



ĐẶNG TUẤN THÀNH
(Sưu tầm và biên soạn)

KỸ THUẬT LẬP TRÌNH

Quyển 1



Tháng 01 năm 2021



LỜI NÓI ĐẦU

Nhằm đáp ứng nhu cầu của các học sinh và giáo viên có thêm tư liệu, bài tập dùng cho ôn tập, luyện tập, củng cố và nắm vững kiến thức, kỹ năng cần thiết phục vụ cho học lập trình trong trường phổ thông, tôi xin ra mắt cuốn sách: “KỸ THUẬT LẬP TRÌNH – Quyển 1”.

Nội dung cuốn sách gồm **10** chuyên đề với hơn **200** bài tập:

Chuyên đề 1: Bài toán và Thuật toán

Chuyên đề 2: Cấu trúc điều khiển rẽ nhánh

Chuyên đề 3: Cấu trúc lặp

Chuyên đề 4: Kiểu dữ liệu mảng

Chuyên đề 5: Kiểu dữ liệu xâu

Chuyên đề 6: Bài toán liệt kê

Chuyên đề 7: Quy hoạch động

Chuyên đề 8: Đồ thị

Chuyên đề 9: Tuyển tập đề Codeforce

Chuyên đề 10: Tuyển tập đề thi học sinh giỏi các cấp

Hệ thống kiến thức được trình bày từ dễ đến khó với mục tiêu “*hình thành và phát triển kỹ năng lập trình cho người học*”. Với mỗi chuyên đề có các bài tập được chia thành hai nhóm: Bài tập có lời giải và nhóm Bài tập rèn luyện.

Cuốn sách này cũng là tài liệu bổ ích để dạy và học ngôn ngữ lập trình ở môn Tin học trong trường phổ thông.

Trong thời gian tới, tác giả viết hoàn thiện và ra mắt học liệu “21 giờ học lập trình” với hệ thống kiến thức, kỹ năng hay hơn nữa. Rất mong các bạn đón đọc.

Quyển sách này đã được tác giả biên tập rất tỉ mỉ nhưng không tránh được những thiếu sót, tác giả rất mong được sự đóng góp của bạn đọc vào email: dtthanh.c3ntt@yenbai.edu.vn.

Tác giả

Chuyên đề 1
BÀI TOÁN VÀ THUẬT TOÁN

I. Bài tập và lời giải chuyên đề 1

1. Nêu khái niệm bài toán và khái niệm Thuật toán?

<p><i>Bài toán trong tin học</i> là những việc giao cho máy tính thực hiện. Mỗi bài toán được cấu tạo bởi hai thành phần cơ bản:</p> <p><i>Input</i>: các thông tin đã có.</p> <p><i>Output</i>: các thông tin cần tìm từ input.</p>	<p>Thuật toán là dãy hữu hạn các bước, mỗi bước mô tả chính xác các phép toán hoặc hành động cần thực hiện, để giải quyết một vấn đề.</p> <p>Có hai cách mô tả Thuật toán là liệt kê và sơ đồ khối.</p>
--	---

2. Nêu các bước giải bài toán trên máy tính?

<p>Bước 1. Xác định bài toán: xác định input, output và quan hệ giữa chúng.</p> <p>Bước 2. Lựa chọn Thuật toán và tổ chức dữ liệu.</p> <p>Bước 3. Viết chương trình: Sử dụng ngôn ngữ lập trình để viết chương trình.</p> <p>Bước 4. Hiệu chỉnh: Kiểm tra để phát hiện và sửa lỗi.</p> <p>Bước 5. Viết tài liệu: Mô tả toàn bộ bài toán, Thuật toán, tổ chức chương trình, kết quả chạy thử và hướng dẫn sử dụng.</p>

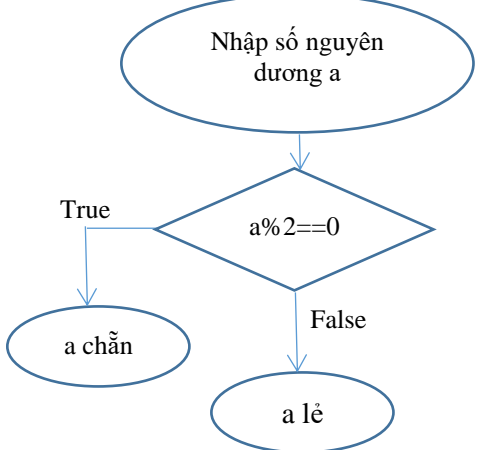
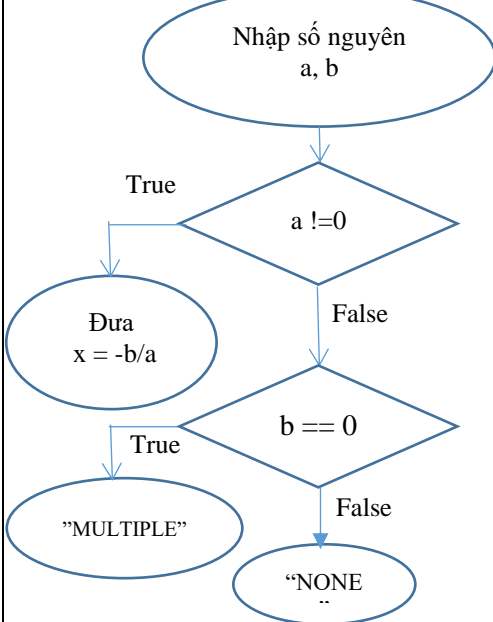
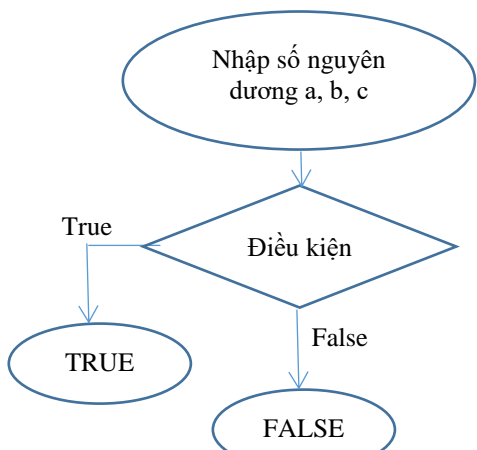
3. Nêu khái niệm chương trình dịch, thông dịch và biên dịch?

<p>Chương trình dịch là chương trình dùng để dịch chương trình được viết bởi ngôn ngữ lập trình bậc cao thành ngôn ngữ máy để máy tính có thể trực tiếp thực hiện được.</p> <p>Có hai loại chương trình dịch đó là thông dịch và biên dịch.</p>

4. Trình bày Thuật toán giải các bài toán

- Nhập a, b nguyên dương. Hãy tính diện tích hình chữ nhật có cạnh a, b.
- Nhập a, b, c nguyên dương. Hãy tính tổng diện tích các mặt của hình hộp chữ nhật cạnh a, b, c.
- Nhập số nguyên dương a. Kiểm tra a là số chẵn hay số lẻ.
- Nhập số nguyên a, b. Giải và biện luận nghiệm của phương trình $ax + b = 0$. Nếu phương trình vô nghiệm thông báo NONE. Nếu có một nghiệm thông báo dạng $X = -b/a$. Nếu có vô số nghiệm thông báo MULTIPLE.
- Nhập số nguyên dương a, b và c. Hãy kiểm tra a, b và c có là ba cạnh của tam giác. Nếu có thông báo TRUE, ngược lại thông báo FALSE.
- Nhập số nguyên dương a và b. Tìm ước chung lớn nhất của a và b.

5. Lời giải bài 4.

<p>a)</p> <p>Bước 1: Nhập số nguyên dương a, b.</p> <p>Bước 2: Tính $S \leftarrow a + b$;</p> <p>Bước 3: Đưa S ra.</p>	<p>b)</p> <p>Bước 1: Nhập số nguyên dương a, b, c.</p> <p>Bước 2: Tính $S \leftarrow 2(ab+ac+bc)$</p> <p>Bước 3: Đưa S ra.</p>
<p>c)</p> <p>Bước 1: Nhập số nguyên dương a.</p> <p>Bước 2: Kiểm tra nếu $(a \% 2 == 0)$ thì Thông báo “a chẵn”;</p> <p>Ngược lại Thông báo “a lẻ”;</p>	 <pre> graph TD A([Nhập số nguyên dương a]) --> B{a%2==0} B -- True --> C([a chẵn]) B -- False --> D([a lẻ]) </pre>
<p>d)</p> <p>Bước 1: Nhập số nguyên a và b.</p> <p>Bước 2: Kiểm tra nếu $(a \neq 0)$ thì: + Tính $x = -b/a$ + Đưa x ra.</p> <p>Ngược lại, sang bước 3</p> <p>Bước 3: Kiểm tra nếu $(b == 0)$ thì thông báo MULTIPLE</p> <p>Ngược lại thông báo NONE.</p>	 <pre> graph TD A([Nhập số nguyên a, b]) --> B{a != 0} B -- True --> C([Đưa x = -b/a]) B -- False --> D{b == 0} D -- True --> E([MULTIPLE]) D -- False --> F([NONE ..]) </pre>
<p>e)</p> <p>Bước 1: Nhập số nguyên dương a, b, c.</p> <p>Bước 2: Kiểm tra nếu $(a+b > c \ \&\& \ b+c > a \ \&\& \ c+a > b)$ thì Thông báo “TRUE”;</p> <p>Ngược lại Thông báo “FALSE”;</p>	 <pre> graph TD A([Nhập số nguyên dương a, b, c]) --> B{Điều kiện} B -- True --> C([TRUE]) B -- False --> D([FALSE]) </pre>

f)

Bước 1: Nhập số nguyên dương a, b .

Bước 2: Kiểm tra nếu $(a == b)$ thì sang bước 4.

Ngược lại sang bước 3.

Bước 3: Kiểm tra nếu $a > b$ thì $a \leftarrow a - b$, quay lại bước 2.

Ngược lại, $b \leftarrow b - a$, quay lại bước 2

II. Bài tập rèn luyện 1

Trình bày **Thuật toán** giải các bài toán sau:

- a) Nhập số nguyên dương X . Tính căn bậc 3 của X . Kết quả lấy chính xác đến 2 chữ số thập phân.
- b) Nhập số nguyên a và số nguyên b . Tìm bội chung nhỏ nhất của a và b .
- c) Nhập số nguyên N . Tìm số lượng ước của N .
- d) Nhập số nguyên N . Hãy phân tích N thành các thừa số nguyên tố.
- e) Nhập số nguyên a và số nguyên b . Tìm các ước chung của a và b .

Chuyên đề 2
CÁU TRÚC ĐIỀU KHIỂN Rẽ NHÁNH

I. Bài tập và lời giải về cấu trúc rẽ nhánh

1. Cấu trúc rẽ nhánh

Dạng 1: Nếu (điều kiện) thì (câu lệnh);

If (điều kiện) then (câu lệnh);

Dạng 2: Nếu (điều kiện) thì (Câu lệnh 1)

Ngược lại (Câu lệnh 2);

If (điều kiện) then (câu lệnh 1)

Else (câu lệnh 2);

2. Viết chương trình giải các bài toán

a) Chẵn lẻ

Nhập số nguyên X. Kiểm tra X là số chẵn hay số lẻ.

b) Số chính phương

Nhập số nguyên Y. Nếu Y là số chính phương thông báo 1, ngược lại thông báo 0.

c) Tìm max2

Nhập số nguyên a và b. Tìm giá trị lớn nhất của a, b.

d) Tìm max3

Nhập số nguyên a, b, c. Tìm giá trị lớn nhất của a, b và c.

e) Tìm max4

Nhập số nguyên a, b, c, d. Tìm giá trị lớn nhất của a, b, c, d.

f) Ghép số

Nhập số nguyên a, b có giá trị trong đoạn [0,9]. Jame ghép nó thành số có hai chữ số. Tìm số lớn nhất mà Jame có thể có được.

g) Phương trình bậc nhất

Nhập a, b nguyên. Hãy giải và biện luận phương trình $ax + b = 0$.

- Nếu vô nghiệm thông báo None.
- Nếu vô số nghiệm thông báo Multiple.
- Nếu có nghiệm, đưa ra dạng $x = -b / a$;

h) Phương trình bậc hai

Nhập a, b, c nguyên. Giải và biện luận phương trình $ax^2 + bx + c = 0$;

i) Hệ phương trình

Cho hệ phương trình $\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$. Hãy giải và biện luận hệ phương trình trên biết $a_1, a_2, b_1, b_2, c_1, c_2$ nguyên được nhập từ bàn phím.

j) Tìm X

Nhập số thực X. Tìm số nguyên A lớn nhất và B nhỏ nhất sao cho $A < X < B$. Ví dụ: $x=3.14 \rightarrow A = 3; B = 4$.

3. Lời giải bài tập mục 2

a) Chẵn lẻ

```

Program Chanle;
Var X: byte;
Begin
    Write('Nhap X ='); Readln(X);
    If (X mod 2 = 0) write('X chan')
    Else write('X le');
    Readln;
End.

```

b) Số chính phương

```

Program Chinhphuong;
User crt;
Var y: byte;
Begin
    Write('Nhap Y ='); Readln(y);
    If (trunc(sqrt(Y)) = sqrt (Y) ) write(1)
    Else write(0);
    Readln;
End.

```

c) Tìm max2

```

Program Timmax2;
Var a, b, max2: byte;
Begin
    Write('Nhap a ='); Readln(a);
    Write('Nhap b ='); Readln(b);
    Max2 := a;
    If max2 < b then max2 := b;
    Write('GTLN=', max2);
    Readln;
End.

```

d) Tìm max3

```

Program Timmax3;
Var a, b, c, max3: byte;
Begin
    Write('Nhap a ='); Readln(a);
    Write('Nhap b ='); Readln(b);
    Write('Nhap c ='); Readln(c);
    Max3 := a;
    If max3 < b then max3 := b;
    If max3 < c then max3 := c;

```

```

    Write('GTLN=', max3);
    Readln;
End.
e) Tìm max4
Program Timmax4;
Var a, b, c, d, max4: byte;
Begin
    Write('Nhap a ='); Readln(a);
    Write('Nhap b ='); Readln(b);
    Write('Nhap c ='); Readln(c);
    Write('Nhap d ='); Readln(d);
    Max4 := a;
    If max4 < b then max4 := b;
    If max4 < c then max4 := c;
    If max4 < d then max4 := d;
    Write('GTLN=', max4);
    Readln;
End.

```

```

f) Ghép số
Program Ghepso;
Var a, b, max1, max2, max: byte;
Begin
    Write('Nhap a ='); Readln(a);
    Write('Nhap b ='); Readln(b);
    Max1 := 10*a + b;
    Max2 := 10*b + a;
    Max := Max1;
    If max < max2 then max := max2;
    Write('Ket qua =', max);
    Readln;
End.

```

g) ***Phương trình bậc nhất***

h) ***Phương trình bậc hai***

i) ***Hệ phương trình***

- Gợi ý: Dùng định thức cấp 2, tính D, Dx, DY.

j) ***Tìm X***

```

Program TimX;

```

```

User crt;

```

```

Var X: Real;

```

```

    A, B: byte;

```



```

Begin
    Write('Nhap X ='); Readln(X);
    A := Trunc(X);
    B := A + 1;
    Writeln ('A = ', A);
    write('B = ', B);
    Readln;
End.

```

4. Bài tập rèn luyện 2

a) **Tổng các chữ số**

Nhập số nguyên dương a là số có hai chữ số. Tính tổng các chữ số của a.

Ví dụ: a = 23 → tong = 5.

b) **Tích các chữ số**

Nhập số nguyên dương N là số có 3 chữ số. Tìm tích các chữ số của N.

Ví dụ: N = 234 → tích = 24.

c) **Tìm S**

Nhập N nguyên dương, $N \leq 2^{32}$. Tính giá trị của $S = \frac{n(n+1)(n+2)}{3}$

d) **Đổi thời gian**

Nhập số thời gian là T (giây). Người ta đổi T thành a giờ, b phút, c giây.

Hãy tìm a, b, c.

e) **Đổi tiền 1**

Jame có X đồng. Biết rằng tờ tiền có mệnh giá là 100, 50, 20, 10, 5, 2, 1. Hỏi số lượng tờ tiền ít nhất mà Jame cầm trên tay là bao nhiêu?

Ví dụ: X = 165 → 4 tờ (1 tờ 100, 1 tờ 50, 1 tờ 10, 1 tờ 5)

f) **Tìm P**

Nhập a, b nguyên. Tính $P = \frac{a^2 + \sqrt{a+b} + b^3}{a+b}$. Kết quả lấy chính xác đến 2 chữ số thập phân.

g) **Khoảng cách**

Cho điểm A(x1,y1) điểm B (x2,y2) và điểm M(x3, y3). Tính khoảng cách từ điểm M đến đường thẳng đi qua 2 điểm A, B.

h) **Tam giác 1**

Nhập toạ độ các điểm A(x1,y1), B(x2,y2), C(x3,y3) là ba đỉnh của một tam giác. Hãy tính độ dài các cạnh và diện tích tam giác ABC.

i) **Hình chữ nhật 1**

Cho hình chữ nhật có góc trái trên là A(x1, y1), góc phải dưới B(x2, y2). Gọi a, b là chiều dài và chiều rộng của hình chữ nhật. Hãy tìm a, b.

j) **Hình chữ nhật 2**

Cho hình chữ nhật có cạnh a, b và được chia thành lưới ô vuông có cạnh 1 cm. Hãy đếm số giao điểm của các lưới ô vuông đơn vị bên trong hình chữ nhật.

k) **Hình chữ nhật 3**

Cho hình chữ nhật có cạnh a, b và được chia thành lưới ô vuông có cạnh 1 cm. Hãy đếm số cạnh của các lưới ô vuông đơn vị bên trong hình chữ nhật.

l) Hình chữ nhật 4

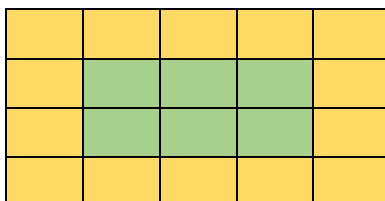
Cho hình chữ nhật có kích thước a, b . Gọi X là số ô đường viền, Y là số ô trong lõi. Hãy tìm X, Y .

