

Chuyên đề 2

CÁC KIẾN THỨC CƠ BẢN

1. Hệ đếm

Hệ đếm được hiểu là tập các kí hiệu và quy tắc sử dụng tập các kí hiệu đó để biểu diễn và xác định giá trị các số. Trong hệ đếm cơ số b ($b > 1$), các kí hiệu được dùng có các giá trị tương ứng $0, 1, \dots, b - 1$. Giả sử N có biểu diễn:

$$d_n d_{n-1} d_{n-2} \dots d_1 d_0, d_{-1} d_{-2} \dots d_{-m}$$

trong đó $n + 1$ số các chữ số bên trái, m là số các chữ số bên phải dấu phân chia phần nguyên và phần phân của số N và các d_i phải thoả mãn điều kiện

$$0 \leq d_i < b (-m \leq i \leq n).$$

Khi đó giá trị của số N được tính theo công thức:

$$N = d_n b^n + d_{n-1} b^{n-1} + \dots + d_0 b^0 + d_{-1} b^{-1} + \dots + d_{-m} b^{-m} \quad (1)$$

Chú ý: Để phân biệt số được biểu diễn ở hệ đếm nào người ta viết cơ số làm chỉ số dưới của số đó. Ví dụ: N_b là biểu diễn N ở hệ đếm b .

1.1. Các hệ đếm thường dùng:

Hệ thập phân (hệ cơ số 10) dùng 10 kí hiệu 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ví dụ: $28,9_{10} = 2 \times 10^1 + 8 \times 10^0 + 9 \times 10^{-1}$

Hệ nhị phân (hệ cơ số 2) chỉ dùng hai kí hiệu 0, 1

Ví dụ: $10_2 = 1 \times 2^1 + 0 \times 2^0 = 2_{10}$

$101,1_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} = 5,5$

Hệ cơ số mười sáu, còn gọi là hệ hexa, sử dụng các kí hiệu 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, trong đó A, B, C, D, E, F có các giá trị tương ứng 10, 11, 12, 13, 14, 15 trong hệ thập phân

Ví dụ: $AF0_{16} = 10 \times 16^2 + 15 \times 16^1 + 0 \times 16^0 = 2800_{10}$

1.2. Chuyển đổi biểu diễn số ở hệ thập phân sang hệ đếm cơ số khác

Để chuyển đổi biểu diễn một số ở hệ thập phân sang hệ đếm cơ số khác, trước hết ta tách phần nguyên và phần phân rồi tiến hành chuyển đổi từng phần, sau đó ghép lại.

Chuyển đổi biểu diễn phần nguyên: Từ (1) ta lấy phần nguyên:

$$X = d_n b^n + d_{n-1} b^{n-1} + \dots + d_0 \text{ (trong đó } 0 \leq d_i < b).$$

Do $0 \leq d_0 < b$ nên khi chia X cho b thì phần dư của phép chia đó là d_0 còn thương số X_1 sẽ là: $d_n b^{n-1} + d_{n-1} b^{n-2} + \dots + d_1$. Tương tự d_1 là phần dư của phép chia X_1 cho b . Quá trình được lặp cho đến khi nhận được thương bằng 0.

Chuyển đổi biểu diễn phần phân: Từ (1) ta lấy phần sau dấu phẩy:

$$Y = d_{-1} b^{-1} + \dots + d_{-m} b^{-m}.$$

$$Y_1 = Y \times b = d_{-1} + d_{-2} b^{-1} + \dots + d_{-m} b^{-(m-1)}$$

Ta nhận thấy d_{-1} chính là phần nguyên của kết quả phép nhân, còn phần phân của kết quả là $Y_2 = d_{-2} b^{-1} + \dots + d_{-m} b^{-(m-1)}$. Quá trình được lặp cho đến khi nhận đủ số chữ số cần tìm.

2. Số nguyên tố

Một số tự nhiên p ($p > 1$) là số nguyên tố nếu p có đúng hai ước số là 1 và p .

Ví dụ các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23, ...

2.1. Kiểm tra tính nguyên tố

a) Để kiểm tra số nguyên dương n ($n > 1$) có là số nguyên tố không, ta kiểm tra xem có tồn tại một số nguyên k ($2 \leq k \leq n - 1$) mà k là ước của n (n chia hết k) thì n không phải là số nguyên tố, ngược lại n là số nguyên tố.

Nếu n ($n > 1$) không phải là số nguyên tố, ta luôn có thể tách $n = k_1 \times k_2$ mà $2 \leq k_1 \leq k_2 \leq n - 1$. Vì $k_1 \times k_2 \leq k_1 \times k_2 = n$ nên $k_1 \leq \sqrt{n}$. Do đó, việc kiểm tra với k từ 2 đến $n - 1$ là không cần thiết, mà chỉ cần kiểm tra k từ 2 đến \sqrt{n} .

```
function is_prime(n:longint):boolean;
var k :longint;
begin
  if n=1 then exit(false);
```

```

for k:=2 to trunc(sqrt(n)) do
    if (n mod k=0) then exit(false);
    exit(true);
end;

```

Hàm `is_prime(n)` trên tiến hành kiểm tra lần lượt từng số nguyên k trong đoạn $[2, \sqrt{n}]$, để cải tiến, cần giảm thiểu số các số cần kiểm tra. Ta có nhận xét, để kiểm tra số nguyên dương n ($n > 1$) có là số nguyên tố không, ta kiểm tra xem có tồn tại một số nguyên tố k ($2 \leq k \leq \sqrt{n}$) mà k là ước của n thì n không phải là số nguyên tố, ngược lại n là số nguyên tố. Thay vì kiểm tra các số k là nguyên tố ta sẽ chỉ kiểm tra các số k có tính chất giống với tính chất của số nguyên tố, có thể sử dụng một trong hai tính chất đơn giản sau của số nguyên tố:

- 1) Trừ số 2 và các số nguyên tố là số lẻ.
- 2) Trừ số 2, số 3 các số nguyên tố có dạng $6k \pm 1$ (vì số có dạng $6k \pm 2$ thì chia hết cho 2, số có dạng $6k \pm 3$ thì chia hết cho 3).

