1 ≤ a < k/3, a < b, c = k−a−b.

Nếu ta duyệt ngây thơ ba vòng lặp để phát hiện bộ ba Pythagor như sau:

for (a = 1; a ≤ 1000; ++a)

for (b = 1; b ≤ 1000; ++b)

for (c = 1; c ≤ 1000; ++c)

if ((a+b+c)==1000)&&(a\*a+b\*b==c\*c))

return a\*b\*c;

thì chi phí tính toán sẽ là 10003 = 1 tỷ. Nhận xét 3 cho phép ta giảm số vòng lặp và giới hạn đầu cuối của mỗi vòng lặp như sau:

for (a = 1; a ≤ 300; ++a)

for (b = a+1; b ≤ 600; ++b) {

c = 1000 – a – b;

if (a\*a+b\*b==c\*c)

return a\*b\*c;

}

Theo sơ đồ này thì chi phí tính toán sẽ là 300×600 = 180000 giảm được khoảng 5000 lần.

Đề bài cho biết chỉ có duy nhất một bộ ba Pythagor (a, b, c) thỏa tính chất a+b+c=1000 do đó ta có thể bỏ qua việc kiểm tra ucln(a,b,c) = 1.

Nhận xét 4. Bạn có thể tính toán thêm chút nữa để tiếp tục cải tiên thuật toán. Cụ thể như sau:

Thay c = 1000-(a+b) vào hệ thức Pythagor ta thu được

a2 + b2 = (1000-(a+b))2

Sau khi biến đổi hệ thức này ta thu được hệ thức

500000 + ab = 1000(a+b)

Thủ tục E9B được cài đặt theo hệ thức này.

## Problem 10. [Summation of primes](https://projecteuler.net/problem=10)

The sum of the primes below 10 is 2 + 3 + 5 + 7 = 17.

Find the sum of all the primes below two million.

**Bài 10. Tổng các số nguyên tố**

Tổng các số nguyên tố dưới 10 là 2 + 3 + 5 + 7 = 17.

Hãy tìm tổng các số nguyên tố dưới hai triệu.

Thuật toán

Dùng sàng Eratosthenes liệt kê các số nguyên tố < 2000000 sau đó tìm tổng của chúng. Thuật toán sàng Eratosthenes như sau:

Chức năng: liệt kê các số nguyên tố ≤ n

S1. Viết các số 1..n. Xóa 1 là số đặc biệt, không phải là nguyên tố và không phải là hợp số.

S2. Lặp lần

S2.2. Tìm số i đầu tiên tiếp theo chưa bị xóa: đó là số nguyên tố.

S2.2. Xóa toàn bộ các bội của i kể từ i2 đến n.

Sau khi kết thúc các số còn lại chưa bị xóa sẽ là các số nguyên tố.

Ta dùng một mảng bit b để đánh dấu các số nguyên tố. Ta qui định b[i] = 1 nếu i là số nguyên tố.

Câu lệnh

bitset<MN+1> b;

khai báo mảng bit gồm MN+1 phần tử, mỗi phần tử có thể nhận trik 0 / 1.

Câu lệnh

b.set();

gán trị 1 cho mọi phần tử của b.

Để ý rằng ngoài 2 là số nguyên tố chẵn duy nhất, các số nguyên tố khác đều lẻ nên khi duyệt ta có thể chỉ chạy trên các số lẻ.

## Problem 11. [Largest product in a grid](https://projecteuler.net/problem=11)

In the 20×20 grid below, four numbers along a diagonal line have been marked in red.

08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08  
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00  
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65  
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91  
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80  
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50  
32 98 81 28 64 23 67 10 **26** 38 40 67 59 54 70 66 18 38 64 70  
67 26 20 68 02 62 12 20 95 **63** 94 39 63 08 40 91 66 49 94 21  
24 55 58 05 66 73 99 26 97 17 **78** 78 96 83 14 88 34 89 63 72  
21 36 23 09 75 00 76 44 20 45 35 **14** 00 61 33 97 34 31 33 95  
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92  
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57  
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58  
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40  
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66  
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69  
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36  
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16  
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54  
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

The product of these numbers is 26 ×63 ×78 ×14 = 1788696.

What is the greatest product of four adjacent numbers in any direction (up, down, left, right, or diagonally) in the 20×20 grid?

**Bài 11. Tổng lớn nhất trên lưới**

Trong lưới 20×20 trên, bốn số trên đường chéo được tô đậm. Tích của các số đó là 26 ×63 ×78 ×14 = 1788696.

Hãy xác định tích lớn nhất của bốn số liền kề nhau theo mọi hướng (lên, xuống, trái, phải hoặc chéo) trong lưới 20×20?

Thuật toán

Bài này chủ yếu rèn luyện các phương thức duyệt lưới. Gọi ss là mảng hai chiều biểu điên lưới, ta duyệt lần lượt 4 nhóm làn sau đây:

* Nhóm 1: 20 cột từ trên xuống với;
* Nhóm 2: 20 dòng từ trái qua phải;
* Nhóm 3: 17 đường chéo từ trên qua phải;
* Nhóm 4: 17 đường chéo từ trên qua trái.

Với mỗi làn ta lấy tích 4 phần tử liên tiếp và lưu lại tích pmax. Điều cần quan tâm ở mỗi làn là xác định điểm đầu và cuối của làn. Vì mỗi tích ta chỉ lấy DELTA = 4 nhân tử, nên ta chỉ cần biết điểm đầu của mỗi làn và phương thức di chuyển để lấy 4 số liên tiếp theo làn đó thì có thể tính được tích. Bảng dưới đây cho biết cách xác định điểm đầu và phương thức di chuyển theo mỗi làn. Ta kí hiệu d là biến chỉ số dòng, c là biến chỉ số cột. Nếu di chuyển theo cột thì ta chỉ cần tăng chỉ số dòng, ++d cho mỗi phần tử tiếp theo, di chuyển theo dòng: ++c; theo chéo phải: ++d, ++c; chéo trái ++d, --c.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nhóm |  | Điểm đầu | Di chuyển | |
| dòng d | cột c |
| 1 | cột c | **ss[0][c]**  **c = 0..19** | **d = 0..16** |  |
| 2 | dòng d | **ss[d][0]**  **d = 0..19** |  | **c = 0..16** |
| 3 | chéo phải \ | **ss[0][c]**  **c = 0..16** | **++d** | **++c** |
| 4 | chéo trái / | **ss[0][c]**  **c = 3..19** | **++d** | **--c** |

## Problem 12. [Highly divisible triangular number](https://projecteuler.net/problem=12)

The sequence of triangle numbers is generated by adding the natural numbers. So the 7th triangle number would be 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28. The first ten terms would be:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

Let us list the factors of the first seven triangle numbers:

**1**: 1  
**3**: 1,3  
**6**: 1,2,3,6  
**10**: 1,2,5,10  
**15**: 1,3,5,15  
**21**: 1,3,7,21  
**28**: 1,2,4,7,14,28

We can see that 28 is the first triangle number to have over five divisors.

What is the value of the first triangle number to have over five hundred divisors?

**Bài 12. Số tam giác nhiều ước**

Dãy số tam giác được sinh bằng cách cộng dồn các số tự nhiên đầu tiên. Như vậy, số tam giác thứ bảy sẽ là 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28. Mười số tam giác đầu tiên sẽ là:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

Chúng ta thử liệt kê các ước của bảy số tam giác đầu tiên:

**1**: 1  
**3**: 1,3  
**6**: 1,2,3,6  
**10**: 1,2,5,10  
**15**: 1,3,5,15  
**21**: 1,3,7,21  
**28**: 1,2,4,7,14,28

Chúng ta thấy 28 là số đầu tiên có trên năm ước.

Hãy tìm số tam giác đầu tiên có trên năm trăm ước?

Đáp án: 76576500

Thuật toán

Kí hiệu T(n) là số tam giác thứ n, ta thấy

T(1) = 1;

T(n) = T(n-1) + n; n > 1.

Tuy nhiên ta sẽ vận dụng công thức Gauss tính T(n) cho bài toán này.

Để ý rằng trong hai số tự nhiên liên tiếp luôn có đúng một số chẵn. Khi đó:

* Nếu n là số chẵn, n = 2k, thì T(n) = 2k(2k+1)/2 = k(2k+1);
* Nếu n là số lẻ, n = 2k+1, thì T(n) = 2(2k+1)(k+1)/2 = (k+1)(2k+1).

Trước hết ta chứng minh k và 2k+1 nguyên tố cùng nhau và k+1 và 2k+1 nguyên tố cùng nhau. Thật vậy, nếu k có ước d, k = ad thì 2k+1 = 2ad+1. Giả sử 2k+1 cũng có ước d, k = bd, khi đó 2k+1 = 2ad+1 = bd, hay bd - 2ad = d(b-2a) = 1. Từ đây suy ra d = 1.

Tương tự, giả sử k+1 và 2k+1 đều có chung ước d, k+1 = ad, 2k+1 = bd. Khi đó

2k+1-(k+1) = k = bd-ad = d(b-a) tức là k cũng có ước d. Nhưng theo chứng minh trên thì k và 2k+1 là nguyên tố cùng nhau nên d = 1.

Ta biết hàm tính số ước của số tự nhiên n, τ(n) là hàm tích, nghĩa là nếu a và b là hai số nguyên tố cùng nhau, (a,b) = 1, thì τ(ab) = τ(a)τ(b). Khi đó,

* Nếu n là số chẵn, n = 2k, thì T(n) = k(2k+1), do đó τ(T(n)) = τ(k)τ(2k+1);
* Nếu n là số lẻ, n = 2k+1, thì T(n) = (k+1)(2k+1), do đó τ(T(n)) = τ(k+1)τ(2k+1).

Ta tìm cách tính nhanh τ(T(n)). Ta thử viết vài giá trị đầu tiên của T(n).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***n*** | ***T(n) = ab*** | ***a*** | ***b*** | ***τ(a)*** | ***τ(b)*** | ***τ(T(n)) = τ(a)τ(b)*** |
| 1 | 1⋅2/2 = 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2⋅3/2 =3 | 1 | 3 | 1 | 2 | 2 |
| 3 | 3⋅4/2 = 6 | 2 | 3 | 2 | 2 | 4 |
| 4 | 4⋅5/2 = 10 | 2 | 5 | 2 | 2 | 4 |
| 5 | 5⋅6/2 = 15 | 3 | 5 | 2 | 2 | 4 |
| 6 | 6⋅7/2 = 21 | 3 | 7 | 2 | 2 | 4 |
| 7 | 7⋅8/2 = 28 | 4 | 7 | 3 | 2 | 6 |
| 8 | 8⋅9/2 = 36 | 4 | 9 | 3 | 3 | 9 |
| 9 | 9.10/2 = 45 | 5 | 9 | 2 | 3 | 6 |

Ta rút ra qui tắc sau đây để tính T(n), n ≥ 3:

* Khởi trị: T(3) = a⋅b, a = 1, b = 3
* Nếu n chẵn thì b = b+2, T(n) = a⋅b
* Nếu n lẻ thì a = a + 1, T(n) = a⋅b

Và do đó ta suy ra qui tắc tính τ(T(n)):

* Khởi trị: T(3) = a⋅b, ta = τ(a) = τ(1) = 1, tb = τ(b) = τ(3) = 2
* Nếu *n* chẵn thì *b* = *b*+2, *tb* = τ(*b*) *tt* = τ(*n*) = *ta* \* *tb*
* Nếu *n* lẻ thì *a* = *a* + 1, *ta* = τ(*a*), *tt* = τ(*n*) = *ta* \* *tb*

Có hai cách tính hàm Tau: vận dụng tính chia hết hoặc vận dụng định lí cơ bản của số học. Cả hai cách đều đòi hỏi tối đa phép kiểm tra số p có phải là ước của n hay không? tức là kiểm tra n mod p = 0?

Theo cách thứ nhất ta dựa vào nhận xét, nếu n = a⋅b và a ≠ b thì n có hai ước là a và b. Khi ta phát hiện n mod d = 0 thì ta ghi nhận n có 2 ước. Tuy nhiên, khi n là số chính phương, tức là khi ⋅ = *n* thì ta bớt đi một ước.

// So uoc cua n

int Tau(int n) {

int sqr = int(sqrt(n));

int s = 2, d;

for (d = 2; d <= sqr; ++d)

if ((n % d) == 0) s += 2;

return (sqr\*sqr == n) ? s-1 : s;

}

Định lí cơ bản của số học, còn được gọi là định lí phân tích ra thừa số nguyên tố, cho ta biết mọi số tự nhiên n > 1 đều có thể viết dưới dạng duy nhất tích của các thừa số nguyên tố theo trật tự tăng dần, cụ thể là,

trong đó *p*1 < … < *pk* là các số nguyên tố.

Dạng biểu diễn trên cho ta công thức tính số ước của số tự nhiên n là

τ(*n*) = (*m*1-1)⋅⋅⋅(*mk*-1)

Hàm Deg(n, p) cho ta số mũ cao nhất *m* thỏa *pm* là ước của *n*. Sau lời gọi hàm, bản thân số n sẽ thay đổi, cụ thể là *n* = *n* div *pm* , thí dụ, với n = 72, p = 2 thì sau lời gọi Deg(n,p) ta thu được m = 3, n = 9. Dựa vào hàm Deg ta cài đặt hàm Tau(n) tính số ước của số tự nhiên n như sau:

Cải tiến

Khảo sát sự biến thiên của a và b trong bảng tính hàm Tau bạn thấy b luôn luôn lẻ, còn a biến thiên tuần tự hai lần mang giá trị giống nhau lần lượt là 1, 1; 2, 2; 3, 3; … Như vậy khi a lẻ ta khỏi phải tính lại Tau(a) vì trước đó ta đã tính Tau(b) mà nay ta có a = b. Ta sẽ lưu lại giá trị Tau(b) vào một hàng đợi không lớn, chừng 6 phần tử là đủ theo tổ chức vòng tròn.

