ỨNG DỤNG THUẬT TOÁN EUCLIDE

1. Modulo – Đồng dư thức

Phép đồng dư thức cho bạn số dư của phép chia số này cho số khác. Dấu của phép đồng dư là %.

Ví dụ: Ta có hai số 5 và 2, khi đó 5%2 bằng 1 do khi chia 5 cho 2, ta được số dư là 1.

Tính chất: Đồng dư thức có một số tính chất sau:

(a+b)%c=a%c+b%c

(a.b)%c=((a%c).(b%c))%c

2. GCD – Ước chung lớn nhất

Ước chung lớn nhất (GCD, viết tắt của từ Greatest Common Divisor) của hai hay nhiều số là số nguyên dương lớn nhất mà là **ước chung (common divisor)** của tất cả các số đó.

Ví dụ: GCD của 6 và 10 là 2 vì 2 là số nguyên dương lớn nhất mà là ước chung của 6 và 10.

Để lập trình tìm ƯCLN ta có thể lập trình như sau:

```
int gcd(int A, int B)
{
   for (int i = min(A, B); i > 0; --i)
      if (A % i == 0 && B % i == 0) {
         return i;
      }
}
```

Đây là cách đơn giản Ta duyệt tất cả các số từ min(A,B) đến 1 và kiểm tra xem số đang xét có phải là ước của của A và B hay không. Nếu đúng như vậy thì số đang xét sẽ là GCD của A và B.

Độ phức tạp của thuật toán: O(min(A,B)).

Để cải tiến, ta sử dụng thuật toán Euclid

Thuật toán Euclid dựa trên tính chất sau của ước chung lớn nhất GCD(A,B)=GCD(B,A%B). Thuật toán sẽ quy nạp cho đến khi A%B=0.

```
int gcd(int A, int B)
{
   if (B == 0) return A;
   else return gcd(B, A % B);
}
```

Hoặc khử đệ quy

```
int GCD(int a, int b)
{
   while (b != 0)
   {
```

```
    int temp = a % b;
    a = b; b = temp;
  }
  return a;
}
```

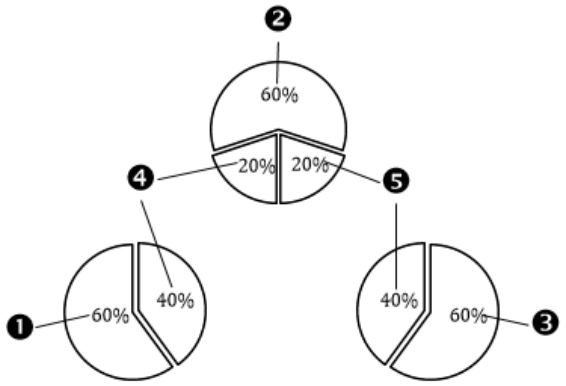
Ta cũng có thể sử dụng hàm $\_GCD()$ được viết sẵn trong thư viện của C++

I. Bài tập vận dụng

Bài 1: Chia bánh – share.cpp

Đề bài:

Hôm nay là sinh nhật của Bờm. Biết Bờm rất thích bánh gato nên mọi người mang tới rất nhiều bánh gato. Bờm muốn chia n cái bánh cho m người bạn của Bờm, ban đầu mỗi cái bánh là một phần. Bờm có cách duy nhất là dùng một con dao cắt bánh, ở mỗi thao tác cắt, Bờm được chia một phần bánh thành 2 phần với tỉ lệ tùy ý. Hãy tìm cách giúp Bờm dùng ít thao tác cắt nhất để chia bánh thành các phần cho m người bạn, mỗi phần thuộc về đúng một người và lượng bánh mỗi người nhận được là bằng nhau.



Dữ liệu vào: Một dòng chứa hai số nguyên dương n, m ($n \leq 10^{18}$).

Dữ liệu ra: Gồm một số nguyên duy nhất là số thao tác cắt phải sử dụng.