Hàm `is_prime2(n)` dưới đây kiểm tra tính nguyên tố của số n bằng cách kiểm tra xem n có chia hết cho số 2, số 3 và các số có dạng $6k \pm 1$ trong đoạn $[5, \sqrt{n}]$.

```

function is_prime2(n:longint):boolean;
var k,sqrt_n:longint;
begin
    if (n=2) or (n=3) then exit(true);
    if (n=1) or (n mod 2=0) or (n mod 3=0) then exit(false);
    sqrt_n:=trunc(sqrt(n));
    k:=-1;
    repeat
        inc(k, 6);
        if (n mod k=0) or (n mod (k+2)=0) then break;
    until k>sqrt_n;
    exit(k>sqrt_n);
end;

```

b) Phương pháp kiểm tra số nguyên tố theo xác suất

Từ định lí nhỏ Fermat:

|| nếu p là số nguyên tố và a là số tự nhiên thì $a^p \bmod p = a$

Ta có cách kiểm tra tính nguyên tố của Fermat:

|| nếu $2^n \bmod n \neq 2$ thì n không là số nguyên tố
 || nếu $2^n \bmod n = 2$ thì nhiều khả năng n là số nguyên tố

Ví dụ:

$2^9 \bmod 9 = 512 \bmod 9 = 8 \neq 2$, do đó số 9 không là số nguyên tố.

$2^3 \bmod 3 = 8 \bmod 3 = 2$, do đó nhiều khả năng 3 là số nguyên tố, thực tế 3 là số nguyên tố.

$2^{11} \bmod 11 = 2048 \bmod 11 = 2$, do đó nhiều khả năng 11 là số nguyên tố, thực tế 11 là số nguyên tố.

2.2. Liệt kê các số nguyên tố trong đoạn $[1, N]$

Cách thứ nhất là thử lần lượt các số m trong đoạn $[1, N]$, rồi kiểm tra tính nguyên tố của m .

```

procedure generate(N:longint);
var m :longint;
begin
  for m:=2 to N do
    if is_prime(m) then writeln(m);
end;
  
```

Cách này đơn giản nhưng chạy chậm, để cải tiến có thể sử dụng các tính chất của số nguyên tố để loại bỏ trước những số không phải là số nguyên tố và không cần kiểm tra các số này.

Cách thứ hai là sử dụng sàng số nguyên tố, như sàng Eratosthene, liệt kê được các số nguyên tố nhanh, tuy nhiên nhược điểm của cách này là tốn nhiều bộ nhớ. Cách làm được thực hiện như sau:

Trước tiên xoá bỏ số 1 ra khỏi tập các số nguyên tố. Số tiếp theo số 1 là số 2, là số nguyên tố, xoá tất cả các bội của 2 ra khỏi bảng. Số đầu tiên không bị xoá sau số 2 (số 3) là số nguyên tố, xoá các bội của 3... Giải thuật tiếp tục cho đến khi gặp số nguyên tố lớn hơn \sqrt{N} thì dừng lại. Tất cả các số chưa bị xoá là số nguyên tố.

```

{$M 1100000}
procedure Eratosthene(N:longint);
const MAX      = 1000000;
var i,j       :longint;
    Prime    :array [1..MAX] of byte;
begin
  
```

```

fillchar(Prime, sizeof(Prime), 0);
for i:=2 to trunc(sqrt(N)) do
  if Prime[i]=0 then
    begin
      j:=i*i;
      while j<=N do
        begin
          Prime[j]:=1;
          j:=j+i;
        end;
    end;
  for i:=2 to N do
    if Prime[i]=0 then writeln(i);
end;

```

3. Ước số, bội số

3.1. Số các ước số của một số

Giả sử N được phân tích thành thừa số nguyên tố như sau:

$$N = a^i \times b^j \times \dots \times c^k$$

Ước số của N có dạng: $a^p \times b^q \times \dots \times c^r$ trong đó

$$0 \leq p \leq i, 0 \leq q \leq j, \dots, 0 \leq r \leq k.$$

Do đó, số các ước số của N là $(i+1) \times (j+1) \times \dots \times (k+1)$.

Ví dụ:

$N = 100 = 2^2 \times 5^2$, số ước số của 100 là: $(2+1)(2+1) = 9$ ước số (các ước số đó là: 1, 2, 4, 5, 10, 20, 25, 50, 100).

$N = 24 = 2^3 \times 3$, số ước số của 24 là: $(3+1)(1+1) = 8$ ước số (các ước số đó là: 1, 2, 3, 4, 6, 8, 12, 24).

3.2. Tổng các ước số của một số

$$N = a^i \times b^j \times \dots \times c^k$$

Đặt $N1 = b^j \times \dots \times c^k$

Gọi $F(t)$ là tổng các ước của t , ta có,

$$F(N) = F(N1) + a \times F(N1) + \dots + a^i \times F(N1)$$

$$\begin{aligned}
 &= (1 + a + \dots + a^i) \times F(N1) = \frac{(a^{i+1} - 1)}{a-1} \times F(N1) \\
 &= \frac{(a^{i+1} - 1)}{a-1} \times \frac{(b^{j+1} - 1)}{b-1} \times \dots \times \frac{(c^{k+1} - 1)}{c-1}
 \end{aligned}$$

Ví dụ: Tổng các ước của 24 là:

$$\frac{(2^{3+1} - 1)}{2-1} \times \frac{(3^{1+1} - 1)}{3-1} = 60$$

3.3. Ước số chung lớn nhất của hai số

Ước số chung lớn nhất (USCLN) của 2 số được tính theo thuật toán Euclid
 $USCLN(a, b) = USCLN(b, (a \bmod b))$

```

function USCLN(a,b:longint):longint;
var tmp :longint;
begin
  while b>0 do begin
    a:=a mod b;
    tmp:=a; a:=b; b:=tmp;
  end;
  exit(a);
end;

```

3.4. Bội số chung nhỏ nhất của hai số

Bội số chung nhỏ nhất (BSCNN) của hai số được tính theo công thức:

$$BSCNN(a, b) = \frac{a \times b}{USCLN(a, b)} = \frac{a}{USCLN(a, b)} \times b$$

4. Lý thuyết tập hợp

4.1. Các phép toán trên tập hợp

1. Phần bù của A trong X , kí hiệu \bar{A} , là tập hợp các phần tử của X không thuộc A :

$$\bar{A} = \{x \in X : x \notin A\}$$

2. Hợp của A và B , kí hiệu $A \cup B$, là tập hợp các phần tử hoặc thuộc vào A hoặc thuộc vào B :

$$A \cup B = \{x: x \in A \text{ hoặc } x \in B\}$$

3. Giao của A và B , kí hiệu $A \cap B$, là tập hợp các phần tử đồng thời thuộc cả A và B

$$A \cap B = \{x: x \in A \text{ và } x \in B\}$$

4. Hiệu của A và B , kí hiệu là $A \setminus B$, là tập hợp các phần tử thuộc A nhưng không thuộc B .

$$A \setminus B = \{x: x \in A \text{ và } x \notin B\}$$

4.2. Các tính chất của phép toán trên tập hợp

1. Kết hợp

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

2. Giao hoán

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

3. Phân bố

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

4. Đôi ngẫu

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

$$\overline{A \cap B} = \bar{A} \cup \bar{B}$$

4.3. Tích Đè-các của các tập hợp

Tích Đè-các ghép hai tập hợp:

$$A \times B = \{(a, b) | a \in A, b \in B\}$$

Tích Đè-các mở rộng ghép nhiều tập hợp:

$$A_1 \times A_2 \times \dots \times A_k = \{(a_1, a_2, \dots, a_k) | a_i \in A_i, i = 1, 2, \dots, k\}$$