## Problem 13. [Large sum](https://projecteuler.net/problem=13)

Work out the first ten digits of the sum of the following one-hundred 50-digit numbers.

37107287533902102798797998220837590246510135740250  
46376937677490009712648124896970078050417018260538  
74324986199524741059474233309513058123726617309629  
91942213363574161572522430563301811072406154908250  
23067588207539346171171980310421047513778063246676  
89261670696623633820136378418383684178734361726757  
28112879812849979408065481931592621691275889832738  
44274228917432520321923589422876796487670272189318  
47451445736001306439091167216856844588711603153276  
70386486105843025439939619828917593665686757934951  
62176457141856560629502157223196586755079324193331  
64906352462741904929101432445813822663347944758178  
92575867718337217661963751590579239728245598838407  
58203565325359399008402633568948830189458628227828  
80181199384826282014278194139940567587151170094390  
35398664372827112653829987240784473053190104293586  
86515506006295864861532075273371959191420517255829  
71693888707715466499115593487603532921714970056938  
54370070576826684624621495650076471787294438377604  
53282654108756828443191190634694037855217779295145  
36123272525000296071075082563815656710885258350721  
45876576172410976447339110607218265236877223636045  
17423706905851860660448207621209813287860733969412  
81142660418086830619328460811191061556940512689692  
51934325451728388641918047049293215058642563049483  
62467221648435076201727918039944693004732956340691  
15732444386908125794514089057706229429197107928209  
55037687525678773091862540744969844508330393682126  
18336384825330154686196124348767681297534375946515  
80386287592878490201521685554828717201219257766954  
78182833757993103614740356856449095527097864797581  
16726320100436897842553539920931837441497806860984  
48403098129077791799088218795327364475675590848030  
87086987551392711854517078544161852424320693150332  
59959406895756536782107074926966537676326235447210  
69793950679652694742597709739166693763042633987085  
41052684708299085211399427365734116182760315001271  
65378607361501080857009149939512557028198746004375  
35829035317434717326932123578154982629742552737307  
94953759765105305946966067683156574377167401875275  
88902802571733229619176668713819931811048770190271  
25267680276078003013678680992525463401061632866526  
36270218540497705585629946580636237993140746255962  
24074486908231174977792365466257246923322810917141  
91430288197103288597806669760892938638285025333403  
34413065578016127815921815005561868836468420090470  
23053081172816430487623791969842487255036638784583  
11487696932154902810424020138335124462181441773470  
63783299490636259666498587618221225225512486764533  
67720186971698544312419572409913959008952310058822  
95548255300263520781532296796249481641953868218774  
76085327132285723110424803456124867697064507995236  
37774242535411291684276865538926205024910326572967  
23701913275725675285653248258265463092207058596522  
29798860272258331913126375147341994889534765745501  
18495701454879288984856827726077713721403798879715  
38298203783031473527721580348144513491373226651381  
34829543829199918180278916522431027392251122869539  
40957953066405232632538044100059654939159879593635  
29746152185502371307642255121183693803580388584903  
41698116222072977186158236678424689157993532961922  
62467957194401269043877107275048102390895523597457  
23189706772547915061505504953922979530901129967519  
86188088225875314529584099251203829009407770775672  
11306739708304724483816533873502340845647058077308  
82959174767140363198008187129011875491310547126581  
97623331044818386269515456334926366572897563400500  
42846280183517070527831839425882145521227251250327  
55121603546981200581762165212827652751691296897789  
32238195734329339946437501907836945765883352399886  
75506164965184775180738168837861091527357929701337  
62177842752192623401942399639168044983993173312731  
32924185707147349566916674687634660915035914677504  
99518671430235219628894890102423325116913619626622  
73267460800591547471830798392868535206946944540724  
76841822524674417161514036427982273348055556214818  
97142617910342598647204516893989422179826088076852  
87783646182799346313767754307809363333018982642090  
10848802521674670883215120185883543223812876952786  
71329612474782464538636993009049310363619763878039  
62184073572399794223406235393808339651327408011116  
66627891981488087797941876876144230030984490851411  
60661826293682836764744779239180335110989069790714  
85786944089552990653640447425576083659976645795096  
66024396409905389607120198219976047599490197230297  
64913982680032973156037120041377903785566085089252  
16730939319872750275468906903707539413042652315011  
94809377245048795150954100921645863754710598436791  
78639167021187492431995700641917969777599028300699  
15368713711936614952811305876380278410754449733078  
40789923115535562561142322423255033685442488917353  
44889911501440648020369068063960672322193204149535  
41503128880339536053299340368006977710650566631954  
81234880673210146739058568557934581403627822703280  
82616570773948327592232845941706525094512325230608  
22918802058777319719839450180888072429661980811197  
77158542502016545090413245809786882778948721859617  
72107838435069186155435662884062257473692284509516  
20849603980134001723930671666823555245252804609722  
53503534226472524250874054075591789781264330331690

**Bài 13. Tổng lớn**

Ghi ra mười chữ số đầu tiên của tổng một trăm số dài 50 chữ số cho trong bảng trên.

Thuật toán

Thuật ngữ “… *chữ số đầu tiên*” có thể được hiểu là các chữ số đầu trái (hàng cao) hoặc đầu phải (hàng thấp). Thí dụ 3 chữ số đầu tiên của số 12345678 có thể là 123 hoặc 678. Ta đành cài đặt theo hai cách hiểu vậy.

Bài này yêu cầu phép cộng 100 số lớn (50 chữ số). Nếu các số đều là những số lớn nhất, tức là có dạng 50 chữ số 9, s = 9…9 thì tổng của 100 số sẽ là 100s = 9..900, và sẽ có độ dài 52 chữ số.

Ta qui định dữ liệu vào được ghi trong mảng gồm 100 string (mỗi string 50 chữ số). Ta cần đổi giá trị char trong các string sang giá trị số, thí dụ short (kiểu nguyên ngắn 2 byte có trị từ -32768 đến 32767). Ta sẽ lưu các số vào biến kiểu short gồm tối đa 52 phần tử (trong chương trình khai báo rộng hơn, 55 phần tử). Thông thường, khi cộng xong mỗi cặp hai chữ số ta cần thực hiện phép chuyển số nhớ. Trong bài này, bạn nên cộng không nhớ đủ 100 số sau đó mới xử lí số nhớ. Bạn cũng nên viết ngược các số để khi xử lí các số nhớ ta còn có đủ chỗ. Bảng dưới đây minh họa việc xử lí số nhớ sau khi cộng 5 số, mỗi số 6 chữ số. Bên trái là dữ liệu vào, bên phải minh họa cách chuyển sang dạng viết ngược và cộng không nhớ các cột, cuối cùng mới xử lí số nhớ theo trật tự từ trái (hàng thấp) qua phải (hàng cao) theo thủ tục Carry. Kết quả cuối cùng sẽ là 2205834.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dạng xuôi | | | | | |  |  |  |  | Dạng ngược | | | | | |  |
| 5 | 3 | 4 | 9 | 7 | 4 |  |  |  |  | 4 | 7 | 9 | 4 | 3 | 5 |  |
| 6 | 5 | 7 | 8 | 2 | 6 |  |  |  |  | 6 | 2 | 8 | 7 | 5 | 6 |  |
| 2 | 4 | 6 | 9 | 9 | 7 |  |  |  |  | 7 | 9 | 9 | 6 | 4 | 2 |  |
| 4 | 4 | 6 | 2 | 9 | 9 |  |  |  |  | 9 | 9 | 2 | 6 | 4 | 4 |  |
| 3 | 1 | 9 | 7 | 3 | 8 |  |  |  |  | 8 | 3 | 7 | 9 | 1 | 3 |  |
|  |  |  |  |  |  |  |  |  | Tổng không nhớ | **34** | **30** | **35** | **32** | **17** | **20** |  |
|  |  |  |  |  |  |  |  |  | Kết quả | **4** | **3** | **8** | **5** | **0** | **2** | **2** |

Chương trình hiển thị tổng đầy đủ 100 số, 10 chữ số đầu và 10 chữ số cuối của kết quả. Chú ý rằng nếu đề bài chỉ yêu cầu hiển thị 10 chữ số hàng thấp thì bạn chỉ cần thực hiện phép cộng 100 số nói trên, với mỗi số bạn chỉ cần lấy 10 chữ số hàng thấp là đủ.

Chương trình DevCPP

// E13.CPP: Large Sum

#include <iostream>

#include <math.h>

#include <windows.h>

using namespace std;

// 50 chu so x 100 = 52 chu so

const short MN = 55;

const short LEN = 50;

short x[MN];

short len;

string s[100] = {

"37107287533902102798797998220837590246510135740250",

"46376937677490009712648124896970078050417018260538",

"74324986199524741059474233309513058123726617309629",

"91942213363574161572522430563301811072406154908250",

"23067588207539346171171980310421047513778063246676",

"89261670696623633820136378418383684178734361726757",

"28112879812849979408065481931592621691275889832738",

"44274228917432520321923589422876796487670272189318",

"47451445736001306439091167216856844588711603153276",

"70386486105843025439939619828917593665686757934951",

"62176457141856560629502157223196586755079324193331",

"64906352462741904929101432445813822663347944758178",

"92575867718337217661963751590579239728245598838407",

"58203565325359399008402633568948830189458628227828",

"80181199384826282014278194139940567587151170094390",

"35398664372827112653829987240784473053190104293586",

"86515506006295864861532075273371959191420517255829",

"71693888707715466499115593487603532921714970056938",

"54370070576826684624621495650076471787294438377604",

"53282654108756828443191190634694037855217779295145",

"36123272525000296071075082563815656710885258350721",

"45876576172410976447339110607218265236877223636045",

"17423706905851860660448207621209813287860733969412",

"81142660418086830619328460811191061556940512689692",

"51934325451728388641918047049293215058642563049483",

"62467221648435076201727918039944693004732956340691",

"15732444386908125794514089057706229429197107928209",

"55037687525678773091862540744969844508330393682126",

"18336384825330154686196124348767681297534375946515",

"80386287592878490201521685554828717201219257766954",

"78182833757993103614740356856449095527097864797581",

"16726320100436897842553539920931837441497806860984",

"48403098129077791799088218795327364475675590848030",

"87086987551392711854517078544161852424320693150332",

"59959406895756536782107074926966537676326235447210",

"69793950679652694742597709739166693763042633987085",

"41052684708299085211399427365734116182760315001271",

"65378607361501080857009149939512557028198746004375",

"35829035317434717326932123578154982629742552737307",

"94953759765105305946966067683156574377167401875275",

"88902802571733229619176668713819931811048770190271",

"25267680276078003013678680992525463401061632866526",

"36270218540497705585629946580636237993140746255962",

"24074486908231174977792365466257246923322810917141",

"91430288197103288597806669760892938638285025333403",

"34413065578016127815921815005561868836468420090470",

"23053081172816430487623791969842487255036638784583",

"11487696932154902810424020138335124462181441773470",

"63783299490636259666498587618221225225512486764533",

"67720186971698544312419572409913959008952310058822",

"95548255300263520781532296796249481641953868218774",

"76085327132285723110424803456124867697064507995236",

"37774242535411291684276865538926205024910326572967",

"23701913275725675285653248258265463092207058596522",

"29798860272258331913126375147341994889534765745501",

"18495701454879288984856827726077713721403798879715",

"38298203783031473527721580348144513491373226651381",

"34829543829199918180278916522431027392251122869539",

"40957953066405232632538044100059654939159879593635",

"29746152185502371307642255121183693803580388584903",

"41698116222072977186158236678424689157993532961922",

"62467957194401269043877107275048102390895523597457",

"23189706772547915061505504953922979530901129967519",

"86188088225875314529584099251203829009407770775672",

"11306739708304724483816533873502340845647058077308",

"82959174767140363198008187129011875491310547126581",

"97623331044818386269515456334926366572897563400500",

"42846280183517070527831839425882145521227251250327",

"55121603546981200581762165212827652751691296897789",

"32238195734329339946437501907836945765883352399886",

"75506164965184775180738168837861091527357929701337",

"62177842752192623401942399639168044983993173312731",

"32924185707147349566916674687634660915035914677504",

"99518671430235219628894890102423325116913619626622",

"73267460800591547471830798392868535206946944540724",

"76841822524674417161514036427982273348055556214818",

"97142617910342598647204516893989422179826088076852",

"87783646182799346313767754307809363333018982642090",

"10848802521674670883215120185883543223812876952786",

"71329612474782464538636993009049310363619763878039",

"62184073572399794223406235393808339651327408011116",

"66627891981488087797941876876144230030984490851411",

"60661826293682836764744779239180335110989069790714",

"85786944089552990653640447425576083659976645795096",

"66024396409905389607120198219976047599490197230297",

"64913982680032973156037120041377903785566085089252",

"16730939319872750275468906903707539413042652315011",

"94809377245048795150954100921645863754710598436791",

"78639167021187492431995700641917969777599028300699",

"15368713711936614952811305876380278410754449733078",

"40789923115535562561142322423255033685442488917353",

"44889911501440648020369068063960672322193204149535",

"41503128880339536053299340368006977710650566631954",

"81234880673210146739058568557934581403627822703280",

"82616570773948327592232845941706525094512325230608",

"22918802058777319719839450180888072429661980811197",

"77158542502016545090413245809786882778948721859617",

"72107838435069186155435662884062257473692284509516",

"20849603980134001723930671666823555245252804609722",

"53503534226472524250874054075591789781264330331690"

};

## Problem 14. [Longest Collatz sequence](https://projecteuler.net/problem=14)

The following iterative sequence is defined for the set of positive integers:

n → n/2 (n is even)

n → 3n + 1 (n is odd)

Using the rule above and starting with 13, we generate the following sequence:

13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

**NOTE:** Once the chain starts the terms are allowed to go above one million.

**Đáp số:** 837799 525

**Bài 14.** **Dãy Collatz dài nhất**

Dãy lặp dưới đây gồm các số nguyên dương đựơc sinh như sau:

n → n/2 (n chẵn)

n → 3n + 1 (n lẻ)

Vận dụng qui tắc trên, xuất phát từ số 13 chúng ta sinh ra dãy sau đây:

13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1

Dãy này bao gồm 10 số (xuất phát từ 13 và kết thúc tại 1). Có vẻ như xuất phát từ bất kì số nguyên dương nào, sớm muộn dãy lặp trên đều kết thúc tại 1 (Bài toán Collatz), dù điều này chưa được chứng minh.

