

LỜI NÓI ĐẦU

Nhóm chúng tôi thích giúp các bạn yêu thích môn Tin học, đặc biệt là những bạn đang chuẩn bị cho các kỳ thi lập trình có thêm những kiến thức chuyên sâu hơn về thuật toán và kiến lập trình, chúng tôi đã biên soạn cuốn tài liệu

“Mô tả thuật toán và cấu trúc dữ liệu nâng cao”.

Tài liệu là tập hợp những kiến thức, kinh nghiệm mà chúng tôi và 7 bạn thuật toán và kiến lập trình quan trọng nhất của chúng tôi ghi lại nhằm giúp bài toán trong các kỳ thi Học sinh giỏi Quốc gia, Olympiad Tin học Quốc tế, ACM, các kỳ thi lập trình trên mạng...v.v..

Trong quá trình biên soạn, chúng tôi có tham khảo, sử dụng giáo trình, bài viết của thầy Lê Minh Hoàng (thầy Sphm Hà Nội) và anh Lê Khắc Minh Tu (Huy chương IOI - Thái Lan 2011). Phần lớn các bài toán trong tài liệu này có lấy từ các kỳ thi có uy tín, các trang ghi bài trắc nghiệm. Các bài tập đều kèm bộ test hoặc liên kết kiểm tra kết quả.

Dù đã dành nhiều thời gian và tâm huyết song không thể có thể sót. Chúng tôi hy vọng những đóng góp của các bạn có thể hoàn thiện cuốn tài liệu hơn.

Nhóm tác giả :

Nguyễn Văn Hoàng Văn, Hoàng Minh Quang, Nguyễn Văn Văn Cường, Thái Minh Phúc, Nguyễn Xuân Cường, Hoàng Minh Châu, Lê Khắc Thành - Học sinh chuyên Tin khóa 39, THPT chuyên Phan Bội Châu



K39

PBC

Vinh - 4/2013

M C L C

| | |
|-------------------------------|-----------|
| L i nói u | 1 |
| ng d ng c a phép nhân ma tr n | 3 |
| QH theo mô hình quy có nh | 14 |
| C u trúc Dequeue & Left-Right | 34 |
| Bài toán LCA | 55 |
| Lu ng c c i trên m ng | 70 |
| Hash | 82 |
| Suffix Tree | 93 |

Chương I**NG D NG C A PHÉP NHÂN MA TR N****I/ Phân tích v n :****Bài toán1:Lát g ch**

(Ngu n <http://vn.spoj.com/problems/LATGACH4/>)

Cho m t hình ch nh t kích th c $2 \times N$ ($1 \leq N < 10^9$). Hãy m s cách lát các viên g ch nh kích th c 1×2 và 2×1 vào hình trên sao cho không có ph n nào c a các viên g ch nh th a ra ngoài, c ng không có vùng di n tích nào c a hình ch nh t khô ng c lát.

Input

G m nhi u test, dòng u ghi s l ng test T ($T \leq 100$). T dòng sau m i dòng ghi m t s N .

Output

Ghi ra T dòng là s cách lát t ng ng l y ph n d cho 111539786.

Example**Input:**

3
1
2
3

Output:

1
2
3

H ng d n: Công th c quy ho ch ng bài này khá d ngh : G i $F[N]$ là s cách lát các viên g ch nh kích th c 1×2 và 2×1 vào hình ch nh t kích th c $2 \times N$, ta có

$F[1]=1$

$F[2]=2$

$F[i]=F[i-1]+F[i-2]$ v i $i \geq 3$. K t qu bài toán là $F[N]$.

Bằng cách tính tuần tự $F[1], F[2], F[3], \dots, F[N]$ ta có 1 thuật toán với độ phức tạp là $O(N)$, nhưng bài 1 cho $N < 10^9$, nên với độ phức tạp này ta không thể test bài này được. Áp dụng kỹ thuật nhân ma trận ta có thể Accept với độ phức tạp $\log(N)$.

Trước tiên ta tìm hiểu khái niệm ma trận. Cho định nghĩa, 1 ma trận n hàng m cột 2 chiều gọi là các phần tử $n \times m$, ký hiệu $A(M, N)$ là ma trận A gồm M hàng và N cột. Phép nhân 2 ma trận $A(M, N)$ và $B(P, Q)$ có thể nhân được với nhau khi $N=P$, khi đó ma trận tích của A và B là $C(M, Q)$ với

$$C[i, j] = A[i, 1] * B[1, j] + A[i, 2] * B[2, j] + \dots + A[i, N] * B[j, N].$$

Trong đó, ta có thể tính toán $C[i, j]$ là tổng các tích từng số hạng từng số hạng trong hàng i của A với cột j của B .

Giả sử A, B là ma trận vuông $N \times N$, ta có tính chất hoán vị $A * B = B * A$, tính chất với phép nhân nhiều ma trận vuông cùng kích thước với nhau cũng có tính hoán vị như vậy, như là $A * B * C * D = (A * C) * (B * D) = (B * A * D) * C = \dots$

Quay lại bài toán Lát gạch.

(Các ma trận dùng để xây dựng có kích thước 2×2).

Ban đầu ta cho $A[1, 1] = F[1]$, $A[1, 2] = F[2]$. Ý tưởng sử dụng nhân ma trận là cho A nhân với 1 ma trận B sao cho khi nhân xong, $A[1, 1] = F[2]$, $A[1, 2] = F[3]$, nhân với B tiếp thì $A[1, 1] = F[3]$, $A[1, 2] = F[4]$, cứ tiếp tục như vậy, sau $N-2$ lần nhân ta có $A[1, 1] = F[N-1]$ và $A[1, 2] = F[N]$ là kết quả bài toán. Bằng cách sử dụng mối quan hệ $F[i] = F[i-1] + F[i-2]$ ta dễ dàng xác định được B

| | |
|---|---|
| 0 | 1 |
| 1 | 1 |

Các bước kiểm tra tính đúng đắn của mảng B .

Áp dụng tính chất hoán vị đã nói trên của phép nhân ma trận, ta có thể tính $B * B * B * \dots * B$ (N ma trận B) $= B^N$ trong $\log(N)$.

Sau khi thực hiện $C := A * B^{N-2}$ ta có $C[1, 2] = F[N]$ là kết quả cần tìm.

Sau đây là code mẫu cho các bước tham khảo:

```
constf    inp='';
          fout='';
          base=111539786;
type matrix=array[1..2,1..2] of longint;
var fi,fo:text;
    a,b,c:matrix;
test,n:longint;
```

```
procedure openfile;
begin
    assign(fi,finp);
    reset(fi);
    assign(fo,fout);
    rewrite(fo);
end;
procedure closefile;
begin
    close(fi);
    close(fo);
end;
procedure enter;
begin
    readln(fi,n);
end;

function nhan(a,b:matrix):matrix;
var i,j,k,x:longint;
    c:matrix;
begin
    for i:=1 to 2 do
        for j:=1 to 2 do
            begin
                x:=0;
                for k:=1 to 2 do x:=(x+int64(a[i,k])*b[k,j] mod
base) mod base;
                c[i,j]:=x
            end;
        exit(c);
    end;

function mu(a:matrix; n:longint):matrix;
var b:matrix;
begin
    if n=1 then exit(a);
    b:=mu(a,n div 2);
    b:=nhan(b,b);
    if odd(n) then b:=nhan(b,a);
    exit(b);
end;

procedure process;
var kq:longint;
```

```

begin
    if n<=2 then
    begin
        if n=1 then kq:=1 else kq:=2;
        writeln(fo,kq);
        exit;
    end;
    c:=nhân(a,mu(b,n-2));
    writeln(fo,c[1,2]);
end;

procedure init;
begin
    a[1,1]:=1; a[1,2]:=2;
    b[1,1]:=0; b[1,2]:=1;
    b[2,1]:=1; b[2,2]:=1;
end;
BEGIN
    openfile;
    init;
    readln(fi,test);
    while test>0 do
    begin
        enter;
        process;
        dec(test);
    end;
    closefile;
END.

```

Bài toán 2: Xếp hình.

(Nguồn: <http://vn.spoj.com/problems/FBRICK/>)

Nguyên thích trò chơi xếp tháp. Tòa tháp của Nguyên bao gồm những khối lập phương có đáy hình vuông và chiều cao bằng 1. Nguyên sắp xếp các khối lập phương lên nhau để tạo thành một tòa tháp cao.

Mỗi lần trong lớp học toán, Nguyên có cô giáo dạy về cách tính thể tích các hình khối không gian. Nguyên thích thú với kiến thức mới học và cậu ta muốn tính thể tích tòa tháp của mình.

Tháp của Nguyên bao gồm N khối lập phương có chiều cao 1 và có đáy hình vuông và dài cạnh đáy từ trên xuống dưới theo thứ tự là A_1, A_2, \dots, A_N . Dãy A có thể như sau:

1. $A_1 = 1$.
2. A_2 s là m t s đ ng tùy ý mà Nguyên ch n trong m i l n ch i tránh nhàm chán.
3. $A_i (i > 2)$ b ng $2 \times A_2 \times A_{i-1} - A_{i-2}$.

Nguyên bi t rõ th tích hình m t hình l ng tr s b ng chi u cao nhân v i đi n tích ấy nh ng vì ng i tính toán, Nguyên mu n nh b n vì t m t ch ng trình giúp c u ta. K t qu có th r t l n vì v y b n ch c n ghi ra theo modulo M v i M là m t s nguyên đ ng cho tr c.

Input

- Dòng 1: Ghi s nguyên đ ng K 10 là s b d li u.
- K dòng ti p: M i dòng ghi 3 s nguyên A_2, N, M t ng ng v i m t b d li u. ($1 \leq A_2, M \leq 10^9, 2 \leq N \leq 10^9$)

Output

- V i m i b test ghi ra m t s duy nh t là k t qu t ng ng trên m t dòng.

Example

Input:

2

1 10 1000

2 3 100

Output:

10

54

H ng d n: Ta c n tính $A[1]^2 + A[1]^2 + \dots + A[N]^2$.

Ta có th t o 1 ma tr n B ban u nh sau:

$$B[1,1] = A[1]$$

$$B[1,2] = A[2]$$

$$B[1,3] = A[1]^2$$

$$B[1,4] = A[2]^2$$

$$B[1,5] = A[1] * A[2]$$

$$B[1,6] = A[1]^2 + A[2]^2$$

Ta c n tìm ma tr n C sao cho m i l n gán $B := B * C$ ta c ma tr n B đ ng

$$B[1,1] = A[i-1]$$

$$B[1,2] = A[i]$$

$$B[1,3]=A[i-1]^2$$

$$B[1,4]=A[i]^2$$

$$B[1,5]=A[i]*A[i-1]$$

$$B[1,6]=A[1]^2+A[2]^2+...+A[i]^2$$

Tại thời điểm này, ta đã có các mối quan hệ:

$$A[i]=2A[2]*A[i-1]-A[i-2];$$

$$A[i]^2=4A[2]^2*A[i-1]^2+A[i-1]^2-4A[2]*A[i-1]*A[i-2];$$

$$A[i]*A[i-1]=2A[2]*A[i-1]^2-A[i-1]*A[i-2].$$

Từ đó ta đã dàng xác định các C. Bài toán đã được giải quyết.

Cách giải trên sử dụng ma trận $6*6$, còn ít nhất 1 cách khác sử dụng ma trận $4*4$, các bạn thử nghĩ xem.

II/ Bài tập luyện:

Bài 1: Xâu Fibonacci.

(Nguồn: <http://vn.spoj.com/problems/LQDFIBO2/>)

Cho 2 xâu khác rỗng S1, S2 có độ dài không lớn hơn 100. Xét các dãy F[1], F[2], ..., F[N] trong đó:

$$F[1]=S1; F[2]=S2;$$

$$F[i]=F[i-1]+F[i-2] \text{ với } i>2.$$

Cho xâu S không quá 100 ký tự và N ($N \leq 10^9$). Xác định số lần xâu S xuất hiện trong xâu F[N] sau khi đã modulo với 15111992.

Input

Dòng 1 chứa số nguyên dương N.

Dòng 2 chứa xâu S1. Dòng 3 chứa xâu S2.

Dòng cuối cùng chứa xâu S.

Output

Đưa ra một dòng chứa kết quả.

Example

Input:

8

A

B
AB

Output:

8

Bài 2: Thành c An D ng V ng.

(Ngu n: Vòng 2 n m 2009).

Gi s X và Y là các xâu. Phép “c ng i x ng” hai xâu X và Y , ký hi u là \oplus , c nh ngh a nh sau:

$$X \oplus Y = X_1 + Z + Y_1.$$

trong ó Z là xâu có dài l n nh t th a m n các i u ki n:

- $X = X_1 + Z$;
- $Y = Z + Y_1$.

Ví d :

$$\text{'ABCDEF'} \oplus \text{'DEFG'} = \text{'ABCDEFG'}, \text{ còn } \text{'ABC'} \oplus \text{'DE'} = \text{'ABCDE'}.$$

Cho tr c hai xâu F_1, F_2 , b ng phép c ng i x ng 2 xâu ng i ta t o ra các xâu $F_3, F_4, \dots, F_k, \dots$ theo công th c:

$$F_k = F_{k-1} \oplus F_{k-2}, k = 3, 4, 5, \dots$$

Ví d , v i $F_1 = \text{'ABCDEF'}$ và $F_2 = \text{'DEFG'}$. Ta có

$$F_3 = \text{'DEFGABCDEF'}$$

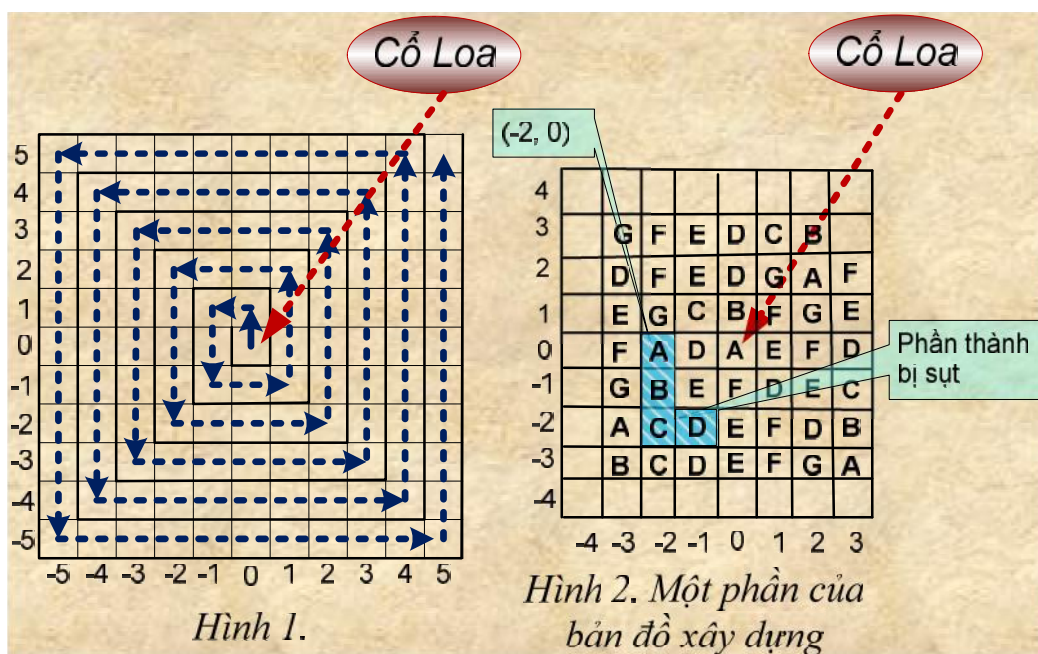
$$F_4 = \text{'DEFGABCDEFG'}$$

$$F_5 = \text{'DEFGABCDEFGABCDEF'},$$

...

L y các xâu F_i ($i = 1, 2, 3, \dots$) làm c m nang, An D ng V ng ã cho xây d ng thành c C Loa nh sau: V m t l i ô vuông vô h n, các c t c ánh s 0, 1, 2,

3, ... t trái sang phải tính từ cột qua C Loa, các cột bên trái C Loa ánh s -1, -2, ... t phải qua trái. Các hàng ánh s 0, 1, 2, 3, ... k t hàng i qua C Loa, t d i lên trên. Các hàng d i C Loa ánh s -1, -2, ..., t trên xuống d i. Ta g i ô n m giao c a c t i và hàng j là ô (i, j) . C Loa n m ô $(0, 0)$. B t u t ô $(0, 0)$, theo chi u xo n trôn c (xem hình 1), nhà vua i n m i ô m t ký t l y l n l t t ký t u tiên n ký t cu i cùng c a các xâu F_1 , ti p n là xâu F_2, F_3, \dots M i ký t ng v i m t lo i á, các ký t khác nhau ng v i các lo i á khác nhau. Ô ch a ký t nào thì o n thành trong ô ó ph i xây b ng lo i á t ng ng v i ký t ó. N u ô ánh d u b i ký t Ch thì chi phí xây d ng o n thành trong ô ó s là $ord(Ch)-64$. Ví d ô ánh d u là 'B' thì chi phí xây d ng o n thành trong ô này là $ord('B')-64 = 66-64 = 2$.



Nh ng h ban ngày quân dân xây d ng c n âu thì n êm l i có m t o n thành b s t, hôm sau ph i xây l i. Chi phí xây l i úng b ng chi phí xây m i. M t hôm, nhà vua r t phi n lòng vì quan t t ng b m báo là có m t o n thành i qua m ô b s t, b t u t ô (x, y) . Nhà vua yêu c u tính ngay chi phí c n thi t khôi ph c o n thành b s t này.

Yêu c u: Cho 2 xâu khác r ng F_1 và F_2 ch ch a các ch cái la tính in hoa, m i xâu có dài không quá 15, các s nguyên x, y và m . Hãy xác nh chi phí khôi ph c o n thành b s t i qua m ô b t u t ô (x, y) .

D li u: Vào t file v n b n CASTLE.INP:

- Dòng đầu tiên chứa chuỗi F_1 ;
- Dòng thứ hai chứa chuỗi F_2 ;
- Dòng thứ ba chứa 3 số nguyên x , y và m cách nhau một dấu cách ($1 \leq m \leq 10^{15}$, $|x|, |y| \leq 10^4$).

Kết quả : Tạo ra file văn bản CASTLE.OUT một số nguyên – chỉ phí tối thiểu tìm được.

Ví dụ :

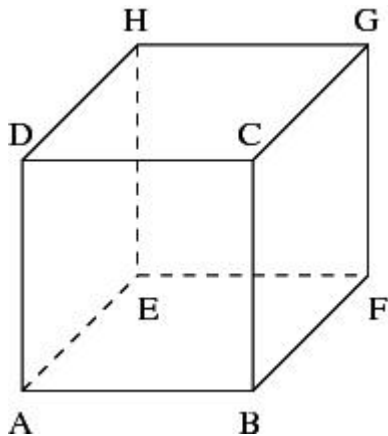
| CASTLE.INP | CASTLE.OUT |
|------------|------------|
| ABCDEF | 10 |
| DEFG | |
| -2 0 4 | |

(Mục đích của hai bài trên là phát hiện ra tính chất tuần hoàn, còn phần xử lý nhân ma trận thì khá đơn giản.)

Bài 3: Kiến.

(Nguồn: <http://vn.spoj.com/problems/PA06ANT/>)

Một chú kiến đi dọc trên mặt hình lập phương ABCDEFGH có một đường đi như:



Chú kiến muốn biết rằng có bao nhiêu con đường đi từ một đỉnh khác cho trước, đi qua đúng k cạnh (chú kiến luôn đi theo hướng này sang hướng kia liên tục). Nếu chú kiến đi qua một cạnh x lần, ta đếm cạnh đó x lần. Chú muốn

có một hành trình thứ v, vậy nên tìm kiếm chú s không cần phải đi mà mình đã tìm được ngay từ đầu.

Chú kí n của chúng ta không cần thông minh cho lắm, chú chỉ cần đi đúng các số từ 0 tới $p-1$, vậy nên bạn cần tính toán kết quả theo modulo p .

Yêu cầu

Hãy viết một chương trình thể hiện các công việc sau:

- * Nhận xuất phát và nhận kết thúc trên hành trình của chú kí n, số lượng của chú kí n muốn đi qua và modulo p ,

- * tính toán số lượng hành trình thứ v thỏa mãn các yêu cầu của chú kí n, theo modulo p ,

- * ghi đáp án ra output chuẩn.

Dữ liệu

Dòng đầu tiên của input chuẩn chứa hai chữ cái in hoa v_1 và v_2 , cách nhau bởi một dấu cách trống. Hai chữ cái này lần lượt thể hiện nhận xuất phát và nhận kết thúc trên hành trình của chú kí n. Dòng tiếp theo chứa hai số nguyên k và p , cách nhau bởi một dấu cách trống.

Kết quả

Ghi ra output chuẩn một số nguyên duy nhất là đáp án.

Ví dụ

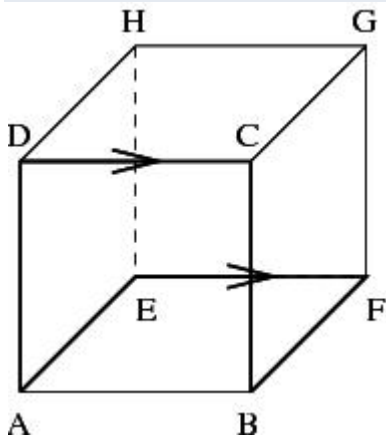
Dữ liệu:

A B

3 100

Kết quả :

2



Giải thích

- * A v_1, v_2 H, $v_1 = v_2$.

- * $1 \leq k \leq 2 \times 10^9, 2 \leq p \leq 10^9$.

Bài 4: Cal(ngu n: <http://vn.spoj.com/problems/C11CAL/>)

Cho N, k , tính $(1^k + 2^k + \dots + N^k) \bmod 1000000007$. Giả sử $1 \leq N \leq 10^9$, $1 \leq k \leq 50$.

Input

Gồm nhiều dòng, mỗi dòng chứa 2 số N, k là nhữg b test của bài

Output

Ghi ra kết quả của mỗi test

Ví dụ :

| Input | Output |
|-------|--------|
| 1 1 | 1 |
| 4 2 | 30 |

Chương II**QUY HOẠCH NG****THEO MÔ HÌNH QUY CỐ NH****I/ Phân tích vấn đề :**

Bài toán 1: Số Fibonacci xác định bởi công thức :

$$F[0] = 0; \quad F[1] = 1$$

$$F[n] = F[n-1] + F[n-2]$$

Hãy xác định số Fibonacci thứ n ($n \leq 40$)

Phân tích: Bài toán đã nêu rõ cho ta công thức truy hồi của $F[n]$, ta có thể dễ dàng viết được hàm quy tính $F[n]$:

```
Function F(n:longint):int64;
Begin
    If n<=1 then exit(n)
    else exit(F(n-1)+F(n-2));
End;
```

tính $F(n)$, máy tính sẽ phải gọi $F(n-1)$, 2 lần $F(n-2)$, 3 lần $F(n-3)$... => gọi nhiều lần liên tiếp, ta sẽ dùng mảng $Fx[i]$ để lưu lại giá trị của bài toán con $F(i)$

```
Fillchar(Fx,sizeof(Fx),255);
//Sau thao tác này Fx[i]=-1 với mọi i
Fx[0]:=0; Fx[1]:=1;
Function F(n:longint):int64;
Begin
    If fx[n]<>-1 then exit(fx[n]);
    Fx[n]:=f(n-1)+f(n-2);
Exit(fx[n]);
End;
```

pt $O(n)$

Vì hàm quy có như này, mọi bài toán con chỉ phải tính 1 lần. Ta cũng có thể cài đặt như hình vẽ vòng lặp như sau:

```
Fx[0]:=0; Fx[1]:=1;
For i:=2 to n do Fx[i]:=fx[i-1]+fx[i-2];
```

Vì cài đặt bằng quy có như có ưu điểm gì so với cách cài đặt bằng vòng lặp? Cùng xét bài toán sau:

Bài toán 2: Cho số nguyên dương N và S ($N \leq 20$; $S \leq 10^9$). Có bao nhiêu số nguyên dương $\leq 10^n$ và có tổng các chữ số bằng S ?

Phân tích: Khác với bài toán 1, vì cần tìm công thức truy hồi cho bài toán này không hề nghĩ đến như thuật duy theo lời quy có như bài toán trên nên như hàng h. Cần hình nghĩ mới của bài toán:

$$X_1 X_2 \dots X_n \text{ and } (X_1 + X_2 + \dots + X_n = S)$$

Theo thuật duy thông thường ta xây dựng cấu hình với 2 tham số (i, k) : xây dựng nên chuỗi i và tổng các chữ số hiện tại là k . Xét ra chuỗi $i+1$, lần lượt

th ch n ch s $i+1$ b ng $0,1,...,9$. Khi ã n ch s ta ki m tra xem $X_1+X_2+...+X_n=S$ hay không. N u th a m n ta c l nghi m.

G i $F(i,k)$ là hàm tính s c u hình th a m n bài toán khi xét n ch s th i c a nghi m và t ng các ch s $X_1+X_2+...+X_i=k$. Ta có hàm quy có :

```
Function F(i,k:longint):longint;
Var j,r:longint;
Begin
    If i>n then exit(ord(k=s));
    R:=0;
    For j:=0 to 9 do r:=r+f(i+1,k+j);
    Exit(r);
End;
```

T ng t bài toán 1, ta l u k t qu các bài toán con tránh ph i tính l p l i :

```
Fillchar(fx,sizeof(fx),255);
Function F(i,k:longint):longint;
Var j,r:longint;
Begin
    If fx[i,k]<>-1 then exit(fx[i,k]);
    If i>n then
        Begin
            Fx[i,k]:=ord(k=s);
            exit(ord(k=s));
        end;
    R:=0;
    For j:=0 to 9 do r:=r+f(i+1,k+j);
    Fx[i,k]:=r;
    Exit(r);
End;
```

pt : $O(n*s*9)$

K t qu bài toán : $F(1,0)$

Bài toán 3: Cho s nguyên d ng N, S, P ($N \leq 20; S \leq 10^9$). Xét dãy các nguyên d ng $\leq 10^n$ và t ng các ch s b ng S , tìm s l n th P trong dãy.

Phân tích: C u hình nghi m t ng t bài trên, áp d ng t t ng quy ta l n l t xây d ng nghi m b ng th t c try(i,k) : chu n b xây d ng n ch s th i, t ng các ch s h i n t i là k.

```
Procedure try(i,k:longint);
Var j:longint;
```

```

Begin
  If i>n then
    Begin
      If (p=1) and (k=s) then
        Begin
          Xuat;
          Halt;
        End;
      End;
    End;
  For j:=0 to 9 do
    Begin
      X[i]:=j;
      P:=p-1;
      Try(i+1,k+j);
      P:=p+1;
    End;
  End;
End;

```

Ta thấy khi xây dựng chuỗi s thì, ta có thể xác định các lần quy c n ph i th c hi n n u i t i p vào th t c **try(i+1,k+j)** (c ng chính b ng s c u hình s ph i th n u i t i p vào th t c ó) đ a vào hàm **F(i+1,k+j)** c a bài toán 2. Vì v y n u **p>F(i+1,k+j)**, t c l n u i t i p vào th t c **try(i+1,s+j)** c ng không th xây d ng c nghi m, thì ta s gi m p i m t l ng **F(i+1,s+j)** và không c n i vào ch ng trình con quy ó. Ng c l i n u **p<=F(i+1,s+j)** thì ch c ch n ch s th i là j, ta i t i p vào th t c **try(i+1,s+j)** r i break quá trình.

Code y c a bài toán 2 và bài toán 3:

```

Uses math;
Const fi='';
      fo='';
var f,g : text;
      res,n,k,p:int64;
      fx:array[1..21,0..21*9] of int64;
      x:array[1..21] of longint;
procedure mo;
begin
  assign(f,fi);
  reset(F);
  assign(g,fo);
  rewrite(g);
end;
procedure dong;
begin

```



```

        close(F);
        close(G);
end;
procedure nhap;
begin
    readln(F,n,k,p);
    fillchar(fx,sizeof(fx),255);
end;
function tinh(i,s:longint):int64;
var j:longint;
sum:int64;
begin
    if fx[i,s]<>-1 then exit(fx[i,s]);
    if i>n then
        begin
            fx[i,s]:=ord(s=k);
            exit(fx[i,s]);
        end;
    sum:=0;
    for j:=0 to 9 do
        sum:=sum+tinh(i+1,s+j);
    fx[i,s]:=sum;
    exit(fx[i,s]);
end;
procedure xuất;
var i:longint;
begin
    for i:=1 to n do if x[i]<>0 then break;
    for i:=i to n do write(g,x[i]);
end;
procedure try(i,s:longint);
var j:longint;
begin
    if i>n then
        begin
            if (p=1) and (s=k) then
                begin
                    xuất;
                    dong;
                    halt;
                end;
        end;
    for j:=0 to 9 do
        begin

```

```

        if p>fx[i+1,s+j]then p:=p-fx[i+1,s+j]
        else
        if p=fx[i+1,s+j] then
        begin
            x[i]:=j;
            try(i+1,s+j);
            break;
        end;
    end;
end;
procedure solve;
begin
    writeln(g,tinh(1,0));
    try(1,0);
end;
begin
    mo;
    nhap;
    solve;
    dong;
end.

```

- ph c t p c a quy có nh :

(kích th c m ng l u) * (s tr ng h p th trong m i CT con quy)

- ng d ng: - bài toán xác nh s l ng c u hình
 - bài toán xác nh c u hình khi bi t v trí
 - các bài toán quy ho ch ng nói chung, c bi t là nh ng bài khó tìm công th c truy h i
- Nh c i m: khó debug
- Mô hình chung c a th t c quy có nh :

```

procedure try(<các tham s >)
begin
    if <xây d ng xong c u hình> then

```

```

begin
<c p nh t k t qu >;
    halt;
end;
< tính toán các giá tr c n thi t t ch ng trình con
quy >
    //s d ng quy có nh
    if <giá tr này>giá tr c n i n nghi m> then
    {< i vào ch ng trình con quy>
        break;
    }
    else <tr giá tr ó vào nghi m hi n t i>
end;

```

II/ Ví d :

Bài 1:TỔ HỢP

Xét t p T ch a các s t nhiên t l n n ($1 \leq n \leq 50$). T h p ch p k ($1 \leq k \leq n$) c a n là t p con k s khác nhau t T . Trong m i t h p tìm c ng i ta s p x p các ph n t theo th t t ng d n và t ó – s p x p các t p con theo th t t i n. Ví d , v i $n = 4$ và $k = 2$ ta có 6 t h p ch p 2: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4) và (3, 4). Các t p con này c ánh s t 0 tr i.

