CÁC BÀI TOÁN DUYỆT

1. Robot quét vôi (http://vn.spoj.pl/problems/NKROBOT)

Có 9 căn phòng (đánh số từ 1 đến 9) đã được quét vôi với màu trắng, xanh hoặc vàng. Có 9 robot (đánh số từ 1 đến 9) phụ trách việc quét vôi. Mỗi robot chỉ quét một số phòng nhất định. Việc quét vôi được thực hiện nhờ một chương trình cài sẵn theo qui tắc:

- Nếu phòng đang có màu trắng thì quét màu xanh
- Nếu phòng đang có màu xanh thì quét màu vàng
- Nếu phòng đang có màu vàng thì quét màu trắng

Cần phải gọi lần lượt một số các robot ra quét vôi (mỗi lần một robot, một robot có thể gọi nhiều lần và có thể có robot không được gọi. Robot được gọi sẽ quét vôi tất cả các phòng mà nó phụ trách) để cuối cùng các phòng đều có màu trắng.

Yêu cầu: Hãy tìm một phương án như vậy sao cho số lần gọi robot là ít nhất. Giả thiết rằng lượng vôi cho mỗi lượt quét đối với các phòng là như nhau.

Input

- 9 dòng đầu: dòng thứ i mô tả một danh sách các phòng do robot i phụ trách việc quét vôi. Mỗi dòng là một chuỗi các chữ số từ 1..9 biểu diễn các số hiệu của các phòng, các chữ số viết sát nhau.
- Dòng cuối mô tả mầu vôi ban đầu của các phòng. Dòng gồm 9 ký tự viết sát nhau gồm toàn các chữ cái T (trắng), X (xanh), V(vàng) biểu diễn mầu ban đầu của 9 căn phòng theo trật tự số hiệu của chúng.

Output

gồm một dòng

- Nếu không có phương án thì in ra số 0
- Trái lại thì in ra dãy thứ tự các robot được gọi (số hiệu các robot được viết sát nhau và in ra kết quả có thứ tự từ điển bé nhất)

Example

Input:
159
123
357
147
5
369
456
789
258
XVXVXVTXT

Output:
2255799

Hướng dẫn:

Chú ý rằng, sau 3 lần quét vôi thì màu của một căn phòng trở lại như cũ. Bởi vậy, việc sử dụng một robot ≥ 3 lần là không tối ưu. Ta có thể giải bài toán bằng cách sinh dãy tam phân độ dài 9, mỗi giá trị a[i] chính là số lần gọi robot i.

2. DÃY ABC

Cho trước một số nguyên dương N ($N \leq 100$), hãy tìm một xâu chỉ gồm các ký tự A, B, C thoả mãn 3 điều kiện:
- Có độ dài N
- Hai đoạn con bất kỳ liền nhau đều khác nhau (đoạn con là một dãy ký tự liên tiếp của xâu)
- Có ít ký tự C nhất.

Hướng dẫn:

Nếu dãy $X_1X_2...X_n$ thoả mãn 2 đoạn con bất kỳ liền nhau đều khác nhau, thì trong 4 ký tự liên tiếp bất kỳ bao giờ cũng phải có 1 ký tự "C". Như vậy với một dãy con gồm k ký tự liên tiếp của dãy X thì số ký tự C trong dãy con đó bắt buộc phải $\geq$ k div 4.

Tại bước thử chọn X_i, nếu ta đã có T_i ký tự "C" trong đoạn đã chọn từ X_1 đến X_i, thì cho dù các bước đệ quy tiếp sau làm tốt như thế nào chăng nữa, số ký tự "C" sẽ phải chọn thêm bao giờ cũng $\geq$ (n - i) div 4. Tức là nếu theo phương án chọn X_i như thế này thì số ký tự "C" trong dãy kết quả (khi chọn đến X_n) cho dù có làm tốt đến đâu cũng $\geq$ **T_i + (n - i) div 4**. Ta dùng con số này để đánh giá nhánh cận, nếu nó nhiều hơn số ký tự "C" trong "Cấu hình tối ưu" thì chắc chắn có làm tiếp cũng chỉ được một cấu hình tồi tệ hơn, ta bỏ qua ngay cách chọn này và thử phương án khác.

Các bạn tham khảo Code:

```
program ABC_STRING;
const
  InputFile  = 'ABC.INP';
  OutputFile = 'ABC.OUT';
  max = 100;
var
  N, MinC: Integer;
  X, Best: array[1..max] of 'A'..'C';
  T: array[0..max] of Integer;
  f: Text;

function Same(i, l: Integer): Boolean;
var
  j, k: Integer;
begin
  j := i - l;
  for k := 0 to l - 1 do
    if X[i - k] <> X[j - k] then
      begin
        Same := False; Exit;
      end;
  Same := True;
end;

function Check(i: Integer): Boolean;
var
  l: Integer;
begin
  for l := 1 to i div 2 do
    if Same(i, l) then
      begin
        Check := False; Exit;
      end;
  Check := True;
```

```
end;

procedure KeepResult;
begin
  MinC := T[N];
  Best := X;
end;

procedure Try(i: Integer);
var
  j: 'A'..'C';
begin
  for j := 'A' to 'C' do
    begin
      X[i] := j;
      if Check(i) then
        begin
          if j = 'C' then T[i] := T[i - 1] + 1
          else T[i] := T[i - 1];
          if T[i] + (N - i) div 4 < MinC then
            if i = N then KeepResult
            else Try(i + 1);
        end;
    end;
end;

procedure PrintResult;
var
  i: Integer;
begin
  for i := 1 to N do Write(f, Best[i]);
  WriteLn(f);
  WriteLn(f, '"C" Letter Count  : ', MinC);
end;

begin
  Assign(f, InputFile); Reset(f);
  ReadLn(f, N);
  Close(f);
  Assign(f, OutputFile); Rewrite(f);
  T[0] := 0;
  MinC := N;
  Try(1);
  PrintResult;
  Close(f);
end.
```

3. BÀI TOÁN NGƯỜI DU LỊCH

Cho n thành phố đánh số từ 1 đến n và m tuyến đường giao thông hai chiều giữa chúng, mạng lưới giao thông này được cho bởi bảng C cấp nxn, ở đây $C_{ij} = C_{ji}$ = Chi phí đi đoạn đường trực tiếp từ thành phố i đến thành phố j. Giả thiết rằng $C_{ii} = 0$ với $\forall i$, $C_{ij} = +\infty$ nếu không có đường trực tiếp từ thành phố i đến thành phố j.

Một người du lịch xuất phát từ thành phố 1, muốn đi thăm tất cả các thành phố còn lại mỗi thành phố đúng 1 lần và cuối cùng quay lại thành phố 1. Hãy chỉ ra cho người đó hành trình với chi phí ít nhất. Bài toán đó gọi là bài toán người du lịch hay bài toán hành trình của một thương gia (Traveling Salesman)

Hướng dẫn:

Hành trình cần tìm có dạng ($x_1 = 1, x_2, \ldots, x_n, x_{n+1} = 1$) ở đây giữa x_i và x_{i+1}: hai thành phố liên tiếp trong hành trình phải có đường đi trực tiếp ($C_{ij} \neq +\infty$) và ngoại trừ thành phố 1, không thành phố nào được lặp lại hai lần. Có nghĩa là dãy ($x_1, x_2, \ldots, x_n$) lập thành 1 hoán vị của (1, 2, …, n). Duyệt quay lui: x_2 có thể chọn một trong các thành phố mà x_1 có đường đi tới (trực tiếp), với mỗi cách thử chọn x_2 như vậy thì x_3 có thể chọn một trong các thành phố mà x_2 có đường đi tới (ngoài x_1). Tổng quát: x_i có thể chọn 1 trong các thành phố **chưa đi qua** mà **từ x_{i-1} có đường đi trực tiếp tới ($1 \leq i \leq n$).**

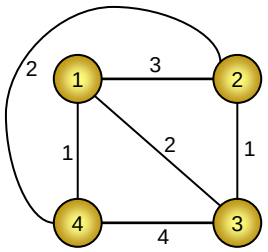
Nhánh cận: Khởi tạo cấu hình BestConfig có chi phí = $+\infty$. Với mỗi bước thử chọn x_i xem chi phí đường đi cho tới lúc đó có < Chi phí của cấu hình BestConfig?, nếu không nhỏ hơn thì thử giá trị khác ngay bởi có đi tiếp cũng chỉ tốn thêm. Khi thử được một giá trị x_n ta kiểm tra xem x_n có đường đi trực tiếp về 1 không ? Nếu có đánh giá chi phí đi từ thành phố 1 đến thành phố x_n cộng với chi phí từ x_n đi trực tiếp về 1, nếu nhỏ hơn chi phí của đường đi BestConfig thì cập nhật lại BestConfig bằng cách đi mới.

Sau thủ tục tìm kiếm quay lui mà chi phí của BestConfig vẫn bằng $+\infty$ thì có nghĩa là nó không tìm thấy một hành trình nào thoả mãn điều kiện đề bài để cập nhật BestConfig, bài toán không có lời giải, còn nếu chi phí của BestConfig < $+\infty$ thì in ra cấu hình BestConfig - đó là hành trình ít tốn kém nhất tìm được

Input: file văn bản TOURISM.INP

- Dòng 1: Chứa số thành phố n ($1 \leq n \leq 20$) và số tuyến đường m trong mạng lưới giao thông
- m dòng tiếp theo, mỗi dòng ghi số hiệu hai thành phố có đường đi trực tiếp và chi phí đi trên quãng đường đó (chi phí này là số nguyên dương ≤ 100)

Output: file văn bản TOURISM.OUT, ghi hành trình tìm được.



TOURISM.INP	TOURISM.OUT
4 6	1->3->2->4->1
1 2 3	Cost: 6
1 3 2	
1 4 1	
2 3 1	
2 4 2	
3 4 4	

Các bạn tham khảo Code:

```
program TravellingSalesman;
const
  InputFile  = 'TOURISM.INP';
  OutputFile = 'TOURISM.OUT';
  max = 20;
  maxC = maxlongint div 2;
var
  C: array[1..max, 1..max] of longint;
  X, BestWay: array[1..max + 1] of longint;
  T: array[1..max + 1] of longint;
  Free: array[1..max] of Boolean;
  m, n: longint;
  MinSpending: longint;

procedure Enter;
var
  i, j, k: longint;
  f: Text;
begin
  Assign(f, InputFile); Reset(f);
  ReadLn(f, n, m);
  for i := 1 to n do
    for j := 1 to n do
      if i = j then C[i, j] := 0 else C[i, j] := maxC;
  for k := 1 to m do
    begin
      ReadLn(f, i, j, C[i, j]);
      C[j, i] := C[i, j];
    end;
  Close(f);
end;

procedure Init;
begin
  FillChar(Free, n, True);
  Free[1] := False;
  X[1] := 1;
  T[1] := 0;
  MinSpending := maxC;
end;
```

```
procedure Try(i: longint);
var
  j: longint;
begin
  for j := 2 to n do
    if Free[j] then
      begin
        X[i] := j;
        T[i] := T[i - 1] + C[x[i - 1], j];
        if T[i] < MinSpending then
          if i < n then
            begin
              Free[j] := False;
              Try(i + 1);
              Free[j] := True;
            end
          else
            if T[n] + C[x[n], 1] < MinSpending then
              begin
                BestWay := X;
                MinSpending := T[n] + C[x[n], 1];
              end;
      end;
end;
procedure PrintResult;
var
  i: longint;
  f: Text;
begin
  Assign(f, OutputFile); Rewrite(f);
  if MinSpending = maxC then WriteLn(f, 'NO SOLUTION')
  else
    for i := 1 to n do Write(f, BestWay[i], '->');
  WriteLn(f, 1);
  WriteLn(f, 'Cost: ', MinSpending);
  Close(f);
end;
begin
  Enter;
  Init;
  Try(2);
  PrintResult;
end.
```

4. Tour du lịch của Sherry (http://vn.spoj.pl/problems/LEM3)

Trong kì nghỉ hè năm nay sherry được bố thưởng cho 1 tour du lịch quanh N đất nước tươi đẹp với nhiều thắng cảnh nổi tiếng (vì sherry rất ngoan). Tất nhiên sherry sẽ đi bằng máy bay.
Giá vé máy bay từ đất nước i đến đất nước j là C_{ij} (dĩ nhiên C_{ij} có thể khác C_{ji}). Tuy được bố thưởng cho nhiều tiền để đi du lịch nhưng sherry cũng muốn tìm cho mình 1 hành trình với chi phí rẻ nhất có thể để dành tiền mua quà về tặng mọi người (Các chuyến bay của sherry đều được đảm bảo an toàn tuyệt đối).
Bạn hãy giúp sherry tìm 1 hành trình đi qua tất cả các nước, mỗi nước đúng 1 lần sao cho chi phí là bé nhất nhé.

Input

Dòng 1: N (5 < N < 16)
Dòng thứ i trong N dòng tiếp theo: Gồm N số nguyên, số thứ j là C_{ij} (0 < C_{ij} < 10001)

Output

Gồm 1 dòng duy nhất ghi chi phí bé nhất tìm được

Example

Input:	Output:
6 0 1 2 1 3 4 5 0 3 2 3 4 4 1 0 2 1 2 4 2 5 0 4 3 2 5 3 5 0 2 5 4 3 3 1 0	8

5. Chấm điểm (http://vn.spoj.pl/problems/V8SCORE)

Có N vị giám khảo trong kỳ thi chọn đội tuyển tin học. Kỳ thi bao gồm K bài. Vị giám khảo thứ i đề nghị số điểm của bài j là A_{ij}.
Hội đồng giám khảo muốn xác định số điểm cho mỗi bài sao cho:

- Tổng số điểm bằng S.
- Điểm của mỗi bài không bé hơn điểm của bài trước đó.
- Số điểm của mỗi bài bằng điểm đề nghị cho bài này của một vị giám khảo nào đó.

Dữ liệu

- Dòng đầu tiên chứa ba số nguyên S ($1 \le S \le 200$), ($1 \le K \le 20$), ($1 \le N \le 20$).
- Dòng thứ i trong số N dòng tiếp theo chứa K số nguyên, số thứ j cho biết giá trị A_{ij} là số điểm vị giám khảo thứ i đề nghị cho bài thứ j.