Xuất phát từ số nào trong các số dưới một triệu có thể sinh ra dãy dài nhất?

**CHÚ Ý:** Quá trình lặp có thể sinh ra các số trên một triệu. Thật vậy, có thể lúc đầu giá trị n nhỏ (dưới 1 triệu), nhưng sau một số bước giá trị này có thể tăng lên đến vài tỷ. Trong thí dụ dưới đây, hầu hết số trung gian đều vượt quá một triệu,

837799 → 2513398 → 1256699 → 3770098 → 1885049 → 5655148 → 2827574 → 1413787 → 4241362 → 2120681 → 6362044 → 3181022 → 1590511→ 4771534→ 2385767→ 7157302→ 3578651 → 10735954 → 5367977 → 16103932 → 8051966 → 4025983 → 12077950 → 6038975 → 18116926 → 9058463 → 27175390 → 13587695 → 40763086 → 20381543 → 61144630 → 30572315 → 91716946 → 45858473 → 137575420 → 68787710 → 34393855 → 103181566 → 51590783 → 154772350 → 77386175 → 232158526 → 116079263 → 348237790 → 174118895 → 522356686 → 261178343 → 783535030 → 391767515 → 1175302546.

Thuật toán

Ta tạm kí hiệu dãy Collatz sinh từ số nguyên dương n là C(n) và #C(n) là chiều dài của dãy C(n).

Nhận xét

1. Với mọi trị nguyên dương n, C(n) là dãy không lặp, nghĩa là trong C(n) không có hai phần tử giống nhau. Vì giả thuyết Collatz chưa được chứng minh nên nhận xét này cũng chưa được chứng minh. Tuy nhiên có thể kiểm chứng nhận xét này với các giá trị n dưới vài triệu, đủ để giải bài toán này.

2. Nếu x xuất hiện trong C(n) tại vị trí thứ k thì #C(n) = #C(x)+k-1. Thật vậy, giả sử C(n) = (b1,…,bk = x,…,bm = 1), ta có #C(n) = m = k+(m-k) = k-1+#C(bk) = k-1+#C(x).

Nhận xét này gợi ý ta tìm cách lưu lại các chiều dài đã tính trước.

Phương án đầu tiên khá đơn giản. Ta biết C(1) = 1, do đó #C(1) = 1. Tiếp theo ta tính lần lượt các #C(i) từ nhỏ đến lớn, tức là từ i = 2 đến 999999 và lưu kết quả #C(i) vào mảng len[i]. Nếu khi tính C(n), tại bước thứ d ta gặp phần tử trung gian bd < n thì ta có ngay kết quả len[n] = len(bd)+d-1. Thí dụ, sau khi tính được #C(n) = len[n] với n = 1..16. Giả sử ta cần tính #C(17). Ta có

17 → 52 → 26 → 13. Đến đây ta có ngay len[17] = len[13]+3.

Cải tiến

Bạn có thể tính trước khá nhiều giá trị cho len[n], thí dụ, len[2i] = i+1, mà 2i thì khá dễ tính, 2i = (1 ≪ i).

Ngoài ra bạn có thể sinh ngược dãy Collat như sau:

Xuất phát từ một số nguyên dương n tùy ý. Thay vì cho n tiến đến 1, bạn cho n phát triển theo chiều ngược lại để xác định len của các phần tử trung gian.

Qui tắc sinh ngược là như sau:

Nếu n-1 chia hết cho 3 thì đặt lại n = n div 3;

nếu không thì đặt lại n = n\*2;

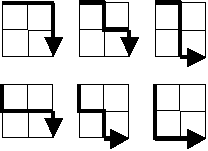
Sau mỗi bước, tăng chiều dài thêm 1.

Thí dụ, bạn xuất phát từ n = 16, len = 5 = #C(16) sẽ thu được dãy ngược như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 16 | 5 | 10 | 3 | 6 | 12 | 24 | 48 | 96 | 192 | 384 | 768 | 1536 | 3072 | 6144 | … |
| len(n) | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | … |

## Problem 15. [Lattice paths](https://projecteuler.net/problem=15)

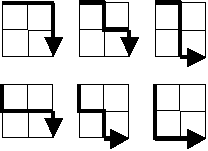
Starting in the top left corner of a 2×2 grid, there are 6 routes (without backtracking) to the bottom right corner.



How many routes are there through a 20×20 grid?

**Bài 15. Các đường dạng dàn**

Xuất phát từ góc trên trái của lưới (đơn vị) 2×2 đi xuống đến góc dưới phải theo các cạnh đơn vị dọc hoặc ngang (và không quay lui) sẽ thu được 6 đường đi khác nhau.



Có bao nhiêu đường đi khi duyệt lưới 20×20?

Thuật toán

Ta giải bài toán tổng quát với lưới đơn vị a × b, trong đó a là chiều dọc, b là chiều ngang của lưới. Trên lưới này ta đánh nhãn các cạnh đơn vị nằm ngang là 0, dọc là 1. Mọi đường đi không quay lui từ góc trên trái đến góc dưới phải đều phải qua đúng b cạnh đơn vị ngang và a cạnh đơn vị dọc. Bài toán quy về dạng tổ hợp như sau. Có thể xây dựng bao nhiêu xâu kí tự độ dài a+b từ hai kí tự 0 và 1 trong đó có đúng a kí tự 1 và b kí tự 0. Đáp số sẽ là tổ hợp chặp a của a+b.

Thí dụ, 6 đường trong lưới 2×2 sẽ là

0011, 0101, 0110, 1001, 1010, 1100.

Công thức tính tổ hợp chặp k của n phần tử là như sau:

 \binom nk = \frac{n(n-1)\dotsb(n-k+1)}{k(k-1)\dotsb1}

Ta xây dựng hàm C(n,k) tính cho công thức trên.

Ta có

* C(n,0) = 1
* C(n,k) = C(n,k-1)(n-k+1)/k

Theo qui tắc này bạn có thể cài đặt hàm đệ qui C(n,k) như sau:

UL C(UL n, UL k) {

return (k == 1) ? n : C(n,k-1)\*(n-k+1)/k;

}

trong đó UL là kiểu số nguyên không âm 64 bit (8 bytes) được định nghĩa như sau:

typedef unsigned long long UL;

Bạn cũng có thể cài đặt hàm CC theo phương án không đệ qui như sau:

UL CC(UL n, UL k) {

UL r = 1;

for (int i = 0; i < k; ++i) {

r = r \* (n-i) / (i+1);

}

return r;

}

## Problem 16. [Power digit sum](https://projecteuler.net/problem=16)

215 = 32768 and the sum of its digits is 3 + 2 + 7 + 6 + 8 = 26.

What is the sum of the digits of the number 21000?

**Bài 16. Tổng các chữ số của lũy thừa**

215 = 32768 và tổng các chữ số của số này là 3 + 2 + 7 + 6 + 8 = 26.

Tổng các chữ số của số 21000 là bao nhiêu?

Thuật toán

Bài này đòi hỏi kĩ năng biểu diễn số lớn. Xuất phát từ 1 ta nhân 2 1000 lần sau đó lấy tổng các chữ số của tích cuối cùng.

Ta dùng một mảng num để lưu các tích trung gian với n chữ số. lưu ý rằng n sẽ tăng dần do đó tốt nhất là ta lưu số dưới dạng ngược, thí dụ, số x = 65536 sẽ được lưu như sau: n = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 |
| num | **6** | **3** | **5** | **5** | **6** |

Sau khi nhân 2 ta thu được: n = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 |
| num | **6** | **3** | **5** | **5** | **6** |  |
| num | **2** | **7** | **0** | **1** | **3** | **1** |

Số sẽ phát triển về đầu phải do vậy bạn khỏi phải dịch mảng.

Ngoài ra ta để ý điều quan trọng sau đây:

Trong máy tính dữ liệu được lưu dưới dạng nhị phân, do đó, khi xử lí các số nguyên, bạn nên thay các phép toán số học bằng các phép toán trên bit. Dưới đây là một số dạng thay thế.

|  |  |  |  |
| --- | --- | --- | --- |
| Phép toán  thập phân | x\*2k | x / 2k | x % 2k |
| Nên thay bằng  phép toán nhị phân | x << k | x >> k | x & (2k−1) |
| *x là biến nguyên* | | | |

Thí dụ,

int x = 1;

x <<= 10; // x = 2^10 = 1024

x >>= 2; // x = x / 4 = 256

++x; // x = 257

x &= 7; // x = x % 8 = 1

## Problem 17. [Number letter counts](https://projecteuler.net/problem=17)

If the numbers 1 to 5 are written out in words: one, two, three, four, five, then there are 3 + 3 + 5 + 4 + 4 = 19 letters used in total.

If all the numbers from 1 to 1000 (one thousand) inclusive were written out in words, how many letters would be used?

**NOTE:** Do not count spaces or hyphens. For example, 342 (three hundred and forty-two) contains 23 letters and 115 (one hundred and fifteen) contains 20 letters. The use of "and" when writing out numbers is in compliance with British usage.

**Bài 17. Đếm số kí tự khi viết số**

Nếu viết bằng các chữ tiếng Anh các số từ 1 đến 5 như sau: one, two, three, four, five, thì phải dùng tổng cộng 3 + 3 + 5 + 4 + 4 = 19 chữ cái.

Nếu viết toàn bộ các số từ 1 đến 1000 (một nghìn) kể cả số đầu (1) và số cuối (1000) thì phải dùng tổng cộng bao nhiêu chữ cái.?

**CHÚ Ý:** Không tính các dấu cách, dấu nối, hoặc dấu ngoặc (đơn, kép, nháy,…). Thí dụ, 342 (three hundred and forty-two) chứa 23 chữ cái; 115 (one hundred and fifteen) chứa 20 chữ cái. Việc dùng chữ "and" là tuân theo qui tắc Anh (British).

**CHÚ THÍCH** (của người dịch)**:** Theo ngữ pháp Tiếng Việt, *chữ cái* gồm các con chữ như a, b, c, …, *chữ* được ghép từ các con chữ (trong tiếng Anh gọi là word, *từ*).

Thuật toán

Bài này thường gặp khi phải viết rõ các số dưới dạng chữ, thí dụ như viết hóa đơn hoặc giấy biên nhận khi giao dịch tài chính. Ngoài ra, nếu bạn phải dạy cho robot nói tiếng người, chắc chắn bạn sẽ phải xử lí và tổng hợp tiếng nói.

Chúng ta thử cài đặt 3 Demo để có thể tự hoàn thiện tiếng Anh:

Demo 1: Hiển thị dạng viết toàn bộ từ 1 đến 1000,

Demo 2: Hiển thị dạng viết cho số nạp từ bàn phím,

Demo 3: Tính tổng các chữ số dùng để viết các số từ 1 đến 1000.

trong đó Demo 3 chính là lời giải bài 17.

Ta tạm gọi việc hiển thị dạng chữ cho một số n là *diễn từ số n.* Việc đầu tiên cần làm là phải diễn từ cho các số cơ sở, bao gồm:

* Các số từ 1 đến 19
* Các số chẵn chục 20, 30, …, 90
* 100
* 1000

|  |  |  |  |
| --- | --- | --- | --- |
| *Diễn từ cơ sở* | | | |
| 1 | one |  |  |
| 2 | two | 20 | twenty |
| 3 | three | 30 | thirty |
| 4 | four | 40 | forty |
| 5 | five | 50 | fifty |
| 6 | six | 60 | sixty |
| 7 | seven | 70 | seventy |
| 8 | eight | 80 | eighty |
| 9 | nine | 90 | ninety |
| 10 | ten |  |  |
| 11 | eleven |  | and |
| 12 | twelve | 100 | hundred |
| 13 | thirteen | 1000 | thousand |
| 14 | fourteen |  |  |
| 15 | fifteen |  |  |
| 16 | sixteen |  |  |
| 17 | seventeen |  |  |
| 18 | eighteen |  |  |
| 19 | nineteen |  |  |

Với mỗi số tự nhiên n ta xét các khả năng sau đây:

K1. n = 1000 là số lớn nhất: hiển thị diễn từ của n; dừng thuật toán.

K2. 99 < n < 1000: Tách chữ số hàng trăm của n, t = n / 100 và phần hàng chục còn lại là n = n % 100: hiển thị diễn từ của t rồi làm tiếp K3 với n còn lại;

K3. n < 20: hiển thị diễn từ cơ sở của n; dừng thuật toán

K4. 20 ≤ n ≤ 99: Hiển thị chữ số hàng chục của n là n/10 và chữ số hàng đơn vị của n là n % 10.

Nếu chỉ cần đếm số kí tự của diễn từ thì bạn có thể tính trước chiều dài của các từ và ghi sẵn vào các mảng, biến tương ứng như trong thủ tục E17B.

## Problem 18. Maximum sum in the triangle

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

**3**  
**7** 4  
2 **4** 6  
8 5 **9** 3

That is, 3 + 7 + 4 + 9 = 23.