Yêu c u: Cho 3 s nguyên n, k và m . Hãy tìm t h p ch p k c a n th m .

D li u vào trong file COMBIN.INP g m m t dòng ch a 3 s nguyên n, k, m .

K t qu ra file COMBIN.OUT g m k s nguyên c a t h p tìm c. Các s nguyên a ra theo th t t ng d n.

| COMBIN.INP | COMBIN.OUT |
|------------|------------|
| 4 2 4 | 2 4 |

H ng d n: th t c quy có nh g m 2 tham s (i, m) : chu n b xác nh ch s th i và ch s này $\geq m$

```

Procedure num(i,m:longint)
Begin
    if i>k then
    Begin
        res:=kq; exit;
    end;

```

```

        for j:=m to N do
            if P> x then P:=P-x
                {x là s l ng ch ng trình con quy t n cùng
t c khi vào num(i+1,j+1) }
            else
                Begin
                    kq[i]:=j;
                    num(i+1,j+1);
                    break;
                end;
            end;
        end;
end;

```

Ý t ng c a ph n ch ng trình chính trên là:

- Ch ng trình còn quy n v là ch ng trình con v i l tham s là $i=K+1$ (trong ch ng trình là $i>k$).
- Ch ng trình con quy num(i,m) s có nhi m v tìm giá tr th i cho t h p ch p K c a N c n tìm (nói úng h n là nghi m c a bài toán) sao cho giá tr ó không c bé h n m ($m-1$ là giá tr th i-1, do nghi m c n tìm ph i có giá tr t ng d n).
- V i m i ch ng trình quy, ta s xét các giá tr cho v trí i là $m \rightarrow N$. N u chúng ta chu n b i vào ch ng trình con quy mà s l n tìm t i ch ng trình con quy n v (hay nói cách khác là s t h p thu c) bé h n giá tr c a P còn l i thì ta s b qua và gi m P i s l n g i ch ng trình con quy n v ó.
- x c tính b ng quy có nh .

Bài 2: Light

(Ngu n: Vòng 2 n m 2011)

Hoàng m i mua c m t dây èn trang trí g m n bóng èn c m c n i t i p. Các bóng èn c ánh s t l n theo th t t bóng èn g n phích c m n bóng èn xa phích c m h n. Sau m t th i gian s d ng, Hoàng phát hi n ra qui lu t ho t ng c a dây èn. Khi ang c m i n ch có úng k bóng èn là sáng. C sau m i giây, dây èn l i chuy n tr ng thái theo qui t c sau ây:

- Tho t tiên lúc m i c m dây èn vào i n, các bóng èn t l n k sáng, các bóng èn còn l i t t. ây là tr ng thái u tiên c a dây èn.

- Trạng thái cuối cùng của dãy đèn là trạng thái trong đó k bóng đèn cuối cùng sáng, các bóng đèn còn lại tắt. Trạng thái này, xảy ra tiếp theo dãy đèn là lặp lại trạng thái đầu tiên.
- Tập hợp trạng thái trung gian chứa phải là trạng thái cuối cùng, dãy đèn chuyển trạng thái theo quy tắc sau:

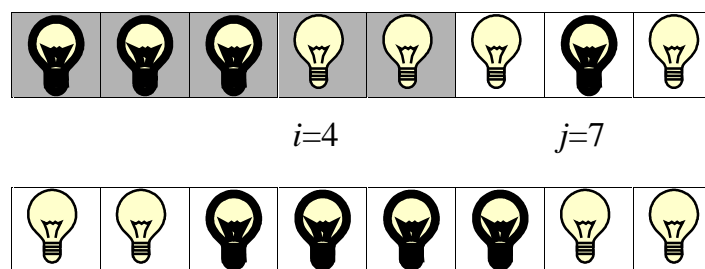
Bước 1. Tìm i là vị trí bóng đèn sáng gần nhất phải.

Bước 2. Tìm $j > i$ là vị trí bóng đèn tắt gần nhất.

Bước 3. Chuyển trạng thái bất kỳ của hai bóng đèn thứ $j-1$ và j .

Bước 4. Nếu $j > 3$, trạng thái của dãy bóng đèn từ 1 đến $j-2$ thu được bằng cách đảo ngược thứ tự dãy trạng thái hiện tại của dãy bóng đèn từ 1 đến $j-2$.

Ví dụ: Hình vẽ sau đây minh họa hai trạng thái liên tiếp của dãy gồm $n = 8$ bóng đèn và $k = 4$:



Phát hiện ra quy luật hoạt động của dãy bóng đèn, Hoàng muốn xác định trạng thái của dãy bóng đèn xảy ra tiếp theo bất kỳ lúc nào.

Yêu cầu: Cho ba số nguyên dương n , k và t , hãy đưa ra trạng thái của dãy bóng đèn xảy ra tiếp theo bất kỳ lúc nào.

Dữ liệu: Vào tệp văn bản `LIGHT.INP`:

- Dòng đầu tiên chứa p là số lượng dữ liệu;
- Mỗi dòng trong p dòng tiếp theo mô tả một bộ dữ liệu gồm ba số nguyên dương n , k và t ($k < n \leq 100$; $t \leq 2^{100}$). Các số trên cùng dòng ghi cách nhau bằng dấu cách.

K t qu : Ghi ra file v n b n LIGHT.OUT g m k dòng, m i dòng là m t xâu S dài n mô t tr ng thái c a dây bóng èn giây th t là k t qu t ng ng v i m t b d li u theo th t xu t hi n trong file d li u vào, trong ó ký t th j c a xâu S là B n u bóng èn th j sáng, và là T n u bóng èn th j t t.

Ví d :

| LIGHT.INP | LIGHT.OUT |
|-----------|------------|
| 2 | TBTTB |
| 5 2 8 | BTTTTTBBBT |
| 10 4 2011 | |

H ng d n: N u o ng c th t dây èn li thì bài toán tr thành bài Combin + x lí s nguyên l n

Bài 3: N13

Cho A, B hãy xác nh s l ng các s n m trong o n $[A, B]$ mà trong d ng bi u di n c a nó không xu t hi n s 13.

D li u vào: N13.INP

G m nhi u dòng, m i dòng ch a 2 s nguyên A, B ($0 \leq A \leq B \leq 10^{15}$)

D li u ra: N13.OUT

ng v i m i dòng trong input là m t dòng ghi s l ng s th o m n.

M t ví d v Input, Output:

| N13.inp | N13.out |
|----------|---------|
| 1 13 | 12 |
| 100 1000 | 882 |

H ng d n: s l ng nghi m c a bài toán thu c $[A, B]$ chính b ng s l ng nghi m trong $[0, B]$ tr i s l ng nghi m trong $[0, A-1]$

Bài toán tr thành : m s l ng s n m trong o n $[0, A]$ mà trong bi u di n c a nó không xu t hi n s 13

C u hình nghi m : $(X_1X_2...X_n \text{ and } X_iX_{i+1} \leq 13 \text{ and } X_1X_2...X_n \leq A)$ ($n = \text{length}(A)$)

Các tham s c n dùng : $\text{tinh}(i,d,ok)$: xây d ng n ch s th i, ch s th i-1 là d, bi n ok thu c ki u boolean dùng ki m soát xem nghi m ang xây d ng ã $\leq A$ ch a?

- $Ok = \text{true} \Leftrightarrow X_1X_2...X_i0...0 \leq A$

- $Ok = \text{false} \Leftrightarrow X_1X_2...X_i... = A$

$\text{Fx}[i,d,ok] = \text{tinh}(i+1,j,ok \text{ or } (j < A[i]))$

$J = 0 \text{ to } \max(A[i], \text{ord}(ok)*9)$

```
function tinh(i,d:longint;ok:boolean):int64;
var j:longint;
sum:int64;
begin
    if fx[i,d,ok] <> -1 then exit(fx[i,d,ok]);
    if i > n then
        begin
            fx[i,d,ok] := 1;
            exit(1);
        end;
    sum := 0;
    for j := 0 to max(ord(st[i])-48, ord(ok)*9) do
        if d*10+j <> 13 then sum := sum + tinh(i+1,j,ok or
(j < ord(st[i])-48));
        tinh := sum;
        fx[i,d,ok] := sum;
    end;
```

K t qu : $\text{tinhB}(1,0,\text{false}) - \text{tinhA}(1,0,\text{false})$

pt : $O(15*9*2*9)$

Bài 4: Clear <ngu n bài: <http://vn.spoj.com/problems/CLEAR/>>

B m m i tìm c m t tài li u nh ngh a s rõ ràng nh sau: V i s nguyên d ng n, ta t o s m i b ng cách l y t ng bình ph ng các ch s c a nó, v i s m i này ta l i l p l i công vi c trên. N u trong quá trình ó, ta nh n c s m i là 1, thì s n ban u c g i là s rõ ràng. Ví d , v i n = 19, ta có:

19 82 ($= 1^2 + 9^2$) 68 100 1

Nh v y, 19 là s rõ ràng. Không ph i m i s u rõ ràng. Ví d , v i n = 12, ta có:

12 5 25 29 85 89 145 42 20 4 16 37 58 89
145

B m r t thích thú v i nh ngh a s rõ ràng này và thách phú ông: Cho m t s nguyên d ng n, tìm s $S(n)$ là s rõ ràng li n sau s n, t c là $S(n)$ là s rõ ràng nh nh t l n h n n. Tuy nhiên, câu h i ó quá đ v i phú ông và phú ông ã l i B m: Cho hai s nguyên d ng n và m ($1 \leq n, m \leq 10^{15}$), hãy tìm s $S^m(n) = S(S(\dots S(n)))$ là s rõ ràng li n sau th m c a n.

D li u

- Dòng u là s t ($0 < t \leq 20$) là s b d li u.
- t dòng sau, m i dòng ch a 2 s nguyên n và m.

K t qu

G m t dòng, m i dòng là k t qu t ng ng v i d li u vào.

Ví d

D li u

2
18 1
1 145674807

K t qu

19
1000000000

H ng d n:

M t s là s rõ ràng n u t ng bình ph ng các ch s c a nó là m t s rõ ràng.

Chu n b tr c m ng Isclear[0..maxlengh*9*9] of boolean d dàng ki m tra tính rõ ràng c a m t s .

C u hình nghi m : $(X_1 X_2 \dots X_{16} \text{ and Isclear}[X_i^2])$

Các tham s c a hàm clear(i,s,ok) : xây d ng n ch s th i, t ng bình ph ng các ch s hi n t i là s, bi n ok ki m soát nghi m $> n$

fx[i,s,ok]:=fx[i,s,ok]+clear(i+1,s+j*j,ok or ord(j)>N[i]);

j:=(1-ok)*A[i] to 9

Code m u có th tham kh o SGK chuyên tin quy n 1, trang 236

Bài 5:PDS

Một số tự nhiên n được gọi là số PDS nếu tích các chữ số của n chia hết cho tổng các chữ số của n .

Yêu cầu: Cho một số n trong $[A, B]$, đếm số PDS trong n $[A, B]$

Input : Một số nguyên không âm A và B ($A \leq B \leq ???$)

Output Ghi số PDS trong n $[A, B]$

| PDS.INP | PDS.OUT |
|---------|---------|
| 5 20 | 7 |

Hướng dẫn: Ý tưởng: liệt kê xây dựng các nghiệm có tổng các chữ số là s .

Các tham số sử dụng cho hàm **tinh(i,ok1,ok2,start,r,s:longint)** : xây dựng dãy chữ số thứ i , 2 biến **ok1, ok2** kiểm soát nghiệm thu $[A, B]$ (có thể tách riêng 2 số $[0, A]$ và $[0, B]$ như các ví dụ trên), biến **start**: boolean kiểm soát xem nghiệm đã bắt đầu có nghĩa chưa, s là tổng các chữ số của nghiệm đang xây dựng, r là số để tích các chữ số của nghiệm đang xây dựng khi chia cho s .

Res= tinh(1,false,false,false,1,s)

S=1 to length(B)*9

pt: length(B)^4*9^4*2^3 => complexity $n \leq 10^{13}$

Bài 7: T p s <nguồn bài: chuyên đề tuyển Chuyên - HSP Hà Nội 2011>

Cho số nguyên không âm n và các số 0 không có nghĩa. Bằng cách xóa một hoặc nhiều chữ số của n (không xóa hết các chữ số của n) ta nhận được một số nguyên mới. Số mới được tạo ra bằng cách xóa các chữ số 0 vô nghĩa của n có. Tập số nguyên D được xây dựng bằng cách đưa vào nó số, các số mới khác nhau về giá trị và khác n . Ví dụ, với $n=101$ ta có thể nhận được các số mới như sau:

- bằng cách xóa một chữ số ta có các số (101), 11, 10;
- bằng cách xóa hai chữ số ta có các số : 1, 1, 0;

Tập **D** nhận được là $\{0, 1, 10, 11, 101\}$

Yêu cầu: Cho số nguyên n và 2 số nguyên không âm A, B . Hãy xác định số lượng các số m trong $[A, B]$ có mặt trong tập **D** được tạo thành từ n .

D li u: Numset.inp

- Dòng 1: ch a s nguyên n;
- Dòng 2: ch a s nguyên A;
- Dòng 3: ch a s nguyên B;

K t qu : Numset.out :s l ng s tìm c

| Numset.inp | Numset.out |
|------------|------------|
| 101 | 3 |
| 1 | |
| 100 | |

Subtask 1: $n, A, B < 10^{18}$

Subtask 2: $B - A < 1000$ và n, A, B có không quá 30000 ch s

Subtask 3: $n, A, B < 10^{1000}$

H ng d n:

Subtask 1 : quy

Subtask 2 : duy t + bignum

Subtask 3: QH quy có nh + bignum

Chu n b tr c next[i,j] : v trí c a ch s j xu t hi n ngay sau v trí i c a n

(vd $n=4321$ thì $\text{next}[1,4]=1$)

tinh(i,j:longint;ok1,ok2,start:boolean)

- Xây d ng n ch s th i
- S d ng n ch s th j c a n
- Ok1,ok2 : ki m soát nghi m thu c [A,B]
- Start : ki m soát nghi m ã có ngh a ch a?

III/ Bài t p t luy n:

Bài 1: M S L NG

Cho 3 số nguyên không âm N, A và $B (0 \leq N \leq A \leq B \leq 10^{18})$. Hãy tính số lượng số nguyên trong đoạn $[A, B]$ mà số chia N không chia hết cho N .

Ví dụ, với $N = 3, A = 0$ và $B = 17$ ta có 2 số (các số 3 và 13).

Dữ liệu: vào tệp văn bản *COUNT3.INP*, gồm nhiều tests, mỗi dòng chứa 3 số nguyên A, B, N ứng với mỗi bộ dữ liệu cần tính và kết thúc bằng dòng chứa 3 số -1 -1 -1, dòng này không cần tính.

Kết quả: ra tệp văn bản *COUNT3.OUT*, mỗi kết quả ứng với mỗi bộ 3 số test dữ liệu vào sẽ in ra trên một dòng.

| COUNT3.INP | COUNT3.OUT |
|------------|------------|
| 3 17 3 | 2 |
| 0 20 0 | 3 |
| -1 -1 -1 | |

Bài 2: Palinn

Xét các số tự nhiên có N chữ số thoả mãn tính chất sau:

- Là số lẻ
- Số chia hết cho M

Ví dụ: $N=3, M=9$ ta có 10 số 171, 252, 333, 414, 585, 666, 747, 828, 909, 999 là số lẻ và chia hết cho 9

Yêu cầu: Cho N, M và K

- Hãy đếm số lượng số lẻ có N chữ số và chia hết cho M
- Trong các số đó, in ra số lớn nhất K

Dữ liệu: vào trong tệp “Palinn.inp” có dạng

- Dòng thứ nhất gồm 2 số $N, M (2 < N < 40, 0 < M < 100)$
- Dòng thứ hai gồm 1 số K

Kết quả: ghi ra tệp “Pallin.out” có dạng

- Dòng đầu là số lượng số cần xét
- Dòng thứ 2 là số lớn nhất K

| Palinn.inp | Palinn.out |
|------------|------------|
| 3 9 | 10 |
| 3 | 333 |

Bài 3: Minlex2

Cho 2 số nguyên dương $1 \leq L \leq R \leq 10^9$ và số nguyên dương K ($K \leq 10^9$)

Tìm số X có thể viết thành tổng của n số nguyên dương $L \leq X \leq R$ và chia hết cho K

Input: minlex2.inp

- Gồm hai dòng, mỗi dòng chứa 3 số L, R, k

Output: minlex2.out

- Gồm hai dòng, mỗi dòng chứa số X tìm được. Nếu không tồn tại ghi -1

| Minlex2.inp | Minlex2.out |
|-------------|-------------|
| 4 7 5 | 5 |
| 11 19 10 | -1 |
| 4 111 20 | 100 |

Bài 4: Số siêu may mắn

Một số được gọi là siêu may mắn nếu nó có k chữ số 6 liên tiếp. Cho số n và k , hãy tìm số siêu may mắn nhỏ nhất có n chữ số.

Input

- Gồm 2 số nguyên dương n, k ($n, k \leq 1000$)

Output

- Gồm một số là số siêu may mắn nhỏ nhất có n chữ số mod 111539786.

| SLN.INP | SLN.OUT |
|---------|---------|
| 2 1 | 18 |

2 2

1

Bài 5: Puzzle of numbers

Khi m t s ph n ch s trong ng th c úng c a t ng hai s nguyên b m t (c thay b i các d u “*”). C m t câu là: Hãy thay các d u sao b i các ch s cho ng th c v n úng.

Ví d b t u t ng th c sau:

9334

789

10123 (9334+789=10123)

Ví d các ch s b m t c thay b ng các d u sao nh sau:

*3*4

78*

10123

Nhi m v c a b n là vi t ch ng trình thay các d u sao thành các ch s c m t ng th c úng. N u có nhi u l i gi i thì a ra m t trong s ó. N u không có thì a ra thông báo: “No Solution”. Chú ý các ch s u m i s ph i khác 0.

D li u: vào trong file “REBUSS.INP”: g m 3 dòng, m i dòng là m t xâu ký t g m các ch s ho c ký t “*”. dài m i xâu không quá 50 ký t . Dòng 1, dòng 2 th hi n là hai s c c ng, dòng 3 th hi n là t ng hai s .

K t qu : ra file “REBUSS.OUT”: N u có l i gi i thì file k t qu g m 3 dòng t ng ng v i file d li u vào, n u không thì thông báo “No Solution”

| REBUSS.INP | REBUSS.OUT |
|------------|------------|
| *3*4 | 9334 |
| 78* | 789 |
| 10123 | 10123 |

Bài 6: Jesnakost <ngu n bài: Croatia regional competition 2008>

<http://vn.spoj.com/problems/JEDNAKOS/>

Trong khi c m t quy n sách toán, Mirko nh n th y có m t s ng th c l d ng $A=S$. i u kì l là ng th c này không úng ($A \neq S$). Mirko nh n th y r ng c n ph i thêm m t s d u c ng vào gi a l s c p ch s liên ti p v trái c a ng th c. B n

hãy tìm cách thêm vào ít nhất các dấu cộng vào vế trái của đẳng thức để đẳng thức trở nên đúng. Các số nguyên có thể có các chữ số 0 vô nghĩa.

Input

Gồm một dòng duy nhất chứa đẳng thức $A=S$

A và S không có chữ số 0 vô nghĩa. $A \neq S$

A có nhiều nhất 1000 chữ số; $S \leq 5000$

Dữ liệu liên tiếp cho đến khi kết thúc

Output

Gồm một dòng duy nhất chứa số lượng dấu cộng ít nhất cần thêm

Example

Input:

143175=120

Output:

2

Input:

5025=30

Output:

1

Input:

999899=125

Output:

4

Bài 7: Degree <nguồn bài: Rybinsk State Avia Academy>

<http://vn.spoj.com/problems/DEGREE/>

Một số nguyên dương A gọi là có bậc K nếu tồn tại B và n sao cho:

$$A = B^{x_1} + B^{x_2} + \dots + B^{x_k}$$

(trong đó x_1, x_2, \dots, x_k là các số nguyên không âm thoả mãn $x_1 < x_2 < x_3 < \dots < x_k$)

Ví dụ:

• 17 có b c 2 i v i c s 2 vì $17 = 2^4 + 2^0$.

• 151 có b c 3 i v i c s 5 vì $151 = 5^3 + 5^2 + 5^0$.

Yêu c u : Cho tr c l o n $[X, Y]$. Hãy xác nh xem trong o n này có bao nhiêu s có b c K i v i c s B.

Gi i h n :

• $1 \leq X \leq Y \leq 10^9$

• $1 \leq K \leq 25, 2 \leq B \leq 9$

• Ch y c v i b nh thông báo < 800 K b n m i th c s là thành công

Input

1 dòng g m 4 s nguyên d ng X, Y, K, B

Output

G m 1 dòng duy nh t ghi ra s l ng s tìm c .

Example

Input:

15 20 2 2

Output:

3

(Gi i thích : ó là các s $17 = 2^4 + 2^0$, $18 = 2^4 + 2^1$, $20 = 2^4 + 2^2$)

G i ý: K t qu = s s v i t trong h c s B ch dùng ch s 0 và 1.

Bài 8: Pizza <ngu n bài: The 36th Annual ACM Asian Regional – Daejeo

https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&category=513&page=show_problem&problem=3858

Có 1 c a hàng pizza c t trên 1 ng th ng, trên ng th ng ó c ng có nhi u h gia ình là khách hàng mua pizza. thu hút khách hàng, ch c a hàng qu ng cáo r ng anh ta s gi m giá cho nh ng l t a pizza b n tr . C m i giây b tr thì khách hàng s c gi m 1 Vi t Nam ng. Vi t m t ch ng chnh tính ra nên a pizza n cho khác hàng b ng ng ình th nào t i a l i nhu n. L u ý là ph i không c n ph i a h t cho t t c khách hàng n u th y không c n thi t. L i nhu n = t ng giá c a pizza sau khi ã tr th i gian a tr . B i tr ng m i n v dài i m t 1 giây và không m t th i gian giao pizza.

Xem ví d trong link hi u rõ h n.

Input:

Dòng u tiên là T – s test. T i p theo là các dòng mô t T test.

M i test b t u b ng 1 dòng là s gia ình – n. $1 \leq n \leq 100$

Dòng t i p theo là n s nguyên mô t v trí trên ng th ng c a n h gia ình theo th t t ng d n. Các v trí này khác 0 – là v trí c a c a hàng pizza.

Dòng tiếp n là n s nguyên d ng bi u di n giá tr pizza các h gia ình yêu c u.
T t c các s 2 hàng trên n m trong kho ng -100000 n 100000.

Output:

M i test ghi ra 1 dòng là t ng l i nhu n l n nh t có th t c

Ví d :

| Input | output |
|---|--------|
| 3 | 32 |
| 5 | 13 |
| -6 -3 -1 2 5 | 1937 |
| 27 10 2 5 20 | |
| 6 | |
| 1 2 4 7 11 14 | |
| 3 6 2 5 18 10 | |
| 11 | |
| -14 -13 -12 -11 -10 1 2 3 4 5 100 | |
| 200 200 200 200 200 200 200 200 200 200 200 200 | |

Bài 9: M T MÃ

M t mã h c là ngành khoa h c nghiên c u và ng d ng các k thu t, công c Toán h c b o v an toàn thông tin. Là ng i r t yêu Toán h c c ng nh M t mã h c, Minh quy t nh t mình xây d ng m t h th ng mã hóa riêng cho mình. Minh ch n tr c m t s nguyên d ng n . Khóa có dài k là m t dãy g m k s nguyên d ng (a_1, a_2, \dots, a_k) th a mãn các tính ch t sau:

- M i s a_i u là c s c a n,
- $a_1 < a_2 < \dots < a_k$,
- Các s a_i và a_{i+1} là nguyên t cùng nhau v i $1 \leq i < k$.



Ghi chú: S nguyên d ng a c g i là c s c a s nguyên d ng b n u t n t i s nguyên d ng c sao cho $b = a \times c$. Hai s nguyên d ng c g i là **nguyên t cùng nhau** n u c s chung l n nh t c a hai s này là 1. Ví d nh 16 và 27 là hai s nguyên t cùng nhau.

Ví d : V i $n = 360$, dãy s $(2, 9, 10)$ là m t khóa có dài $k = 3$.

Yêu c u: Cho tr c giá tr c a n và k (v i $2 \leq n \leq 10^8$ và $2 \leq k \leq 10$). Hãy xác nh s l ng các khóa khác nhau có dài k .

D li u: c cho trong file v n b n **MATMA.INP** g m 1 dòng ch a 2 s nguyên d ng n và k .

Kết quả : Ghi vào file văn bản **MATMA.OUT** kết quả nguyên là số lẻ của dãy số chẵn hóa giá trị của số n .

Các số trên cùng một dòng (trong file dữ liệu cũng như file kết quả) cách nhau bởi ký tự khoảng trắng.

Ví dụ :

| MATMA.INP |
|-----------|
| 10 3 |

| MATMA.OUT |
|-----------|
| 1 |

Chương III

C U TRÚC DEQUEUE & LEFT RIGHT

I/ Phân tích vấn đề :

Bài toán: Cho 1 dãy số nguyên $A[1], A[2], \dots, A[n]$ và 1 số nguyên dương k ($k \leq n \leq 1000000$). Gọi $Kmin[i]$ là giá trị nhỏ nhất trong dãy gồm k phần tử $A[i], A[i+1], \dots, A[i+k-1]$ ($i \leq n-k+1$). Hãy xác định $Kmin[1], Kmin[2], \dots, Kmin[n-k+1]$ và ghi ra kết quả theo thứ tự đó.

Hướng dẫn : Nhận thấy $Maxn = 10^6$. Vì quy tắc bài, ta có thể dùng cấu trúc deque với độ phức tạp $O(n)$. Xét dãy gồm k phần tử liên tiếp $A[i], A[i+1], \dots, A[i+k-1]$, gọi $left[i]$ là số ưu tiên bên trái nhỏ nhất của $A[i]$; Nếu ta xây dựng dãy con có q phần tử ($q \leq k$): $A[j_1], A[j_2], \dots, A[j_q]$ (trong đó $i \leq j_1 < j_2 < \dots < j_q = i+k-1$), sao cho

$left[j_t] = a[j_{t-1}]$ ($t > 1$), và $left[j_1]$ không tồn tại trong dãy đang xét thì khi đó dãy đang nhận thấy $A[j_1]$ là giá trị nhỏ nhất của dãy $A[i], A[i+1], \dots, A[i+k-1]$. (chú ý: $j_q = i+k-1$);

Trên đây là thuật toán, trong khi cài đặt ta không cần dùng đến mảng $left$ mà chỉ cần dùng mảng $d[1..maxn]$, 2 chỉ số "đầu" (u) và "cuối" (cuối) (tên gọi stack 2 chiều) lưu các chỉ số j_1, j_2, \dots, j_q nêu trên (lưu ý, ta chỉ cần lưu chỉ số (số thứ tự) của phần tử là quan trọng nhất). Khi đó, $j_1 = d[đầu]$ (kết quả tìm vị trí đang xét $= a[j_1] = a[d[đầu]]$, xem hướng dẫn bên dưới);

Ban đầu deque rỗng ($dau:=1; cuoi:=0$). Duy trì mảng A từ trái sang phải, nên vị trí i , ta có 2 thao tác:

1. Add(i): Thêm i vào mảng deque.

```
procedure add(i:longint);
begin
    while (cuoi>=dau) and (a[d[cuoi]]>=a[i]) do
        dec(cuoi); {chúng ta phải di chuyển con trỏ cuối về phía trước để có thể thêm phần tử mới}
    inc(cuoi); d[cuoi]:=i; {thêm phần tử mới vào mảng}
end;
```

2. Extract(i): Nếu $i \geq k$ thì xóa phần tử thứ i khỏi deque.

```
procedure extract(i:longint);
begin
    if d[dau]<i-k+1 then inc(dau); {nếu phần tử cần xóa nằm ngoài phạm vi deque, ta phải di chuyển con trỏ đầu về phía trước}
end;
```

→ Với vị trí $i (i \geq k)$ ta thu được $Kmin[i-k+1]=a[d[dau]]$;

Code mẫu:

```
const    fi='';
         fo='';
maxn=1000001;

var  f,g:text;
     n,k,dau,cuoi:longint;
     a,d:array[0..maxn]of longint;

procedure mf;
begin
    assign(f,fi);reset(f);
```

```

        assign(g,fo);rewrite(G);
end;

procedure df;
begin
    close(f);close(g);
end;

procedure nhap;
var i:longint;
begin
    readln(f,n,k);
    for i:=1 to n do read(f,a[i]);
end;

procedure add(i:longint);
begin
    while (cuoi>=dau) and (a[d[cuoi]]>=a[i])do dec(cuoi);
    inc(cuoi);
    d[cuoi]:=i;
end;

procedure extract(i:longint);
begin
    if d[dau]<i-k+1 then inc(dau);
end;

procedure xl;
var i,j:longint;
begin
    dau:=1;cuoi:=0;
    for i:=1 to n do
        begin
            add(i);
            if i>k then extract(i);
            if i>=k then writeln(g,a[d[dau]]);
        end;
    end;
end;
BEGIN
    mf;
    nhap;
    xl;
    df;
END.

```

L u ý : Trên đây là bài toán tìm min, n u yêu c u tìm max, trong procedure add(i) ta có ch c n s a d u '>=' thành d u '<=' là OK ^^

```
procedure add(i:longint);
begin
    while (cuoi>=dau) and (a[d[cuoi]]<=a[i]) do
    dec(cuoi); {ch ng nào deque không r ng và A[ph n t cu i
    cùng trong deque]>=A[i] thì lo i b ph n t cu i cùng ố}
    inc(cuoi); d[cuoi]:=i; {n p ch s i vào m ng d}
end;
```

Trong gi i thích c a thu t toán trên có c p t i khái ni m Left[i] là **giá tr** c a s u tiên bên trái i mà nh h n A[i]. Ta có th xác nh L[i] là **v trí tr** c i và g ni nh t c a ph n t có giá tr bé h n A[i], R[i] là **v trí c** a ph n t sau i và g ni nh t c a ph n t bé h n A[i] b ng o n code n gi n, d hi u nh sau:

```
A[0]:=-maxlongint ; A[N+1]:=-Maxlongint;
For i:=1 to N do
Begin
    J:=i-1;
    While A[j]>=A[i] do j:=L[j];
    L[i]:=j;
End;
For i:=N downto 1 do
Begin
    J:=i+1;
    While A[j]>=A[i] do j:=R[j];
    R[i]:=j;
End;
```

Tuy trong vòng for có vòng l p while nh ng ng i ta ã ch ng minh c là trung bình m i vòng while th c hi n ch t m 3 n 4 phép tính thôi nên ph c t p c a thu t toán trên v n là $O(N)$.