Kết qủa

- Nếu tồn tại một cách cho điểm thỏa mãn yêu cầu:
  - Dòng thứ nhất: in ra 'YES'.
  - Dòng thứ hai: in ra K số nguyên là điểm của mỗi bài tìm được.
- Nếu không tồn tại cách cho điểm, in ra 'NO'.

- **Ví dụ**

Dữ liệu	Kết quả
100 3 2 30 20 40 50 30 50	YES 30 30 40
100 2 3 1 1 2 2 3 3	NO

Hướng dẫn:

Kết quả của bài toán (nếu có) sẽ có dạng $X_1X_2...X_k$. Trong mỗi bước duyệt, ta sẽ thử chọn X_i trong tập A_{ji} (1 <= j <= n). Sau khi chọn X_i, ta có thể đánh giá được S_{min} (tổng điểm bé nhất có thể đạt được khi duyệt xong) bằng dữ kiện dãy X tăng dần.

S_{min} (tổng điểm khi chọn xong) = P[i] + X[i] * (k-i) với P[i] là tổng điểm đã có

- S_{min} <= S : duyệt tiếp bài toán vẫn có thể có nghiệm
- S_{min} > S : vô nghiệm

Đó cũng chính là cận trong quá trình duyệt để chương trình có thể chạy trong thời gian cho phép!

6. Hoán vị chữ cái (http://vn.spoj.pl/problems/QBHV)

Cho một xâu S chỉ gồm các chữ cái in hoa, 1 <= độ dài <= 9.

Yêu cầu:

1: Có bao nhiêu cách hoán vị các chữ cái của xâu S

2: Liệt kê các hoán vị đó theo thứ tự từ điển

Input

Gồm 1 dòng duy nhất chứa xâu S

Output

Dòng 1: Ghi số lượng hoán vị tìm được (K)

K dòng tiếp theo, mỗi dòng ghi một xâu hoán vị của xâu S theo đúng thứ tự từ điển

Example

```
Input:
ABAB
Output:
6
AABB
ABAB
ABBA
BAAB
BABA
BBAA
```

Hướng dẫn:

Hai yêu cầu cần giải quyết của bài toán này chính là số lượng cấu hình thỏa mãn và liệt kê các cấu hình đó. Nếu như duyệt 2 lần, lần thứ nhất để đếm số lượng và lần thứ 2 để liệt kê các cấu hình thì chương trình sẽ chạy quá thời gian cho phép. Bởi thế, ta nên tìm cách đếm số cấu hình mà không cần phải duyệt quay lui.

Công thức: **Result = N! div (Count[ch])**

với N là độ dài của S và Count[ch] là số lần xuất hiện của ký tự "ch".

7. Quan hệ (http://vn.spoj.pl/problems/COND)

Xét một tập N đối tượng có thể so sánh được (2<=n<=10). Giữa 2 đối tượng a và b có thể tồn tại 1 trong 3 quan hệ phân loại:

a = b; a < b; a > b;

Như vậy, với 3 đối tượng (a, b, c) có thể tồn tại 13 quan hệ phân loại như sau:

a = b = c; a = b < c; c < a = b; a < b = c

b = c < a; a = c < b; b < a = c; a < b < c

a < c < b; b < a < c; b < c < a; c < a < b

c < b < a;

Cho số n, hãy xác định số lượng quan hệ phân loại khác nhau.

Input

Gồm nhiều số n. Mỗi số trên 1 dòng. Kết thúc file là -1.

Output

Với mỗi n, đưa ra số lượng quan hệ phân loại tìm được, mỗi số trên 1 dòng (không có dòng trống).

Example

Input:

2
3
-1

Output:

3
13

Hướng dẫn:

Bài này là một bài toán đếm tổ hợp. Bạn nhận xét một quan hệ phân loại sẽ có dạng
(= =) < (= =) < ... < (= =)
tức là chia thành các nhóm bằng nhau
Giả sử có k nhóm và gọi số phần tử trong mỗi nhóm là a1, a2, ..., ak.
Điều kiện: a1 + a2 + ... + ak <= n

Thế thì số quan hệ phân loại loại này bằng: n! / (a1! * a2! * a3! * ... * ak!)

Tóm lại công thức cuối cùng bằng:

Sum (n! / (a1! * a2! * ... * ak!) | a1 + a2 + ... + ak <= n)

8. Quan hệ có điều kiện (http://vn.spoj.pl/problems/QBCOND)

Ngày nay khi nghiên cứu quan hệ giữa các phần tử các nhà khoa học không đơn giản chỉ nghiên cứu các quan hệ bình thường mà để thêm phần phức tạp là thêm vào đó 1 vài bộ điều kiện. Một trong những điều kiện đó là số quan hệ '='

Như ta đã biết giữa 2 phần tử a, b sẽ có 3 quan hệ:

a = b, a > b, a < b.

Các nhà khoa học đưa ra 1 bộ gồm n phần tử. Sau khi tìm ra số lượng các quan hệ của n phần tử này họ muốn biết nếu như số quan hệ '=' trong tập n phần tử này đúng bằng k thì sẽ có bao nhiêu quan hệ như thế?

Input

Gồm nhiều bộ số n, k. Mỗi bộ số trên 1 dòng. Kết thúc file là -1. (1 < n < 11)

Output

Với mỗi bộ số (n, k) đưa ra số quan hệ có điều kiện tìm được

Example

Input:

3 0
3 1
3 2
3 3
-1

Output:

6
6
0
1

Giải thích:

Với bộ 3 phần tử (a, b, c).

n=3, k=0:

a < b < c; a < c < b; b < a < c;
b < c < a; c < a < b; c < b < a;

n=3, k=1:

a = b < c; c < a = b; a < b = c
b = c < a; a = c < b; b < a = c;

n=3, k=3:

a = b = c;

CÁC THAO TÁC XỬ LÝ BIT

Dưới đây là những kiến thức về việc sử dụng các phép toán logic từ đó giúp cho việc thiết kế các biểu thức logic dùng rất nhiều trong các phép toán điều kiện được nhanh chóng, chính xác, hiệu quả.

*** Quy ước về vị trí của các bit:**

Mỗi byte bao gồm 8 bit được mã số từ phải sang trái còn gọi là bit thấp đến bit cao. Bit nằm ở bên phải được xem là thấp hơn bit nằm ở bên trái. Các bit được đánh số như sau: 7 6 5 4 3 2 1 0

Mỗi bit có thể nhận 1 trong 2 giá trị là 0 hoặc 1. Tại mỗi thời điểm thực hiện chương trình mỗi bit được nhận giá trị xác định. Mọi số nguyên trong máy đều biểu diễn dưới dạng nhị phân, thí dụ số 19 được biểu diễn như sau:

Bit 7 6 5 4 3 2 1 0

Giá trị 0 0 0 1 0 0 1 1 (số 19)

*** Các phép toán logic**

Các phép toán sau đây thực hiện trên các giá trị nguyên và cho kết quả là các giá trị nguyên.

1. Phép đảo bit NOT: đổi giá trị của mọi bit từ 0 thành 1 và ngược lại.

2. Phép cộng logic trên các bit OR thực hiện trên từng cặp bit tương ứng của các toán hạng theo bảng cộng sau:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Quy tắc: Tổng hai bit bằng 0 khi và chỉ khi cả hai bit bằng 0 ngoài ra tổng nhận giá trị 1. Phép OR còn được gọi là phép "hoặc".

3. Phép nhân logic trên các bit AND: thực hiện trên từng cặp bit tương ứng của các toán hạng theo bảng nhân sau:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Quy tắc: Tích hai bit bằng 1 khi và chỉ khi cả hai bit bằng 1, ngoài ra tích nhận giá trị 0. Phép AND còn được gọi là phép "và".

4. Phép cộng loại trừ trên các bit (XOR) : thực hiện trên từng cặp bit tương ứng của các toán hạng theo bảng sau

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Quy tắc: Tổng loại trừ của hai bit bằng 1 khi và chỉ khi hai bit đó chứa các giá trị khác nhau, ngoài ra tổng loại trừ nhận giá trị 0.
Phép toán trên cũng còn được gọi là phép so sánh khác (khác nhau là đúng, bằng nhau là sai) hay còn gọi là phép "triệt tiêu".

x SHR i : Phép dịch phải, cho giá trị có được từ số nguyên x sau khi dịch sang phải i bit.
x SHL i : Phép dịch trái, cho giá trị có được từ số nguyên x sau khi dịch sang trái i bit.

Với x = 2 ta có:
Trên đây là một số phép toán làm việc trên các bit mà ta hay dùng, trên cơ sở đó, ta xây dựng được một số hàm, thủ tục hay dùng sau.

1. Hàm lấy giá trị bit: Hàm trả về giá trị 0 hoặc 1

```
Function GetBit(x, i:longint):longint;
Begin
   GetBit:=(x SHR i) and 1;
End;
```

2. Thủ tục bật bit: Thủ tục gán trị 1 cho bit thứ i trong số nguyên x.

```
Procedure OnBit(Var x:longint; i:longint);
Begin
  x:=x OR (1 SHL i);
End;
```

3. Thủ tục tắt bit: Thủ tục gán trị 1 cho bit thứ i trong số nguyên x.

```
Procedure OffBit(Var x:longint; i:longint);
Begin
  x:=x AND (NOT(1 SHL i));
End;
```

Chúng ta xét qua các bài toán sau để tìm hiểu về ứng dụng của xử lý bit.

1. Số đặc biệt:

Xét một dãy gồm N số nguyên A_1, A_2, A_3, ... A_n. Trong dãy số trên có 1 số chỉ xuất hiện đúng một lần, và các số còn lại xuất hiện một số chẵn lần.

Yêu cầu: Hãy tìm số đặc biệt của một dãy cho trước.

Dữ liệu: Trong file văn bản SDB gồm:

- Dòng đầu là số N ($N \le 10^7$)
- N dòng tiếp, dòng thứ i là A_i với $|A_i| \le 10^9$.

Kết quả: Một dòng duy nhất là số cần tìm.

Ví dụ:

SDB.INP	SDB.OUT
5 2 1 2 3 3	1

Hướng dẫn: Lần lượt thực hiện phép XOR số thứ 1 với số thứ 2, lấy kết quả thực hiện với số thứ 3 và cứ thế cho hết N số. Vì phép XOR là phép "triệt tiêu", do đó kết quả cuối cùng là số chỉ xuất hiện 1 lần (không bị triệt tiêu). Bằng cách biểu diễn các số dưới dạng nhị phân rồi thực hiện phép XOR, các bạn có thể dễ dàng chứng minh được thuật toán trên là đúng đắn.

```
res := 0;
For i := 1 to N do
    begin
       read(fi, a);
       res := res xor a;
    end;
writeln(fo, res);
```

2. Xâu cô lập:

Cho trước N (N ≤ 10000) xâu ký tự độ dài không quá 255 ký tự. Xâu cô lập được định nghĩa là xâu chỉ xuất hiện duy nhất 1 lần trong N xâu đã cho, các xâu còn lại luôn xuất hiện một số chẵn lần.

Yêu cầu: Tìm xâu cô lập từ N xâu đã cho.

Dữ liệu: Trong file văn bản SINGLE.INP gồm N dòng biểu diễn N xâu đã cho ban đầu.

Kết quả: Một dòng duy nhất là xâu cô lập.

Ví dụ:

SINGLE.INP	SINGLE.OUT
- Hello, sir . - Go away! - Can I help you? - Hello, sir. - Can I help you?	- Go away!

Hướng dẫn: Cách làm tương tự với bài "Số đặc biệt", ta thực hiện phép XOR với các mã ASCII.

3. Liệt kê tập con:

Cho tập hợp gồm N phần tử ($1 \leq N \leq 20$) Hãy liệt kê tất cả các tập con (kể cả rỗng) của tập hợp đã cho.

Hướng dẫn: Xem như N phần tử là dãy N bit. Ta có thể biểu diễn tất cả các tập con bằng dãy N bit, giá trị 1 (hoặc 0) biểu diễn sự tồn tại (hoặc không tồn tại) của mỗi phần tử. Giá trị dãy bit tương ứng từ 0 .. 2^n - 1. Để kiểm tra sự tồn tại của 1 phần tử trong dãy bit có giá trị x, ta sử dụng hàm GetBit như đã nêu trên. Đây là 1 bài toán rất cơ bản, các bạn có thể tự code.

DUYỆT BẰNG CÁCH CHIA ĐÔI TẬP HỢP

1. Tổng vector (http://vn.spoj.pl/problems/VECTOR)

Trong mặt phẳng tọa độ có N véc tơ. Mỗi một véc tơ được cho bởi hai chỉ số x và y. Tổng của hai véc tơ (x_i, y_i) và (x_j, y_j) được định nghĩa là một véc tơ $(x_i + x_j, y_i + y_j)$. Bài toán đặt ra là cần chọn một số véc tơ trong N véc tơ đã cho sao cho tổng của các vec tơ đó là véc tơ (U, V).

Yêu cầu: Đếm số cách chọn thoả mãn yêu cầu bài toán đặt ra ở trên.

Input

Dòng thứ nhất ghi số N ($0 \le N \le 30$).

N dòng tiếp theo, dòng thứ i ghi các số nguyên x_i, y_i lần lượt là hai chỉ số của véc tơ thứ i. ($|x_i|, |y_i| \le 100$).

Dòng cuối cùng ghi số hai số nguyên U V ($|U|, |V| \le 10^9$).

Output

Gồm một số duy nhất là số cách chọn thoả mãn.

Example

Input:
```
4
0  0
-1  2
2  5
3  3
2  5
```

Output:
```
4
```

Hướng dẫn: Nếu duyệt tổ hợp của N vector thì độ phức tạp của thuật toán là 2^N và chương trình sẽ không cho kết quả trong thời gian cho phép với cỡ $N \le 32$. Cách giải quyết như sau:

- Chia tập N vector thành 2 tập bằng nhau A và B.
- Gọi F[x, y] là số cách chọn để có vector tổng (x, y) trên tập A. Để tính F[x, y] thì ta sẽ duyệt tất cả các tập con của tập A, sau khi tính được vector tổng của mỗi tập con ta chỉ việc **inc**(F[x, y]).
- Tương tự ta sẽ duyệt trên tập B, sau khi tính được vector tổng (x1, y1) của mỗi tập con ta sẽ **inc**(Res, F[U - x1, V - y1]) với Res là kết quả của bài toán.

