CHAP 1
MATHEMATIC

HỆ CƠ SỐ :
Cơ số thông dụng : thường là cơ số 2, 8 và 16.
Ví dụ : cơ số 2 dc gồm 2 chữ số là 0 và 1.
Note :
Cơ số div 2 ➔ chuyển về hệ nhị phân rồi đổi qua cơ số div 2 ấy
Chú ý : những bài toán mang tính cơ số mũ - X^k .

LÝ THUYẾT TẬP HỢP :
Các định nghĩa và kí pháp về phần bù , hợp, giao , hiệu của tập hợp
Tích decattes của các tập hợp bao gồm nguyên lí cộng, nguyên lí bù trừ, nguyên lí nhân.

LÝ THUYẾT ĐẾM :
Các bài toán về chỉnh hợp lặp, không lặp, tổ hợp, hoán vị.
Note : một số lưu ý cần thiết :

$C(0,n) = C(n,n) = 1$
$C(k,n) = C(k-1,n-1) + C(k,n-1)$
$C(k,n) = C(n-k,n)$

$C(1,n) + C(2,n) + \ldots + C(n,n) = 2^{n-1}$
$C(k,n) = C(k-1,n-1) + C(k-1,n-2) + \ldots + C(k-1,k-1)$

$P_n = 1 + P_1 + 2P_2 + 3P_3 + \ldots + (n-1)P_{n-1}$
$A(k,n) = A(k,n-1) + k*A(k-1,n-1)$
n^k - (n-1)^k = C[k][1]*n^(k-1) - C[k][2]*n^(k-2)+……

note :
số stirling loại 1 : cho n phần tử, k tập , xếp n phần tử vào k tập sao cho trong 1 tập, các phần tử này phải có chu trình khác nhau (1 > 2 > 3 >1 và 2 >3 >1 >2 tính là 2 cách khác nhau .
CT tính :
f[n,k] := f[n-1,k-1] + n*f[n-1,k] với f[0,0] = 1,f[n,0] = 0
số stirling loại 2 : cách xếp n phần tử vào k tập :
f[n,k] := f[n-1,k-1] + k*f[n-1,k] với f[0,0] = 1,f[n,0] = 0

1 pp đếm cân bằng :
Xét bài toán : cho 1 dãy nhị phân. Hãy đếm xem có bao nhiêu cách chọn trong đoạn (i..j) mà sao cho số chữ số 1 = số chữ số 0.
Hint : ta sẽ dùng 1 mảng cân bằng để đếm.
Code :

```
c[0]:=1;sum:=0;count:=0;
for i:=1 to length(s) do
begin
    if s[i]='0' then sum:=sum - 1
      else sum:=sum + 1;
    count:=count + c[sum];
    inc(c[sum]);
  end;
```

GCD :

Ước số : ước số của 1 số có những lưu ý sau :

Số lượng ước số :

Trong phép phân tích số n = a^i x b^j thì tổng các ước số sẽ là (I + 1) x (j + 1) …

Cm : dựa theo quy tắc đếm :D

Tổng ước số của 1 số : ta có công thức dc chứng minh theo quy nạp.

$$S = \frac{(a^{i+1}-1)}{a-1} x \frac{(b^{j+1}-1)}{b-1} x ... x \frac{(c^{k+1}-1)}{c-1}$$

Tìm ước số chung lớn nhất của a và b

Với a,b bất kì → `a*u + b*v = GCD(a,b)` `// u,v có thể âm`

Để giải quyết vấn đề này, ta dùng Euclide agorigthm → tìm dc GCD, u, v

u,v tìm bằng công thức :

```
xb := xt - q*xb
yb := yt - q*yb
```

với khởi trị là `(xa,ya) = (1,0), (xb,yb) = (0,1).`

USCLN: $(a, b) = p_1^{min(a1, b1)} p_2^{min(a2, b2)} ... p_k^{min(ak, bk)}$

BSCNN: $[a, b] = p_1^{max(a1, b1)} p_2^{max(a2, b2)} ... p_k^{max(ak, bk)}$

Với p_n là số phân tích

Mở rộng :

GCD(a,b,…,c) = GCD(GCD(a,b),…,c)

LCD(a,b) * GCD(a,b) = a*b

Euclide nhị phân :

```
a = b → GCD = a
a > b → a = a - b
a < b → b = b - a
```

code :

```
function gcd(a,b : longint;var x,y : longint) : longint;
var
  x1,y1 : longint;
begin
  if a < b then gcd := gcd(b,a,y,x)
  else
    if b = 0 then
      begin
        gcd := a;
        x := 1;y := 0;
        exit;
      end
    else
      begin
        gcd := gcd(b, a mod b,x1,y1);
        x := y1;
        y := x1 - (a div b) * y1;
      end;
end;
```

một số tính chất khác :

Nếu p là ước nguyên tố bé nhất của n, thì $p^2 \le n$ (n=pq, mà p ≤ q, do đó $p^2 \le pq = n$)

```
BSCLN(a,b) x USCNN(a,b) = a x b.
USCNN(a,b,c) = USCNN(USCNN(a,b),c) = USCNN(a,USCNN(a,b)).
```

Giải phương trình đồng dư tuyến tính

Giải pt ax ≡ b (mod c) (1)

Ta đưa pt (1) về dạng ax + cy = b.

Step 1 : tìm d = GCD(a,c) và u,v

Step 2 : pt có nghiệm khi b mod d = 0

Step 3 : nghiệm của pt là $x = x_o + k*e$

Note : $x_o = u * b / d$, $e = c / d$, k dương

Giải pt Diophantine ax + by = c (1)

Thực chất cách giải pt diophantine giống pt trên, chỉ khác là tìm y thôi :D

Step 1 : tìm nghiệm của pt ax ≡ c (mod b)

Step 2 : y = (c - ax) / b

Step 3 : $(x,y) = (x_o,y_o) + k*(-b_1 ,a_1)$

Note : $a_1 = a / d$, $b_1 = b / d$

Phi hàm euler : pi(n) = số các số có ước chung = 1 trong khoảng 1…n

Ct : với p_i là số nhận dc khi phân tích n ra các thừa số nt

$$\mathbf{p(n) = n*(1 -1/p_1)*(1 - 1/p_2)*...*(1 - 1/p_k)}$$

Số nguyên tố :

Kiểm tra n có là số nguyên tố O(sqrt(n))

Định lí fermat bé : $a^p = a \pmod p$ → xác xuất p là số nguyên tố cao

Miller Rabin algorigthm :

INPUT Số tự nhiên lẻ *n*.

OUTUT NguyenTo: TRUE/FALSE

1. Phân tích $n - 1 = 2^s \cdot m$ trong đó s $\geq$ 1 và m là số tự nhiên lẻ
2. Chọn ngẫu nhiên số tự nhiên a $\in$ {2,...,n-1}.
3. Đặt $b = a^m \pmod n$
4. Nếu $b \equiv 1 \pmod n$ thì trả về TRUE. Kết thúc.
5. Cho k chạy từ 0 đến s-1:
   1. Nếu $b \equiv -1 \pmod n$ thì trả về TRUE. Kết thúc.
   2. Thay b:= $b^2 \pmod n$.
6. Trả lời FALSE. Kết thúc.