M t cách t ng t ta có th xác nh c **v trí c** a s li n tr c hay li n sau i mà **l nh n** A[i]. Tuy b n ch t là t ng ng nh ng c u trúc Left-Right có th thay th Dequeue trong nhi u bài toán b i tính n gi n, ti n l i c a nó.

II/ Bài tập luyện:**Bài 1: MINK**(Ngu n: <http://vn.spoj.com/problems/MINK/>)

Do này tivi công nghệ chỉ u phim L c Vân Tiên , s n t n l y luôn làm tiêu L c Vân Tiên công nghệ Samurai Jack , b Quan Thái S y vào vòng xoáy th i gian và b chuyển t i t ng lại c a nh ng n m 2777 .

v t qua vòng lo i , Vân Tiên c n tham gia cu c thi sát h ch . Ban Giám Kh o cu c thi sát h ch g m có N ng i , h u là các cao th trong gi i IT . Các thành viên trong Ban Giám Kh o c ánh s t $1 \rightarrow N$ và m i ng i l i có m t ch s s c m nh g i là APM (Actions Per Minute) . Các giám kh o s x p hàng l n l t t $1 \rightarrow N$. M i thí sinh s ph i u v i K v giám kh o và K v giám kh o này ph i ng li n thành l o n (T c là $i, i+1, i+2, \dots, i+K-1$) , ch c n th ng l v giám kh o thì s v t qua vòng lo i .

Tuy nhiên thí sinh kô c ch n xem nh ng giám kh o nào s u v i mình . Vân Tiên r t lo vì l may ng v i nh ng v giám kh o nào "khó nh n" thì s tiêu m t . Nên chi n thu t c a Vân Tiên là t p trung h v giám kh o có ch s APM th p nh t trong s K v . B n h y l p trình giúp L c Vân Tiên xác nh c t t c các ph ng án thì ch s APM c a v giám kh o th p nh t s là bao nhiêu (Có t t c $N-k+1$ ph ng án :

Ph ng án 1 : Vân Tiên ph i u v i v $1 \rightarrow v_k$ Ph ng án 2 : Vân Tiên ph i u v i v $2 \rightarrow v_{k+1}$

...

Ph ng án $N-k+1$: Vân Tiên ph i u v i v $N-k+1 \rightarrow v_N$) .($1 \leq N \leq 17000$, ch s APM c a l giám kh o ≥ 1 và $\leq 2t$, $1 \leq K \leq N$) .**Input**

Dòng 1 : s T là s test .

T i p theo là T b test , m i b test có format nh sau :

Dòng 1 : N k

Dòng 2 : N s nguyên d ng $A[1], \dots, A[N]$.**Output**

K t qu m i test ghi ra trên dòng , dòng th i g m $N-k+1$ s , s th j t ng ng là ch s APM c a v giám kh o y u nh t trong ph ng án j .

Example

Input:

2
4 2
3 2 4 1
3 3
1 2 3

Output:

$$\begin{array}{ccc} 2 & 2 & \overline{1} \\ 1 & & \end{array}$$

G i ý: ây là bài toán t ng ng c a bài toán t ra ph n u v n .

Bài 2: VOCARD (Nguồn: <http://vn.spoj.com/problems/VOCARD/>)

RR và *pirate* là hai cao thủ chửi bài. Bài gì họ cũng bắt nạt chửi qua và lần nào cũng ngang tài ngang sức với nhau. Bây giờ thì họ đã chửi tất cả các thủ lĩnh bài rồi mà vẫn chưa phân thắng bại. Vì vậy họ tìm gặp pháp sư *flashmt* nhờ anh ta sáng chế ra một kỹ thuật chửi bài mới để tranh tài tranh sức với nhau.

flashmt v n c ng thích ch i bài t nh nên anh y ã nhanh chóng ngh ra c m t trò ch i nh sau: V pháp s l y t túi qu n ra l b bài N lá, và t xu ng bàn. Các lá bài c ánh s t l n N t d i lên. M i l t ch i, ng i ch i ph i ch n l y ít nh t là l lá bài và nhi u nh t là K lá bài t trên xu ng, t c là u tiên lá bài N s c l y, r i n N-1, N-2,..., t t nhiên s l ng bài c ch n không v t quá s l ng lá bài còn l i. Trong các lá bài ó có m t s lá bài màu en, còn m t s khác thì màu . *RR và pirate* c bi t tr c v trí c a nh ng lá bài màu en và trên b bài, ng i nào b c c l lá bài màu en t ng ng v i l i m. Hai ng i s l n l t l y các lá bài cho t i khi nào t t c N lá bài u c l y h t. Cu i cùng ai c nhi u i m h n s th ng.

RR và *pirate* ch ị bài r t gi ị nên cho dù v ị trò ch ị m ị này h c ng luôn bi t cách ch ị t ị u, ngh a là h s luôn c g ng c nhi u ị m nh t có th . Nh ng h ang th c m c li u trò ch ị m ị này có giúp h phân th ng b ị hay không? Hay k t qu l ị hòa nh nh ng l n tr c? ị u này thì ngay c pháp s *flashmt* c ng không bi t c. B n là m t l p trình vi ị gi ị, hãy giúp *RR*, *pirate* và *flashmt* tr l ị câu h ị này nh ế.

Yêu c u

Cho s N, K, trạng thái màu của các lá bài t 1 n N. Hãy trả lời xem li u khi k t thúc trò ch i thì hai ng i ch i có th phân c th ng b i hay không? N u câu tr l i là có thì ng i thua s c bao nhiêu i m? Chú ý: Chúng ta không quan tâm n ng i nào i tr c ng i nào i sau mà ch quan tâm n i m c a ng i th ng và ng i thua sau khi trò ch i k t thúc.

Input

- Dòng u tiên ch a 2 s N, K
- Dòng th 2 ch a N kí t '0' ho c '1' (không có d u ngo c d n) mô t trạng thái màu c a các lá bài t d i lên. Kí t th i là '0' n u lá bài c ánh s i màu . Kí t th i là '1' n u lá bài c ánh s i màu en.

Output

- Dòng u tiên là "YES" (không có d u ho c kép) n u sau khi k t thúc trò ch i thì hai ng i ch i có th phân b i t c th ng b i. Ho c là "NO" n u trò ch i k t thúc v i k t qu hòa.
- N u dòng u tiên là "YES" thì dòng này s là s i m cao nh t mà ng i thua t c.

Gi i h n:

- 40% s test có 1 K N 5000.
- Trong t t c các test có 1 K N $2 \cdot 10^6$ (2 tri u)

Example

Input 1:

5 3
10110

Output 1:

YES
1

Input 2:

4 2
1111

Output 2:

NO

Gi i thích:

- Ví dụ 1: Nếu người chơi có 3 lá bài trên cùng (lá 5, 4 và 3) thì sẽ có 2 điểm (lá 3 và lá 4), người chơi sau có 2 lá bài còn lại (lá 2 và 1) thì có 1 điểm (lá 1). Như vậy người chơi trước đã thắng người chơi sau. Người chơi sau là người thua có thể có nhiều nhất là 1 điểm.

- Ví dụ 2: Sau khi kết thúc trò chơi có 2 người chơi có nhiều nhất là 2 điểm. Vì vậy hai người không phân thắng bại.

Ghi ý: Xét dãy các quân bài từ 1 đến i : Gọi $a[i]$ là số điểm của quân bài thứ i (người chơi trước).

Gọi $sum[i]$ là số quân bài màu đen trong các quân bài từ 1 đến i .

- Nếu $2 * a[n] = sum[n]$ thì người chơi Hòa;

- Nếu $2 * a[n] < sum[n]$ thì người chơi phân thắng bại. Người chơi thua là $\min(a[n], sum[n] - a[n])$;

xây dựng mảng $a[i]$, ta thấy các trường hợp bài có thể có vị trí i thì a mãi bài (bộ 1 hay 2 hay 3... quân bài). Ta có

$a[0] := 0$;

$a[i] = sum[i] - \min(a[j])$ vs j thì a mãi người chơi có thể bộ các quân bài từ $j+1$ đến i nên số điểm của deque.

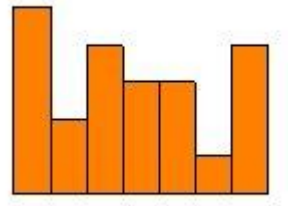
* Với các vị trí từ 1 đến k thì cách tính là bộ h t s bài còn lại.

Bài 3: KPLANK <http://vn.spoj.com/problems/KPLANK/>

Nếu các bạn biết câu chuyện tình tâm "nàng đi leo trèo vàng" của Pirate hẳn sẽ phải khóc hết nước mắt khi anh ấy, vì lòng thương chim, đã bán trái tim đi leo siêu bộ của mình.

Đi leo xong đã bị chim to lấy đi, Pirate giờ chuyển sang nghề bán dưa bầu. Bộ đồ chơi, vì trên đó toàn là dưa...

Nhưng mà bán cái gì thì ưu tiên công phôi có bán hay không. Pirate quyết định lòng s c trên đó các mảnh ván còn sót lại của anh chàng con tàu sẽ ghép lại thành tấm biển. Cùng anh chàng tìm kiếm N tấm ván hình chữ nhật, tấm thứ i có chiều rộng là 1 và chiều dài là a_i đơn vị. Pirate đang đứng trên mặt đất và dán lại với nhau các mảnh ván to hơn (xem hình minh họa).



Vì c cụ i cùng ch là em m nh ván này i c a thành t m bi n thôi. Nh ng hóa ra ây l i là công vì c khó kh n nh t. Pirate r t thích hình vuông và mu n t m bi n c a mình càng to càng t t, nh ng kh n i trên o l i không có nhi u d ng c o c. Không êke, không th c o , nên Pirate ch còn cách đ a vào c nh c a N t m vấ n ban u c a cho th ng thôi. Pirate ch có th c a theo nh ng o n th ng ch a m t c nh nào ó (đ c ho c ngang) c a các t m vấ n.

Hãy giúp anh y c a c t m bi n l n nh t có th .

Input

- Dòng th nh t: ghi s nguyên N - s t m vấ n.
- N dòng ti p theo: mô t cao c a các t m vấ n theo th t trái sang ph i sau khi ã dán l i.

Output

- M t s nguyên duy nh t là dài c nh c a t m bi n l n nh t có th c a c.

Gi i h n

- cao c a các t m vấ n là các s nguyên đ ng không v t quá 10^9 .
- $1 \leq N \leq 10^6$.
- 60% s test có $1 \leq N \leq 2000$.
- 80% s test có $1 \leq N \leq 10^5$.

Example

Input:

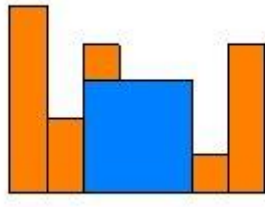
7
5
2

4
3
3
1
4

Output:

3

Ghi thích: Hình dưới đây minh họa phần nhập dữ liệu.



Ghi ý: duyệt các phần tử $1 \rightarrow n$, với phần tử i , tìm phần tử ưu tiên bên trái thấp hơn nó, tìm phần tử ưu tiên bên phải thấp hơn nó \rightarrow ta tìm kiếm hình chữ nhật có chiều dài giới hạn bởi 2 phần tử tìm được, có chiều cao = chiều cao của phần tử $i \Rightarrow$ tìm hình vuông có cạnh = $\min(\text{chiều dài}, \text{chiều rộng}) \Rightarrow$ cập nhật vào kết quả.

Bài 4: MPYRAMID (IOI-2006-MEXICO)

<http://vn.spoj.com/problems/MPYRAMID/>

Sau khi nhận thấy vua Jaguar muốn xây dựng một tháp để kỷ niệm chiến thắng và tôn thờ các chiến sĩ anh hùng. Tháp sẽ được xây dựng trên một mặt hình chữ nhật với các ô vuông bao gồm các cột và hàng. Bên trong tháp sẽ xây một tầng để kỷ niệm các ngôi sao là hình chữ nhật với kích thước cột và hàng.

Các nhà kiến trúc sư của nhà vua đã tìm ra vị trí cần xây tầng để là một hình chữ nhật bao gồm các hàng và cột, và tìm vị trí ô trên lối đi cao là một ngôi sao.

Tháp và tầng để kỷ niệm sẽ được xây dựng trên lối đi ô vuông với các ngôi sao song song với khung hình chữ nhật. Sau khi xây dựng xong vị trí các ô trong tầng để kỷ niệm sẽ được ghi nguyên nhúng tại các ô còn lại của tháp, tầng sẽ san bằng các ô cao xuống các ô thấp hơn nên một vùng bằng phẳng, cao này bằng trung bình cộng của các ô cao các ô sẽ san lấp. Các kiến trúc sư quy định vị trí xây

đ ng tháp và v trí xây t ng ài trong tháp v i i u ki n trong tháp xung quanh t ng ài ph i có t i thi u không gian là m t ô xung quanh t ng ài.

B n hãy giúp các ki n trúc s tìm ra v trí xây đ ng tháp và v trí t ng ài sao cho sau khi san l p, m t ph ng c a tháp có cao l n nh t có th c.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|----|----|---|---|---|----|---|
| 1 | 1 | 5 | 10 | 3 | 7 | 1 | 2 | 5 |
| 2 | 6 | 12 | 4 | 4 | 3 | 3 | 1 | 5 |
| 3 | 2 | 4 | 3 | 1 | 6 | 6 | 19 | 8 |
| 4 | 1 | 1 | 1 | 3 | 4 | 2 | 4 | 5 |
| 5 | 6 | 6 | 3 | 3 | 3 | 2 | 2 | 2 |

Trong hình nh trên, các s trong các ô là cao c a ô này. Các ô màu xám là vùng tháp c san l p xung quanh t ng ài. Ví d trên ch ra v trí t i u ã tìm c.

Yêu c u

Hãy vi t ch ng trình, cho tr c kích th c m nh t, cao các ô, kíc h th c tháp và t ng ài, ch ra v trí tháp và t ng ài v i cao l n nh t có th c cho tháp sau khi san l p.

Ràng bu c

3 m,n 1000

3 a m;3 b n

1 c a – 2; 1 d b – 2

Các cao là s nguyên trong kho ng t 1 n 100.

INPUT

Ch ng trình c n c d li u có đ ng sau:

| | |
|--|------|
| | Mô t |
|--|------|

| | |
|---|---|
| 8 5 5 3 2 1 1 5 10 3 7 1 2 5 6 12 4 4 3 3 1 5 2 4 3 1 6 6 19 8 1 1 1 3 4 2 4 5 6 6 3 3 3 2 2 2 | Dòng 1: Ch a 6 s t nhiên cách nhau b i d u cách là các s m, n, a, b, c, d . <i>n dòng ti p theo:</i> M i dòng ch a m s t nhiên cách nhau b i d u cách là cao các ô trong khung ch nh t. Dòng u tiên ch ra các cao c a hàng u tiên. Dòng cu i cùng ch ra các cao c a hàng th n. |
|---|---|

OUTPUT

Ch ng trình c n a ra k t qu có d ng sau:

| | Mô t |
|------------|---|
| 4 1 6 2 | Dòng 1: Ch a 2 s t nhiên ch ra v trí trái trên c a tháp. Dòng 2: Ch a 2 s t nhiên ch ra v trí trái trên c a t ng ài li t s . tháp. |

Chú ý: N u có nhi u ph ng án t i u thì ch c n ch ra m t trong chúng.

Cách ch m

V i m t s các Test có t ng i m s 30 và các ràng bu c th a mẫn:

3 m 10 ; 3 n 10

G i ý: - Xây d ng m ng c ng d n trên m ng 2 chi u có th tính c t ng các s c a l hình ch nh t b t k trong hình ch nh t l n(cái này các b n t suy ngh , áp d ng c cho nhi u bài, và l là thao tác c b n).

- Duy t t t c các hình ch nh t có th dùng xây d ng tháp. V i m i kh n ng duy t ó, ta c n tìm hcn(xây d ng t ng ài) có t ng nh nh t n m trong hình ch nh t ang duy t n thì khi ó cao c a m t ph ng tháp s l n nh t \Rightarrow c p nh t k t qu . có th làm c i u ó, ta dùng 2 m ng deque qu n lý hàng d c và hàng ngang. M ng ngang qu n lý min n m trên 1 hàng c a hình ch nh t xây d ng tháp, m ng d c qu n lý “min c a các min c a các hàng ngang”.

Bài 5: NUMTAB

(Ngu n: ch n i tuy n qu c t n m 2011-2012)

Gi s A là l i ô vuông g m m dòng và n c t. Các dòng c a l i c ánh s t l n m, t trên xu ng d i. Các c t c a l i c ánh s t l n n, t trái sang ph i. Ô n m trên giao c a dòng i và c t j c a l i g i là ô (i, j) .

V i s nguyên d ng x , g i $f(x)$ là s l ng s nguyên d ng không v t quá x mà trong bi u di n nh phân có hai bít l ng c nh nhau. Ví d , $f(5)=1$ vì trong các s

nguyên dương bé hơn hoặc bằng 5 chỉ có số 3 có biểu diễn như phân vị hai bit 1
ngay cạnh nhau.

Cho dãy số nguyên dương gồm $m \times n$ số $b_1, b_2, \dots, b_{m \times n}$. Ta sắp xếp tăng dần các số hạng
của dãy

$$f(b_1) \bmod 3, f(b_2) \bmod 3, \dots, f(b_{m \times n}) \bmod 3$$

vào các ô của bảng A theo thứ tự trên xuống rồi trái qua phải. Gọi bảng số thu
được là B .

Xét truy vấn sau đây về bảng số thu được B : Cho hai số nguyên p và q ($1 \leq p \leq q \leq m$), hãy cho biết tích lũy của hình chữ nhật gồm các ô nằm trong phạm vi
từ dòng thứ p đến dòng thứ q của bảng B mà trong đó chênh lệch giữa giá trị phần tử nằm
nhỏ hơn và phần tử nằm lớn hơn không vượt quá 1.

Yêu cầu: Cho m, n , dãy số $b_1, b_2, \dots, b_{m \times n}$ và k bộ p_i, q_i ($i = 1, 2, \dots, k$) truy vấn về giá trị
truy vấn, hãy trả ra các câu trả lời cho k truy vấn.

Dữ liệu: Vào tệp file văn bản NUMTAB.INP trong đó:

- Dòng đầu tiên chứa hai số nguyên m, n ($1 \leq m, n \leq 1000$);
- Dòng tiếp theo chứa dãy số $b_1, b_2, \dots, b_{m \times n}$ (mỗi số không vượt quá 10^9);
- Dòng tiếp theo chứa số nguyên k ($1 \leq k \leq 10^6$);
- Dòng thứ i trong số k dòng tiếp theo chứa 2 số nguyên p_i và q_i ($i = 1, 2, \dots, k$).

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

Kết quả: Ghi ra file văn bản NUMTAB.OUT gồm k dòng, mỗi dòng chứa một số là
câu trả lời cho truy vấn theo thứ tự xuất hiện trong dữ liệu vào.

Ví dụ:

| NUMTAB.INP | NUMTAB.OUT |
|-------------------|------------|
| 3 3 | 3 |
| 3 8 7 6 3 2 4 6 6 | 4 |
| 4 | 4 |
| 1 1 | 3 |

| | |
|-----|--|
| 1 2 | |
| 1 3 | |
| 3 3 | |

H n ch : Có 50% s test ng v i 50% s i m c a bài v i $b_i \leq 10, \forall i$.

G i ý: ây là 1 bài toán hay, K t h p duy t+qh +deque. Dành cho các b n t tìm tòi, nghiên c u .(b test trong file kèm theo sách).

Bài 6: QBSQUARE (Ngu n : <http://vn.spoj.com/problems/QBSQUARE/>)

Cho m t b ng kích th c $M \times N$, c chia thành l i ô vuông n v M dòng N c t ($1 \leq M, N \leq 1000$)

Trên các ô c a b ng ghi s 0 ho c 1. Các dòng c a b ng c ánh s 1, 2... M theo th t t trên xu ng đ i và các c t c a b ng c ánh s 1, 2..., N theo th t t trái qua ph i

Hãy tìm m t hình vuông g m các ô c a b ng tho măn các i u ki n sau:

1 - Hình vuông là ng nh t: t c là các ô thu c hình vuông ó ph i ghi các s g i ng nhau (0 ho c 1)

2 - C nh hình vuông song song v i c nh b ng.

3 - Kích th c hình vuông là l n nh t có th

Input

Dòng 1: Ghi hai s m, n

M dòng ti p theo, dòng th i ghi N s mà s th j là s ghi trên ô (i, j) c a b ng

Output

G m 1 dòng duy nh t ghi kích th c c nh c a hình vuông tìm c

Example

Input:

```
11 13
0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0
```

```

0011111110000
0011111110000
0111111111000
1111111111100
0111111111000
0011111110000
0011111110000
0000111000011
0000010000011

```

Output:

7

G i ý: V i m i hàng tính di n tích hình vuông ng nh t l n nh t có c nh d i n m úng trên hàng ang xét (a v bài Bán d a KPLANK trên).

Bài 7:QBRECT <http://vn.spoj.com/problems/QBRECT/>

Cho m t b ng kích th c $M \times N$, c chia thành l i ô vuông n v M dòng N c t ($1 \leq M, N \leq 1000$). Trên các ô c a b ng ghi s 0 ho c 1. Các dòng c a b ng c ánh s 1, 2... M theo th t t trên xu ng đ i và các c t c a b ng c ánh s 1, 2..., N theo th t t trái qua ph i

Yêu c u:

Hãy tìm m t hình ch nh t g m các ô c a b ng tho măn các i u ki n sau:

- 1 - Hình ch nh t ó ch g m các s 1
- 2 - C nh hình ch nh t song song v i c nh b ng
- 3 - Di n tích hình ch nh t là l n nh t có th

Input

Dòng 1: Ghi hai s M, N

M dòng ti p theo, dòng th i ghi N s mà s th j là s ghi trên ô (i, j) c a b ng

Output

G m 1 dòng duy nh t ghi di n tích c a hình ch nh t tìm c

Example**Input:**

Một số thuật toán và cấu trúc dữ liệu nâng cao

```

11 13
0000010000000
0000111000000
0011111110000
0011111110000
0111111111000
1111111111100
0111111111000
0011111110000
0011111110000
0000111000011
0000010000011

```

Output: 49

Bài 8: CREC01 <http://vn.spoj.com/problems/CREC01/>

Cho một bảng ô vuông kích thước $M \times N$. Mỗi ô chứa một số 0 hoặc 1. Hãy tìm hình chữ nhật con chứa nhiều nhất các cặp song song với các cạnh và gồm toàn số 1.

Gợi ý: Đây là một bài cơ bản, cách làm tương tự bài [QBRECT](#).

Sau khi giải bài toán này, bạn có thể tham khảo bài [CRECT](#).

Input

Dòng đầu chứa hai số nguyên M, N . ($1 \leq M, N \leq 1000$)

M dòng sau, mỗi dòng chứa N kí tự 0/1.

Output

In ra số lượng hình chữ nhật thỏa mãn.

Example

Input:

```

4 3
111
101
111
001

```


Output:

24

G i ý: V i m i hàng m s hình ch nh t toàn l có c nh d i n m úng trên hàng ang xét. Ta tính b ng quy ho ch ng. G i $H[i]$ là cao l n nh t toàn l **liên ti p** c t i tính t **hàng ang xét** i lên. G i $L[i]$ là v trí c a s n m bên trái i và bé h n $H[i]$, $F[i]$ là s hình ch nh t toàn l có c nh d i n m úng trên hàng ang xét và c nh ph i n m úng trên c t i. D th y $F[i]=F[L[i]]+(i-L[i])*H[i]$ (=s hình ch nh t có c nh trái n m v trí $\leq L[i]$ +s hình ch nh t có c nh trái n m v trí $>L[i]$).

Bài 9: CRECT <http://vn.spoj.com/problems/CRECT/>

Hùng M ang theo h c m t khóa h c c b n v t i ng c. V i m i b t u, Hùng M m i b i t m t 5 ch cái là A, B, C, D, E. Ngày sinh nh t, Hùng M c t ng m t b ng hình ch nh t có các ch cái ghi các ô. Nhi m v c a Hùng M s là tìm các t n trong b ng này. Tuy nhiên, do v n l i h c ngo i ng , am mê l p trình, Hùng M l i ngh ra m t trò ch i khác: m s hình ch nh t con c a b ng này có ch a úng 3 ch cái khác nhau(V i Hùng M không thích quá ít, c ng ch ng a quá nhi u). Tuy nhiên, bài này không n gi n Hùng M có th gi i c đ dàng. Các b n hãy giúp Hùng M Hùng M có th nhanh chóng t p trung vào v i c h c t i ng c.

Input

Dòng u ghi 2 s M, N ($M, N \leq 400$). B ng ch c a Hùng M c chia làm M dòng, m i dòng g m N ô vuông n v . M dòng sau, m i dòng là m t xâu dài N th hi n m t dòng c a b ng ch ch g m các ch cái A, B, C, D, E.

Output

G m m t s duy nh t là s hình ch nh t con tìm c.

Example**Input:**

4 3

CED

CEB

CBC

DDA

Output:

12

G i ý: Tính t ng t h p 3 ch cái m t. N u g i $F(xy)$ là s hình ch nh t ch ch a x +ch ch a y+ch ch a x và y, t ng t m r ng v i khái ni m $F(x), F(xyz)$ thì s hình ch nh t ch a **úng** 3 ch cái A,B,C là $F(A)+F(B)+F(C)-F(AB)-F(BC)-F(CA)+F(ABC)$ (Nguyên lí bù tr -bài c thêm Toán 11).

Bài 10: S n t ng

(ngu n: COCI 2009-2010 contest 4 <http://vn.spoj.com/problems/DTOGRADA/>)

Matija mu n s n l i hàng rào nhà mình. Hàng rào c a anh y c ghép b i N t m ván liên ti p, m i t m r ng 1 cm và có chi u cao khác nhau. s n nhanh chóng và d dàng h n, anh y ã mua "Super Pain Roller Deluxe" (m t cây l n x n). Cây l n có chi u r ng x cm. M i l n dùng, anh y ph i cây l n ch m vào b c t ng hoàn toàn, n u không s không s n c gì c . M t khác, lúc nào c ng ph i cho cây l n song song v i m t t. Nói cách khác, anh y s ch n ra x t m ván liên ti p và s n t đ i lên n cao c a t m ván th p nh t trong x t m ván ó.

Tuy nhiên, cây l n không th giúp Matija s n h t c hàng rào. Ph n di n tích còn l i không th s n b ng cây l n, anh y s s n b ng c nh . Hãy giúp anh y tính xem ph n di n tích ph i s n b ng c nh nh t có th là bao nhiêu, và s l n dùng cây l n ít nh t t c i u ó.

Input

- Dòng u ghi s nguyên N ($1 \leq N \leq 10^6$) là s t m ván c n ph i s n và s nguyên X ($1 \leq X \leq 10^5$).
- Dòng ti p theo ghi N s nguyên d ng (m i s không quá 10^6) mô t chi u cao c a t ng t m ván.

Output

- Dòng th nh t ghi di n tích nh nh t ph i s n b ng c .
- Dòng th hai ghi s l n dùng cây l n ít nh t t c di n tích ó.

Example

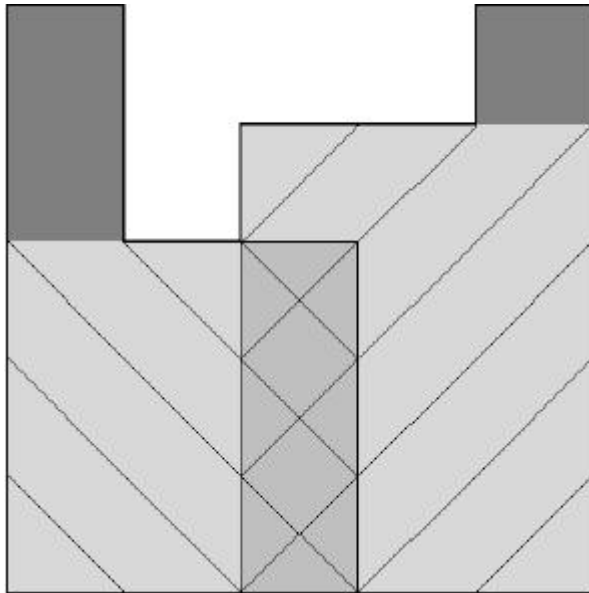
Input:

5 3

5 3 4 4 5

Output:

3
2

**Bài 11: Vùng bị n.**

(Ngu n: <http://vn.spoj.com/problems/LSEA/>)

Lulu là tên c a m t chú h i c u. Chú ta cai qu n m t vùng bị n r ng. Vùng bị n y bao g m o (**ph n c n**), ao, h , các con sông (**ph n n c** trên t li n) và bị n xung quanh các o (**ph n n c** d i bị n). t i n vì c qu n lí, Lulu ã v b n cho vùng bị n c a mình. B n có th xem nh m t nh hình ch nh t có kích th c các c nh là M , N và bao g m $M \times N$ bit tr ng en. M i bit s t ng ng v i m t **m nh t n v** trên th c t . Bit en s t ng tr ng cho ph n c n và bit tr ng s t ng tr ng cho ph n n c.

Vùng bị n c a Lulu v a m i c UNESCO công nh n là Di s n t nhiên c a Th gi i. Nên các nhà u t ào t t i ây xây các Resort, khu ngh mát, ngh d ng, v.v... H s mua nh ng m nh t có **hình ch nh t**. Và vì c s d ng xây các khu du l ch bị n, nên m nh t ó ph i có c **ph n n c** và **ph n c n**. H g i nh ng m nh t nh th là nh ng **m nh t p**.

Sau khi bi t i u này, Lulu s bán các m nh t c a mình nh sau:

- Các m nh t c bán ph i n m *khít* trên b n .

- Không chấp nhận việc chia nhỏ một mảnh để bán.
- Giá bán của một mảnh để bán là số mảnh để bán mà nó nằm trên.
- Giá bán của một mảnh để bán tối thiểu là tổng giá bán của các mảnh để bán của nó.

Alex là một nhà đầu tư trên thị trường. Anh đã quy định mua hàng từ vùng biên $M \times N$ của Lulu. Vì có khả năng giá của 1 mảnh để bán hay 2 mảnh để bán hay 3 mảnh để bán là chuyển đổi với Lulu. Nhưng giá trị lại phải cùng lúc khả năng số lượng là $M \times N$ mảnh để bán, đó là một điều không thể nào kết thúc trong nháy mắt.

Bây giờ bạn là người cần thực hiện tính toán để biết Alex phải trả.

Giới hạn

- 1 ≤ M, N ≤ 2000.
- 30% số test có M, N ≤ 20.
- 50% số test có M, N ≤ 100.