Việc duyệt tập con của tập A, B có thể cài đặt như bài toán ở phần trước đã nhắc tới!

2. 34 đồng xu (http://vn.spoj.pl/problems/COIN34)

Bạn có 34 đồng xu có giá trị như sau:

xu [1] có giá trị 2

xu [2] có giá trị 3

xu [3] có giá trị 5

for n := 4 to 34 do
    xu [n] có giá trị (xu[n-1] + xu[n-2] + xu[n-3])
Bạn hãy dùng nhiều đồng xu nhất để mua một món hàng có giá là X

Dữ liệu

Dòng đầu tiên là số test (không quá 1000). Mỗi dòng tiếp theo chứa một số nguyên X (1 ≤ X ≤ 2000000000).

Kết quả

Với mỗi test, in ra "Case #" + số hiệu test + ": " + số lượng lớn nhất đồng xu cần dùng. Nếu không có cách nào để đạt giá trị X thì in ra -1.

Ví dụ

Dữ liệu
4
1
5
8
9

Kết quả
Case #1: -1
Case #2: 2
Case #3: 2
Case #4: -1

Hướng dẫn: Gọi T[i] là số đồng xu nhiều nhất có thể mua món hang có giá trị là i. Trong mỗi bước duyệt ở tập thứ 2, ta tiến hành cập nhật T[i]: **T[i] := Max(T[i], F[i - x] + y)** với x là khối lượng tập hợp con, y là số lượng xu của tập. Mảng F có ý nghĩa tương tự T và đã được tính trước ở bước 1.

3. Nhà hàng Trung Quốc (http://vn.spoj.pl/problems/CHNREST)

Hàng năm vì muốn có không khí ấm cúng và cũng để tiết kiệm nên bạn thường tổ chức sinh nhật ở nhà. Tuy nhiên trước sinh nhật năm nay vài hôm bạn đã thi đậu vào đội tuyển tin học quốc gia. Đây là một sự kiện đặc biệt có ý nghĩa nên bạn quyết định mừng ngày sinh nhật của mình tại một nhà hàng Trung Quốc sang trọng và bạn tự nhủ lần này nhất định phải tiêu xài rộng tay hơn. Mọi việc chuẩn bị đã gần xong nhưng còn một vấn đề làm bạn khá nhức đầu, đó là làm sao chọn được những món ăn mà mọi người cùng thích.

Nhà hàng có M món ăn khác nhau và thú vị ở chỗ là mỗi món ăn rất nhiều nên có thể đủ cho bao nhiêu người cũng được, vì thế vấn đề là gọi món nào chứ không phải mỗi món gọi bao nhiêu. Có tất cả N người đến dự tiệc sinh nhật (bao gồm cả bạn trong đó). Bạn đã tìm hiểu được danh sách những món ăn yêu thích của từng người và bạn muốn rằng đối với mỗi người phải có ít nhất 2 món mà họ thích. Tuy nhiên sau khi ăn xong còn nhiều tiết mục hấp dẫn khác nên bạn cũng muốn rằng bất kỳ ai cũng không có quá 2 món ăn yêu thích trong danh sách được đặt trước. Và vấn đề cuối cùng, đây là tiền của bố mẹ nên cũng không nên tiêu xài quá đáng.

Yêu cầu

Hãy cho biết số tiền ít nhất phải trả để gọi một thực đơn thỏa mãn các yêu cầu trên.

Dữ liệu

- Dòng đầu tiên chứa hai số M, N
- Dòng thứ hai chứa M số P_i là giá của món thứ i.
- Trong N dòng cuối cùng, dòng thứ k ghi danh sách các món yêu thích của người thứ k.

Kết quả

- Gồm một số duy nhất là kết quả của bài toán, hoặc
- in ra -1 nếu không có cách gọi món nào thỏa mãn.

Ví dụ

Dữ liệu:
5 3
100 150 300 425 200
1 2 4
1 3 4 5
1 4 5

Kết quả:
450

Giới hạn

- M ≤ 30.
- N ≤ 10.

Hướng dẫn: Theo đề ra thì mỗi người thích đúng 2 món trong thực đơn được chọn. Duyệt với m div 2 món, với mỗi tổ hợp ta có dãy A_1, A_2, A_3, ..., A_n là số các món yêu thích của N người. Tương tự với lần thứ 2 là B_1, B_2, B_3, ..., B_n. (Với A_i, B_i <= 2 và $A_i + B_i = 2$). Chúng ta tìm cách mã hóa dãy A, B thành số tự nhiên để tiện cho việc lưu trữ và tính toán.

4. Phân tập (http://vn.spoj.pl/problems/LQDDIV)

Cho N người($2 \le N \le 32$) ,mỗi người có một số a_i($1 \le a_i \le 10^9$) được gọi là độ tin cậy

Cần phân chia n người này vào 2 tập sao cho:

- Mỗi người thuộc đúng một tập
- Chênh lệch tổng độ tin cậy của 2 phần là bé nhất

Input

Dòng đầu chứa số nguyên N
Dòng tiếp theo chứa N số : số thứ i là độ tin cậy của người thứ i

Output

Ghi ra hai số u và v với u là độ chênh lệch nhỏ nhất và v là số cách phân chia

Example

Input:
5
1 5 6 7 8
Output:
1 3

Chú thích : Độ chênh lệch ít nhất của 2 phần là 1
Có 3 cách phân chia .3 cách phân chia nhóm 1 là (3,5) ,(1,3,4) và (1,2,5)

Hướng dẫn: Tư tưởng của bài toán vẫn là chia đôi để duyệt. Ta sẽ chọn một số người ở lần duyệt thứ nhất và một số người ở lần duyệt thứ hai để cho vào "nhóm 1".
Giả sử tổng độ tin cậy lần duyệt thứ nhất là x, thứ hai là y, gọi S là tổng độ tin cậy của N người. Tổng độ tin cậy của "nhóm 1" trong TH này là x + y, và của "nhóm 2" là S - (x + y). Độ chênh lệch tạo thành là Abs(S - (x + y) - (x + y)) = Abs(S - 2*(x+y)). Cách giải quyết cụ thể như sau :

- Duyệt N div 2 người, các tổng độ tin cậy thu được lưu vào một mảng C có tối đa là 2^{16} phần tử. Để tiện cho tính toán sau này, ta sẽ tối ưu mảng bằng cách loại bỏ những tổng bằng nhau và chỉ giữ lại một, đồng thời dùng thêm 1 mảng để đếm số lần xuất hiện của tổng đó (cần sắp xếp lại mảng C trước khi tối ưu nó). Ví dụ D[i] là số lần xuất hiện của tổng độ tin cậy C[i].

- Duyệt phần còn lại, mỗi tổng độ tin cậy X sinh ra, ta sẽ tìm C[i] sao cho Abs(S - 2*(X+C[i])) nhỏ nhất có thể. Sau đó cập nhật kết quả tối ưu. Tìm kiếm nhị phân giá trị C[i] sẽ là rất hợp lí bởi mảng C đã được sắp xếp và kích thước của nó cũng khá lớn.

TÌM KIẾM NHỊ PHÂN

Các bạn có thể tham khảo kỹ thuật tìm kiếm nhị phân ở các tài liệu khác. Trong tài liệu này, chúng ta sẽ lướt qua một vài ví dụ thật cơ bản để các bạn có thể hiểu thêm về nó.

1. Tải trọng của tuyến đường

Một hệ thống giao thông liên thông gồm N thành phố với tên 1..N (N <= 100). Có một số đoạn đường hai chiều giữa một số cặp thành phố và mỗi đoạn đường có một tải trọng tối đa mà chỉ có các xe với tải trọng không lớn hơn mới đi qua được.

Cần đi từ thành phố U tới V. Hãy tìm một hành trình sao cho tải trọng tối đa cho phép trên hành trình đó là lớn nhất có thể được.

Dữ liệu : Trong file văn bản TAITRONG.INP gồm

- Dòng đầu là 3 số N, U, V
- Tiếp theo là một số dòng, mỗi dòng ghi ba số nguyên dương X Y Z với ý nghĩa có đường đi giữa X và Y với tải trọng tối đa cho phép là Z (0 < Z <= 10000).

Kết quả : Ra file văn bản TAITRONG.OUT gồm

- Dòng thứ nhất ghi tải trọng H tối đa của xe có thể.
- Trong các dòng tiếp, mỗi dòng ghi tên một thành phố trong hành trình từ U kết thúc tại V.

Ví dụ :

TAITRONG.INP	TAITRONG.OUT
4 1 4 1 2 10 2 4 1 1 3 5 3 4 3	3 1 3 4

Hướng dẫn:

Đặt H_{min} = Min(a[i, j]), H_{max} = Max(a[i, j]). Với mỗi giá trị h ($H_{min} \le h \le H_{max}$) ta xây dựng đồ thị G(h) thỏa mãn: - N đỉnh tương ứng với N thành phố.

- 2 đỉnh i, j có cạnh nối nếu a[i, j] ≥ h.

Như vậy, nếu ta tìm thấy được một đường đi từ U tới V thì ta nói rằng : "Mạng giao thông có tải trọng tối thiểu h". Bài toán trở thành "Tìm giá trị h lớn nhất để tồn tại đường đi từ U tới V".

Ta sẽ sử dụng kỹ thuật tìm kiếm nhị phân dựa theo nhận xét: Nếu mạng có tải trọng tối thiểu k, và h là giá trị lớn nhất để "mạng giao thông có tải trọng tối thiểu h" thì $k \le h \le H_{max}$. Ngược lại, nếu không có lộ trình với tải trọng tối thiểu là k thì $H_{min} \le h < k$.

Với mỗi giá trị h, có thể duyệt DFS hoặc BFS để kiểm tra tồn tại đường đi từ U tới V hay không.

2. Bộ sưu tập các đồng xu (http://vn.spoj.pl/problems/LEM1/)

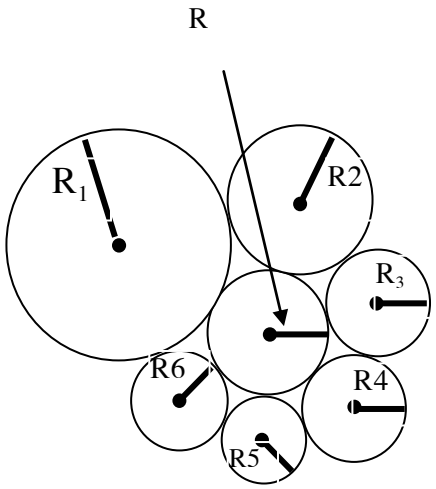
Cho N đồng xu có bán kính lần lượt là các số thực dương r_1.. r_N. Được đặt xung quanh một vòng tròn sao cho:

Mỗi đồng xu tiếp xúc với 2 đồng xu

đặt cạnh nó và tiếp xúc với vòng tròn.

Biết được bán kính của từng đồng xu.

Yêu cầu: Tìm bán kính vòng tròn.



Input

Dòng đầu ghi số nguyên dương N

Dòng tiếp theo ghi N số r_i ($1 \le i \le N$)

Output

Gồm 1 dòng duy nhất ghi bán kính hình tròn (độ chính xác đến 3 chữ số sau dấu phẩy)

Example

Input:

4

2 2 2 2

Output:

0.828

Giới hạn

- $1 \le N \le 10000$
- $1 \le r_i \le 100000$

Hướng dẫn:

Gọi O là tâm của đường tròn bán kính R, O_1, O_2, O_3, ... O_n là tâm của các đường tròn bán kính tương ứng R_1, R_2, R_3, ... R_n.

Khi R thỏa mãn yêu cầu, ta có: $\angle O_1OO_2 + \angle O_2OO_3 + \angle O_3OO_4 + ... + \angle O_{n-1}OO_n = 2pi$ (*)

Dễ dàng tính được các góc này theo độ dài 3 cạnh của các tam giác tương ứng. Để tìm R thỏa mãn, ta sẽ chia nhị phân giá trị của R với $R_{min} = 0$ và R_{max} = tổng các R_i. Giá trị R thỏa mãn (*) chính là nghiệm của bài toán.

Chú ý xử lý dữ liệu tránh việc sai số khá lớn khi làm việc với số thực.

QUY HOẠCH ĐỘNG

Quy hoạch động là dạng bài toán khá phổ biến trong các kỳ thi HSG môn Tin học. Mục đích của chúng là giải quyết các bài toán tối ưu. Vì không có một thuật toán tổng quát để giải tất cả các bài toán quy hoạch động, do đó các ví dụ sau đây giúp các bạn làm quen và tiếp cận một số dạng toán quy hoạch động.

1. Đi xem phim (http://vn.spoj.pl/problems/VCOWFLIX)

Nông dân John đang đưa các con bò của anh ta đi xem phim! Xe tải của anh ta thì có sức chứa có hạn thôi, là C (100 <= C <= 5000) kg, anh ta muốn đưa 1 số con bò đi xem phim sao cho tổng khối lượng của đống bò này là lớn nhất, đồng thời xe tải của anh ta vẫn chịu được.
Cho N (1 <= N <= 16) con bò và khối lượng W_i của từng con, hãy cho biết khối lượng bò lớn nhất mà John có thể đưa đi xem phim là bao nhiêu.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: C và N
- Dòng 2..N+1: Dòng i+1 chứa 1 số nguyên: W_i

Kết quả

- Dòng 1: Một số nguyên là tổng khối lượng bò lớn nhất mà John có thể mang đi xem phim.

Ví dụ

Dữ liệu
259 5
81
58
42
33
61

Kết quả
242

Giải thích

81+58+42+61 = 242; đây là tổng khối lượng bò lớn nhất có thể được.

Hướng dẫn: Sử dụng mảng F : boolean với ý nghĩa F[i] = true nếu có cách chọn các con bò để có khối lượng là i và ngược lại.

```
F[0] := true;
For i := 1 to n do
   For j := C downto w[i] do
      F[j] := F[j] or (F[j - w[i]]);

For i := C downto 0 do
   if F[i] then // cách chọn tối ưu nhất!
        begin
          writeln(fo, i);
          exit;
        end;
```

Hãy tự kiểm tra lại xem vòng for thứ 2 là "for downto" chứ không phải là "for to".