4.4. Nguyên lí cộng

Nếu A và B là hai tập hợp rời nhau thì

$$|A \cup B| = |A| + |B|$$

Nguyên lí cộng mở rộng cho nhiều tập hợp đôi một rời nhau:

Nếu $\{A_1, A_2, \dots, A_k\}$ là một phân hoạch của tập X thì:

$$|X| = |A_1| + |A_2| + \dots + |A_k|$$

4.5. Nguyên bù trừ

Nếu A và B không rời nhau thì

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Nguyên lí mở rộng cho nhiều tập hợp:

Giả sử A_1, A_2, \dots, A_m là các tập hữu hạn:

$$|A_1 \cup A_2 \cup \dots \cup A_m| = N_1 - N_2 + \dots + (-1)^{m-1} N_m$$

trong đó N_k là tổng phần tử của tất cả các giao của k tập lấy từ m tập đã cho

4.6. Nguyên lí nhân

Nếu mỗi thành phần a_i của bộ có thứ tự k thành phần (a_1, a_2, \dots, a_k) có n_i khả năng lựa chọn ($i = 1, 2, \dots, k$), thì số bộ sẽ được tạo ra là tích số của các khả năng này $n_1 \times n_2 \times \dots \times n_k$

Một hệ quả trực tiếp của nguyên lí nhân:

$$|A_1 \times A_2 \times \dots \times A_k| = |A_1| \times |A_2| \times \dots \times |A_k|$$

4.7. Chính hợp lặp

Xét tập hữu hạn gồm n phần tử $A = \{a_1, a_2, \dots, a_n\}$

Một chính hợp lặp chập k của n phần tử là một bộ có thứ tự gồm k phần tử của A , các phần tử có thể lặp lại. Một chính hợp lặp chập k của n có thể xem như một phần tử của tích Đècac A^k . Theo nguyên lí nhân, số tất cả các chính hợp lặp chập k của n sẽ là n^k .

$$\bar{A}_n^k = n^k$$

4.8. Chính hợp không lặp

Một chính hợp không lặp chập k của n phần tử ($k \leq n$) là một bộ có thứ tự gồm k thành phần lấy từ n phần tử của tập đã cho. Các thành phần không được lặp lại.

Để xây dựng một chính hợp không lặp, ta xây dựng dần từng thành phần đầu tiên. Thành phần này có n khả năng lựa chọn. Mỗi thành phần tiếp theo, số khả năng

lựa chọn giảm đi 1 so với thành phần đứng trước, do đó, theo nguyên lí nhân, số chỉnh hợp không lặp chập k của n sẽ là $n(n - 1) \dots (n - k + 1)$.

$$A_n^k = n(n - 1) \dots (n - k + 1) = \frac{n!}{(n - k)!}$$

4.9. Hoán vị

Một hoán vị của n phần tử là một cách xếp thứ tự các phần tử đó. Một hoán vị của n phần tử được xem như một trường hợp riêng của chỉnh hợp không lặp khi $k = n$. Do đó số hoán vị của n phần tử là $n!$

4.10. Tổ hợp

Một tổ hợp chập k của n phần tử ($k \leq n$) là một bộ không kể thứ tự gồm k thành phần khác nhau lấy từ n phần tử của tập đã cho.

$$C_n^k = \frac{n(n - 1) \dots (n - k + 1)}{k!} = \frac{n!}{k!(n - k)!}$$

Một số tính chất

- $C_n^k = C_n^{n-k}$
- $C_n^0 = C_n^n = 1$
- $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ (với $0 < k < n$)

5. Số Fibonacci

Số Fibonacci được xác định bởi công thức sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \end{cases}$$

Một số phần tử đầu tiên của dãy số Fibonacci:

n	0	1	2	3	4	5	6	...
$Fibonacci_n$	0	1	1	2	3	5	8	...

Số Fibonacci là đáp án của các bài toán:

a) Bài toán cổ về việc sinh sản của các cặp thỏ như sau:

- Các con thỏ không bao giờ chết;

- Hai tháng sau khi ra đời, mỗi cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực, một cái);
- Khi đã sinh con rồi thì cứ mỗi tháng tiếp theo chúng lại sinh được một cặp con mới.

Giả sử từ đầu tháng 1 có một cặp mới ra đời thì đến giữa tháng thứ n sẽ có bao nhiêu cặp.

Ví dụ, $n = 5$, ta thấy:

Giữa tháng thứ 1:

1 cặp (cặp ban đầu)

Giữa tháng thứ 2:

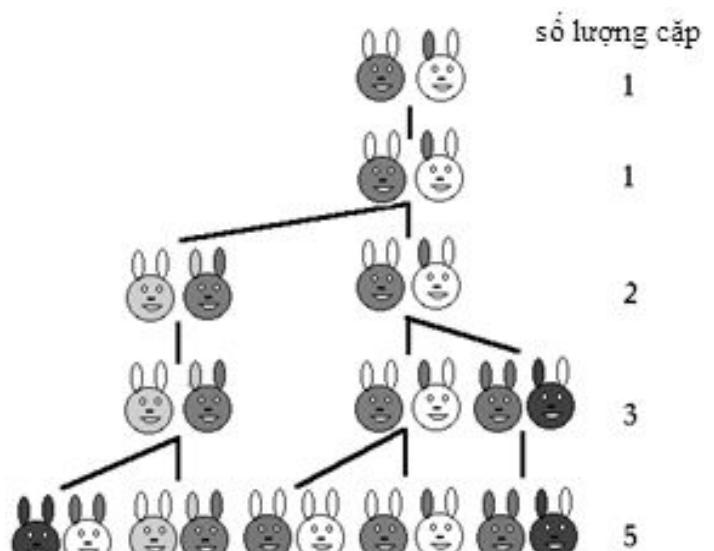
1 cặp cặp (ban đầu vẫn chưa đẻ)

Giữa tháng thứ 3:

2 cặp (cặp ban đầu đẻ ra thêm 1 cặp con)

Giữa tháng thứ 4:

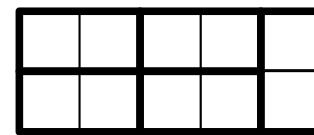
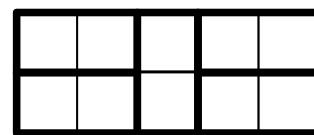
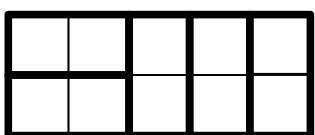
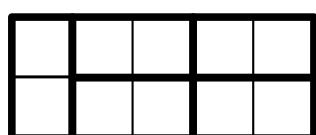
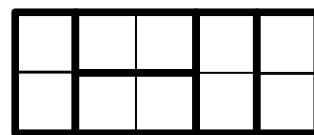
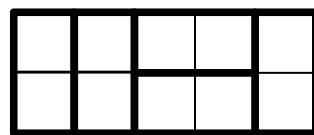
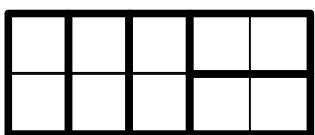
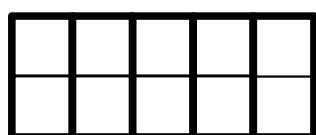
3 cặp (cặp ban đầu tiếp tục đẻ)



Giữa tháng thứ 5: 5 cặp.

b) Đếm số cách xếp $n - 1$ thanh DOMINO có kích thước 2×1 phủ kín bảng có kích thước $2 \times (n - 1)$.

Ví dụ: Có tất cả 8 cách khác nhau để xếp các thanh DOMINO có kích thước 2×1 phủ kín bảng 2×5 ($n = 6$, $Fibonacci_6 = 8$).



Hàm tính số Fibonacci thứ n bằng phương pháp lặp sử dụng công thức

$$F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \text{ và } F_0 = 0, F_1 = 1.$$

```
function Fibo(n : longint):longint;
var fi_1, fi_2, fi, i :longint;
begin
```

```

if n<=1 then exit(n);
fi_2:=0; fi_1:=1;
for i:=2 to n do begin
    fi:=fi_1 + fi_2;
    fi_2:=fi_1;
    fi_1:=fi;
end;
exit(fi);
end;

```

$$\text{Công thức tổng quát } F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

6. Số Catalan

Số Catalan được xác định bởi công thức sau:

$$Catalan_n = \frac{1}{n+1} C_{2n}^n = \frac{(2n)!}{(n+1)! n!} \text{ với } n \geq 0$$

Một số phần tử đầu tiên của dãy số Catalan là:

n	0	1	2	3	4	5	6	...
$Catalan_n$	1	1	2	5	14	42	132	...

Số Catalan là đáp án của các bài toán:

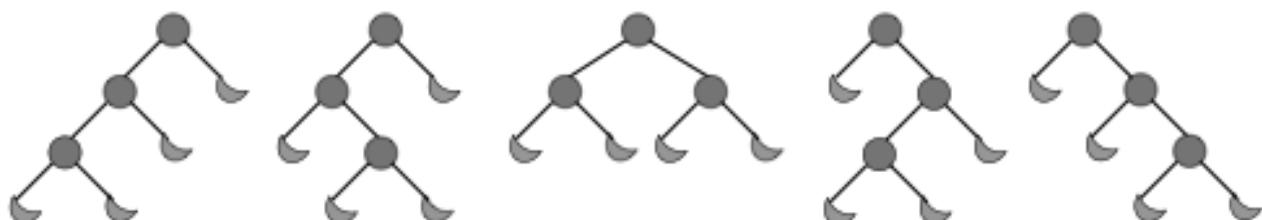
1) Có bao nhiêu cách khác nhau đặt n dấu ngoặc mở và n dấu ngoặc đóng đúng đắn?

Ví dụ: $n = 3$ ta có 5 cách sau:

$$((())), (()()), (()()) , ()(()) , ()()()$$

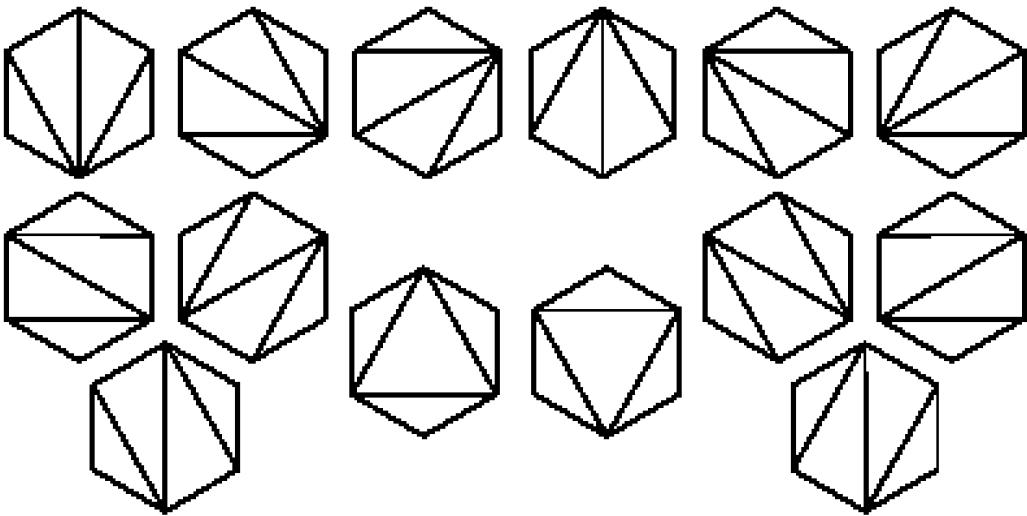
2) Có bao nhiêu cây nhị phân khác nhau có đúng $(n+1)$ lá?

Ví dụ: $n = 3$



3) Cho một đa giác lồi ($n + 2$) đỉnh, ta chia đa giác thành các tam giác bằng cách vẽ các đường chéo không cắt nhau trong đa giác. Hỏi có bao nhiêu cách chia như vậy?

Ví dụ: $n = 4$



7. Xử lý số nguyên lớn

Nhiều ngôn ngữ lập trình cung cấp kiểu dữ liệu nguyên khá lớn, chẳng hạn trong Free Pascal có kiểu số 64 bit (khoảng 19 chữ số). Tuy nhiên để thực hiện các phép tính với số nguyên ngoài phạm vi biểu diễn được cung cấp (có hàng trăm chữ số chẳng hạn), chúng ta cần tự thiết kế cách biểu diễn và các hàm thực hiện các phép toán cơ bản với các số nguyên lớn.