Ví dụ: $a = 4 ; b = 5 \rightarrow X = 14; Y = 6$.

m) Hình chữ nhật 5

Cho hình chữ nhật có kích thước $a \times b$, được chia thành lưới ô vuông đơn vị 1×1 . Gọi X là số ô ở đường viền, Y là số ô ở trong lõi HCN.

Nhập X, Y nguyên từ bàn phím. Hãy tìm a, b .



Ví dụ: $X = 14; Y = 6 \rightarrow 4 \ 5$

n) Tìm X, Y

Nhập số nguyên dương X, Y .

Nếu X chẵn thì giảm X đi một nửa và tăng Y gấp đôi.

Ngược lại, thì tăng X một đơn vị và tăng Y thêm một lượng là X .

Đưa X, Y ra.

o) Tìm Q

Cho số nguyên dương P có 3 chữ số. Tìm Q là số lớn nhất có 1 chữ số trong P .

Ví dụ: $P = 123 \rightarrow Q = 3$

p) Xóa số

Cho số nguyên dương N là số có 3 chữ số. Jame thực hiện xóa đi 1 số trong 3 chữ số của N để thu được số N mới là số có 2 chữ số. Hãy tìm N lớn nhất có thể.

Ví dụ: $N = 123 \rightarrow N = 23; N = 321 \rightarrow M = 32$.

q) Đổi tiền 2

Jame có X đồng muốn đổi tiền gồm các tờ mệnh giá 50, 20 và 10. Hỏi Jame có thể đổi tiền được không? Nếu có ghi ra số tờ tiền ít nhất Jame có được, nếu không đổi được thì ghi ra -1.

r) Số trung bình cộng

Jame ghi lên bảng 4 số nguyên bất kì. Sau đó Jame muốn biết trong 4 số trên, có số nào là số trung bình cộng hay không? Nếu có ghi ra 1 ngược lại ghi ra -1.

Biết số trung bình cộng bằng tổng 4 số đó chia cho 4.

Ví dụ: $1 \ 2 \ 3 \ 6 \rightarrow 1; \ 1 \ 2 \ 4 \ 6 \rightarrow -1$.

s) Giải phương trình

Nhập a, b, c, d, m nguyên dương. Giải và biện luận phương trình: $\frac{ax+b}{cx+d} = m$

- Nếu phương trình Vô nghiệm ghi ra 'Multiple'.
- Nếu phương trình Vô nghiệm ghi ra 'None'.

- Nếu phương trình có nghiệm ghi ra dạng $x = p / q$. Với P, Q nguyên tố cùng nhau.

t) Tam giác

Cho số N nguyên dương là số có 3 chữ số. Jame muốn biết 3 chữ số của số N có là độ dài ba cạnh của một tam giác vuông hay không?

Nếu có ghi ra diện tích của tam giác đó. Nếu không ghi ra -1.

Ví dụ: $N = 345 \rightarrow S = 6$;

$N = 145 \rightarrow -1$.

u) Điểm và đường tròn 2

Cho điểm M (x1, y1) và đường tròn tâm I(x0, y0) bán kính R. Hãy xét vị trí tương đối của điểm M với đường tròn. Nếu M nằm trong hoặc trên đường tròn ghi ra 1, ngược lại ghi ra 0.

v) Vị trí tương đối của hai đường tròn

Cho đường tròn tâm I1(x1, y1) bán kính R1, đường tròn tâm I2(x2, y2) bán kính R2. Xét vị trí tương đối của hai đường tròn này.

- Nếu hai đường tròn cắt nhau tại hai điểm ghi ra 2.
- Nếu hai đường tròn tiếp xúc nhau, ghi ra 1.
- Hai đường tròn không cắt nhau ghi ra 0.

w) Giải phương trình

Giải và biện luận phương trình $\frac{ax^2+bx+c}{dx+e} = m$

Với a, b, c, d, e, m nguyên dương nhập từ bàn phím ($a \neq 0$, d và e không đồng thời bằng 0).

- Nếu phương trình có nghiệm thì đưa các nghiệm ra dạng $x = p/q$ với p, q nguyên tố cùng nhau.
- Nếu phương trình vô nghiệm thì đưa ra 'None'.

x) Số đẹp X

Một số là số đẹp nếu không có hai chữ số cạnh nhau mà giống hệt nhau.

Cho một số nguyên dương C có n chữ số, hãy đếm xem có bao nhiêu số nguyên dương A và B sao cho:

- A và B là những số nguyên dương có N chữ số, không bắt đầu bằng số 0.
- $A + B = C$
- A, B là những số đẹp.

Yêu cầu: Đếm xem có bao nhiêu cặp số (A, B) thỏa mãn đề bài. Vì kết quả lớn nhất, chỉ ghi ra kết quả chia dư cho $10^9 + 7$.

Dữ liệu vào: SodepX.inp gồm số nguyên dương C ($C < 10^6$)

Kết quả: Ghi ra số cặp số (A, B) tìm được

Ví dụ: $C=23 \rightarrow kq = 2$ cặp. (10+13 và 13+10)

y) Hình chữ nhật

Cho điểm M(x1, y1) và hình chữ nhật có góc trái trên (x2, y2) góc phải dưới (x3, y3).

Kiểm tra M nằm trong hay ngoài hình chữ nhật. Nếu nằm trong và trên cạnh thì ghi ra 1, còn lại nằm ngoài ghi ra 0.

Ví dụ:

3 0; 0 2; 2 0; $\rightarrow 1$

Chuyên đề 3
CẤU TRÚC ĐIỀU KHIỂN LẶP

I. Bài tập và lời giải về cấu trúc lặp For

1. Cấu trúc lặp For

Dạng 1: **For** biến chạy := Giá trị đầu **To** Giá trị cuối **Do**

Câu lệnh;

Dạng 2: **For** biến chạy := Giá trị cuối **downTo** Giá trị đầu **Do**

Câu lệnh;

2. Viết chương trình giải các bài toán

a) Liệt kê 1

Liệt kê các số nhỏ hơn hoặc bằng 100.

b) Liệt kê 2

Nhập N. Liệt kê các số chia hết cho 5 nhỏ hơn hoặc bằng N.

c) Chia hết

Nhập N. Liệt kê nhỏ hơn hoặc bằng N mà chia hết cho 3 và không chia hết cho 5.

d) Tính S1

Nhập n. $s1 = 1 + 2 + 3 + \dots + n$

e) Tính S2

Nhập n. $s2 = 2019 + 2 + 4 + 6 + \dots + 2n$

f) Tính S3

Nhập n. $s3 = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$

g) Tính S4

Nhập n. $s4 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} \dots + (-1)^{n+1} \frac{1}{n}$

h) Tính S5

$s = e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$

i) Tìm ước

Nhập N. Liệt kê các ước nguyên dương của N.

Ví dụ: N = 10; Các ước 1, 2, 5, 10.

j) Tính tổng ước

Nhập N. Hãy tính tổng các ước nguyên dương của N.

Ví dụ: N = 12 \rightarrow tong = 1 + 2 + 3 + 4 + 6 + 12 = 28.

k) Số nguyên tố 1

Số nguyên tố là số chỉ chia hết cho 1 và chính nó. Số 1 không là số nguyên tố.

Nhập N. Kiểm tra N có là số nguyên tố. Nếu có ghi ra 1 ngược lại ghi ra 0.

3. Lời giải bài tập mục 2

a) Liệt kê 1

Program Lietke1;

```

Var i: byte;
Begin
    For i := 1 to 100 do write(i, ' ');
    Readln;
End.

```

b) Liệt kê 2

```

Program Lietke2;
Var i, N: byte;
Begin
    Write('Nhap N='); Readln(N);
    For i := 1 to N do
        If i mod 5 = 0 then write(i, ' ');
    Readln;
End.

```

c) Chia hết

```

Program Lietke3;
Var i, N: byte;
Begin
    Write('Nhap N='); Readln(N);
    For i := 1 to N do
        If (i mod 3 = 0) and (i mod 5 <> 0) then write(i, ' ');
    Readln;
End.

```

d) Tính S1

```

Program TinhS1;
Var i, N: byte;
    S: integer;
Begin
    Write('Nhap N='); Readln(N);
    S := 0;
    For i := 1 to N do
        S := S + i;
    Write('Tong S = ', S);
    Readln;
End.

```

e) Tính S2

```

Program TinhS2;
Var i, N: byte;
    S: integer;
Begin

```

```

Write('Nhap N='); Readln(N);
S := 2019;
For i := 1 to N do
    S := S + 2*i;
Write('Tong S = ', S);
Readln;

```

End.

*f) **Tính S3***

```

Program TinhS3;
Var i, N: byte;
    S:Real;
Begin
    Write('Nhap N='); Readln(N);
    S := 0;
    For i := 1 to N do
        S := S + 1/i;
    Write('Tong S = ', S:6:2);
    Readln;

```

End.

*g) **Tính S4***

```

Program TinhS4;
Var i, N: byte;
    S:Real;
Begin
    Write('Nhap N='); Readln(N);
    S := 0;
    For i := 1 to N do
        If i mod 2 = 0 then S := S + 1/i;
        Else S := S - 1/i;
    Write('Tong S = ', S:6:2);
    Readln;

```

End.

*h) **Tính S5***

```

Program TinhS5;
Var i, N: byte;
    S:Real;
    Tuson, mauso: integer;
Begin
    Write('Nhap N='); Readln(N);
    Write('Nhap X='); Readln(X);

```

```

Tuso := x;
Mauso := 1;
S := 0;
For i := 1 to N do
Begin
    Tuso := tuso * x;
    Mauso := mauso * i;
    S := S + tuso/mauso;
End;
Write('Tong S = ', S:6:2);
Readln;

```

End.

i) Tìm ước

```

Program Timuoc;
Var i, N: byte;
Begin
    Write('Nhap N='); Readln(N);
    For i := 1 to N do
        If N mod i = 0 then write(i, ' ');
    Readln;

```

End.

j) Tính tổng ước

```

Program Tonguoc;
Var i, N: byte;
    Res : integer;
Begin
    Write('Nhap N='); Readln(N);
    Res := 0;
    For i := 1 to N do
        If N mod i = 0 then Res := Res + i;
    Writeln('Tong cac uoc =', Res);
    Readln;

```

End.

k) Số nguyên tố 1

```

Program Songuyento1;
User Crt;
Var i, N : byte;
    Res : integer;
Begin
    Write('Nhap N='); Readln(N);

```

```

Res := 0;
For i := 2 to trunc(sqrt(N)) do
    If N mod i = 0 then
        Begin
            Res := 1;
            Break;
        End;
If N = 1 then Res := 1;
If Res = 0 then Write('N nguyên tố')
Else Write('N không nguyên tố');
Readln;
End.

```

4. Bài tập rèn luyện 3

a) *Liệt kê số nguyên tố*

Liệt kê các số nguyên tố nhỏ hơn hoặc bằng N.

Ví dụ: $N = 12 \rightarrow 2\ 3\ 5\ 7\ 11$

b) *Nguyên tố 2*

“Những số nguyên tố khi chia cho 4 dư 1 thì luôn biểu diễn được dưới dạng tổng của 2 số chính phương”. Kiểm tra N có nguyên tố hay không? Nếu N nguyên tố, kiểm tra xem có thể phân tích N thành tổng của hai số chính phương a và b hay không? Nếu được hãy tìm a và b.

VD: $N = 16$ thông báo “N không là số nguyên tố”

$N = 17$ thông báo “N bằng tổng của hai số chính phương: 1 16”

c) *Tìm số 1*

Tìm một số có 4 chữ số vừa là số chính phương, vừa là số lập phương. Tức là N có thể phân tích dạng $N = x^2 = y^3$, với x, y nguyên.

d) *Tìm số 2*

Tìm tất cả các chữ số có ba chữ số \overline{abc} sao cho tổng các lập phương của các chữ số thì bằng chính số đó ($\overline{abc} = a^3 + b^3 + c^3$).

e) *Số hoàn thiện*

Số H được gọi là *số hoàn thiện* nếu tổng tất cả các ước của H gấp đôi H.

Ví dụ: Số 6 là số hoàn thiện, 6 chia hết cho 1, 2, 3, 6 và tổng ước = 12 = 2x6.

a) Kiểm tra H có là số hoàn thiện? Nếu có ghi ra 1 ngược lại ghi ra 0.

b) Tìm các số hoàn thiện nhỏ hơn hoặc bằng N.

c) Tìm các số hoàn thiện nằm trên đoạn [M, N].

f) *Phân tích số*

Nhập số nguyên dương N. Đếm số cặp (a, b) đôi một khác nhau để: $a + b = N$.

g) *Tính P*

Tính $P = a^x$ với x, a nguyên dương. Vì P có thể rất lớn, nên ta chỉ cần lưu kết quả chia dư cho 10^9+7 .

h) Số nguyên tố 2

- Tìm các số nguyên tố nằm trên đoạn $[M, N]$.
- Nhập số nguyên N . Hãy tìm số nguyên tố bé nhất lớn hơn N .

i) Số Fibonacci

Biết $F_1 = 1, F_2 = 1, F[N] = F[N-1] + F[N-2]$

- Tìm số Fibonacci thứ N ($N \leq 40$).
- Tìm các số Fibonacci nhỏ hơn hoặc bằng N .
- Tìm các số Fibonacci nằm trên đoạn $[M, N]$.
- Tìm các số vừa là số Fibonacci, vừa là số nguyên tố nằm trên đoạn $[M, N]$.

j) Số chính phương

Một số được gọi là *số chính phương* nếu $\sqrt{N} = [\sqrt{N}]$ (Căn bậc hai của N = phần nguyên của căn bậc 2 của N , hay $N = a^2$, với a nguyên)

- Kiểm tra N có là số chính phương? Nếu có ghi ra 1 ngược lại ghi ra 0.
- Tìm tất cả các số chính phương dạng $\overline{22ab}$.

II. Bài tập và lời giải về cấu trúc lặp While

1. Cấu trúc lặp While

While (điều kiện) do

Câu lệnh;

2. Viết chương trình giải các bài toán

a) Tính tổng 1

Tìm n nguyên dương nhỏ nhất sao cho $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \geq a$.

Với $a > 1$ được nhập từ bàn phím.

b) Tính P

Tính $Pi/4$ theo công thức:
$$p = \frac{Pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \dots + (-1)^{n+1} \cdot \frac{1}{2n-1} + \dots$$

cho đến khi $|(-1)^{n+1} \cdot \frac{1}{2n-1}| < 0.001$

c) Tính tổng 2

Viết chương trình tính:
$$S = \frac{1}{1!} + \frac{2}{2!} + \dots + \frac{n}{n!} + \dots$$

Sao cho phần tử cuối cùng của dãy bé thua một số a được nhập từ bàn phím với $0 < a < 1$.

d) Căn bậc 3

Tính $P = \sqrt[3]{X}$. Đưa ra kết quả lấy chính xác đến 3 chữ số thập phân.

Ví dụ: $\text{canbac3}(27) = 3.000$

e) Tìm ước chung lớn nhất

Tìm ước chung lớn nhất của số nguyên dương a và b .

f) Tìm bội chung nhỏ nhất

Tìm bội chung nhỏ nhất của số nguyên dương a và b .

g) Phân tích thừa số nguyên tố

Nhập N nguyên dương. Hãy phân tích N thành các thừa số nguyên tố.

Ví dụ: $N = 100 \rightarrow 2 \ 2 \ 5 \ 5$

h) Tìm số nguyên tố

Tìm số nguyên tố gần số nguyên N nhất. Nếu có nhiều kết quả thì đưa ra kết quả nhỏ nhất.

3. Lời giải bài tập mục 2

a) Tính tổng 1

```
Program TinhTong1;
Var
    N: byte;
    S, a: Real;
Begin
    Write('Nhap a = '); Readln(a);
    N := 0;
    S := 0;
    While S < a do
        Begin
            N := N + 1;
            S := S + 1/N;
        End;
    Write('N = ', N);
    Readln;
End.
```

b) Tính P

```
Program TinhP;
Var
    N: byte;
    P: Real;
Begin
    N := 1;
    P := 0;
    While 1/(2*N - 1) > 0.001 do
        Begin
            N := N + 1;
            If N mod 2 = 0 then P := P - 1/(2*N-1)
            Else P := P + 1/(2*N-1);
        End;
    Write('P = ', P:6:3);
    Readln;
End.
```

c) **Tính tổng 2**

```
Program TinhTong2;  
Var  
    N: byte;  
    Mausoi: integer;  
    S, a: Real;  
Begin  
    Write('Nhap a = '); Readln(a);  
    N := 1;  
    S := 1;  
    Mausoi := 1;  
    While N/mausoi >= a do  
        Begin  
            N := N + 1;  
            Mausoi := Mausoi*N;  
            S := S + N/Mausoi;  
        End;  
    Write('S = ', S:6:3);  
    Readln;  
End.
```

d) **Căn bậc 3**

```
Program Canbac3;  
Uses math;  
Var  
    X: Real;  
    Dau, cuoi, giua: Real;  
Begin  
    Write('Nhap X = '); Readln(X);  
    Dau := 1;  
    Cuoi := sqrt(X);  
    While (cuoi - dau >= 0.001) do  
        Begin  
            Giua := (dau + cuoi) / 2;  
            If (giua*giua*giua > X) then cuoi := giua  
            Else dau:=giua;  
        End;  
    Write('Can bac ba cua X = ', giua:6:3);  
    Readln;  
End.
```

e) Tìm ước chung lớn nhất

```
Program Timucln;  
Var a, b, du: byte;  
Begin  
    Write('Nhap a= '); Readln(a);  
    Write('Nhap b= '); Readln(b);  
    While b > 0 do  
        Begin  
            Du := a mod b;  
            a := b;  
            b := r;  
        End;  
    Writeln('Res = ', a);  
    Readln;  
End.
```

f) Tìm bội chung nhỏ nhất

```
Program TimBCND;  
Var a, b, du, res: byte;  
    aluu, bluu: byte;  
Begin  
    Write('Nhap a= '); Readln(a);  
    Write('Nhap b= '); Readln(b);  
    aluu:= a;  
    bluu:= b;  
    While b > 0 do  
        Begin  
            du := a mod b;  
            a := b;  
            b := r;  
        End;  
    Res := (aluu*bluu) div a;  
    Writeln('Res = ', Res);  
    Readln;  
End.
```

g) Phân tích thừa số nguyên tố

```
Program Phantichthuasonguyento;  
Var N, i: integer;  
Begin  
    Write('Nhap N= '); Readln(N);  
    i := 2;
```

```

While N >=1 do
Begin
    While (N mod i = 0) do
        Begin
            Write(i, ' ');
            N := N div i;
        End;
        i := i + 1;
    End;
    Readln;
End.