2. Bậc thang (http://vn.spoj.pl/problems/VSTEPS)

Bờm chơi trò chơi điện tử Lucky Luke đến màn phải điều khiển Lucky leo lên một cầu thang gồm n bậc.

Các bậc thang được đánh số từ 1 đến n từ dưới lên trên. Lucky có thể đi lên một bậc thang, hoặc nhảy một bước lên hai bậc thang. Tuy nhiên một số bậc thang đã bị thủng do cũ kỹ và Lucky không thể bước chân lên được. Biết ban đầu, Lucky đứng ở bậc thang số 1 (bậc thang số 1 không bao giờ bị thủng).

Chơi đến đây, Bờm chợt nảy ra câu hỏi: có bao nhiêu cách để Lucky leo hết được cầu thang? (nghĩa là leo đến bậc thang thứ n). Bờm muốn nhờ bạn trả lời câu hỏi này.

Dữ liệu

- Dòng đầu tiên: gồm 2 số nguyên n và k, là số bậc của cầu thang và số bậc thang bị hỏng ($0 \le k < n \le 100000$).
- Dòng thứ hai: gồm k số nguyên cho biết chỉ số của các bậc thang bị hỏng theo thứ tự tăng dần.

Kết quả

In ra phần dư của số cách Lucky leo hết cầu thang khi chia cho 14062008.

Ví dụ

Dữ liệu
4 2
2 3

Kết qủa
0

Dữ liệu
90000 1
49000

Kết qủa
4108266

Hướng dẫn: Gọi F[i] là số cách để tới được bậc thang thứ i. Ta dễ dàng tìm ra công thức QHĐ là F[i] := F[i-1] + F[i-2]. Độ phức tạp O(n).

3. Xếp hàng mua vé (http://vn.spoj.pl/problems/NKTICK)

Có N người sắp hàng mua vé dự buổi hoà nhạc. Ta đánh số họ từ 1 đến N theo thứ tự đứng trong hàng. Mỗi người cần mua một vé, song người bán vé được phép bán cho mỗi người tối đa hai vé. Vì thế, một số người có thể rời hàng và nhờ người đứng trước mình mua hộ vé. Biết t_i là thời gian cần thiết để người i mua xong vé cho mình. Nếu người i+1 rời khỏi hàng và nhờ người i mua hộ vé thì thời gian để người thứ i mua được vé cho cả hai người là r_i.

Yêu cầu: Xác định xem những người nào cần rời khỏi hàng và nhờ người đứng trước mua hộ vé để tổng thời gian phục vụ bán vé là nhỏ nhất.

Dữ liệu

- Dòng đầu tiên chứa số N ($1 \le N \le 60000$).
- Dòng thứ 2 ghi N số nguyên dương $t_1, t_2, ..., t_N$. ($1 \le t_i \le 30000$)
- Dòng thứ ba ghi N-1 số nguyên dương $r_1, r_2, ..., r_{N-1}$. ($1 \le r_i \le 30000$)

Kết qủa

In ra tổng thời gian phục vụ nhỏ nhất.

Ví dụ

Dữ liệu:
5
2 5 7 8 4
4 9 10 10

Kết qủa
18

Dữ liệu:
4
5 7 8 4
50 50 50

Kết qủa
24

Hướng dẫn: Gọi F[i] là thời gian ít nhất để i người đầu tiên mua vé xong.

Ta có công thức QHĐ F[i] := Max(F[i-1] + T[i], F[i-2] + R[i-1]).

Độ phức tạp O(n).

4. Nối mạng (http://vn.spoj.pl/problems/NKCABLE)

Các học sinh khi đến thực tập trong phòng máy tính thường hay chơi trò chơi điện tử trên mạng. Để ngăn ngừa, người trực phòng máy đã ngắt tất cả các máy tính ra khỏi mạng và xếp chúng thành một dãy trên một cái bàn dài và gắn chặt máy xuống mặt bàn rồi đánh số thứ tự các máy từ 1 đến N theo chiều từ trái sang phải. Các học sinh tinh nghịch không chịu thua, họ đã quyết định tìm cách nối các máy trên bàn bởi các đoạn dây nối sao cho mỗi máy được nối với ít nhất một máy khác. Để tiến hành công việc này, họ đã đo khoảng cách giữa hai máy liên tiếp. Bạn hãy giúp các học sinh này tìm cách nối mạng thoả mãn yêu cầu đặt ra sao cho tổng độ dài cáp nối phải sử dụng là ít nhất.

Dữ liệu

- Dòng đầu tiên chứa số lượng máy N ($1 \le N \le 25000$).
- Dòng thứ i trong số N-1 dòng tiếp theo chứa các khoảng cách từ máy i đến máy i+1 (i=1,2,...,N-1). Giả thiết rằng khoảng cách từ máy 1 đến máy N không vượt quá 10^6.

Kết quả

Ghi ra độ dài của cáp nối cần sử dụng.

Ví dụ

Dữ liệu:
6
2
2
3
2
2

Kết qủa
7

Hướng dẫn: Gọi F[i] là độ dài của cách nối mạng ngắn nhất xét từ máy 1 ... i.

Ta có F[i] := Min(F[i-2], F[i-1]) + L[i-1];

Thời gian O(n)

5. Mua vé tàu hỏa (http://vn.spoj.pl/problems/QBTICKET)

Tuyến đường sắt từ thành phố A đến thành phố B đi qua một số nhà ga. Tuyến đường có thể biểu diễn bởi một đoạn thẳng, các nhà ga là các điểm trên đó. Tuyến đường bắt đầu từ A và kết thúc ở B, vì thế các nhà ga sẽ được đánh số bắt đầu từ A (có số hiệu là 1) và B là nhà ga cuối cùng.

Giá vé đi lại giữa hai nhà ga chỉ phụ thuộc vào khoảng cách giữa chúng. Cách tính giá vé như sau: Khoảng cách giữa hai nhà ga (X)

Khoảng cách 0 < X <= L1 -> Giá vé C1

Khoảng cách 0 < X <= L2 -> Giá vé C2

Khoảng cách 0 < X <= L3 -> Giá vé C3

Nghĩa là với các giá vé C1, C2, C3 tương ứng bạn sẽ đi quảng đường tối đa là L1, L2, L3.

Vé để đi thẳng từ nhà ga này đến nhà ga khác chỉ có thể đặt mua nếu khoảng cách giữa chúng không vượt quá L3. Vì thế nhiều khi để đi từ nhà ga này đến nhà ga khác ta phải đặt mua một số vé. Hơn thế nữa, nhân viên đường sắt yêu cầu hành khách chỉ được giữ đúng một vé khi đi trên tàu và vé đó sẽ bị huỷ khi hành khách xuống tàu.

Yêu cầu: Tìm cách đặt mua vé để đi lại giữa hai nhà ga cho trước với chi phí mua vé là nhỏ nhất

Input

Dòng đầu tiên ghi các số nguyên L1, L2, L3, C1, C2, C3 (1 <= L1 <= L2 <= L3 <= 10^9; 1 <= C1 <= C2 <= C3 <= 10^9) theo đúng thứ tự liệt kê ở trên.
Dòng thứ hai chứa số lượng nhà ga N (2 <= N <= 100000)
Dòng thứ ba ghi hai số nguyên s, f là các chỉ số của hai nhà ga cần tìm cách đặt mua vé với chi phí nhỏ nhất để đi lại giữa chúng.
Dòng thứ i trong số N - 1 dòng tiếp theo ghi số nguyên là khoảng cách từ nhà ga A (ga 1) đến nhà ga thứ i + 1.

Output

Gồm 1 dòng duy nhất ghi chi phí nhỏ nhất tìm được

Example

Input:
3 6 8 20 30 40
7
2 6
3
7
8
13
15
23

Output:
70

Hướng dẫn: Gọi F[i] là chi phí ít nhất để đi đến ga i trên chặng từ s đến f.

Ta có F[i] := Min(F[i], F[j] + Chi phí từ j -> i)

Chú ý: Cài đặt khéo léo tránh TLE (Time Limit Exceeded)

6. Dạo chơi bằng xe buýt (http://vn.spoj.pl/problems/KMBUS)

Một tuyến đường ở thành phố có các bến xe bus ở từng km tuyến đường. Mỗi lần qua bến, xe đều đỗ để đón khách. Mỗi bến đều có điểm xuất phát. Một xe chỉ chạy không quá B km kể từ điểm xuất phát của nó. Hành khách khi đi xe sẽ phải trả tiền cho độ dài đoạn đường mà họ ngồi trên xe. Cước phí cần trả để đi đoạn đường độ dài i là Ci(i=1,2..B). Một du khách xuất phát từ 1 bến nào đó muốn đi dạo L km theo tuyến nói trên. Hỏi ông ta phải lên xuống xe như thế nào để tổng số tiền phải trả là nhỏ nhất có thể.

Dữ liệu vào:

Dòng đầu ghi 2 số nguyên dương B, L.
Dòng thứ i trong số B dòng tiếp theo ghi 1 số nguyên dương Ci ($1 \le i \le B$).

Kết qủa

Một dòng duy nhất là số tiền nhỏ nhất phải trả

Giới hạn

$0 \le B \le 100$
$0 \le L \le 10000$
$0 \le Ci \le 100$

Ví dụ

Dữ liệu:
5 7
3
4
6
9
22
Kết qủa
14

Hướng dẫn: Gọi F[i] là số tiền ít nhất người đó phải trả khi đi i km.

Ta có công thức QHĐ: F[i] := Min(F[i], F[i-j] + a[j]) với j <= i

7. Đổi tiền (http://vn.spoj.pl/problems/DTDOI)

Bạn được cho một tập hợp các mệnh giá tiền. Tập hợp luôn chứa phần tử mang gía trị 1. Mỗi mệnh giá có vô hạn các đồng tiền mang mệnh giá đó. Cho số tiền S, hãy tìm cách đổi S thành ít đồng tiền nhất, sao cho mỗi đồng tiền có mệnh giá thuộc vào tập hợp đã cho.

Input

Dữ liệu vào gồm 2 dòng:

- Dòng 1: Hai số nguyên dương N (số phần tử của tập hợp mệnh giá tiền) và S (số tiền cần đổi) ($1 \le N \le 100$; $1 \le S \le 10^9$).
- Dòng 2: N số nguyên dương biểu thị mệnh giá của các phần tử trong tập hợp (giá trị không vượt quá 100).

Output

Dữ liệu ra gồm một số nguyên duy nhất là số đồng tiền ít nhất có thể đổi được.

Example

Input:
2 3
1 2

Output:
2

Hướng dẫn: Với S khá lớn, ta sẽ dùng thuật toán tham lam cho đến khi S "đủ nhỏ" sau đó tiến hành QHĐ để đảm bảo cho kết quả chính xác.

8. Hội trường (http://vn.spoj.pl/problems/NKREZ)

Nhà trường có một phòng hội trường. Có những yêu cầu muốn sử dụng phòng hội trường này, mỗi yêu cầu cho biết thời điểm bắt đầu và thời điểm kết thúc. Nhà trường có thể chấp nhận hoặc từ chối đối với một yêu cầu.
Yêu cầu: hãy giúp nhà trường chọn các yêu cầu sử dụng hội trường sao cho tổng thời gian hội trường được sử dụng là lớn nhất.

Dữ liệu

Dòng đầu tiên chứa một số nguyên dương n ($n \le 10000$), số yêu cầu.
Mỗi dòng trong số n dòng tiếp theo chứa 2 số nguyên dương p và k ($0 \le p < k \le 30000$), mô tả một yêu cầu bắt đầu tại thời điểm p và kết thúc tại thời điểm k.

Kết quả

Gồm một dòng duy nhất là tổng thời gian lớn nhất mà hội trường được sử dụng

Ví dụ

Dữ liệu:

12
1 2
3 5
0 4
6 8
7 13
4 6
9 10
9 12
11 14
15 19
14 16
18 20

Kết qủa
16

Hướng dẫn: - Sắp xếp N yêu cầu tăng dần theo thời gian kết thúc.

- Gọi F[i] là tổng thời gian sử dụng nhiều nhất của hội trường xét từ yêu cầu 1 đến i.

- Ta có F[i] := Max(F[i-1], F[j] + k[i] - p[i]) với j là yêu cầu sao cho k[j] <= p[i] và k[j] lớn nhất trong tất cả các yêu khác thỏa mãn điều kiện đó.

9. Thang máy vũ trụ (http://vn.spoj.pl/problems/ELEVATOR)

Những con bò muốn đi vào vũ trụ! Chúng muốn đến được quỹ đạo bằng cách xây một kiểu thang máy: một cái tháp khổng lồ làm bằng các khối chồng lên nhau. Chúng có K ($1 \le K \le 400$) loại khối có thể xây tháp. Mỗi khối loại i có chiều cao h_i ($1 \le h_i \le 100$) và có số lượng c_i ($1 \le c_i \le 10$). Do khả năng bị phá hủy bởi các tia vũ trụ, không có phần nào của khối loại i có thể vượt qua độ cao a_i ($1 \le a_i \le 40000$).

Giúp những con bò xây thang máy cao nhất có thể bằng cách chồng các khối lên nhau theo luật trên.

Input

* Dòng 1: Một số nguyên: K
* Dòng 2..K+1: Mỗi dòng chứa 3 số nguyên được phân cách bởi khoảng trắng: h_i, a_i, và c_i. Dòng i+1 miêu tả loại khối i.

Output

* Dòng 1: Một số nguyên H, chỉ độ cao lớn nhất của tháp có thể xây được.

Example

Input:
3
7 40 3
5 23 8
2 52 6

Output:
48

Giải thích: Từ dưới lên: 3 khối loại 2, 3 khối loại 1, 6 khối loại 3. Chồng 4 khối loại 2 & 3 loại 1 không hợp lệ vì đỉnh của khối loại 1 vượt quá độ cao 40.