Find the maximum total from top to bottom of the triangle below:

75  
95 64  
17 47 82  
18 35 87 10  
20 04 82 47 65  
19 01 23 75 03 34  
88 02 77 73 07 63 67  
99 65 04 28 06 16 70 92  
41 41 26 56 83 40 80 70 33  
41 48 72 33 47 32 37 16 94 29  
53 71 44 65 25 43 91 52 97 51 14  
70 11 33 28 77 73 17 78 39 68 17 57  
91 71 52 38 17 14 91 43 58 50 27 29 48  
63 66 04 68 89 53 67 30 73 16 69 87 40 31  
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

**NOTE:** As there are only 16384 routes, it is possible to solve this problem by trying every route. However, [Problem 67](https://projecteuler.net/problem=67), is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method!)

**Bài 18. Tổng lớn nhất trong tam giác số**

Xuất phát từ đỉnh của tam giác số nhỏ di chuyển xuống cạnh đáy qua các số kề dưới số đứng trên có thể thu được tổng lớn nhất là 23 (xem các số tô đậm).

**3**  
**7** 4  
2 **4** 6  
8 5 **9** 3

Đó là **3** + **7** + **4** + **9** = 23.

Hãy tính tổng lớn nhất khi xử lí tam giác số lớn trên kia.

**CHÚ Ý:** Bài này chỉ có 16384 đường di chuyển khác nhau nên bạn có thể vét cạn. Tuy nhiên với bài tương tự là bài số 67 có đến một trăm dòng thì không thể vận lực thuần túy được mà phải tìm một phương pháp sáng láng.

Thuật toán

Bài này đã được lấy làm đề thi Olympic Tin học Quốc tế tại Thụy Điển.

Qui hoạch động.

Gỉa sử tam giác số có n dòng, dòng thứ i có i số. Từ số thứ k của dòng trên ta được phép di chuyển đến 2 số của dòng dưới là số thứ k-1 và k. Gọi v(r, k) là giá trị max của tổng thu được khi di chuyển từ đỉnh tam giác đến vị trí k của dòng thứ r đang xét. Ta có hệ thức sau:

v(r, k) = max { v(r-1, k-1)+r(k), v(r, k)+r(k) } = max { v(r-1, k-1), v(r, k) } + r(k)

trong đó r(k) là số thứ k trên dòng r.

Giá trị max của toàn bài sẽ là max {v(n, k) | k = 1..n}

Ta qui ước dữ liệu vào được ghi trong file E18.INP gồm n+1 dòng. Dòng đầu tiên là số nguyên dương n. Tiếp đến là n dòng, dòng thứ i trong số dòng tiếp theo gồm i số nguyên cách nhau qua dấu cách.

Ta dùng hai mảng: row chứa các phần tử của dòng đang xét, và val chứa các giá trị max thu được tính từ đỉnh tam giác đến mỗi dòng.

Độ phức tạp: Nếu tam giác số có m số, mỗi số được xử lí một lần với một phép so sánh để lấy trị max thì số phép so sánh sẽ là O(m) (cỡ m).

## Problem 19. [How many Sundays on the first of a month](http://www.mathblog.dk/project-euler-19/) ?

You are given the following information, but you may prefer to do some research for yourself.

1 Jan 1900 was a Monday.

Thirty days has September,

April, June and November.

All the rest have thirty-one,

Saving February alone,

Which has twenty-eight, rain or shine.

And on leap years, twenty-nine.

A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

How many Sundays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?

**Bài 19. Có bao nhiêu đầu tháng là ngày Chủ Nhật ?**

Bạn biết các thông tin sau đây, tuy nhiên, tốt nhất là bạn tự tìm hiểu để có thể kiểm chứng.

Mồng 1 tháng Giêng năm 1900 là thứ Hai.

Tháng Chín có 30 ngày,

Các tháng Tư, Sáu, Chín và Mười Một đều có 30 ngày.

Các tháng còn lại đều có 31 ngày, trừ Tháng Hai,

Tháng Hai có 28 ngày vào năm thường và 29 ngày vào năm nhuận.

*Năm nhuận* là năm chia hết cho 4 nhưng không phải là năm đầu thế kỉ trừ khi nó chia hết cho 400.

Trong thế kỉ XX (tính từ 1/1/1901 đến 31/12/2000) có bao nhiêu ngày đầu tháng là Chủ Nhật?

Thuật toán

Ta có thể ghi các thông tin dưới dạng bảng như sau:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tháng | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| Số ngày | 31 | 2? | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |
| *Tháng 2 năm thường có 28 ngày, năm nhuận có 29 ngày* | | | | | | | | | | | | |
| *Năm abcd là nhuận nếu: (cd ≠ 0 và cd mod 4 = 0)*  *hoặc (cd = 0 và ab mod 4 = 0)* | | | | | | | | | | | | |

Vì có nhiều qui ước khác nhau về niên lịch nên có nhiều cách tính ngày tháng và thứ (trong tuần) khác nhau. Trong bài này chúng ta tìm hiểu các công thức tính do nhà Toán học Đức Carl Friedrich Gauss đề xuất và được tìm thấy trong các bản thảo viết tay của Người.

***Mã số thứ trong tuần***: Gauss qui ước Chủ nhật mang mã số 0, Thứ Hai: 1, …, Thứ Bảy: 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Mã số thứ trong tuần* (*Gauss*) | | | | | | | |
| Thứ | Chủ Nhật | Thứ Hai | Thứ Ba | Thứ Tư | Thứ Năm | Thứ Sáu | Thứ Bảy |
| Anh | Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
| Mã số | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

***Tính thứ của ngày đầu năm***: Ngày 1 tháng Giêng năm y là thứ mấy ?

Thu = (A+B+C+1) mod 7

trong đó

A = 5(y−1) mod 4

B = 4(y−1) mod 100

C = 6(y−1) mod 400

Kết quả thu được là mã số của Thu (Gauss).

Thí dụ: Ngày 1 tháng Giêng năm 2016 là thứ mấy ?

Ta có,

y − 1 = 2016 − 1= 2015

A = 5⋅2015 mod 4 = 3

B = 4⋅2015 mod 100 = 60

C = 6⋅2015 mod 400 = 60

Thu = (A+B+C+1) mod 7 = (3+60+60+1) mod 7 = 5.

Trả lời : Ngày 1 tháng Giêng năm 2016 là Thứ Sáu

Hàm FirstWeekDay(int year)cho ta thu của ngày 1 / 1 năm year:

// The first week day of the year (Gauss)

// Ngay dau nam la thu may

// Sunday (Chu nhat) = 0, ..., Saturday (Thu Bay) = 6

int FirstWeekDay(int year) {

--year;

return (1+5\*(year % 4)+4\*(year % 100)+6\*(year % 400))%7;

}

***Tính thứ của ngày bất kì***: Ngày d tháng m năm y là thứ mấy ?

Thu = (T+H+d-1) mod 7

trong đó

T là thứ của ngày đầu năm

H là hệ số tháng m (xem trong bảng)

d là ngày

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | *Hệ số tháng* | | |
| Tháng | Anh | | Năm thường | Năm Nhuận |
| 1 | January | | 0 | 0 |
| 2 | February | | 3 | 3 |
| 3 | March | | 3 | 4 |
| 4 | April | | 6 | 0 |
| 5 | May | | 1 | 2 |
| 6 | June | | 4 | 5 |
| 7 | July | | 6 | 0 |
| 8 | August | | 2 | 3 |
| 9 | September | | 5 | 6 |
| 10 | October | | 0 | 1 |
| 11 | November | | 3 | 4 |
| 12 | December | | 5 | 6 |

Thí dụ: Ngày 17 tháng 5 năm 2016 là thứ mấy ?

Ta có,

T = 5

H = 2 (2016 là năm nhuận)

d = 17

Thu = (T+H+d − 1) mod 7 = (5+2+17 − 1) mod 7 = 2.

Trả lời: Ngày 17 tháng 5 năm 2016 là thứ Ba.

Hàm WeekDay(day,month,year) cho ta mã số thứ của ngày day, tháng month năm year theo cách tính của Gauss.

// Week day

int WeekDay(int day, int month, int year) {

return (FirstWeekDay(year)

+MONTH\_CODE[LeapYear(year)][month]+day-1) % 7;

}

Do ta có thể tính thứ của ngày bất kì nên thông tin

Mồng 1 tháng Giêng năm 1900 là thứ Hai.

sẽ là không cần thiết.

Có hai cách giải bài E19.

E19: cách thứ nhất

Ta duyệt đủ 100 năm, từ năm 1901 đến năm 2000. Với mỗi năm ta duyệt 12 tháng, từ tháng 1 đến tháng 12. Nếu ngáy 1 của tháng đó có mã thứ là 0 (Chủ Nhật) thì tăng biến đếm.

// So ngay chu nhat 1901 .. 2000

// Phuong an 1.

void E19() {

int sum = 0;

int year, month;

for (year = 1901; year <= 2000; ++year) {

for (month = 1; month <= 12; ++month)

if (WeekDay(1,month,year) == 0) ++sum;

}

cout << "\n Result: " << sum;

}

E19B: cách thứ hai

Ta dựa vào công thức tính thứ của ngày bất kì để tính trước thông tin sau đây:

Nếu biết ngày đầu năm là thứ mấy thì trong năm đó có bao nhiêu ngày đầu tháng là ngày Chủ Nhật. Số liệu này khác nhau theo năm thường và năm nhuận

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Thứ đầu năm | 0  Hai | 1  Ba | 2  Tư | 3  Năm | 4  Sáu | 5  Bảy | 6  CN |
| Năm thường | 2 | 2 | 2 | 1 | 3 | 1 | 1 |
| Năm nhuận | 3 | 2 | 1 | 2 | 2 | 1 | 1 |
| *Số ngày đầu tháng là Chủ Nhật* | | | | | | | |

Theo bảng này, nếu ngày đầu năm thường là thứ Hai (mã số 0) thì năm đó sẽ có 2 ngày đầu tháng rơi vào Chủ Nhật, nếu là năm nhuận thì sẽ có 3 ngày đầu tháng rơi vào Chủ Nhật.

Nhắc lại công thức tính thứ của ngày bất kì trong năm.

Thu = (T+H+d-1) mod 7

Vì chỉ cần tính thứ của ngày 1 mỗi tháng, tức là d = 1. Thay trị này vào công thức trên ta thu được công thức tính thứ của ngày đầu tháng.

Thu = (T+H) mod 7

trong đó T là thứ của ngày đầu năm, H là hệ số tháng. Ta cần tìm các ngày đầu tháng là Chủ Nhật (mã số 0). Ta có đẳng thức sau:

(T+H) mod 7 = 0

Từ đây suy ra tổng T+H là bội của 7, tức là T+H = 0, 7, 14, ….

Lần lượt thay T = 0..6 và cho H biến thiên theo 12 tháng bạn sẽ đếm được số lượng các ngày Chủ Nhật đầu tháng.

Nếu không muốn tính sẵn, bạn cũng có thể gọi hàm tính số ngày Chủ Nhật đầu tháng trong năm **year** như sau:

// So ngay chu nhat dau thang trong nam year

int NumOfSun(int year) {

int weekday, d, month;

bool leap = LeapYear(year);

weekday = FirstWeekDay(year); // Thu dau nam

d = 0;

for (month = 1; month <= 12; ++month)

if ((weekday + MONTH\_CODE[leap][month]) % 7 == 0)

++d;

return d;

}

Biết dữ liệu này ta chỉ cần duyệt các năm để lấy tổng các ngày Chủ Nhật đầu tháng.

// Phuong an 2

void E19B() {

int sum = 0, year;

for (int month = 1901; month <= 2000; ++month)

sum += NUM\_OF\_SUN[LeapYear(month)][FirstWeekDay(month)];

cout << "\n Result: " << sum;

}

Tổ chức chương trình

Ta xây dựng chương trình thực hiện các chức năng sau đây:

* Giải bài E19 theo 2 phương án: E19 và E19B.
* Hiển thị lịch tháng và năm

Đọc thêm

*D1. Có thể nhớ được hệ số tháng không?*

Có thể.

* Câu chú sau đây có thể giúp bạn nhớ hệ số tháng của năm thường:

“*Không* (0) *bán* (3) *bò* (3) *sáng* (6) *mười* *bốn* (1,4)

*sau* (6) *hai* (2) *năm* (5) *không* (0) *bán* (3) *nữa* (5)”

* Với năm nhuận:
* các tháng 1 và 2 có cùng hệ số như năm thường
* Các tháng còn lại (m = 3..12) được tính như sau:

*HS tháng m của năm nhuận* = (*HS tháng m của năm thường* + 1) mod 7.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tháng | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| HS năm thường | 0 | 3 | 3 | 6 | 1 | 4 | 6 | 2 | 5 | 0 | 3 | 5 |
| Cách nhớ | không | bán | bò | sáng | mười | bốn | sau | hai | năm | không | bán | nữa |
| HS năm nhuận | 0 | 3 | 4 | 0 | 2 | 5 | 0 | 3 | 6 | 1 | 4 | 6 |

*D2. Vì sao có hệ số tháng?*

Nếu biết ngày 1/1 của năm y là thứ mấy thì bạn hoàn toàn có thể tính được ngày d tháng m bất kì trong năm y là thứ mấy. Thật vậy, đầu tiên bạn tính xem ngày đó là ngày thứ bao nhiêu trong năm, ta gọi đó là *số thứ tự của ngày*. Thí dụ, ngày 5/4/1944 có số thứ tự S là bao nhiêu?

Ta cộng dồn số ngày của từng tháng, từ tháng 1 đến tháng 3, tạm ghi vào biến S. Ta có,

Tháng 1: 31 ngày, do đó S = 31

Tháng 2: 29 ngày (vì 1944 là năm nhuận), do đó S = 31 + 29 = 60

Tháng 3: 31 ngày, do đó S = 60 + 31 = 91.

Cộng thêm phần ngày của tháng 4 là 5 vào tổng trên ta thu được: S = 91 + 5 = 96.

Vậy 5/4/1944 là ngày thứ 96 (của năm 1944).