Input

- Dòng đầu tiên 2 số M, N cho biết kích thước của vùng biên Lulu sẽ mua.
- M dòng sau, mỗi dòng một dãy gồm N bit (0 hoặc 1) cho biết biên của vùng biên (0 là trống, 1 là đất).

Output

- Một số duy nhất là kết quả của bài toán.

Example

Input:

2 2
01
01

Output:

8

Ghi ý: Dãy kết quả = Tổng diện tích các hình chữ nhật nằm trong bảng $M \times N$
- Tổng diện tích các hình chữ nhật chỉ chứa toàn 0 hoặc toàn 1.

Bài 12: Hình chữ nhật hoàn hảo.

(Nguồn: <http://vn.spoj.com/problems/PERREC>)

Cho 1 bảng kích thước $N * N$ chia thành các ô vuông nhỏ. Mỗi ô vuông có thể có màu đen hoặc trắng. Bây giờ, chúng ta 1 hình chình t t t là 1 hình chình t t t có các cạnh song song với cạnh của bảng và chia các ô vuông màu trắng. 1 hình chình t t t có nghĩa là hoàn hảo, nếu nó là 1 hình chình t t t, và không tồn tại 1 hình chình t t t nào khác chia nó (tức không thể rẽ hình chình t này sang trái, phải, trên hay dưới).

Yêu cầu: Xác định số hình chình t hoàn hảo của bảng đã cho.

Lưu ý:

giới hạn kích thước của input, bảng số màu theo quy tắc sau:

- Bảng chia thành các ô vuông màu trắng

- Sinh 2 dãy số X và Y dài m theo quy tắc

+ $X[0] = x_0 \bmod N, Y[0] = y_0 \bmod N$

+ $X[i] = (X[i-1] * a + b) \bmod N, Y[i] = (Y[i-1] * c + d) \bmod N$ với $1 \leq i < m$

trong đó x_0, y_0, a, b, c, d, m là các số cho trước, và $P \bmod Q$ ký hiệu là phép chia P cho Q

- Tập hợp các ô có tọa độ $(X[0], Y[0]), (X[1], Y[1]), \dots, (X[m-1], Y[m-1])$. (Tập của bảng chia thành các ô có tọa độ từ 0 đến $N-1$ theo thứ tự trái qua phải, và từ trên xuống dưới)

Input:

- 1 dòng duy nhất gồm 8 số nguyên $N, m, x_0, a, b, y_0, c, d$ như mô tả trong bài

Output:

- 1 dòng duy nhất ghi ra số lượng hình chình t hoàn hảo thu được

Giới hạn:

- $0 < N \leq 2000$

- $1 \leq m \leq 4000000$

- $0 \leq a, b, c, d, x_0, y_0 \leq 2000$
- Time limit: 5s

Example:

| Input | Output |
|---------------------|--------|
| 5 1 2 0 0 2 0 0 | 4 |
| 4 4 0 1 1 0 1 1 | 6 |
| 10 20 4 76 2 6 2 43 | 12 |

G i ý: T t ñg c a ta v n là xét m i hàng nh các bài trên.

Ch ñng IV

BÀI TOÁN LCA

I/ Phân tích v n

Bài toán LCA là bài toán tìm t iên chung g n nh t c a 2 ñnh u, v trong m t cây. Th ñng thì công vi c này s giúp l y c nh ñng thông tin c n thi t trong ñng i t u n v (là ñng i duy nh t vì th ñang xét là cây). Có nhi u thu t toán gi i bài toán trên, ñây ta xét cách gi i d a trên RMQ.

Ta có m t m ñng cha[i, j] = t iên cách ñnh i 2^j c ñnh (n u ta ch ñn tr c m t ñnh làm g c). Cha[i, 0] kh i t o trong quá trình duy t cây, chính là ñnh cha tr c ti p c a ñnh i (t iên cách ñnh i m t c ñnh chính là cha tr c ti p).

Các cha[i, j] khác kh i t o b ñng công th c cha[i, j] := cha[cha[i, j - 1], j - 1].

Thêm m t m ñng h v i h[i] b ñng cao c a ñnh i so v i g c. M ñng h c tính khi duy t cây.

Ph n chu n b ã xong.

Bây gi v i m i truy v n u, v. Gi s ta có $h[u] > h[v]$ ($h[v]$ l n h n làm t ng t).
u tiên ta a u v t tiên c a u sao cho $h[u] = h[v]$. n ây x y ra hai tr ng h p.

- N u $u = v$. Bài toán gi i quy t xong. Xu t ra u.
- N u $u \neq v$. G i A là t tiên chung g n nh t c a u và v. Ta s a u và v v hai nh là con tr c ti p c a A, mà m i nh là t tiên c a u và v. Sau ó xu t ra $cha[u, 0]$.

Sau ây là o n các th t c:

```

procedure duyett(v:longint);
var t:node;
begin
    tham[v]:=true;t:=ke[v];
    while t<>nil do
        begin
            if tham[t^.v]=false then
                begin
                    h[t^.v]:=h[v]+1;
                    cha[t^.v,0]:=v;
                    duyett(t^.v);
                end;
            t:=t^.next;
        end;
    end;
procedure init;
begin
    m:=trunc(ln(n)/ln(2))+1;
    for i:=1 to m do
        for j:=1 to n do
            cha[j,i]:=cha[cha[j,i-1],i-1];
        lt2[0]:=1;
        for i:=1 to m do lt2[i]:=lt2[i-1]*2;
    end;
procedure nhay(var v:longint;x:longint);
var i:longint;
begin
    for i:=m downto 0 do
        if x>=lt2[i] then
            begin
                v:=cha[v,i];
            end;
end;

```

```

        x:=x-lt2[i];
    end;
end;
function lca(u,v:longint): longint;
begin
    if h[u]>h[v] then nhay(u,h[u]-h[v]) else nhay(v,h[v]-
h[u]);
    if u=v then exit(u);
    for i:=m downto 0 do
        if cha[u,i]<>cha[v,i] then
            begin
                u:=cha[u,i];
                v:=cha[v,i];
            end;
    exit(cha[u,0]);
end;
procedure duyet(v:longint);
var t:node;
begin
    tham[v]:=true;
    t:=ke[v];
    while t<>nil do
        begin
            if tham[t^.v]=false then
                begin
                    h[t^.v]:=h[v]+1;
                    cha[t^.v,0]:=v;
                    duyet(t^.v);
                end;
            t:=t^.next;
        end;
end;
procedure init;
begin
    m:=trunc(ln(n)/ln(2))+1;
    for i:=1 to m do
        for j:=1 to n do
            cha[j,i]:=cha[cha[j,i-1],i-1];
        lt2[0]:=1;
        for i:=1 to m do lt2[i]:=lt2[i-1]*2;
    end;
procedure nhay(var v:longint;x:longint);
var i:longint;

```



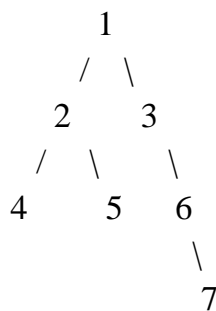
```

begin
  for i:=m downto 0 do
    if x>=lt2[i] then
      begin
        v:=cha[v,i];
        x:=x-lt2[i];
      end;
    end;
  end;
function lca(u,v:longint): longint;
begin
  if h[u]>h[v] then nhay(u,h[u]-h[v]) else nhay(v,h[v]-h[u]);
  if u=v then exit(u);
  for i:=m downto 0 do
    if cha[u,i]<>cha[v,i] then
      begin
        u:=cha[u,i];
        v:=cha[v,i];
      end;
  end;
  exit(cha[u,0]);
end;

```

Mảng $lt2$ là các lũy thừa của 2. Hàm nhẩy làm công việc đưa u và v về cùng cao. Các bạn hãy thử suy nghĩ gì thích đây tại sao $cha[u,i] \neq cha[v,i]$ thì mới nhẩy u và v lên nhé.

Xét ví dụ sau:



Giả sử cây trên như 1 là gốc cây. Thì thì ta có $cha[2,0] = 1$, $cha[5,1] = 1$, $cha[7,1] = 3$, $cha[6,2] = 0$, vv... và $h[1] = 0$, $h[2] = 1$, $h[6] = 2$, $h[7] = 3$, vv...

Giả sử đang cần tìm $LCA(5, 7)$. Thì thì lúc $u = 5$ còn $h[5] = 2$ còn $h[7] = 3$. Đưa vào thuật toán nhẩy ta đưa 7 về 6 . Bây giờ $u = 5$ và $v = 6$. Tiếp tục thì chắc hẳn nhẩy u và v thì ta

chính y khi mà $\text{cha}[u, i] \leq \text{cha}[v, i]$ nên $s[u, v] = 2$ và $v = 3$. ($\text{cha}[5, 0]$ và $\text{cha}[6, 0]$). Kết quả xuất ra là $\text{cha}[u, 0] = 1$.

II/Bài tập luyện:

Bài 1 :LUBENICA

(Nguồn <http://vn.spoj.com/problems/LUBENICA/>)

Mạng là giao thông liên lạc bao gồm N thành phố (ánh xạ từ 1 đến N) và $N-1$ đường nối các thành phố với nhau. Có một đường đi duy nhất giữa mọi cặp thành phố. Mọi con đường có một chiều xác định.

Viết chương trình, với mỗi K cặp thành phố cho trước, tìm chiều dài của con đường ngắn nhất và dài nhất trên đường đi giữa 2 thành phố này.

Dữ liệu

Dòng đầu tiên chứa số nguyên N , $2 \leq N \leq 100\,000$.

Mỗi dòng trong số $N-1$ dòng tiếp theo chứa 3 số nguyên A, B, C cho biết có một con đường chiều C giữa thành phố A và thành phố B . Chiều dài của mọi con đường là số nguyên dương không vượt quá 1000000.

Dòng tiếp theo chứa số nguyên K , $1 \leq K \leq 100\,000$.

Mỗi dòng trong số K dòng tiếp theo chứa 2 số nguyên D và E - chỉ số của 2 thành phố cần truy vấn.

Kết quả

Mỗi dòng trong số K dòng chứa 2 số nguyên - chiều dài của con đường ngắn nhất và dài nhất trên đường đi giữa 2 thành phố tương ứng.

Ví dụ

Dữ liệu:

```
5
2 3 100
4 3 200
1 5 150
1 3 50
3
2 4
3 5
```

1 2

K t q a

100 200

50 150

50 100

D li u:

7

3 6 4

1 7 1

1 3 2

1 2 6

2 5 4

2 4 4

5

6 4

7 6

1 2

1 3

3 5

K t q a

2 6

1 4

6 6

2 2

2 6

D li u:

9

1 2 2

2 3 1

3 4 5

2 7 4

1 5 3

5 6 1

5 9 2

1 8 3

5

6 9

7 8

9 4

1 2
7 3

K t q a

1 2
2 4
1 5
2 2
1 4

G i ý: Bài toán này có tr ng s nh ng t t ng gi i quy t v n nh bài toán t v n . ây ngoài m ng cha[i,j] qu n lý cha c a các nút thì ta s d ng thêm 2 m ng mi[i,j] và ma[i,j] v i ý ngh a: mi[i,j] là c nh nh nh t trong các c nh liên t c t i-> cha[i,j], ma[i,j] là c nh l n nh t trong các c nh liên t c t i lên cha[i,j].

Code tham kh o:

```
{ $r-,l-,q- }
{ $m 1000000000 }
Uses math;
Const fi='';
fo='';
maxn=100001;
type node=^re;
re=record
    v,w:longint;
    next:node;
end;
var f,g: text;
    n,m,kq1,kq2:longint;
    lt:array[0..21] of longint;
    ke:array[1..maxn] of node;
    cha,mi,ma:array[0..maxn,0..21] of longint;
    h:array[1..maxn] of longint;
    tham:array[1..maxn] of boolean;
procedure mo;
begin
    assign(f,fi);
    reset(F);
    assign(g,fo);
    rewrite(G);
end;
procedure dong;
begin
```

```

        close(F);
        close(G);
    end;
    procedure nap(var ke:node; v,w:longint);
    var p:node;
    begin
        new(p);
        p^.v:=v;
        p^.w:=w;
        p^.next:=ke;
        ke:=p;
    end;
    procedure nhap;
    var i,u,v,w:longint;
    begin
        readln(F,n);
        for i:=1 to n-1 do
            begin
                readln(F,u,v,w);
                nap(ke[u],v,w);
                nap(ke[v],u,w);
            end;
        end;
    procedure dfs(u:longint);
    var v:longint;
    P:node;
    begin
        tham[u]:=true;
        p:=ke[u];
        While p<>nil do
            begin
                v:=p^.v;
                if not tham[v] then
                    begin
                        h[v]:=h[u]+1;
                        cha[v,0]:=u;
                        mi[v,0]:=p^.w;
                        ma[v,0]:=mi[v,0];
                        dfs(v);
                    end;
                p:=p^.next;
            end;
        end;
    end;
    procedure init;

```

```

var i,j:longint;
begin
    fillchar(tham,sizeof(tham),false);
    fillchar(mi,sizeof(mi),0);
    fillchar(ma,sizeof(ma),0);
    for i:=1 to n do if not tham[i] then
        dfs(i);
    m:=trunc(ln(n)/ln(2))+1;
    lt[0]:=1;
    for i:=1 to m do lt[i]:=lt[i-1]*2;

    for j:=1 to m do
    for i:=1 to n do
    begin
        cha[i,j]:=cha[cha[i,j-1],j-1];
        mi[i,j]:=min(mi[cha[i,j-1],j-1],mi[i,j-1]);
        ma[i,j]:=max(ma[cha[i,j-1],j-1],ma[i,j-1]);
    end;
end;
procedure nhay(var u:longint; x:longint);
var i:longint;
begin
    for i:=m downto 0 do
    if x>=lt[i] then
    begin
        kq1:=min(kq1,mi[u,i]);
        kq2:=max(kq2,ma[u,i]);
        u:=cha[u,i];
        x:=x-lt[i];
    end;
end;
procedure lca(u,v:longint);
var i:longint;
begin
    if h[u]>h[v] then nhay(u,h[u]-h[v]) else
    nhay(v,h[v]-h[u]);
    if u=v then exit;
    for i:=m downto 0 do
    if cha[u,i]<>cha[v,i] then
    begin
        kq1:=min(kq1,min(mi[u,i],mi[v,i]));
        kq2:=max(kq2,max(ma[u,i],ma[v,i]));
        u:=cha[u,i];
        v:=cha[v,i];
    end;
end;

```

```

        end;
        kq1:=min(kq1,min(mi[u,0],mi[v,0]));
        kq2:=max(kq2,max(ma[u,0],ma[v,0]));
    end;
    procedure solve;
    var que,u,v,i:longint;
    begin
        readln(f,que);
        for i:=1 to que do
            begin
                readln(F,u,v);
                kq1:=maxlongint;
                kq2:=0;
                lca(u,v);
                writeln(g,kq1,' ',kq2);
            end;
        end;
    begin
        mo;
        nhap;
        init;
        solve;
        dong;
    end.

```

Bài 2:FSELECT

(Ngu n: Croatia OI 2006 <http://vn.spoj.com/problems/FSELECT/>)

Sau khi tham d IOI và OLPSV, Nguyên chuy n n m t ngôi nhà m i. Khu nhà m i c a Nguyên có N ng i b n hàng xóm ($N \leq 200000$). Vì d b nh m nên Nguyên ánh s các b n y t l n N . Gi a các ngôi nhà có ng i t o thành cây. Kho ng cách gi a hai c n nhà k nhau là $1 \leq n \leq v$. Có K cu c h n ($K \leq N/2$) c Nguyên a ra làm quen v i các b n m i. tính toán chi phí m i các b n, Nguyên mu n bi t xem kho ng cách xa nh t c a 2 ngôi nhà trong m t cu c h n là bao nhiêu ? B n hãy giúp Nguyên gi i quy t v n này.

Input

- Dòng 1 g m 2 s N và K .

- N dòng ti p theo, dòng th i g m 2 s x y. Trong ó x là th t c a cu c h n mà b n th i tham gia, y là nhà hàng xóm c a b n th i. N u $y = 0$ thì ó là g c c a khu dân c (có th hi u là g c c a cây).

Output

G m **K** dòng, dòng th i th hi n ng i xa nh t tìm c gi a 2 ngôi nhà c a 2 ng i b n trong cu c h n th i.

Example**Input:**

```
6 2
1 3
2 1
1 0
2 1
2 1
1 5
```

Output:

```
3
2
```

Gi i thích :

```
-3-
|
-1-
/|\
2 4 5
|
-6-
```

Trong cu c h n th 1 g m 3 b n là b n s 1, s 3 và s 6. Kho ng cách xa nh t gi a 2 ngôi nhà trong cu c h n

th 1 là 3 (gi a nhà b n s 3 và s 6). T ng t , cu c h n th 2 g m 3 b n s 2, s 4 và s 5, kho ng cách xa nh t là 2.

G i ý: T ng t bài trên, thay vì qu n lý min thì ta qu n lý t ng.

Bài 3: UPGRANET

(Ngu n : VOI-2011<http://vn.spoj.com/problems/UPGRANET/>)

M t h th ng g m n máy tính ánh s t 1 n n c k t n i thành m t m ng b i m o n cấp m ng ánh s t 1 n m. o n cấp m ng th i có thông l ng w_i k t n i hai máy u_i, v_i cho phép truy n đ li u theo c hai chi u gi a hai máy này.

M t dãy các máy x_1, x_2, \dots, x_p trong ó gi a hai máy x_j và x_{j+1} ($j = 1, 2, \dots, p-1$) có o n cấp n i c g i là m t ng truy n tin t máy x_1 t i máy x_p . Thông l ng c a

ng truy n tin c xác nh nh là thông l ng nh nh t trong s các thông l ng c a các o n cấp m ng trên ng truy n. Gi thi t là m ng c k t n i sao cho có ng truy n tin gi a hai máy b t kì và gi a hai máy có không quá m t o n cấp m ng n i chúng.

Ng i ta mu n nâng c p m ng b ng cách t ng thông l ng c a m t s o n cấp n i trong m ng. t ng thông l ng c a m i o n cấp m ng thêm m t l ng d ($d > 0$) ta ph i tr m t chi phí úng b ng d. Vì c nâng c p m ng ph i m b o là sau khi hoàn t t, thông l ng c a m i o n cấp m ng i u b ng thông l ng c a ng truy n tin có thông l ng l n nh t t máy u_i t i máy v_i .

Yêu c u: Tìm ph ng án nâng c p các o n cấp m ng sao cho t ng chi phí nâng c p là nh nh t.

D li u:

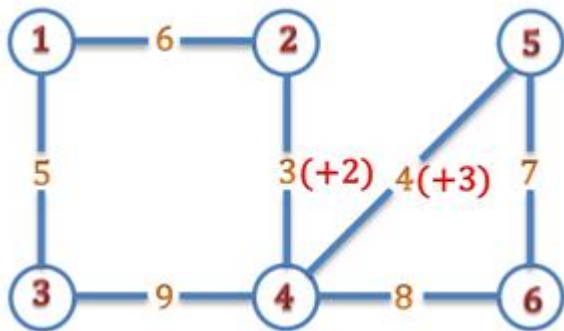
- Dòng th nh t: Ch a hai s nguyên d ng n, m ($n, m \leq 10^5$).
- Dòng th i trong s m dòng ti p theo ch a ba s nguyên d ng u_i, v_i, w_i ($w_i \leq 10^6$),
 $i = 1, 2, \dots, m$.

Các s trên cùng m t dòng c ghi cách nhau ít nh t m t d u cách.

K t qu : ghi ra m t s nguyên duy nh t là t ng chi phí nâng c p theo ph ng án tìm c.

Ví d :

| D li u | | K t qu |
|--------|---|--------|
| 6 | | 75 |
| 1 | 2 | 6 |
| 1 | 3 | 5 |
| 2 | 4 | 3 |
| 3 | 4 | 9 |
| 4 | 5 | 4 |
| 4 | 6 | 8 |
| 5 6 7 | | |



Ràng bu c: 50% s test ng v i 50% s i m c a bài có n ≤ 100 .

G i ý: Dùng KRUSKAL+ LCA

Bài 4: VMQTREE

(Ngu n: <http://vn.spoj.com/VM12/problems/VMQTREE/>)

RR và TA ngoài vi c làm admin VNOI còn có ngh tay trái là ng i m u ch p nh trang bìa. Ngh này giúp ôi b n không nh ng nuôi s ng c b n thân mà còn dành đ m c m t s ti n kha khá. Vào m t ngày p tr i, hai ng i quy t nh i tìm cho mình m t t m. Các b n ng suy ngh l ch l c, h ch mu n mua hai ng i nhà g n nhau ti n b àm o v i nhau v nh n tình th thái, V-Pop, K-Pop và các th t ng t ...

RR và TA tìm mãi m i ra m t khu ph r t c bi t mà hai ng i r t hài lòng. Ngoài N ng i nhà p r ng ng i c ánh s t l n N thì h th ng ng xá c a khu ph này c ng r t áng chú ý. T i ây, m t s c p ngôi nhà c n i v i nhau b i m t con ph có dài m t n v. i u này thì ch ng có gì l , nh ng c bi t là s l ng các con ph đư r t ít nh ng l i c thì t k r t kh o h c. C th là ch có $N - 1$ con ph nh ng c ng hai ngôi nhà b t kì u n c nhau. Mua nhà ây thì không s b l c mà ch ng cho b n bè n ch i c ng ti n, h nh phúc là ây r i! Th là RR và TA h m h i rút ti n mua nhà.

flashmt (v a trúng vé s) c ng nh mua nhà khu ph này. Nh ng bu n m t n i, th y phong th y tr danh technolt (c ng là ngh tay trái thôi) l i phán r ng flashmt và " ôi b n hoàn h o" RR - TA không h p tu i nên anh không nên g n hai ng i này, n u không s nghi p c ng tiêu tan mà tình duyên c ng ngang trái. Nh ng vì là b n thân c a hai ng i, flashmt v n mu n mua nhà ây ti n b h ú hí v i nhau. Th y phong th y c nh báo flashmt r ng an toàn c a flashmt chính b ng kho ng cách t

nhà anh n ngôi nhà g n nh t trong s hai ngôi nhà c a RR và TA. Mu n c bình yên thì d nhiên an toàn ph i càng l n càng t t!

Bì t c v trí hai ngôi nhà c a RR và TA, b n hãy tính xem an toàn l n nh t mà flashmt có th t c là bao nhiêu!

Input

Dòng u ghi s nguyên d ng T - s b test (T 5).

Ti p theo là T test, m i test g m:

- Dòng u tiên ch a s nguyên d ng N.
- Ti p theo là N-1 dòng, m i dòng g m 2 s nguyên d ng u, v cho bì t có c nh n i gi a nh ngôi nhà u và ngôi nhà v.
- Dòng ti p theo ch a s nguyên d ng Q.
- Q dòng ti p theo, m i dòng g m 2 s nguyên d ng u, v mô t v trí hai ngôi nhà c a RR và TA.

Các d u cách và dòng tr ng th a có th xu t hi n b t k v trí nào trong file input.

Gi i h n:

- 1 N, Q 50,000
- Trong 30% s test, N và Q 100

Output

Ouput g m Q dòng. M i dòng in ra m t s nguyên d ng duy nh t là áp án c a truy v n t ng ng.

Example

Input:

```
1
7
1 2
1 3
3 4
3 5
3 6
5 7
7
```

3 7
5 7
4 6
1 2
1 1
3 5
7 2

Output:

2
3
3
3
3
2
3

Bài 5: (b test trong file kèm) LPG

í khái là lúc u th có l nh, có n truy v n, truy v n th í có d ng j , trong ó $1 \leq j \leq i$, v í ý ngh a là thêm nh $i + 1$ vào th, nh này c n í v í nh j . V í m í truy v n, in ra dài ng í dài nh t trên th sau khi thêm nh vào.

Input: LPG.inp

Dòng u tiên là s n ($1 \leq n < 100000$). Sau ó là n dòng miêu t n truy v n

Output: LPG.out

In ra n dòng, m í dòng là k t qu c a l truy v n

Ví d :

| Input | Output |
|-------|--------|
| 4 | 1 |
| 1 | 2 |
| 2 | 2 |
| 2 | 3 |
| 3 | |

G i ý: Ch ng minh r ng n u ng i dài nh t trong cây hi n t i là gi a 2 nh (u,v) thì sau khi thêm 1 nh c vào cây ó thì ng i dài nh t là gi a hai nh (u,v) ho c (u,c) ho c (v,c).

Ch ng V

LU NG C C I TRÊN M NG

I/Phân tích v n

Lu ng c c i trên m ng là m t công c gi i quy t c r t nhi u bài toán, bi t nó là m t l i th trong các kì thi l p trình.

Bài toán: Trong lý thuy t th , *m ng lu ng* là m t th có h ng, trong ó m i c nh có m t thông qua và m t giá tr lu ng. L ng lu ng trên m i c nh không c v t quá thông qua c a c nh ó. L ng lu ng i vào m t nh ph i b ng l ng lu ng i ra kh i nó, tr khi ó là nh ngu n (có nhi u l ng lu ng i ra h n), hay nh ích (có nhi u l ng lu ng i vào h n). M ng lu ng có th dùng mô hình hóa h th ng ng giao thông, dòng ch y c a ch t l ng trong ng, dòng i n trong m ch, hay b t k các bài toán nào t ng t khi có s di chuy n trong m t m ng các nút.

Mô hình các ng d n n c b ng m t m ng lu ng. M i ng n c có ng kính xác nh nên ch cho phép m t l u l ng n c xác nh ch y qua. n i giao i m gi a các ng, l u l ng n c i vào ph i b ng l u l ng n c i ra n u không n c s nhanh chóng b th t thoát. Ta có m t b n n c, hay nh phát, và m t vôi n c, hay nh thu. M t cách tr c quan, giá tr lu ng trên m ng chính là v n t c n c ch y ra t vôi. Lu ng còn có th mô hình s l u chuy n c a ng i hay hàng hóa trên các m ng

giao thông, dòng điện trong hệ thống phân phối,... để vẽ các hệ thống mạng này, lưu ý vào các nút trung gian cần phải bổ sung lưu ý ra khỏi nút đó. Tính chất này cũng giống như luật dòng điện Kirchhoff. Mạng lưu ý còn có ứng dụng trong sinh thái học: mạng lưu ý xuất hiện khi xem xét sự lưu chuyển chất dinh dưỡng và năng lượng giữa các nhóm khác nhau trong một hệ thống sinh thái. Các bài toán gần với loại mạng sinh thái này hoàn toàn khác với trình bày mạng chuyển động hay mạng giao thông.

Bài toán *lưu ý* có thể dựa trên một yêu cầu tìm một lưu ý thích có giá trị lớn nhất trong mạng lưu ý có một đỉnh phát và một đỉnh thu. Bạn hãy thử giải quyết bài toán lưu ý có thể dựa trên mạng: cho một mạng lưu ý, hãy tìm giá trị lưu ý có thể.

Dữ liệu

- Dòng đầu tiên chứa 4 số nguyên dương n, m, s, t , ($2 \leq n \leq 1000$) tương ứng là số đỉnh, số cạnh của đồ thị, chỉ số của đỉnh phát và đỉnh thu.
- m dòng tiếp theo, mỗi dòng có dạng ba số u, v, c cách nhau ít nhất một dấu cách thể hiện có cung u, v trong mạng với khả năng thông qua là c ($1 \leq c \leq 10^6$).

Kết quả

In ra một số duy nhất là giá trị của lưu ý có thể dựa trên mạng.

Ví dụ

Dữ liệu:

```
6 8 1 6
1 2 5
1 3 5
2 4 6
2 5 3
3 4 3
3 5 1
4 6 6
5 6 6
```

Kết quả

```
9
```

Hàng đầu: Đây là code mẫu theo thuật toán Ford-Fulkerson :

```

uses math;
const
    finp='';
    fout='';
    maxn=1001;
    vc=maxlongint;

var
    fi,fo:text;
    f,c:array[0..maxn,0..maxn] of longint;
    ke:array[0..maxn,0..maxn] of boolean;
    tr,q:array[0..maxn] of longint;
    n,m,s,t,kq:longint;
procedure openfile;
begin
    assign(fi,finp);
    reset(fi);
    assign(fo,fout);
    rewrite(fo);
end;
procedure closefile;
begin
    close(fi);
    close(fo);
end;
procedure enter;
var i,u,v:longint;
begin
    readln(fi,n,m,s,t);
    for i:=1 to m do
    begin
        readln(fi,u,v,c[u,v]);
        ke[u,v]:=true; ke[v,u]:=true;
    end;
end;
function timthayduongtangluong:boolean;
var i,j,dau,cuoi:longint;
begin
    fillchar(tr,sizeof(tr),0);
    tr[s]:=1;
    q[1]:=s;
    dau:=1; cuoi:=1;

```

```

        while dau<=cuoi do
        begin
            i:=q[dau]; inc(dau);
            for j:=1 to n do
            if (ke[i,j]) and (tr[j]=0) then
            begin
                if f[j,i]>0 then
begin
inc(cuoi); q[cuoi]:=j;
tr[j]:=-i;

                end else
                if f[i,j]<c[i,j] then
begin
inc(cuoi); q[cuoi]:=j;
tr[j]:=i;

                end;
                if tr[t]<>0 then exit(true);
            end;
        end;
        exit(false);
    end;
procedure tangluong;
var d,i,j:longint;
begin
    d:=vc;
    i:=t;
    while i<>s do
    begin
        j:=tr[i];
        if j>0 then d:=min(d,c[j,i]-f[j,i])
        else d:=min(d,f[i,-j]);
        i:=abs(j);
    end;
    i:=t;
    while i<>s do
    begin
        j:=tr[i];
        if j>0 then inc(f[j,i],d)
        else dec(f[i,-j],d);
        i:=abs(j);
    end;
    inc(kq,d);
end;
procedure process;

```



```

begin
    while timthayduongtangluong do tangluong;
writeln(fo,kq);
end;

BEGIN
    openfile;
    enter;
process;
closefile;
END.

```

Trong code trên, mảng $C[u,v]$ là khả năng thông qua từ u đến v , $Ke[u,v]=true$ nếu có cạnh nối từ u đến v , $F[u,v]$ là lưu lượng trên cạnh (u,v) ($F[u,v]<C[u,v]$).

Có thể hiểu rõ thuật toán này các bạn nên đọc Tài liệu giáo khoa chuyên Tin quyển 2, nhưng trong các kỳ thi lập trình thì chắc hẳn các thu code là có thể áp dụng được rồi.

Code xong các bạn có thể submit tại <http://vn.spoj.com/problems/NKFLOW/>

II/ Ví dụ :

Bài 1: Lập lịch thi đấu bóng đá.