```

h) Tìm số nguyên tố

```

Program TimSNT;
Uses Crt;
Var N, Res: byte;
Function nguyento( a: byte): boolean;
Var p: byte;
Begin
    P := 0;
    If a < 2 then p := 1;
    If (a > 3) then
    Begin
        For i:=2 to trunc(sqrt(a)) do
            If a mod i = 0 then
                Begin
                    P:=1;
                    Break;
                End;
        If p = 0 then nguyento := true
        Else nguyento:= false;
    End;
Begin
    Write('Nhap N= '); Readln(N);
    Res := N + 1;
    While nguyento(Res) = false do
        Res := Res + 1;
    Writeln('Res = ', Res);
    Readln;
End.

```

4. Bài tập rèn luyện 4

i) **Số Mersen**

Số P là *số Mersen* nếu P nguyên tố và P có thể phân tích thành $P = 2^r - 1$, mà r cũng là số nguyên tố.

- Kiểm tra P có là số Mersen hay không? Nếu có ghi ra 1 ngược lại ghi 0.
- Tìm các số Mersen nhỏ hơn hoặc bằng N .
- Tìm các số Mersen nằm trên đoạn $[M, N]$.

j) **Ốc sên**

Con ốc sên đang ở gốc của một cái cây cao v mét tính từ gốc. Ốc sên muốn bò lên ngọn cây để ăn những lá non trên đó. Ban ngày ốc sên bò được a mét lên trên, nhưng ban đêm, khi ngủ nó bị trôi xuống dưới b mét.

Yêu cầu: Cho các số nguyên v , a và b ($1 \leq b < a \leq v \leq 10^9$). Hãy xác định số ngày cần thiết để ốc sên lên tới ngọn cây.

Dữ liệu: Vào từ file văn bản SNAIL.INP gồm một dòng chứa 3 số nguyên a , b và v .

Kết quả: Đưa ra file văn bản SNAIL.OUT một số nguyên – kết quả tìm được.

Ví dụ:

SNAIL.INP
2 1 5

SNAIL.OUT
3



k) **Tìm số**

Nhập M ($M \leq 10^9$). Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M . Nếu không tồn tại K , đưa ra -1.

l) **Số đẹp 4**

Số đẹp là số có tổng các chữ số chia hết cho 9. Như vậy, các số 9, 18, 27, 36,... là các số đẹp. Liệt kê các số đẹp nhỏ hơn hoặc bằng N .

m) **Cắt số**

Cho số nguyên dương N . Người ta có thể cắt các chữ số của N từ phải sang trái để thu được số N mới.

Hãy tìm số N mới là số nguyên tố lớn nhất. Biết phải cắt ít nhất 1 số. Nếu không có ghi ra -1.
Ví dụ: $N = 1324 \rightarrow N = 13$.

n) **Số Palidrom**

Số Palidrom là số đối xứng, nghĩa là đọc từ trái sang phải hay từ phải sang trái ta đều được một số. Ví dụ: Các số palidrom 11, 121, 1331, 2222, 1556551,...

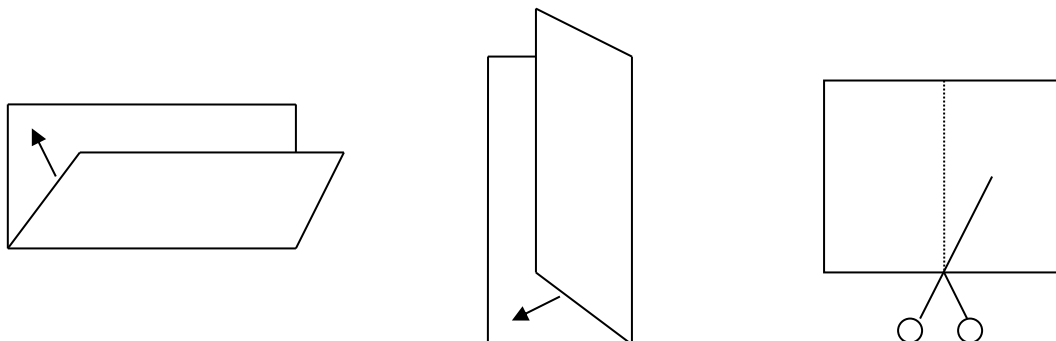
Nhập số nguyên N . Liệt kê các số Palidrom nhỏ hơn hoặc bằng N .

o) **Căn bậc hai**

Tìm căn bậc hai của N . Không dùng hàm có sẵn. Đưa kết quả ra với độ chính xác đến 3 chữ số thập phân.

p) **Gấp giấy**

Người ta lấy một tờ giấy hình vuông, gấp đôi lại để mép dưới đè lên mép trên (hình 1), gấp đôi lại một lần nữa, để mép phải đè lên mép trái (hình 2). Hai phép gấp nêu trên được thực hiện N lần ($0 \leq N \leq 30$). Sau đó người ta dùng kéo cắt dọc theo một đường thẳng ở giữa từ dưới lên trên (hình 3).



Hình 1

Hình 2

Hình 3

Yêu cầu : Xác định xem ta sẽ nhận được bao nhiêu mảnh giấy rời khi mở tờ giấy.

Dữ liệu vào : Vào từ bàn phím một số nguyên N.

Dữ liệu ra : Ghi ra màn hình một số nguyên kết quả.

Ví dụ :

N
2

Số mảnh
5

q) Vòng tay (ntucoder)

Người con gái miền núi rất thích đeo vòng tay, bộ vòng tay thường có 7 chiếc dùng để đếm thời gian. Cứ sau 01 ngày họ tháo chiếc vòng ở tay này đeo qua tay khác và sẽ di chuyển ngược lại nếu như hết 01 tuần. Người con gái hẹn gặp lại người yêu sau n ngày. Hãy cho biết ở thời điểm đó, số lượng vòng trên mỗi tay sẽ bằng bao nhiêu. Giả sử ban đầu cô gái đeo vòng tay bên trái.

Dữ liệu vào: Số nguyên dương n ($1 \leq n \leq 100$).

Kết quả: gồm hai số nguyên ghi trên một dòng, cách nhau một khoảng trắng, cho biết số vòng bên tay trái và số vòng bên tay phải tại ngày thứ n.

r) Chữ số tận cùng

Cho 2 số nguyên dương a và N ($a \leq 100$; $N \leq 10^6$)

Tìm chữ số tận cùng của a^N

+ input: 2 số a và N cách nhau 1 khoảng trắng.

+ output: 1 số duy nhất là kết quả tìm được.

Ví dụ

input

2 5

output

2

2 mũ 5 là 32, chữ số tận cùng của 32 là 2.

Chuyên đề 4

KIỂU DỮ LIỆU MẢNG

I. Bài tập và lời giải về mảng một chiều

1. Kiểu dữ liệu mảng một chiều

Là tập hợp các phần tử có cùng kiểu dữ liệu.

Khai báo:

Tenmang : **array**[chisodau..chisocuoi] **of** Kieudulieu;

Ví dụ: Khai báo mảng a chứa tối đa 100 phần tử là số nguyên dương < 256.

Var

A : array[1..100] of byte;

Truy cập đến từng phần tử của mảng: tenmang[chiso]

2. Viết chương trình giải các bài toán

a) *In mảng*

Nhập N và dãy a_1, a_2, \dots, a_n . Đưa dãy vừa nhập ra màn hình.

b) *Sắp xếp nổi bọt*

Sắp xếp dãy vừa nhập thành dãy không giảm. Đưa dãy sau khi sắp xếp ra màn hình.

VD: n = 6; dãy 1 3 4 2 7 3 \rightarrow 1 2 3 3 4 7

c) *Trung bình cộng*

Tính tổng và tính trung bình cộng các phần tử của dãy số. Lấy chính xác đến 2 chữ số thập phân.

VD: n = 6; 1 3 4 2 7 3 \rightarrow tong = 20; tbc = 3.33

d) *Bài toán đếm 1*

Đếm xem có bao nhiêu phần tử có giá trị lớn hơn hoặc bằng trung bình cộng của dãy số.

VD: n = 6; 1 3 4 2 7 3 \rightarrow 2 (giải thích: đó là 4 và 7)

e) *Vị trí đẹp 1*

Tìm các vị trí I (nếu có) chia dãy số thành 2 phần có tổng bằng nhau. Nếu không có thì ghi ra -1.

VD: n = 6; 1 3 4 2 7 3 \rightarrow i = 4 (giải thích: $1+3+4+2=7+3$)

f) *Tìm min, max*

Tìm giá trị lớn nhất và giá trị nhỏ nhất của dãy số.

VD: n = 6; 1 3 4 2 7 3 \rightarrow 1 ; 7

g) *Vị trí đạt giá trị nhỏ nhất*

Đưa ra các vị trí phần tử đạt giá trị nhỏ nhất.

VD: n = 6; 2 1 4 2 1 4 \rightarrow 2 ; 5

3. Lời giải các bài tập mục 2

a) *In mảng*

Program inmang;

Var

A: array[1..100] of byte;

N, i: byte;

Begin

```

Write('Nhap N='); Readln(N);
Writeln('Nhap cac phan tu cua mang: ');
For i:=1 to n do read(a[i]);
For i:=1 to n do
    Write(a[i], ' ');
Readln;
End.

```

b) Sắp xếp nổi bọt

```

Program sapxepmang;
Var
    A: array[1..100] of byte;
    N, i: byte;
    Tam: byte;
Begin
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to n do read(a[i]);
    For i:=1 to n do Write(a[i], ' ');
    Writeln('Mang sau khi sap xep la:');
    For i:=1 to n - 1 do
        For j:=i+1 to n do
            If a[i]>a[j] then
                Begin
                    Tam := a[i];
                    A[i] := a[j];
                    A[j] := tam;
                End;
    For i:=1 to n do Write(a[i], ' ');
    Readln;
End.

```

c) Trung bình cộng

```

Program trungbinhcong;
Var
    A: array[1..100] of byte;
    N, i: byte;
    Tbc: real;
    Tong: integer;
Begin
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');

```

```

For i:=1 to n do read(a[i]);
Tong := 0;
For i:=1 to n do tong := tong + a[i];
Tbc := tong/n;
Writeln('Tong cac phan tu = ', tong);
Writeln('Trung binh cong = ', tbc:2);
Readln;
End.

```

d) Bài toán đếm 1

```

Program Dem1;
Var
  A: array[1..100] of byte;
  N, i, res: byte;
  Tbc: real;
  Tong: integer;
Begin
  Write('Nhap N='); Readln(N);
  Writeln('Nhap cac phan tu cua mang: ');
  For i:=1 to n do read(a[i]);
  Tong := 0;
  For i:=1 to n do tong := tong + a[i];
  Tbc := tong/n;
  For i:=1 to n do
    If a[i] >= tbc then res := res + 1; //inc(res);
  Writeln('Res = ', res);
  Readln;
End.

```

e) Vị trí đẹp 1

```

Program beautiful1;
Var
  A: array[1..100] of byte;
  S: array[1..100] of integer;
  N, i, p: byte;
Begin
  Write('Nhap N='); Readln(N);
  Writeln('Nhap cac phan tu cua mang: ');
  For i:=1 to n do read(a[i]);
  S[1] := a[1];
  For i:=2 to n do s[i] := s[i-1] + a[i]; //gọi S[i] là tổng các phần tử từ 1 tới i.
  P := 0;

```

```

For i:=1 to n do
    If s[i] = s[n]/2 then
        Begin
            Write(i, ' ');
            P := 1;
        End;
    If p = 0 then write(-1);
    Readln;
End.

```

f) *Tim min, max*

```

Program Timminmax;
Var
    A: array[1..100] of byte;
    N, i, resmin, resmax: byte;
Begin
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to n do read(a[i]);
//Tim min
    resmin := a[1];
    For i:=2 to n do
        If resmin > a[i] then resmin := a[i];
    Writeln('GTNN =', resmin);
//Tim max
    resmax := a[1];
    For i:=2 to n do
        If resmax < a[i] then resmax := a[i];
    Writeln('GTLN =', resmax);
    Readln;
End.

```

g) *Vị trí đạt giá trị nhỏ nhất*

Đưa ra các vị trí phần tử đạt giá trị nhỏ nhất.

VD: n = 6; 2 1 4 2 1 4 → 2 ; 5

```

Program Minpos;
Var
    A: array[1..100] of byte;
    N, i, resmin: byte;
Begin
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');

```

```

    For i:=1 to n do read(a[i]);
//Tìm min
    resmin := a[1];
    For i:=2 to n do
        If resmin > a[i] then resmin := a[i];
    For i:=2 to n do
        If a[i] = resmin then write(i, ' ');
    Readln;
End.

```

4. Bài tập rèn luyện

a) *Giá trị nhỏ nhất*

Tìm giá trị nhỏ nhất (nếu có) của dãy số và đưa ra các vị trí đạt giá trị đó. Nếu không có thì ghi ra -1.

VD: $n = 6$; dãy 1 3 4 2 7 3 $\rightarrow \min = 2$; vitri = 4

b) *Giá trị lớn thứ K*

Tìm giá trị lớn thứ K (nếu có) của dãy số.

Với K nguyên dương, $K \leq N$, K nhập từ bàn phím. Nếu không có thì ghi ra -1.

VD: $n = 6$; 1 3 4 2 7 3; $K=5 \rightarrow \max_k = 7$

c) *Tổng 3 phần tử*

Tìm dãy 3 phần tử liên tiếp có tổng lớn nhất. Đưa tổng lớn nhất tìm được ra màn hình.

VD: $n = 6$; 1 3 4 2 7 3 $\rightarrow \text{tong} = 13$

d) *Tìm kiếm tuần tự*

Nhập giá trị X. Tìm vị trí của X (nếu có) trong dãy số và đếm số lượng phần tử có giá trị bằng X. Nếu không có thì ghi ra -1.

VD: $n = 6$; 1 3 4 2 7 3 ; $x=3 \rightarrow 2$ 6; soluong = 2

e) *Dãy không giảm*

Kiểm tra dãy có là dãy không giảm hay không?

Nếu có ghi ra 'YES' ngược lại ghi ra 'NO'.

VD: $n = 6$; dãy 1 3 4 2 7 3 $\rightarrow \text{NO}$

VD: $n = 6$; dãy 1 3 4 6 7 8 $\rightarrow \text{YES}$

f) *Số Fibonacci*

Số Fibonacci là số được định nghĩa:

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

$F[n] = F[n-1] + F[n-2]$ với $N \geq 3$.

Cho N, hãy xác định $F[n]$.

Vd: $n = 3$; $f[3] = 2$; $n = 4$; $f[4] = 3$.

g) *Bài toán đếm 2*

Đếm số cặp (i,j) sao cho $a_i + a_j = k$. Nếu không có thì ghi ra -1.

VD: $n = 6$; dãy 1 3 4 2 7 3; $k=4 \rightarrow \text{res} = 2$

h) *Dãy không giảm 2*

Liệt kê các dãy con liên tiếp không giảm (có nhiều hơn 1 phần tử) của dãy ban đầu, mỗi dãy trên một dòng. Nếu không có ghi ra -1.

VD: $n = 6$; 1 3 4 2 7 3;

1 3 4

2 7

i) Mảng chẵn lẻ

Tạo mảng b lưu các số chẵn, mảng c lưu các số lẻ từ mảng a.

Đưa mảng b, mảng c ra màn hình.

VD: $n = 6$; 1 3 4 2 7 3

→ 4 2

1 3 7 3

j) Tạo mảng C

Cho mảng a (n phần tử) và b (m phần tử) là dãy không giảm. Tạo mảng c có $m+n$ phần tử cũng là dãy không giảm. Đưa mảng C ra.

VD: $n=3$; 1 3 4; $m=3$; 2 3 7; → 1 2 3 3 4 7

k) Tìm tổng nhỏ nhất

Cho mảng a (n phần tử) và mảng b (m phần tử). Tìm giá trị của $|a_i + b_j|$ nhỏ nhất. Đưa giá trị nhỏ nhất đó ra.

VD: $n=3$; 4 1 3; $m=3$; 2 7 3; → 3

l) Xóa phần tử thứ K

Xóa phần tử thứ K trong dãy số. Với K nhập từ bàn phím ($K \leq N$). Đưa mảng a sau khi xóa ra.

VD: $n = 6$; 1 3 4 2 7 3; $k=3$ → 1 3 2 7 3

m) Xóa phần tử chia hết cho K

Xóa tất cả các phần tử chia hết cho K. Với K nhập từ bàn phím. Đưa mảng a sau khi xóa ra.

VD: $n = 6$; 1 6 4 2 7 3; $k=3$ → 1 2 7

n) Chèn phần tử sau vị trí K

Chèn X vào sau phần tử thứ K của dãy số. Đưa mảng a sau khi chèn ra màn hình.

VD: $n = 6$; 1 3 4 2 7 3; $x=5$; $k=3$; → 1 3 4 5 2 7 3

o) Xóa phần tử giống nhau trong dãy số

Xóa các phần tử giống nhau đứng cạnh nhau trong dãy số, chỉ giữ lại đại diện mỗi số một số. Đưa dãy sau khi xóa ra, giữ nguyên thứ tự.

VD: $n=9$; 1 3 3 4 2 2 2 7 3; → 1 3 4 2 7 3

p) Bài toán đếm 2

Đếm số lần xuất hiện của mỗi phần tử trong dãy số.

VD: $n=8$; 1 3 3 4 2 2 2 7 ;

1 1

2 3 (giải thích: Số 2 xuất hiện 3 lần)

3 2

4 1

7 1

q) Đảo ngược dãy số

Đảo ngược dãy số. Đưa dãy đảo ngược ra.

Vd: N=4; 1 2 3 4 \rightarrow 4 3 2 1

r) Dãy liên tiếp không giảm dài nhất

Tìm dãy con liên tiếp không giảm dài nhất của dãy. Đưa ra độ dài lớn nhất tìm được.