Ta lại biết ngày đầu năm 1944, tức là 1/1/1944 là thứ 7 (có mã số 6). Do tính tuần hoàn của tuần lễ (7 ngày) nên ta tính được

Thứ = (S – 1 + mã số ngày đầu năm) mod 7

Cụ thể là

Thứ của ngày 5/4/1944 = (96 – 1 + 6) mod 7 = 101 mod 7 = 3 (thứ tư)

Ta có công thức sau:

Thứ của ngày d/m/y = (S – 1 + T) mod 7

Trong đó S là số thứ tự của ngày d/m/y trong năm y

T là thứ của ngày đầu năm y.

Phép cộng dồn các ngày trong tháng cho ta tổng tích lũy (TTL), do đó ta cộng trước TTL cho các tháng theo công thức

S(m) = Tổng số ngày của các tháng trước tháng m = S(m-1)+T(m-1)

trong đó S(m) là TTL của tháng m, T(m) lá số ngày trong tháng m.

Lưu ý rằng trước tháng 1 không có tháng nào, do đó S(1) = 0 (xem bảng).

Vận dụng tính chất sau đây của phép toán modulo:

(a + b) mod n = ((a mod n) + (b mod n)) mod n

ta thay TTL của các tháng bằng số dư theo mod 7 ta có bảng mã số tháng theo mod 7 như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| *Tổng tích lũy và mã số tháng* | | | |
| Tháng | Số ngày | S | H = S mod 7 |
| 1 | 31 | 0 | 0 |
| 2 | 28/29 | 31 | 3 |
| 3 | 31 | 59/60 | 3/4 |
| 4 | 30 | 90/91 | 6/0 |
| 5 | 31 | 120/121 | 1/2 |
| 6 | 30 | 151/152 | 4/5 |
| 7 | 31 | 181/182 | 6/0 |
| 8 | 31 | 212/213 | 2/3 |
| 9 | 30 | 243/244 | 5/6 |
| 10 | 31 | 273/274 | 0/1 |
| 11 | 30 | 304/305 | 3/4 |
| 12 | 31 | 334/335 | 5/6 |

*Chú thích:* Vì tháng 2 năm nhuận có 29 ngày nên kể từ tháng 3 trở đi, hệ số của các tháng có khác nhau. Các số a/b cho biết a tính theo năm thường, b tính thao năm nhuận.

Theo bảng này ta có thể dễ dàng tính được số thứ tự S’ (theo mod 7) của ngày 5/4/1944 như sau:

Vì năm 1944 nhuận nên H(4) = 0.

5/4: S’ = 0 + 5 = 5. Ta thấy 96 mod 7 = 5.

Do thứ(1/1/1944) = 6 nên

Thứ(5/4/1944) = (6+5-1) mod 7 = 10 mod 7 = 3 (Thứ Tư).

*D3. Qui luật biến đổi của thứ.*

Mỗi năm có 365 ngày, năm nhuận có 366 ngày. Mặt khác, 365 mod 7 = 1, 366 mod 7 = 2. Từ đây suy ra các thứ của cùng một ngày trong 2 năm liên tiếp sẽ lệch nhau 1 hoặc 2 ngày. Thí dụ,

Ngày 1/1/2000 là Thứ Bảy thì 1/1/2001 là Thứ Hai vì năm 2000 là năm nhuận.

Ngày 1/1/2002 sẽ là Thứ Ba, …

Tuy nhiên,

Nếu bạn đón sinh nhật 5/6/2000 vào Thứ Hai thì năm sau, 2001 bạn sẽ đón sinh nhật vào Thứ Ba chứ không phải là Thứ Tư. Vì sao vậy? Vì ngày 29/2 dôi vào tháng 2 không ảnh hưởng đến các tháng sau tháng 2. Từ 5/6/2000 đến 5/6/2001 vẫn có đúng 365 ngày.

Chỉ những ngày trong tháng 1 và 2 mới bị lệch 2 đơn vị.

## Problem 20. [Factorial digit sum](https://projecteuler.net/problem=20)

*n*! means *n* × (*n* − 1) × ... × 3 × 2 × 1. For example, 10! = 10 × 9 × ... × 3 × 2 × 1 = 3628800, and the sum of the digits in the number 10! is 3 + 6 + 2 + 8 + 8 + 0 + 0 = 27.

Find the sum of the digits in the number 100!

**Bài 20. Tổng các chữ số của giai thừa**

Ta biết giai thừa của số tự nhiên *n* được tính theo công thức *n*! = *n* × (*n* − 1) × ... × 3 × 2 × 1. Thí dụ, 10! = 10 × 9 × ... × 3 × 2 × 1 = 3628800, và tổng các chữ số của 10! là

3 + 6 + 2 + 8 + 8 + 0 + 0 = 27.

Hãy tính tổng các chữ số của 100!

Thuật toán

Trước hết, để tính n! cần cài đặt phép nhân một số lớn có len chữ số với một số nguyên x ≤ n. Gọi hàm này là Mult(x). Kết quả sẽ ghi trong biến tổng thể kiểu mảng num[0..len-1] dưới dạng ngược. Gỉa sử bạn cần tính 11! = 11×10!. Sau khi tính được 10! = 3628800 bạn sẽ phải lấy số này nhân tiếp với 11. Số 3628800 sẽ được ghi ngược trong mảng num và quá trình tính toán sẽ được giải trình như sau:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |
| num | 0 | 0 | 8 | 8 | 2 | 6 | 3 |  |  | 10! = 3628800 |
| num[i] × 11 | 0 | 0 | 8 | 6 | 1 | 9 | 9 | 3 |  | 11! = 10! × 11 = 39916800 |
| số nhớ | 0 | 0 | 8 | 9 | 3 | 6 | 3 |  |  |  |

Khi nhân số num[0..6] với 11 bạn làm phép nhân từng chữ số với 11 như sau:

Khởi trị số nhớ carr = 0;

num[0] × 11 + carr = 0 × 11 + 0 = 0: viết num[0] = 0; số nhớ carr = 0;

num[1] × 11 + carr = 0 × 11 + 0 = 0: viết num[1] = 0; số nhớ carr = 0;

num[2] × 11 + carr = 8 × 11 + 0 = 88: viết num[2] = 8, số nhớ carr = 8;

num[3] × 11 + carr = 8 × 11 + 8 = 96: viết num[3] = 6, số nhớ carr = 9;

num[4] × 11 + carr = 2 × 11 + 9 = 31: viết num[4] = 1, số nhớ carr = 3;

num[5] × 11 + carr = 6 × 11 + 3 = 69: viết num[5] = 9, số nhớ carr = 6;

num[6] × 11 + carr = 3 × 11 + 6 = 39 viết num[6] = 9, số nhớ carr = 3;

Đặt num[7] = carr = 3.

Như vậy num[0..6] × 11 = (0,0,8,8,2,6,3) × 11 = (0,0,8,6,1,9,9,3) = num[0..7], chiều dài len được tăng thêm 1.

Sau đó bạn viết thuật toán tính giai thừa để ghi kết quả vào mảng num[0..len-1]. Cuối cùng tính tổng các phần tử của num.

Chương trình DevCPP

## Problem 21. [Amicable numbers](https://projecteuler.net/problem=21)

Let d(*n*) be defined as the sum of proper divisors of *n* (numbers less than *n* which divide evenly into *n*). If d(*a*) = *b* and d(*b*) = *a*, where *a* ≠ *b*, then *a* and *b* are an amicable pair and each of *a* and *b* are called amicable numbers. For example, the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110; therefore d(220) = 284. The proper divisors of 284 are 1, 2, 4, 71 and 142; so d(284) = 220.

Evaluate the sum of all the amicable numbers under 10000.

**Bài 21. Các số thân thiện**

Kí hiệu d(*n*) là tổng các ước thực sự của số nguyên dương n (ND. a là ước thực sự của n nếu n chia hết cho a và a < n). Nếu d(*a*) = *b,* d(*b*) = *a* và *a* ≠ *b*, thì ta gọi a và b là *cặp số thân thiện* và gọi a và b là *hai số thân thiện*. Thí dụ, các ước thực sự của 220 là 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 và 110. Do đó, d(220) = 284. Các ước thực sự của 284 là 1, 2, 4, 71 và 142. Như vậy, d(284) = 220 và do đó (220, 284) là một cặp số thân thiện và 220 và 284 là hai số thân thiện.

Tìm tổng của các số thân thiện dưới 10000.

Thuật toán

Duyệt theo đầu bài, nếu d(a) = b và d(b) = a và a ≠ b thì cộng dồn a và b vào tổng. Với mỗi số nguyên dương a có không quá 1 số thân thiện b, do đó, để tránh tính hàm d hai lần cho mỗi cặp số ta dùng một mảng bit để đánh dấu các số đã tìm được. Trong thí dụ của đầu bài, khi tìm được 220 và 284 ta đánh dấu cả hai số này, do đó khi xét đến 284 ta bỏ qua.

Để tính tổng các ước của số n ta lưu ý rằng nếu n = a⋅b thì n có đồng thời 2 ước là a và b. Hai ước này bằng nhau khi và chỉ khi n là số chính phương tức là khi và chỉ khi n = x2.

## Problem 22. [Names scores](https://projecteuler.net/problem=22)

Using [names.txt](https://projecteuler.net/project/resources/p022_names.txt) (right click and 'Save Link/Target As...'), a 46K text file containing over five-thousand first names, begin by sorting it into alphabetical order. Then working out the alphabetical value for each name, multiply this value by its alphabetical position in the list to obtain a name score.

For example, when the list is sorted into alphabetical order, COLIN, which is worth 3 + 15 + 12 + 9 + 14 = 53, is the 938th name in the list. So, COLIN would obtain a score of 938 × 53 = 49714.

What is the total of all the name scores in the file?

**Bài 22. Điểm của tên**

Sử dụng [names.txt](https://projecteuler.net/project/resources/p022_names.txt) (nhấn chuột phải rồi 'Save Link/Target As...'), bạn sẽ nhận được một file text dung lượng 46K chứa trên năm nghìn tên người. Bạn cần sắp (tăng) dãy tên này theo thứ tự từ điển. Sau đó với mỗi tên trong dãy đã sắp, bạn tính điểm của tên đó theo công thức sau:

(điểm của tên S) = (giá trị của tên S) × (số thứ tự của tên S trong dãy đã sắp, tính từ 1)

(giá trị của tên S) = (tổng các giá trị của các chữ cái trong S)

giá trị của chữ cái : ‘A’ = 1; ‘B’ = 2, …, ‘Z’ = 26

Thí dụ, COLIN đứng thứ 938 (tính từ 1) trong dãy sắp tăng theo thứ tự từ điển. Giá trị của COLIN là 3(C) + 15(O) + 12(L) + 9(I) + 14(N) = 53. Vậy COLIN sẽ có điểm là 938 × 53 = 49714.

Hãy cho biết tổng giá trị của tất cả các tên trong file?

Thuật toán

Bài này nhằm rèn luyện các kĩ năng tách từ trong văn bản, sắp dữ liệu và mã hóa dữ liệu.

Bạn nên xem trước file po22\_names.txt để biết cấu trúc của file. Dưới đây là đoạn đầu tiên của file

"MARY","PATRICIA","LINDA","BARBARA","ELIZABETH","JENNIFER","MARIA","SUSAN","MARGARET","DOROTHY","LISA","NANCY","KAREN","BETTY","HELEN","SANDRA","DONNA","CAROL","RUTH","SHARON","MICHELLE","LAURA","SARAH","KIMBERLY","DEBORAH","JESSICA","SHIRLEY","CYNTHIA","ANGELA","MELISSA","BRENDA","AMY","ANNA","REBECCA","VIRGINIA","KATHLEEN","PAMELA","MARTHA","DEBRA","AMANDA","STEPHANIE","CAROLYN","CHRISTINE","MARIE","JANET","CATHERINE","FRANCES","ANN","JOYCE","DIANE","ALICE","JULIE","HEATHER","TERESA","DORIS","GLORIA","EVELYN","JEAN","CHERYL","MILDRED","KATHERINE","JOAN","ASHLEY","JUDITH","ROSE","JANICE","KELLY","NICOLE","JUDY","CHRISTINA","KATHY","THERESA","BEVERLY","DENISE","TAMMY","IRENE","JANE","LORI","RACHEL","MARILYN","ANDREA","KATHRYN","LOUISE","SARA","ANNE","JACQUELINE","WANDA","BONNIE","JULIA","RUBY","LOIS","TINA","PHYLLIS","NORMA","PAULA","DIANA","ANNIE","LILLIAN","EMILY","ROBIN","PEGGY","CRYSTAL","GLADYS","RITA","DAWN","CONNIE","FLORENCE","TRACY","EDNA","TIFFANY","CARMEN","ROSA","CINDY","GRACE","WENDY","VICTORIA","EDITH","KIM","SHERRY","SYLVIA","JOSEPHINE","THELMA","SHANNON","SHEILA"...

Bạn nhận thấy

Tên người được viết IN HOA bằng chữ cái tiếng Anh, đặt trong hai nháy kép, các tên cách nhau qua dấu phảy (,).

Trình tự xử lí sẽ như sau:

1. Tách từ: Mở file, đọc từng chữ cái rồi ghép thành từng tên để ghi vào mảng names. Ghi nhận n là số người.
2. Sắp tăng mảng names[1..n] theo thứ tự từ điển.
3. Duyệt dãy sắp tăng để tính tổng điểm.

Để tách từ ta dùng một biến trạng thái q với 2 trạng thái

q = 0: đang duyệt các kí tự ngoài tên, tức là duyệt các kí tự không phải là chữ cái như dấu phảy (,), dấu nháy ("), dấu kết dòng...

q = 1: đang duyệt trong tên, tức là duyệt các kí tự IN HOA (tiếng Anh, A..Z)

Bạn nên sắp names theo chỉ số và vận dụng quick sort cho nhanh.