(Nguồn: <http://vn.spoj.com/problems/BONGDA/>)

Mỗi giải thi đấu bóng đá sẽ tổ chức theo thể thức thi đấu vòng tròn, có hai đội bóng bất kỳ sẽ thi đấu với nhau ứng mỗi trận.

Trong mỗi trận đấu, mỗi đội thắng có 3 điểm, còn đội thua có 0 điểm. Không có kết quả hòa (nếu hòa trong hai hiệp chính thì hai đội sẽ phải thi đấu luân lưu để phân định thắng thua).

Kết thúc mùa giải, đội vô địch là đội có số điểm cao nhất. Trong trường hợp có nhiều đội có số điểm cao nhất thì các đội này sẽ bốc thăm để chọn đội vô địch.

Giải đã diễn ra một thời gian. Lúc này một bài toán nảy sinh: liệu ta cần biết mỗi đội bóng còn có khả năng bốc thăm vô địch nữa hay không?

Yêu cầu

Biết kết quả các trận đấu đã diễn ra, bạn hãy cho biết mỗi bóng còn có khả năng
 lọt vào khung môn mùa giải này hay không?

Dữ liệu

- Dòng 1: chứa số nguyên dương N là số bóng tham gia giải đấu. Các bóng được đánh số từ 1 đến N .
- N dòng tiếp theo, dòng i chứa N số nguyên dương $a_{i1}, a_{i2}, \dots, a_{iN}$, trong đó a_{ij} cho biết trạng thái của trận đấu giữa i và j :
 - $a_{ij} = 0$, nếu $i \neq j$ thì đúng.
 - $a_{ij} = 1$ nếu $i = j$ thì đúng.
 - $a_{ij} = 2$ nếu trận đấu chưa diễn ra.

Nếu i khác j , dữ liệu vào đảm bảo $a_{ij} = a_{ji} = 2$ hoặc $a_{ij} + a_{ji} = 1$. Dữ liệu vào đảm bảo $a_{ii} = 0$ với mọi i .

Kết quả

Gồm 1 dòng duy nhất, là một dãy nhị phân dài N , có bit thứ i bằng 1 nếu bóng i còn khả năng vào khung môn mùa giải, và bằng 0 nếu bóng i không còn khả năng vào khung môn mùa giải.

Giải thích

N là số nguyên dương trong phạm vi $[1, 30]$

Ví dụ

Dữ liệu

5

0 2 0 2 1

2 0 0 1 0

1 1 0 2 1

2 0 2 0 2

0 1 0 2 0

Kết quả

10110

Hướng dẫn: Xét mỗi bóng, xem xem vị trí của nó có thể thắng hay không.

Giả sử bạn xét bóng i , rõ ràng vị trí của nó mà bóng i chưa vào, ta nên cho i thắng. Sau thao tác này, giá trị của các bóng $1, 2, \dots, n$ thắng sẽ là $D[1], D[2], \dots, D[n]$. Nếu có

G m 1 dòng duy nh t là k t qu .

Example

Input:

2 2
11
11

Output:

2

H ng đ n : D th y có th ch t nh phân k t qu . Ta ph i ki m tra xem v i giá tr v có t n t i cách ch n $2*n$ ô en sao cho s ô en trong m i c t u không quá v không. T ó t a t o th nh sau:

- Có nh phát s, nh thu t.
- G i nh ng v i hàng i là i', nh ng v i c t j là j''.
- V i m i ô (i,j) mà giá tr t i ô ó =1 thì có c nh i t s n i' v i tr ng s 2, có c nh t i' n j'' v i tr ng s 2, có c nh t j'' n t v i tr ng s v.

N u lu ng c c i trên m ng v a t o b ng $2*n$ thì có t n t i cách ch n $2*n$ ô en trong m i c t u không quá v.

III/ Bài t p t luy n:

Bài 1: Chim cánh c t.

T i Nam C c có N hòn o c ánh s t 1 n N, hòn o th i v i t a ($X[i], Y[i]$) trên m t ph ng t a d Oxy có $A[i]$ con chim cánh c t. Theo t p t c ây, h ng th ng hòn o c ch n s ph i t ch c m t b a t i c thì t ã i t t c các con chim cánh c t khác. Oái o m thay, không ph i chim cánh c t t m t o b t kì có th i n c o b t kì khác b i có m t s rang bu c sau:

- Chim cánh c t o i có h i n o j n u kho ng cách -c -lit gi a hai o không quá D.
- o i có cao s v i m c n c b i n là $H[i]$, và c m i con chim cánh c t i ra kh i o này, cao c a nó l i gi m i l. Do v y, o không b chìm, s con chim cánh c t i ra kh i o i ph i không quá $H[i]$.

Nh ng con chim cánh c t ây r t kém tính toán, b n h y giúp chúng ki m tra xem hòn o nào có th ch n t ch t i c mà t t c các con chim cánh c t u có th tham gia nh.

INPUT: Vào t file v n b n Penguins.inp

-Dòng u tiên là s l ng hòn o N và s th c D.

-N dòng tiếp theo, mỗi dòng chứa 4 số nguyên dương $X[i], Y[i], A[i]$ và $H[i]$ với ý nghĩa như trên.

-Giới hạn: $N \leq 100$

$A[i], H[i] \leq 10^9$.

$0 < D \leq 10^{15}$.

+Giá trị tuyệt đối của $X[i]$ và $Y[i]$ không vượt quá 10^9 .

OUTPUT: Ghi ra file văn bản Penguins.out

-Giới hạn dòng, ghi ra chính xác các hòn có thể tích của các theo thứ tự tăng dần.

-Nếu không có hòn nào thỏa mãn, ghi ra -1.

Ví dụ :

| PENGUINS.INP | PENGUINS.OUT |
|--------------|--------------|
| 4 1.42 | 2 4 |
| 1 1 1 1 | |
| 2 1 2 5 | |
| 1 2 2 3 | |
| 3 2 4 4 | |

Bài 2: Nknet

(nguồn: <http://vn.spoj.com/problems/NKNET/>)

Trong một chỉ định, người ta thu thập các thông tin về một mạng truy cập cá nhân, bao gồm nút và mối liên hệ giữa các nút này. Các nút được đánh số từ 1 đến n. Hai nút liên lạc với nhau nếu có một nút trung gian nào giữa chúng hoặc có một dãy liên tiếp các nút nối qua một chuỗi trung gian nào đó.

Yêu cầu là tìm cách phá hủy một số nút và hai nút cho trước không liên lạc với nhau. Giả sử bạn chỉ hủy nhân lực của một người phụ trách vì cách phá hủy một nút và liên hệ phá hủy phát sinh. Do ảnh hưởng khác nhau nên việc phá hủy một nút và mối liên hệ cần một khoảng thời gian khác nhau. Hãy tìm

m t ph ng án th i i m hoàn thành nhi m v là s m nh t. N u có nhi u ph ng án nh th , hãy tìm ph ng án ph i c ít i nh t.

D li u

- Dòng u ghi giá tr n là s tr m c a m ng (không quá 100).
- Dòng ti p ghi giá tr m là s ng n i c a m ng (không quá $n(n-1)/2$).
- m dòng ti p theo, m i dòng ghi thông tin c a m t ng n i g m 3 giá tr nguyên d ng: hai giá tr u là s hi u c a ha tr m xác nh ng n i, giá tr sau (không quá 100) là th i gian c n thi t cho vi c phá h y ng n i này.
- * Dòng cu i cùng ghi hai giá tr là s hi u c a hai tr m c n c t t liên l c.

K t qu

- Dòng u ghi giá tr m là s ng n i c n phá h y.
- m dòng ti p theo, m i dòng mô t m t ng n i c n phá h y g m hai giá tr là s hi u c a hai tr m xác nh ng n i này.

Các giá tr s ghi trên cùng m t dòng cách nhau ít nh t m t d u tr ng. D li u vào luôn m b o có ng truy n tin n i hai tr m c n c t liên l c.

Ví d

D li u:

```
5
6
1 2 3
1 5 1
2 3 1
2 5 1
3 4 4
3 5 3
1 4
```

K t q a

```
3
1 5
2 3
2 5
```

Gi i thích: th i gian hoàn thành nhi m v là 1.

Bài 3: Flow1

(ngu n: <http://vn.spoj.com/problems/FLOW1/>)

Cu c thi giao l u "T t Ta Tin (TTT)" gi a hai i S Ph m (SP) và T ng H p (TH) có m bài toán tin h c, m i i có n h c sinh tham d . Các bài toán c ánh s t l n m và các h c sinh c a m i i c ánh s t l t i n.

H c sinh c a hai i u là nh ng l p trình viên xu t s c, tuy nhiên m i h c sinh có th gi i quy t nh ng bài toán thu c s tr ng c a mình hi u qu h n nh ng bài khác.

Hãy giúp th y My t ch c cu c thi theo th th c sau:

- Ch n úng n c p u, m i c p g m 01 h c sinh SP và 01 h c sinh TH làm 01 bài toán trong s nh ng bài toán này.
- Có úng n bài toán c mang ra thi
- H c sinh nào c ng c tham gia
- Bài toán cho c p u b t k ph i thu c s tr ng c a c hai thí sinh trong c p

Bi t r ng luôn t n t i ph ng án th c hi n yêu c u trên .

Input

- Dòng 1: Ch a hai s n, m (1 n m 255)
- n dòng ti p theo, dòng th i ghi danh sách các bài toán thu c s tr ng c a h c sinh SP th i.
- n dòng ti p theo, dòng th j ghi danh sách các bài toán thu c s tr ng c a h c sinh TH th j.

*Chú ý dùng **Eoln** ch không dùng **SeekEoln***

Output

G m m dòng, dòng th k ghi s hi u thí sinh SP và s hi u thí sinh TH trong c p u b ng bài toán k, n u bài toán k không c mang ra thi thì ghi vào dòng này hai s 0 .

Example

Input:

```
4 6
3 6
1 2
2 4
5
6
3 5 6
4
1 2 6
```

Output:

```
2 4
0 0
0 0
3 3
4 2
```

11

Bài 4: PBCWAYS(nguồn: <http://vn.spoj.com/problems/PBCWAYS/>)

Cho một bảng hình chữ nhật được chia ra thành $N \times M$ ô vuông (gồm N dòng và M cột). Một con tắc sau một lần đi có thể di chuyển từ ô cột này sang ô cột kế tiếp. Trong mỗi ô vuông, cho biết số hiệu các ô kế tiếp mà con tắc có thể di chuyển đến sau lần đi. Con tắc không thể di chuyển nếu nó đã đi qua tất cả các ô. Thời gian con tắc đi từ một ô nào đó đến ô cuối cùng là tổng thời gian đi của nó. Sau đó nó di chuyển về phía cuối cùng. Khi con tắc đã đến cuối cùng, nó lại đi vào một ô nào đó để bắt đầu lần đi tiếp theo.

Trò chơi kết thúc khi không thể tìm thấy con tắc.

Yêu cầu

Xác định xem có thể tìm thấy con tắc bao nhiêu lần di chuyển con tắc từ ô đầu tiên đến cuối cùng.

Input

Dòng đầu tiên chứa 2 số nguyên dương N, M ($1 \leq N \leq 50, 1 \leq M \leq 10$).

Tiếp theo là $M-1$ nhóm dòng, mỗi nhóm gồm N dòng, mô tả khả năng di chuyển của con tắc từ mỗi ô cột tiếp theo. Dòng thứ i của nhóm dòng j mô tả khả năng di chuyển của con tắc từ ô dòng i của cột j đến các ô tiếp theo. Số đầu tiên cho biết khả năng di chuyển, tiếp theo là tổng thời gian đi của các ô trong cột kế tiếp mà con tắc có thể di chuyển sang (các tổng thời gian đi theo thứ tự tăng dần).

Output

Gồm 1 dòng duy nhất là kết quả bài toán.

Example**Input**

```
4 3
2 1 3
3 1 2 4
0
2 2 3
1 2
1 2
1 3
```


2 2 4

Output

3

Ghi chú: Trong ví dụ có thể thể hiện nhiều nhất 3 lần cont t t c t ưu tiên n c t cu i cùng. Ch ng h n: (1->3->3; 2->4->4; 4->2->2).

Ch ng VI

HASH

I/ Phân tích v n :

M t l p nh ng bài toán r t c quan tâm trong khoa h c máy tính nói chung và l p trình thi c nói riêng, ó là x lý xâu chu i. Trong l p bài toán này, ng i ta th ng r t hay ph i i m t v i m t bài toán: tìm ki m xâu chu i.

Bài toán: Cho m t o n v n b n, g m m ký t và m t o n m u, g m n ký t. Máy tính c n tr l i câu h i: o n m u xu t hi n bao nhiêu l n trong o n v n b n và ch ra các v trí xu t hi n ó.

H ng d n: Có r t nhi u thu t toán có th gi i quy t bài toán này (Brute-force approach, KMP)

Thu t toán Hash:

a. Ký hi u:

- i. T p h p các ch cái c s d ng: Σ .
- ii. o n v n b n: $[1..m]$,
- iii. o n m u: $[1..n]$,
- iv. o n cont i n j c a m t xâu s : $[i..j]$

v. Chúng ta cần tìm ra tất cả các vị trí $(1 \leq i \leq m-n+1)$ thỏa mãn:
 $T[i..i+n-1]=P$.

b. Mô tả thuật toán:

Nguyên, giá trị ký tự $\Sigma = \{a, b, \dots, z\}$, nghĩa là Σ chứa các chữ cái Latin in thường. Để biểu diễn một chuỗi, thay vì dùng chữ cái, chúng ta sẽ chuyển sang biểu diễn dưới dạng số. Ví dụ: chuỗi “aczd” có vị trí ký tự là một dãy gồm 4 số: (0, 2, 25, 3). Như vậy, một chuỗi có biểu diễn dưới dạng mảng số h có số 26. Tuy nhiên, suy ra, 2 chuỗi bằng nhau khi và chỉ khi biểu diễn của 2 chuỗi h có số 10 gần nhau.

Ý chính là thuật toán của thuật toán: lấy 2 chuỗi h có số 26 ra h có số 10, rồi đem so sánh h có số 10. Tuy nhiên, chúng ta nhận thấy rằng, khi lấy 1 chuỗi ra biểu diễn h có số 10, biểu diễn này có thể rất lớn và nằm ngoài phạm vi lưu trữ nguyên của máy tính.

Khắc phục điều này, chúng ta chuyển sang so sánh 2 biểu diễn của 2 chuỗi h có số 10 sau khi lấy phần dư cho một số nguyên $base$ lớn. Cần chú ý: nếu biểu diễn trong hệ thập phân của chuỗi a là x và biểu diễn trong hệ thập phân của chuỗi b là y , chúng ta sẽ coi a bằng b

‘khi và chỉ khi’ $x \bmod base = y \bmod base$, trong đó $base$ là một số nguyên lớn.

Dễ dàng nhận thấy vì có so sánh $x \bmod base$ với $y \bmod base$ nên kết luận có bằng hay không là sai, $x \bmod base = y \bmod base$ chỉ là điều kiện cần chứ không phải điều kiện đủ. Tuy nhiên, chúng ta sẽ chấp nhận lỗi luận sai này trong thuật toán Hash. Và coi điều kiện cần như điều kiện đủ. Trên thực tế, lỗi luận sai này có thể xảy ra khi kiểm tra so sánh chuỗi không chính xác và chương trình bẻ gãy ra kết quả sai. Nhưng ngược lại cho thấy rằng, khi chọn là một số nguyên lớn, số lượng sai lầm trong hệ thập phân sẽ rất ít, và ta có thể coi Hash là một thuật toán chính xác.

Nguyên trong việc trình bày tìm kiếm thuật toán, chúng ta sẽ lấy biểu diễn của một chuỗi trong hệ thập phân sau khi lấy phần dư cho là mã Hash của chuỗi đó. Như vậy, 2 chuỗi bằng nhau ‘khi và chỉ khi’ mã Hash của 2 chuỗi bằng nhau.

Trong bài toán ban đầu, chúng ta cần đưa ra P xuất hiện như ký tự nào trong T . Để làm việc này, chúng ta chỉ cần duyệt qua mọi vị trí xuất phát có thể của P trong T . Giả sử vị trí đó là i , chúng ta sẽ kiểm tra $T[i..i+n-1]$ có bằng với P hay không. Để kiểm tra điều này, chúng ta cần tính mã Hash của đoạn $T[i..i+n-1]$ và mã Hash của chuỗi P .

Để tính mã Hash của chuỗi P chúng ta chỉ cần làm việc như sau:

```

hashP := 0
for i:=1 to n do
hashP = (hashP * 26 + P[i] - ord('a')) mod base

```

Phần khó hơn của thuật toán Hash là: Tính mã Hash của một đoạn $[i..j]$ của chuỗi (1 ≤ j ≤ m). Hình dung cho dễ, xét ví dụ sau:

Xét chuỗi s và biểu diễn của nó dưới cơ số 26: (4,1,2,5,17,8). Chúng ta cần lấy mã Hash của đoạn $[i..j]$ thì 3 nên phải lấy 6, nghĩa là cần lấy mã Hash của chuỗi (2,5,1,7). Như vậy, lấy chuỗi $[3..6]$, chuyển chuỗi $[1..6]$ là (4,1,2,5,1,7) trừ cho s ($[1..2]$ nên thêm (0,0,0,0)) là (4,10,0,0,0) ta sẽ thu được (2,5,1,7). Vậy, lấy mã Hash của chuỗi $[3..6]$, chính là mã Hash của $[1..6]$ trừ đi (mã Hash của $[1..2]$ nhân với 26^4).

Cần biết rằng, chúng ta cần biết $26^x \bmod \text{base}$ ($0 \leq x \leq m$) và mã Hash của từng vị trí của s , chính là mã Hash của từng ký tự $[1..i]$ (1 ≤ i ≤ m).

```

pow[0] = 1 ;
for i:=1 to m do
pow[i] = (pow[i-1] * 26) mod base ;
hashT[0] = 0;
for i:=1 to m do
hashT[i] = (hashT[i-1]*26+ord(T[i])-97)mod base

```

Trên đoạn code trên, chúng ta thu được 2 mảng:

pow[i] (lưu lại $26^i \bmod \text{base}$)

hashT[i] (lưu lại mã Hash của $t[1..i]$).

Lấy mã Hash của $[i..j]$ ta viết hàm sau:

```

function getHashT(i,j:longint):int64
begin
    exit(hashT[j]-hashT[i-1]*pow[j-i+1]+base*base)mod
base)
end;

```

Bài toán chính thức có gì quy tắc, và đây là chương trình chính:

```

for i:=1 to m-n+1do

```

```
if hashP = getHashT(i, i + n - 1) then
    write("Match position: ", i)
```

c. Mã chương trình:

```
const MAXN=100000;
base=1000000007;
var pow,hashT:array[0..MAXN] of int64;
t,p:string;
i,m,n,hashP:longint;
function gethashT(i,j:longint):int64;
begin
    exit(
        (hashT[j]-hashT[i-1]*pow[j-i+1]+int64(base)*base) mod
        base);
end;
begin
    readln(t);
    readln(p);
    m:=length(t);
    n:=length(p);
    pow[0]:=1;
    for i:=1 to m do pow[i]:=pow[i-1]*26 mod base;
    for i:=1 to m do
        hashT[i]:=(hashT[i-1]*26+ord(t[i])-97) mod base;
    hashP:=0;
    for i:=1 to n do
        hashP:=(hashP*26+ord(p[i])-97) mod base;
    for i:=1 to m-n+1 do
        if hashP=gethashT(i,i+n-1) then
            write(i, ' ');
    end.
```

d. Đánh giá:

phức tạp của thuật toán là $O(m+n)$. Nhưng lưu ý quan trọng là: chúng ta có thể kiểm tra 2 chuỗi có giống nhau hay không trong $O(1)$. Đây là lý do nên sử dụng thuật toán Hash. Ngoài sử dụng và thực thi, chúng ta có thể thay cài đặt thuật toán này thành bất kỳ ngôn ngữ lập trình nào khác.

a. Ưu điểm:

Ưu điểm của thuật toán Hash là cài đặt rất dễ dàng. Linh hoạt trong ứng dụng và có thể thay thế các thuật toán chuỗi 'hàm băm' khác.

b. Nhắc lại:

Nhắc lại mà thuật toán Hash là tính chính xác. Mặc dù rất khó sinh test có thể làm cho thuật toán chệch sai, nhưng không phải là không thể. Vì vậy, nâng cao tính chính xác của thuật toán, nên ta thường dùng modulo khác nhau so sánh mã Hash.

II/ Ứng dụng:

Như đã đề cập trên, thuật toán này sẽ có trường hợp chệch sai. Tuy nhiên, bên cạnh việc sử dụng Hash, còn có nhiều thuật toán xử lý chuỗi khác, mang lại sự chính xác tuyệt đối. Tôi tạm gọi thuật toán đó là ‘thuật toán chuỗi’. Vì cài đặt ‘thuật toán chuỗi’ có thể mang lại một cách xử lý chương trình cao hơn, chính xác của chương trình lớn hơn. Tuy nhiên, nên làm bài tập phải trả giá là số phép tính khi cài đặt các ‘thuật toán chuỗi’ đó.

Sử dụng Hash không chỉ giúp nên làm bài dễ dàng cài đặt hơn mà quan trọng nhất: Hash có thể làm được những việc mà ‘thuật toán chuỗi’ không làm được. Sau đây, tôi sẽ xét một vài ví dụ chứng minh điều này.

a. Longest palindrome substring

Bài toán đưa ra như sau: Cho một chuỗi s dài n ($n \leq 50\,000$). Bạn cần tìm dài nhất của chuỗi con dài nhất gồm các ký tự liên tiếp trong s . (Chuỗi con là chuỗi con tiếp nối nhau).

Một ‘thuật toán chuỗi’ không thể áp dụng vào bài toán này đó là thuật toán KMP. Ngoài KMP ra, có 2 ‘thuật toán chuỗi’ có thể áp dụng được. Thuật toán thứ nhất đó là sử dụng thuật toán Manacher tính bán kính lớn nhất từ vị trí trong chuỗi. Thuật toán thứ 2 đó là sử dụng Suffix Array và LCP (Longest Common Prefix) cho chuỗi con bất kỳ và chuỗi con tiếp theo theo thứ tự. 2 thuật toán này đều không dễ dàng cài đặt, và nằm ngoài phạm vi bài viết, nên tôi chỉ nêu sơ qua mà không đi vào chi tiết.

Bây giờ, chúng ta sẽ xét thuật toán ‘không chuỗi’ là thuật toán Hash. Như đã nói, chúng ta xét trường hợp dài nhất của chuỗi con là l (trường hợp chính xác lý hoàn toàn tuyệt đối). Giả sử chuỗi con dài l dài nhất có độ dài là l . Đầu tiên, trong chuỗi con tiếp theo của chuỗi con dài $l-2, l-4, \dots$. Tuy nhiên, chuỗi s không tồn tại chuỗi con dài $l+2, l+4, \dots$. Như vậy, l phải là một tính chất chia hết. Chúng ta sẽ chia hết để tìm l lớn nhất có thể. Vì mỗi độ dài l , chúng ta cần kiểm tra xem

trong xâu có tồn tại một xâu con là xâu i x ng dài l hay không. Làm vì c này, ta duyệt qua tất cả các xâu con dài l trong s .

Bài toán còn lại là: kiểm tra xem $s[i..j](1 \leq j \leq n; (j-i+1) \geq 2)$ có phải là xâu i x ng hay không.

Cách làm như sau. Gọi t là xâu s vì t theo thuật ngữ cũ. Bằng thuật toán Hash, chúng ta có thể kiểm tra xem xâu con nào có bắt đầu bằng xâu con nào có kết thúc hay không. Như vậy, chúng ta cần kiểm tra $s[i..k]$ có bằng $t[n-j+1..n-k+1]$ hay không với k là tâm i x ng , nói cách khác $k = \frac{i+j}{2}$. Như vậy bài toán đã giải quyết được cho cách làm này là $O(n \log(n))$.

b. k-th alphabetical cyclic

Bài toán đặt ra như sau: Cho một dãy a_1, a_2, \dots, a_n ($50\,000$). Số phép hoán vị vòng quanh của dãy này theo thứ tự i n. Cần, các hoán vị vòng quanh của dãy này là (a_1, a_2, \dots, a_n) , (a_2, a_3, \dots, a_1) , ... Dãy này có thứ tự i n như dãy kia n u s ưu tiên khác nhau của dãy này như dãy kia. Yêu cầu bài toán là: In ra dãy có thứ tự i n l n th k .

Nếu tính đến một cách trực tiếp, chúng ta sẽ sinh ra tất cả các dãy hoán vị vòng quanh, rồi sau đó dùng một thuật toán sắp xếp để sắp xếp lại chúng theo thứ tự i n, cuối cùng chọn ra dãy thứ k sau khi sắp xếp. Tuy nhiên, phức tạp của thuật toán này là $O(n^2)$ và không thể chấp nhận được yêu cầu về thời gian. Cần, cách này có phức tạp là $(n^2 * \log(n))$, đây là tích của phức tạp của sắp xếp và phức tạp của phép so sánh dãy.

Vấn đề tiếp theo là sắp xếp lại tất cả các dãy hoán vị vòng quanh rồi in ra dãy ng vị trí thứ k , chúng ta cần nghĩ đến một cách để so sánh thứ tự i n của 2 dãy.

Như cũ, nghĩ về thứ tự i n của 2 dãy: Xét 2 dãy a và b có cùng số phần tử. Gọi vị trí thứ i là vị trí ưu tiên thứ i trái sang mà ai bé, $a_i < b_i$ hoặc $a_i > b_i$. Như vậy, ta phải tìm một điểm nào đó để so sánh hai dãy a và b , rồi so sánh kết quả theo. Nếu tìm được một điểm nào đó để so sánh hai dãy, ta có thể sử dụng Hash để chia nhỏ phần.

Gọi i c bài này, cần sử dụng thêm một kỹ thuật nữa: Thay vì sinh ra tất cả các hoán vị vòng quanh, chúng ta chỉ cần nhân đôi dãy a lên, dãy mới sẽ có $2 * n$ phần tử. Một hoán vị vòng quanh s là một dãy con liên tiếp dài n của dãy nhân đôi này.

c. Longest substring and appear at least k times

Bài toán đặt ra như sau: Cho một chuỗi s dài ($n \leq 50000$), bạn cần tìm ra chuỗi con của s có độ dài l nhất, và chuỗi con này xuất hiện ít nhất k lần.

Bài toán này có thể giải bằng Suffix Array, tuy nhiên cách cài đặt phức tạp và không phù hợp trọng tâm của bài vì thế nên tôi sẽ không nêu ra đây.

Tiếp tục bàn về thuật toán Hash thay thế thuật toán chuần. Nhận xét rằng, giả sử độ dài l nhất tìm được là l , thì với mọi $l' \geq l$, luôn tồn tại chuỗi có độ dài l' xuất hiện ít nhất k lần. Tuy nhiên, với mọi $l' > l$, không tồn tại chuỗi có độ dài l' xuất hiện ít nhất k lần (do l đã là l nhất). Như vậy, l thỏa mãn tính chất chia nhỏ phần. Chúng ta có thể áp dụng thuật toán tìm kiếm nhị phân tìm ra l nhất.

Bây giờ, với mọi l khi đang chia nhỏ phần, chúng ta sẽ phải kiểm tra liệu có tồn tại chuỗi con nào xuất hiện ít nhất k lần hay không. Việc này sẽ làm rất tốn kém, bằng cách sinh ra mã Hash của các chuỗi con độ dài l trong s . Sau đó sắp xếp các mã Hash này theo thứ tự tăng dần, rồi kiểm tra xem có một số liên tiếp các mã Hash nào giống nhau độ dài k hay không.

Như vậy, độ phức tạp chia nhỏ phần là $O(n \log(n))$, độ phức tạp sắp xếp là $O(n \log(n))$, vậy độ phức tạp của bài toán là $(n \log(n))^2$.

III/ Bài tập:

Bài 1: MESSAGE

(Nguồn bài: VOI 2012-2013 <http://vn.spoj.com/problems/MESSAGE1/>)

An và Bình thường trao đổi thông tin qua mạng. Để tránh gặp rắc rối, hai bạn đã thống nhất cách truyền thông tin qua hai bước:

- Bước 1: *Giới thiệu thông tin*. Nội dung thông tin cần gửi sẽ gửi vào một bảng kết hình chữ nhật bằng cách in lần lượt các ký tự của chuỗi bị ẩn vào các ô của bảng trên xuống dần theo mỗi hàng và trái qua phải theo mỗi cột. Bảng này sẽ gửi vào một bảng kết hình chữ nhật có kích thước $M \times N$ nhất định. Các ô trống của bảng $M \times N$ sẽ in ký tự trống.
- Bước 2: *Giải thích thông tin*. Bảng $M \times N$ sẽ gửi qua mạng. Vị trí của hình chữ nhật chứa nội dung gửi qua sẽ in thông tin nhận.

Trong một lần An chuyển bảng A qua cho Bình, tuy nhiên Bình không nhận được. An thắc mắc và chuyển bảng B qua. Bảng A và B đều chứa hình chữ nhật nội dung thông tin, tuy nhiên vị trí hình này có thể khác nhau. Em gái Bình bí mật quy định trao đổi thông tin. Tò mò, cô muốn biết An đã gửi thông tin gì cho Bình bằng

cách tìm m t b ng hình ch nh t có di n tích l n nh t xu t hi n trong c 2 b ng A và B.

Input

- Dòng u ch a T – s l ng testcase. T nhóm dòng, m i nhóm miêu t 1 testcase.
- Dòng th nh t ch a 2 s nguyên d ng M, N.
- Dòng th i trong M dòng ti p theo ch a m t xâu g m N kí t ch g m các ch cái la tinh th ng mô t b ng A.
- Dòng th j trong M dòng ti p theo ch a m t xâu g m N kí t ch g m các ch cái la tinh th ng mô t b ng B.

Output

- G m T dòng, dòng th i ghi m t s nguyên là di n tích hình ch nh t l n nh t tìm c t ng ng v i testcase th i.

Example

| Input | Output |
|--|--------|
| 1 4 5 tinaa hocaa aaaaa ccccc bbbbd btind bhocd bbbbd | 6 |

H ng d n: ‘Thu t toán chu n’ c a bài này có ph c t p là $O(N^4)$ (các b n th ngh xem), nh ng v i Hash ta có 1 thu t toán ‘không chu n’ v i ph c t p nh h n là $O(N^3 \log(N))$. Ý t ng c a ta là v i x, y ta ki m tra xem c 2 b ng A và B có ch a cùng 1 hình ch nh t kích th c $x*y$ không. Ta t o Hash c a t t c các hình ch nh t kích th c $x*y$ trong b ng A, sort l i, sau ó v i m i Hash c a hình ch nh t $x*y$ trong B ta ch t nh phân xem có t n t i giá tr này không. Công vi c này m t chi phí $O(N^2 \log(n))$, c ng thêm vì c duy t t t c các t h p x, y m t $O(N^2)$ n a, nh ng nh n xét th y r ng v i m i x có th ch t nh phân dài y, nh v y ch m t chi phí là $O(N \log(N))$ thôi. V y thu t toán này có ph c t p là $O(N^3 \log(N))$. Còn 1 l u ý n a là, vì s Hash c a bài này khá nhi u nên mu n Accept ta ph i cài 2 b ng Hash v i 2 c s mod khác nhau.