Code :

```
readln(n);
pt(n - 1,s,m);
test := random(1000);
for i := 1 to test do
  begin
    a := 2 + random(n - 2);
    b := mu(a,m,n);
    if b mod n = 1 then continue;
    for k := 0 to t do
      begin
        if b mod n = n - 1 then break;
        b := (b * b) mod n;
      end;
```

```
      if b mod n = n - 1 then continue else
        begin
          writeln('NO');exit;
        end;
    end;
  writeln('YES');
```

Sàng số nguyên tố :

Khi cần biết các số nguyên tố đến một phạm vi nào đó, ví dụ từ 2 đến 10^8, sử dụng sàng số nguyên tố Eratosthenes sẽ hiệu qủa hơn về thời gian.

Thủ tục sau tạo sàng số nguyên tố từ 2 đến N:

```
procedure sieve(n);
begin
     fillchar(p, sizeof(p), true);
     for i:=2 to n do
          if (p[i]) then
begin
     j:=i+i; // tạo bước nền
     while (j<=n) do
     begin
          p[j]:=false;
          j:=j+i; // ta loại đi bội của i
     end;
end;
end;
```

Thông tin trả vềtrong mảng p: `p[i] = true` nếu và chỉ nếu i là số nguyên tố.

Bạn có thể ước lượng thời gian chạy của thuật toán sàng Eratosthenes là O(nlogn), ta sẽ đề cập đến một số ước lượng cần thiết ở những phần sau.

Ước lượng số số nguyên tố

Kí hiệu π(n) là số số nguyên tố bé hơn n. Đây là một ước lượng tương đối tốt và ngắn gọn cho π(n):

$$\pi(n) \approx n / \ln(n)$$

Ví dụ : $\pi(10^6) \approx 10^6 / \ln(10^6) \approx 72382$

Con số này sẽ giúp cho việc ước lượng thời gian tính toán cho các bài toán liên quan đến số nguyên tố

BIGNUM :

Bignum 2 loai : dùng ansistring, mảng số nguyên

Loại ansistring : làm như toán bình thường, các phép + ,- ,*, / như tính tay

Code :

```
Type
    Bignum = ansistring;
```

So sánh :

```
function cmd(a,b : bignum) : integer;
begin
  while length(a) > length(b) do b := '0' + b;
  while length(b) > length(a) do a := '0' + a;
  if a > b then exit(1);
  if a < b then exit(-1);
  exit(0);
end;
```

Phép cộng (số lớn vs số lớn) :

```
function cong(a,b : bignum) : bignum;
begin
  while length(a) > length(b) do b := '0' + b;
  while length(b) > length(a) do a := '0' + a;
  nho := 0; c := '';
  for i := length(a) downto 1 do
    begin
      x := ord(a[i]) - 48;
      y := ord(b[i]) - 48;
      sum := x + y + nho;
      nho := sum div 10;
      sum := sum mod 10;
      c := chr(sum + 48) + c;
    end;
  if nho >= 1 then c := '1' + c;
  cong := c;
end;
```

phép trừ (số lớn vs số lớn) :

```
function tru(a,b : bignum) : bignum;
begin
  while length(a) > length(b) do b := '0' + b;
  while length(b) > length(a) do a := '0' + a;
  nho := 0;c := '';
  for i := length(a) downto 1 do
    begin
      x := ord(a[i]) - 48;
      y := ord(b[i]) - 48;
      sum := x - y - nho;
      if  sum < 0 then
        begin
          sum := sum + 10;
          nho := 1;
        end
      else nho := 0;
      c := chr(sum + 48) + c;
    end;
  while (length(c) > 1) and (c[1] = '0') do delete(c,1,1);
  tru := c;
end;
```

phép nhân (số nhỏ vs số lớn)

```
function nhannho(a : bignum;b : integer) : bignum;
begin
  nho := 0;c := '';
  for i := length(a) downto 1 do
    begin
      x := ord(a[i]) - 48;
      sum := x*b + nho;
      nho := sum div 10;
      sum := sum mod 10;
      c := chr(sum + 48) + c;
    end;
  if nho > 0 then str(nho,tmp) else tmp :='';
  nhannho := tmp + c;
end;
```

phép nhân (số lớn vs số lớn)

```
function nhanlon(a,b : bignum) : bignum;
begin
  m := -1;c := '';
  for i := length(a) downto 1 do
    begin
      inc(m);
      tich := nhannho(b,ord(a[i]) - 48);
      for k := 1 to m do tich := tich + '0';
      c := cong(c,tich);
    end;
  nhanlon := c;
end;
```

phép div (số nhỏ vs số lớn)

```
function divnho(a : bignum;b : integer) : bignum;
begin
  du := 0;c := '';
  for i := 1 to length(a) do
    begin
      x := ord(a[i]) - 48;
      sum := x + du*10;
      thuong := sum div b;
      du := sum mod b;
      c := c + chr(thuong + 48);
    end;
  while (length(c) > 1) and (c[1] = '0') do delete(c,1,1);
  divnho := c;
end;
```

phép mod (số nhỏ vs số lớn)

```
function modnho(a : bignum;b : integer) : integer;
begin
  du := 0;
  for i := 1 to length(a) do
    begin
      x := ord(a[i]) - 48;
      sum := x + du*10;
      du := sum mod b;
    end;
  modnho := du;
end;
```

phép div (số lớn vs số lớn)

```
function divlon(a,b : bignum) : bignum;
var
  tich : array[0..10] of bignum;
  i : integer;
  c,du : bignum;
  k : integer;
begin
  tich[0] := '0';
  for i := 1 to 10 do
    tich[i] := cong(tich[i - 1],b);
  du := '';c := '';
  for i := 1 to length(a) do
    begin
      du := du + a[i];
```

```
      k := 1;
      while cmd(du,tich[k]) <> -1 do inc(k);
      dec(k);
      c := c + chr(k + 48);
      du := tru(du,tich[k]);
    end;
  while (length(c) > 1) and (c[1] = '0') do delete(c,1,1);
  divlon := c;
end;
```

phép mod (số lớn vs số lớn)

```
function modlon(a,b : bignum) : bignum;
begin
  tich[0] := '0';
  for i := 1 to 10 do
    tich[i] := cong(tich[i - 1],b);
  du := '';c := '';
  for i := 1 to length(a) do
    begin
      du := du + a[i];
      k := 1;
      while cmd(du,tich[k]) <> -1 do inc(k);
      dec(k);
      du := tru(du,tich[k]);
    end;
  modlon := du;
end;
```

Có các phép tính cơ bản : cộng, trừ, nhân , div, mod.

OTHER :

Dãy fibonacci : $F_n = F_{n-1} + F_{n-2}$
đây là dãy ứng dụng khá nhiều trong các bài toán.