7.1. Biểu diễn số nguyên lớn

Thông thường người ta sử dụng các cách biểu diễn số nguyên lớn sau:

- **Xâu kí tự:** Đây là cách biểu diễn tự nhiên và đơn giản nhất, mỗi kí tự của xâu tương ứng với một chữ số của số nguyên lớn tính từ trái qua phải.
- **Mảng các số:** Sử dụng mảng lưu các chữ số (hoặc một nhóm chữ số), và một biến ghi nhận số chữ số để thuận tiện trong quá trình xử lí.
- **Danh sách liên kết các số:** Sử dụng danh sách liên kết các chữ số (hoặc một nhóm chữ số), cách làm này sẽ linh hoạt hơn trong việc sử dụng bộ nhớ.

Trong phần này, sử dụng cách biểu diễn thứ nhất, biểu diễn số nguyên lớn bằng xâu kí tự và chỉ xét các số nguyên lớn không âm.

```
Type bigNum = string;
```

7.2. Phép so sánh

Để so sánh hai số nguyên lớn a, b được biểu diễn bằng xâu kí tự, trước tiên ta thêm các chữ số 0 vào đầu số có số chữ số nhỏ hơn để hai số có số lượng chữ số bằng nhau. Sau đó sử dụng trực tiếp phép toán so sánh trên xâu kí tự.

Hàm cmp so sánh hai số nguyên lớn a, b . Giá trị hàm trả về

$$\begin{cases} 0 \text{ nếu } a = b \\ 1 \text{ nếu } a > b \\ -1 \text{ nếu } a < b \end{cases}$$

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

7.3. Phép cộng

Phép cộng hai số nguyên được thực hiện từ phải qua trái và phần nhớ được mang sang trái.

```
function      add(a,b : bigNum) : bigNum;
var   sum, carry, i, x, y : integer;
      c                  : bigNum;
begin
    carry:=0;c:='';
    while length(a)<length(b) do a:='0'+a;
    while length(b)<length(a) do b:='0'+b;
    for i:=length(a) downto 1 do
        begin
            x:= ord(a[i])-ord('0'); {ord('0')=48}
            y:= ord(b[i])-ord('0');

            sum:=x + y + carry;
            carry:=sum div 10;
```

```

        c:=chr(sum mod 10 +48)+c;
    end;
    if carry>0 then c:='1'+c;
    add:=c;
end;

```

7.4. Phép trừ

Thực hiện phép trừ ngược lại với việc nhớ ở phép cộng ta phải chú ý đến việc vay mượn từ hàng cao hơn. Trong hàm trừ dưới đây, chỉ xét trường hợp số lớn trừ số nhỏ hơn.

```

function    sub(a,b:bigNum):bigNum;
var         c           :bigNum;
s,borrow,i :integer;
begin
    borrow:=0;c:='';
    while length(a)<length(b) do a:='0'+a;
    while length(b)<length(a) do b:='0'+b;
    for i:=length(a) downto 1 do
        begin
            s:=ord(a[i])-ord(b[i])-borrow;
            if s<0 then
                begin
                    s:=s+10;
                    borrow:=1;
                end else borrow:=0;
            c:=chr(s +48)+c;
        end;
    while (length(c)>1) and (c[1]='0') do delete(c,1,1);
    sub:=c;
end;

```

7.5. Phép nhân một số lớn với một số nhỏ

Số nhỏ ở đây được hiểu là số nguyên do ngôn ngữ lập trình cung cấp (như: longint, integer,...). Hàm multiply1(a:bigNum;b:longint):bigNum, trả về là một số nguyên lớn (bigNum) là kết quả của phép nhân một số nguyên lớn a (bigNum) với một số b (longint).

```

function      multiply1 (a:bigNum;b:longint) :bigNum;
var i          :integer;
    carry,s    :longint;
    c,tmp      :bigNum;
begin
    c:=' ';
    carry:=0;
    for i:=length(a) downto 1 do
        begin
            s:=(ord(a[i])-48) * b + carry;
            carry:= s div 10;
            c:=chr(s mod 10 + 48)+c;
        end;
    if carry>0 then str(carry,tmp) else tmp:=' ';
    multiply1:=tmp+c;
end;

```

7.6. Phép nhân hai số nguyên lớn

```

function      multiply2 (a,b:bigNum) :bigNum;
var sum,tmp    :bigNum;
    m,i,j      :integer;
begin
    m:=-1;sum:=' ';
    for i:=length(a) downto 1 do
        begin
            m:=m+1;
            tmp:=multiply1(b,ord(a[i])-48);
{có thể thay câu lệnh tmp:=multiply1(b,ord(a[i])-48);
bằng cách cộng nhiều lần như sau:
            tmp:=";
            for j:=1 to ord(a[i])-48 do tmp:=add(tmp,b);
như vậy hàm nhân multiply2 chỉ gọi hàm cộng hai số nguyên lớn add}
                for j:=1 to m do tmp:=tmp+'0';
                sum:=add (tmp,sum);
            end;
            multiply2:=sum;
end;

```

7.7. Phép toán chia lấy thương nguyên (div) của một số lớn với một số nhỏ

```
function bigDiv1(a:bigNum;b:longint):bigNum;
var s,i,hold:longint;
c:bigNum;
begin
    hold:=0;s:=0; c:=' ';
    for i:=1 to length(a) do
        begin
            hold:=hold*10 + ord(a[i])-48;
            s:=hold div b;
            hold:=hold mod b;
            c:=c+chr(s+48);
        end;
    while (length(c)>1) and (c[1]='0') do
delete(c,1,1);
    bigDiv1:=c;
end;
```

7.8. Phép toán chia lấy dư (mod) của một số lớn với một số nhỏ

```
function bigMod1(a:bigNum;b:longint):longint;
var i,hold:longint;
begin
    hold:=0;
    for i:=1 to length(a) do
        hold:=(ord(a[i])-48+hold*10) mod b;
    bigMod1:=hold;
end;
```

Chú ý: Ta có các công thức sau:

- 1) $(A + B) \bmod N = ((A \bmod N) + (B \bmod N)) \bmod N$
- 2) $(A \times B) \bmod N = ((A \bmod N) \times (B \bmod N)) \bmod N$

7.9. Phép toán chia lấy thương nguyên (div) của hai số lớn

```
function bigDiv2(a,b:bigNum):bigNum;
var c,hold :bigNum;
```

```

kb      :array[0..10]of bigNum;
i,k     :longint;
begin
  kb[0]:='0';
  for i:=1 to 10 do
    kb[i]:=add(kb[i-1],b);
  hold:='';
  c:='';
  for i:=1 to length(a) do
    begin
      hold:=hold+a[i];
      k:=1;
      while cmp(hold, kb[k])<>-1 do
        inc(k);
      c:=c+chr(k-1+48);
      hold:=sub(hold, kb[k-1]);
    end;
  while (length(c)>1) and (c[1]='0') do delete(c,1,1);
  bigDiv2:=c;
end;