Bài 2: Hai thao tác trên chu i

(Nguồn bài <http://vn.spoj.com/problems/TWOOPERS/>)

John có một chuỗi S . John yêu cầu thực hiện hai thao tác sau **theo thứ tự** trên S :

- Chọn một vị trí c của S , và thay thế bằng bất kỳ ký tự nào John muốn.
- Đổi chuyển chuỗi S , nghĩa là, John có thể chọn một vị trí k và đổi chuỗi S theo vòng tròn sao cho kết quả thành vị trí bắt đầu của chuỗi mới.

John muốn sau khi thực hiện hai phép toán trên, kết quả thu được là một chuỗi cho trước. Bạn hãy giúp John tính số cách biến đổi chuỗi S thành một chuỗi T cho trước.

Định nghĩa

- Định nghĩa bao gồm hai chuỗi S và T trên một dòng. Mỗi chuỗi bao gồm nhiều nhất 100000 ký tự và chỉ gồm các ký tự in hoa.
- Mỗi bộ nhớ S và T có cùng số ký tự.

Kết quả

Một số duy nhất là số cách biến đổi chuỗi S thành chuỗi T .

Ví dụ

Định nghĩa

AHYANGYI YANGYIAH

Kết quả

8

Gợi ý thích

John có thể thay thế chữ "A" đầu tiên bằng "A", hoặc "H" bằng "H", v.v...

nghĩa là có thể thay thế một chữ bằng chính nó.

Sau đó, có một cách đổi chuyển chuỗi.

Định nghĩa

VSUMSU MSUMSU

Kết quả

2

Gợi ý thích

John có thể thay thế chữ "V" đầu tiên bằng "M".

Sau đó, John có hai cách đổi chuyển chuỗi ($k=1$ hoặc $k=4$).

Hướng dẫn: Cách nghĩ thông thường là vì mỗi vị trí của chuỗi S xem ta có thể thay ký tự vị trí đó bằng ký tự gì và vị trí thay thế đó có bao nhiêu cách đổi chuyển

$S=T$. Cách này xem ra khá khó khăn. Ta thay vì cách nghĩ như sau: Với mỗi cách dịch chuyển n và vị trí k của chữ S và u ($k=1,2,...,n$ và $n=length(S)$), giả sử A là chữ có ở vị trí k sau khi dịch chuyển, ta tính xem có bao nhiêu cách thay thế các vị trí khác $A=T$. Giả sử là u là vị trí đầu tiên sao cho $A[1..u]=T[1..u]$ và v là vị trí đầu tiên sao cho $A[n-v+1..n]=T[n-v+1..n]$. Đặt $y=n-u+v=n-1$ thì tổng bị tính từ u đến y , còn $n-u+v \geq n$ thì tổng bị tính từ u đến n . Ta có thể dùng Hash để tính nhanh tìm u, v .

Bài 3: Davy Jones 's Organ

(nguồn: <http://acm.timus.ru/problem.aspx?space=1&num=1937>)

Cho 2 chuỗi, chuỗi 1 dài n , chuỗi 2 dài m chỉ chứa các chữ cái latin in thường. $1 \leq m \leq n \leq 10^5$. Cho số k của chuỗi 2 ảnh hưởng tới 1. Bạn cần kiểm tra xem ghép chuỗi 2 vào sau chuỗi 1 thì có tạo nên chuỗi mới hay không. Tuy nhiên, mỗi chuỗi trong 2 chuỗi trên bạn có thể xoay vòng trước khi nối. Ví dụ chuỗi $abcd$ thì có thể xoay vòng để tạo thành các chuỗi $bcda$; $cdab$; $dabc$.

Input:

Gồm 2 dòng, dòng đầu là chuỗi 1, dòng 2 là chuỗi 2.

Output: Nếu có thể xoay vòng và nối chuỗi để tạo thành chuỗi mới thì in ra Yes dòng đầu tiên, dòng thứ 2 in ra vị trí của ký tự đầu tiên sau khi xoay vòng của chuỗi. Ví dụ nếu chuỗi ban đầu là $abcd$ mà bạn xoay nó thành $cdab$ trước khi ghép thì in ra 3 (vị trí của c trong chuỗi ban đầu). Nếu không tìm thấy cách nào thì in ra No.

Ví dụ :

| Input | output |
|---------------|------------|
| cdedab bac | Yes 5 3 |
| aaaa bbb | No |

(dùng 2 hash mới được chấp nhận)

Bài 4:STRCUT

Cho 2 chuỗi $S1$ và $S2$ dài bằng nhau, mỗi chuỗi có độ dài không quá 2000 ký tự, các ký tự là chữ cái la tinh thường.

Yêu cầu: Hãy xác định số cách cắt S2 thành 3 xâu khác nhau, mà tất cả có thể là phần thành xâu S1. Hai cách cắt khác nhau là khác nhau nếu vị trí cắt khác nhau.

Ví dụ, có thể cắt xâu 'beast' thành 3 phần và là phần xâu 'betas', nhưng 'royalitem' không thể chia thành 'romeitaly'.

Dữ liệu: Vào tệp file văn bản STRCUT.INP gồm 2 dòng, dòng 1 chứa xâu S1, dòng 2 chứa xâu S2

Kết quả: In ra file văn bản STRCUT.OUT số cách cắt thành 3 phần.

| STRCUT.inp | STRCUT.out |
|----------------|------------|
| beast betas | 1 |
| aaaaa aaaaa | 6 |

Subtask 1: $\text{length}(S1), \text{length}(S2) \leq 300$

Subtask 2: $\text{length}(S1), \text{length}(S2) \leq 2000$

Ghi ý: Duyệt tất cả các cách cắt, vì mỗi cách cắt kiểm tra xem có thể chia thành xâu S1 không. Vì số cách cắt là N^2 nên thao tác kiểm tra cần phải làm trong $O(1)$ (có thể dùng hash-cách không dùng hash văn bản $O(1)$ và khá hay, các bạn thử xem).

Bài 5: GCITP

Cho 2 dãy số a_1, a_2, \dots, a_n và b_1, b_2, \dots, b_n . Ta gọi bộ $([x_A, y_A], [x_B, y_B])$ là một cặp chung của hai dãy số:

- $1 \leq x_A < y_A \leq n$
- $1 \leq x_B < y_B \leq n$
- $\{a_i \mid i \in [x_A, y_A]\} = \{b_i \mid i \in [x_B, y_B]\}$

Yêu cầu: Cho hai dãy số và tìm số cặp chung của hai dãy.

Input

- Dòng 1: chứa số n ;
- Dòng 2: chứa n số mô tả dãy a_i ($|a_i| \leq 10^9$)
- Dòng 3: chứa n số mô tả dãy b_i ($|b_i| \leq 10^9$)

Output

- Gồm 1 dòng chứa số cặp chung của 2 dãy

| GCITP.inp | GCITP.out |
|-----------|-----------|
| 3 | 3 |

| | |
|---------------------|---|
| 1 2 3 3 2 1 | |
| 3 1 1 1 1 1 3 | 3 |

Subtask 1: $n \leq 30$

Subtask 2: $n \leq 300$; 2 dãy ai và bi là 2 hoán vị của $\{1, 2, \dots, n\}$

Subtask 3: $n \leq 30$

Subtask 4: $n \leq 5000$; 2 dãy ai và bi là 2 hoán vị của $\{1, 2, \dots, n\}$

G i ý: Subtask 2 và 4 là tr ãng h p c bi t, ta ý 1 chút là s ra còn subtask 1 và 3 có th dùng Hash.

Ch ãng VII

SUFFIX TREE

I/ Phân tích v n :

Cây h u t là m t c u trúc d li u bi u di n các h u t c a m t xâu, c ãng d ãng r ãng rãi trong các thu t toán x lý xâu b i nó cung nhi u phép toán hi u qu giúp làm gi m th i gian th c hi n gi i thu t. M t c u trúc d li u d n xu t là m ãng h u t tuy ph m vi ãng d ãng h p h n cây h u t nh ãng l i r t ãng i n trong cài t. T ãng d ãng nh ãng u i m c a c hai c u trúc d li u ó, r t nhi u thu t toán hi u qu ã c công b trong nh ãng n m g n ãy. Chuyên ã này gi i thi u m t s thu t toán xây d ãng cây h u t và m ãng h u t cùng v i m t s bài toán c b n mà thu t toán gi i quy t chúng là nh ãng ví d i n hình c a v i c ãng d ãng hai c u trúc d li u này. Bên c nh ó, chuyên ã ãng trình bày m t s m r ãng và th o lu n v ãng ãng ãng m cài t trong các k thi l p trình v i th i gian h n ch .

1. Gi i thi u

Cây h u t (*suffix trees*) là m t c u trúc d li u quan tr ãng c s d ãng trong r t nhi u thu t toán x lý xâu. S c m nh c a cây h u t n m kh ãng ãng bi u di n t t c các h u t c a m t xâu và cung c p nhi u phép toán quan tr ãng giúp ãng ãng tính hi u qu c a nh ãng thu t toán. Chính nh ãng ãng tính ch t ó mà cây h u t c s

đang trong r t nhi u l nh v c khác nhau nh : x lý v n b n, trích ch n và tìm ki m thông tin, phân tích d li u sinh h c, i sánh m u v.v...

Bên c nh u i m là m t c u trúc d li u m nh, các thu t toán tr c ti p xây d ng cây h ut có nh c i m là ph c t p và t n b nh . M ng h ut (*suffix arrays*) là m t c u trúc d li u d n xu t t cây h ut và là m t s thay th h p lý cho cây h ut trong m t s ng d ng c thù. Xét v tính n ng, m ng h ut không h tr nhi u phép toán nh cây h ut nh ng l i có th cài t khá d dàng.

M c dù m ng h ut là c u trúc d li u d n xu t và có th xây d ng t cây h ut t ng ng, ã có r t nhi u thu t toán có th xây d ng m ng h ut m t cách tr c ti p mà không c n dùng n cây h ut . Nh ng thu t toán nh v y cho phép n gi n hóa r t nhi u thao tác x lý xâu, b i trong tr ng h p có th s d ng m ng h ut gi i quy t, ta không c n bi t v khái ni m cây h ut n a.

C ng t khi có nh ng thu t toán tr c ti p và hi u qu xây d ng m ng h ut , r t nhi u nghiên c u ã tìm ra ph ng pháp xây d ng theo chỉ u ng c l i: D ng cây h ut t m ng h ut . Nh ng ph ng pháp này có u i m là nhanh và ti t ki m b nh h n so v i phép xây d ng tr c ti p cây h ut . Ngoài ra, nh ng ph ng pháp này còn cung c p nhi u k thu t hay trong x lý d li u, có th k th a ng d ng trong nh ng l nh v c khác.

Trong các ph n ti p theo c a chuyên , ph n 2 gi i thi u các khái ni m c s v trie h ut , cây h ut , m ng h ut và m ng ti n t chung dài nh t. Ph n 3 trình bày m t s thu t toán xây d ng m ng h ut và cây h ut . Ph n 4 nêu m t s bài toán c b n cho th y hi u qu c a v i c ng d ng c u trúc d li u m ng h ut và cây h ut .

Cu i cùng là k t lu n và m t s m r ng c a c u trúc d li u.

2. M t s khái ni m c s

G i Σ là m t t p h u h n có th t g i là b ng ch cái (*alphabet*), các ph n t $\in \Sigma$ c g i là ký t . Σ^* là t p các xâu (*string*) g m các ký t $\in \Sigma$. Có th coi m i xâu $\in \Sigma^*$ là m t dãy h u h n các ký t $\in \Sigma$. Ký hi u ϵ là xâu r ng, t p các xâu khác r ng c g i là $\Sigma^+ = \Sigma - \{\epsilon\}$.

Chi u dài c a m t xâu x , ký hi u $|x|$, là s ký t trong xâu x . Các ký t trong xâu x c ánh s t 0 t i $|x| - 1$: $x = x_0x_1 \dots x_{|x|-1}$.

Xâu n i c a hai xâu x và y , ký hi u xy , có chi u dài $|x| + |y|$ và t o thành b ng cách l y các ký t trong x sau ó n i ti p v i các ký t trong y .

Ta gọi w là tiền tố (*prefix*) của x , ký hiệu $w \sqsubset x$, nếu tồn tại y

$x = wy$, w cũng là hậu tố (*suffix*) của x , ký hiệu $w \sqsupset x$, nếu tồn tại y

$x = yw$. Dãy y ngắn hơn w là tiền tố hoặc hậu tố của x thì $|w| \leq |x|$. Một chuỗi có thể vừa là tiền tố vừa là hậu tố của một chuỗi khác. Ví dụ **ABA** vừa là tiền tố vừa là hậu tố của chuỗi **ABABA**. Chuỗi rỗng ϵ vừa là tiền tố, vừa là hậu tố của tất cả các chuỗi.

Hai quan hệ \sqsubset, \sqsupset có tính bắc cầu, tính là:

- Nếu $x \sqsubset y$ và $y \sqsubset z$ thì $x \sqsubset z$.
- Nếu $x \sqsupset y$ và $y \sqsupset z$ thì $x \sqsupset z$.

B 1 (về tính giao nhau của các tiền tố và các hậu tố)

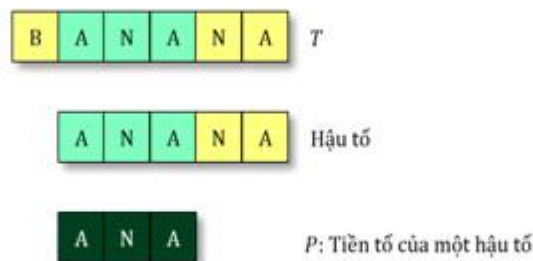
- Nếu x và y cùng là tiền tố của một chuỗi z thì s là tiền tố của y nếu $|x| \leq |y|$, ys là tiền tố của x nếu $|y| \leq |x|$.
- Nếu x và y cùng là hậu tố của một chuỗi z thì s là hậu tố của y nếu $|x| \leq |y|$, ys là hậu tố của x nếu $|y| \leq |x|$.

Vì dễ dàng chứng minh **B 1** khá hiển nhiên: Hai tiền tố của cùng một chuỗi có quan hệ tiền tố và hai hậu tố của cùng một chuỗi có quan hệ hậu tố.

Cho $T = t_0 t_1 \dots t_{n-1}$ và $P = p_0 p_1 \dots p_{m-1}$ là hai chuỗi ký tự, ta nói chuỗi P xuất hiện trong chuỗi T tại vị trí k nếu $P = t_k t_{k+1} \dots t_{k+m-1}$. Nếu chuỗi P xuất hiện trong chuỗi T tại vị trí nào đó thì P là chuỗi con (*substring*) của T .

Có thể coi chuỗi con của một chuỗi T là một dãy các ký tự liên tiếp trong T . Một cách

định nghĩa khác về chuỗi con của T đó là một **tiền tố của một hậu tố** của T .



Hình 1. Chuỗi con = tiền tố của một hậu tố

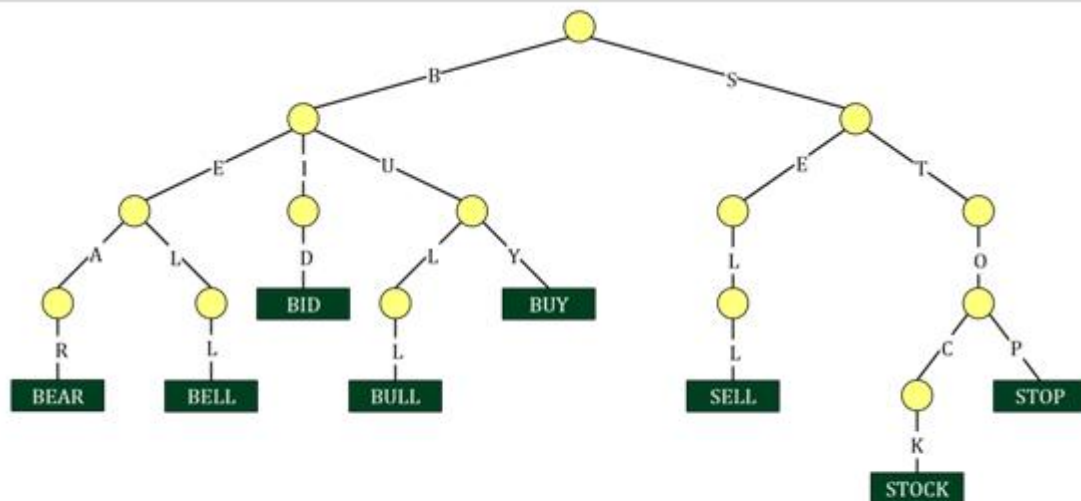
Trong các ví dụ của chuyên đề này, ta coi tập hợp các ký tự Σ là tập 26 ký tự hoa tiếng Anh: từ 'A' đến 'Z' cộng thêm một ký tự đặc biệt ký hiệu '@'. Tập hợp các ký tự ghi nhận trong các bảng mã thông dụng ANSI/ASCII/Unicode: Ký tự '@' là ký tự đầu tiên trong bảng chữ cái Σ , tiếp theo là các ký tự từ 'A' đến 'Z'.

2.1. Trie hay cây

Cho S là một tập hợp n chuỗi khác nhau: không chuỗi nào là tiền tố của một chuỗi khác. Trie¹ [3] của tập S là một cấu trúc dữ liệu dạng cây biểu diễn các chuỗi $\in S$:

- Mỗi đỉnh của cây có nhãn là một ký tự $\in \Sigma$. Các cạnh kết nối các nút con của nó phải mang các nhãn hoàn toàn phân biệt.
- Mỗi nút v trên trie có nhãn, nhãn của nút v , ký hiệu $\tau(v)$ là chuỗi tạo thành bằng cách nối tiếp các ký tự nhãn của các đỉnh trên đường đi từ gốc đến nút v . Chiều dài của chuỗi $\tau(v)$ gọi là **sâu** của nút v , ký hiệu $depth(v)$. Theo cấu trúc của trie, hai nút khác nhau phải có chuỗi nhãn khác nhau.
- Có một ứng ánh xạ 1-1 giữa các chuỗi $\in S$ và các nút lá trên trie: trie có đúng n lá và mỗi lá có nhãn là một chuỗi $\in S$.

Hình 2 là trie biểu diễn 8 chuỗi: BEAR, BELL, BID, BULL, BUY, SELL, STOCK, STOP

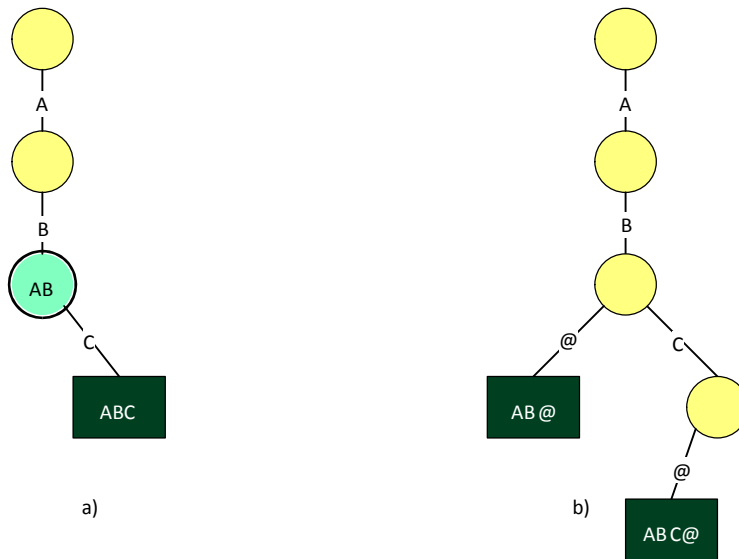


Hình 2. Trie

Các chuỗi trong tập S phải thỏa mãn tính chất phi tiền tố (*prefix-free*): không có chuỗi nào là tiền tố của một chuỗi khác, không thể xây dựng một trie biểu diễn tập S nếu không

kiến này bị vi phạm. Thay vì vậy, giả sử $S = \{ABC, AB\}$, nếu ta xây dựng trie biểu diễn các chuỗi ABC, thì có nút lá mang nhãn ABC. Trên đường đi từ gốc xuống lá, ta sẽ đi qua lần lượt các nút có nhãn là ϵ, A, AB, ABC . Có nút nhánh của trie mang nhãn AB, thì có nút lá của trie mang nhãn ABC của nó (Hình 3 a)).

Tính chất phi tối thiểu là một yêu cầu cần quy tắc xây dựng trie, nhưng thuật toán dựa trên trie để phân tích tính chất này cần phải lưu ý (ví dụ như thuật toán mã hóa Huffman). Một trong những kỹ thuật mà bộ dữ liệu có tính phi tối thiểu là bổ sung một ký tự ngăn ngừa việc các ký tự làm ký tự **cấm** mà ta ký hiệu là @: mỗi chuỗi $\in S$ sẽ thêm ký tự @ vào cuối chuỗi mà bộ không có chuỗi nào là tiền tố của một chuỗi khác. Hình 3 b) là ví dụ về trie biểu diễn tập S gồm hai chuỗi: ABC@ và AB@. Có thể thấy rằng khi sử dụng ký tự cấm thì mỗi nút trên trie chỉ là nút lá mang nhãn @.



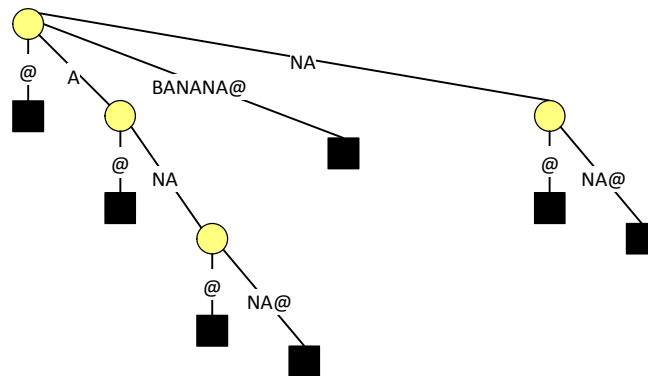
Hình 3. Vai trò của ký tự cấm @

Nếu S là tập các chuỗi khác nhau của một chuỗi $T \in \Sigma^+$ thì trie biểu diễn S cũng là **trie hậu tố** (*suffix trie*) của T . Để đảm bảo tính chất phi tối thiểu của tập S , ta coi chuỗi T có một ký tự cấm @ ngay cuối cùng còn mọi ký tự khác trong T đều không phải ký tự @. Ví dụ nếu T là chuỗi BANANA@, tập S các chuỗi khác nhau của T gồm có 7 chuỗi:

BANANA@
ANANA@
NANA@
ANA@
NA@
A@
@

Chính xác hơn, cây hậu tố của một chuỗi $T \in \Sigma^+$, ký hiệu $ST(T)$ là một cấu trúc dữ liệu dạng cây có các tính chất sau:

- Mọi nhánh của cây có nhãn là một chuỗi $\in \Sigma^+$. Các cạnh từ một nút xu hướng các nút con của nó phải mang nhãn là các chuỗi có ký tự đầu tiên hoàn toàn phân biệt.
- Mọi nút v trên cây hậu tố mang một nhãn, nhãn của nút v , ký hiệu v' là chuỗi tạo thành bằng cách nối tiếp các nhãn cạnh trên đường đi từ gốc xu hướng nút v . Chiều dài của chuỗi $v' = |v'|$ cũng gọi là **sâu** của nút v , ký hiệu $depth(v)$. Theo cấu trúc của cây hậu tố, hai nút khác nhau phải có chuỗi nhãn khác nhau.
- Ngoi trừ nút gốc, không nút nào trên cây hậu tố có 1 nút con.
- Có một ứng đồng 1-1 giữa các hậu tố của T với các nút lá trên $ST(T)$: $ST(T)$ có ứng $|T|$ lá và mọi lá có nhãn là một hậu tố của T .



Hình 6. Cây hậu tố

Hình 6 là ví dụ về cây hậu tố của chuỗi BABABA@. Bảng 2 chỉ ra chi phí bình quân của một cây hậu tố khi bị duyệt xuôi.

Bảng 2

Cây hậu tố của một chuỗi dài n có không quá $2n$ nút

Chứng minh

Giả sử cây hậu tố có m nút nhánh tức là có tổng cộng $m + n$ nút. Ta biết rằng số nhánh của cây bằng $m + n - 1$ và tổng số con của tất cả các nút trên cây ứng bằng số nhánh trên cây. Các nút lá trên cây có số con bằng 0, các nút nhánh ngoi trừ nút gốc có số con ≥ 2 , nút gốc có ít nhất 1 con. Vì vậy tổng số con của tất cả các nút trên cây không

thành $2m - 1$, thì nó có $2m - 1 \leq n + m - 1$ hay $m \leq n$, tức là cây h có không quá n nút nhánh và tổng có không quá $2n$ nút.

Mặc dù cây h có phát sinh từ s , nhưng nghiên cứu trên cây h lại xây dựng nên nghiên cứu có lợi trong những lĩnh vực khác nhau vì cấu trúc cây có một số khác biệt. Khi tìm kiếm những tài liệu liên quan tới cây h , ta có thể bắt gặp nhiều tên gọi khác nhau như “suffix trees”, “compacted bi-trees”, “prefix trees”, “PAT trees”, “position trees”, “repetition finder”, “subword trees”,... Cây h có giới thiệu lần đầu tiên bởi Morrison với tên gọi cây PATRICIA [14]. Tuy vậy Weiner mới là người chú n hóa cấu trúc cây h (dĩ nhiên gọi compacted bi-trees) và đưa ra thuật toán tuyến tính xây dựng cây [18], thuật toán này sau đó của Donald Knuth bình chọn là “thuật toán của năm 1973”. Tiếp theo nghiên cứu của Weiner, các thuật toán tuyến tính xây dựng cây h liên tục cải tiến và tối ưu hóa, chẳng hạn như thuật toán của McCreight năm 1976 [13], của Slissenko năm 1983 [16]. Những thuật toán tuyến tính ưu tiên có thể làm việc trực tuyến cũng xuất hiện Kosaraju năm 1994 [11] và Ukkonen năm 1995 [17], các thuật toán này có khả năng xây dựng cây h bằng cách cắt ngắn ký tự trong chuỗi ngược trái qua phải, chính vì vậy nó thích hợp khi muốn xây dựng cây h bằng cách nhúng từng ký tự lên cây truy cập.

Bên cạnh việc xuất các thuật toán xây dựng cây, các nghiên cứu cũng đã đưa ra thêm một số ứng dụng của cây h , chẳng hạn trong lĩnh vực xử lý văn bản và dữ liệu sinh học. Tuy nhiên, có hai nhược điểm chung của các phương pháp xây dựng cây h trực tiếp, đó là:

- Các thuật toán có thể thể hiện trong thời gian tuyến tính và sử dụng bộ nhớ tuyến tính, nhưng có một hạn chế lớn trong ký pháp O đánh giá phức tạp tính toán. Trên thực tế, những trình cài đặt thuật toán xây dựng cây h khá chậm và tốn bộ nhớ.
- Mặc dù đã có nhiều cố gắng trong việc tối ưu hóa thuật toán nhưng cho tới nay, các mô hình cài đặt những thuật toán trên vẫn còn rất phức tạp và dễ nhầm lẫn.

2.3. Mảng hậu tố

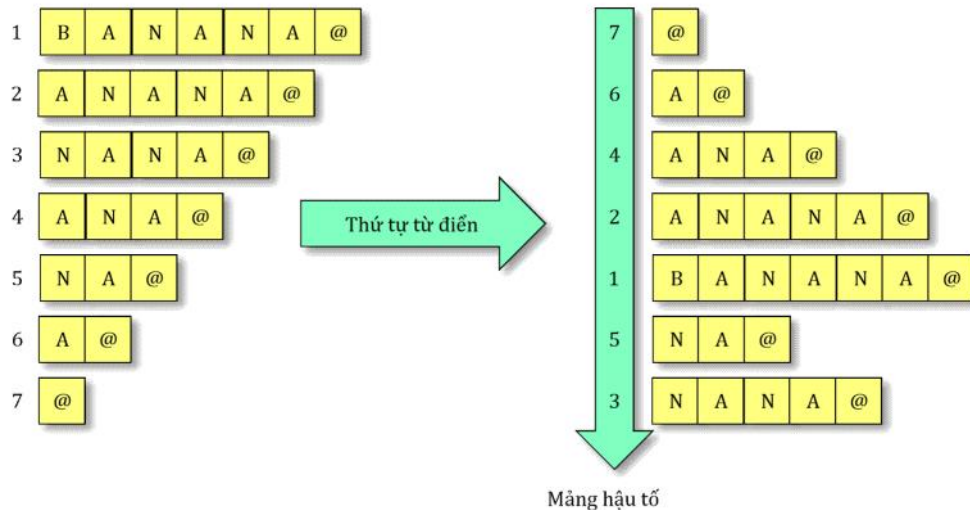
Cho một chuỗi $T = t_0 t_1 \dots t_{n-1} \in \Sigma^+$, quy ước có duy nhất $t_{n-1} = @$. **Mảng hậu tố** (suffix array) của T , ký hiệu $SA(T)$ là một tập các hậu tố của T .

Mỗi hậu tố $t_{i \dots n-1}$ có thể được nhận biết vị trí i , khi đó mảng hậu tố của chuỗi T có thể biểu diễn như là một hoán vị $(a_0, a_1, \dots, a_{n-1})$ của dãy số $(0, 1, \dots, n-1)$ sao cho:

$$t[a_0 \dots n-1] < t[a_1 \dots n-1] < \dots < t[a_{n-1} \dots n-1]$$

(Đây ta dùng ký hiệu “<” cho thứ tự của các chuỗi, nếu $x < y$ thì x là chuỗi nhỏ hơn chuỗi y theo thứ tự này)

Ví dụ với chuỗi $T = BANANA@$, các nút của T và mảng nút $(7,6,4,2,1,5,3)$ đang có thể thấy trong Hình 7.