Bạn cũng có thể dùng cây nhị phân. Sau khi đọc mỗi tên bạn xen tên đó vào cây nhị phân.

Hai xâu lí tự u, v trong bài này được so sánh theo thứ tự từ điển như sau:

* Thêm vào đầu phải xâu ngắn các dấu cách cho hai xâu có cùng chiều dài m.
* Duyệt từ kí tự đầu đến cuối hai xâu:
* Nếu gặp cặp kí tự đầu tiên khác nhau, u[i] ≠ v[i] thì xâu nào mang kí tự nhỏ hơn sẽ nhỏ hơn.
* Nếu mọi cặp kí tự đều bằng nhau thì hai xâu bằng nhau.

Hàm

strcmp(u, v)

trong C++ cho ra giá trị

0, nếu u = v

-1, nếu u < v

1, nếu u > v.

## Problem 23. [Non-abundant sums](https://projecteuler.net/problem=23)

A perfect number is a number for which the sum of its proper divisors is exactly equal to the number. For example, the sum of the proper divisors of 28 would be 1 + 2 + 4 + 7 + 14 = 28, which means that 28 is a perfect number.

A number n is called deficient if the sum of its proper divisors is less than n and it is called abundant if this sum exceeds n.

As 12 is the smallest abundant number, 1 + 2 + 3 + 4 + 6 = 16, the smallest number that can be written as the sum of two abundant numbers is 24. By mathematical analysis, it can be shown that all integers greater than 28123 can be written as the sum of two abundant numbers. However, this upper limit cannot be reduced any further by analysis even though it is known that the greatest number that cannot be expressed as the sum of two abundant numbers is less than this limit.

Find the sum of all the positive integers which cannot be written as the sum of two abundant numbers.

**Bài 23. Tổng các số lệch**

*Số hoàn thiện* là số có tổng các ước thực sự bằng chính số đó. Thí dụ, tổng các ước thực sự của số 28 là 1 + 2 + 4 + 7 + 14 = 28, do đó 28 là một *số hoàn thiện*.

Số *n* được gọi là số *hụt* nếu tổng các ước thực sự của *n* nhỏ thua *n.* Số *n* được gọi là số *dôi* nếu tổng các ước thực sự của *n* lớn hơn *n.*

Vì 12 là số dôi nhỏ nhất, 1 + 2 + 3 + 4 + 6 = 16, nên số nhỏ nhất có thể viết dưới dạng tổng của hai số dôi là 24. Theo các phân tích toán học, có thể chứng minh rằng mọi số nguyên lớn hơn 28123 có thể được viết dưới dạng tổng của hai số dôi. Người ta vẫn chưa hạ được giới hạn này, mặc dù dễ dàng thấy rằng số lớn nhất không thể viết dưới dạng tổng của hai số dôi phải nhỏ hơn giới hạn này.

Hãy tính tổng của các số nguyên dương không thể viết dưới dạng tổng của hai số dôi.

Thuật toán

Ta tạm gọi số không thể viết dưới dạng tổng của hai số dôi là số *lệch.* Bài này yêu cầu tìm tổng các số lệch trong khoảng 1..28123. Theo đề bài, mọi số lệch đều không lớn hơn giới hạn 28123, do đó ta chỉ cần duyệt các số từ 1 đến 28123 để tìm ra các số lệch rồi lấy tổng của chúng.

Gọi d(n) là hàm tính tổng các ước thực sự của số nguyên dương n. Trước hết ta tìm các số dôi k là những số thỏa điều kiện d(k) > k và ghi vào một mảng. Ta thấy nếu x và y là hai số dôi thì số z = x+y sẽ là số được biểu diễn dưới dạng tổng hai số dôi. Như vậy, khi có danh sách các số dôi thì ta tính tổng của các cặp số dôi và đánh dấu các số này. Các số còn lại trong danh sách sẽ là các số lệch.

Theo cách khác, ta có thể kiểm tra trực tiếp một số x có là số lệch hay không như sau. Giả sử ta đã có danh sách D các số dôi. Với mỗi số dôi k trong danh sách D, nếu x-k không phải là số dôi, tức là x-k không có trong danh sách D thì x là số lệch. Nếu ta đã đánh dấu các số dôi vào mảng bít mark với ý nghĩa, mark[i] = 1 khi và chỉ khi i là số dôi thì ta có thể kiểm tra trực tiếp x-d có là số dôi hay không chỉ bằng một phép so sánh mark[x-d] = 1. Như vậy, các bit 1 trong mảng bit mark đóng vai trò của danh sách các số dôi D.

Gợi ý nghiên cứu

Trang web [abundant number article on Wolfram](http://mathworld.wolfram.com/AbundantNumber.html), cho biết rằng mọi số lớn hơn 20161 đều có thể biểu diễn dưới dạng tổng của hai số dôi. Kiểm tra điều này không khó. Bạn thử sức xem sao. Nếu làm được điều này bạn có thể hạ thấp giới hạn đã nói trong đề bài.

## Problem 24. Lexicographic permutations

A permutation is an ordered arrangement of objects. For example, 3124 is one possible permutation of the digits 1, 2, 3 and 4. If all of the permutations are listed numerically or alphabetically, we call it lexicographic order. The lexicographic permutations of 0, 1 and 2 are:

012   021   102   120   201   210

What is the millionth lexicographic permutation of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9?

**Bài 24. Các hoán vị theo thứ tự từ điển**

Hoán vị là một phân bố thứ tự của các đối tượng. Thí dụ, 3124 là một hoán vị của các số 1, 2, 3 và 4. Ta có thể sắp toàn bộ các hoán vị theo trật tự số (ND: tức là coi mỗi hoán vị như là một số rồi sắp các số này từ nhỏ đến lớn) hoặc theo thứ tự từ điển. Các hoán vị theo thứ tự từ điển của 0, 1 và 2 là:

012   021   102   120   201   210

Hãy cho biết hoán vị từ điển thứ một triệu của các chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8 và 9?

Thuật toán

Bài giải dưới đây giả thiết rằng n phần tử của các hoán vị được mã số từ 0 đến n-1 và các hoán vị trong dãy sắp tăng theo thứ tự từ điển cũng được mã số từ 0 đến n!-1.

Có hai phương án giải bài này, mỗi phương án được xây dựng trên phương thức sinh hoán vị. Phương án thứ nhất, E24A, dựa trên thuật toán sinh tuần tự các hoán vị, từ hoán vị đơn vị, tức là hoán vị nhỏ nhất, E = 0…(n-1) đến hoán vị lớn nhất M = (n-1)…0. Phương án này do Dijkstra đề xuất trong cuốn “*A method of programming*”. Phương án thứ hai, E24B, dựa trên thuật toán tính trực tiếp hoán vị khi biết số thứ tự của nó trong dãy sắp tăng theo từ điển.

Phương án E24A

Dijkstra đề xuất thuật toán sinh toàn bộ hoán vị của n phần tử, gọi là các *phần tử cơ sở,* mã số 0, 1, ..., n-1 theo thứ tự từ điển như sau:

1. Khởi trị hoán vị đơn vị h[0..n-1] = (0,1,2,...,n-1). Đây là hoán vị nhỏ nhất trong dãy các hóan vị sắp theo thứ tự từ điển.
2. Lặp đến khi tạo ra hoán vị lớn nhất h[01..n-1] = (n-1,...,1, 0). Mỗi lần lặp gọi hàm Next để sinh ra hoán vị sát sau hoán vị h.

Hàm Next hoạt động như sau.

Input: Hoán vị h[0..n-1] trong dãy hoán vị sắp theo thứ tự từ điển.

Output: Sửa h[0..n-1] để thu được hoán vị sát sau hoán vị hiện có.

Thí dụ, nếu h[0..3] = (2,1,3,0) thì sau khi gọi hàm Next ta thu được h[0..3] = (2,3,0,1).

Method

1. Duyệt ngược h tìm điểm gãy i thỏa: h[i] < h[i+1]. Nếu không có điểm gãy chứng tỏ h là hoán vị lớn nhất, h = (n-1,…,0). Dừng thuật toán.
2. Duyệt ngược h lần thứ hai để tìm điểm vượt j thỏa: h[j] > h[i].
3. Đổi chỗ h[i] và h[j].
4. Lật lại đoạn h[i+1..n-1].

Hàm Next đòi hỏi hai lần duyệt mảng n phần tử, do đó có độ phức tạp tính toán cỡ n (O(n)).

Theo phương án E24A, với n = 10 phần tử 0..9, để tìm hoán vị thứ t = 1000000 ta xuất phát từ hoán vị đơn vị h = 0123456789 (có số thứ tự 0), sau đó gọi hàm Next 999999 lần sẽ thu được kết quả.

Phương án E24A đòi hỏi t lần tính hàm Next, mỗi lần tính hàm Next ta cần cỡ n lần duyệt mảng. Tổng hợp lại, phương án này có độ phức tạp tính toán cỡ t⋅n, trong đó t là số thứ tự của hoán vị cần tìm, n là số phần tử cơ sở. Với n = 10 và t = 1 triệu thì bài giải theo phương án E24A đòi hỏi cỡ 10 triệu phép toán.

Chú ý rằng, trong các ứng dụng đòi hỏi xét toàn bộ các hoán vị thì thuật toán Dijkstra luôn tỏ ra là đơn giản và trong sáng.

Phương án E24B

Về bản chất, phương án này dựa trên thủ tục sinh hoán vị theo tiếp cận đệ qui. Ta mường tượng các phần tử cơ sở để tạo thành hoán vị được ghi trên các tấm bìa, với n = 10 ta có 0 tấm bìa nh] sau:

[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]

Ta tạm lấy ra tấm bìa [b] nào đó, sẽ còn lại n-1 tấm bìa. Dùng n-1 tấm bìa còn lại này ta sẽ xây dựng được (n-1)! hoán vị và giả sử ta đã sắp các hoán vị này theo thứ tự từ điển.

Nếu ta ghép tấm bìa [b] vào mỗi hoán vị H trong dãy nói trên thì sẽ thu được (n-1)! hoán vị mới, dạng [b]H. Các hoán vị này có chung phần tử đầu tiên là [b] và do đó tạo thành một lớp, ta gọi là lớp b.

Cho b biến thiên từ 0 đến n-1 ta sẽ thu được n lớp, lần lượt là lớp 0, lớp 1, …, lớp n-1. Bạn quan sát bảng minh họa với n = 4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Phân hoạch của các hoán vị n = 4* | | | | | | | |
| Lớp 0 | | Lớp 1 | | Lớp 2 | | Lớp 3 | |
| 0 | **0123** | 6 | **1023** | 12 | **2013** | 18 | **3012** |
| 1 | **0132** | 7 | **1032** | 13 | **2031** | 19 | **3021** |
| 2 | **0213** | 8 | **1203** | 14 | **2103** | 20 | **3102** |
| 3 | **0231** | 9 | **1230** | 15 | **2130** | 21 | **3120** |
| 4 | **0312** | 10 | **1302** | 16 | **2301** | 22 | **3201** |
| 5 | **0321** | 11 | **1320** | 17 | **2310** | 23 | **3210** |

Sau khi phân hoạch các hoán vị thành n lớp dựa theo phần tử đầu tiên của mỗi hoán vị. Theo bảng trên, với n = 4, ta thu được 4 lớp là lớp 0, lớp 1, lớp 2 và lớp 3.

Ta luôn luôn giả thiết là các hoán vị của n phần tử 0, …, n-1 được sắp (tăng) theo thứ tự từ điển. Dựa vào tính chất phân hoạch hoán vị theo lớp ta suy ra được thủ tục sau đây. Nếu biết số thứ tự t của một hoán vị nào đó trong dãy các hoán vị của n phần tử thì ta có thể xác định được ngay hai thông số quan trọng sau đây:

* Số thứ tự của lớp chứa hoán vị t: b = t div (n-1)!
* Số thứ tự riêng của hoán vị t trong lớp b: r = t mod (n-1)!

Lớp b bao gồm (n-1)! hoán vị của n-1 phần tử còn lại ghép sau phần tử b.

Trong lớp này, các hoán vị cũng được sắp và có số thứ tự riêng tính từ 0 đến (n−1)! - 1, trong đó hoán vị t xuất hiện tại vị trí r.

Sau khi tính được b và r ta lấy tấm bìa b để xuất ra kết quả và tiếp tục tính toán với giá trị r trên dãy (n-1) hoán vị còn lại (không chứa bìa b).

Chú ý rằng thuật ngữ “lớp b” cần được hiểu là số thứ tự của lớp đó. Thí dụ, lúc đầu ta có 10 phần tử như sau:

[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]

sau khi bỏ đi bìa [4] thì 9 phần tử còn lại sẽ là:

[0], [1], [2], [3], [5], [6], [7], [8], [9]

Lúc này lớp thứ b = 5 (tính từ 0) sẽ ứng với tấm bìa [6].

Việc xác định phần tử cơ sở dựa trên số hiệu lớp là dễ dàng. Ta chỉ cần duyệt từ đầu dãy đến cuối dãy chứa các phần tử cơ sở còn lại để chọn ra phần tử đứng thứ b tính từ 0 là được. Việc loại một phần tử ra khỏi danh sách các phần tử cơ sở cũng rất dễ. Ta đánh dấu 1 nếu phần tử chưa bị loại, là 0 nếu đã bị loại. Mảng used trong chương trình đảm nhận nhiệm vụ này.

Sơ đồ cho thuật toán OrdToPer khi đó sẽ như sau:

Thuật toán OrdToPer

Chức năng: Xác định hoán vị thứ t

Input:

- n phần tử cơ sở: 0, 1, …, n-1

- số thứ tự t (tính từ 0) của hoán vị cần tìm trong dãy sắp tăng theo thứ tự từ điển các hoán vị của n phần tử cơ sở.

Output:

- Hoán vị h[0..n-1] có số thứ tự t.

Method

P1. Tính các giai thừa fac[k] = k!, k = 0,…n-1.