Ví dụ:

SHARE.INP	SHARE.OUT
3 5	4

Hướng dẫn:

Gần tương tự như bài 1, bài này đáp án cần tìm sẽ là hiệu của số lượng người và ước chung lớn nhất của số người và số bánh.

Bài 2: Văn nghệ - vn.cpp

Đề bài:

Đội văn nghệ xung kích của trường đại học X được cử đi diễn giao lưu ở các huyện trong tỉnh Y. Khi đi đoàn có n bạn nam và m bạn nữ. Mỗi tổ sẽ được giao nhiệm vụ biểu diễn tại một địa điểm phục vụ các em nhỏ. Biết rằng số lượng nam và nữ phải chia đều giữa các tổ. Hỏi người quản lý có thể chia số sinh viên ra tối đa được bao nhiêu tổ? Mỗi tổ có bao nhiêu nam và bao nhiêu nữ.

Dữ liệu vào:

- Hai số nguyên n, m cách nhau một khoảng trắng($1 < n, m < 10^9$)

Dữ liệu ra:

- Dòng 1 ghi số lượng tổ tối đa có thể chia
- Dòng 2 ghi 2 số a, b tương ứng là số nam và số nữ của mỗi tổ

Ví dụ:

Vn.inp	Vn.out
48 72	24 2 3

Hướng dẫn:

Bài toán này ta thấy vì số lượng giữa nam và nữ của các tổ là như nhau, do đó ta tìm ước chung lớn nhất giữa số nam và số nữ tương ứng đó chính là số tổ nhiều nhất có thể chia được. Khi có số tổ ta chỉ việc chia số lượng nam và số lượng nữ cho số lượng tổ để tìm được số lượng nam và số lượng nữ trong từng tổ.

Bài 3: Trực nhật – tn.cpp

Đề bài:

Ở một lớp học có n học sinh. Mỗi bạn đều phải trực nhật và cứ sau một số y ngày nhất định bạn đó mới phải trực nhật lại. Biết rằng xuất phát điểm ban đầu tất cả sẽ đều trực nhật vào ngày đầu tiên. Bạn hãy giúp lớp trưởng tính xem sau bao nhiêu ngày thì tất cả các bạn mới lại cùng nhau trực nhật và khi đó mỗi bạn đã trực nhật bao nhiêu lần.

Dữ liệu vào:

- Dòng đầu chứa số nguyên n ($2 \leq n < 100$)
- Dòng thứ hai chứa n số nguyên y. ($1 \leq y < 100$)

Dữ liệu ra:

- Dòng đâu tiên ghi ra số ngày mà tất cả cùng nhau trực nhật lại.
- Dòng thứ hai chứa n số là số lần một bạn đã trực nhật cho tới lúc tất cả cùng trực nhật

Ví dụ:

tn.inp	tn.out
3 2 3 4	12 6 4 3

Hướng dẫn:

Để tìm được sau bao nhiêu ngày thì tất cả học sinh mới cùng nhau trực nhật, ta lần lượt tìm bội chung nhỏ nhất của các cặp học sinh trong n học sinh. Khi đó kết quả cuối cùng chính là số ngày cần tìm. Với từng học sinh, ta chia số ngày cho số ngày phải trực của từng bạn để tìm ra số lần phải trực nhật.

Để tìm bội chung nhỏ nhất ta có thể áp dụng công thức

BCNN(a,b) = a*b / ƯCLN(a,b)

Bài 4: Dãy ước số chung lớn nhất – dayucln.cpp

Đề bài:

An mới bắt đầu học toán và cậu ta vẫn chưa biết gì về ước số chung lớn nhất. Cho nên cậu ta cần sự giúp đỡ của bạn để làm một vài phép toán. An có một dãy A gồm N số

nguyên, được đánh số từ 1 đến N và An muốn tạo một mảng B gồm N+1 phần tử được đánh số từ 1 đến N+1 và các phần tử của mảng B có tính chất sau: UCLN(B[i], B[i + 1]) = A[i], ∀ $1 \le i \le N$.