Số Catalan : $Catalan_n = \frac{1}{n+1} C_{2n}^n$

Tính chất sort : với 1 dãy đã sắp tăng, thì phần tử tại `n div 2 + 1` luôn cho đẳng thức sau luôn bé nhất :

$F_{max} = |A_1 - X| + |A_2 - X| + \ldots + |A_N - X|$

Dạng về hàm số chứa tri tuyệt đối : khi gặp những bài chứa trị tuyệt đối, ta phải cẩn thận chia bài toán thành 2 trường hợp giải quyết :D

Tính chất phép mod :
(a + b) mod d = (a mod d + b mod d) mod d
(a – b) mod d = a mod d – b mod d + ord(a mod n < b mod n)*n = (a mod d – b mod d + d) mod d
(a * b) mod d = (a mod d * b mod d) mod d
(a / b) mod d = a x b^{p-2} mod d

Problem :
BCDIV http://vn.spoj.pl/problems/BCDIV/
BIGNUM http://vn.spoj.pl/problems/BIGNUM/
GPT http://vn.spoj.pl/problems/GPT/

PAGAIN http://vn.spoj.pl/problems/PAGAIN/
ILSMATH http://vn.spoj.pl/problems/ILSMATH/
MDIGITS http://vn.spoj.pl/problems/MDIGITS/
CON http://vn.spoj.pl/problems/CON/
MMOD29 http://vn.spoj.pl/problems/MMOD29/
MPRIME http://vn.spoj.pl/problems/MPRIME/
MPRIME1 http://vn.spoj.pl/problems/MPRIME1/
MYSTERY http://vn.spoj.pl/problems/MYSTERY/
FBCFIBO http://vn.spoj.pl/problems/PBCFIBO/
ETF http://vn.spoj.pl/problems/ETF/
MZVRK http://vn.spoj.pl/problems/MZVRK/
NKABD http://vn.spoj.pl/problems/NKABD/
NKNUMFRE http://vn.spoj.pl/problems/NKNUMFRE/
TREENUM http://vn.spoj.pl/problems/TREENUM/
FINDNUM http://vn.spoj.pl/problems/FINDNUM/ (cách tối ưu là sinh số nguyên tố + duyệt :D)
PBCDIV http://vn.spoj.pl/problems/PBCDIV/
SPSUM http://vn.spoj.pl/problems/SPSUM/
VTRI http://vn.spoj.pl/problems/VTRI/
VTRI2 http://vn.spoj.pl/problems/VTRI2/
CT http://vn.spoj.pl/problems/CT/
1189 – Pairs of Integers acm.timus.ru/problem.aspx?space=1&num=1189
11B - Jumping Jack http://codeforces.com/problemset/problem/11/B
9C - Hexadecimal's Numbers http://codeforces.com/problemset/problem/9/C
16C - Monitor http://codeforces.com/problemset/problem/16/C (odd - i x (i + 1) div 2)

NHÂN MA TRẬN :
Nhân ma trận là 1 pp xử lý các lớp trên mảng, chủ yếu xây đựng qua mảng ma trận 3 chiều là a[I,j,k]
Cho 2 ma trận, A kích thước là M x N và B kích thước là N x P.
Kết quả phép nhân ma trận A và B là ma trận C kích thước M x P.

$$C_{i,j} = \sum_{k=1}^{N} A_{i,k} * B_{k,j} \quad (1 \le i \le M, 1 \le j \le P)$$

Để thực hiện được, ta làm như sau :

```
Type
maxtrix = array[1..maxn,1..maxn] of integer;
operator *(a,b : maxtrix) c : maxtrix;
var
  i,j,k : integer;
begin
  for i := 1 to m do
    for j := 1 to n do
      begin
        c[i,j] := 0;
        for k := 1 to p do
          c[i,j] := c[i,j] + a[i,k]*b[k,j];
      end;
end;
```

→ `c := a * b;`

Note :
Nhân ma trận không có tính giao hoán, tức là A * B thì được nhưng A * B thì chưa chắc là đc.
Tuy nhiên , nhân ma trận lại có tính chất kết hợp : A * (B * C) = (A * B) * C.

Ngoài ra, ta có thể thay thế phép cộng như ở trên bằng phép nhân
Code :

```
function nhan(a,b : maxtrix) : maxtrix;
var
  c : maxtrix;
  i,j,k : integer;
begin
  for i := 1 to x do
    for j := 1 to z do
      begin
        c[i,j] := 0;
        for k := 1 to y do
          c[i,j] := max(c[i,j],a[i,k]*a[k,j]);
      end;
end;
```

Dạng 1 : tính toán các lớp, dựa trên việc xây dựng lớp thứ 1 và quy nạp
Vd : xây dựng dãy fibonacci bằng nhân ma trận
CT tính dãy fibo là : $F_{I+1} = F_i + F_{i-1}$.
Ta chia bài toán tính ấy thành các lớp :
Lớp 1 : F_O và F_1
Lớp 2 : F_1 và F_2
...
Lớp n : F_{n-1} và F_n
Bước tiếp theo , ta cố gắng tìm 1 ma trận sao cho từ ma trận này có thể biến đổi lớp I thành lớp I + 1.
Tại dây, bảng của ta là $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$? . tại sao ta xây dựng bảng như v , vì
$F_i = F_{i-1} \times 0 + F_i \times 1$.
$F_{i+1} = F_{i-1} \times 1 + F_i \times 1$.
Note :
Lấy C = A x B thì
Bảng ta vừa xây dựng thường là bảng A.
Bảng các lớp ấy ta gọi là bảng B.
Bảng kết quả C luôn phải phù hợp với kích thước bảng B.

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} F_{i-1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_i \\ F_{i+1} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^T * \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} F_T \\ F_{T+1} \end{pmatrix}$$

Dạng 2 : lớp bài toán cộng ma trận và max, min max trận
Vd : bài toán tìm đường đi ngắn nhất qua k đỉnh .
Ta có :

$$C_1 = A$$
$$C_2 = C_1 \otimes C_1 = A \otimes C_1$$
$$C_3 = C_1 \otimes C_2 = A \otimes C_2$$
$$C_4 = C_1 \otimes C_3 = A \otimes C_3$$
$$\dots$$
$$C_k = C_1 \otimes C_{k-1} = A \otimes C_{k-1}$$
$$\Rightarrow C_k = A^k$$

Code :

```
function nhan(a,b : maxtrix) : maxtrix;
var
  c : maxtrix;
  i,j,k : integer;
begin
  for i := 1 to x do
    for j := 1 to z do
      begin
        c[i,j] := 0;
        for k := 1 to y do
          c[i,j] := max(c[i,j],a[i,k]*a[k,j]);
      end;
end;
```

Problem :

SEQ http://www.spoj.pl/problems/SEQ/
SPP http://ww.spoj.pl/problems/SPP/
C11CAL http://vn.spoj.pl/problems/C11CAL/
LQDFIBO http://vn.spoj.pl/prolbems/LQDFIBO/
INKPRINT http://vn.spoj.pl/problems/INKPRINT/
LQDFIBO2 http://vn.spoj.pl/prolbems/LQDFIBO2/
Problem H http://ipsc.ksp.sk/contests/ipsc2003/real/problems/h.html

CHAP 2
GEOMETRY

Phép đời hình trong hệ tọa độ O(x,y) :

Phép tịnh tiến : $\begin{cases} x = X + a \\ y = Y + b \end{cases}$

Phép quay : I(a,b) → I(-b,a)

Tọa độ trong hình học :

Điểm : ta quy định điểm trên hệ trục (x,y) sẽ bao gồm 2 con số là x – biểu diễn hoành độ và y – biểu điển tung độ. Vd : `I(1,2) , I(-3,5), …`

Đường thẳng

Đường thẳng là tập hợp của nhiều điểm nằm trên 1 đường không giới hạn.