```

7.10. Phép toán chia lấy dư (mod) của hai số lớn

```

function bigMod2(a,b:bigNum):bigNum;
var  hold   :bigNum;
      kb      :array[0..10]of bigNum;
      i,k     :longint;
begin
  kb[0]:='0';
  for i:=1 to 10 do
    kb[i]:=add(kb[i-1],b);
  hold:='';
  for i:=1 to length(a) do
    begin
      hold:=hold+a[i];
      k:=1;
      while cmp(hold, kb[k])<>-1 do

```

```

        inc(k);
        hold:=sub(hold,kb[k-1]);
    end;
    bigMod2:=hold;
end;

```

7.11. Ví dụ tính số Fibonacci thứ n ($n \leq 500$)

Số Fibonacci được xác định bởi công thức sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \text{ với } n \geq 2 \end{cases}$$

Trước tiên ta xây dựng chương trình tính số Fibonacci bằng kiểu dữ liệu Extended như sau:

```

function Fibo(n : longint):extended;
var i :longint;
fi_1, fi_2, fi : extended;
begin
    if n<=1 then exit(n);
    fi_2:=0; fi_1:=1;
    for i:=2 to n do begin
        fi:=fi_1 + fi_2;
        fi_2:=fi_1;
        fi_1:=fi;
    end;
    exit(fi);
end;
var n : longint;
BEGIN
write('Nhập N:'); readln(n);
writeln(Fibo(n));
END.

```

Chạy chương trình với $n = 500$ ta nhận được kết quả:

1.3942322456169788E+0104, như vậy số Fibonacci thứ 500 có 105 chữ số (có thể sử dụng cách biểu diễn bằng xâu kí tự), ta xây dựng chương trình tính số Fibonacci lớn bằng cách sau:

- Thay kiểu extended bằng kiểu bigNum.
- Thay các phép toán bằng các hàm tính toán số lớn, xây dựng các hàm tính toán số lớn cần thiết.

```

type bigNum = string;
function      add(a,b : bigNum) : bigNum;
var   sum, carry, i : integer;
      c           : bigNum;
begin
  carry:=0;c:='';
  while length(a)<length(b) do a:='0'+a;
  while length(b)<length(a) do b:='0'+b;
  for i:=length(a) downto 1 do
    begin
      sum:=ord(a[i])-48+ord(b[i])-48+carry;
      carry:=sum div 10;
      c:=chr(sum mod 10 +48)+c;
    end;
  if carry>0 then c:='1'+c;
  add:=c;
end;
function Fibo(n : longint):bigNum;
var i :longint;
fi_1, fi_2, fi : bigNum;
begin
  if n<=1 then exit(char(n+48));
  fi_2:='0'; fi_1:='1';
  for i:=2 to n do begin
    fi:=add(fi_1,fi_2); {fi:=fi_1 + fi_2;}
    fi_2:=fi_1;
    fi_1:=fi;
  end;
  exit(fi);
end;
var n : longint;
BEGIN

```

```

write('Nhập N:'); readln(n);
writeln(Fibo(n));
END.

```

7.12. Ví dụ tính số *Catalan_n* ($n \leq 100$)

Số Catalan được xác định bởi công thức sau:

$$Catalan_n = \frac{1}{n+1} C_{2n}^n = \frac{(2n)!}{(n+1)! n!} \text{ với } n \geq 0$$

Rút gọn tử và mẫu cho $(n+1)!$ ta có:

$$Catalan_n = \frac{(n+2) \times (n+3) \times \dots \times (2n)}{1 \times 2 \times \dots \times n}$$

Ta sẽ tính tử số bằng cách sử dụng hàm nhân số lớn với số nhỏ, sau đó sử dụng hàm chia số lớn cho số nhỏ để được kết quả cần tính.

```

type bigNum      =string;
function         multiply1(a:bigNum;b:longint):bigNum;
var i            :integer;
  carry,s       :longint;
  c,tmp         :bigNum;
begin
  c:='';
  carry:=0;
  for i:=length(a) downto 1 do
    begin
      s:=(ord(a[i])-48) * b + carry;
      carry:= s div 10;
      c:=chr(s mod 10 + 48)+c;
    end;
  if carry>0 then str(carry,tmp) else tmp:='';
  multiply1:=tmp+c;
end;
function bigDiv1(a:bigNum;b:longint):bigNum;
var s,i,hold:longint;
c:bigNum;
begin
  hold:=0;s:=0; c:='';

```

```

        for i:=1 to length(a) do
            begin
                hold:=hold*10 + ord(a[i])-48;
                s:=hold div b;
                hold:=hold mod b;
                c:=c+chr(s+48);
            end;
        while      (length(c)>1)      and (c[1]='0')      do
delete(c,1,1);
        bigDiv1:=c;
    end;
var n,k      :longint;
    s          :bigNum;
BEGIN
    write('Nhập N:'); readln(n);
    s:='1';
    for k:=(n+2) to 2*n do s:=multiply1(s, k); {tính tử số}
    for k:=1 to n do s:=bigDiv(s, k); {chia cho mẫu số}
    writeln(s);
END.

```

Tuy nhiên, ta có thể rút gọn hoàn toàn mẫu số của phân số trên và chỉ cần sử dụng hàm nhân số lớn với số nhỏ, chương trình sẽ chạy nhanh hơn.

Bài tập

- 2.1.** Cho s là một xâu chỉ gồm 2 ký tự '0' hoặc '1' mô tả một số nguyên không âm ở hệ cơ số 2, hãy chuyển số đó sang hệ cơ số 16 (độ dài xâu s không vượt quá 200).

Ví dụ: $10101100_2 = AC_{16}$

$10101011100000100100011_2 = ABC123_{16}$

- 2.2.** Cho số nguyên dương N ($N \leq 10^9$)

- Phân tích N thành thừa số nguyên tố
- Đếm số ước của N
- Tính tổng các ước của N

- 2.3.** Đưa ra những số $\leq 10^6$ mà cách kiểm tra tính nguyên tố của Fermat bị sai.