P2. Lặp với i = 0 đến n-1

P2.1. Tính v = t div fac[n - i - 1]

P2.2. h[i] = phần tử cơ sở có thứ tự v

P2.3. t = t mod fac[n - i - 1]

end OrdToPer

Điều thú vị là ta có thể cài đặt thuật toán PerToOrd là thuật toán đối xứng với thuật toán OrdToPer. Thuật toán PerToOrd cho ra số thứ tự của hoán vị h. Để cho tiện, tham biến vào được viết dưới dạng string. Thí dụ, để tìm số thứ tự của hoán vị h = 5320178946 ta viết PerToOrd(“5320178946”).

Chương trình được xây dựng với các chức năng sau đây:

* Hàm Next sinh hoán vị tiếp theo sau hoán vị h.
* Thuật toán E24A
* Hàm OrdToPer xác định hoán vị h từ số thứ tự t
* Hàm PerToOrd cho ra số thứ tự của hoán vị h

Ngoài ra còn một số hàm tiện ích

* Hàm SetFac tính trước các giai thừa từ 0 đến 9.
* Hàm GetElem(b) cho ra phần tử cơ sở khi biết thứ tự b
* Hàm GetOrd(e) cho ra thứ tự của phần tử cơ sở e.

Phương án E24B chỉ làm việc với n phần tử cơ sở. Khi tính mỗi h[i] cần gọi hàm GetElem với n lần duyệt, tổng hợp lại ta có độ phức tạp tính toán cỡ n2. Với n = 10 ta chỉ cần cỡ 100 phép toán div và mod các số 4 bytes.

## Problem 25. 1000-digit Fibonacci number

The Fibonacci sequence is defined by the recurrence relation:

F*n* = F*n*−1 + F*n*−2, where F1 = 1 and F2 = 1.

Hence the first 12 terms will be:

F1 = 1, F2 = 1, F3 = 2, F4 = 3, F5 = 5, F6 = 8,

F7 = 13, F8 = 21, F9 = 34, F10 = 55, F11 = 89, F12 = 144

The 12th term, F12, is the first term to contain three digits.

What is the index of the first term in the Fibonacci sequence to contain 1000 digits?

**Bài 25. Số Fibonacci 1000 chữ số**

Dãy số Fibonacci được định nghĩa đệ qui như sau:

F1 = 1, F2 = 1.

F*n* = F*n*−1 + F*n*−2, n > 2.

Theo đó 12 số đầu tiên trong dãy sẽ là:

F1 = 1, F2 = 1, F3 = 2, F4 = 3, F5 = 5, F6 = 8,

F7 = 13, F8 = 21, F9 = 34, F10 = 55, F11 = 89, F12 = 144

Số thứ 12 trong dãy, F12 = 144, là số đầu tiên có 3 chữ số.

Hãy xác định trong dãy Fibonacci chỉ số nhỏ nhất n thỏa tính chất: Fn chứa 1000 chữ số?

Thuật toán

Kí hiệu Len(a, x) là số chữ số dùng để biểu diễn số tự nhiên x trong hệ đếm a. Thí dụ,

* Len(10, 1024) = 4: Số 1024 trong hệ thập phân có 4 chữ số.
* Len(2, 1024) = 4: Số 1024 trong hệ nhị phân có 11 chữ số, 1024 = 100000000002.

Theo định nghĩa của hàm số logarit ta có

trong đó hàm cho ta giá trị nguyên (gọi là trần) sát trên số thực v, , .

Ngoài ra, dễ thấy, mọi số thực dương r đều nằm giữa hai lũy thừa liên tiếp của 10, cụ thể là

10k ≤ r ≤ 10k+1

Số Fibonacci thứ n, Fn được đánh giá như sau:

ϕn-2 ≤ Fn ≤ ϕn-1, ϕ =

(Knuth D., *The Art of Programming*, V. 1, Addison-Wesley, 1968)

Mặt khác, do Fn có 1000 chữ số nên 10999 ≤ Fn < 101000.

Đặt a = , kết hợp với 999 ≤ lg(Fn) < 1000 ta thu được hệ thức a+1 ≤ n ≤ a+2.

Giải ra ta tính được a = 4780.19, do đó 4781.19 ≤ n ≤ 4782.19, n nguyên. Vậy n = 4782.

## Problem 26. Reciprocal cycles

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

|  |  |  |
| --- | --- | --- |
| 1/2 | = | 0.5 |
| 1/3 | = | 0.(3) |
| 1/4 | = | 0.25 |
| 1/5 | = | 0.2 |
| 1/6 | = | 0.1(6) |
| 1/7 | = | 0.(142857) |
| 1/8 | = | 0.125 |
| 1/9 | = | 0.(1) |
| 1/10 | = | 0.1 |

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that 1/7 has a 6-digit recurring cycle.

Find the value of *d* < 1000 for which 1/*d* contains the longest recurring cycle in its decimal fraction part.

**Bài 26. Phần thập phân tuần hoàn**

Phân số đơn vị là phân số có tử số bằng 1. Dạng thập phân của các phân số đơn vị với các mẫu số từ 2 đến 10 là:

|  |  |  |
| --- | --- | --- |
| 1/2 | = | 0.5 |
| 1/3 | = | 0.(3) |
| 1/4 | = | 0.25 |
| 1/5 | = | 0.2 |
| 1/6 | = | 0.1(6) |
| 1/7 | = | 0.(142857) |
| 1/8 | = | 0.125 |
| 1/9 | = | 0.(1) |
| 1/10 | = | 0.1 |

trong đó 0.1(6) có nghĩa là 0.166666... với phần tuần hoàn 1 chữ số (là 6). Như vậy, ta thấy  1/7 có phần tuần hoàn gồm 6 chữ số (là 142857).

Hãy tìm số *d* < 1000 thỏa điều kiện: dạng thập phân của 1/*d* chứa phần tuần hoàn dài nhất.

Thuật toán

Phép chia a/b tạo ra phần tuần hoàn khi và chỉ khi dãy số dư bị lặp từ lần chia thứ i nào đó. Nếu ta sử dụng mảng res đánh dấu các lần chia như sau, res[r] = i có nghĩa là tại lần chia thứ i ta gặp số dư r. Nếu đến lần chia thứ j > i ta gặp lại số dư r thì ta có đoạn lặp dài j-i.

Ta có thêm nhận xét sau: vì số dư luôn nhỏ hơn số chia nên 1/d sẽ có tối đa d số dư khác nhau là 0, 1, ..., d-1.

Nhận xét này giúp ta khai báo kích thước mảng res là 1000 để chứa đủ các số dư từ 0 đến 999. Ngoài ra, nhận xét này gợi ý cho ta tổ chức duyệt từ d lớn (999) đến 2. Nếu gặp đoạn lặp có chiều dài d-1 thì ta dừng.

## Problem 27. Quadratic primes

Euler discovered the remarkable quadratic formula:

*n*² + *n* + 41

It turns out that the formula will produce 40 primes for the consecutive values *n* = 0 to 39. However, when *n* = 40, 402 + 40 + 41 = 40(40 + 1) + 41 is divisible by 41, and certainly when *n* = 41, 41² + 41 + 41 is clearly divisible by 41.

The incredible formula  *n*² − 79*n* + 1601 was discovered, which produces 80 primes for the consecutive values *n* = 0 to 79. The product of the coefficients, −79 and 1601, is −126479.

Considering quadratics of the form:

*n*² + *an* + *b*, where |*a*| < 1000 and |*b*| < 1000

where |*n*| is the modulus/absolute value of *n* e.g. |11| = 11 and |−4| = 4

Find the product of the coefficients, *a* and *b*, for the quadratic expression that produces the maximum number of primes for consecutive values of *n*, starting with *n* = 0.

**Bài 27. Hàm bậc hai sinh số nguyên tố**

Euler phát hiện hàm bậc hai tuyệt vời:

*E(n) = n*² + *n* + 41

Hàm này sinh ra 40 số nguyên tố với các giá trị liên tiếp n = 0 đến 39. Tuy nhiên, khi n = 40 thì 402 + 40 + 41 = 40(40 + 1) + 41 = 40⋅41 + 41 = 41⋅41 lại chia hết cho 41 và tất nhiên khi n = 41 thì 41² + 41 + 41 cũng chia hết cho 41.

Một hàm bậc hai khác,

F(n) = *n*² − 79*n* + 1601

cũng từng được khảo cứu, có thể sinh ra 80 số nguyên tố ứng với các giá trị liên tiếp n = 0 đến 79. Tích của các hệ số, −79 và 1601, là −126479.

Xét hàm bậc hai dạng:

*H(n) = n*² + *an* + *b*

trong đó |*a*| < 1000 và |*b*| < 1000, |*x*| là trị tuyệt đối của số (nguyên) *x,* thí dụ, |11| = 11 và |−4| = 4.

Hãy tìm tích của các hệ số a và b trong H(n) để có thể nhận được tối đa số nguyên tố ứng với các giá trị liên tiếp của n từ 0 trở đi.

Thuật toán

Xét hàm

*H(n)* = *n*² + *an* + *b*

Dễ thấy hàm H là tổng quát hóa của hai hàm E và F. Khi thay a = 1, b = 41 vào hàm H ta thu được hàm E. Thay a = -79, b = 1601 vào hàm H ta thu được hàm F.

Do giới hạn của a và b trong khoảng -1000 .. 1000 nên ta gắng xác định các hệ số a và b sao cho, chí ít hàm H cũng “tốt” như hàm E, nghĩa là với mọi n = 0.. m ta có H(n) là số nguyên tố và m ≥ 39.

Ta có H(0) = b, do đó, theo yêu cầu của đầu bài, b phải là số nguyên tố trong khoảng từ 2 đến 997.

H(1) = 1+a+b, do đó, theo yêu cầu của đầu bài, a+b+1 phải là số nguyên tố, tức là a phải là số lẻ.

Để H(n) là số nguyên tố, ta phải xét các trị n thòa điều kiện H(n) > 1.

Sơ đồ đơn giản đầu tiên là như sau:

Với mỗi số nguyên tố b từ 3 đến 997

Với mỗi số lẻ a từ -999 đến 999

Với mỗi n = 0 … maxn

nếu H(n) không phải là số nguyên tố thì cập nhật nmax

và dừng thuật toán;

Ta có thể dùng Sàng Eratosthenes để sinh trước m số nguyên tố. Trước mắt ta sinh các số nguyên tố trong khoảng 1000 để dùng khi duyệt b.

Để kiểm tra tính nguyên tố của H(n) ta cần biết giá trị lớn nhất của hàm. Ta tạm giả thiết là với n = 0..100, mọi H(n) đều nguyên tố. Khi đó với maxa = maxb = 1000, maxn = 100 ta tính được

max(H(n)) = *n*² + *an* + *b =* 10000 + 100000 + 1000 = 111000

Khảo sát chi tiết bằng chương trình máy tính ta có thể xác định được

max(H(n)) = 109699.

Như vậy ta cần dùng Sàng Eratosthenes để sinh trước các số nguyên tố trong khoảng 110000 để dùng khi duyệt b và để kiểm tra tính nguyên tố của H(n). Ta dùng mảng bit primes để đánh dấu các số nguyên tố (primes[i] = 1 nếu i nguyên tố). Ngoài ra, ta lấy ra m số nguyên tố < 1000 và ghi vào mảng s[0..m-1] để duyệt b.

Với mỗi b ta cần xác định giới hạn của a để H(n) > 1. Ta có H(1) = a+b+1 > 1 tức là a > -b.

Với mỗi cặp trị a, b ta cũng không cần duyệt hết mọi khả năng của n. Với n làm cho H(n) < 2 ta bỏ qua ngay.

Khi tính H(n), để tránh phép nhân ta biến đổi như sau:

Khởi trị H = b; ứng với n = 0.

Để ý rằng (n+1)2 = n2 + 2n + 1, ta có

*H(n) = H(n-1) + a + 2(n-1) + 1*

*= H(n-1) + a + n – 1 + n – 1 + 1*

*= H(n-1) + a + n + n – 1.*

*Vậy ta dùng công thức tính H(n) theo H(n-1) như sau:*

*H(n) = H(n-1) + a + n + n – 1*

## Problem 28. Number spiral diagonals

Starting with the number 1 and moving to the right in a clockwise direction a 5 by 5 spiral is formed as follows:

**21** 22 23 24 **25**  
20  **7**  8  **9** 10  
19  6  **1**  2 11  
18  **5**  4  **3** 12  
**17** 16 15 14 **13**

It can be verified that the sum of the numbers on the diagonals is 101.

What is the sum of the numbers on the diagonals in a 1001 by 1001 spiral formed in the same way?

**Bài 28. Các đường chéo của dãy số xoắn ốc**

Xuất phát từ số 1 vả di chuyển vòng về bên phải theo chiều kim đồng hồ (ND. như khi ta quấn cuộn len) để ghi các số tự nhiên liên tiếp vào bảng số 5×5 ta thu được:

**21** 22 23 24 **25**  
20  **7**  8  **9** 10  
19  6  **1**  2 11  
18  **5**  4  **3** 12  
**17** 16 15 14 **13**

Có thể kiểm tra để thấy rằng tổng các số trên hai đường chéo là 101.

Tổng này sẽ là bao nhiêu nếu ghi số theo cách trên vào bảng 1001×1001?

Thuật toán

Ta tạm gọi bảng số n×n, n lẻ là *bảng bậc n*. Gọi S(n) là tổng các số trên hai đường chéo của bảng bậc n. Ta có

S(1) = 1;

S(3) = S(1) + tổng các số tại 4 góc = 1 + 3 + 5 + 7 + 9 = 15.

Tổng quát, ta có: S(n) = S(n-2) + tổng các số tại 4 góc, n > 1.

Rất dễ xác định các số tại 4 góc của bảng bậc lẻ n.

* Góc trên phải: n2.