VD: n = 6; 1 3 4 2 7 3 \rightarrow 3 (Giải thích: dãy 1 3 4 dài nhất)

s) Dãy con liên tiếp có tổng lớn nhất

Tìm dãy con liên tiếp là dãy không giảm có tổng lớn nhất. Đưa tổng lớn nhất tìm được ra.

VD: n = 6; 1 3 4 2 7 3 \rightarrow 9 (giải thích: dãy 2 7 có tổng lớn nhất).

t) Dãy bằng nhau liên tiếp dài nhất

Tìm dãy bằng nhau liên tiếp dài nhất trong dãy. Đưa độ dài dài nhất tìm được ra.

VD: N=10, dãy 2 2 1 1 1 3 4 4 4 \rightarrow 4 (giải thích: Dãy 1 1 1 1 bằng nhau, dài nhất).

u) Tìm kiếm tuần tự 2

Tìm vị trí xuất đầu tiên của K trong dãy. VD: n = 6; 1 3 4 2 7 3 ; k=3 \rightarrow 2.

v) Tìm kiếm nhị phân

Cho dãy a không giảm. Tìm vị trí của K trong dãy, nếu có ghi ra 'TRUE' ngược lại ghi ra 'FALSE'.

VD: n = 6; 1 3 4 7 7 9; k =4 \rightarrow TRUE

w) Nhân đôi mảng

Tạo mảng b có 2n phần tử từ mảng a. Đưa mảng b ra.

VD: n = 6; 1 3 4 2 7 3; \rightarrow 1 3 4 2 7 3 1 3 4 2 7 3

x) Vòng tròn

Cho mảng a sắp trên một vòng tròn. Tìm 3 phần tử liên tiếp có tổng nhỏ nhất. Đưa giá trị nhỏ nhất tìm được ra.

II. Bài tập và lời giải về mảng hai chiều

1. Kiểu dữ liệu mảng hai chiều

Khai báo:

Var Tenmang: array[chisodauhang..chisocuoihang, chisodaucot..chisocuoicot] of Kieudulieu;

Ví dụ: Khai báo một mảng hai chiều có tối đa 100 hàng, 100 cột, mỗi phần tử là số nguyên dương < 255.

Var

A: array[1..100, 1..100] of byte;

Truy cập đến các phần tử của mảng: tenmang[chisohang, chisocot]

2. Viết chương trình giải các bài toán

a) In mảng

Nhập ma trận có m hàng, n cột. Đưa mảng vừa nhập ra.

Tính tổng các phần tử của mảng.

b) Tổng số chẵn

Đưa mảng vừa nhập ra. Tính tổng các phần tử là số chẵn.

c) **Đếm K**

Đếm số lượng các phần tử có giá trị bằng K.

d) **Tìm vị trí chia hết cho K**

Đưa ra vị trí những phần tử chia hết cho K. Với K nhập từ bàn phím.

e) **Tổng hàng, tổng cột**

Tính và đưa ra tổng của từng hàng, tổng của từng cột.

f) **Tìm max**

Tìm giá trị lớn nhất của mảng.

g) **Tìm max hàng**

Tìm giá trị lớn nhất của từng hàng.

h) **Tìm giá trị lớn nhì**

Tìm giá trị lớn nhì của mảng.

i) **Ma trận đảo**

Tìm ma trận đảo của ma trận a.

3. Lời giải bài tập mục 2

a) **In mảng**

Program inmang2;

Var

A: array[1..100,1..100] of byte;

M, N, i, j: byte;

Tong: integer;

Begin

Write('Nhap M='); Readln(M);

Write('Nhap N='); Readln(N);

Writeln('Nhap cac phan tu cua mang: ');

For i:=1 to m do

For j:=1 to n do read(a[i,j]);

For i:=1 to m do

Begin

For j:=1 to n do Write(a[i,j], ' ');

Writeln();

End;

Tong := 0;

For i:=1 to m do

For j:=1 to n do

Tong := tong + a[i,j];

Writeln('Tong cac phan tu: ', tong);

Readln;

End.

b) **Tổng số chẵn**


```

Program tinhTong;
Var
  A: array[1..100,1..100] of byte;
  M, N, i, j: byte;
  Tong: integer;
Begin
  Write('Nhap M='); Readln(M);
  Write('Nhap N='); Readln(N);
  Writeln('Nhap cac phan tu cua mang: ');
  For i:=1 to m do
    For j:=1 to n do read(a[i,j]);
  Tong := 0;
  For i:=1 to m do
    For j:=1 to n do
      If a[i,j] mod 2 = 0 then
        Tong := tong + a[i,j];
  Writeln('Tong cac phan tu chan: ', tong);
  Readln;
End.

```

c) **Đếm K**

```

Program DemK;
Var
  A: array[1..100,1..100] of byte;
  M, N, i, j, k: byte;
  Res: byte;
Begin
  Write('Nhap M='); Readln(M);
  Write('Nhap N='); Readln(N);
  Write('Nhap K='); Readln(K);
  Writeln('Nhap cac phan tu cua mang: ');
  For i:=1 to m do
    For j:=1 to n do read(a[i,j]);
  Res := 0;
  For i:=1 to m do
    For j:=1 to n do
      If a[i,j] = k then
        Res := Res + 1;
  Writeln('So phan tu bang K: ', Res);
  Readln;
End.

```

d) Tìm vị trí chia hết cho K

Program VitriK;

Var

A: array[1..100,1..100] of byte;

M, N, i, j, K: byte;

Begin

Write('Nhap M='); Readln(M);

Write('Nhap N='); Readln(N);

Write('Nhap K='); Readln(K);

Writeln('Nhap cac phan tu cua mang: ');

For i:=1 to m do

For j:=1 to n do read(a[i,j]);

For i:=1 to m do

For j:=1 to n do

If a[i,j] mod k = 0 then

Writeln(i, ' ', j);

Readln;

End.

e) Tổng hàng, tổng cột

Program TongHangCot;

Var

A: array[1..100,1..100] of byte;

M, N, i, j: byte;

Tonghang, tongcot: integer;

Begin

Write('Nhap M='); Readln(M);

Write('Nhap N='); Readln(N);

Writeln('Nhap cac phan tu cua mang: ');

For i:=1 to m do

For j:=1 to n do read(a[i,j]);

//Tinh tong hang

For i:=1 to m do

Begin

Tonghang := 0;

For j:=1 to n do tonghang := tonghang + a[i,j];

Writeln('Tong hang ', i, ' = ', tonghang);

End;

//Tinh tong cot

For j:=1 to n do

Begin

```

        Tongcot := 0;
        For i:=1 to m do tongcot := tongcot + a[i,j];
        Writeln('Tong cot ', j, ' = ', tongcot);
    End;
    Readln;
End.

```

f) Tìm max

```

Program Timmax;
Var
    A: array[1..100,1..100] of byte;
    M, N, i, j, Resmax: byte;
Begin
    Write('Nhap M='); Readln(M);
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to m do
        For j:=1 to n do read(a[i,j]);
    Resmax := a[1,1];
    For i:=1 to m do
        For j:=1 to n do
            If resmax < a[i,j] then resmax := a[i,j];
        Writeln(resmax);
    End.

```

g) Tìm max hàng

```

Program TimmaxHang;
Var
    A: array[1..100,1..100] of byte;
    M, N, i, j: byte;
    Res: byte;
Begin
    Write('Nhap M='); Readln(M);
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to m do
        For j:=1 to n do read(a[i,j]);
    For i:=1 to m do
        Begin
            Res := a[i,1];
            For j:=1 to n do
                If res < a[i,j] then res := a[i,j];

```

```

        Writeln('Max hang ', i, '= ', res);
    End;
    Readln;
End.

```

h) Tìm giá trị lớn nhì

Tìm giá trị lớn nhì của mảng.

Program TimMax2;

Var

```

    A: array[1..100,1..100] of byte;
    M, N, i, j, Resmax1, Resmax2: byte;

```

Begin

```

    Write('Nhap M='); Readln(M);
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to m do
        For j:=1 to n do read(a[i,j]);
    Resmax1 := a[1,1];
    For i:=1 to m do
        For j:=1 to n do
            If resmax1 < a[i,j] then resmax1 := a[i,j];
    Resmax2 := -255;
    For i:=1 to m do
        For j:=1 to n do
            If (resmax2 < a[i,j]) and (a[i,j] <> resmax1) then resmax2 := a[i,j];
    Writeln('Gia tri lon thu nhi = ', resmax2);
    Readln;

```

End.

i) Ma trận đảo

Tìm ma trận đảo của ma trận a.

Program Matrandao;

Var

```

    A: array[1..100,1..100] of byte;
    M, N, i, j: byte;

```

Begin

```

    Write('Nhap M='); Readln(M);
    Write('Nhap N='); Readln(N);
    Writeln('Nhap cac phan tu cua mang: ');
    For i:=1 to m do
        For j:=1 to n do read(a[i,j]);
    For j:=1 to n do

```

```

Begin
    For i:=1 to m do Write(a[i,j], ' ');
    Writeln();
End;
Readln;
End.

```

4. Bài tập rèn luyện

a) *Tìm giá trị lớn thứ K*

Tìm giá trị lớn thứ k của mảng.

b) *Sắp xếp hàng*

Sắp xếp các hàng thành các dãy không giảm.

c) *Vị trí phần tử nguyên tố*

Đưa ra vị trí những phần tử là số nguyên tố.

d) *Tìm chỉ số*

Tìm các bộ chỉ số (i,j,k,z) sao cho $a_{ij} + a_{kz} = X$. Với x nhập từ bàn phím.

e) *Hình vuông cạnh K 1*

Tìm các tổng tất cả các ô trong hình vuông có cạnh bằng K. Đưa ra các tổng tìm được.

f) *Hình vuông cạnh K 2*

Tìm hình vuông có cạnh bằng K có tổng lớn nhất. Đưa ra các góc trái trên của các hình vuông tìm được.

g) *Hình vuông có tổng bằng S*

Tìm tất cả các hình vuông có tổng bằng S. với S nhập từ bàn phím

h) *Hình vuông lớn nhất*

Tìm hình vuông lớn nhất có tổng bằng S. Đưa ra độ dài cạnh hình vuông tìm được.

i) *Đếm bộ chỉ số*

Đếm các bộ chỉ số (i,j,k,z) sao cho $a_{ij} + a_{kz} = X$. Với x nhập từ bàn phím.

j) *Hình chữ nhật có tổng bằng S.*

Tìm hình chữ nhật có tổng bằng S. Đưa ra các bộ số là góc trái trên và phải dưới của HCN.

k) *Đếm hình chữ nhật*

Cho ma trận kích thước $m \times n$. Và số nguyên K.

Hãy đếm số lượng hình chữ nhật có tổng chia hết cho K.

l) *Hình chữ nhật toàn số 1*

Cho ma trận kích thước $m \times n$ mà các phần tử có giá trị bằng 0 hoặc 1.

Hãy tìm hình chữ nhật chứa toàn số 1 có diện tích lớn nhất.

Đưa ra diện tích lớn nhất đó.

m) *Xây kè*

Cho ma trận kích thước $m \times n$ mà các phần tử có giá trị bằng 0 hoặc 1.

Tại mỗi ô là số 1 người ta xây 4 bức tường xung quanh, trừ các đường viền của bảng ko phải xây.

Hãy tính số bức tường phải xây ứng với ma trận này.

Ví dụ:

1	0	1	1
0	1	1	0
1	0	0	0

Res = 13

n) Điểm yên ngựa

Cho ma trận $m \times n$. Điểm yên ngựa là điểm có giá trị lớn nhất hàng và nhỏ nhất cột.

Hãy đếm số lượng điểm yên ngựa trong ma trận.

o) Quân xe 1

Cho ma trận $m \times n$. Quân xe trong cờ vua hoặc cờ tướng có đặc điểm di chuyển và ăn quân theo hình dấu (+).

Người ta đặt quân xe ở ô (i, j) thì số điểm thu được bằng tổng giá trị các phần tử ở hàng i và tổng giá trị các phần tử ở cột j và $a[i, j]$.

Hãy tìm vị trí đặt quân xe để đạt tổng điểm lớn nhất. Đưa ra tổng điểm lớn nhất đó.

p) Con kiến 1

Cho ma trận $m \times n$.

Một con kiến muốn di chuyển từ một ô bất kỳ ở cột 1 đến một ô bất kỳ ở cột N . Khi đi qua ô (i, j) con kiến sẽ nhận được $a[i, j]$ (kg) bánh. Hỏi con kiến có thể thu được nhiều nhất bao nhiêu kg bánh?

Biết rằng từ ô (i, j) con kiến chỉ có thể di chuyển đến ô $(i, j+1)$ hoặc $(i-1, j+1)$ hoặc $(i+1, j+1)$.

1	0	1	1
0	1	1	0
1	0	0	0

Res = 4;

q) Quân mã

Cho ma trận kích thước m, n . Tìm vị trí đặt quân Mã để kiểm soát được các ô có tổng lớn nhất (bao gồm cả ô con Mã đang đứng).

r) Chia hết cho 11

Người ta tạo mảng kích thước $m \times 10$ gồm các số nguyên dương liên tiếp từ 1 đến $m \times 10$.

Hãy tìm hình vuông lớn nhất có tổng chia hết cho 11.

s) Con kiến 2

Cho tờ giấy kích thước $m \times n$, ở ô (i, j) ghi các số $a[i, j]$. Người ta cuộn tờ giấy có mép trên của bảng và mép dưới của bảng kề nhau để được hình một ống trụ nằm ngang. Một con kiến đi từ cột 1 đến cột N , xuất phát từ 1 ô bất kỳ ở cột 1. Đi qua ô nào con kiến sẽ có điểm bằng giá trị ở ô đó. Hãy tìm điểm lớn nhất mà con kiến có được với cách đi như trên.

Chuyên đề 5

KIỂU DỮ LIỆU XÂU

I. Bài tập và lời giải về xâu

1. Kiểu dữ liệu xâu

Khai báo

Var

Tenbien : String;

Các hàm cơ bản: length, pos, ord, chr, copy, upcase, concat

Thủ tục: Delete, insert, str, val.

2. Viết chương trình giải các bài toán

a) *Độ dài xâu*

Nhập xâu S. Đưa xâu vừa nhập ra. Đưa ra độ dài của xâu.

Program Dodai;

Uses crt;

Var

S: string;

Begin

Write('Nhap xau S='); readln(s);

Write(length(s));

Readln;

End.

b) *Vị trí S1 trong S*

Nhập xâu S1, S. Đưa ra vị trí đầu tiên của xâu S1 trong xâu S.

Program Vitri;

Uses crt;

Var

S, s1: string;

Begin

Write('Nhap xau S='); readln(s);

Write('Nhap xau S1='); readln(s1);

Write(pos(s1,s));

Readln;

End.

c) *Đếm S1 trong S*

Nhập xâu S1, S. Đếm số lần xuất hiện S1 trong S.

Program Dems1;

Uses crt;

Var

S, s1: string;

Res, p: byte;

```

Begin
  Write('Nhap xau S='); readln(s);
  Write('Nhap xau S1='); readln(s1);
  Res := 0;
  While pos(s1,s) > 0 do
    Begin
      P := pos(s1,s);
      If p > 0 then res := res + 1;
      S := copy(s,p+length(s1)-1, length(s)-length(s1)+1);
    End;
  Write('Res =', res);
  Readln;
End.

```

d) Nối xâu

Nhập xâu S1, S2. Hãy tạo xâu S bằng cách nối S2 vào sau xâu S1. Đưa S ra.

```

Program Congxau;
Uses crt;
Var
  S, s1, s2: string;
Begin
  Write('Nhap xau S1='); readln(s1);
  Write('Nhap xau S2='); readln(s2);
  S := s1 + s2;
  Writeln('S1 + S2 =', S);
  S := s2 + s1;
  Writeln('S2 + S1 =', S);
  Readln;
End.

```

e) Đếm kí tự số

Nhập xâu S. Đếm số kí tự là số trong xâu S.

```

Program Demkitu;
Var
  S: string;
  Res, i: byte;
Begin
  Write('Nhap xau S='); readln(s);
  For i:=1 to length(s) do
    If (s[i] >='0') and (s[i] <='9') then
      Res := Res + 1;
  Write('So ki tu trong xau:', res);

```


End.

f) *Xâu đảo*

Tìm chuỗi đảo của chuỗi S.

Program Xaudao;

Uses crt;

Var

S,Sdao: string;

I: byte;

Begin

Write('Nhap xau S='); readln(s);

Sdao := '';

For i:=1 to length(s) do

Sdao := s[i] + Sdao;

Write('Xau dao: ', sdao);

Readln;

End.

g) *Xâu đối xứng*

Cho chuỗi S. Kiểm tra S có là chuỗi palidrom (chuỗi đối xứng) hay không?

Program XauPalidrom;

Uses crt;

Var

S,Sdao: string;

I: byte;

Begin

Write('Nhap xau S='); readln(s);

Sdao := '';

For i:=1 to length(s) do

Sdao := s[i] + Sdao;

If (S = Sdao) then write('S la xau Palidrom')

Else write('S khong la xau Palidrom');

Readln;

End.

h) *Xâu chữ hoa*

Đổi chuỗi S thành toàn chuỗi chữ hoa.

Program Xauchuhoa;

Uses crt;

Var

S: string;

I: byte;

```

Begin
  Write('Nhap xau S='); readln(s);
  For i:=1 to length(s) do
    S[i] := upcase(s[i]);
  Write('Xau chu hoa: ', s);
  Readln;
End.

```

*i) **Xâu chữ thường***

Đổi xâu S thành toàn xâu chữ thường.

Program Xauchtuong;

Uses crt;

Var

S: string;

I: byte;

Begin

```

  Write('Nhap xau S='); readln(s);
  For i:=1 to length(s) do
    If (s[i] >='A') and (s[i] <='Z') then
      S[i] := chr(ord(s[i])+32);
  Write('Xau chu thuong: ', s);
  Readln;

```

End.

*j) **Chuẩn hóa xâu***

Xóa các dấu cách thừa trong xâu, nếu có nhiều dấu cách đứng cạnh nhau thì giữ lại 1 dấu cách.