- 2.4.** Sử dụng sàng số nguyên tố liệt kê các số nguyên tố trong đoạn $[L, R]$
- 2.5.** Người ta định nghĩa một số nguyên dương N được gọi là số đẹp nếu N thoả mãn *một trong hai* điều kiện sau:
- N bằng 9
 - Gọi $f(N)$ là tổng các chữ số của N thì $f(N)$ cũng là số đẹp
- Cho số nguyên dương N ($N \leq 10^{100}$), hãy kiểm tra xem N có phải là số đẹp không?
- 2.6.** Dùng cách biểu diễn số nguyên lớn bằng xâu và thêm thông tin dấu ($\text{sign}=1$ nếu số lớn là số không âm, $\text{sign}=-1$ nếu số lớn là số âm) để xử lí số nguyên lớn có dấu như sau:
- ```
type bigNum = record
 sign : longint;
 num : string;
 end;
```
- Hãy xây dựng các hàm xử lí số nguyên lớn có dấu.
- 2.7.** Dùng cách biểu diễn số nguyên lớn bằng mảng (mỗi phần tử của mảng là một nhóm các chữ số).
- a) Hãy xây dựng các hàm xử lí số nguyên lớn.
  - b) Sử dụng hàm nhân số nguyên lớn với số nhỏ tính  $N!$  với  $N \leq 2000$ .
- 2.8.** Tìm  $K$  chữ số cuối cùng của  $M^N$  ( $0 < K \leq 9$ ,  $0 \leq M, N \leq 10^6$ )
- Ví dụ:  $K=2$ ,  $M=2$ ,  $N=10$ , ta có  $2^{10}=1024$ , như vậy 2 chữ số cuối cùng của  $2^{10}$  là 24
- 2.9.** Cho  $N$  ( $N \leq 10$ ) nguyên dương  $a_1, a_2, \dots, a_N$  ( $a_i < 10^9$ ). Tìm ước số chung lớn nhất, bội số chung nhỏ nhất của  $N$  số trên (chú ý: BSCNN có thể rất lớn).
- 2.10.** Cho hai số nguyên không âm  $A, B$  ( $0 \leq A \leq B \leq 10^{200}$ ), tính số lượng số Fibonacci trong đoạn  $[A, B]$ .
- 2.11.** Cho số nguyên dương  $N$  ( $N \leq 10^{100}$ ), hãy tách  $N$  thành tổng các số Fibonacci đôi một khác nhau.
- Ví dụ:  $N=16=1+5+13$
- 2.12.** Cho  $N$  là một số nguyên dương không vượt quá  $10^9$ . Hãy tìm số chữ số 0 tận cùng của  $N!$

- 2.13.** Cho  $s$  là một xâu mô tả số nguyên không âm ở hệ cơ số  $a$ , hãy chuyển số đó sang hệ cơ số  $b$  ( $1 < a, b \leq 16$ , độ dài xâu  $s$  không vượt quá 50).
- 2.14.** Xây dựng hàm kiểm tra số nguyên dương  $N$  có phải là số chính phương không? ( $N < 10^{100}$ )
- 2.15.** Tính  $C_n^k$  ( $0 < k \leq n \leq 2000$ )
- 2.16.** Tính Catalan $_n$  ( $n \leq 2000$ )
- 2.17.** Hãy đếm số cách đặt  $k$  quân xe lênh bàn cờ  $n \times n$  sao cho không có quân nào ăn được nhau. ( $1 \leq k \leq n \leq 100$ )
- 2.18.** Giả thiết  $N$  là số nguyên dương. Số nguyên  $M$  là tổng của  $N$  với các chữ số của nó.  $N$  được gọi là nguồn của  $M$ . Ví dụ,  $N = 245$ , khi đó  $M = 245 + 2 + 4 + 5 = 256$ . Như vậy, nguồn của  $256$  là  $245$ . Có những số không có nguồn và có số lại có nhiều nguồn. Ví dụ, số  $216$  có 2 nguồn là  $198$  và  $207$ .
- Cho số nguyên  $M$  ( $M$  có không quá 100 chữ số) hãy tìm nguồn nhỏ nhất của nó. Nếu  $M$  không có nguồn thì đưa ra số 0.
- 2.19.** Tính số ước và tổng các ước của  $N!$  ( $N \leq 100$ )
- 2.20.** Cho một chiếc cân hai đĩa và các quả cân có khối lượng  $3^0, 3^1, 3^2, \dots$
- Hãy chọn các quả cân để có thể cân được vật có khối lượng  $N$  ( $N \leq 10^{100}$ )
- Ví dụ: cần cân vật có khối lượng  $N=11$  ta cần sử dụng các quả cân sau:
- Cân bên trái: quả cân  $3^1$  và  $3^2$
  - Cân bên phải: quả cân  $3^0$  và vật  $N=11$
- 2.21.** Đếm số lượng dãy nhị phân khác nhau độ dài  $n$  mà không có 2 số 1 nào đứng cạnh nhau?
- Ví dụ:  $n = 3$ , ta có 5 dãy 000, 001, 010, 100, 101
- 2.22.** Cho xâu  $s$  chỉ gồm kí tự từ 'a' đến 'z' (độ dài xâu  $s$  không vượt quá 100), hãy đếm số hoán vị khác nhau của xâu đó.
- Ví dụ:  $s='aba'$ , ta có 3 hoán vị 'aab', 'aba', 'baa'
- 2.23.** John Smith quyết định đánh số trang cho quyển sách của anh ta từ 1 đến  $N$ . Hãy tính toán số lượng chữ số 0 cần dùng, số lượng chữ số 1 cần dùng, ..., số lượng chữ số 9 cần dùng.
- Dữ liệu vào trong file: “*digits.inp*” gồm 1 dòng duy nhất chứa một số  $N$  ( $N \leq 10^{100}$ ).

Kết quả ra file “*digits.out*” có dạng gồm 10 dòng, dòng thứ nhất là số lượng chữ số 0 cần dùng, dòng thứ hai là số lượng chữ số 1 cần dùng,..., dòng thứ 10 là số lượng chữ số 9 cần dùng.

### 2.24. *TAM GIÁC SỐ* (đề thi học sinh giỏi Hà Tây 2006)

Hình bên mô tả một tam giác số có số hàng  $N=5$ . Đi từ đỉnh (số 7) đến đáy tam giác bằng một đường gấp khúc, mỗi bước chỉ được đi từ số ở hàng trên xuống một trong hai số đứng kề bên phải hay bên trái ở hàng dưới, và tính tích các số trên đường đi lại ta được một tích.

|   |   |    |   |
|---|---|----|---|
|   |   |    | 7 |
|   |   | 3  | 8 |
|   | 8 | 1  | 0 |
|   | 2 | 7  | 4 |
| 4 | 5 | -2 | 6 |
|   |   |    | 5 |

Ví dụ: đường đi 7 8 1 4 6 có tích là  $S=1344$ , đường đi 7 3 1 7 5 có tích là  $S=735$ .