Nếu như có nhiều kết quả thì bạn chỉ cần cho An biết mảng B có tổng các phần tử là nhỏ nhất (vì cậu ta chỉ mới bắt đầu học toán nên rất ngại các con số lớn).

Dữ liệu vào:

- Dòng đầu tiên chứa một số nguyên **T** - số lượng test case ($1 \le \mathbf{T} \le 10$). Tiếp theo là các test case:
- Dòng đầu tiên của mỗi test case chứa một số nguyên dương **N** - số lượng phần tử của mảng A ($2 \le \mathbf{N} \le 10^5$).
- Dòng thứ hai chứa N số nguyên $A_1\ A_2\ ...\ A_N$ ($1 \le A_i \le 10^9$)

Dữ liệu ra:

- Với mỗi test case xuất trên một dòng chứa N+1 phần tử của mảng B ($0 < B_i$)

Ví dụ:

dayucnl.inp	dayucnl.out
2 3 1 2 4 3 6 10 2	1 2 4 4 6 30 10 2

Hướng dẫn:

Để tạo ra dãy B với các phần tử có điều kiện như đầu bài, đầu tiên ta in ra phần tử đầu dãy A[1], với mỗi phần tử A[i] tiếp theo, ta tìm bội chung nhỏ nhất của cặp phần tử A[i] và A[i-1]. Cuối cùng ta in ra phần A[n] vào cuối mảng B.

Bài 5: Tổng các ước chung lớn nhất – sumgcd.cpp

Đề bài:

Gọi gcd(x, y) là ước chung lớn nhất của 2 số x và y. Ta có F[n] là tổng gcd(i, j) với 1 <= i, j <= n và i < j. Yêu cầu hãy tính F[n].

Dữ liệu vào:

- Dòng đầu chứa số T (T <= 100000) là số test;
- Dòng thứ 2 đến T + 1, mỗi dòng chứa số N (1 <= N <= 1000000);

Dữ liệu ra:

- Gồm T dòng là kết quả tương ứng.

Nguồn: ACM World Final Warm up 1 - 2008

Ví dụ:

sumgcd.inp	sumgcd.out
5 1 2	0 1 3

3 4 5	7 11

Giải thích: F[4] = gcd(1, 2) + gcd(1, 3) + gcd(1, 4) + gcd(2, 3) + gcd(2, 4) + gcd(3, 4) = 7;

Hướng dẫn:

Nếu vẫn sử dụng phương pháp cũ là dùng hàm GCD(a,b)

```
for(int i=1;i< N;++i)
  for(int j=i+1;j<=N;++j)
    sum += gcd(i,j);
```

Ta cho chạy 2 vòng for để duyệt và cộng vào tính tổng thì sẽ bị quá thời gian, không giải quyết được các test lớn. Ta có thể dùng cách dưới đây để giải quyết bài toán.

Bài 6: Tổng các số - sum.cpp

Đề bài:

Cho hai số tự nhiên N1 và N2. Hãy tính tổng tất cả các chữ số của tất cả các số từ N1 đến N2 ($N1 \le N2 \le 2.10^9$).

Dữ liệu vào:

- 2 chữ số nguyên N1, N2 cách nhau 1 khoảng trắng

Dữ liệu ra:

- Tổng tất cả các chữ số của tất cả các số từ N1 đến N2

Ví dụ.

sum.inp	sum.out
25 49	205

Hướng dẫn:

tổng các chữ số từ N1 đến N2 bằng tổng các chữ số từ 1 đến N2 trừ đi tổng các chữ số từ 1 đến N1-1.

Đặt C=X div 10^5. Chúng ta phân chia các số từ 1 đến X thành C+1 nhóm: A_0, A_1, …, A_C. Nhóm A_0 chứa các số từ 1 đến 99999, nhóm A_1 chứa các số từ 10^5 đến 199999. Chính xác hơn, nhóm A_i ($0 \le i < C$) chứa các số $10^5.i$, $10^5.i + 1$, …, $10^5.i + 99999$. Cuối cùng nhóm A_C chứa tất cả các số còn lại. Nhận thấy rằng, mọi nhóm đều thỏa mãn 2 tính chất sau:

Nếu mỗi số trong A_i bỏ đi 5 chữ số cuối cùng thì còn lại 10^5 số bằng i.