Hình 7. Mảng hậu tố

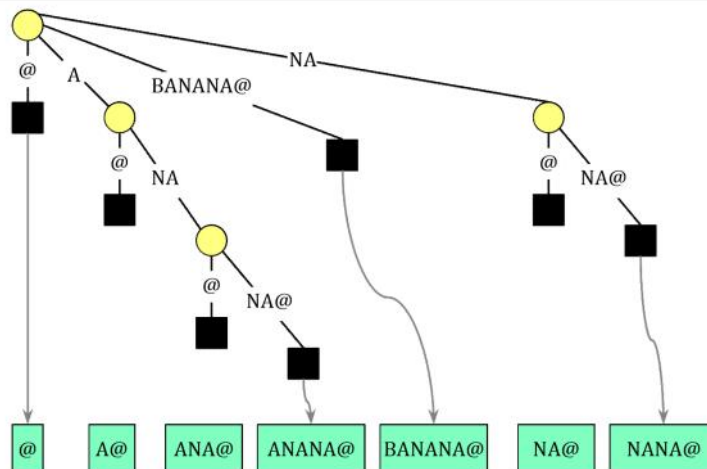
Mảng hậu tố (T) của chuỗi độ dài n có thể xây dựng trực tiếp từ cây hậu tố (T) $SAT n ST$

trong thời gian $O(n)$. Thuật toán có thể mô tả như sau:

Quy trình duyệt các nút con của mỗi nút: Theo thứ tự từ điển, nút con nhỏ hơn về vị trí mang nhãn nhúng sẽ đứng trước nút con nhỏ hơn về vị trí mang nhãn lớn hơn. Vì các chuỗi nhãn của các cạnh liên tiếp tạo thành một nút phụ có ký tự ưu tiên khác nhau, thứ tự của các cạnh nên là thứ tự ngược của ký tự ưu tiên trong các nhãn cạnh.

Duyệt cây bằng DFS bắt đầu từ gốc, khi thăm tới mỗi nút ta liệt kê tất cả các nút con của nó theo thứ tự đã quy định. Khi có danh sách các nút lá theo thứ tự

thăm sẽ ứng với danh sách các hậu tố liệt kê theo thứ tự từ điển (Hình 8).



Hình 8. Cây hậu tố và mảng hậu tố

M ng h u t c x u t b i Manber và Myer [12] nh m t s thay th cho cây h u t trong m t s bài toán x lý xâu. u i m chính c a m ng h u t là tính n g i n trong c u trúc và s t i t k i m b nh trong b i u di n. Manber và Mayer c ng x u t thu t toán xây d ng m ng h u t tr c t i p mà không ph i s d ng cây h u t g i là thu t toán nhân ôi ti n t (doubling prefix). M c dù thu t toán có th i gian th c h i n $\Omega(n \log n)$ trong tr ng h p x u nh t, trung bình thu t toán ch c n th i gian $O(n)$ xây d ng m ng h u t c a xâu dài n .

C ng ã có r t nhi u n l c tìm k i m thu t toán hi u qu h n xây d ng m ng h u t . N m 2003, hai nghiên c u c l p c a Kärkkäinen [7] và Ko [10] ã tìm ra c hai thu t toán tuy n tính xây d ng m ng h u t . M t i m áng chú ý trong các thu t toán c a Kärkkäinen và Ko là chúng u d a trên nh ng nh n nh r t tinh t v tính ch t c a các h u t và m i quan h g i a các v trí trong xâu. V i c phá b c rào c n $\Omega(n \log n)$ trong tr ng h p x u nh t ã m ra m t t i m n ng m i cho v i c s d ng m ng h u t mà các nghiên c u t i p sau ã s d ng xây d ng cây h u t t m ng h u t mà không làm t ng ph c t p tính toán c a g i thu t.

2.4. M ng t i n t chung dài nh t

Ti n t chung dài nh t (*longest common prefix*) c a hai xâu x, y là xâu z có dài l n nh t th a m n: z v a là t i n t c a x v a là t i n t c a y . Ví d t i n t chung dài nh t c a SUFFIXTRIE và SUFFIXTREE là xâu SUFFIXTR.

Cho $T = t_0 t_1 \dots t_{n-1} \in \Sigma^+$, $SA(T) = (a_0, a_1, \dots, a_{n-1})$ là m ng h u t c a T . M ng t i n t chung dài nh t $LCP(T)$ là dãy s nguyên $(l_0, l_1, \dots, l_{n-1})$ nh ng a nh sau:

- $l_0 = 0$;

- $\forall i > 0: l_i$ là dài tỉn t chung dài nh t gi a h u t t i v trí a_i và h u t t i v trí a_{i-1} trong xâu T

Ví d v i xâu $T = \text{BANANA@}$, m ng h u t c a T là $(7, 6, 4, 2, 1, 5, 3)$, ta có:

$$l_0 = 0;$$

$$l_1 = 0 \text{ (dài tỉn t chung dài nh t c a } A@ \text{ và } @ \text{)}$$

$$l_2 = 1 \text{ (dài tỉn t chung dài nh t c a } \underline{A}NA@ \text{ và } \underline{A}@ \text{)}$$

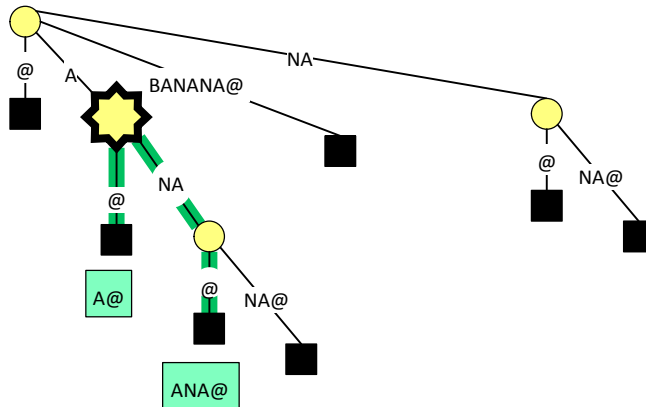
$$l_3 = 3 \text{ (dài tỉn t chung dài nh t c a } \underline{ANANA}@ \text{ và } \underline{ANA}@ \text{)}$$

$$l_4 = 0 \text{ (dài tỉn t chung dài nh t c a } BANANA@ \text{ và } ANANA@ \text{)}$$

$$l_5 = 0 \text{ (dài tỉn t chung dài nh t c a } NA@ \text{ và } BANANA@ \text{)}$$

$$l_6 = 2 \text{ (dài tỉn t chung dài nh t c a } \underline{NANA}@ \text{ và } \underline{NA}@ \text{)}$$

B n ch t c a m ng tỉn t chung dài nh t có th phân tích trên c u trúc c a cây h u t . V i hai h u t c a xâu T ng v i hai lá trong $ST(T)$, tỉn t chung dài nh t c a hai h u t là nhĩn c a nút tỉn b i chung th p nh t (*lowest common ancestor-LCA*) c a hai nút lá ó. Hình 9 là ví d v tỉn t chung dài nh t c a hai h u t $A@$ và $ANA@$, tỉn t này ng v i nút mang nhĩn A.



Hình 9. Tiền tố chung dài nh ất giữa hai hậu tố là nhĩn c ủa nút tỉn b ối chung th ấp nh ất.

Trên cây h u t , có r t nhi u thu t toán LCA có th áp d ng tìm nút tỉn b i chung g n nh t c a hai nút, m i truy v n LCA c th c hi n trong th i gian $O(1)$ và vì th có th xây d ng m ng tỉn t chung dài nh t trong th i gian $O(n)$. M c dù v y, Kasai

[8] đã xuất các thuật toán tuyến tính để tìm vị trí của ký tự và hiệu quả xây dựng mảng bit chung dài nhất của hai chuỗi.

3. Các thuật toán xây dựng cấu trúc dữ liệu

Các thuật toán trong chuyên đề này sẽ giới thiệu theo thứ tự sau: Trước tiên là thuật toán xây dựng mảng bit của chuỗi $SA(T)$ tiếp theo là thuật toán xây dựng mảng bit chung dài nhất $LCP(T)$, và cuối cùng là thuật toán xây dựng cây hậu tố $ST(T)$ từ $SA()$ và $LCP(T)$.

3.1. Xây dựng mảng bit

Bài toán: Cho chuỗi $T = t_0 t_1 \dots t_{n-1} \in \Sigma^+$ trong đó duy nhất $t_{n-1} = @$. Cần xây dựng mảng bit: $SA(T) = (a_0, a_1, \dots, a_{n-1})$:

$$T[a_0 \dots n-1] < T[a_1 \dots n-1] < \dots < T[a_{n-1} \dots n-1]$$

Mục đích của mảng bit là để tìm vị trí của các ký tự, vì việc áp dụng các thuật toán sắp xếp dựa trên phép so sánh chuỗi không phải là một phương pháp hay. Lý do chính là thời gian thực hiện phép so sánh hai chuỗi theo thứ tự từ trái sang phải của ký tự đầu chuỗi trong trường hợp xấu nhất. Hơn nữa các thuật toán hiệu quả dựa trên thời gian thực hiện trên chuỗi và chỉ dùng các phép toán trên ký tự. Trong chuyên đề này, ta định nghĩa mảng bit và vị trí ký tự đầu tiên trong chuỗi T , ký tự đầu tiên của $T = t_0 t_1 \dots t_{n-1}$ thì ký tự i là $t_i t_{i+1} \dots t_{n-1}$.

3.1.1. Thuật toán nhân đôi bit

Phương pháp nhân đôi bit xây dựng từ bit của mảng bit mà không cần dùng cây hậu tố có tên là thuật toán nhân đôi bit, được xuất bản bởi Manber và Myers [12]. Thuật toán này cho đến nay vẫn là cơ sở để phân tích trong các kỳ thi lập trình bài hai lý do:

- Việc cài đặt thuật toán khá đơn giản, thích hợp vì việc lập trình trong thời gian ngắn.
- Mục đích trong trường hợp xấu nhất, thuật toán cần thời gian $\Theta(n \log n)$ để xây dựng mảng bit, nhưng trung bình thuật toán chỉ cần thời gian $O(n)$ để thực hiện trong trường hợp phân bố đều của ký tự trong mảng bit.

Ý tưởng của thuật toán nhân đôi bit:

Khái niệm: Sắp xếp các ký tự của T theo thứ tự tăng dần của ký tự đầu tiên. Ký tự này tiếp theo sẽ là ký tự đầu tiên của các ký tự trong T theo thứ tự tăng dần. Sau đó ta gán

cho mỗi vị trí (mỗi vị trí) một khóa số nguyên $\in [0, n-1]$ thì đảm bảo: Hai vị trí có ký tự bằng nhau phải mang khóa bằng nhau, hai vị trí có ký tự khác nhau phải mang hai khóa khác nhau và vị trí nào có ký tự như nhau phải mang khóa như nhau. Việc gán khóa số mất thời gian $O(n)$. Khóa số là chỉ định cho ký tự đứng trước các vị trí, tức là dãy các vị trí xếp theo thứ tự tăng dần của khóa số cũng là dãy các vị trí theo thứ tự tăng dần của ký tự đầu tiên.

Phần chính của thuật toán sẽ thể hiện lập luận qua nhiều pha, tìm thấy là đã có dãy các vị trí xếp theo thứ tự tăng dần của l ký tự cùng các khóa số tăng dần với thứ tự xếp, thuật toán sẽ xây dựng dãy các vị trí xếp theo thứ tự tăng dần của $2l$ ký tự và dãy khóa số mới như sau:

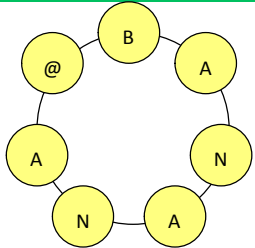
- Gọi các khóa đang gán cho các vị trí là các khóa số chính (primary keys), mỗi vị trí sẽ bổ sung một khóa phụ (secondary keys). Khóa phụ của một vị trí tại vị trí i chính bằng khóa số chính của vị trí ngay sau nó l vị trí ($i+l$) hoặc bằng -1 nếu $i+l \geq n$.
- Sắp xếp các vị trí theo quy tắc: Thứ tiên xếp tăng dần theo khóa số chính, nếu hai vị trí có khóa số chính bằng nhau thì vị trí nào có khóa phụ nhỏ hơn sẽ xếp trước. Theo dõi thứ tự dãy khóa số chính và cách xây dựng dãy khóa phụ, ta sẽ thu được dãy các vị trí xếp theo thứ tự tăng dần của $2l$ ký tự sau khi sắp xếp (khi hai vị trí có l ký tự khác nhau thì l ký tự sau sẽ dùng quy tắc như vị trí nào đứng trước).
- Với các vị trí đã sắp xếp, mỗi vị trí sẽ gán khóa số chính mới: Vị trí đứng đầu dãy sẽ gán số 0. Bắt đầu từ vị trí thứ hai trong dãy, nếu nó có khóa số chính và khóa phụ ngay vị trí liền trước thì khóa số chính mới của nó bằng khóa số chính của vị trí liền trước, nếu không thì khóa số chính mới của nó bằng khóa số chính của vị trí liền trước cộng thêm 1. Thao tác gán khóa số chính mất thời gian $O(n)$.

Như vậy các bước lập luận để xây dựng sẽ thể hiện các vị trí xếp theo 2, 4, 8, 16, 32, ... ký tự đầu tiên. Mỗi vị trí sẽ thu được sau bước p là $\lceil \lg n \rceil$. Thuật toán có thể dừng sớm tìm được bước nào đó mà tất cả các khóa gán cho các vị trí là các số nguyên hoàn toàn phân biệt từ 0 tới $n-1$ (các bước sau chắc chắn không còn thay đổi thứ tự xếp nữa).

Thời gian thực hiện ghi nhận thu thập thu vào thuật toán sắp xếp theo hai dãy khóa tìm kiếm. Có thể dùng các thuật toán sắp xếp so sánh, chẳng hạn như QuickSort. Tuy nhiên vì tập các giá trị khóa là các số nguyên nhỏ trong phạm vi từ 1 tới n , ta có thể áp dụng các thuật toán sắp xếp các số (*Radix Sort*) hoặc đếm phân phối (*Counting Sort*) để có độ phức tạp thời gian thực hiện ghi nhận thu thập là $O(n)$. Thuật toán nhân đôi từ đó có thể thực hiện trong thời gian $O(n \log n)$. **Cài đặt**

Cài đặt thuật toán nhân đôi từ đó cần ghi nhận và hiệu chỉnh, ta cần đảm bảo rằng việc xét có phải là phép toán chèn dữ liệu hợp lý.

Xét hoán vị vòng quanh $t_i t_{i+1} \dots t_{n-1} t_1 \dots t_{i-1}$ của chuỗi T tại vị trí i . Hoán vị vòng quanh này là ghép các chữ $t_{i \dots n-1}$ với từ $t_0 \dots i-1$. Có thể coi hoán vị vòng quanh các từ thành bảng cách vị trí các ký tự trong T quanh một vòng tròn theo chiều kim đồng hồ. n ký tự liên tiếp theo chiều kim đồng hồ từ vị trí i . Với vị trí i của ký tự c trong $@$, thì từ i đến các chữ của T sẽ là từ i đến c của các hoán vị vòng quanh từ i đến c . Vì vậy có thể suy ra rằng vì ký tự $@$ là ký tự nhỏ nhất trong bảng chữ cái và ký tự này chỉ xuất hiện một lần duy nhất trong chuỗi T . Bảng dưới đây là từ i đến c của các chữ của T sẽ là từ i đến c của hoán vị vòng quanh từ i đến c .

| | H u t | Hoán v vòng quanh |
|---|---------|-------------------|
|  | @ | @BANANA |
| | A@ | A@BANAN |
| | ANA@ | ANA@BAN |
| | ANANA@ | ANANA@B |
| | BANANA@ | BANANA@B |
| | NA@ | NA@BANA |
| | NANA@ | NANA@BA |

Bảng vị trí của hoán vị vòng quanh, khái niệm vị trí trong từ/câu sau l chữ số vị trí i luôn tồn tại trên vòng tròn nên ta không cần phải xử lý trường hợp riêng khi từ vị trí không tồn tại trong dãy.

Để tính toán hoán vị vòng quanh vị trí i của ký tự trong u . Giả sử u là một chuỗi l p, ta có dãy $(a_0, a_1, \dots, a_{n-1})$ là dãy các hoán vị vòng quanh x p theo thứ tự từ i đến c của l ký tự đầu tiên. Giả sử l ký tự đầu tiên của m hoán vị vòng quanh là $o n s c p$ và l ký tự tiếp theo của $o n s c p$ là $o n t h c p$.

Xây dựng dãy $B = (b_0, b_1, \dots, b_{n-1})$ như sau: b_i là vị trí trong từ l chữ số vị trí a_i trên vòng tròn. Đây chính là dãy hoán vị vòng quanh x p theo thứ tự từ i đến c của $o n t h c p$.

Tiếp theo, ta sắp xếp lại dãy $(b_0, b_1, \dots, b_{n-1})$ theo thứ tự tăng dần của khóa sắp xếp để tìm ra vị trí của l ký tự đầu tiên của $o n t h c p$ (chẳng hạn như thuật toán đếm phân

ph i) c dãy $(a_0, a_1, \dots, a_{n-1})$ m i. Trong thu t toán s p x p n nh, hai hoán v vòng quanh có o n s c p gi ng nhau thì hoán v nào ang ng tr c (có o n th c p nh h n) v n s ng tr c. T ó suy ra dãy $(a_0, a_1, \dots, a_{n-1})$ m i thu c chính là th t t i n c a các hoán v vòng quanh theo $2l$ ký t u tiên. Chú ý r ng ta không c n s d ng m t m ng nào ch a khóa th c p, khóa th c p c a m i v trí c suy ra t khóa s c p c a v trí ng sau nó l b c.

ti n l i cho v i c gán khóa s c p, ta có thêm m ng boolean $mark[0 \dots n - 1]$ ánh d u. Sau m i pha l p, $mark[i]$ s c t b ng True n u a_i ph i mang khóa s khác v i a_{i-1} . N u tr c pha l p $mark[i] = \text{True}$ t c là a_i có khóa s c p khác v i a_{i-1} thì sau b c l p ó a_i v n mang khóa s c p khác a_{i-1} , ta ch t thêm $mark[i] = \text{True}$ n u sau pha l p a_i mang khóa th c p khác v i a_{i-1} ;

D i ây là o n ch ng trình tính m ng h u t theo thu t toán nhân ôi t i n t

Input:

Dòng 1: dài xâu T ($n \leq 10^5$)

Dòng 2: Xâu T (có ký t c m canh @ ng cu i)

Output

M ng h u t c a T

| Sample Input | Sample Output |
|--------------|---------------|
| 7 BANANA@ | 6 5 3 1 0 4 2 |

SUFFIXARRAY.PAS Tính m ng h u t

```
{ $MODE OBJFPC }
program SuffixArrayConstruction;
const maxN = 100000;
type
  TAlphabet = '@'..'Z';
var
  T: array [0..maxN - 1] of TAlphabet;
  n: Integer;
  key, head, a, b: array [0..maxN - 1] of Integer;
```

```

mark: array [0..maxN - 1] of Boolean;
procedure Init; //Khởi tạo
var   i: Integer;
c: TAlphabet;
ccount: array[TAlphabet] of Integer;
begin
  ReadLn(n);
  for i := 0 to n - 1 do Read(T[i]);
//Thuật toán đếm phân phối, tạo dãy a[0...n-1] là các hoán vị vòng quanh xếp theo
chữ cái đầu
  FillChar(ccount, SizeOf(ccount), 0);
  for i := 0 to n - 1 do Inc(ccount[T[i]]);
  for c := Succ(Low(TAlphabet)) to High(TAlphabet) do
    Inc(ccount[c]ccount[pred(c)]);
  for i := n - 1 downto 0 do
  begin
    c := T[i];
    Dec(ccount[c]);
    a[ccount[c]] := i;
  end;
//Khởi tạo mảng mark[0...n-1]
  mark[0] := True;
  for i := 1 to n - 1 do
    mark[i] := T[a[i]] <> T[a[i - 1]]; //a[i] phải mang khóa khác
a[i-1] end;

procedure SuffixArray; //Thuật toán nhân đôi tiền tố
var
  i, j, nkeys, kv: Integer;
len: Integer;
begin
  len := 1;
  while len < n do
  begin
//Trước mỗi pha lặp đã có a[0...n-1] là dãy các HVVQ xếp theo đoạn sơ cấp gồm k ký
tự đầu
//Dựa vào mảng mark tính các giá trị khóa sơ cấp và tạo b[0...n-1] là dãy các
HVVQ xếp theo đoạn thứ cấp
    nkeys := 0;
    for i := 0 to n - 1 do

```

```

begin
    if mark[i] then //Tính luôn head[nkeys] = i là con trỏ tới vị trí đầu
tiên sẽ mang khóa sơ cấp nkeys
begin
        head[nkeys] := i;
Inc(nkeys);
end;
        key[a[i]] := nkeys - 1;
        b[i] := (a[i] - len + n) mod n; //b[i] = hvvq đứng trước vị
trí i đúng len bước
    end;
    if nkeys = n then Break; //Nếu các khóa sơ cấp hoàn toàn phân
biệt thì xong
    for i := 0 to n - 1 do
begin
kv := key[b[i]];
a[head[kv]] := b[i];
Inc(head[kv]);
end;
//a[0...n-1] giờ là dãy hvvq xếp theo 2len ký tự đầu, cập nhật lại mảng mark
        kv := -1;
for i := 0 to n - 1 do
begin
        j := (a[i] + len) mod n; //j đứng sau a[i] đúng len bước,
key[j] là khóa thứ cấp của a[i]
if key[j] <> kv then //key[j] = khóa thứ cấp của a[i] khác với khóa thứ cấp
của a[i-1]
begin
                mark[i] := True; //Đặt mark để đánh dấu
                kv := key[j];
end;
        end;
        len := len shl 1; //Nhân đôi độ
dài tiền tố
end;
end;
procedure PrintResult; //In kết quả
var i: Integer;
begin
    for i := 0 to n - 1 do Write(a[i], ' ');

```

```

    WriteLn;
end;
begin
    Init;
    SuffixArray;
    PrintResult;
end.

```

3.1.2. Thuật toán chia trị

Một trong những kỹ thuật quan trọng được dùng trong thuật toán nhân ôi ti n t ó là thay thế vì số so sánh xâu b i phép so sánh số nguyên. Kỹ thuật này gọi là mã hóa (*encoding*) bằng khóa s .

xây dựng mảng h u t , còn có hai thuật toán khác được phát triển để tính nh ng u d a trên kỹ thuật chia trị và cách mã hóa bằng khóa s . Tuy cách tiếp cận có khác nhau nhưng chúng đều là các thuật toán tuyến tính (còn về thời gian và bộ nhớ). Ngoài việc tính toán kho ng tr ng lý thuyết², những thí nghiệm trên dữ liệu thực tế cho thấy việc tìm rõ ràng các phép pháp này so với thuật toán nhân ôi ti n t khi xử lý dữ liệu lớn.

Thuật toán của Kärkkäinen và Sanders (2003) [7]: Tìm kiếm, hai phần ba số h u t s được mã hóa theo 3 ký tự và các số p x p b ng quy, một phần ba còn lại các số p d a vào một mảng t i n c a s h u t ã s p sau đó chuyển ngược lại. Kích thước dữ liệu giảm xuống còn 2/3 sau mỗi bước. Thời gian thực thi thuật toán là $T(n) = T\left(\frac{2n}{3}\right) + O(n) = O(n)$.

Thuật toán của Ko và Aluru (2003) [10]: Tìm kiếm, các h u t được phân làm hai loại: Loại 1 là các h u t t i v trí i mà nh h n h u t t i v trí $i + 1$ và loại 2 là các h u t t i v trí i mà l n h n h u t t i v trí $i + 1$. Các tác giả chứng minh rằng các số p x p nằm trong hai loại thì các thuật toán tuyến tính là có thể chuyển các h u t còn lại vào thành toàn bộ mảng h u t . Thời gian thực thi thuật toán quy về bài toán số p x p t i a $\left\lfloor \frac{n}{2} \right\rfloor$ h u t . Thời gian thực thi thuật toán là

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + O(n) = O(n).$$

² Vì thuật toán dựng cây hậu tố trực tiếp là thuật toán tuyến tính nhưng khá phức tạp, người ta đã tìm được thuật toán tuyến tính đơn giản hơn nhưng cần dựng cây hậu tố một cách gián tiếp qua mảng hậu tố. Tuy vậy điều này sẽ không còn ý nghĩa lý thuyết nếu như thuật toán xây dựng mảng hậu tố không phải là thuật toán tuyến tính.

Khuôn khổ của chuyên đề không cho phép chúng tôi trình bày toàn bộ chứng minh và mã nguồn của hai thuật toán này. Bên cạnh có thể tham khảo bài báo gốc và chỉ tiết của chúng. Chúng ta thấy rằng đã có thuật toán xây dựng mảng h và u trong thời gian và bộ nhớ tuyến tính.

3.2. Xây dựng mảng tỉ lệ chung dài nhất.

Bài toán: Cho chuỗi $T = t_0 t_1 \dots t_{n-1} \in \Sigma^+$ trong đó duy nhất $t_{n-1} = @$ cùng với mảng h và u tính được $SA(T) = (a_0, a_1, \dots, a_{n-1})$. Cần phải xây dựng mảng tỉ lệ chung dài nhất $LCP(T) = (l_0, l_1, \dots, l_{n-1})$:

- $l_0 = 0$,
- $\forall i > 0, lcp_i$ là tỉ lệ chung dài nhất giữa hai chữ a_i và a_{i-1} .

3.2.1. Thuật toán

Thuật toán phân biệt nhất xây dựng mảng $LCP(T)$ của xuất bản Kasai [8]. Sau đây ta trình bày thuật toán đó.

Trước hết ta tính mảng $rank_{0 \dots n-1}$ trong đó $rank_i$ là vị trí của chữ i trong mảng h và u . Tức là:

$$rank_i = j \Leftrightarrow a_j = i$$

Mảng $(l_0, l_1, \dots, l_{n-1})$ sẽ được xây dựng theo thuật: $l[rank_1], l[rank_2], \dots, l[rank_n]$, trong đó và $l[rank_{i+1}]$ sẽ được tính dựa vào $l[rank_i]$.

Với mỗi giá trị i , gọi $q = l[rank_i]$, và $j = a[rank_i - 1]$ là chữ đứng liền trước chữ i trong mảng h và u . Theo định nghĩa về mảng $LCP(T)$ ta có q là tỉ lệ chung dài nhất giữa hai chữ i và j . Lo ngại ký tự ưu tiên của hai chữ này, ta có: tỉ lệ chung dài nhất giữa hai chữ $i+1$ và $j+1$ có thể dài $q-1$ nếu $q \geq 1$ và bằng 0 trong trường hợp ngược lại.

Nếu $q \geq 1$, ta có $t_i = t_j$. Mặt khác, chữ j đứng liền trước chữ i trong mảng h và u nên nếu lo ngại ký tự ưu tiên nhau hai chữ này thì thu được hai chữ m và $i+1$ và $j+1$ trong đó chữ $j+1$ và n và ngược lại chữ $i+1$

trong mảng h và u . Xét trên thuật toán này, chữ $j+1$ có thể không đứng liền trước chữ $i+1$ nhưng tỉ lệ chung dài nhất của chúng có thể dài $q-1$, mà chữ

n mà giả sử chúng theo thứ tự tăng dần thì có $q-1$ ký tự trùng với vị trí $j+1$ của mảng t và vị trí $i+1$. Vì vậy đây chỉ là một vị trí trùng với vị trí $i+1$ có ít nhất $q-1$ ký tự trùng với vị trí $i+1$ trong mảng t hay $l[\text{rank}_{i+1}] \geq$

$q-1$. Do điều kiện này đúng với vị trí $q=0$. Vậy ta có bảng sau:

Bảng 3

Vì $\forall i: 0 \leq i < n-1$, ta có $l[\text{rank}_{i+1}] \geq l[\text{rank}_i] - 1$.

Bảng 3 cho phép ta cài đặt thuật toán tính mảng l như sau:

```

l[0] := 0; q:=0;
for i := 0 to n - 2 do
begin
    j := a[rank[i] - 1]; //j là hậu tố đứng liền trước i trong mảng hậu tố
    while t[i + q] = t[j + q] do
        q := q + 1; //Tăng q nếu ký tự thứ q+1 của hậu tố i và hậu tố j khớp nhau
    l[rank[i]] := q; //Do ký tự thứ q+1 của hậu tố i và hậu tố j khác nhau
    if q > 0 then q := q - 1; //Giảm q chuẩn bị cho bước sau
end;
```

Vì T có duy nhất một ký tự đặc biệt nên hình thành nên một ký tự khác, vị trí $n-1$ của T chỉ chứa một ký tự đặc biệt trong các vị trí và vị trí y $l[\text{rank}_{n-1}] = l[0] = 0$ và ta không cần tính $l[\text{rank}_{n-1}]$ nữa. Mặt khác, thuật toán tính $l[\text{rank}_i]$ với vị trí i mà vị trí i chỉ chứa một ký tự đặc biệt là $l[\text{rank}_i] \geq q$. Vì j là vị trí mà vị trí j trong thứ tự tăng dần của mảng t , khi đó hai vị trí này có ít nhất q ký tự trùng nhau. Tăng q lên đến khi ký tự thứ $q+1$ của hai vị trí i và j không khớp, ta có $l[\text{rank}_i] = q$. Vì vị trí i cùng là vị trí $q-1$ nên (nếu $q > 0$) chuẩn bị cho bước sau (tính $l[\text{rank}_{i+1}]$).

Thời gian thực hiện của thuật toán có thể đánh giá qua số lần giá trị q tăng hoặc giảm bởi hai lần $q := q-1$ và $q := q+1$. Giá trị của q chỉ tăng hoặc giảm 0. Vòng lặp while chỉ thực hiện tìm kiếm giá trị q mà ký tự thứ $q+1$ của hai vị trí i và j không khớp nhau ($t[i+q] \neq t[j+q]$), do đó nên giảm $q := q-1$ không thể làm cho q vượt quá $n-1$. Ngoài ra có tối đa $n-1$ lần q giảm (bởi lần $q := q-1$) sau mặt khác, q

suy ra 1 nh t ng q ($q := q + 1$) có s l n th c hi n không v t quá $2n - 2$. V y thu t toán Kasai có th đ ng c m ng $LCP(T)$ trong th i gian $O(n)$.