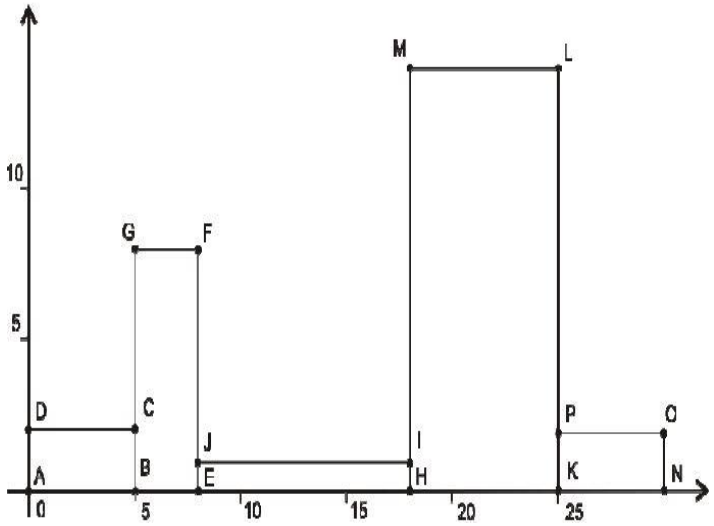
Hướng dẫn: Vì khối loại i không thể vượt quá chiều cao a[i] nên với a[i] càng bé thì khối loại i phải được xếp càng "thấp". Ta sẽ tiến hành sắp xếp h, a, c theo chiều tăng dần của a. Dùng mảng F[i, j] : boolean với ý nghĩa F[i, j] = true tức là ta đã dùng i loại khối đầu tiên và xây được thang máy có chiều cao là j.

Cách tính F[i, j] như sau:

```
-------------------------------------------------------------------------------
F[i, j] := (k = 0) or ((k > 0) and (F[i-1, k]);   với k = j - h[i] * t
                                                  t : số khối loại i đem dùng
-------------------------------------------------------------------------------
```

10. Rectangles Perimeter (http://vn.spoj.pl/problems/MMAXPER)

Cho n hình chữ nhật đánh số từ 1 đến n, các hình chữ nhật này được đặt tiếp xúc với trục OX và nằm kề nhau từ trái qua phải theo thứ tự chỉ số.
Mỗi hình chữ nhật có thể tiếp xúc với trục Ox theo bất kỳ cạnh cạnh (xem hình).
Cần tính độ dài lớn nhất của đường gấp phía trên (xem hình)



INPUT

Dòng đầu tiên ghi số hình chữ nhật n, n dòng tiếp theo mỗi dòng ghi hai số a_i và b_i, độ dài cạnh của hình chữ nhật thứ i.
Ràng buộc : 0 < n < 1000; 0 < a_i < b_i < 1000, với i = 1, 2, …, n.

OUTPUT

Ghi ra độ dài lớn nhất tìm được

Ví dụ:

Input

5
2 5
3 8
1 10
7 14
2 5

Output

68

Giải thích

Cách xếp mà thu được chiều dài lớn nhất là hình trên. Cạnh phía trên gồm các đoạn DC, CG, GF, FJ, JI, IM, ML, LP, và PO.
Độ dài của đoạn này là 5 + 6 + 3 + 7 + 10 + 13 + 7 + 12 + 5 = 68

11. Xâu con liên tiếp đối xứng dài nhất

Cho một xâu S <= 1000 kí tự. Tìm palindrome dài nhất là xâu con của S (xâu con là một dãy các kí tự liên tiếp).

Hướng dẫn:

Đây cũng là một bài cơ bản với nhiều cách làm.

Cách 1: QHĐ

Dùng mảng F[i, j] có ý nghĩa: F[i, j] = true/false nếu đoạn gồm các kí tự từ i đến j của S có/không là palindrome.
Ta có công thức là:

* F[i, i] = True
* F[i, j] = F[i+1, j-1]; (nếu s[i] = s[j])
* F[i, j] = False; (nếu s[i] <> s[j])

Đoạn chương trình như sau:

```
-------------------------------------------------------------
FillChar( F, sizeof(F), false );
for i := 1 to n do F[i, i] := True;
for k := 1 to (n-1) do
    for i := 1 to (n-k) do
          begin
              j := i + k;
              F[i, j] := ( F[i+1, j-1] ) and (s[i] = s[j] );
          end;
-------------------------------------------------------------
```

Kết quả là : Max(j-i+1) <=j thỏa F[i, j] = True.
Độ phức tạp thuật toán là $0(N^2)$.

Cách 2: Duyệt có cận.

Ta xét từng vị trí i:

- xem a[i] có phải là tâm của Palindrome có lẻ kí tự không? (ví dụ MADAM có tâm là kí tự D)

- xem a[i] và a[i+1] có phải là tâm của Palindrome có chẵn kí tự không?

(ví dụ Palindrome ABBA có tâm là 2 kí tự BB)

với mỗi kí tự ta tìm palindrome dài nhất nhận nó là tâm, cập nhập lại kết quả khi duyệt. Ta duyệt từ giữa ra để dùng kết quả hiện tại làm cận.

Đoạn chương trình như sau:

```
procedure Lam;
var i, j : Longint ;
{ }
procedure try( first, last : Longint );
var đ : Longint;
Begin
   if first = last then
       begin
         đ := 1;
         dec(first);
         inc(last);
       end
   else đ := 0;
   repeat
      if (first < 1) or (last > N) then break;
      if s[i] = s[j] then
           begin
             đ := đ + 2;
             first := first - 1;
             last := last + 1;
           end
     else break;
   until false;
   if max < dd then max := dd;
End;
{ }

Begin
   i := n div 2;
   j := n div 2 + 1;
   max := 1;
   while (i > max div 2) and (j <= N-max div 2) do
        begin
           if i > max div 2 then
                 begin
                    try( i, i );
                    try( i, i+1 );
                 end;
           if j <= N - max div 2 then
                 begin
                    try( j, j );
                    try( j, j+1 );
                 end;
           i := i - 1;
           j := j + 1;
        end;
End;
-----------------------------------------------------------
```

Cách làm này có độ phức tạp: max*(N-max). Vì vậy nó chạy nhanh hơn cách QHĐ trên, thời gian chậm nhất khi max = N/2 cũng chỉ mất $N^2/4$ nhanh gấp 4 lần cách dùng QHĐ. Nhờ vậy, chúng ta biết là: không phải lúc nào QHĐ cũng chấp nhận được về mặt thời gian và không phải lúc nào duyệt lúc nào cũng chậm.

Bài trên còn có một cách NlogN nữa là dùng Suffix Aray, thậm chí có cách O(N) là sử dụng Suffix Tree và thuật toán tìm LCA. Đương nhiên cách cài đặt không hề dễ dàng! Các bạn có thể tự tìm hiểu!

12. Chia một xâu thành ít nhất các Palindrome (độ dài ≤ 1000)

Bài này phức tạp hơn bài trên, cách làm thì vẫn là QHĐ.
Gọi F[i] là số palindrome ít nhất mà đoạn 1..j chia thành được.
Ta có công thức: F[i] = max(F[j] + 1; j < i thỏa mãn : đoạn j+1..i là palindrome)
Đoạn chương trình như sau:

```
F[0] := 0;
for i := 1 to n do
    begin
       for j := i-1 downto 0 do
           if (đoạn j+1..i là palindrome) then
           F[i] := max( F[i], F[j]+1 );
    end;
```

Hai vòng for lồng nhau mất $O(N^2)$, phần kiểm tra đoạn j+1..i là palindrome hay không mất O(N), vậy độ phức tạp thuật toán là $O(N^3)$. Sẽ không được khả thi nếu N = 1000. Để giảm độ phức tạp thuật toán, ta sử dụng mảng L[i, j] có ý nghĩa tương tự như mảng F[i, j] ở bài 1. QHĐ lập mảng L[i, j] mất N^2. Tổng cộng là $O(N^2)$ vì mỗi lần kiểm tra chỉ mất O(1).
Có thể cải tiến bằng cách dùng hai mảng một chiều L[i] và C[i] có ý nghĩa:
* L[i] là độ dài lớn nhất của palindrome độ dài lẻ nhận s[i] làm tâm;
* C[i] là độ dài lớn nhất của palindrome độ dài chẵn nhận s[i] và s[i+1] làm tâm;
L[i] và C[i] có thể tính được bằng cách 2 bài 2 trong $O(N^2)$. Phần kiểm tra ta viết lại như sau:

```
-------------------------------------------------------------
Function  is_palindrome(i, j : integer) : boolean;
var t : integer;
Begin
  t := j-i+1;
  if odd (t) then is_palindrome := (L[(i+j) div 2] >= n)
  else is_palindrome := (C[(i+j) div 2] >= n)
end;
-------------------------------------------------------------
```

Vậy thuật toán của chúng ta có độ phức tạp tính toán là $O(N^2)$, chi phí bộ nhớ là O(N).

13. Đếm chuỗi đối xứng (http://vn.spoj.pl/problems/QBPAL)

Trong một buổi học viết chữ, Bờm phát hiện trong một số từ khi bỏ đi một số ký tự thì đọc ngược hay đọc xuôi đều giống nhau.
Ví dụ từ IOICAMP, khi xóa đi các chữ cái C,A,M,P, thì còn lại IOI là một từ đối xứng.

Bờm cảm thấy thú vị, và cậu tiếp tục thử xóa các ký tự khác, kết quả là có thêm nhiều từ đối xứng nữa: II, I, O, C… Nhưng nếu với một từ dài, cứ thử từng cách xóa như vậy thì thật mất thời gian. Bạn hãy viết chương trình giúp Bờm tính số cách xóa sao cho từ thu được đối xứng. Hai cách xóa chỉ khác nhau bởi thứ tự xóa các ký tự thì coi như trùng nhau.

Input

Một dòng duy nhất là từ cần tính số cách xóa, từ này chỉ chứa các chữ cái in hoa A, B, .., Z. (Độ dài từ không quá 120)

Output

Một số duy nhất là số cách xóa.

Example

Input:
IOICAMP

Output:
9

Hướng dẫn: Việc đếm số cách xóa các ký tự để tạo thành Palindrome cũng chính là đếm xem xâu ban đầu có bao nhiêu xâu con (không cần liên tiếp) là xâu Palindrome

Gọi F[i, j] : số palindrome là xâu con của đoạn i..j. Ta có công thức:

```
-----------------------------------------------------------------
F[i, i] := 1;
If s[i] = s[j] then  F[i, j] := F[i+1, j] + F[i, j-1] + 1
If s[i] <> s[j] then F[i, j] := F[i+1, j] + F[i, j-1] - F[i+1, j-1]
-----------------------------------------------------------------
```

Đoạn chương trình trên chỉ có tính mô phỏng, muốn hoàn thiện bạn phải cài đặt các phép tính cộng trừ số lớn vì kết quả có thể lên tới 2^N - 1. Độ phức tạp của thuật toán là $O(N^2)$.

14. Palindrome - IOI 2000

Cho một xâu, hỏi phải thêm vào nó ít nhất bao nhiêu xâu kí tự để nó trở thành một palindrome (độ dài <= 5000).
Gọi F[i, j] là số phép biến đổi ít nhất cần thêm vào đoạn i..j để đoạn i..j trở thành palindrome.
Ta có công thức :
- F[i, i] = 0;
- Nếu s[i] = s[j] thì F[i, j] = F[i+1, j-1]
- Nếu s[i] <> s[j] thì F[i, j] = Min(F[i, j-1], F[i+1, j]) + 1;

15. Chuỗi đối xứng (http://vn.spoj.pl/problems/NKPALIN)

Một chuỗi được gọi là đối xứng (palindrome) nếu như khi đọc chuỗi này từ phải sang trái cũng thu được chuỗi ban đầu.
Yêu cầu: tìm một chuỗi con đối xứng dài nhất của một chuỗi s cho trước. Chuỗi con là chuỗi thu được khi xóa đi một số ký tự từ chuỗi ban đầu.

Dữ liệu vào

Gồm một dòng duy nhất chứa chuỗi s, chỉ gồm những chữ cái in thường.

Kết quả

Gồm một dòng duy nhất là một xâu con đối xứng dài nhất của xâu s. Nếu có nhiều kết quả, chỉ cần in ra một kết quả bất kỳ.

Giới hạn

Chuỗi s có độ dài không vượt quá 2000.

Ví dụ

Dữ liệu mẫu

lmevxeyzl

Kết qủa

level

Hướng dẫn: Gọi F[i, j] là chuỗi con đối xứng dài nhất của chuỗi xét từ i..j

Nếu s[i] = s[j] thì F[i, j] := F[i+1, j-1] + 2 ngược lại F[i, j] := Max(F[i+1, j], F[i, j-1]);

Cơ sở quy hoạch động: - F[i, i] = 1;

- Nếu s[i] = s[j] thì F[i, i+1] = 2 ngược lại F[i, i+1] = 1;

16. Xâu con chung dài nhất (http://vn.spoj.pl/problems/QBSTR)

Xâu ký tự X được gọi là xâu con của xâu ký tự Y nếu ta có thể xoá đi một số ký tự trong xâu Y để được xâu X.
Cho biết hai xâu ký tự A và B, hãy tìm xâu ký tự C có độ dài lớn nhất và là con của cả A và B.

Input

Dòng 1: chứa xâu A
Dòng 2: chứa xâu B

Output

Chỉ gồm một dòng ghi độ dài xâu C tìm được

Example

Input:
abc1def2ghi3
abcdefghi123

Output:
10

Hướng dẫn: Gọi F[i, j] là độ dài xâu con chung dài nhất của xâu gồm i ký tự đầu của xâu A và j ký tự đầu của xâu B. Ta có công thức QHĐ

```
if A[i] = B[j] then
     F[i, j] := F[i-1, j-1] + 1
else
     F[i, j] := Max(F[i-1, j], F[i, j-1], F[i-1, j-1]);
```

17. Dãy con chung không liền kề dài nhất (http://vn.spoj.pl/problems/LNACS)

Dãy C = $c_1, c_2, ..., c_k$ là dãy con không liền kề của dãy A = $a_1, a_2, ..., a_m$ nếu C có thể nhận được bằng cách chọn một dãy các phần tử không liền kề của A, nghĩa là tìm dược dãy các chỉ số $i_1, i_2, ..., i_k$ sao cho:

$1 \le i_1, i_2, ..., i_k \le m;$
$i_1 < i_2 - 1, i_2 < i_3 - 1, ..., i_{k-1} < i_k - 1;$
$c_1 = a_{i1}, c_2 = a_{i2}, c_k = a_{ik}.$

Ta gọi độ dài của dãy số là số phần tử của nó.
Cho hai dãy:

A = $a_1, a_2, ..., a_m$ và

B = $b_1, b_2, ..., b_n$

Dãy C được gọi là dãy con chung không liền kề của hai dãy A và B nếu như nó vừa là dãy con không liền kề của A, vừa là dãy con không liền kề của B.