Nếu mỗi số trong A_i chỉ gữ lại 5 chữ số cuối cùng thì nhận được các số như trong A_0 không phụ thuộc vào i.

Do đó, nếu ký hiệu tổng các chữ số của số i là S(i), ký hiệu tổng các chữ số của nhóm A_i là $S(A_i)$ thì dễ thấy:

Với mọi i, $0 \le i < C$, $S(A_i) = 10^5.S(i)+S(A_0)$

Tổng các chữ số của các số từ 1 đến X bằng: $S(A_0)+S(A_1)+\ldots+S(A_C)$

$=10^5.[S(0)+S(1)+…S(i)+…+S(C-1)] + C.S(A_0) + S(A_C)$ (1).

Bài 7: Bậc của số

Một số A gọi là có bậc K đối với cơ số B nếu như:

$A = B^{x1} + B^{x2} + … + B^{xk}$

(trong đó $x1 \neq x2 \neq x3 … \neq xk$)

Ví dụ:

- 17 có bậc 2 đối với cơ số 2 vì $17 = 2^4 + 2^0$.
- 151 có bậc 3 đối với cơ số 5 vì $151 = 5^3 + 5^2 + 5^0$.

Yêu cầu: Cho trước 1 đoạn [X, Y]. Hãy xác định xem trong đoạn này có bao nhiêu số có bậc K đối với cơ số B.

Giới hạn:

- $1 \leq X \leq Y \leq 10^9$.
- $1 \leq K \leq 20$, $2 \leq B \leq 9$.
- Bộ nhớ: 200KB.
- Time limit: 1 s.

Input	Output
X Y K B	Gồm 1 số duy nhất là kết quả tìm được.

Ví dụ

Input	Output
15 20 2 2	3

Giải thích: Đó là các số $17 = 2^4 + 2^0$; $18 = 2^4 + 2^1$; $20 = 2^4 + 2^2$.

Hướng dẫn:

Ta sẽ đưa bài toán này về một bài toán nhỏ hơn đó là cho biết trong đoạn [1, A] có bao nhiêu số có bậc K đối với cơ số B.

Với mỗi giá trị A ta đều xác định được B^n sao cho $B^{n+1} > A$. Áp dụng tính chất: $B^n > B^{n-1} + B^{n-2} + … B^0 \rightarrow A > B^{n-1} + B^{n-2} + … B^0$.

➔ Tổng bất kỳ của một tổ hợp K phần tử của (n-1) phần tử này cũng đều < A (tức là: $A > B^{x1} + B^{x2} + …+ B^{xi} +… + B^{xk}$ (với mọi $xi \leq n–1$)).

Như vậy ta đã giải quyết xong vấn đề với các số mà không chứa B^n, nếu như không chứa B^n thì số lượng số thoả mãn sẽ = Tổ hợp chập K của (n-1). Ta sẽ tiếp tục giải quyết vấn đề với các số mà có dạng = $B^n + B^{x2}+…+B^{xk}$.Dễ thấy vấn đề ở đây lại quay về là tìm số lượng các số có bậc (K-1) đối với cơ số B trong đoạn [1, A-B^n] và không chứa B^n, vậy thì ở đây ta chỉ cần xác định lại n mà thôi, n(mới) ≤ n (cũ) – 1.

Như vậy bài toán đã được giải quyết.

Nói tóm lại đây là một bài khá đặc trưng cho QHĐ = Đệ quy với công thức truy hồi đơn giản:

Cal(A, K, B) = Tổ hợp chập K của (N-1) + Cal(A-B^n, K-1, B).