3.2.2.Cài t

Input:

- Dòng 1: đ ài xâu T ($n \leq 10^5$)
- Dòng 2: Xâu T (có ký t c m canh @ ng cu i).
- Dòng 3: Các giá tr a_0, a_1, \dots, a_{n-1} ng v i m ng h u t $SA(T)$

Output

M ng ti n t chung đ ài nh t $LCP(T)$

| Sample Input | Sample Output |
|-------------------------------|---------------|
| 7 BANANA@ 6 5 3 1 0 4 2 | 0 0 1 3 0 0 2 |

LCPARRAY.PAS Tính m ng ti n t chung đ ài nh t

```
{ $MODE OBJFPC } program
LCPArrayConstruction;
const maxN = 100000;
type
  TAlphabet =
    '@'..'Z';
var
  t: array[0..maxN - 1] of
    TAlphabet;
  a, rank, l: array[0..maxN - 1] of Integer;
  n: Integer;
procedure Enter;
var i:
  Integer;
begin
  ReadLn(n);
```

```

    for i := 0 to n - 1 do
    Read(t[i]);
    ReadLn;
    for i := 0 to n - 1 do
    Read(a[i]); end;

procedure LCPArray; //Tính mảng tiền tố chung dài nhất
var    i, j, q:
Integer; begin
    for i := 0 to n - 1 do rank[a[i]] := i; //Tính hạng
của mỗi hậu tố
    l[0] := 0;
    q := 0;
    for i := 0 to n - 2 do
    begin
        j := a[rank[i] - 1]; //j là hậu tố đứng liền trước i trong thứ tự từ điển
của mảng hậu tố
        while t[i + q] = t[j + q] do Inc(q);
        l[rank[i]] := q; //Do ký tự thứ q + 1 của hậu
tổ i và hậu tố j khác nhau
        if q > 0 then Dec(q); //Giảm q chuẩn bị cho bước sau: tính
l[rank[i + 1]]
    end;
    end;
procedure
PrintResult; var
i: Integer;
begin
    for i := 0 to n - 1 do Write(l[i], ' ');
    WriteLn;
end;
begin
Enter;
    LCPArray;
PrintResult;
end.

```

3.3. Xây dựng cây hậu tố

Bài toán: Cho xâu $T = t_0 t_1 \dots t_{n-1} \in \Sigma^+$ trong đó có duy nhất $t_{n-1} = @$. Cho mảng $SA(T) = (a_0, a_1, \dots, a_{n-1})$ và mảng tỉ lệ chung dài nhất $LCP(T) = (l_0, l_1, \dots, l_{n-1})$ của T . Cần xây dựng cây hậu tố của T : $ST()$.

3.3.1. Thuật toán

Ý tưởng của thuật toán là chèn lần lượt các hậu tố trong mảng hậu tố vào cây. Bắt đầu từ một cây chỉ gồm nút gốc, thuật toán lần lượt xét các hậu tố theo thứ tự tăng dần. Mỗi khi chèn một hậu tố, có thể có một nút nhánh và một nút lá được bổ sung vào cây, trong đó nút lá có nhãn là hậu tố vừa chèn.

Có thể thấy, ta bắt đầu từ một cây chỉ gồm nút gốc và nút lá có nhãn x để bắt đầu quá trình xây dựng. Với nút x sẽ có một nhánh lá mới được chèn vào cây tại vị trí i . Xét lần lượt các hậu tố trong mảng hậu tố. Mỗi khi xét tới hậu tố a_i , ta đi từ nút lá có nhãn x lên phía gốc cho tới khi gặp một nút u có $depth(u) \leq l_i$. Gọi v là nút con của u đi qua trong quá trình di chuyển từ x lên u .

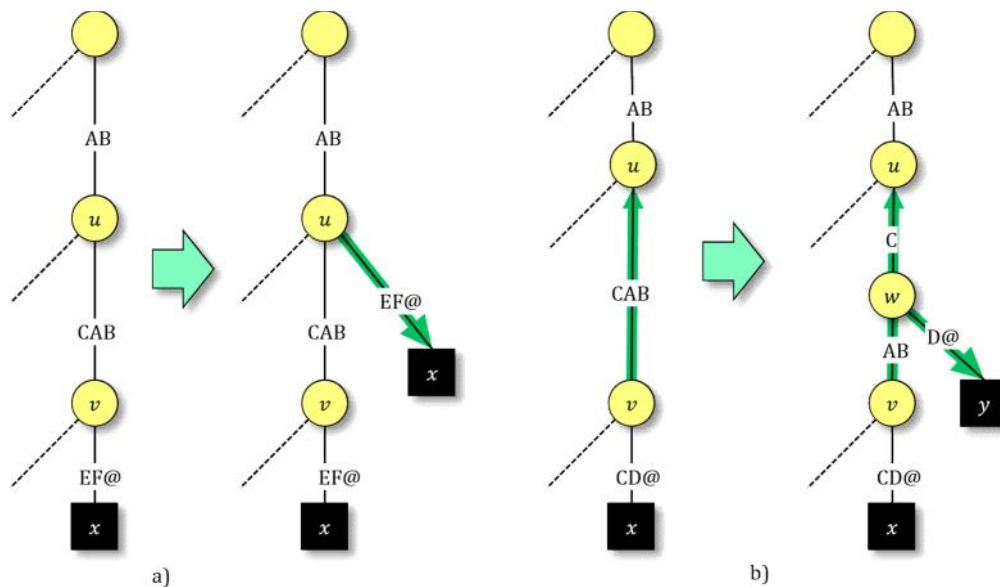
Trường hợp 1: Nếu $depth(u) = l_i$, ta chia nút u thành hai nút lá mới y và z làm con của u và gán nhãn cho (u, y) sao cho nhãn cho nhánh này là a_i (nhãn cho nhánh này bằng $t[a_i + depth(u) \dots n - 1]$). Hình 10 a) là ví dụ về cây hậu tố được chèn

ABCAEF@ và sau đó chèn thêm ABEF@, trong ví dụ này ta tìm được nút u có $\bar{u} = AB$ là tỉ lệ chung dài nhất của ABCAEF@ và ABEF@

Trường hợp 2: Nếu $depth(u) < lcp_i$:

- Thêm nút w làm con của u và cha của v tách nhánh (u, v) làm hai nhánh (u, w) và (w, v) . Nhãn cho nhánh (u, v) được tách ra hai phần: $l_i - depth(u)$ ký tự đầu tiên của u để làm nhãn cho nhánh (u, w) , các ký tự sau để làm nhãn cho nhánh (w, v)
- Nút w có sâu đúng bằng lcp_i , đặt $u := w$, vì cần quy về trường hợp 1.

Hình 10 b) là một ví dụ về cây hậu tố được chèn ABCABCD@ và sau đó chèn thêm ABCD@, quá trình đi lên để tìm nút u như $\bar{u} = AB$ chia phần đầu tiên của tỉ lệ chung dài nhất của ABCABCD@ và ABCD@ (bằng ABC). Ta chia nhánh (u, v) ra làm hai phần bởi nút mới w sao cho $\bar{w} = ABC$ sau đó bổ sung nút lá y làm con của w với nhãn cho nhánh (w, y) là D@ để $\bar{y} = ABCD@$.



Hình 10. Chèn ABCGH@ vào cây hậu tố

Không khó khăn để chứng minh giải thuật dựng cây hậu tố từ mảng hậu tố và mảng tỉ lệ chung dài nhất trên có thể thực hiện trong thời gian $O(n)$ vì nó chỉ thực hiện phép duyệt các cạnh trên cây:

- Mỗi cạnh chỉ được duyệt qua tối đa một lần trong phép di chuyển từ lá lên gốc (chỉ qua các cạnh cùng có thể bị xóa và thay bằng 2 cạnh mới)
- Mỗi phép chèn hậu tố vào cây làm tăng số cạnh lên nhiều nhất là 2.

3.3.2. Cài đặt

Input:

- Dòng 1: Chiều dài chuỗi $T (n \leq 10^5)$
- Dòng 2: Chuỗi T (có ký tự đặc biệt '@' để kết thúc).
- Dòng 3: Các giá trị a_0, a_1, \dots, a_{n-1} để tính mảng hậu tố $SA(T)$
- Dòng 4: Các giá trị l_0, l_1, \dots, l_{n-1} để tính mảng tỉ lệ chung dài nhất $LCP(T)$

Output

Cây hậu tố $ST(T)$

| Sample Input | Sample Output |
|---------------|---------------|
| 7 | --ROOT |
| BANANA@ | { |
| 6 5 3 1 0 4 2 | --@ |
| 0 0 1 3 0 0 2 | --A |
| | { |
| | --@ |
| | --NA |
| | { |
| | -- |
| | @ |
| | -- |
| | NA@ |
| | } |
| | } |
| | --BANANA@ |
| | --NA |
| | { |
| | --@ |
| | --NA@ |
| | } |
| | } |

Trong chương trình cài đặt ở đây, ta tổ chức các nút của cây hậu trong mảng *nodes*. Mỗi nút là một bộ ghi gồm các trường:

- *indexL, indexH*: Nhận các chỉ số nút cha là xâu $t[indexL \dots indexH - 1]$
- *depth*: Sâu của nút
- *parent*: Chỉ số nút cha
- *child['@' ... 'Z']*: *child[c]* là chỉ số nút con ứng với các nhận chỉ có ký tự là *c*.

SUFFIXTREECONSTRUCTION.PAS Dạng cây hậu

```
{ $MODE OBJFPC }
program
SuffixTreeConstruction;
const  maxN = 100000;
```

```

type
TAlphabet = '@'..'Z';
TNode = record //Cấu
trúc nút
    indexL, indexH: Integer; //Nhãn cạnh nối từ nút cha là
t[indexL...indexH - 1]
    depth: Integer; //Độ sâu
    parent: Integer; //Chỉ số nút cha
    child: array[TAlphabet] of Integer; //Chỉ số các nút con
end;
var T: array[0..maxN - 1] of TAlphabet;
a, l: array[0..maxN - 1] of Integer;
nodes: array[1..2 * maxN] of TNode;
n: Integer; x, nodeptr: Integer;
procedure Enter; //Nhập dữ liệu
var i: Integer;
begin ReadLn(n);
    for i := 0 to n - 1 do Read(t[i]);
ReadLn;
    for i := 0 to n - 1 do Read(a[i]);
ReadLn;
    for i := 0 to n - 1 do
Read(l[i]); end;
function NewNode: Integer; //Tạo nút mới có chỉ số nodeptr
begin
Inc(nodeptr);
FillChar(nodes[nodeptr], SizeOf(nodes[nodeptr]), 0);
Result := nodeptr;
end;
procedure Init; //Tạo cây chỉ gồm nút gốc chỉ số 1
begin
nodeptr := 0;
x := NewNode;
end;
procedure SetLink(u, v: Integer); //Cho v
là con của u var ch: TAlphabet;
begin
ch := T[nodes[v].indexL]; //Đọc ký tự đầu nhãn cạnh
nodes[v].parent := u;
nodes[u].child[ch] := v;
end;

```

```

//TT chính: Chèn hậu tố suffindex vào cây, biết độ dài tiền tố chung dài nhất của hậu
tổ này với hậu tố vừa chèn là lcpvalue
procedure InsertSuffix(suffindex, lcpvalue:
Integer); var u, v, w, y: Integer;
k, p: Integer;
begin
u := x;
while nodes[u].depth > lcpvalue do //Đi từ lá x lên gốc, tìm nút u có
độ sâu ≤ lcpvalue
begin
v := u;
u := nodes[v].parent;
end;
if nodes[u].depth < lcpvalue then //Nếu u có độ sâu
< lcpvalue
begin
//Tạo nút w có độ sâu lcpvalue chèn vào giữa cạnh (u, v), nhãn (u, w) nối với
nhãn (w, v) = nhãn (u, v) cũ
w := NewNode;
k := nodes[v].indexL;
p := lcpvalue - nodes[u].depth;
nodes[w].indexL := k;
nodes[w].indexH := k + p;
nodes[w].depth := lcpvalue;
nodes[v].indexL := k + p;
SetLink(u, w);
setLink(w, v);
u := w;

end;
//u có độ sâu bằng lcpvalue, tạo lá y làm con của u
y := NewNode;
with nodes[y] do
begin
indexL := suffindex + lcpvalue;
indexH := n;
depth := n - suffindex; //Độ sâu của y bằng chiều dài hậu tố
end;
SetLink(u, y);
x := y; //Cập nhật lá cực phải mới
end;

```

```

procedure SuffixTree; var    i:
Integer; begin
    for i := 0 to n - 1 do
        InsertSuffix(a[i], l[i]); //Chèn lần lượt các hậu tố theo thứ tự từ
điển vào cây
    end;
//Các thủ tục trình bày output, không quan trọng
procedure WriteBlank(nb: Integer);
var    i: Integer;
begin
    for i := 1 to nb do Write(' ');
end;
procedure Visit(i: Integer; indent: Integer);
var    ch: TAlphabet;    j: Integer;
begin
    if i = 0 then Exit;
    WriteBlank(indent);
    Write('--');
    if i = 1 then Write('ROOT')    else    with
nodes[i] do
        for j := indexL to indexH - 1 do Write(t[j]);
    WriteLn;
    if nodes[i].indexH <> n then
    begin
        WriteBlank(indent + 2); WriteLn('{');
        for ch := Low(TAlphabet) to High(TAlphabet) do
            Visit(nodes[i].child[ch], indent + 4);
        WriteBlank(indent + 2); WriteLn('}');
    end;
end;
procedure PrintResult;
begin
    Visit(1, 0);
end;
begin
    Enter;
    Init;
    SuffixTree;
    PrintResult;
end.

```


II/ Bài tập áp dụng:

4.1. Mật mã n (ACM 2003)

Cho chuỗi dài $n \leq 10^5$, tìm hoán vị vòng quanh có thể tồn tại nhiều nhất. Ví dụ với chuỗi "ALABALA" thì hoán vị vòng quanh nhiều nhất là "AALABAL" Thuật toán:

Mặc dù có thuật toán $O(n)$ cài đặt đơn giản hơn, vì áp dụng các thuật toán đếm mảng hữu hạn là một giải pháp không nhiều công sức suy nghĩ. Chú ý là vì không cần dùng kỹ thuật cạnh có thể đếm vài giây trong cài đặt thuật toán

4.2. Sâu con phân biệt (IOI training camp 2003)

Bạn cho chuỗi T dài không quá 10^5 , cho biệt có bao nhiêu sâu con khác nhau và khác rỗng của T .

Thuật toán

Bài toán nhận thu nhận là đếm số nút trên trie hữu hạn ngược lại, đếm ngược lại vì đếm số trên trie. Vấn đề đếm ngược lại và thời gian khi xây dựng trie có thể phức tạp bằng cây hữu hạn và tìm kiếm. Tuy nhiên cách hay nhất là dùng mảng liên tiếp chung dài nhất.

Giả sử ta có mảng hữu hạn $(a_0, a_1, \dots, a_{n-1})$ và mảng liên tiếp chung dài nhất (l_0, l_1, \dots, l_n) . Chèn lần lượt các hữu hạn vào trie theo thứ tự, phân tích quá trình hữu hạn a_i (đài $n - a_i$) chèn vào trie, l_i ký tự duy nhất qua mà không có sự bổ sung nút và cạnh. Nhưng ký tự sau, mới ký tự bổ sung lần cạnh và 1 nút trên trie. Suy ra áp dụng là:

$$\sum_{i=0}^{n-1} (n - a_i - l_i) = \frac{n(n+1)}{2} - \sum_{i=0}^{n-1} l_i$$

(Do $\sum_{i=0}^{n-1} (n - a_i) = \frac{n(n+1)}{2}$)

4.3. Sâu con (Training Camp 2003)

Cho chuỗi T gồm n ký tự ($n \leq 10^5$) và một số $k \leq n$, tìm sâu con dài nhất xuất hiện trong chuỗi T ít nhất k lần.

Thuật toán:

Ta sẽ trình bày phương pháp dùng trie h u t , trên cây h u t c ng có thể dùng phương pháp t ng t . Tuy nhiên bài toán này có thể giải bằng cách n g i n b ng m ng h u t và m ng t i n t chung g n nh t. G i ý: i u k i n có m t xâu dài q xuất hiện k lần trong xâu T là trong m ng $LCP(T)$ t n t i $k - 1$ s liên tiếp $\geq q$.

4.4. Xâu con i x ng dài nh t (USACO training gate)

Cho xâu S dài $n \leq 10^5$, tìm xâu con i x ng dài nh t.

Thu t toán:

G i là xâu o ng c c a xâu S , nh n xét r ng m t xâu i x ng dài l c a S có ký t n g g i là s_i s ph i có đ ng $\bar{A}s_iA$ trong ó A là m t xâu con c a S b t u t v trí $i + 1$ và c ng là xâu con c a S' b t u t i v trí $n - i - 1$. D ng xâu $T = S + \bar{S}$ và m ng h u t $SA(T)$, khi ó:

- M i xâu con c a S b t u t v trí i ph i là t i n t c a h u t th i c a T .
- M i xâu con c a \bar{S} b t u t v trí $n - i$ ph i là t i n t c a h u t th $2n - 1 - i$ c a T .

V n quy v tìm t i n t chung dài nh t g i a hai h u t c a T . Trên m ng h u t $SA(T) = (a_0, a_1, \dots, a_{2n-1})$ và m ng $LCP(T) = (l_0, l_1, \dots, l_{2n-1})$. T i n t chung dài nh t g i a h u t a_i và h u t a_j ($i < j$) là giá tr nh nh t trong các giá tr

$lcp[i + 1 \dots j]$, truy v n giá tr nh nh t trong m t kho ng liên ti p (*range-minimum query*) có thể c th c hi n trong th i gian $O(\log n)$ b ng c u trúc d li u segment trees ho c th c hi n trong th i gian $O(1)$ b ng phép quy đ n LCA hay Bucket Pointers. Khi xét trên m i v trí i , thu t toán tìm xâu con i x ng dài nh t dài l m t th i gian $O(n \log n)$ ho c $O(n)$ tùy theo c u trúc d li u c l a ch n.

V n t ng t trong v i c tìm xâu con i x ng dài nh t dài ch n.

4.5. M u ghép (Polish Olympiad in Informatics 2004)

Cho xâu X dài $n \leq 10^5$, tìm xâu Y ng n nh t sao cho m i ký t trong X u t n t i m t xâu con nào ó c a X úng b ng Y ch a v trí ký t ó. Hay nói cách khác, X là m t phép ghép g i c a m t lo t các xâu Y

ababbababbababbabaababba

ba (X) ababbaba (Y)

ababbaba ababbaba

ababbaba Thu t toán

Cách gì là s d ng m ng h u t $SA(X) = (a_0, a_1, \dots, a_{n-1})$ k t h p v i m t c u trúc d l i u truy v n ph m vi. B t u v i $Y = X_0$. Nh ng h u t có ký t u b ng X_1 s n m trong m t kho ng liên ti p trong m ng h u t (t v trí L t i v trí H). V trí ban u c a các h u t này trong xâu c ánh d u b i s 1, nh ng v trí khác c ánh d u b i s 0.

L n l t thêm các ký t X_1, X_2, \dots, X_{n-1} vào Y . M i khi Y dài thêm m t ký t , s có thêm nh ng h u t c a X không còn nh n Y làm t i n t n a, ta co ng n ph m vi ho t ng $[L, H]$ l i và ánh d u v trí trong xâu c a các h u t n m ngoài ph m vi ho t ng b i s 0. Thu t toán s d ng ngay khi t i m t b c mà dài xâu $Y \geq$ d ãy nhi u s 0 liên ti p nh t. V i c o dài d ãy g m nhi u s 0 liên ti p nh t có th th c hi n trong th i gian $O(\log n)$ b ng m t c u trúc d l i u truy v n ph m vi nh segment trees. Toàn b thu t toán có ph c t p $O(n \log n)$.

4.6. Liên k t h u t (suffix links)

Th c ra trong c u trúc c a cây h u t , còn có m t thành ph n n a g i là các liên k t h u t (suffix links). M i nút nhánh u c a cây h u t ch a m t con tr t i m t nút v khác sao cho n u $\bar{u} = \alpha S$ thì $\bar{v} = S$ (ây α là m t ký t còn S là m t xâu)

T t c các thu t toán tuy n tính d ng cây h u t tr c ti p theo t i bi t u ph i s d ng liên k t h u t . Tuy nhiên n u ta d ng cây h u t t m ng h u t và m ng t i n t chung dài nh t, các liên k t h u t b b qua.

Các liên k t h u t ôi khi r t quan tr ng trong m t s thu t toán x lý xâu. V i v y ta t v n : cho cây h u t $ST(T)$ c a m t xâu T dài n , c n ph i xây d ng toàn b các liên k t h u t .

G i i pháp:

Gán cho m i nút nhánh u c a cây h u t m t c p (i, j) th a m ãn: i, j là hai h u t ng v i hai lá n m hai nhánh con khác nhau c a u . V i c gán c p (i, j) cho t t c các nút nhánh có th th c hi n trong th i gian $O(n)$ b ng thu t toán duy t cây t d i lên: gán c p h u t cho nút u , ta gán c p h u t cho t t c các con c a u tr c b ng quy. Sau ó ch n u_1, u_2 là hai con b t k c a u , g i s c p h u t gán cho u_1 là (i_1, j_1) và c p h u t gán cho u_2 là (i_2, j_2) , khi ó ta có th l y c p (i_1, i_2) làm c p lá ng v i

u .

V i c p h u t (i, j) c a nút u , chúng ng v i hai lá n m hai nhánh con khác nhau mà u là t i n b i chung th p nh t c a c a hai lá ó, v y nên t i n t chung dài nh t c a

hai nút i và j chính là \bar{v} mà ta ký hiệu là αS . Ngược lại, tồn tại chung dài nhất của hai nút $i+1$ và $j+1$ phải là S . Xác định hai lá chĩa nút $i+1$ và $j+1$ và v là tồn tại chung thặng dư của hai lá này. Ta có $\bar{v} = S$ là con trùng liên kết từ u phải từ i và v .

Có thuật toán tìm tồn tại chung thặng dư của hai nút trong thời gian $O(1)$, bằng cách các thuật toán xuất bản của Tarjan [6] hay Fisher [2]. Vì cây có $O(n)$ nút, vì vậy thời gian để tính toán tất cả các liên kết có thể thực hiện trong thời gian $O(n)$. Bên cạnh đó có thể tham khảo thêm trong các tài liệu về hai bài toán LCA và RMQ và mối liên hệ giữa chúng.

4.7. Xâu con chung dài nhất

Bài toán tìm xâu con chung dài nhất (*longest common substring*) là một bài toán quan trọng trong xử lý xâu. Tên của bài toán đã nêu lên nội dung của nó: Cho hai xâu X, Y , tìm xâu Z dài lớn nhất mà là xâu con của X , và là xâu con của Y .

Thuật toán 1

Giả sử $X = x_1x_2 \dots x_m$ và $Y = y_1y_2 \dots y_n$

Duyệt cây hậu tố của Y . Bắt đầu từ gốc, ta duyệt các ký tự trong X và rẽ xuống nhánh con tiếp theo trên cây hậu tố. Nếu tìm thấy nút u nào đó không có nhánh con tiếp theo rẽ xuống, ta ghi nhận nút u cùng với độ sâu của nó (\bar{v} là xâu con dài nhất của X khớp với tiền tố xâu Y), tiếp theo ta nhảy theo liên kết từ u sang nút v (xuất hiện trong Y tại vị trí y_2) và tiếp theo cách như vậy.

Sau khi duyệt hết xâu Y , nút có độ sâu lớn nhất ghi nhận sẽ có nhãn là xâu con chung dài nhất cần tìm.

phức tạp tính toán: $O(m + n)$.

Thuật toán 2

Bổ sung thêm ký tự chấm $\$$. Xét xâu $X@Y\$$, nhận xét cây hậu tố có vị trí ngay sau vị trí $@$ chính là hậu tố xanh (hậu tố của $Y\$$) và nhận xét cây hậu tố khác chính là hậu tố đỏ. Duyệt cây hậu tố $X@Y\$$ trong đó các lá con cùng màu với vị trí tương ứng.

Bài toán trở thành tìm nút sâu nhất mà nhánh cây gốc đến các lá xanh và lá đỏ.

phức tạp tính toán: $O(m + n)$. Có thể mở rộng tìm LCS của nhiều xâu.

Thuật toán 3

Tô màu h u t t n g t nh thu t toán 2. D n g m n g h u t và m n g t i n t chung dài n h t c a $X@Y\$$. Trên m n g h u t , tìm v trí i sao cho h u t a_i và h u t n g l i n t r c (a_{i-1}) khác màu, ch n v trí i có $lcp_i \geq 1$ n h t.

phức tạp tính toán: $O(m + n)$.

5. K t l u n

Cây h u t , m ng h u t và m ng ti n t chung dài nh t là nh ng c u trúc d li u có m i liên h ch t ch . Ngoài vi c cung c p n hi u phép toán quan tr ng trong x lý xâu, nh ng k thu t hay c áp d ng trong quá trình xây d ng c u trúc d li u c ng r t áng chú ý.

M t trong nh ng h ng nghiên c u c quan tâm là s d ng cây h u t x lý d li u thu c m t b ng ch cái l n. C u trúc nút c a cây có th tr nên r t c ng k nh n u b ng ch cái Σ l n. Nh ví d trong chuyên này, m i nút ph i ch a m t m ng các con tr liên k t t i các nút con. Kích th c c a m ng con tr này úng b ng $||$.

Khi kích thích bằng cách cái l n, m ng con tr có th c thay th b ng danh sách móc n i hay cây nh phân tìm ki m t cân b ng ti t ki m b nh h n, tuy nhiên i u ó có th làm ph c t p tính toán c a thu t toán b ph thu c vào |. B ng d i ây tóm t t v nh h ng c a c u trúc nút lên thao tác r nhánh (t m t nút i sang nút con theo c nh mang nhãn có ký t $u \in \Sigma$)

| C u trúc | R nhánh | B nh |
|------------------------------------|-------------------|----------------|
| M ng con tr | $O(1)$ | $O(n \Sigma)$ |
| Cây nh phân tìm ki m t cân b ng | $O(\log \Sigma)$ | $O(n)$ |
| Danh sách móc n i | $O(\Sigma)$ | $O(n)$ |
| Danh sách ng c s p x p | $O(\log \Sigma)$ | $O(n)$ |

Khác với cây hút, mệnh đề cũng như các thuật toán xây dựng mệnh đề lại không gặp khó khăn gì khi biến đổi cái Σ1n. Tuy vậy, có rất nhiều thao tác quan trọng trên cây hút không thể dùng mệnh đề thay thế.

Năm 2004, nhóm nghiên cứu của Abouelhoda sau khi phân tích các ứng dụng đã có của cây hậu tố đã chỉ ra rằng: mức độ thu thập lý trên cây hậu tố trong các ứng dụng đã biết có thể quy về ba thao tác cơ bản [1]:

Duyệt cây từ dưới lên, tăng dần thông tin từ các nút con lên nút cha (bottom-up)

Duyệt cây từ trên xuống, giảm dần nút cha xuống nút con theo một nhãn cho trước (top-down)

Tìm nút liên quan đến nút khác theo liên kết hậu tố.

Tóm lại, các tác giả đã xuất hiện thêm những thuật toán xây dựng cấu trúc dữ liệu bền vững. Những cấu trúc dữ liệu này sẽ kết hợp với những mô phỏng ba thao tác cơ bản trên. Cấu trúc dữ liệu mới này có tên là **mảng hậu tố tăng cường** (*enhanced suffix arrays*), cho phép minh chứng có thể thay thế cho cây hậu tố trong tất cả các ứng dụng đã biết³. Kết quả này đã kéo theo nhiều nghiên cứu nhằm nâng cao tính hiệu quả của thuật toán xây dựng mảng hậu tố. Cho tới năm 2007, trong bài tổng quan của Puglisi trên tạp chí ACM Computing Survey [15], đã có tới 20 thuật toán xây dựng mảng hậu tố được đánh giá, chúng dựa trên những cách tiếp cận khác nhau, có những ưu/nhược điểm khác nhau tùy thuộc vào dữ liệu.

Tuy nhiên, vì các số liệu mảng hậu tố mô phỏng cây hậu tố đôi khi làm mất tính trực quan và gây khó khăn trong thiết kế thuật toán. Cũng trong nghiên cứu về mảng hậu tố tăng cường, các tác giả còn xuất hiện một số biến đổi của cây hậu tố thành một cấu trúc dữ liệu mới gọi là **cây lcp-interval** (*lcp-interval trees*) [9] với ý nghĩa các tính toán trên cây hậu tố cũng như mảng hậu tố. Cấu trúc dữ liệu này khá dễ cài đặt, tiết kiệm bộ nhớ và quan trọng nhất là vận hành nguyên tắc cấu trúc cây. Năm 2008, nhóm nghiên cứu của Kim [9] còn tìm ra nhiều tính chất quan trọng của cây lcp-interval so với cây hậu tố truy vấn thông thường, đưa ra phương pháp cài đặt trong trình biên dịch để cài đặt và chứng minh là đưa ra các thuật toán tính các liên kết hậu tố một cách ngắn gọn và trực tiếp, không cần thông qua truy vấn LCA hay RMQ.

Trong các kỳ thi lập trình có sẵn như thi thi đấu, ngoài tiêu chí về tính hiệu quả trong việc lựa chọn thuật toán, luôn phải quan tâm tới tính ngắn gọn. Những cấu trúc dữ liệu trong chuyên ngành này không dễ cài đặt, chính vì vậy thí sinh không nên lựa chọn chúng trong phòng thi. Ví dụ: thay vì mất ~100 dòng code để cài đặt mảng hậu tố (hoặc hơn với cây hậu tố) cho bài toán xác định hậu tố con, thí sinh có thể cài đặt thuật toán KMP với chỉ 20 dòng code như những người thi đấu cao hơn rất nhiều.

Nếu thí sinh đã bị t v cây/m ng h u t và g p m t bài toán có thể gì i quy t tri t b ng các c u trúc d li u này, thí sinh ít nh i u s có l i th v tâm lý b i vì c còn l i ch là v n th i gian. Tuy nhiên l i th này s nhanh chóng t o ra b t l i n u:

Ràng bu c d li u và yêu c u c a bài toán cho phép thí t k thu t toán n gì n và hi u qu h n nh i u. Nếu không phân tích k bài, thí sinh s m t c h i tìm ra thu t toán t t và b thi t v th i gian. Nói chung n u c u ho c không phân tích k bài thì càng bị t nh i u s càng b t l i.

Kh n ng cài t c a thí sinh không t t, quy trình ki m th c a thí sinh không c n th n, ho c thí sinh không th i gian ki m th . C u trúc d li u hi u qu nh ng cài t sai có th m t nh i u i m h n c nh ng ch ng trình cài t thu t toán t m th ng.

s d ng các thu t toán hay c u trúc d li u ph c t p m t cách hi u qu (nói riêng v i cây/m ng h u t), thí sinh ph i phân tích k ràng bu c và yêu c u bài toán và c g ng tìm thu t toán n gì n. Trong tr ng h p b t bu c ph i s d ng gì i pháp ph c t p, c n l ng tr c th i gian l p trình, g r i và ki m th . Ngoài ra, vì c cài t thu t toán t m th ng c ng là c n thi t i sánh k t qu và tránh m t i m quá nh i u n u không k p th i gian hoàn thi n.