Pt đường thẳng dc viết dưới dạng là :

$$\frac{x - x_B}{x_B - x_A} = \frac{y - y_B}{y_B - y_A}$$

Trong tin học, có 2 dạng đường thẳng thường được sử dụng , đó là :

Phương trình đường thẳng theo hệ số góc : Y = aX + b

Trong đó, a và b được tính từ 2 điểm : $A(x_A,y_A)$ và $B(x_B,y_B)$.

Đưa pt trên thành dạng

$$Y = \frac{y_B - y_A}{x_B - x_A} X + y_B - \frac{y_B - y_A}{x_B - x_A} x_B$$

Trong đó , đặt k = a = $\frac{y_B - y_A}{x_B - x_A}$. ta gọi k là hệ số góc.

Hệ số góc biểu hiện số góc theo tan của đường thẳng hợp với trục hoành , tuy nhiên, 1 góc có thể biểu thị từ 2 hệ số góc , cho nên phải cẩn thận.

Tính chất hệ số góc k :

a > 0 : hệ số góc quay góc nhọn.

a < 0 : hệ số góc quay góc tù.

a = 0 : đường thẳng // với trục hoành.

Mối kiên hệ giữa dt (d) và (d_1)

D // d_1 : k = k_1

D vuông góc với d_1 : k x k_1 = -1

D tạo với Ox 1 góc α : có 2 hệ số góc k = tana hoặc k = tan(180 – a)

D tạo với d_1 1 góc α : $\tan\alpha = \left|\frac{k_1 - k_d}{1 + k_1 k_d}\right|$

Phương trình đường thẳng pháp tuyến :

vector pháp tuyến : vector pháp tuyến(x,y) = vector chỉ phương(-y,x)

phương trình có dạng : **Xa + Yb + c = 0**

trong đó, a,b,c dc tính :

a = y1 – y2

b = x2 – x1

c = $\begin{pmatrix} y2 & x2 \\ y1 & x1 \end{pmatrix}$

Khoảng cách :

H = f(x) / sqrt(A^2 + B^2)

AB = sqrt((x2 – x1)^2 + (y2 – y1)^2)

Quan hệ giữa điểm và đường :

Gọi F(x,y) = ax + by + c. ta có các mối quan hệ sau :

Điểm – đường : F(x,y) = 0

Điểm – đoạn : $\begin{cases} F(x) = 0 \\ \min(x1, x2) <= X <= \max(x1, x2) \\ \min(y1, y2) <= Y <= \max(y1, y2) \end{cases}$

Ngoài ra, ta có thể dùng tích chéo kết hợp với tích chấm (hint : lấy điểm x làm chuẩn)

Điểm – tia : $\begin{cases} F(x) = 0 \\ (x - x1)(y - y1) >= 0 \\ (x - x2)(y - y2) >= 0 \end{cases}$

2 điểm A, B nằm cùng phía : F(A) x F(B) > 0

2 điểm A, B nằm khác phía : F(A) x F(B) < 0

Giao điểm giữa đường với đường :

Đường – đường:

Cách 1 : ta tính D, D_x, D_y.

$$D = \begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix} \qquad D_X = \begin{pmatrix} B_1 & C_1 \\ B_2 & C_2 \end{pmatrix} \qquad D_Y = \begin{pmatrix} C_1 & A_1 \\ C_2 & A_2 \end{pmatrix}$$

Ta có :

2 đường thẳng cắt nhau ⇔ D <> 0 ⇔ $\begin{cases} x = \dfrac{D}{D_x} \\ y = \dfrac{D}{D_y} \end{cases}$

2 đường thẳng // nhau ⇔ $\begin{cases} D = 0 \\ D_x <> 0 \vee D_y <> 0 \end{cases}$

2 đường thẳng trùng nhau ⇔ $\begin{cases} D = 0 \\ D_x = D_y = 0 \end{cases}$

Cách 2 :

Nếu như f3(x1,y1)*f3(x2,y2) < 0 và f1(x3,y3)*f1(x4,y4) < 0 thì 2 dt cắt nhau.

Cách 1 có thể áp dụng cho bài toán quan hệ giữa dừơng và đoạn, đoạn và tia, …

Góc:

Ta tính góc dựa vào quan hệ tọa độ và góc lượng giác

```
begin
  dx := p2.x - p1.x;dy := p2.y - p1.y;
  ax := abs(dx);ay := abs(dy);
  if ax + ay = 0 then t := 0
  else t := dy / (ax  + ay);
  if dx < 0 then t := 2 - t
  else if dy < 0 then  t := 4 + t;
  goc := t * 90;
end;
```

Tích chấm - tích chéo : cho ta mối quan hệ giữa 2 vector

$\vec{a}$ =(x1,y1) , $\vec{b}$ = (x2,y2).

Tích chấm – dot(a,b): đặt trưng cho cos :

Dot($\vec{a},\vec{b}$) = x1 x x2 + y1 x y2

Tính chất :

Dot($\vec{a},\vec{b}$) = 1 → góc hợp = 0^o

Dot($\vec{a},\vec{b}$) = 0 → góc hợp = 90^o

Dot($\vec{a},\vec{b}$) = -1 → góc hợp = 180^o

Cos($\vec{a},\vec{b}$) = dot($\vec{a},\vec{b}$) / sqrt($\bar{a}\,x\,\bar{b}$)

Tích chéo – det(a,b) : đặt trưng cho sin, đặt trưng cho cả hướng của góc quay, hướng rẽ của vector,…

Tích chéo có ứng dụng rất lớn trong nhiều bài toán tin , biến đổi từ phức tạp trở thành đơn giản với độ sai số thấp hơn là viết phương trình ở trên.

Tính chéo được gọi là det($\vec{a},\vec{b}$)

Det($\vec{a},\vec{b}$) = $\begin{vmatrix} x1 & y1 \\ x2 & y2 \end{vmatrix}$ = x1* y2 – x2 * y1;

Tính chất :

Det($\vec{a},\vec{b}$) > 0 : góc quay ngược chiều kim đồng hồ (quay trái) – CCW = 1

Det($\vec{a},\vec{b}$) = 0 : thẳng hàng – CCW = 0

Det($\vec{a},\vec{b}$) < 0 : quay cùng chiều (quay phải) – CCW = -1

Sin(1,2) = Det($\vec{a},\vec{b}$) / sqrt($\bar{a}\,x\,\bar{b}$)

Ta viết được function CWW, CCW biểu diễn góc quay :

```
function CCW(p1, p2, p3: Point): Integer;
var
  a1, b1, a2, b2, t: Real;
begin
  a1 := p2.x - p1.x;
  b1 := p2.y - p1.y;
  a2 := p3.x - p2.x;
  b2 := p3.y - p2.y;
  t := a1*b2 - a2*b1;
  if Abs(t) < Eps then CCW := 0
  else
    if t > 0 then CCW := 1
    else CCW := -1;
```

```
end;
```