Từ góc này chuyển qua các góc khác theo chiều ngược kim đồng hồ ta được:

* Góc trên trái: n2 - (n-1)
* Góc dưới trái: n2 = n2 - 2(n-1)
* Góc dưới phải: n2 = n2 - 3(n-1)

Tổng các số trên 4 góc sẽ là: 4n2 - 6(n-1).

Ta có:

* S(1) = 1;
* S(n) = S(n-2) + 4n2 - 6(n-1), n > 1.

Bình luận

Thay cho công thức truy hồi, bạn có thể xây dựng công thức tính trực tiếp S(n) như sau:

S(n) = S(n-2) + 4n2 - 6(n-1) = [S(n-4) + 4(n-2)2 – 6(n-3)] + 4n2 – 6(n-1) =

= S(n-4) + 4[(n-2)2 + n2]– 6[(n-1) + (n-3)].

Tổng quát, ta có thể vận dụng qui nạp toán học để chứng minh:

S(n) = 1 + 4(A(n)-1) - 6B(n-1), n lẻ

trong đó A(n) là tổng bình phương của các số lẻ: A(n) = 12 + 32 + … + n2.

B(n-1) là tổng các số chẵn: B(n-1) = 2 + 4 + … + (n-1).

Ta tiếp tục vận dụng hai công thức sau để tìm ra công thức tính S(n).

Tổng n số tự nhiên đầu tiên: D(n) = 1 + … + n:

Tổng bình phương của n số tự nhiên đầu tiên: E(n) = 12 + … + n2:

Đặt n = 2k+1, k = 0, 1,… ta thu được

A(2k+1) = 12 + 32 + … + (2k+1)2

Kí hiệu C(2k) là tổng bình phương của các số chẵn, ta có:

C(2k) = 22 + 42 +..+ (2k)2 = 4(1+22+…+k2) = 4E(k) =

Ta lại có A(2k+1) + C(2k) = E(2k+1). Từ đây suy ra

A(2k+1) = E(2k+1) - C(2k)

= E(n) – C(2k)

= =

=

B(2k) = 2 + 4 + … + 2k = 2(1+2+…+k) = k(k+1).

Vậy,

S(2k+1) = 4(A(2k+1)-1) – 5B(2k)

Tiếp tục thu gọn công thức trên ta nhận được

Đây là công thức tính trực tiếp tổng các số trên đường chéo của bảng bậc 2k+1. Thay k = 500 vào công thức trên bạn thu được đáp số.

Theo công thức này bạn chỉ phải làm khoảng 5 phép nhân số học.

DevCPP

int E28B(int k) {

int kk = k\*k;

return ((16\*kk\*k + 26\*k)/3 + 10\*kk + 1);

}

Free Pascal

type int = longint;

function E28B(k: int): int;

var kk: int;

begin

kk := k\*k;

exit((16\*kk\*k + 26\*k)div 3 + 10\*kk + 1);

end;

## Problem 29. Distinct powers

Consider all integer combinations of *ab* for 2 ≤ *a* ≤ 5 and 2 ≤ *b* ≤ 5:

22 = 4, 23 = 8, 24 = 16, 25 = 32,

32 = 9, 33 =27, 34 = 81, 35 = 243,

42 = 16, 43 = 64, 44 = 256, 45 = 1024,

52 = 25, 53 = 125, 54 = 625, 55 = 3125.

If they are then placed in numerical order, with any repeats removed, we get the following sequence of 15 distinct terms:

4, 8, 9, 16, 25, 27, 32, 64, 81, 125, 243, 256, 625, 1024, 3125

How many distinct terms are in the sequence generated by *ab* for 2 ≤ *a* ≤ 100 and 2 ≤ *b* ≤ 100?

**Bài 29. Các lũy thừa khác nhau**

Xét toàn bộ các tổ hợp của *ab* với 2 ≤ *a, b* ≤ 5:

22 = 4, 23 = 8, 24 = 16, 25 = 32,

32 = 9, 33 =27, 34 = 81, 35 = 243,

42 = 16, 43 = 64, 44 = 256, 45 =1024,

52 = 25, 53 = 125, 54 = 625, 55 = 3125.

Nếu viết các kết quả trên thành dãy sắp tăng và loại bớt các giá trị trùng lặp ta thu được 15 số khác nhau như sau:

4, 8, 9, 16, 25, 27, 32, 64, 81, 125, 243, 256, 625, 1024, 3125

Có bao nhiêu số trong dãy được xây dựng theo cách trên với 2 ≤ *a, b* ≤ 100?

Thuật toán

Giả sử ta thực hiện một cách ngây thơ và tuần tự như mô tả trong đề bài, nghĩa là:

+ Với a = 2 thực hiện ab với b = 2..100

Với a = 3 thực hiện ab với b = 2..100

…

\* Với a = 4 = 22 thực hiện ab với b = 2..100

…

\* Với a = 8 = 23 thực hiện ab với b = 2..100

…

\* Với a = 2k thực hiện ab với b = 2..100

…

Với a = 3 thực hiện ab với b = 2..100

…

Với a = 100 thực hiện ab với b = 2..100

Ta thấy ngay là các dòng có dâu \* chính là nơi có thể sinh ra các giá trị lặp (là những giá trị đã xuất hiện tại các dòng trước). Chẳng hạn, khi a = 8 = 23 thì với b = 2 ta có a2 = (23)2 = 26 là giá trị đã xuất hiện ở dòng +. Việc cần làm là tổ chức duyệt để phát hiện các giá trị trùng lặp mà không cần thực hiện các phép tính lũy thừa.

Với mỗi a trong khoảng 2..100, và với mỗi b = 2..100, thay vì tính ab ta chỉ đánh dấu giá trị b vào một mảng bít deg[b] = 1. Giả sử với a = 2 và ta đã đánh dấu deg[b] = 1 với b = 2..100. Khi xét đến a = 8 = 23 và b = 2, ta sẽ gặp ab = a2 = (23)2 = 26 và do đó deg[6] = 1 tức là ta gặp giá trị lặp.

Điều thú vị là mảng deg được dùng cho mọi giá trị a = 2..100. Ta có thuật toán sau:

T1. Khởi trị total = 99\*99 là chiều dài tối đa của dãy ab, 2 ≤ a, b ≤ 100.

T2. Với mỗi số a = 2.. 10 = chưa xử lí ta duyệt

T2.1 Đánh dấu deg[b] = 1, b = 2..100 với ý nghĩa là đã tính ab, b = 2..100.

T2.2 Với mỗi x = ak, 2 ≤ k ≤ 100, 2 ≤ x ≤ 100 ta duyệt b = 2..100

* + Nếu deg[kb] = 1 tức là xb = akb đã được tính tại bước trước thì loại giá trị trùng lặp bằng cách giảm total 1 đơn vị.
  + Ngược lại, nếu deg[kb] = 0 tức là xb = akb được xuất hiện lần đầu thì đánh dấu deg[b] = 1.

Vì deg được dùng cho mọi a nên ta tổ chức như một mảng bit. Ngoài ra trong quá trình duyệt các a, ta có thể đã duyệt cả các x = ak, nên ta cần thêm một mảng bit taked để đánh dấu các giá trị a đã duyệt.

Lưu ý

Tại bước T2 ta chỉ cần duyệt các a từ 2 đến 10 = .

## Problem 30. Digit fifth powers

Surprisingly there are only three numbers that can be written as the sum of fourth powers of their digits:

1634 = 14 + 64 + 34 + 44

8208 = 84 + 24 + 04 + 84

9474 = 94 + 44 + 74 + 44

As 1 = 14 is not a sum it is not included.

The sum of these numbers is 1634 + 8208 + 9474 = 19316.

Find the sum of all the numbers that can be written as the sum of fifth powers of their digits.

**Bài 30. Tổng các chữ số của lũy thừa bậc 5**

Điều ngạc nhiên là chỉ có ba số có thể được viết dưới dạng tổng các lũy thừa bậc bốn của các chữ số của số đó:

1634 = 14 + 64 + 34 + 44

8208 = 84 + 24 + 04 + 84

9474 = 94 + 44 + 74 + 44

Tổng các số này là: 1634 + 8208 + 9474 = 19316.

Ta không tính số 1, mặc dù 1 = 14, vì đó không phải là tổng (ND. của hai số hạng trở lên).

Hãy tính tổng tất cả các số có thể được viết dưới dạng tổng các lũy thừa bậc năm của các chữ số của số đó?

Thuật toán

Trước hết ta tạm gọi các số có tổng lũy thừa của các chữ số bằng chính số đó là các *số bậc 5*. Ta sử dụng các kí hiệu sau đây:

Len(x) = số chữ số của số x, thí dụ, len(2034) = 4, len(1000000) = 7.

Sum(x) = tổng các lũy thừa bậc 5 cuả các chữ số của x, thí dụ

Sum(12) = 15 + 25 = 1 + 32 = 33

Ta thử tính lũy thừa bậc 5 của các chữ số 0..9 và xác định số chữ số của chúng. Ta có

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| cn | 0 | 1 | 32 | 243 | 1024 | 3125 | 7776 | 16807 | 32768 | 59049 |
| len(cn) | 1 | 1 | 2 | 3 | 4 | 4 | 4 | 5 | 5 | 5 |

Bảng này cho ta biết điều gì?

1. Nếu a là số có 1 chữ số (là a) thì a không phải là số bậc 5. Thật vậy, vì a5 ≠ a, trừ trường hợp a = 1 như đầu bài đã qui định.

2. Nếu a = 10x + y là số có 2 chữ số (là x và y) thì a không phải là số bậc 5. Thật vậy, vì a chỉ có 2 chữ số mà c5 với c = 3..9 thì c5 có trên 2 chữ số nên x, y < 3, cụ thể là x, y ∈ {0, 1, 2}. Khi đó Ta chỉ có 6 số 2 chữ số sau đây: 10, 20, 11, 12, 21, 22. Ta có

Sum(10) = 1 ≠ 10

Sum(20) = 32 ≠ 20

Sum(11) = 2 ≠ 11

Sum(12) = Sum(21) = 33 ≠ 12 và 21

Sum(22) = 64 ≠ 22.

Vậy các số từ 1..99 đều không phải là các số bậc 5.

3. Bằng lập luận tương tự bạn phát hiện ra rằng các số 3 chữ số cũng không phải phải là các số bậc 5.

Tổng quát, ta muốn chứng minh mệnh đề sau:

Mệnh đề

Mọi số tự nhiên n có ít hơn 4 chữ số hoặc nhiều hơn 6 chữ số không phải là số bậc 5.

Trước hết ta cần một bổ đề.

Bổ đề

Với mọi số tự nhiên k = 6+d, d = 1, 2, … ta có k < 10d.

*Chứng minh*

Ta chứng minh bổ đề bằng qui nạp toán học theo d.

Cơ sở: d = 1. Ta có k = 6+1 = 7 < 10 = 101. Bổ đề đúng với d = 1.

Giả thiết qui nạp: Giả sử với d ≥ 1 ta có 6+d < 10d.

Kết luận qui nạp: Ta cần chứng minh 6+(d+1) < 10d+1.

Vì Len(10d+1) = d+1, Len(10d+1) = d+2 nên 10d+1 < 10d+1. Mặt khác, theo giả thiết qui nạp, ta có, 6+d < 10d. Từ đây suy ra 6+d+1 < 10d+1 < 10d+1. Bổ đề được chứng minh.

Tiếp theo, ta chứng minh rằng nếu số tự nhiên x có k ≥ 7 chữ số thì Sum(x) < x. Thật vậy, vì k ≥ 7 nên ta viết k dưới dạng k = 6+d, d = 1, 2,… Vì x có k chữ số nên

10k-1 ≤ x ≤ 9…9 = 10k-1

trong đó c…c là số có k chữ số c.

Khi đó Sum(x) ≤ Sum(9…9) = k⋅95.

Theo bổ đề trên, k = 6+d < 10d. Mặt khác, k-1 > 5+d. Từ đây suy ra

Sum(x) ≤ k⋅95 < k⋅105 < 105 ⋅10d = 105+d < 10k-1 ≤ x. Như vậy, Sum(x) < x do đó x không thể là số bậc 5, điều phải chứng minh.

Tóm lại, ta chỉ cần tìm các số bậc 5 trong các số có 4, 5 và 6 chữ số. Ta dùng phương pháp chia để trị như sau:

Ta tạm kí hiệu mink và maxk là giới hạn nhỏ nhất và lớn nhất khi cần khảo sát các số có k chữ số để tìm các số bậc 5, tức là tìm x trong khoảng mink..maxk thỏa Sum(x) = x.

Với các số có 4 chữ số, lúc đầu ta có min4 = 1000, max4 = 9999. Ta sẽ gắng thu hẹp giới hạn này để giảm độ phức tạp của thuật toán.

Vì 75, 85 và 95 đều có 5 chữ số nên các số x có 4 chữ số cần tìm không thể chứa chữ số c > 6, nghĩa là x chỉ có thể chứa các chữ số 0…6, max4 = 6666. Vì 65+55 = 7776+3125 = 10901 có 5 chữ số nên max4 = 6500. Vậy với lớp 4 chữ số ta chỉ cần duyệt trong khoảng 1000 đến 6500. Thủ tục Scan(min4, max4) khi đó sẽ tìm tổng các số x có 4 chữ số trong khoảng min4..max4 và thỏa Sum(x) = x.

5. Tương tự, để duyệt các số 5 chữ số ta có thể ước lượng được min5 = 10000 và max5 = 98700.

6. Để duyệt các số 6 chữ số ta có thể ước lượng được min6 = 100000 và max6 = 999999.

Trong thủ tục Scan ta cũng có thể cải tiến chút it. Cụ thể là với mỗi số x trong khoảng đang xét, ta tổ chức tính tổng lũy thừa bậc 5 của các chữ số lớn trước. Nếu tổng này > xta thoát ngay.

Thursday, May 12, 2016