Yêu cầu

Cho hai dãy số A và B. Hãy tìm độ dài của dãy con chung không liền kề dài nhất của hai dãy đã cho.

Dữ liệu

- Dòng đầu tiên chứa hai số nguyên dương m và n ($2 \le m, n \le 10^3$) được ghi cách nhau bởi dấu cách, lần lượt là số lượng phần tử của dãy A và dãy B.
- Dòng thứ i trong m dòng tiếp theo chứa số nguyên không âm a_i ($a_i \le 10^4$), i = 1, 2, ..., m.
- Dòng thứ j trong n dòng tiếp theo chứa số nguyên không âm b_j ($b_j \le 10^4$), j = 1, 2, ..., n.

Kết quả

Ghi ra trên một dòng duy nhất độ dài của dãy con chung không liền kề dài nhất của hai dãy A và B

Input:

```
4 5
4
9
2
4
1
9
7
3
4
```

Output:

```
2
```

Hướng dẫn: Đây là bài số 1 trong đề thi HSG QG Tin học năm 2010

Bài toán này hoàn toàn tương tự với bài toán 16. Các bạn tự code!

18. Đoạn cao trào của bản nhạc (http://vn.spoj.pl/problems/NKTHEME)

Trong một bản nhạc thường có những đoạn nhạc được sử dụng nhiều lần (ít nhất 2 lần). Những đoạn đó gọi là "đoạn cao trào". Do có thể sử dụng nhiều giọng khác nhau (son, la, si...) nên nốt đầu tiên của các lần xuất hiện có thể khác nhau, nhưng chênh lệch độ cao giữa hai nốt liên tiếp thì như nhau.
Chẳng hạn 1 2 5 4 10 và 4 5 8 7 13 được coi là thể hiện một đoạn cao trào, vì chúng cùng chênh lệch độ cao : +1,+3,-1,+6.
Trong một bản nhạc, đoạn cao trào còn phải thỏa mãn những điều kiện:

- Đoạn cao trào phải có từ 5 nốt nhạc trở lên.
- Những lần xuất hiện của đoạn không được chồng lên nhau (không có nốt nhạc chung).

Yêu cầu: Cho một bản nhạc, tìm đoạn cao trào dài nhất.

Dữ liệu vào

- Dòng 1: chứa số nguyên n, số nốt nhạc (n ≤ 5000).
- Một số dòng sau chứa n số nguyên trong phạm vi 1..88, thể hiện n nốt nhạc.

Kết quả

In ra 1 dòng duy nhất chứa 1 số nguyên là độ dài đoạn cao trào dài nhất. Nếu không tìm được đoạn cao trào, in ra 0.

Ví dụ

Dữ liệu mẫu
30
25 27 30 34 39 45 52 60 69 79 69 60 52 45 39 34 30 26 22 18
82 78 74 70 66 67 64 60 65 80

Kết qủa
5

5 nốt cuối dòng 1 và 5 nốt đầu dòng 2 thể hiện đoạn cao trào dài nhất.

Hướng dẫn: Gọi F[i, j] là đoạn cao trào dài nhất của bản nhạc xét từ 1..i và i+5..j. Với tư tưởng của bài "xâu con chung dài nhất" ta có thể giải quyết bài toán nếu trên! Kết quả là Max(F[i, j])

19. Blast (http://vn.spoj.pl/problems/MBLAST)

Cho hai xâu A, B. Mở rộng của 1 xâu X là xâu thu được bằng cách chèn (0,1,2 ..) kí tự trống vào xâu. Ví dụ : X là 'abcbcd', thì 'abcb-cd', '-a-bcbcd-' và 'abcb-cd-' là các mở rộng của X. (Dấu cách kí hiệu bằng '-').
A1,B1 là mở rộng của A và B, và giả sử chúng cùng độ dài. Khoảng cách giữa A1 và B1 là tổng khoảng cách giữa các kí tự cùng vị trí. Nếu hai kí tự không là dấu cách thì khoảng cách giữa 2 kí tự này là trị tuyệt đối mã ASCII của chúng. Còn ngược lại, khoảng cách là 1 số K cố định.
Cho hai xâu A, B. Tìm khoảng cách nhỏ nhất giữa hai xâu mở rộng của nó.

Input

Dòng 1 chứa A, dòng 2 chứa B, chỉ gồm chữ thường a-z và số kí tự <=2000.
Dòng thứ 3 là số K, khoảng cách của 1 kí tự bất kỳ với kí tự trống, 1 ≤ K ≤ 100.

Output

Khoảng cách nhỏ nhất.

Sample

BLAST.IN	BLAST.OUT
cmc snmn 2	10
koiv ua 1	5
mj jao 4	12

Hướng dẫn: Gọi F[i, j] là khoảng cách nhỏ nhất giữa 2 xâu mở rộng của A xét từ 1..i và B xét từ 1..j. Ta có

$$F[i, j] := \text{Min} \begin{cases} F[i-1, j-1] + abs(ORD(s1[i]) - ORD(s2[j])) \\ F[i-1, j-1] + 2 * K \\ F[i-1, j] + K \\ F[i, j-1] + K \end{cases}$$

Sau đây chúng ta sẽ xét đến chuỗi các bài toán QHĐ liên quan tới dãy số, bảng số.

20. Dãy con dài nhất (http://vn.spoj.pl/problems/NKMAXSEQ)

Cho dãy số nguyên $a_1, a_2, \ldots, a_n$.

Dãy số $a_i, a_{i+1}, \ldots, a_j$ với $1 \le i \le j \le n$ được gọi là dãy con của dãy số đã cho và khi đó, j-i+1 được gọi là độ dài, còn $a_i+a_{i+1}...+a_j$ được gọi là trọng lượng của dãy con này.

Yêu cầu: cho số nguyên p, trong số các dãy con của dãy số đã cho có trọng lượng không nhỏ hơn p hãy tìm dãy con có độ dài lớn nhất.

Dữ liệu vào

- Dòng đầu tiên ghi hai số nguyên n và p cách nhau bởi dấu cách.
- Dòng thứ i trong số n dòng tiếp theo chứa số nguyên a_i là số hạng thứ i của dãy số đã cho, i = 1, 2, …, n.

Kết quả

Ghi ra số nguyên k là độ dài của dãy con tìm được (qui ước: nếu không có dãy con nào thỏa mãn điều kiện đặt ra thì k = -1).

Hạn chế

Trong tất cả các test: $1 \le n \le 50000$; $|a_i| \le 20000$; $|p| \le 10^9$. Có 50% số lượng test với $n \le 2000$.

Ví dụ

Dữ liệu mẫu

5 6
-2
3
2
-2
3

Kết qủa

4

Dữ liệu mẫu

4 9
2
3
2
-2

Kết qủa

-1

Hướng dẫn: Gọi F[i] trọng lượng của dãy xét từ 1..i. Sử dụng mảng D với ý nghĩa: D[i] = true nếu i là vị trí "đáng quan tâm", đây là vị trí làm cho trọng lượng của 1 dãy liên tục giảm so với vị trí kề trước nó. Cách xác định vị trí "đáng quan tâm" được thực hiện qua đoạn chương trình sau:

```
-------------------------------------
       D[0] := true;
       min := 0;
       For i:=1 to n do
           begin
             If F[i] < min then
                   begin
                     min := F[i];
                     D[i] := true;
                   end;
           end;
-------------------------------------
```

Đoạn chương trình tìm ra độ dài lớn nhất.

```
-------------------------------------
         max_length := -1;
         pos := n;
         For i := n downto 0 do
            If D[i] then
               begin
                 For j := pos downto i+1 do
                    If F[j] - F[i] >= p then
                        begin
                          If j - i > max_length then max_length := j - i;
                          break;
                        end;
                    pos := j;
               end;
-------------------------------------
```

Đây là bài toán cơ bản và có nhiều ứng dụng. Tuy code ngắn nhưng các bạn cần đọc và hiểu rõ bản chất của nó!!!

21. Dãy con ngắn nhất

Cho dãy số nguyên a_1, a_2, …, a_n.
Dãy số a_i, a_{i+1}, …, a_j với $1 \le i \le j \le n$ được gọi là dãy con của dãy số đã cho và khi đó, j-i+1 được gọi là độ dài, còn $a_i+a_{i+1}...+a_j$ được gọi là trọng lượng của dãy con này.
Yêu cầu: cho số nguyên p, trong số các dãy con của dãy số đã cho có trọng lượng không nhỏ hơn p hãy tìm dãy con có độ dài bé nhất.

Dữ liệu vào

- Dòng đầu tiên ghi hai số nguyên n và p cách nhau bởi dấu cách.
- Dòng thứ i trong số n dòng tiếp theo chứa số nguyên a_i là số hạng thứ i của dãy số đã cho, i = 1, 2, …, n.

Kết quả

Ghi ra số nguyên k là độ dài của dãy con tìm được (qui ước: nếu không có dãy con nào thỏa mãn điều kiện đặt ra thì k = -1).

Hạn chế

Trong tất cả các test: $1 \le n \le 50000$; $0 \le a_i \le 20000$; $|p| \le 10^9$.

Hướng dẫn: Cần chú ý điểm khác biệt giữa bài toán này và bài "Dãy con dài nhất" ở chỗ giá trị của a[i] ≥ 0, và kết quả là ngắn nhất. Bài toán này có thể giải quyết trong O(n) như sau:

```
-----------------------------------------------------------------
         i := 1; j := 1;
         s := a[1];
         min := maxlongint;
         while (i <= j) and (j <= n) do
              begin
                if s < k then
                      begin
                        inc(j);
                        inc(s, a[j]);
                      end
                else
                      begin
                        if min > j - i + 1 then min := j - i + 1;
                        dec(s, a[i]);
                        inc(i);
                      end;
              end;
-----------------------------------------------------------------
```

Các bạn tự kiểm tra tính đúng đắn của thuật toán!

22. Help the PM! (http://vn.spoj.pl/problems/HELPPM)

GIÚP NGÀI THỦ TƯỚNG!

Năm 2050, lúc này Lê Đôn Khuê đã trở thành Thủ tướng Việt Nam. Ông nhận được một đề nghị cho phép khai thác K m^3 gỗ ở một khu rừng để xuất khẩu. Khu rừng này có dạng hình chữ nhật MxN km^2. Để tiện quản lý thì người ta chia khu vực thành MxN vùng (M hàng, N cột) và lượng gỗ tại mỗi khu vực (tính theo m^3) đã được biết. Các hàng được đánh số 1 đến M từ trên xuống dưới. Các cột được đánh số từ 1 đến N từ trái sang phải. Tọa độ của vùng nằm tại hàng i, cột j là (i, j).

Ngài Thủ tướng quyết định cho phép khai thác và vùng khai thác để dễ quản lý nên là một vùng hình chữ nhật. Ngài Thủ tướng muốn tìm một phương án khai thác gỗ sao cho diện tích khai thác là nhỏ nhất và vẫn đủ lượng gỗ cần thiết để xuất khẩu.

Do lâu ngày không lập trình nên ngài Thủ tướng cần đến sự giúp đỡ của các bạn. Các bạn hãy giúp ngài Thủ tướng nào.

Dữ liệu

- Dòng thứ nhất ghi ba số M, N, K ($1 \le M, N \le 500$, $1 \le K \le 10^9$).
- Dòng thứ i trong M dòng tiếp theo ghi N số nguyên không âm, trong đó số thứ j cho biết lượng gỗ tại khu vực (i, j). Biết lượng gỗ tại mỗi khu vực không vượt quá 10^4 m^3.

Kết quả

Nếu không tồn tại vùng khai thác gỗ nào cho đủ gỗ xuất khẩu, in ra -1. Ngược lại in ra:

- Dòng thứ nhất ghi diện tích nhỏ nhất có thể của vùng khai thác gỗ.
- Dòng tiếp theo ghi bốn số là chỉ số của góc trái trên và góc phải dưới của vùng khai thác gỗ. Nếu có nhiều vùng cùng thỏa mãn thì in ra tọa độ của một vùng bất kỳ.

Ví dụ

Dữ liệu
3 3 19
5 4 0
4 7 0
0 0 2

Kết quả
4
1 1 2 2

Hướng dẫn: Xét hai cạnh L, R của bảng số.

For L := 1 to N do

    For R := L to N do

Bây giờ ta có dãy m phần tử, phần tử i có giá trị bằng A[i, L] + A[i, L+1] + ... + A[i, R]

Áp dụng phiên bản một chiều của bài toán để giải - bài toán "Dãy con ngắn nhất".

23. Dãy con có tổng bằng S

Cho dãy các số nguyên A_1, A_2, A_3, ...A_n. Hãy tìm một dãy con của dãy đó (các phần tử không cần liên tiếp) có tổng bằng S.

Hướng dẫn: Đặt L[i, t] = true nếu có thể tạo ra tổng t từ một dãy con của dãy gồm các phần tử A_1, A_2, ..A_i. Ngược lại thì L[i, t] = false. Nếu L[n,S] = true thì đáp án của bài toán trên là "có".

Ta có thể tính L[i, t] theo công thức: L[i, t] = true nếu L[i–1,t] = true hoặc L[i–1,t–a[i]] = true.

Cài đặt: Nếu áp dụng luôn công thức trên thì ta cần dùng bảng phương án hai chiều. Ta có thể nhận xét rằng để tính dòng thứ i, ta chỉ cần dòng i-1. Bảng phương án khi đó chỉ cần 1 mảng 1

chiều L[0..S] và được tính như sau:

```
L[0]:=1;
for  i := 1 to n do
    for  t := S downto a[i] do
        L[t] := L[t] xor L[t–a[i]];
```

Dễ thấy đề bài chỉ là cách phát biểu khác của Bài 1: "Đi xem phim". Chi phí không gian của cách cài đặt trên là O(m), chi phí thời gian là O(n.m), với m là tổng của n số. Hãy tự kiểm tra xem tại sao vòng for thứ 2 lại là for downto chứ không phải là for to.