Program Chuanhoaxau;

Uses crt;

Var

S: string;

Begin

```

  Write('Nhap xau S='); readln(s);
  While pos(' ',s) > 0 do           // trong cặp dấu ' ' là hai dấu cách
    Delete(s,pos(' ',s),1);
  If s[1] = ' ' then delete(s,1,1);
  If s[length(s)] = ' ' then delete(s,length(s),1);
  Write('Xau da chuan hoa: ', s);
  Readln;

```

End.

*k) **Đếm từ***

Đếm số từ trong xâu.

```

Program Demtu;
Uses crt;
Var
    S: string;
    Res,i : byte;
Begin
    Write('Nhap xau S='); readln(s);
    While pos(' ',s) > 0 do           // trong cặp dấu ' ' là hai dấu cách
        Delete(s,pos(' ', s),1);
    If s[1] = ' ' then delete(s,1,1);
    If s[length(s)] = ' ' then delete(s,length(s),1);
    For i:=1 to length(s) do
        If s[i] = ' ' then res := res + 1;
    Write('So tu trong xau: ', res);
    Readln;
End.

```

l) *Kí tự đầu tiên các từ in hoa*

Chuyển các kí tự đầu tiên của mỗi từ trong xâu thành chữ in hoa.

```

Program Inhoatu;
Uses crt;
Var
    S: string;
    i : byte;
Begin
    Write('Nhap xau S='); readln(s);
    While pos(' ',s) > 0 do           // trong cặp dấu ' ' là hai dấu cách
        Delete(s,pos(' ', s),1);
    If s[1] = ' ' then delete(s,1,1);
    If s[length(s)] = ' ' then delete(s,length(s),1);
    For i:=1 to length(s) do
        If s[i] = ' ' then s[i+1] := upacase(s[i+1]);
    Write('Xau: ', s);
    Readln;
End.

```

m) *Đếm phân phối*

Cho xâu S chỉ gồm các kí tự từ 'a' đến 'z'. Đếm số lần xuất hiện của mỗi kí tự.

```

Program Demphanphoi;
Uses crt;
Var
    S: string;

```

```

A: array['a'..'z'] of byte;
i : char;
Begin
  Write('Nhap xau S='); readln(s);
  For i:= 1 to length(s) do
    A[s[i]] := A[s[i]] + 1;
  For i:= 'a' to 'z' do
    If (a[i] > 0) Writeln(i, ' ', a[i]);
Readln;
End.

```

3. Bài tập rèn luyện

a) *Tách số trong xâu*

Nhập xâu S. Tách các số trong xâu ra.

b) *Số lớn nhất trong xâu*

Nhập xâu S. Tìm số lớn nhất trong xâu.

c) *Sắp xếp kí tự*

Sắp xếp các kí tự trong xâu thành xâu không giảm.

d) *Đếm kí tự 1*

Nhập xâu S. Đếm số lượng kí tự 'a' trong xâu.

e) *Nén xâu*

Nhập xâu S. Hãy nén xâu S. Đưa xâu nén ra. Ví dụ: S= 'aaabbccccc' → S_{nen}='a3b3c4d1'

f) *Giải nén xâu.*

Nhập xâu S_{nen}. Hãy giải nén xâu. Ví dụ: S_{nen}='a3b3c4d1' → S='aaabbccccc'

g) *Xóa S1 trong S*

Nhập xâu S và S1. Hãy xóa tất cả các xâu S1 trong xâu S.

Đưa xâu S sau khi xóa ra.

h) *Xâu Fibonacci 1*

F[1]='a'; F[2]='b'; F[n] = F[n-1] + F[n-2].

Nhập N. Hãy tìm xâu Fibonacci thứ N.

i) *Xâu Fibonacci 2*

F[1]='a'; F[2]='b'; F[n] = F[n-1] + F[n-2].

Nhập N. Hãy đếm số kí tự a và kí tự b trong xâu Fibonacci thứ N.

j) *Xâu con palidrom*

Cho xâu S. Tìm xâu con liên tiếp dài nhất là xâu palidrom.

k) *Sắp xếp số trong xâu*

Cho xâu S. Hãy sắp xếp các số trong xâu thành dãy không giảm.

Ví dụ: S='ab12c3d4' → S_{moi}='ab3c4d12'.

l) *Mã hóa xâu*

Mã hóa xâu S chỉ gồm các kí tự từ a tới z, bằng cách dịch phải các kí tự theo vòng tròn đi K kí tự trên bảng chữ cái. Với K nhập từ bàn phím. Cho xâu S. Tìm xâu mã hóa.

m) Giải mã xâu

Cho xâu mã hóa bằng cách dịch phải K kí tự. Nhập S và K. Hãy giải mã tìm xâu S

n) Ghép xâu số

Cho N xâu số. Hãy ghép các xâu này để được số lớn nhất.

o) Xâu con chung dài nhất

Cho xâu S1 và S2. Tìm xâu con chung của hai xâu.

Chuyên đề 6

BÀI TOÁN LIỆT KÊ

I. Bài tập và lời giải chuyên đề 5

1. Thuật toán Greedy (Tham)

Thử tất cả các khả năng xảy ra, với mỗi khả năng kiểm tra xem nó có thỏa mãn điều kiện hay không? Nếu có thì cập nhật lại nghiệm của bài toán.

Độ phức tạp **Thuật toán**: Bằng số lượng khả năng được sinh ra $[X]$ độ phức tạp của thao tác kiểm tra.

4. Bài tập áp dụng

1. Liệt kê dãy nhị phân độ dài N

Liệt kê các dãy nhị phân có độ dài N .

p) Liệt kê 2

Liệt kê dãy nhị phân không có 3 số 1 đứng cạnh nhau.

q) Liệt kê 3

Liệt kê dãy nhị phân có số lượng số 0 bằng số lượng số 1.

r) Liệt kê 4

Liệt kê dãy nhị phân có đúng K số 1. Với K nhập từ bàn phím.

s) Liệt kê 5

Liệt kê dãy nhị phân có không quá K số 1. Với K nhập từ bàn phím.

Mid: Nhập N và dãy b_1, b_2, \dots, b_N .

t) Dãy con có tổng bằng K

Hãy đưa ra các cách chọn một số phần tử trong dãy b sao cho có tổng $= K$. Với K nhập từ bàn phím.

u) Dãy con có nhiều phần tử nhất có tổng $\leq K$

Hãy đưa ra các cách chọn nhiều phần tử nhất trong dãy b sao cho có tổng $\leq K$. Đưa ra số lượng nhiều nhất các phần tử được chọn.

v) Chia nhóm 1

Hãy liệt kê ra các cách chia dãy trên thành hai nhóm sao cho tổng mỗi nhóm bằng nhau. Đưa ra hai dãy là chỉ số của các phần tử được chọn của mỗi nhóm.

w) Dãy có tổng chia hết cho K

Hãy đưa ra cách chọn nhiều phần tử nhất có tổng chia hết cho K .

x) Chia nhóm có độ chênh lệch nhỏ nhất

Hãy liệt kê ra các cách chia dãy trên thành hai nhóm sao cho tổng mỗi nhóm có độ chênh lệch nhỏ nhất. Đưa ra hai dãy là chỉ số của các phần tử được chọn của mỗi nhóm và độ chênh lệch nhỏ nhất.

High: Nhập N và dãy b_1, b_2, \dots, b_N , dãy c_1, c_2, \dots, c_N .

y) Dãy con 1

Hãy đưa ra các cách chọn các phần tử $b[i]$ để tổng chia hết cho K và tổng $c[i]$ được chọn lớn nhất. Đưa ra giá trị tổng $c[i]$ lớn nhất tìm được.

z) Dãy con 2

Hãy đưa các cách chọn các phần tử $b[i]$ để tổng các phần tử được chọn $\leq K$ và tổng $c[i]$ được chọn là lớn nhất. Với K nhập từ bàn phím.

aa) Dãy con 3

Hãy đưa ra các cách chọn một số phần tử để tổng $b[i]$ được chọn đúng bằng K và $|C_{\max} - C_{\min}|$ lớn nhất. Đưa hiệu $|C_{\max} - C_{\min}|$ lớn nhất. Với K nhập từ bàn phím.

bb) Dãy con 4

Hãy đưa ra các cách chọn đúng K phần tử để tổng $b[i]$ lớn nhất và tổng $c[i]$ là lớn nhất. Đưa tổng lớn nhất của $b[i]$ và $c[i]$ ra.

Hard: Nhập N và dãy b_1, b_2, \dots, b_N , dãy c_1, c_2, \dots, c_N , dãy d_1, d_2, \dots, d_N .

cc) Dãy con 5

Hãy đưa ra các cách chọn một số phần tử để tổng $b[i]$ được chọn $\leq M$, tổng $d[i]$ được chọn $\leq V$ và tổng $c[i]$ được chọn là lớn nhất. Đưa ra tổng lớn nhất tìm được.

dd) Dãy con 6

Hãy đưa ra các cách chọn K phần tử để tổng $b[i] \leq M$, tổng $d[i] \leq V$ và tổng $c[i]$ là lớn nhất. Đưa ra tổng lớn nhất thu được.

ee) Dãy con 7

Hãy đưa ra cách chọn nhiều nhất các phần tử để tổng $b[i] \leq M$, tổng $d[i] \leq V$ và tổng $c[i]$ là bằng K . Đưa ra tổng lớn nhất thu được.

LỜI GIẢI BÀI TẬP ĐỀ XUẤT

1. Bài tập đề xuất 2

a) *Tổng các chữ số*

Thuật toán: Ta dùng phép toán chia nguyên / và chia dư % cho 10 để tách các chữ số của n.

```
#include <bits/stdc++.h>
using namespace std;
int a;
int main()
{
    freopen("NSUM.INP", "r", stdin);
    freopen("NSUM.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);
    cin >> a;
    cout << a/10 + a%10;
    return 0;
}
```

b) *Tích các chữ số*

Thuật toán: Ta dùng phép toán chia nguyên / và chia dư % cho 10 để tách các chữ số của n.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int main()
{
    freopen("MMUL.INP", "r", stdin);
    freopen("MMUL.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);
    cin >> n;
    cout << (n/100)*(n%10)*((n/10)%10);
    return 0;
}
```

c) *Đổi thời gian*

Thuật toán: Ta sử dụng phép chia nguyên / và chia dư % để tính toán. Ta đổi giây thành giờ trước, số giây còn lại tiếp tục được đổi thành phút, sau đó ta tính được số giây còn dư.

```
#include <bits/stdc++.h>
using namespace std;
int t, a, b, c;
int main()
{

```



```

freopen("TIME.INP", "r", stdin);
freopen("TIME.OUT", "w", stdout);
ios_base::sync_with_stdio(false);
cin.tie(NULL);
cout.tie(NULL);
cin >> t;
a = t/3600;
t -= a*3600;
if (t > 0)
{
    b = t/60;
    t -= b*60;
}
cout << a << ' ' << b << ' ' << t;
return 0;
}

```

d) Đổi tiền 1

Thuật toán: Để thu được số tờ tiền là ít nhất, ta sẽ rút tiền theo mệnh giá từ cao xuống thấp. Với số tiền dư còn lại ta sẽ rút tiếp với mệnh giá tiếp theo.

```

#include <bits/stdc++.h>
using namespace std;
int x,a,b,c,d,e,f;
int main()
{
    freopen("JAMEA.INP", "r", stdin);
    freopen("JAMEA.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x;
    a = x/100;
    x -= a*100;
    if (x > 0)
    {
        b = x/50;
        x -= b*50;
        if (x > 0)
        {
            c = x/20;
            x -= c*20;

```

```

    if (x > 0 )
    {
        d = x/10;
        x -= d*10;
        if (x > 0)
        {
            e = x/5;
            x -= e*5;
            if (x > 0)
            {
                f = x/2;
                x -= f*2;
            }
        }
    }
}

cout << a + b + c + d + e + f + x;
return 0;
}

```

e) ***Khoảng cách***

```

#include <bits/stdc++.h>
using namespace std;
int x11, y11, x2, y2, x3, y3;
float vlong(int m, int n, int p, int q)
{
    return sqrt(pow((m-n),2) + pow((p-q),2));
}
int main()
{
    freopen("DIS.INP", "r", stdin);
    freopen("DIS.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x11 >> y11 >> x2 >> y2 >> x3 >> y3;
    float S = abs((x2-x11)*(y3-y11)-(x3-x11)*(y2-y11))/2;
    cout << 2*S/vlong(x11, x2, y11, y2);
    return 0;
}

```

f) Tam giác 1

```
#include <bits/stdc++.h>
using namespace std;
int x11, y11, x2, y2, x3, y3;
float P, S;
float vdis (int m, int n, int p, int q)
{
    return sqrt(pow((m-p),2)+pow((n-q),2));
}
int main()
{
    freopen("TRI.INP", "r", stdin);
    freopen("TRI.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x11 >> y11 >> x2 >> y2 >> x3 >> y3;
    cout << vdis(x11, y11, x2, y2) << " " << vdis(x11, y11, x3, y3) << " " << vdis(x2, y2, x3,
y3) << endl;
    S = abs((x2-x11)*(y3-y11)-(x3-x11)*(y2-y11))/2; //?
    cout << S;
    return 0;
}
```

g) Hình chữ nhật 1

```
#include <bits/stdc++.h>
using namespace std;
int x11, y11, x2, y2;
int ds(int m, int n)
{
    return abs(n-m);
}
int main()
{
    freopen("REC1.INP", "r", stdin);
    freopen("REC1.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x11 >> y11 >> x2 >> y2;
    cout << ds(x2, x11) << " " << ds(y2, y11);
}
```

```

    return 0;
}

```

h) Hình chữ nhật 2

```

#include <bits/stdc++.h>
using namespace std;
float a, b;
int main()
{
    freopen("REC2.INP", "r", stdin);
    freopen("REC2.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> a >> b;
    cout << (a-1)*(b-1);
    return 0;
}

```

i) Hình chữ nhật 3

Cho hình chữ nhật có cạnh a, b và được chia thành lưới ô vuông có cạnh 1 cm. Hãy đếm số cạnh của các lưới ô vuông đơn vị bên trong hình chữ nhật.

j) Hình chữ nhật 4

```

#include <bits/stdc++.h>
using namespace std;
int a, b;
int main()
{
    freopen("REC4.INP", "r", stdin);
    freopen("REC4.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> a >> b;
    cout << 2*a + 2*b - 4 << " " << (a - 2)*(b - 2);
    return 0;
}

```

k) Hình chữ nhật 5

```

#include <bits/stdc++.h>
using namespace std;
int X, Y;
float m, n;

```

```

int abfider(float b, int c, float *p, float *q)
{
    float d = b*b - 4*c;
    if (d < 0)
        return -1;
    else if (d == 0)
    {
        *p = b/2;
        return 1;
    }
    else
    {
        *p = (b - sqrt(d))/2;
        *q = (b + sqrt(d))/2;
        return 2;
    }
}

int main()
{
    freopen("REC5.INP", "r", stdin);
    freopen("REC5.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> X >> Y;
    switch(abfider((X+4)/2, X+Y, &m, &n))
    {
    case -1:
    {
        cout << -1;
    }
    case 1:
    {
        cout << m << " " << n ;
        break;
    }
    case 2:
    {
        cout << m << " " << n;
    }
    }
}

```

```

    }
    return 0;
}
l) Tim X, Y
#include <bits/stdc++.h>
using namespace std;
int x, y;
int main()
{
    freopen("FNUM.INP", "r", stdin);
    freopen("FNUM.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x >> y;
    if (x % 2 == 0)
        cout << x / 2 << " " << y * 2;
    else
    {
        x++;
        y += x;
        cout << x << " " << y;
    }
    return 0;
}

```

```

m) Tim Q
#include <bits/stdc++.h>
using namespace std;
string q;
int main()
{
    freopen("FQ.INP", "r", stdin);
    freopen("FQ.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> q;
    sort(q.begin(), q.end());
    cout << q[q.size() - 1];
    return 0;
}

```

```

    }
n) Xóa số
#include <bits/stdc++.h>
using namespace std;
string n;
int main()
{
    freopen("DNUM.INP", "r", stdin);
    freopen("DNUM.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> n;
    string temp = n;
    sort(temp.begin(), temp.end());
    char d = temp[0];
    for (int i = 0; i < 3; i++)
        if (n[i] == d)
        {
            n.replace(i, 1, "");
            break;
        }
    cout << n;
    return 0;
}

```

o) **Đổi tiền 2**

```

#include <bits/stdc++.h>
using namespace std;
int x, a, b;
int main()
{
    freopen("J1.INP", "r", stdin);
    freopen("J1.OUT", "w", stdout);
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cin >> x;
    a = x/50;
    b = (x - 50*a)/20;
    x = x - 50*a - 20*b;
}

```