Dùng tính tích chéo tìm quan hệ đường – đường : (góc quay det)
2 đường giao nhau : det(a,b) <> 0
2 đường thẳng // : det(a,b) = 0
2 đường thẳng Vuông góc : dot(a,b) = 0
Đoạn AB giao đoạn CD: $\begin{cases} \det(AB,AC) \text{ x } \det(AB,AD) < 0 \\ \det(CD,CB) \text{ x } \det(CD,CA) < 0 \end{cases}$

Vị trí điểm so với đa giác (xác định):
Gọi I(x,y) là điểm cần xét a
Đối với đa giác lồi (đa giác hok tự cắt) , ta có
Cách 1 : O(n)
Bước 1 : Lấy trọng điểm M của đa giác (X = (x1 + x2 + .. xn) / n,Y = (y1 + y2 + .. yn)/n)
Bước 2 : nếu như 2 điểm nằm trong đa giác, tức là cùng phía với đường thẳng tạo nên đa giác lồi ấy, thì I nằm trong đa giác
Cách 2 - O(logn): ta chặt nhị phân trên các điểm.
//Cách dưới day dung cho da giác có sort các diem theo chieu kim dong ho,
//phải có them 2 dk về 1 → 2 → a và 1→ a→ n

```
function check:boolean;
begin
  readln(a.x,a.y);
  if cross(p[1],p[2],a)>=0 then exit(false);
  if cross(p[1],a,p[n])>=0 then exit(false);
  l:=2;r:=n-1;
  while l<r do
    begin
      mid:=(l+r+1) >> 1;
      if cross(p[1],p[mid],a)<=0 then l:=mid else r:=mid-1;
    end;
  exit(cross(p[l],p[l+1],a)<0);
end;
```

Vị trí điểm so với đa giác (đa giác không xác định):
Gọi M(x,y) là điểm cần xét
Lấy N có hoành độ lớn nhất (MN // trục tọa độ)
Nếu như M(x,y) thuộc đường hay đỉnh của bao lồi thì ghi ra luôn :D
Or :
đếm các trường hợp sau : (gọi value là biến dếm)
nếu `a[I + 1]` thuộc MN , `a[i],a[I + 2]` khác phía MN
nếu `a[I - 1],a[I + 2]` khác phía MN, `a[i],a[I + 1]` thuộc MN
nếu `d[a[I],a[I + 1]]` cắt MN
nếu như `dem mod 2 = 1` thì điểm năm trong đa giác, = 0 thì không năm trong đa giác.

Đường tròn :
Phương trình : $(x_1 - x_0)^2 + (y_1 - y_0)^2 = R^2$ với (x_o,y_o) là tâm của đường tròn
điểm thuộc đường tròn khi $(x_1 - x_0)^2 + (y_1 - y_0)^2 <= R^2$

Hình tính – điện tích :
Các tính chất của tam giác , tứ giác đặt biệt, đa giác (toán)

Area :
Tam giác :
S = a * h / 2;
S = b * c * sinA / 2
S = sqrt (p(p - a)(p - b)(p - c))
S = abs(det(AB,AC) / 2)
S = 4R/(a*b*c)
P = (a + b + c) → p = (a + b+ c) / 2
S = r^2p
Các đường đặt biệt của tam giác :
H = 2*sqrt(p(p - a)(p - b)(p - c)) / a

đa giác : $S = \sum_{i=1}^{N} \frac{(x[i+1] - x[i])(y[i+1] + y[i])}{2}$ `(a[n + 1] = a[1])`

COMPUTING – GEOMETRY
Đa giác:
Kiểm tra đa giác lồi :
Đa giác lồi ➔ với mọi `d[a[i],a[I + 1]], a[I - 1]` và `a[I + 2]` phải nằm cùng phía với d – O(n).
Thuật toán tìm kiếm bao lồi lớn nhất:
Thuật toán bọc gói : O(n^2) - greedy
Thuật toán quét graham :
Bước 1 : ta sort các đỉnh có góc quay hợp với trục hoành theo chiều tăng dần của góc.
Bước 2 : nạp đỉnh 1 vào bao lồi
Bước 3 : dùng thủ tục CCW quét, nếu như 3 cặp điểm có CCW = 1 thì ta nạp vào, không thì xóa điểm đó và bắt đầu xét CCW lại với bộ 3 điểm mới – O(n)
Thuật toán Andrew :
Bạn có thể kham khảo thuật toán tại đây :
http://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain
Thuật toán này chỉ khác thuật toán graham ở đây là làm graham 2 lần, chỉ nhận các hướng quay phải (ccw <> -1) và có thể giải quyết dc 3 điểm thẳng hàng.

Hint in uses :
Vấn đề về sai số - sử dụng `esp = 1e-9` hay `1e-6` tùy vào bài toán.
Sort đỉnh theo tiêu chí (tung – hoành,góc quanh) : O(n^2) → O(nlogn)
Hình học có thể chứa Greedy algorigthm, DP, quan trọng nhất là một số bài toán hình học chuyển về đồ thị
Phân hoạch (chia nhỏ bài toán hình ra - hard)
Nghiệm dc định tính trên một khoảng [l,r] nào đó – tìm bằng cách chặt nhị phân
Note : các phần hình học liên quan tới dp , gird sẽ được trình bày trong các phần khác.

Problem :
FIRE http://vn.spoj.pl/problems/FIRE/
GPMB http://vn.spoj.pl/problems/GPMB/
HINHTHOI http://vn.spoj.pl/problems/HINHTHOI/
METERAIN http://vn.spoj.pl/problems/METERAIN/
MILITARY http://vn.spoj.pl/problems/MILITARY/
NEAREST http://vn.spoj.pl/problems/NEAREST/

NKLAND http://vn.spoj.pl/problems/NKLAND/
NKPOLI http://vn.spoj.pl/problems/NKPOLI/ (dp)
NKPOLY http://vn.spoj.pl/problems/NKPOLY/ (dp)
PRAVO http://vn.spoj.pl/problems/PRAVO/
QBPOINT http://vn.spoj.pl/problems/QBPOINT/
VECTOR http://vn.spoj.pl/problems/VECTOR/
WALK http://vn.spoj.pl/problems/WALK/
CVJETICI http://vn.spoj.pl/problems/CVJETICI/
NKSPILJA http://vn.spoj.pl/problems/NKSPILJA/
24C - Sequence of points http://codeforces.com/problemset/problem/24/C
136D - Rectangle and Square http://codeforces.com/problemset/problem/136/D
1C - Ancient Berland Circus http://codeforces.com/problemset/problem/1/C
166B – Polygons http://codeforces.com/problemset/problem/166/B
50C - Happy Farm 5 http://codeforces.com/problemset/problem/50/C
136D - Rectangle and Square http://codeforces.com/problemset/problem/136/D
1065. Frontier http://acm.timus.ru/problem.aspx?space=1&num=1065
1020. Rope http://acm.timus.ru/problem.aspx?space=1&num=1020
1043. Cover an Arc http://acm.timus.ru/problem.aspx?space=1&num=1043
1046. Geometrical Dreams http://acm.timus.ru/problem.aspx?space=1&num=1046
1052. Rabbit Hunt http://acm.timus.ru/problem.aspx?space=1&num=1052
1084. Goat in the Garden http://acm.timus.ru/problem.aspx?space=1&num=1084
1103. Pencils and Circles http://acm.timus.ru/problem.aspx?space=1&num=1103
1111. Squares http://acm.timus.ru/problem.aspx?space=1&num=1111
1185. Wall http://acm.timus.ru/problem.aspx?space=1&num=1185
1207. Median on the Plane http://acm.timus.ru/problem.aspx?space=1&num=1207
1215. Exactness of Projectile Hit http://acm.timus.ru/problem.aspx?space=1&num=1215