24. Dãy con liên tiếp có tổng lớn nhất

Cho dãy các số nguyên A_1, A_2, A_3, ...A_n. Hãy tìm dãy con liên tiếp có tổng lớn nhất và in ra tổng đó
Giới hạn: - $|A_i| \le 10^9$
- $N \le 100000$

Hướng dẫn: Nếu gọi F[i] là tổng các số từ 1..i, kết quả là Max(F[i] - F[j]) thì chương trình có độ phức tạp $O(N^2)$, không khả thi với N = 100000. Ta có thể cải tiến để giảm độ phức tạp xuống như sau: Max(F[i] - F[j]) = F[i] - Min(F[0], F[1], F[2], ... F[i-1]) = F[i] - F[C[i]]
với C[i] là chỉ số của phần tử nhỏ nhất đứng trước F[i]. Độ phức tạp lúc này là O(N).
Áp dụng bài toán này có thể giải quyết được bài "Ma trận con có tổng lớn nhất".

25. Dãy con không giảm dài nhất (http://vn.spoj.pl/problems/QBMSEQ)

Cho dãy số nguyên dương a_1, a_2, ..., a_n.
Dãy số: a_i, a_{i+1}, ..., a_j thỏa mãn $a_i \le a_{i+1} \le ... \le a_j$. Với $1 \le i \le j \le n$ được gọi là dãy con không giảm của dãy số đã cho và khi đó số j-i+1 được gọi là độ dài của dãy con này.

Yêu cầu: Trong số các dãy con không giảm của dãy số đã cho mà các phần tử của nó đều thuộc dãy số $\{u_k\}$ xác định bởi $u_1 = 1$, $u_{k = uk-1} + k$ ($k \ge 2$), hãy tìm dãy con có độ dài lớn nhất.

Input

Dòng đầu tiên chứa một số nguyên dương n ($n \le 10^4$).
Dòng thứ i trong n dòng tiếp theo chứa một số nguyên dương a_i ($a_i \le 10^8$) là số hạng thứ i của dãy số đã cho, i = 1, 2, ..., n.

Output

Gồm 1 dòng duy nhất ghi số nguyên d là độ dài của dãy con không giảm tìm được (quy ước rằng nếu không có dãy con nào thỏa mãn điều kiện đặt ra thì d = 0).

Example

Input:
8
2
2007
6
6
15
16
3
21
Output:
3

Hướng dẫn: Ta nhận thấy dãy { U_k } có công thức tổng quát: U_k = k * (k+1) / 2;

Gọi F[i] là độ dài dãy con không giảm dài nhất xét từ 1..i. Ta có:

- Nếu a[i] thuộc dãy { U_k } và (a[i] >= a[i-1]) và (F[i-1] > -1) thì có F[i] := F[i-1] + 1;

- Nếu a[i] không thuộc dãy thì F[i] := -1. Kết quả là Max(F[i])

26. Vị trí tốt (http://vn.spoj.pl/problems/NKSEQ)

Cho dãy số nguyên a1, a2, ..., an (1≤ n≤ 100000), mỗi số không vượt qúa 10000. Dãy số này được viết trên một vòng tròn. Nghĩa là, khi cắt vòng tròn tại vị trí j, ta thu được:
aj, aj+1,..., an, a1, a2, ..., aj–1
Vị trí j được gọi là vị trí tốt, nếu các điều kiện sau đây được thỏa mãn:

- aj > 0
- aj + aj+1 > 0
- ....
- aj + aj+1 + ... + an > 0
- aj + aj+1 + ... + an + a1 > 0
- ...
- aj + aj+1 + ... + an + a1 + a2 + ... + aj–2 > 0
- aj + aj+1 + ... + an + a1 + a2 + ... + aj–2 + aj–1 > 0

Yêu cầu: hãy đếm số vị trí tốt.

Dữ liệu vào

- Dòng đầu tiên chứa số nguyên n.
- Dòng thứ 2 chứa dãy số a1, a2,...,an.

Kết quả

In ra 1 số nguyên duy nhất là số vị trí tốt.

Ví dụ

Dữ liệu mẫu
5
0 1 -2 10 3

Kết qủa
2

Hướng dẫn:

Cách 1: Đặt
L[i] = Min(a[1], a[1] + a[2], ..., a[1] + a[2] + ... + a[i])
R[i] = Min(a[n], a[n-1] + a[n] , ..., a[i] + ... + a[n])
S[i] = a[i] + ... + a[n]
Khi đó i là vị trí tốt khi và chỉ khi R[i] > 0 AND S[i] + L[i-1] > 0
Mảng L, R, S có thể tính trong O(n).
Độ phức tạp thuật toán: O(n).
Cách trên đây khá dễ hiểu. Ngoài cách này ta cũng có một cách khác khá đơn giản như sau:

Cách 2:
Xét S = a[1] + a[2] + ... + a[n]
Nếu S <= 0 thì không có vị trí tốt, in ra 0
Nếu s > 0
Đặt F[i] = Min(a[i], a[i]+a[i+1], ..., a[i]+...+a[i-1])
Trước tiên ta tính
F[n] = Min(a[n], a[n]+a[1], ..., a[n]+...+a[n-1]) mất O(n)
Sau đó ta có nhận xét:
F[n-1] = Min(a[n-1] + F[n], a[n-1])
Vì: a[n-1]+(a[n], a[n]+a[1], ..., a[n]+...+a[n-1]) sẽ tạo thành các tổng, trừ một trường hợp là a[n-1] + (a[n]+...+a[n-1]) = a[n-1] + s.
Tuy nhiên vì s > 0 nên a[n-1] + s > a[n-1], do đó kết quả này sẽ không được tính!
Tương tự, ta có:
F[i] = Min(a[i] + F[i+1], a[i])
Do F[i] phụ thuộc vào F[i+1] nên chỉ cần dùng 1 biến.

27. Dãy con dài nhất có tổng chia hết cho K (http://vn.spoj.pl/problems/QBSEQ)

Cho một dãy gồm n (n <= 1000) số nguyên dương A_1, A_2, ..., A_n và số nguyên dương k (k <= 50). Hãy tìm dãy con (không cần liên tiếp) gồm nhiều phần tử nhất của dãy đã cho sao cho tổng các phần tử của dãy con này chia hết cho k.

Input

Dòng đầu tiên chứa hai số n, k ghi cách nhau bởi ít nhất 1 dấu trống.
Các dòng tiếp theo chứa các số A_1, A_2, ..., A_n được ghi theo đúng thứ tự cách nhau ít nhất một dấu trống hoặc xuống dòng

Output

Gồm 1 dòng duy nhất ghi số lượng phần tử của dãy con dài nhất thoả mãn

Example

Input:
10 3
2 3 5 7
9 6 12 7
11 15

Output:
9

28. Dãy con liên tiếp dài nhất có tổng chia hết cho K

Cho một dãy số gồm N số nguyên và một số nguyên dương k. Hãy tìm một dãy con dài nhất liên tiếp nhau sao cho tổng chia hết cho k.

Dữ liệu vào: từ file DAYSO.INP có dạng:

- Dòng đầu tiên là hai số N và k (N ≤ 500000; k ≤ 10000);
- Các dòng tiếp theo là N số nguyên của dãy (các số kiểu Longint), mỗi số trên một dòng.

Kết quả: ra file DAYSO.OUT gồm một dòng duy nhất chứa hai số m và s, trong đó m là độ dài lớn nhất tìm được và s là vị trí bắt đầu của dãy đó.

Ví dụ:

DAYSO.INP	DAYSO.OUT
3 1 2 3	3 1

Hướng dẫn:

- Gọi S[i] là tổng các số từ a[1] đến a[i] trong dãy.
- Để dễ tính toán và tiết kiệm bộ nhớ, ta dùng luôn mảng a như sau : a[i] := (a[i-1]+a[i]) mod k;
- Vì K ≤ 10000 nên 0 ≤ a[i] ≤ 9999. Dùng mảng Pos : [0..9999] với ý nghĩa Pos[i] là vị trí xuất hiện đầu tiên của a[i] trong dãy, nếu a[i] chưa xuất hiện thì Pos[i] = 0.
- Trong bước duyệt đến phần tử a[j], nếu Pos[a[j]] <> 0 thì ta sẽ cập nhật Max với (j - Pos[a[j]])
- Độ phức tạp O(n)

29. Dãy con tăng dài nhất (http://vn.spoj.pl/problems/LIS)

Cho một dãy số nguyên gồm N phần tử A[1], A[2], ... A[N].

Biết rằng dãy con tăng đơn điệu là 1 dãy $A[i_1]$,... $A[i_k]$ thỏa mãn

$i_1 < i_2 < ... < i_k$ và $A[i_1] < A[i_2] < .. < A[i_k]$. Hãy cho biết dãy con tăng đơn điệu dài nhất của dãy này có bao nhiêu phần tử?

Input

- Dòng 1 gồm 1 số nguyên là số N (1 ≤ N ≤ 30000).
- Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] (1 ≤ A[i] ≤ 2^{32}).

Output

Ghi ra độ dài của dãy con tăng đơn điệu dài nhất.

Example

Input:
6
1 2 5 4 6 2

Output:
4

Giải thích test ví dụ: Dãy con dài nhất là dãy A[1] = 1 < A[2] = 2 < A[4] = 4 < A[5] = 6, độ dài dãy này là 4.

Hướng dẫn: Cài đặt theo cách cải tiến: Kết hợp tìm kiếm nhị phân + QHĐ. Độ phức tạp của bài toán là O(NlogN).

30. Sắp xếp các quân bài (http://vn.spoj.pl/problems/MCARDS)

Mav có n*c quân bài; mỗi quân bài 1 màu; và mỗi màu có n quân bài. Mav muốn sắp xếp các quân bài cùng màu nằm cạnh nhau và chúng theo thứ tự tăng dần về giá trị.

Input

Dòng đầu là 2 số C (số màu), (1 ≤ C ≤ 4), và N, số quân bài cùng màu mỗi loại, (1 ≤ N ≤ 100).
N*C dòng tiếp theo mỗi dòng là 2 số nguyên X, Y; 1 ≤ X ≤ C, 1 ≤ Y ≤ N, là màu của quân bài và giá trị quân bài đó.
Thứ tự quân bài ban đầu (từ trái sang phải) chính là thứ tự dữ liệu input.

Output

Số lần chuyển bài ít nhất để thu được dãy bài được sắp theo yêu cầu trên.

Sample

CARDS.IN	CARDS.OUT
2 2 2 1 1 2 1 1 2 2	2
4 1 2 1 3 1 1 1 4 1	0
3 2 3 2 2 2 1 1 3 1 2 1 1 2	2

Hướng dẫn: Có tất cả N*C quân bài. Nếu như tất cả đều cùng 1 màu thì cách làm dễ nhận ra là tìm ra L là độ dài của "dãy tăng dài nhất". Đáp án sẽ là N*C - L. Vậy trong trường hợp bài toán này ta nên sắp xếp màu nào đứng trước? Thứ tự nào sẽ cho kết quả tối ưu? Ta sẽ thử tất cả các phương án để chọn giải pháp tối ưu! Tạm quên đi việc đề bài mã hóa các màu từ 1..4, thay vì đó ta coi D[i] là thứ tự của màu i ($1 \le i \le 4$) trong dãy sau khi sắp xếp. Ví dụ:

thứ tự	màu 1	màu 4	màu 2	màu 3
D	D[1] = 1	D[4] = 2	D[2] = 3	D[3] = 4

Sau khi có thứ tự của các màu, coi như dãy lúc này có N*C quân chỉ có một màu và giá trị của từng quân bài được tính như sau: a[i] = D[mau[i]] * 1000 + gt[i]

với: - mau[i] : màu của quân bài i.

- gt[i] : giá trị thực của quân bài trong đề cho.

Với cách đề cập ở đầu bài, ta hoàn toàn có thể giải quyết bài toán!

Độ phức tạp tính toán O(NlogN * C!) với NlogN để giải quyết bài toán dãy con tăng dài nhất và C! phép hoán vị.

31. Sequences (http://vn.spoj.pl/problems/SPSEQ)

W. là 1 dãy các số nguyên dương. Nó có các đặc điểm sau:
- Độ dài của dãy là 1 số lẻ: **L = 2*N + 1**
- **N + 1** số nguyên đầu tiên của dãy tạo thành 1 dãy tăng
- N + 1 số nguyên cuối của dãy tạo thành 1 dãy giảm
- Không có 2 số nguyên nào cạnh nhau trong dãy có giá trị bằng nhau

Ví dụ: **1, 2, 3, 4, 5, 4, 3, 2, 1** là 1 dãy W. độ dài **9**. Tuy nhiên, dãy **1, 2, 3, 4, 5, 4, 3, 2, 2** không là 1 dãy W.

Yêu cầu: Trong các dãy con của dãy số cho trước, tìm dãy W. có độ dài dài nhất.

Input

Dòng 1: số nguyên dương N (N <= 100000), độ dài dãy số.
Dòng 2: N số nguyên dương a_i (a_i <= 10^9).

Output

1 số nguyên dương duy nhất là độ dài dãy W. dài nhất.

Example

Input:
10
1 2 3 4 5 4 3 2 1 10

Output:
9

Input:
19
1 2 3 2 1 2 3 4 3 2 1 5 4 1 2 3 2 2 1

Output:
9

32. Super Number

Cho dãy số nguyên $a_1, a_2, \dots, a_N$ khác nhau đôi một ($N \le 10^5$, $1 \le a_i \le 2^{31}$). Số a_i được gọi là số đặc biệt đối với dãy số trên nếu a_i thuộc ít nhất một dãy con tăng dài nhất của dãy số ban đầu.

Yêu cầu: Tìm mọi số đặc biệt của dãy ban đầu của dãy số nguyên $a_1, a_2, \dots, a_N$.

Dữ liệu: vào từ file văn bản SPECIAL.INP như sau:
Dòng đầu là số T (1<= T <= 10) là số bộ test; T nhóm sau, mỗi nhóm gồm 2 dòng: Dòng thứ nhất ghi số N, dòng thứ hai ghi N số $a_1, a_2, \dots, a_N$.

Kết quả: ghi ra tệp văn bản SPECIAL.OUT gồm T nhóm, mỗi nhóm gồm 2 dòng: Dòng đầu ghi số các số đặc biệt của bộ test tương ứng, dòng thứ hai ghi giá trị các số đặc biệt theo giá trị tăng dần.