```
if (x != 0)
    cout << -1;
else
    cout << a+b;
return 0;
}
```

p) Số trung bình cộng
q) Giải phương trình
r) Tam giác
s) Điểm và đường tròn 2
t) Vị trí tương đối của hai đường tròn
u) Giải phương trình

Chuyên đề 7

QUY HOẠCH ĐỘNG

a) *Tính giai thừa của số tự nhiên N (kí hiệu: $N!$)*

Ta có định nghĩa như sau: $n! = \begin{cases} 1 & \text{nếu } n = 0 \\ n * (n-1)! & \text{nếu } n > 0 \end{cases}$

Ví dụ: $3! = 3 * 2! = 3 * 2 * 1 = 6$

Cho một số tự nhiên n ($0 \leq n \leq 20$).

Yêu cầu: Hãy tính giai thừa của số tự nhiên n cho trước.

Dữ liệu vào: Ghi trong file văn bản GT.INP gồm một dòng duy nhất ghi số tự nhiên n .

Dữ liệu ra: Ghi ra file văn bản GT.OUT số duy nhất là giá trị tính được của $n!$

Ví dụ:

GT.INP	GT.OUT
3	6

b) *Số Fibonacci*

Số Fibonacci được xác định bởi công thức sau:

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

Ví dụ:

i	0	1	2	3	4	5	6
F _i	0	1	1	2	3	5	8

Cho một số nguyên dương n ($0 \leq n \leq 50$).

Yêu cầu: Hãy tính số Fibonacci thứ n .

Dữ liệu vào: Ghi trong file văn bản FIBO.INP gồm một dòng duy nhất ghi số nguyên dương n .

Dữ liệu ra: Ghi ra file văn bản FIBO.OUT số duy nhất là giá trị tính được của $F(n)$.

Ví dụ:

FIBO.INP	FIBO.OUT
5	8

c) *Tính tổng của dãy số*

Cho dãy số nguyên gồm n phần tử a_1, a_2, \dots, a_n ($1 \leq n \leq 10^5$) và hai số nguyên dương p và q ($1 \leq p \leq q \leq n$).

Yêu cầu: Hãy tính tổng của các phần tử liên tiếp từ $a_p \dots a_q$.

Dữ liệu vào: Ghi trong file văn bản SUM.INP có cấu trúc như sau:

- Dòng 1: Ghi số nguyên dương n và k , hai số được ghi cách nhau một dấu cách.

- Dòng 2: Ghi n số nguyên a_1, a_2, \dots, a_n , các số được ghi cách nhau ít nhất một dấu cách.

- Dòng thứ i trong k dòng tiếp theo: Mỗi dòng ghi hai số nguyên dương p_i và q_i cách nhau một dấu cách ($1 \leq p_i \leq q_i \leq n$).

Dữ liệu ra: Ghi ra file văn bản SUM.OUT theo cấu trúc như sau:

- Dữ liệu được ghi trên k dòng: Dòng thứ i ghi một số nguyên là tổng giá trị của các phần tử trong đoạn $a_{p_i} \dots a_{q_i}$

Ví dụ:

SUM.INP	SUM.OUT
5 3	21
2 9 -3 5 8	6
1 5	5
2 3	
4 4	

d) Hiệu số

Cho một dãy n số nguyên a_1, a_2, \dots, a_n . Hãy tìm hai chỉ số i, j sao cho $i < j$ và hiệu $a_j - a_i$ là lớn nhất.

Dữ liệu vào: ghi trong file HIEUSO.INP gồm 2 dòng

- Dòng 1: là số nguyên n ($2 \leq n \leq 10^5$)
- Dòng 2: gồm n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^5$)

Dữ liệu ra: ghi ra file HIEUSO.OUT gồm một số duy nhất là giá trị lớn nhất của hiệu $a_j - a_i$.

Ví dụ:

HIEUSO.INP	HIEUSO.OUT
3 1 2 3	2
4 2 5 1 3	3

e) Lưới ô

Cho một lưới ô vuông kích thước $N \times N$ ($0 < N < 1000$). Tại một ô bất kì trên lưới, chỉ có thể di chuyển sang phải một ô hoặc đi xuống dưới một ô. Hãy tìm đường đi có tổng lớn nhất từ ô $(1,1)$ đến ô (N,N) .

Dữ liệu vào: Từ file văn bản “GRIP.INP” gồm nhiều dòng:

- Dòng 1: Ghi số nguyên dương N là kích thước của lưới ô vuông ($0 < N < 1000$).
- N dòng tiếp theo, mỗi dòng ghi N số nguyên $a[i,j]$ ($a[i,j] < 100$).

Dữ liệu ra: Ghi ra tệp văn bản “GRIP.OUT” gồm:

Một số duy nhất là tổng lớn nhất trên đường đi tìm được.

Ví dụ:

GRIP.INP	GRIP.OUT	Giải thích
5 2 7 2 6 5 7 1 8 1 4 4 9 3 6 4 1 1 9 5 2 9 5 2 6 1	46	Đường đi $(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (4,3) \rightarrow (4,4) \rightarrow (5,4) \rightarrow (5,5)$, có tổng lớn nhất là 46.

f) **TRIANGLE PASCAL (Tam giác Pascal)**

Tam giác Pascal là một mô hình dùng để đưa ra các hệ số của khai triển nhị thức Newton bậc $N(x+1)^N$.

Ví dụ: trong khai triển $(x+1)^2 = x^2 + 2x + 1$ có các hệ số là:

1 2 1

Trong khai triển $(x+1)^3 = x^3 + 3x^2 + 3x + 1$ có các hệ số là:

1 3 3 1

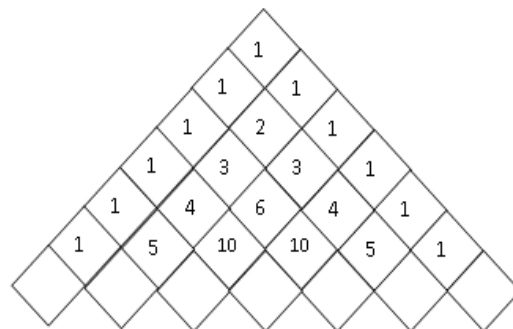
Yêu cầu: Hãy tìm các hệ số trong khai triển nhị thức Newton $(x+1)^N$.

Dữ liệu vào: Cho trong file văn bản TRIANPAS.INP gồm một dòng duy nhất ghi số nguyên dương N ($1 \leq N \leq 100$).

Dữ liệu ra: Ghi ra file văn bản TRIANPAS.OUT: Các số nguyên dương lần lượt là các hệ số trong khai triển nhị thức Newton $(x+1)^N$, các số được ghi cách nhau một dấu cách.

Ví dụ:

TRIANPAS.INP	TRIANPAS.OUT
5	1 5 10 10 5 1



g) **TRIANGLE NUMBER (Tam giác số)**

Cho tam giác số như hình vẽ. Ta định nghĩa một đường đi trong tam giác số là đường đi xuất phát từ hình thoi ở đỉnh tam giác và đi đến được các hình thoi có chung cạnh với nó, đường đi kết thúc khi gặp một hình thoi ở đáy tam giác.

Yêu cầu: Hãy tìm một đường đi trong tam giác số sao cho tổng giá trị của các ô trong đường đi có giá trị lớn nhất.

Dữ liệu vào: Cho trong file văn bản TRIANNUM.INP có cấu trúc như sau:

Dòng 1: Ghi số nguyên dương N là số hàng của tam giác ($1 \leq N \leq 100$).

Dòng thứ i trong N dòng tiếp theo: Ghi i số nguyên dương lần lượt là giá trị của các ô trên dòng thứ i ứng trong tam giác (Các số có giá trị không quá 32000). Các số được ghi cách nhau một dấu cách.

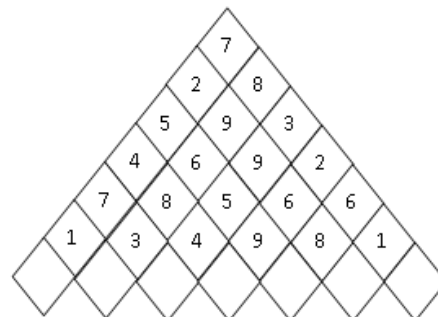
Dữ liệu ra: Ghi ra file văn bản TRIANNUM.OUT theo cấu trúc:

Dòng 1: Ghi ra số nguyên dương S là tổng giá trị của đường đi tìm được.

Ví dụ:

h)

TRIANNUM.INP	TRIANNUM.OUT
6	48
7	
2 8	
5 9 3	
4 6 9 2	
7 8 5 6 6	
1. 3 4 9 8 1	



h) Dãy con đơn điệu tăng dài nhất

Cho dãy số nguyên $A = a_1, a_2, \dots, a_n$ ($n \leq 1000, -10000 \leq a_i \leq 10000$). Một dãy con của dãy A là một cách chọn ra trong A một số phần tử giữ nguyên thứ tự. Như vậy A có 2^n dãy con.

Yêu cầu: Hãy tìm dãy con đơn điệu tăng của A có độ dài lớn nhất bắt đầu từ a_1 .

Dữ liệu vào: Cho trong file văn bản SEQ.INP có cấu trúc như sau:

Dòng 1: Ghi số nguyên dương N là số phần tử của dãy ($1 \leq N \leq 1000$).

Dòng 2: Ghi dãy số a_1, a_2, \dots, a_n , mỗi số cách nhau ít nhất 1 dấu cách

Dữ liệu ra: Ghi vào file văn bản SEQ.OUT theo cấu trúc

Dòng 1: Ghi độ dài của dãy con đơn điệu tăng dài nhất của dãy A

Dòng 2: Ghi các phần tử của dãy con đó.

Ví dụ:

SEQ.INP	SEQ.OUT
10	8
1 2 3 4 9 10 5 6 7 8	1 2 3 4 5 6 7 8

i) Di chuyển từ tây sang đông

Cho ma trận $m \times n$ mỗi ô chứa một số nguyên ta cần di chuyển từ một ô bất kì thuộc cột ngoài cùng bên trái sang một ô bất kì thuộc cột ngoài cùng bên phải. Mỗi bước di chuyển từ một ô (i, j) ta có thể đi sang ô $(i-1, j+1)$ hoặc $(i, j+1)$ hoặc $(i+1, j+1)$. Chi phí cho một đường đi là tổng của các số nguyên trên con đường đó.

Yêu cầu: Hãy tìm ra một con đường đi với chi phí thấp nhất.

Dữ liệu vào: Cho trong file Taydong.inp có cấu trúc như sau:

Dòng 1: Ghi 2 số m, n

Dòng thứ i trong số m dòng tiếp theo: chứa n số nguyên trong phạm vi $(-1000 \dots 1000)$ ($m, n \leq 100$)

Dữ liệu ra: Ghi vào file Taydong.out theo cấu trúc

Dòng đầu: ghi một nguyên là chi phí min tìm được.

n dòng tiếp theo ghi chỉ số của hàng lần lượt di chuyển từ tây sang đông.

Ví dụ:

Taydong.inp	Taydong.out
4 4	9
2 4 3 1	1
8 5 2 9	1
1 7 4 6	2
1 7 8 9	1

HƯỚNG DẪN GIẢI CHUYÊN ĐỀ QUY HOẠCH ĐỘNG

a) *Tính giai thừa của số tự nhiên N (kí hiệu: $N!$)*

GT.PAS

Thuật toán:

Gọi $GT[i]$ là giá trị của $i!$ ($0 \leq i \leq 13$)

Ta có công thức như sau:

$$GT[i] := GT[i-1] * i;$$

Như vậy, việc tính $n!$ sẽ được thực hiện bằng vòng lặp:

$$GT[0] := 1;$$

For $i := 1$ to n do

$$GT[i] := GT[i-1] * i;$$

Kết quả: giá trị của $n!$ nằm trong phần tử $GT[n]$.

```
program giaithua;
Const
    fi = 'GT.INP';
    fo = 'GT.OUT';
var
    GT: array[0..20] of int64;
    n,i: integer;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(f,n);
    GT[0] := 1;
    for i:=1 to n do
        GT[i] := GT[i-1]*i;
    write(GT[n]);
    close(input);
    close(output);
END.
```

b) *Số Fibonacci FIBO.PAS*

Thuật toán:

Gọi $F[i]$ là giá trị Fibonacci của f_i ($0 \leq i \leq 50$).

Ta có công thức như sau:

$$F[i] := F[i-1] + F[i-2];$$

Như vậy, việc tính f_n được thực hiện bằng vòng lặp:

$$F[0] := 0;$$

$$F[1] := 1;$$

For $i := 2$ to n do

$$F[i] := F[i-1] + F[i-2];$$

Kết quả: giá trị f_n nằm trong $F[n]$.

```

program Fibonacci;
Const
    fi = 'FIBO.INP';
    fo = 'FIBO.OUT';
var
    F: array[0..50] of int64;
    n,i: integer;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    F[0] := 0;
    F[1] := 1;
    for i:=2 to n do
        F[i]:= F[i-1] + F[i-2];
    write(F[n]);
    close(input);
    close(output);
END.

```

c) *Tính tổng của dãy số SUM.PAS*

Thuật toán:

Gọi $A[i]$ là giá trị của phần tử thứ i trong dãy số a_1, a_2, \dots, a_n .

Gọi $T[i]$ là tổng giá trị các phần tử a_1, a_2, \dots, a_i ($1 \leq i \leq n$).

Ta có công thức để tính $T[i]$ như sau:

$$T[i] := T[i - 1] + A[i];$$

Như vậy, việc tính $T[n]$ được thực hiện bằng vòng lặp:

$$T[0] := 0;$$

For $i:=1$ to n do

$$T[i] := T[i - 1] + A[i];$$

Kết quả: Tổng các phần tử liên tiếp từ a_p đến a_q được tính theo công thức:

$$\text{Sum} := A[q] - A[p-1];$$

```

[■] SUM.pas
program Tong;
Const
    fi = 'SUM.INP';
    fo = 'SUM.OUT';
var
    A: array[0..1000] of integer;
    T: array[0..1000] of int64;
    n,i,k,p,q: integer;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n,k);
    for i:= 1 to n do
        read(A[i]);
    for i:= 0 to n do
        T[i] := 0;
    for i:=1 to n do
        T[i]:= T[i-1] + A[i];
    for i:=1 to k do
        begin
            readln(p,q);
            writeln(T[q] - T[p-1]);
        end;
    close(input);
    close(output);
END.

```

d) *Hiệu số HIEUSO.PAS*

Thuật toán:

Gọi $A[i]$ là giá trị của phần tử thứ i trong dãy số a_1, a_2, \dots, a_n .

Gọi $M[i]$ là giá trị nhỏ nhất trong các phần tử từ a_1, a_2, \dots, a_i

Gọi $H[i]$ là hiệu số lớn nhất thỏa mãn cần tìm từ a_1, a_2, \dots, a_i

Ta có công thức để tính $M[i]$ và $H[i]$ như sau:

$$M[i] := \min(M[i-1], A[i]);$$

$$H[i] := \max(H[i-1], A[i] - M[i]);$$

Công thức trên được áp dụng với $i = 1 \rightarrow n$.

Kết quả trả về chính là $H[n]$.

```

[■] HIEUS0.pas
program Hieuso;
Const
    fi = 'HIEUS0.INP';
    fo = 'HIEUS0.OUT';
    MaxInt = 32767;
var
    A: array[0..100000] of integer;
    M,H: array[0..100000] of integer;
    n,i: integer;
function min(a, b: integer): integer;
begin
    if (a>b) then
        min := b
    else
        min := a;
end;
function max(a, b: integer): integer;
begin
    if (a>b) then
        max := a
    else
        max := b;
end;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:= 1 to n do
        read(A[i]);
    for i:= 0 to n do
        begin
            M[i] := MaxInt;
            H[i] := 0;
        end;
    M[1] := A[1];
    H[1] := 0;
    for i:= 2 to n do
        begin
            M[i] := min(M[i-1],A[i]);
            H[i] := max(H[i-1], A[i] - M[i]);
        end;
    write(H[n]);
    close(input);
    close(output);
END.

```

e) *Lưới ô GRID.PAS*

Thuật toán:

+ Ta xây dựng mảng hai chiều có kích thước [0..1000, 0..1000]

- Mảng A chứa các phần tử của lưới ô đã cho.
- Mảng L: mỗi phần tử $L[i,j]$ sẽ chứa tổng lớn nhất các ô từ $A[1,1]$ tới $A[i,j]$ theo đúng cách đi.

+ Sử dụng công thức như sau:

Mỗi ô $L[i,j]$ được tính dựa vào ô bên trái $L[i,j-1]$ hoặc ô bên trên $L[i-1,j]$ như sau:

$$L[i, j] = \max(L[i,j-1], L[i-1, j]) + A[i,j];$$

+ **Thuật toán** cụ thể như sau:

$L[1,1] = A[1,1];$

for $i := 2$ to n do

begin

$L[1,i] := L[1,i-1] + a[1,i];$

$L[i,1] := L[i-1,1] + a[i,1];$

end;

for $i := 2$ to n do

for $j := 2$ to n do

if $L[i-1,j] > L[i,j-1]$ then

$L[i,j] := L[i-1,j] + a[i,j]$

else

$L[i,j] := L[i,j-1] + a[i,j];$

+ Kết quả lưu trong $L[n,n]$

```
[■] GRID.pas
program Luoio;
Const
    fi = 'GRID.INP';
    fo = 'GRID.OUT';
var
    A, L: array[0..1000,0..1000] of integer;
    n, i, j: integer;
BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:= 1 to n do
        for j:= 1 to n do
            read(A[i,j]);
    for i:= 1 to n do
        for j:= 1 to n do
            L[i,j] := 0;
    L[1,1] := A[1,1];
    for i:=2 to n do
        begin
            L[1,i] := L[1,i-1] + A[1,i];
            L[i,1] := L[i-1,1] + A[i,1];
        end;
    for i:=2 to n do
        for j:=2 to n do
            begin
                if L[i-1,j] > L[i,j-1] then
                    L[i,j] := L[i-1,j] + A[i,j]
                else
                    L[i,j] := L[i,j-1] + A[i,j];
            end;
    write(L[n,n]);
    close(input);
    close(output);
END.
```

f) **TRIANGLE PASCAL** (*Tam giác Pascal*)

TRIANPAS.PAS

Thuật toán:

+ Ta xây dựng mảng hai chiều có kích thước [0..100, 0..101]

+ Sử dụng công thức như sau:

Dòng thứ i được tính thông qua dòng i-1

$$L[i, j] = L[i-1, j-1] + L[i-1, j]$$

+ **Thuật toán** cụ thể như sau:

$$L[0,1] = 1; L[1,1] = 1; L[1,2] = 1;$$

```

For i:= 2 to N do
Begin
    L[i, 1] :=1;
    For j:=2 to i+1 do
        L[i, j] = L[i-1, j-1] + L[i-1, j];
    End;
+ Kết quả được lưu trữ ở dòng N, cụ thể:
    L[N, 1], L[N, 2], L[N, 3], ..., L[N, N], L[N,N+1]

```

```

program Tam_giac_PASCAL;
Const
    fi = 'TRIANPAS.INP';
    fo = 'TRIANPAS.OUT';
var
    L: array[0..100,0..101] of integer;
    n, i, j: integer;

BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    L[0,1] := 1;
    L[1,1] := 1;
    L[1,2] := 1;
    for i:= 2 to n do
        begin
            L[i,1] := 1;
            for j:= 2 to n+1 do
                L[i,j] := L[i-1,j-1] + L[i-1,j];
            end;
        end;
    for i:=1 to n+1 do
        write(L[n,i], ' ');
    close(input);
    close(output);
END.

```

g) **TRIANGLE NUMBER (Tam giác số)**

TRIANNUM.PAS

Thuật toán:

- + Ta xây dựng mảng hai chiều có kích thước [1..100, 0..101]
- + Sử dụng công thức như sau:
 Dòng thứ i được tính thông qua dòng i-1

$$L[i, j] = \text{Max}(L[i-1, j-1] + L[i-1, j]) + A[i, j]$$
- + **Thuật toán** cụ thể như sau:

```

L[1, 1] = A[1, 1];
For i:= 2 to N do
  Begin
    For j:=1 to i do
      L[i, j] = Max(L[i-1, j-1] + L[i-1, j]) + A[i, j];
    End;
+ Kết quả được lưu trữ ở dòng N, cụ thể:
  Tong:= L[N, 1]
  For j:= 2 to N do
    if (L[N, j] > Tong) then
      Tong := L[N, j];

```

```

[■] TRIANNUM.pas
program Tam_giac_so;
Const
  fi = 'TRIANNUM.INP';
  fo = 'TRIANNUM.OUT';
var
  A, L: array[0..1000,0..1000] of integer;
  n, i, j: integer;
  tong: int64;
BEGIN
  assign(input,fi); reset(input);
  assign(output,fo); rewrite(output);
  readln(n);
  for i:= 1 to n do
    for j:= 1 to i do
      read(A[i,j]);
  for i:= 1 to n do
    for j:= 0 to n do
      L[i,j] := 0;
  L[1,1] := A[1,1];

```

```

for i:=2 to n do
begin
  for j:=1 to i do
  begin
    if L[i-1,j-1] > L[i-1,j] then
      L[i,j] := L[i-1,j-1] + A[i,j]
    else
      L[i,j] := L[i-1,j] + A[i,j];
  end;
end;
tong := L[n,1];
for j:= 2 to n do
  if (L[n,j] > tong) then
    tong := L[n,j];
write(tong);
close(input);
close(output);
END.

```

h) DÃY CON ĐƠN ĐIỀU TĂNG DÀI NHẤT

SEQ.PAS

Thuật toán:

Gọi dãy ban đầu là a_1, a_2, \dots, a_n đặt $D[i]$ là độ dài lớn nhất của dãy con tìm được từ dãy bắt đầu từ vị trí a_1 kết thúc ở a_i .

Như vậy ta có hệ thức quy hoạch động: $D[i] = \text{Max}(D[j] + 1)$ (với $0 \leq j \leq i-1$ và $D[i] > D[j]$)

$D[0] = 0$; $a[0] = -\text{maxint}$;

Ban đầu ta khởi tạo mảng D toàn các giá trị 0. Kết quả tìm được là $\text{Max}(D[i])$ với $1 \leq i \leq n$.

Để truy vết và in ra dãy con tăng dài nhất ta dùng mảng Truoc với $\text{Truoc}[i]$ là vị trí của phần tử trước phần tử i trong dãy con tối ưu đến vị trí i .

Chương trình:

```

Uses crt;
Const infile   = 'SEQ.INP';
      outfile   = 'SEQ.OUT';
      maxn      = 10000;
Type  M1       = Array [0..Maxn] of integer;
Var   a,d,Tr   : M1;
      n,m      : integer;
      f,g      : text;
procedure DocDl;
var i:integer;
begin
  Assign(f,infile);
  Reset(f);
  Readln(f,n);
  For i:=1 to n do

```

```

        Readln(f,a[i]);
    Close(f);
end;
Procedure ChuanBi;
begin
    fillchar(D,sizeof(D),0);
    fillchar(Tr,sizeof(Tr),0);
    A[0]:=-maxint;
end;
Procedure XuLi;
var i,j:integer;
begin
    For i:=1 to n do
        For j:=i-1 downto 0 do
            if (a[i]>a[j]) and (D[i]<D[j]+1) then
                begin
                    D[i]:=D[j]+1;
                    Tr[i]:=j;
                end;
        end;
end;
Procedure GhiKq;
var i,j,u,v,max,t:integer;
begin
    Assign(g,outfile);
    Rewrite(g);
    max:=0;
    for i:=1 to n do
        if max<D[i] then
            begin
                max:=d[i];
                V:=i;
            end;
    T:=1;
    While v<>0 do
        begin
            d[t]:=a[v];
            inc(t);
            v:=Tr[v];
        end;
    Writeln(g,max);

```

```

for i:=max downto 1 do
  Writeln(g,D[i]);
  Close(g);
end;

```

```

Begin
  Clrscr;
  DocDl;
  ChuanBi;
  XuLi;
  Ghikq;
End.

```

i) **DI CHUYỂN TỪ TÂY SANG ĐÔNG TAYDONG.PAS**

Thuật toán:

Theo cách di chuyển như vậy ta nhận thấy tới ô (i,j) có thể đi từ ô (i-1,j-1) (i,j-1) (i+1,j-1) và như vậy ta quy hoạch như sau.

Dùng C là bảng 2 chiều M*N và C[i,j] là chi phí min để di chuyển từ cột trái tới ô (i,j) như vậy chi phí min để di chuyển từ tây sang đông là giá trị nhỏ nhất của cột thứ n trong mảng C

$$C[i,j] := \text{Min}(C[i-1,j-1], C[i,j-1], C[i+1,j-1]) + A[i,j];$$

$$\text{CPMin} = \text{Min}(C[i,n]) \quad (1 \leq i \leq m)$$

Chương trình:

Chương trình dưới đây dựa theo **Thuật toán**: như vậy nhưng làm ngược lại tức là quy hoạch từ đông sang tây để tiện cho việc ghi kết quả và ta không khai báo thêm mảng C. Mảng C nói ở trên chỉ để lờ hướng dẫn được sáng sửa còn trong chương trình ta có thể quy hoạch ngay trên mảng dữ liệu.

```

uses crt;
const   infile      = 'taydong.inp';
        outfile     = 'taydong.out';
        maxmn       = 100;
        maxc        = 1001;

Type    m1          = array [0..maxmn+1,1..maxmn] of integer;

var     a            : m1;
        m,n,CPMin    : integer;
        f,g          : text;

```

```

procedure DocDI;
var i,j:integer;
begin
    Assign(f,infile);
    Reset(f);
    Readln(f,m,n);
    for i:=1 to m do
        begin
            for j:=1 to n do
                read(f,a[i,j]);
            readln(f);
        end;
    Close(f);
end;

procedure ChuanBi;
var i,j:integer;
begin
    for j:=1 to n do
        begin
            a[0,j]:=maxc;
            a[m+1,j]:=maxc;
        end;
end;

function Min(x,y,z:Integer):integer;
begin
    if x>y then x:=y;
    if x>z then x:=z;
    Min:=x;
end;

procedure XuLi;
var i,j:integer;
begin
    for j:=n-1 downto 1 do
        for i:=1 to m do
            A[i,j]:=Min(a[i-1,j+1],a[i,j+1],a[i+1,j+1])+A[i,j];

```



```

end;

procedure GhiKq;
var i,j,u,v,T:integer;
begin
    Assign(g,outfile);
    Rewrite(g);
    CPMIn:=A[1,1];i:=1;
    for u:=2 to m do
        if CPMIn>A[u,1] then
            begin
                CPMIn:=A[u,1];
                i:=u;
            end;
    Writeln(g,CPMIn);
    for j:=1 to n-1 do
        begin
            Writeln(g,i);
            T:= Min(a[i-1,j+1],a[i,j+1],a[i+1,j+1]);
            if T = a[i-1,j+1] then Dec(i)
            else if T = a[i+1,j+1] then Inc(i)
            end;
            Writeln(g,i);
        end;
    Close(g);
end;

Begin
    Clrscr;
    DocDl;
    ChuanBi;
    XuLi;
    GhiKq;
End.

```

Chuyên đề 8

ĐỒ THỊ

Nội dung 1:

CHUYÊN ĐỀ: LÝ THUYẾT ĐỒ THỊ CƠ BẢN

Trong toán học và tin học, đồ thị là đối tượng nghiên cứu cơ bản của lý thuyết đồ thị. Một cách không chính thức, đồ thị là một tập các đối tượng gọi là đỉnh nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh). Tùy theo ứng dụng mà một số cạnh có thể có hướng.

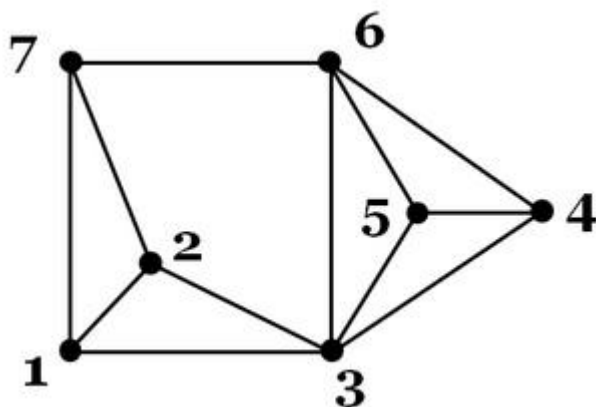
I. Sơ lược các khái niệm cơ bản về đồ thị

Một cách không chính thức, đồ thị là một tập các đối tượng được gọi là các đỉnh nối với nhau bởi các cạnh.

Một đồ thị kí hiệu là $G=(V,E)$

Trong đó:

- V là tập các đỉnh của đồ thị. Đặt $|V|=n$ (số đỉnh).
- E là tập các cạnh của đồ thị. Đặt $|E|=m$ (số cạnh).



Đỉnh:

Đỉnh biểu diễn các đối tượng trong đồ thị, thường được đánh dấu bằng các số hoặc kí hiệu bằng các chữ cái in thường u, v, \dots

Cạnh:

Cạnh nối đỉnh x với đỉnh y là một tập gồm hai phần tử (x,y) . (x,y) thường được vẽ dưới dạng một *đoạn thẳng* nối hai đỉnh.

Cạnh có hướng (cung):

Là một cặp đỉnh có thứ tự. Trong mỗi cặp có thứ tự đó, đỉnh thứ nhất được gọi là đỉnh đầu, đỉnh thứ hai là đỉnh cuối.

Cạnh vô hướng:

Không quan tâm đến hướng và coi hai đỉnh như nhau.

Khuyên:

Là một cạnh nối một đỉnh với chính nó.

Hai cạnh song song:

Là hai cạnh cùng nối hai đỉnh u, v .

Đồ thị có hướng:

Là đồ thị mà tất cả các cạnh trong đồ thị đều có hướng.

Đồ thị vô hướng:

Là đồ thị mà tất cả các cạnh trong đồ thị đều vô hướng.

Đơn đồ thị:

Là đồ thị không có khuyên và không có cạnh song song.

Đa đồ thị:

Là đồ thị không phải là đơn đồ thị.

Bậc:

Trong đồ thị vô hướng, bậc của đỉnh v trong đồ thị G , ký hiệu $d_G(v)$, là số cạnh liên thuộc với v , trong đó, khuyên được tính hai lần.

Ta có định lý:

Giả sử $G=(V,E)$ là đồ thị vô hướng, khi đó tổng các bậc đỉnh trong V sẽ bằng 2 lần số cạnh.

$$\sum_{v \in V} d_G(v) = m * 2$$

Hệ quả: Trong đồ thị vô hướng, số đỉnh bậc lẻ là chẵn.

Trong đồ thị có hướng, ta định nghĩa **bán bậc ra** của u là số cung đi ra khỏi nó, kí hiệu $d^+_G(u)$, **bán bậc vào** của u là số cung đi vào đỉnh đó, kí hiệu $d^-_G(u)$.

Giả sử $G=(V,E)$ là đồ thị có hướng, khi đó tổng các bán bậc vào bằng tổng các bán bậc ra và bằng số cung của đồ thị.

$$\sum_{v \in V} d^+_G(v) = \sum_{v \in V} d^-_G(v) = m$$

Đường đi và chu trình:

Một dãy các đỉnh $P = (p_0, p_1, \dots, p_k)$ sao cho $(p_{i-1}, p_i) \in E, \forall i: 1 \leq i \leq k$ được gọi là một đường đi.

Một đường đi là chu trình khi $p_0 = p_k$.

Liên thông:

Một đồ thị vô hướng là liên thông nếu tồn tại đường đi giữa hai cặp đỉnh bất kì thuộc đồ thị.

Một đồ thị có hướng là liên thông yếu nếu phiên bản vô hướng của đồ thị đó là liên thông.

II. Biểu diễn đồ thị trên máy tính:

Có nhiều cách để biểu diễn đồ thị trên máy tính, tùy thuộc vào tính chất của đồ thị hoặc **Thuật toán** áp dụng với đồ thị... Ta cũng có thể lưu kèm theo các thông tin như trọng số, giá trị phù hợp với từng cạnh. Dưới đây là một vài cách phổ biến.

1. Biểu diễn đồ thị theo ma trận kề:

Tạo một ma trận A kích thước $n \times n$ trong đó n là số đỉnh của đồ thị.

Nếu là đơn đồ thị, vô hướng không trọng số:

Ta gán $a[u][v] = 0$ nếu không có cạnh nối hai đỉnh u, v .

$a[u][v] = a[v][u] = 1$ nếu có cạnh nối hai đỉnh u, v .

Nếu đồ thị là đa đồ thị vô hướng không trọng số, tương tự, chỉ khác ta có thể gán $a[u][v]$ = số cạnh nối u và v .

Nếu đồ thị vô hướng có trọng số thì chúng ta thay $a[u][v]=1$ và $a[v][u]=1$ đó thành $a[u][v]=a[v][u]=c$ (Với c chính là trọng số của cạnh đó).

Nếu đồ thị có hướng, và dữ liệu đề bài nói rõ cạnh đi từ u đến v thì chỉ gán $a[u][v] = 1$ và ko gán chiều ngược lại là được.

Định nghĩa và gán tùy theo lập trình viên hiểu là vô hướng hay có hướng, đơn đồ thị hay đa đồ thị.

1. Ưu điểm:

- Để kiểm tra hai đỉnh u, v có kề nhau không, ta chỉ cần kiểm tra trong độ phức tạp $O(1)$.

2. Nhược điểm:

- Dù đồ thị có nhiều cạnh hay ít cạnh thì cũng phải mất $n*n$ ô nhớ để lưu.
- Để duyệt tất cả các đỉnh kề với u , ta phải duyệt tất cả các đỉnh $v \in V$ cho dù đỉnh u kề với ít hoặc không kề với đỉnh nào khác.

Biểu diễn bằng ma trận kề thường được dùng khi đồ thị có ít đỉnh, hoặc đồ thị dày, nhiều cạnh, hoặc **Thuật toán** để thao tác trên đồ thị yêu cầu.

Cài đặt:

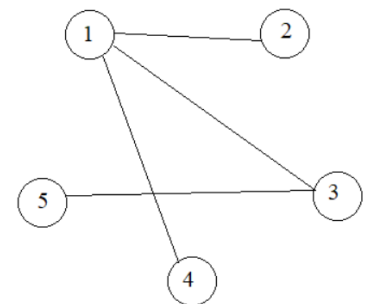
Cho đồ thị đơn vô hướng, dữ liệu lấy từ input, dòng đầu gồm 2 số n, m là số đỉnh và số cạnh của đồ thị, m dòng tiếp theo, mỗi dòng gồm 2 số u, v thể hiện cạnh nối giữa 2 đỉnh.

Ví dụ có $n = 5$ đỉnh, và có $m = 4$ cặp cạnh sau:

1 2
1 3
1 4
3 5

	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	0	0
3	1	0	0	0	1
4	1	0	0	0	0
5	0	0	1	0	0

Dữ liệu ma trận kề:



Nhập dữ liệu đồ thị theo ma trận kề:

```
1 #define NMAX 100
2
3 int a[NMAX+1][NMAX+1];
4 int n, m, u, v;
5
6 cin >> n >> m;
7
11 for (int i=1; i<=m; i++)
12 {
13     cin >> u >> v;
14     a[u][v]=1;
15     a[v][u]=1;
16 }
```

2. Biểu diễn đồ thị theo danh sách cạnh:

Với đồ thị $G=(V,E)$ có n đỉnh, m cạnh, ta có thể liệt kê tất cả các cạnh của đồ thị bằng một danh sách tương ứng, mỗi phần tử của mảng tương ứng là một cặp (u,v) là một cạnh thuộc E , tùy theo người lập trình định nghĩa là có hướng hay vô hướng.

1. Ưu điểm:

- Với đồ thị thưa, ta chỉ cần mất m (số lượng cạnh) ô nhớ để lưu đồ thị.

2. Nhược điểm:

- Khi cần kiểm tra hai đỉnh u,v có kề nhau hay không, ta không thể kiểm tra nhanh trong $O(1)$ như cách lưu bằng ma trận kề, mặc dù tùy theo cách lưu danh sách cạnh mà ta có thể kiểm tra trong $O(\log n)$ hoặc ít hơn.

Cài đặt:

Cho đồ thị đơn vô hướng, dữ liệu lấy từ input, dòng đầu gồm 2 số n, m là số đỉnh và số cạnh của đồ thị, m dòng tiếp theo, mỗi dòng gồm 2 số u, v thể hiện cạnh nối giữa 2 đỉnh.

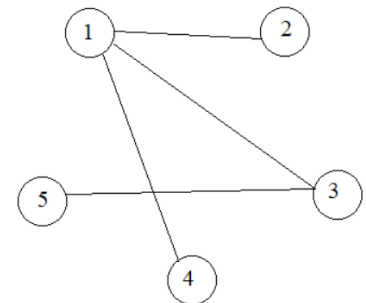
Ví dụ có $n = 5$ đỉnh, và có $m = 4$ cặp cạnh sau:

1 2

1 3

1 4

3 5



Nhập dữ liệu đồ thị theo danh sách cạnh:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int mmax = 100;
4 struct canh
5 {
6     int x, y;
7 };
8 canh a[mmax];
9 int n, m;
10 int main()
11 {
12     cin >> n >> m;
13     for(int i = 1; i <= m; i++)
14         cin >> a[i].x >> a[i].y;
15     return 0;
16 }
17
```

3. Biểu diễn đồ thị theo danh sách kề:

Với mỗi đỉnh của đồ thị, ta lưu một danh sách các đỉnh kề với đỉnh đó.

1. Ưu điểm:

- Với phương pháp này, việc duyệt tất cả các đỉnh kề với đỉnh u vô cùng dễ dàng.

2. Nhược điểm:

- Khi cần kiểm tra hai đỉnh u,v có kề nhau hay không, ta không thể kiểm tra nhanh trong $O(1)$ như cách lưu bằng ma trận kề, mặc dù tùy theo cách lưu danh sách cạnh mà ta có thể kiểm tra trong $O(\log n)$ hoặc ít hơn.