*Yêu cầu:* Cho tam giác số, tìm tích của đường đi có tích lớn nhất

*Dữ liệu:* Vào từ file văn bản TGS.INP:

- Dòng đầu tiên chứa số nguyên  $n$ , ( $0 < n < 101$ )
- $N$  dòng tiếp theo, từ dòng thứ 2 đến dòng thứ  $N+1$ : dòng thứ  $i$  có  $(i-1)$  số cách nhau bởi dấu cách (các số có giá trị tuyệt đối không vượt quá 100)

*Kết quả:* Đưa ra file văn bản TGS.OUT một số nguyên – là tích lớn nhất tìm được

| TGS.INP    | TGS.OUT     |
|------------|-------------|
| 5          | <b>5880</b> |
| 7          |             |
| 3 8        |             |
| 8 1 0      |             |
| 2 7 4 4    |             |
| 4 5 -2 6 5 |             |

### 2.25. HÁI NẤM (bài thi Olympic Sinh viên 2009, khối chuyên)

Một cháu gái hàng ngày được mẹ giao nhiệm vụ đến thăm bà nội. Từ nhà mình đến nhà bà nội cô bé phải đi qua một khu rừng có rất nhiều loại nấm. Trong số các loại nấm, có ba loại có thể ăn được. Cô bé đánh số ba loại nấm ăn được lần lượt là 1, 2 và 3. Là một người cháu hiếu thảo cho nên cô bé

quyết định mỗi lần đến thăm bà, cô sẽ hái ít nhất hai loại nấm ăn được để nấu súp cho bà. Khu rừng mà cô bé đi qua được chia thành lưới ô vuông gồm  $m$  hàng và  $n$  cột. Các hàng của lưới được đánh số từ trên xuống dưới bắt đầu từ 1, còn các cột – đánh số từ trái sang phải, bắt đầu từ 1. Ô nằm giao của hàng  $i$  và cột  $j$  có tọa độ  $(i, j)$ . Trên mỗi ô vuông, trừ ô  $(1,1)$  và ô  $(m, n)$  các ô còn lại hoặc có nấm độc và cô bé không dám đi vào (đánh dấu là -1), hoặc là có đúng một loại nấm có thể ăn được (đánh dấu bằng số hiệu của loại nấm đó). Khi cô bé đi vào một ô vuông có nấm ăn được thì cô bé sẽ hái loại nấm mọc trên ô đó. Xuất phát từ ô  $(1,1)$ , để đến được nhà bà nội ở ô  $(m, n)$  một cách nhanh nhất cô bé luôn đi theo hướng sang phải hoặc xuống dưới.

Việc đi thăm bà và hái nấm trong rừng sâu gấp nguy hiểm bởi có một con chó sói luôn theo dõi và muốn ăn thịt cô bé. Để phòng tránh chó sói theo dõi và ăn thịt, cô bé quyết định mỗi ngày sẽ đi theo một con đường khác nhau (hai con đường khác nhau nếu chúng khác nhau ở ít nhất một ô).

*Yêu cầu:* Cho bảng  $m \times n$  ô vuông mô tả trạng thái khu rừng. Hãy tính số con đường khác nhau để cô bé đến thăm bà nội theo cách chọn đường đi đã nêu ở trên.

*Dữ liệu:* Vào từ file văn bản MUSHROOM.INP:

- Dòng đầu chứa 2 số  $m, n$  ( $1 < m, n < 101$ ),
- $m$  dòng tiếp theo, mỗi dòng chứa  $n$  số nguyên cho biết thông tin về các ô của khu rừng. (riêng giá trị ở hai ô  $(1,1)$  và ô  $(m, n)$  luôn luôn bằng 0 các ô còn lại có giá trị bằng -1, hoặc 1, hoặc 2, hoặc 3).

Hai số liên tiếp trên một dòng cách nhau một dấu cách.

*Kết quả:* Đưa ra file văn bản MUSHROOM.OUT chứa một dòng ghi một số nguyên là kết quả bài toán.

*Ví dụ:*

| MUSHROOM.INP                              | MUSHROOM.OUT |
|-------------------------------------------|--------------|
| <pre> 3 4 0 3 -1 2 3 3 3 3 3 1 3 0 </pre> | 3            |

## 2.26. HỆ THỐNG ĐÈN MÀU (Tin học trẻ bảng B năm 2009)

Để trang trí cho lễ kỉ niệm 15 năm hội thi Tin học trẻ toàn quốc, ban tổ chức đã dùng một hệ thống đèn màu gồm  $n$  đèn đánh số từ 1 đến  $n$ . Mỗi đèn có

khả năng sáng màu xanh hoặc màu đỏ. Các đèn được điều khiển theo quy tắc sau:

- Ban đầu tất cả các đèn đều sáng màu xanh.
- Sau khi kết thúc chương trình thứ nhất của lễ kỉ niệm, tất cả các đèn có số thứ tự chia hết cho 2 sẽ đổi màu... Sau khi kết thúc chương trình thứ  $i$ , tất cả các đèn có số thứ tự chia hết cho  $i + 1$  sẽ đổi màu (đèn xanh đổi thành màu đỏ còn đèn đỏ đổi thành màu xanh)

Minh, một thí sinh dự lễ kỉ niệm đã phát hiện được quy luật điều khiển đèn và rất thích thú với hệ thống đèn trang trí này. Vào lúc chương trình thứ  $k$  của buổi lễ vừa kết thúc, Minh đã nhầm tính được tại thời điểm đó có bao nhiêu đèn xanh và bao nhiêu đèn đỏ. Tuy nhiên vì không có máy tính nên Minh không chắc chắn kết quả của mình là đúng. Cho biết hai số  $n$  và  $k$  ( $n, k \leq 10^6$ ), em hãy tính lại giúp Minh xem khi chương trình thứ  $k$  của buổi lễ vừa kết thúc, có bao nhiêu đèn màu đỏ.

Ví dụ với  $n = 10; k = 3$ .

| Thời điểm          | Trạng thái các đèn                 |
|--------------------|------------------------------------|
| Bắt đầu            | Xanh: 1 2 3 4 5 6 7 8 9 10<br>Đỏ : |
| Sau chương trình 1 | Xanh: 1 3 5 7 9<br>Đỏ : 2 4 6 8 10 |
| Sau chương trình 2 | Xanh: 1 5 6 7<br>Đỏ : 2 3 4 8 9 10 |
| Sau chương trình 3 | Xanh: 1 4 5 6 7 8<br>Đỏ : 2 3 9 10 |

Vậy có 4 đèn đỏ sau chương trình thứ 3.