Ví dụ:

SPECIAL.INP	SPECIAL.OUT
2 7 1 2 3 7 4 5 6 5 1 4 3 2 5	6 1 2 3 4 5 6 5 1 2 3 4 5

Hướng dẫn: Trong bài toán này, nếu tìm tất cả các dãy tăng dài nhất rồi kiểm tra a_i ($1 \le i \le N$) là không khả thi. Một cách giải khá hay của bài toán như sau:

- Gọi L[i], R[i] lần lượt là độ dài dãy con tăng dài nhất *kết thúc, bắt đầu* tại A_i, và Max là độ dài của dãy con tăng dài nhất. A_i là số đặc biệt của dãy khi và chỉ khi L[i] + R[i] = Max + 1.

33. Chia dãy (http://vn.spoj.pl/problems/QBDIVSEQ)

Dãy số M phần tử B được gọi là dãy con của dãy số A gồm N phần tử nếu tồn tại một mã chuyển C gồm M phần tử thoả mãn B[i]=A[C[i]] với mọi I = 1...M và $1 \le C[1] < C[2] < \dots < C[m] \le N$.
Một cách chia dãy A thành các dãy con "được chấp nhận" nếu các dãy con này là các dãy không giảm và mỗi phần tử của dãy A thuộc đúng một dãy con.
Yêu cầu: Bạn hãy chia dãy con ban đầu thành ít dãy con nhất mà vẫn "được chấp nhận".

Input

Dòng đầu tiên ghi số N là số phần tử của dãy A. ($N \le 10^5$)
N dòng tiếp theo ghi N số tự nhiên là các phần tử của dãy A. ($A_i \le 10^9$)

Output

Ghi một duy nhất là số lượng dãy con ít nhất thỏa mãn.

Example

Input:
4
1
5
4
6

Output:
2

Hướng dẫn: Ta nhận thấy là số dãy con ít nhất chính bằng độ dài của dãy con giảm cực đại.

34. Nested Dolls (http://vn.spoj.pl/problems/MDOLLS)

"Dilworth" có một bộ sưu tập các con búp bê Nga. Búp bê với chiều rộng w1 và chiều cao h1 sẽ nằm trong được con lật đật chiều rộng w2 và chiều cao h2 nếu w1 < w2 và h1 < h2.

Tính số lớp búp bê bao nhau ít nhất mà có thể tạo ra được từ các búp bê ban đầu.

Input

Dòng đầu là số test, $1 \le t \le 20$. Mỗi test bắt đầu là số nguyên m, $1 \le m \le 20000$, số lượng búp bê ban đầu. Dòng tiếp theo là 2m số nguyên w1, h1,w2, h2, ... ,wm, hm, là chiều rộng và chiều cao của con búp bê thứ i, $1 \le wi, hi \le 10000$.

SAMPLE INPUT
4
3
20 30 40 50 30 40
4
20 30 10 10 30 20 40 50
3
10 30 20 20 30 10
4
10 10 20 30 40 50 39 51

Output

Ghi số lớp búp bê bao nhau ít nhất có thể trên một dòng cho từng test.

SAMPLE OUTPUT
1
2
3
2

Hướng dẫn: Sắp xếp các búp bê theo chiều tăng dần w[i]. Bài toán được đưa về bài QBDIVSEQ áp dụng trên dãy h[i]. Chú ý: Xử lý sắp xếp một cách "khéo léo" để có thể AC ;)

35. Wooden Sticks (http://vn.spoj.pl/problems/MSTICK)

Có n đoạn gỗ. Để xử lý chúng cần thời gian để chuẩn bị :

(a) Thời gian chuẩn bị cho đoạn gỗ đầu tiên là 1 phút.
(b) Sau khi xử lý xong đoạn gỗ có chiều dài l và trọng lượng w , không mất thời gian xử lý nếu đoạn gỗ tiếp theo có độ dài l' và trọng lượng w' thỏa mãn l ≤ l' and w ≤ w'. Ngược lại mất 1 phút để chuẩn bị.

Tìm thời gian chuẩn bị ít nhất cho n đoạn gỗ. Ví dụ có 5 đoạn (9 , 4) (2 , 5) (1 , 2) (5 , 3) và (4 , 1) thì thời gian ít nhất là 2 vì có thể xử lý theo thứ tự như sau (4 , 1) (5 , 3) (9 , 4) (1 , 2) (2 , 5) .

Input

Dòng đầu là số lượng test, T. Mỗi test gồm 2 dòng : dòng đầu là số n , 1 <= n <= 5000 , và dòng thứ hai gồm 2n số nguyên dương l1 , w1 , l2 , w2 ,..., ln , wn , <= 10000 , li và wi là độ dài và trọng lượng của đoạn gỗ thứ i.

SAMPLE INPUT
3
5
4 9 5 2 2 1 3 5 1 4
3
2 2 1 1 2 2
3
1 3 2 2 3 1

Output

Ghi ra thời gian ít nhất trên từng dòng.

SAMPLE OUTPUT
2
1
3

36. Con voi (http://vn.spoj.pl/problems/MCONVOI)

Mỗi sáng dậy, voi tập thể dục giảm béo bằng cách nhảy cao. Nó nhảy từ ô (x1, y1) đến ô (x2, y2) với x2 > x1 và y2 > y1. Nó có thể bắt đầu nhảy từ một một vị trí bất kỳ. Tính số bước nhảy tối đa theo quy tắc trên và số cách nhảy khác nhau mà có cùng số bước nhảy tối đa, kết quả là phần dư chia cho 1000000007.

Input

Dòng đầu là số vị trí N (1 ≤ N ≤ 300 000), N dòng tiếp theo là tọa độ các vị trí, 0 <=xi,yi<=10^9. Không có 2 tọa độ trùng nhau.

Output

Dòng 1: số bước nhảy nhiều nhất theo quy tắc trên.
Dòng 2: số cách nhảy , lấy dư theo module 1000000007.

Sample

Input	Input
11 8 6 7 4 5 4 5 1 5 6 6 2 3 2 4 3 4 5 3 5 2 4	6 1 3 2 2 3 1 5 3 4 4 3 5
Output	**Output**
4 3	2 7

37. Đường đi có tổng lớn nhất (http://vn.spoj.pl/problems/QBMAX)

Cho một bảng A kích thước m x n (1 <= m, n <= 100), trên đó ghi các số nguyên a_{ij} ($|a_{ij}|$ <= 100). Một người xuất phát tại ô nào đó của cột 1, cần sang cột n (tại ô nào cũng được).

Quy tắc đi: Từ ô (i, j) chỉ được quyền sang một trong 3 ô (i, j + 1); (i - 1, j + 1); (i + 1, j + 1)

Input

Dòng 1: Ghi hai số m, n là số hàng và số cột của bảng.

M dòng tiếp theo, dòng thứ i ghi đủ n số trên hàng i của bảng theo đúng thứ tự từ trái qua phải

Output

Gồm 1 dòng duy nhất ghi tổng lớn nhất tìm được

Example

Input:
5 7
9 -2 6 2 1 3 4
0 -1 6 7 1 3 3
8 -2 8 2 5 3 2
1 -1 6 2 1 6 1
7 -2 6 2 1 3 7

Output:
41

Hướng dẫn: Đây là một bài QHĐ cơ bản, cần lưu ý một chút về thứ tự tính công thức QHĐ.

38. Roboco

Công ty phát triển phần mềm tự động hóa Roboco vừa cho xuất xưởng một mô hình Robot mới. Đặc điểm của Robot mới này là nó có thể làm việc theo chương trình soạn thảo cho nó. Trong những chương trình như vậy có các lệnh: thực hiện một bước di chuyển về phía Đông, Tây, Nam,

Bắc. Robot thực hiện các lệnh chương trình một cách tuần tự và dừng lại khi gặp dấu hiệu kết thúc chương trình. Các chuyen viên của công ty Roboco muốn xác định xem có bao nhiêu chương trình khác nhau gồm K câu lệnh điều khiển trên trục tọa độ tương ứng với độ dài của một bước di chuyển của Robot.

Dữ liệu vào: từ file ROBOCO.INP chứa 3 số nguyên K, x, y ($0 \le K \le 16$, $|x|, |y| \le 16$) được ghi cách nhau bởi dấu cách.

Kết quả: ghi ra file ROBOCO.OUT số lượng chương trình tìm được.

Ví dụ:

ROBOCO.INP	ROBOCO.OUT
5 2 3	10

Hướng dẫn: Gọi F[i, j, k] là số cách đến ô [i, j] sau k bước. Đây là bài khá đơn giản, các bạn có thể đọc code để hiểu:

```
--------------------------------------------------------------------------------
        a[0, 0] := 1;
        For k := 1 to n do
          Begin
             For i := -26 to 26 do
               For j := -26 to 26 do
                 c[i, j] := a[i-1, j] + a[i, j-1] + a[i, j+1] + a[i+1, j];
             a := c;
          End;
--------------------------------------------------------------------------------
```

39. Đường đi trên lưới (http://vn.spoj.pl/problems/NKPATH)

Cho một lưới ô vuông gồm m dòng và n cột. Các dòng được đánh số từ 1 đến m từ trên xuống dưới, các cột được đánh số từ 1 đến n từ trái qua phải. Ô nằm ở vị trí dòng i và cột j của lưới được gọi là ô (i, j) và khi đó, i được gọi là tọa độ dòng còn j được gọi là tọa độ cột của ô này. Trên ô (i, j) của lưới ghi số nguyên dương a_{ij}, i = 1, 2, …, m; j = 1, 2, …, n. Trên lưới đã cho, từ ô (i, j) ta có thể di chuyển đến ô (p, q) nếu các điều kiện sau đây được thỏa mãn:

- $j < n$; $i \le p$; $j \le q$ và $i + j < p + q$;
- a_{ij} và a_{pq} có ước số chung lớn hơn 1.

Ta gọi một cách di chuyển từ mép trái sang mép phải của lưới là cách di chuyển bắt đầu từ một ô có tọa độ cột bằng 1 qua các ô của lưới theo qui tắc di chuyển đã nêu và kết thúc ở một ô có tọa độ cột bằng n.

Yêu cầu: Tính số cách di chuyển từ mép trái lưới sang mép phải lưới.

Dữ liệu vào

- Dòng đầu tiên ghi 2 số nguyên dương m, n.
- Dòng thứ i trong số m dòng tiếp theo ghi n số nguyên dương a_{i1}, a_{i2}, …, a_{in} là các số trên dòng thứ i của lưới, i = 1, 2, …, m.

Hai số liên tiếp trên cùng một dòng được ghi cách bởi ít nhất một dấu cách.

Kết qủa

Ghi ra 1 số nguyên là phần dư của số lượng cách di chuyển tìm được cho 10^9.

Hạn chế

Trong tất cả các test: $1 < m, n \le 100$; $a_{ij} \le 30000$, i=1,2,…,m;j=1,2,…,n. Có 50% số lượng test với $m, n \le 50$.

Ví dụ

Dữ liệu mẫu
2 2
2 4
6 8

Kết qủa
4

Dữ liệu mẫu
2 2
2 5
6 7

Kết qủa
0

Hướng dẫn: Gọi F[i, j] là số cách tới được ô [i, j]. Ta có F[i, j] := F[i, j] + F[x, y] với ($1 \le x \le i$, $1 \le y \le j$). Kết quả là Sum(F[i, n]) với $1 \le i \le m$.

Phần tiếp theo, chúng ta sẽ làm quen với một số bài tập Quy Hoạch Động trạng thái

Các bài toán Quy Hoạch Động đều dựa trên các trạng thái, tuy nhiên có loại trạng thái dễ phát hiện/ khó phát hiện. Do đó khi nhắc tới QHĐ trạng thái, ta ngầm hiểu nó là loại khó hơn, dạng phức tạp hơn!

40. Tour du lịch của Sherry (http://vn.spoj.pl/problems/LEM3)

Trong kì nghỉ hè năm nay sherry được bố thưởng cho 1 tour du lịch quanh N đất nước tươi đẹp với nhiều thắng cảnh nổi tiếng (vì sherry rất ngoan). Tất nhiên sherry sẽ đi bằng máy bay.
Giá vé máy bay từ đất nước i đến đất nước j là C_{ij} (dĩ nhiên C_{ij} có thể khác C_{ji}). Tuy được bố thưởng cho nhiều tiền để đi du lịch nhưng sherry cũng muốn tìm cho mình 1 hành trình với chi phí rẻ nhất có thể để dành tiền mua quà về tặng mọi người (Các chuyến bay của sherry đều được đảm bảo an toàn tuyệt đối).
Bạn hãy giúp sherry tìm 1 hành trình đi qua tất cả các nước, mỗi nước đúng 1 lần sao cho chi phí là bé nhất nhé.

Input

Dòng 1: N (5 < N < 16)

Dòng thứ i trong N dòng tiếp theo: Gồm N số nguyên, số thứ j là C_{ij} (0 < C_{ij} < 10001)

Output

Gồm 1 dòng duy nhất ghi chi phí bé nhất tìm được

Example

Input:	Output:
6 0 1 2 1 3 4 5 0 3 2 3 4 4 1 0 2 1 2 4 2 5 0 4 3 2 5 3 5 0 2 5 4 3 3 1 0	8

Hướng dẫn: Ở phần duyệt ta cũng đã xét qua bài toán này. Ta sẽ tham khảo 1 cách giải khác bằng QHĐ trạng thái. Gọi F[i, j] là chi phí thấp nhất khi đi đến thành phố i với trạng thái lúc này là j (j là một số nguyên mà dãy bit của nó biểu thị cho việc đến hay chưa đến 1 thành phố, với quy ước 0 là chưa đến 1 là đã đến).

Ta có F[i, j] := Min(F[x, y] + C[x, i], F[i, j]);

Công thức trên được hiểu : thành phố x ở trạng thái y đã được thăm, còn thành phố i chưa được thăm. Từ thành phố x ta đến thành phố i và trạng thái lúc này từ y sẽ được chuyển sang j bởi lúc này đã thăm được thành phố i. Để tiện cho cài đặt thì i ta sẽ lấy bit (i-1) làm trạng thái của i (bởi dãy bit bắt đầu từ 0). Dễ thấy cấu hình cuối cùng là dãy bit 11...11111 . Ta có chi phí nhỏ nhất sẽ là Min(F[i, (1 shl n) -1]) với $1 \le i \le n$.