CHAP 3
COMPLETE SEARCH

GENERATION :

Phương pháp : bài toán sinh là bài toán mà đang ở cấu hình này, ta sinh ra cấu hình/ trạng thái tiếp theo từ trạng thái ta đang xét

bài toán sinh có cấu trúc thực hiện như sau :

cách làm chung :

```
repeat
<ghi cau hinh hien tai ra>
<sinh ra cau hinh ke tiep neu con> :
→ tìm vị trí cấu hình tại vị trí I mà nó chưa đạt tới cấu hình max
(dec(i))
→ nếu tìm dc (I > 0)
{
cập nhật cấu hình (tăng thêm 1 bậc)
trả cấu hình thấp hơn 1 bậc cho I + 1 …n
}
until <het cau hinh>
```

note :

khi nạp tai vị trí I cấu hình trên , thì phải trả lại cho I + 1 … n cấu hình thấp hơn cấu hình I 1 bậc

sinh cấu hình rất hay áp dụng cho các bài toán điển hình như là sinh dãy nhị phân, sinh chỉnh hợp không lặp, sinh dãy tổ hợp, sinh hoán vị, …

xem thêm về định nghĩa và quy tắc của thứ tự từ điển – cẩn thận với thứ tự "="

phân biệt và xát định cấu hình trên , cấu hình dưới

chỉ hợp có cấu hình trên là : `p[I – 1] + 1 <= p[i] <= n – k + i`

COMPLETE SEARCH : (backtracking)

cách làm bằng cách liệt kê mọi cấu hình phần tử xem coi nó có thỏa hay hok

sử dụng đệ quy để tìm các bài toán ấy :

quy trình thực hiện có thể viết như sau :

```
backtracking (i)
{
      for <mọi giá trị có thể gán cho x[i]> {
            <update gia tri cho x[i] + mot so thanh phan>
            if <cau hinh cap nhan day du> → ghi cấu hình ra
                  else bactracking(I + 1)
            <reupdate gia tri cho x[i] + mot so thanh phan (neu can)>
      }
}
```

Note :

mọi giá trị gán cho x[i] cũng thỏa quy tắc cấu hình trên , cấu hình dưới

để giảm đi giá trị gán cho x[i] ta dùng cái tích chất toán học

example : các phép đếm, phân tích số, bài toán xếp hậu

BRANCH AND BOUND : phương pháp tìm cấu hình hay phần tử tối ưu của yêu càu bài toán

```
Init() // khoi tao cau hinh dau tien hay object can thiet
{}
branch(i)
{

Khi không còn duyệt tiếp, so khớp tối ưu để cập nhật

for <moi gia tri co the gan cho x[i]>
{
<gán cho x[i] + một số thành phần khác>
if  <giá trị tốt hơn giá trĩ tối ưu hiện tại>
if <đã nạp bài toán đến kết quả>
else branch(I + 1)
<trả về các thành phần đã lưu>
}
}
```

một lưu ý khá quan trọng, ta có thể giới hạn số lân duyệt bằng :
giá trị tại bước I + giá trị thấp(cao) nhất của n – I bước nữa so với giá trị tối ưu hiện tại
→ bước tối ưu khá hiệu quả trong các phép duyệt

Note :
dùng nhiều điều kiện và tính chất đúng đắn để chương trình chạy nhanh hơn
dùng các heusristic để giải quyết các bài toán trong thời gian ngắn hơn với thời gian tối ưu
dùng toán học, tính chất của dãy số hay quy luật để giảm thiển số lần duyệt
debug bằng cây sẽ hiệu quả hơn debug bình thường :D

DUYỆT ƯU TIÊN :
Ta làm như sau :
B1 : sắp sếp các phần tử theo 1 tiêu chí tốt nào đó
B2 : từ các phần tử sắp xếp đó ta đưa ra 1 chiến thuật duyệt hợp lí để tránh trường hop WA và giảm bị TLE :D

Note 1: pp này không những áp dụng đước với trường hợp đệ quy quay lui mà nó còn áp dung cho cả cái bài toán thiêng về ad hoc với dữ liệu lớn và chưa nghĩ ra dc pp suy nghĩ kịp thời :D
Note 2 : pp này cũng làm tăng tốc độ của phép duyệt đệ quy khi ta chắc chắn pp của ta đúng , ví dụ diển hình là bài toán cái túi với bước đầu là duyệt ưu tiên theo giá trị t = c / a

DUYỆT CHIA ĐÔI TẬP HỢP
Là tập hợp những bài toán về duyệt tổ hợp mà ta có thể chia nó làm 2 bài toán con và duyệt 2 phần riêng biệt
Độ phức tạp nếu như duyệt tổ hợp thông thường thường là $O(2^N)$ nhưng bây giờ chỉ giảm xuống còn $O(2^{2/N})$
Note : thường sau phép duyệt đầu, ta thường hay sort giá trị theo 1 phương án nào đó, sau đó ở phép duyệt lần thứ 2 , ta sẽ so khớp tối ưu .
So khớp tối ưu ở đây ta sẽ dùng các pp chủ yếu là tìm kiếm phần tử thõa mảng yêu cau đặt ra của bài toán.

Problem :
NKMINES http://vn.spoj.pl/problems/NKMINES/

FINDNUM http://vn.spoj.pl/problems/FINDNUM/
V8SCORE http://vn.spoj.pl/problems/V8SCORE/
PIZZALOC http://vn.spoj.pl/problems/PIZZALOC/
UVA – 181
FIRE http://vn.spoj.pl/problems/FIRE/
MVECTOR http://vn.spoj.pl/problems/MVECTOR/
VECTOR http://vn.spoj.pl/problems/VECTOR/
COIN34 http://vn.spoj.pl/problems/COIN34/
CHNREST http://vn.spoj.pl/problems/CHNREST/
LQDDIV http://vn.spoj.pl/problems/LQDDIV/

Ứng dụng phương pháp quy nạp toán học

Nguyễn Duy Khương

Trong toán học, quy nạp là một phương pháp đơn giản nhưng hiệu quả để chứng minh các bài toán. Ở bài viết này tôi xin đưa ra một ứng dụng nhỏ của nó trong việc giải các bài toán tin học:

1. Thuật toán quy nạp quy nạp(nhắc lại):

Giả sử có bài toán F cần chứng minh đúng với mọi n $\in$ N. Ta chứng minh bài toán đúng bằng cách quy nạp, cần tiến hành các bước sau:
- n = 1: mệnh đề cần chứng minh đúng.
- Giả sử n = k: mệnh đề cần chứng minh đúng.
- n = k + 1: ta cần chứng nó cũng đúng. Vậy theo nguyên lý quy nạp bài toán đúng với mọi N.