Bài 8: Pinocchio

Cha của Pinocchio muốn làm lại cho Pinocchio một cái mũi mới. Ông có N thanh gỗ, mỗi thanh gỗ có độ dài A_i. Là người yêu thích toán học ông ta đưa ra một giải thuật sau để lấy ra thanh gỗ có độ dài cần thiết:

- Nếu còn lại 1 thanh gỗ thì ông ta sẽ lấy thanh gỗ này làm mũi cho Pinocchio.
- Nếu còn >= 2 thanh gỗ thì ông ta sẽ làm như sau: Chọn ra 2 thanh gỗ i, j sao cho A_i và A_j có độ dài nhỏ nhất trong số N thanh.
    - Nếu $A_i = A_j$ thì vứt bỏ một thanh đi, lại quay về bước 1
    - Nếu $Ai < A_j$ thì ta sẽ cắt thanh A_j đi một đoạn = Ai (tức là $A_j = A_j – Ai$).
- Quay lại bước 1.

Yêu cầu: Hãy tính xem độ dài mà thanh gỗ ông ta nhận được sẽ là bao nhiêu.

Giới hạn:

- $1 \le N \le 100000$.
- $1 \le A_i \le 10^9$.
- Time limit 1s, bộ nhớ 200 KB.

Input	Output
N A_1 A_2 … A_n	Gồm 1 số duy nhất X là độ dài thanh gỗ đạt được.

Ví dụ:

Input	Output
3 2 3 4	1

Hướng dẫn: Dễ dàng chứng minh được đó chính là ước chung lớn nhất của N số. Ta có thể dùng thuật toán Euclid. Cấp độ tính toán của bài này là O(n).

Bài 9: Nikifor

Nikifor có một số tự nhiên N. Anh ta là một người rất yêu thích con số 7. Bởi vậy con số của anh ta phải là một con số chia hết cho 7. Nhưng rất tiếc con số N của anh ta lại chưa phải là bội của 7. Bây giờ anh ta muốn đổi vị trí của các chữ số của số N sao cho tạo được một số mới mà số này chia hết cho 7. Vì Nikifor là một người rất đặc biệt nên con số N của anh ta cũng rất đặc biệt. Số N này luôn có đủ 4 chữ số 1, 2, 3 4 trong biểu diễn thập phân của mình! Hơn nữa nếu số N này đã là bội của 7 rồi thì anh ta vẫn muốn biến đổi để ra được 1 số mới khác cũng chia hết cho 7.

Yêu cầu: Hãy giúp Nikifor đổi vị trí của các chữ số trong số N sao cho được số mới chia hết cho 7.

Giới hạn:

- số N này có không quá 1000 chữ số và có ít nhất 4 chữ số.
- Time limit: 1s, bộ nhớ 200KB.

Input	Output
Dòng 1: 1 số X là số bộ test. ($1 \leq X \leq 10000$). X dòng tiếp theo mỗi dòng ghi 1 số N.	N dòng ghi ra kết quả của từng test, nếu test nào vô nghiệm thì ghi ra "No solution" ngược lại ghi ra số N sau khi biến đổi.

Ví dụ:

Input	Output
2 10234 531234	32410 353241

Hướng dẫn:

Nhận thấy số N luôn có đủ 4 chữ số 1, 2, 3, 4. Với 4 số này ta có thể tạo thành bất kỳ số dư nào trong các số dư từ 0->6:

Ví dụ: 1234 mod 7 = 2, 3241 mod 7 = 0 …

Như vậy ta có một cách làm rất hay cho bài này đó là: đặt tuỳ ý các chữ số khác 0 vào đầu sao cho còn dư lại 4 chữ số 1, 2, 3, 4. Với 4 chữ số này ta sẽ đặt vào tiếp theo, sau cùng mới đặt các chữ số 0. 4 chữ số 1, 2, 3, 4 cuối ta sẽ duyệt hoán vị của nó xem hoán vị nào của nó thì số N tạo được sẽ chia hết cho 7. Vì với mỗi số dư R đều có ít nhất 2 hoán vị nên dù N cũng có dạng như trên thì cũng có ít nhất một số khác cũng cùng dạng này nữa ➔ Bài toán không bao giờ vô nghiệm. Vì 4! = 24 -> Chạy rất nhanh.