Trong tin học, thuật toán nay cũng được áp dụng. Tuy thuật toán đơn giản nhưng nó lại được áp dụng một cách rất linh động và khéo léo trong các bài toán tin.

2. Phát biểu bài toán tổng quát giải bằng quy nạp:

Thông thường bài toán giải bằng quy nạp không phải là một bài toán tối ưu hoá. Nó chỉ đơn giản là bài toán cần chỉ ra cách biến đổi theo quy luật cho trước để thu được kết quả mong đợi. Và bài toán đó thường được phát biểu như sau:Cho N đối tượng và một số thao tác biến đổi. Yêu cầu sử dụng các thao tác biến đổi để thu được kết mong đợi.

Cách làm thông thường:
- Nếu n = 0; 1: ta luôn có cách biến đổi đúng.
- Nếu có n > 1 mà ta luôn chỉ ra một cách biến đổi sao cho giản bớt được số đối tượng mà điều kiện bài toán vẫn không thay đổi.
- Như vậy vì số đối tượng trong một bài toán tin học luôn là hữu hạn nên sau một số hữu hạn bước thì số lượng đối tương bằng 1 hoặc 0. Suy ra bài toán được giải quyết một cách hoàn toàn.

3. Các ví dụ cụ thể:

P1. Cho N vecto v1, v2, …, vn độ dài nhỏ hơn L cho trước. Cần đặt trước các vecto một dấu cộng hoặc trừ sao cho vecto tổng thu được có độ dài nhỏ hơn căn 2 nhân L.Input: Segment.In
- Dòng đầu ghi số N (1 < N ≤ 10000) và L (0 < L ≤ 15000.00)
- N dòng tiếp theo mỗi dòng ghi một cặp số xi, yi là toạ độ của vecto thứ i. (|xi|, |yi| ≤ 10000).

Ouput: Segment.out
- Dòng thứ nhất ghi YES nó có thể đặt các dấu cộng trừ thoả mãn yêu cầu bài toán, NO trong trường hợp ngược lại.
- Nếu là YES dòng thứ hai ghi N kí tự cộng hoặc trừ cần đặt trước các vecto sao cho tổng vecto nhỏ hơn sprt(2)*L.

Ví dụ:

Thuật giải:
- n = 1 bài toán đã đúng.
- n = 2 có trường hợp xảy ra với hai vecto:

+ góc giữa hai vecto nhỏ hơn 90 độ, lấy vecto 1 trừ cho vecto 2 lúc đó ta thu được 1 vecto co độ dài nhỏ

hơn sqrt(2) * L.
+ góc giữa hai vecto lớn hơn hoặc bài một 90 độ, lấy vecto 1 cộng cho vecto 2 lúc đó ta thu được 1 vecto có độ dài nhỏ hơn sqrt(2) * L.
- n ≥ 3 suy ra luôn có 2 vecto mà góc giữa hai phương của vecto nhỏ hơn hoặc 60 bằng độ có hai trường xảy ra:
+ góc giữa hai vecto đó nhỏ hơn hoặc bằng 60 độ, trừ vecto 1 cho vecto 2 thay hai vecto bằng vecto mới nhận được.
+ góc giữa hai vecto đó lớn hơn hoặc bằng 120 độ, cộng vecto 1 cho vecto 2 thay hai vecto bằng vecto mới nhận được. Lưu ý sau này nếu trừ vecto mới nhận được phải trừ (đổi dấu) toàn vecto bộ hình thành lên nó, vecto mới nhận được có độ dài nhỏ hơn L, số đoạn thẳng giảm 1.
Như vậy đây là bài toán luôn có lời giải, độ phức tạp O(N^2) nếu xử lý khéo có thể giản xuống O(n*log(n)).
Chương trình mô tả thuật giải:

```
Procedure Xuly (n: integer //* số đối tượng *//);
Begin
If n <= 1 then exit else
If n = 2 then
Begin
If goc (v1, v2) <= 90 độ Then đổi dấu (v2);
//* đổi dấu tức là đổi dấu toàn bộ vecto thành phần *//
Thay hai vecto bằng 1 vecto (v1 + v2);
//* vecto mới gồm các thành phần của v1 và v2 *//
//* lúc này v2 nếu cần thiết đã đổi dấu rồi *//
Exit;
End else
Begin
lấy 3 vecto bất kỳ;
chọn ra được 2 vecto v1, v2 sao cho
góc nhỏ giữa phương 2 vec to nhỏ 60;
if goc (v1, v2) <= 60 then đổi dấu (v2);
Thay hai vecto này bằng (v1 + v2);
End;
End;
```

P2: Utopia (IOI 2002)

Đất nước Utopia tươi đẹp bị chiến tranh tàn phá. Khi chiến tranh tạm ngừng, đất nước bị chia cắt thành bốn miền bởi một đường kinh tuyến (đường theo hướng Bắc-Nam) và một đường vĩ tuyến (đường theo hướng Đông-Tây). Giao điểm của hai đường này xem như điểm (0, 0). Tất cả các miền này đều muốn mang tên Utopia, theo thời gian các miền này được gọi là Utopia 1 (phía Đông Bắc), Utopia 2 (phía Tây Bắc), Utopia 3 (phía Tây Nam), Utopia 4 (phía Đông Nam). Một điểm trong bất kỳ miền nào cũng được xác định bởi khoảng cách theo hướng Đông và khoảng cách theo hướng Bắc của nó so với điểm (0, 0). Bộ hai khoảng cách này được gọi là toạ độ của điểm. Các thành phần của toạ độ có thể là số âm; do đó một điểm miền Utopia 2 có toạ độ (âm, dương), một điểm miền Utopia 3 có toạ độ (âm, âm), một điểm miền Utopia 4 có toạ độ (dương, âm), một điểm miền Utopia 1 có toạ độ (dương, dương) (Giống như các góc phần tư trong hệ toạ độ đề các).
Một vấn đề là các công dân không được phép đi qua biên giới giữa các miền. May thay, một số thí sinh tham dự IOI của Utopia sáng chế ra một loại máy an toàn để vận chuyển từ xa. Máy này cần các số mã, mỗi số mã chỉ có thể được dùng một lần. Bây giờ, nhóm thí sinh tham dự và bạn phải hướng dẫn máy vận chuyển từ xa xuất phát từ vị trí ban đầu (0, 0) đến các miền của Utopia theo một trình tự cho trước. Khi bạn

nhận được một dãy N số hiệu miền mà theo trình tự đó máy vận chuyển từ xa phải hạ cánh, với mỗi miền, bạn không cần phải quan tâm đến việc hạ cánh tại điểm nào của miền đó miễn là điểm đó thuộc miền yêu cầu. Người ta có thể yêu cầu máy hạ cánh liên tiếp hai lần hay nhiều lần hơn ở cùng một miền. Sau khi rời khỏi điểm ban đầu (0, 0), bạn không được hạ cánh trên các đường biên giới.
Bạn sẽ nhận được input là một dãy 2N số mã và phải viết chúng thành một dãy gồm N cặp mã, sau khi đặt một dấu + hoặc một dấu - thích hợp trước mỗi số. Nếu hiện tại bạn ở điểm (x,y) và sử dụng cặp mã số (+u,-v), bạn sẽ được chuyển tới điểm (x+u,y-v). Bạn có 2N số và bạn có thể sử dụng các số này theo thứ tự bạn muốn, mỗi số kèm theo một dấu + hoặc một dấu -.
Giả sử bạn có các số mã 7, 5, 6, 1, 3, 2, 4, 8 và phải hướng dẫn máy vận chuyển từ xa lần lượt đến các miền thuộc dãy miền 4, 1, 2, 1. Dãy các cặp mã (+7, -1), (-5, +2), (-4, +3), (+8, +6) đáp ứng yêu cầu này vì nó đưa bạn từ (0,0) lần lượt đến các vị trí (7, -1), (2, 1), (-2, 4) và (6, 10). Những điểm này nằm tương ứng ở Utopia 4, Utopia 1, Utopia 2 và Utopia 1.

Nhiệm vụ:

Cho 2N số mã khác nhau và một dãy N số hiệu miền là dãy các miền mà máy vận chuyển từ xa phải lần lượt hạ cánh. Hãy xây dựng một dãy các cặp mã từ các số đã cho để hướng dẫn máy vận chuyển từ xa lần lượt đi đến các miền thuộc dãy đã cho.

Input

Chương trình của bạn phải đọc từ input chuẩn. Dòng đầu gồm một số nguyên dương N ($1 \le N \le 10000$). Dòng thứ hai gồm 2N số mã nguyên khác nhau, tất cả đều là các số nguyên dương không lớn hơn 100000. Dòng cuối cùng gồm một dãy N số hiệu miền, mỗi một trong các số đó là 1, 2, 3 hoặc 4. Hai số liên tiếp trên một dòng cách nhau đúng một dấu trống.

Output

Chương trình của bạn phải viết ra output chuẩn. Ouput gồm N dòng, mỗi dòng chứa một cặp số mã mà trước mỗi số có dấu + hoặc dấu -. Đó là các cặp mã sẽ hướng dẫn cho máy vận chuyển đến dãy miền đã cho. Chú ý rằng không được có dấu trống sau dấu + hoặc dấu - nhưng phải có một dấu trống sau số mã thứ nhất.Nếu có nhiều lời giải, chương trình của bạn có thể đưa ra một lời giải bất kỳ trong số đó. Nếu không có lời giải, chương trình của bạn phải đưa ra một số 0 duy nhất.

Ví dụ:

Thuật giải:
Đây cũng là bài toàn luôn giải được để giải bài toán trên ta cần giải bài toán nhỏ sau:Cho một gồm N số nguyên dương khác nhau a1 < a2 < … < an. Cần sắp xếp lại và thêm các dấu vào các số ai sao cho tổng các số từ 1 đến vị trí i là dương hoặc âm biết trước:
Ví dụ:
Dãy 3 số: 1 2 3
Dấu cần xây dựng: +-+
Một cách thêm dấu và sắp xếp: +1 -2 + 3

Bổ đề:
Nếu dãy có dãy {ai} dương tăng và một chuỗi dấu cần biến đổi thì ta luôn có cách thêm dấu và thay đổi vị trí sao cho tổng cách số từ 1 đến i mang đấu của chuỗi dấu biết trước.
Chứng minh:
- số an mang dấu cuối cùng của chuỗi trên, các số còn lại đan dấu. (điều kiện về dấu để luôn xây được dãy)
Ví dụ: {ai} = {1, 5, 10}; S = '++-'; suy ra dãy {ai} được đánh dấu như sau: -1, +5, -10;
- nhận xét rằng: tổng của n số trên cuối cùng sẽ mang dấu của S[n]
- với n = 1 ta nhận được dãy cần tìm.
- Giả sử n =k ta xây dựng được chuỗi như trên.
- n = k + 1 chia hai trường hợp:
+ s[n-1] cùng dấu s[n]: đặt a[1] vào vị trí cuối cùng và lấy phần tử a[1] và s[n] ra khỏi dãy, nhưng còn là (n - 1) số và (n-1) dấu và dấu cuối cùng với số cuối, dãy đan dấu (vẫn thoả mãn điều kiện về dấu)
+ s[n - 1] và s[n] khác dấu: đặt a[n] vào vị trí cuối cùng và lấy phần tử a[n] và s[n] ra khỏi dãy, nhưng còn là (n - 1) số và (n-1) dấu và dấu cuối cùng với số cuối, dãy đan dấu. Theo giả thiết quy nạp thì dãy còn lại luôn xây dựng được.
Chương trình:

```
Procedure Xuly (Var a, dau, q: array [1…10000] of integer; n, cd, ct : integer);
//*ta đang xử lý dãy các số từ a[cd] đến a[ct] có dấu từ dấu[1] đến dấu[n] *//
Begin
If n <= 1 then exit;
If dau[n - 1] = dau[n] then
Begin
q[n] := a[cd];
Xuly (a, dau, q, n - 1, cd + 1, ct);
End else
Begin
q[n] := a[ct];
Xuly (a, dau, q, n - 1, cd, ct - 1);
End;
End;
//* mảng q thu được là kết quả cần tìm*//
```

Ở bài toán trên các dấu tọa độ x, y là đã xác định: Chia dãy 2N phần tử khác nhau thành hai gồm N phần tử làm hoành độ và tung độ, sau đó sắp tăng. Tiến hành thuật giải trên ta thu được hai dãy toạ độ thoả mãn điều kiện về dấu, đó cũng chính là dãy toạ độ cần tìm.

P3: (Bài tập tự giải) 2N point (POI)

Cho 2N diểm trên mặt phẳng gồm N diểm màu xanh và N điểm màu trắng sao cho không có ba điểm nào thẳng hàng. Hãy tìm N đoạn thẳng mà mỗi đoạn thẳng nối giữa một điểm màu xanh và một điểm màu trắng và các doạn nay không cắt nhau.
Input: 2Npoint.Int
- Dòng dầu ghi số N. (1 ≤ N ≤ 5000)
- N dòng tiếp theo mỗi dòng ghi toạ độ xi, yi của điểm màu xanh (|xi|, |yi| ≤ 10000).
- N dòng tiếp theo mỗi dòng ghi toạ độ xi, yi của điểm màu trắng (|xi|, |yi| ≤ 10000).
Output: 2Npoint.Out
- Gồm N dòng mỗi dòng ghi hai số a, b để chỉ điểm màu xanh thứ a, nối với điểm màu trắng thứ b.

4. Lời kết:

.Mọi tư duy toán học đều có thể vận dụng một cách linh động trong tin học. Tôi hy vọng sau bài viết này, ngoài việc học thêm được một hướng giải mới, các bạn sẽ tìm tòi thêm các ứng dụng toán học khác vào tin học.