Tổ chức dữ liệu:

- Mảng động **vector** `<int> g[n]` lưu trữ danh sách các đỉnh kề (khai báo n vector, trong đó vector `g[i]` lưu trữ các đỉnh kề với đỉnh `i` ($1 \leq i \leq n$))

Cách nhập và in dữ liệu đồ thị vô hướng theo danh sách kề:

```
3  using namespace std;
4  const int nmax = 1001;
5  vector<int> g[nmax];
6  int n, m, u, v;
7  void nhap()
8  {
9      cin >> n >> m;
10     for(int i = 1; i<=m; i++)
11     {
12         cin >> u >> v;
13         g[u].push_back(v);
14         g[v].push_back(u);
15     }
16     return;
17 }
18 void in()
19 {
20     for(int i = 1; i<=n; i++)
21     {
22         for(int j = 0; j<g[i].size(); j++)
23             cout << g[i][j] << " ";
24         cout << endl;
25     }
26 }
27 int main()
28 {
29     nhap();
30     in();
31     return 0;
32 }
```

Test:

Input	Output
5 7	2 3
1 2	1 3 4 5
1 3	1 2 5
2 3	2 5
2 4	2 3 4
2 5	
3 5	
4 5	

III. Các Thuật toán tìm kiếm trên đồ thị

Bài toán duyệt tất cả các đỉnh có thể đến được từ một đỉnh xuất phát nào đó. Bài toán này đưa về bài toán liệt kê mà yêu cầu của nó là không được bỏ sót hay lặp lại bất kì đỉnh nào. **Thuật**

toán cho phép duyệt một cách có hệ thống các đỉnh gọi là những **Thuật toán** tìm kiếm trên đồ thị.

1. Thuật toán tìm kiếm theo chiều sâu (DFS)

DFS dùng để:

1. Tìm đường đi bất kì
2. Tìm đường đi có thứ tự từ điển nhỏ nhất từ gốc tới đỉnh
3. Sắp xếp topo
4. Kiểm tra tồn tại chu trình và tìm chu trình
5. Tìm thành phần liên thông mạnh
6. Tìm khớp và cầu đồ thị
7. Kiểm tra một đỉnh có phải là cha của đỉnh khác hay không?
8. Tìm cha chung gần nhất (LCA)

Ý tưởng của Thuật toán DFS:

Từ đỉnh (nút) gốc ban đầu, **Thuật toán** duyệt đi xa nhất theo từng nhánh, khi nhánh đã duyệt hết, lùi về từng đỉnh để tìm và duyệt những nhánh tiếp theo. Quá trình duyệt chỉ dừng lại khi tìm thấy đỉnh cần tìm hoặc tất cả đỉnh đều đã được duyệt qua.

Thuật toán:

1. Thăm đỉnh xuất phát - đỉnh u, đánh dấu đỉnh u.
2. Xét tất các đỉnh v kề với đỉnh hiện đang thăm
 - Nếu đỉnh v chưa được đánh dấu (chưa thăm), thăm đỉnh v như thăm u.
 - Nếu đỉnh v đã được đánh dấu, bỏ qua.

```
37 void dfs(int u)
38 {
39     cout << u << " "; //thăm u
40     visit[u] = true; //đánh dấu đã thăm u
41     for(int i = 0; i < g[u].size(); i++)
42     {
43         int v = g[u][i];
44         if (!visit[v]) //nếu v chưa thăm thì thăm v như thăm u
45             dfs(v);
46     }
47 }
```

2. Thuật toán tìm kiếm theo chiều rộng (BFS)

BFS là một **Thuật toán** tìm kiếm trong đồ thị, có thể dùng để tìm các thành phần liên thông, kiểm tra đồ thị hai phía.

Ý tưởng:

Thuật toán tìm kiếm theo chiều rộng BFS là **Thuật toán** tìm kiếm trong đồ thị bằng cách tìm kiếm dựa trên 2 thao tác chính là: cho trước một đỉnh của đồ thị và thêm các đỉnh kề với nó vào danh sách chờ duyệt.

Chúng ta sẽ xây dựng một danh sách chứa các đỉnh đang chờ duyệt, tại mỗi bước chúng ta thăm đỉnh ở đầu danh sách và thêm những đỉnh kề với nó chưa có trong danh sách chờ vào

cuối danh sách. Vì nguyên tắc đó nên chúng ta có thể tổ chức danh sách chờ đó bằng cấu trúc dữ liệu hàng đợi (Queue).

Thuật toán giả ngôn ngữ:

```
1 Free[u]=true; //với mọi u=1...n
2 Queue ban đầu rỗng.
3 Push(s); // đẩy đỉnh đầu tiên vào queue
4 Free[s]=false; // đánh dấu đỉnh s
5 while (not empty())
6 {
7     u = pop(); // lấy từ queue đỉnh u
8     for (v=1; v<=n; v++)
9         if ((tồn tại cạnh u,v) và Free[v]==true)
10        {
11            Free[v]=false; // đánh dấu đỉnh v
12            Push(v); // đẩy đỉnh v vào queue
13        }
14 }
```

Ví dụ: Viết chương trình ghi ra thứ tự duyệt BFS xuất phát từ đỉnh s. Đồ thị gồm n đỉnh, m cạnh vô hướng, các thành phần trên đồ thị liên thông với nhau.

Dữ liệu vào:

- Dòng đầu: gồm 3 số nguyên n, m, s ($1 \leq n, m \leq 100, 1 \leq s \leq n$)
- M dòng tiếp theo: mỗi dòng gồm 2 số u, v, mô tả 1 cạnh trong đồ thị

Dữ liệu ra:

- Gồm nhiều dòng, là thứ tự duyệt BFS

```
3 using namespace std;
4 const int nmax = 101;
5 vector<int> g[nmax];
6 vector<int> visit(nmax, false);
7 int n, m, u, v;
8 queue<int> q;
9 void nhap()
10 {
11     cin >> n >> m;
12     for(int i = 1; i<=m; i++)
13     {
14         cin >> u >> v;
15         g[u].push_back(v);
16         g[v].push_back(u);
17     }
18     return;
19 }
```



```

20 void BFS(int s)
21 {
22     q.push(s);
23     visit[s] = true;
24     while(!q.empty())
25     {
26         u = q.front();
27         q.pop();
28         cout << u << " ";
29         for(int i = 0; i < g[u].size(); i++)
30             if (!visit[g[u][i]])
31             {
32                 q.push(g[u][i]);
33                 visit[g[u][i]] = true;
34             }
35     }
36     return;
37 }
38
39 int main()
40 {
41     freopen("BFS.inp", "r", stdin);
42     freopen("BFS.out", "w", stdout);
43     nhap();
44     // in();
45     BFS(s);
46     return 0;
47 }

```

Test:

Input	Output
7 7 4	4 2 1 6 3 5 7
1 2	
1 3	
1 5	
2 4	
2 6	
3 7	
5 6	

Bài tập: Tính thời gian vào/ra các đỉnh trong đồ thị vô hướng liên thông $G = \langle V, E \rangle$

Input:

n đỉnh, m cạnh ($1 \leq n \leq 10^5$) ($1 \leq m \leq \min(10^5, n(n-1)/2)$)
m dòng, dòng j chứa 2 số x_j và y_j ($x_j \neq y_j, 1 \leq x_j, y_j \leq n$)
Không có hai cạnh nào trùng nhau.

Output:

Dòng 1: Thời gian vào các đỉnh i ($i = 1 \rightarrow n$) theo DFS.
Dòng 2: Thời gian ra các đỉnh i theo DFS.
Dòng 3: Thời gian vào các đỉnh i theo BFS.

Dòng 4: Thời gian ra các đỉnh i theo BFS.

Ví dụ:

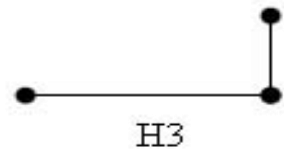
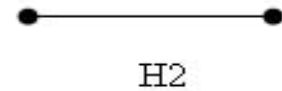
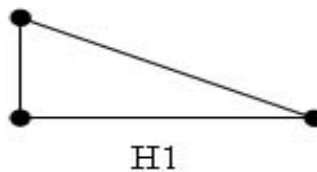
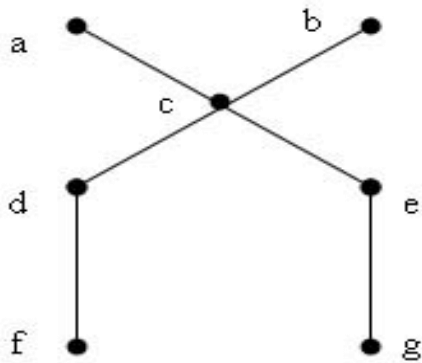
Input	Output
7 7 4	3 2 4 1 8 9 5
1 2	12 13 7 14 11 10 6
1 3	5 3 8 1 9 6 12
1 5	7 4 11 2 13 10 14
2 4	
2 6	
3 7	
5 6	

IV. Tính liên thông của đồ thị

II. Đồ thị liên thông:

Đồ thị vô hướng $G = (V, E)$ được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Ví dụ: Hình dưới đồ thị G là liên thông, còn đồ thị H là không liên thông.



Đồ thị G và H .

III. Thành phần liên thông:

Ta gọi đồ thị con của đồ thị $G = (V, E)$ là đồ thị $H = (W, F)$, trong đó $W \subseteq V$ và $F \subseteq E$.

Trong trường hợp đồ thị là không liên thông, nó sẽ rã ra thành một số đồ thị con liên thông đôi một không có đỉnh chung. Những đồ thị con liên thông như vậy ta sẽ gọi là các thành phần liên thông của đồ thị.

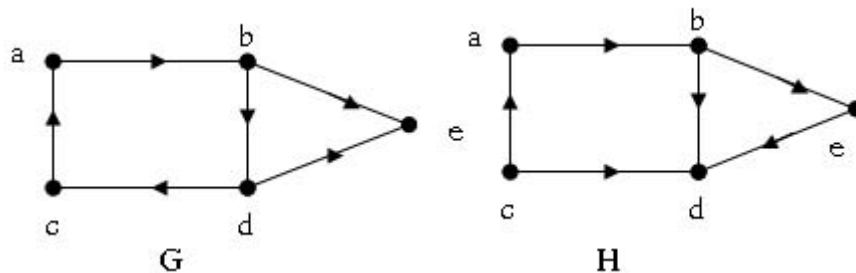
Ví dụ: Hình trên, đồ thị H gồm 3 thành phần liên thông H_1, H_2, H_3 .

Đồ thị liên thông mạnh:

Đồ thị có hướng G được gọi là liên thông mạnh, nếu luôn tồn tại đường đi (theo các cung định hướng) giữa hai đỉnh bất kỳ của đồ thị.

Đồ thị liên thông yếu:

Đồ thị có hướng G được gọi là liên thông yếu nếu phiên bản vô hướng của nó là đồ thị liên thông.



Ví dụ: Hình trên, đồ thị G là đồ thị liên thông mạnh, đồ thị H là đồ thị liên thông yếu.

IV. Xác định các thành phần liên thông trên đồ thị vô hướng

Bài toán: Hãy xác định các thành phần liên thông của đồ thị vô hướng $G = \langle V, E \rangle$

Ý tưởng: Bắt đầu từ một đỉnh bất kì, ta liệt kê những đỉnh đến được từ đỉnh đó vào một thành phần liên thông, sau đó loại bỏ tất cả các đỉnh đã liệt kê ra khỏi đồ thị và lặp lại, **Thuật toán** sẽ kết thúc khi tập đỉnh của đồ thị trở thành rỗng.

Thuật toán:

```
void scan( $u \in V$ )
{
    //Dùng DFS hoặc BFS liệt kê và đánh dấu những đỉnh có thể đến được từ u.
}
void coutlt()
{
    ct = 0;
    for ( $\forall u \in V$ )
        if <u chưa thăm>
        {
            ct++;
            //Thông báo thành phần liên thông thứ ct
            scan(u);
        }
}
```

Bài tập: Liệt kê các thành phần liên thông của đồ thị

Input:

Dòng 1: Chứa số đỉnh ($n \leq 200$) và số cạnh m của đồ thị.

m dòng tiếp theo, mỗi dòng chứa một cặp số u và v tương ứng với một cạnh (u,v)

Output:

Số thành phần liên thông của đồ thị. Các dòng tiếp theo, mỗi dòng liệt kê các đỉnh thuộc một thành phần liên thông.

Ví dụ:

Input	Output
12 10	3
1 4	1 4 5
2 3	2 3 6 7
3 6	8 9 10 11 12
4 5	
6 7	
8 9	
8 10	
9 11	
11 8	
11 12	

Chương trình tham khảo:

```
5  const int nmax = 205;
6  vector <int> g[nmax], lt[nmax];
7  vector <bool> visit(nmax, false);
8  int n, m, u, v, ct = 0;
9  void nhap();
10
11 void dfs(int u)
12 {
13     visit[u] = true;
14     lt[ct].push_back(u);
15     //vecto lt[ct] chứa các đỉnh thuộc thành phần liên thông thứ ct
16     for(auto &i: g[u])
17         if(!visit[i])
18             dfs(i);
19     return;
20 }
21
22 void coutlt()
23 {
24     for (int i = 1; i<=n; i++)
25         if (!visit[i])
26         {
27             ct++;
28             dfs(i);
29         }
30     cout << ct << endl;
31     for(int i = 1; i<=ct; i++)
32     {
33         sort(lt[i].begin(), lt[i].end());
34         //Sắp xếp lại các đỉnh trong vectơ lt[i]
35         for(auto &j: lt[i])
36             cout << j << " ";
37         cout << endl;
38     }
39     return;
40 }
41
42 int main()
43 {
44     freopen("tplt.inp", "r", stdin);
45     freopen("tptl.out", "w", stdout);
46     nhap();
47     coutlt();
48     return 0;
49 }
```

BÀI TẬP:

Bài 1. Chú bò hư hỏng (BCDAISY)

Đề bài:

Nông dân John có N ($1 \leq N \leq 250$) con bò đánh số từ $1..N$ chơi trên bãi cỏ. Để tránh bị lạc mất các con bò, mỗi con bò có thể được nối với một số con bò khác bằng dây thừng. Có tất cả M ($1 \leq M \leq N*(N-1)/2$) dây thừng nối các con bò. Tất nhiên, không có 2 con bò mà có nhiều hơn 1 dây thừng nối giữa chúng. Dữ liệu cho biết mỗi cặp con bò $c1$ và $c2$ là nối với nhau ($1 \leq c1 \leq N; 1 \leq c2 \leq N; c1 \neq c2$).

Nông dân John buộc cố định con bò 1 bằng sợi dây xích. Các con bò khác phải nối với con bò 1 bằng một số sợi dây thừng. Tuy nhiên, một số con bò hư hỏng không như vậy. Hãy giúp nông dân John tìm các con bò hư hỏng đó (không kết nối tới bò 1). Dĩ nhiên, con bò thứ 1 luôn nối tới chính nó.

Input:

Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và M

Dòng 2 đến dòng M+1: Dòng i+1 cho biết 2 con bò nối với nhau bằng sợi dây thứ i là c1 và c2 cách nhau bởi dấu cách.

Output:

Nếu không có con bò hư hỏng, in ra 0. Ngược lại, in ra trên mỗi dòng 1 số nguyên là thứ tự con bò hư hỏng theo thứ tự tăng dần.

Ví dụ:

Input	Output
6 4	4
1 3	5
2 3	6
1 2	
4 5	

Bài 2. Robin C11BC2

Đề bài:

Một ngày đẹp trời nọ, trên vương quốc của các Coders 2011, bỗng xuất hiện 1 lão phù thủy độc ác, lão phù thủy sirDat_LS đã có âm mưu thôn tính đất nước của đức vua vodanh9x. Lão phù thủy này rất yêu con gái của đức vua là Rose và đã bắt Rose về nơi ở của lão ta. Đức vua vodanh9x liền tìm hiệp sĩ Robin và sẽ hứa gả con gái cho Robin nếu chàng cứu được công chúa Rose trở về. Lão phù thủy sirDat_LS độc ác với khuôn mặt rất ghê tởm khiến công chúa mỗi khi nhìn thấy hắn thì công chúa lại ngất đi. Và rồi, chàng Robin của chúng ta đã tìm được đến nơi ở của lão phù thủy. Nơi ở của lão là 1 mê cung có N phòng, và N phòng này liên thông với nhau và có đúng N-1 đường đi (coi mỗi đường đi là 1 cạnh). Nhưng khó khăn thay, lão phù thủy đã đánh số mỗi đường đi là 1 hoặc 2. Nếu chàng Robin muốn đến cứu công chúa, thì từ nơi xuất phát đến nơi có công chúa phải có ít nhất một đường đi được đánh số 2, nếu không chàng Robin sẽ chết. Yêu cầu: Cho m truy vấn ($m \leq 10^5$) mỗi truy vấn có dạng (x,y), trong đó x là nơi xuất phát của Robin và y là nơi nhốt công chúa. Xác định đường đi ngắn nhất từ x đến y có cạnh có trọng số 2 hay không.

Input:

- Dòng đầu là số nguyên N ($N \leq 10^4$) – số đỉnh của đồ thị và M – số truy vấn.
- Từ dòng 2 đến dòng N: dòng thứ i chứa 2 số nguyên dương x ($x < i$) và k ($k \leq 2$) nghĩa là có cạnh nối giữa i và x và được đánh số là k.
- M dòng sau: mỗi dòng chứa 2 số x và y (Biểu thị cho truy vấn (x,y)).

Output: Với mỗi truy vấn, xuất ra “YES” nếu tồn tại đường đi có ít nhất 1 cạnh có trọng số 2, ngược lại xuất ra “NO”.

Ví dụ:

Input	Ouput
6 7	YES
1 1	YES
1 2	NO
3 1	NO
1 2	NO
5 2	YES
1 3	NO
5 1	
2 1	
2 1	
1 2	
2 4	
1 2	

Bài 3. Ốc sên ăn rau (OCSE)

Đề bài

Có một khu vườn hình chữ nhật kích thước $n \times m$ ô vuông (n dòng, m cột). Ta đánh số các dòng từ 1 đến n theo chiều từ trên xuống dưới, các cột từ 1 đến m theo chiều từ trái qua phải. Tại những ô vuông là đất bình thường người ta trồng rau. Tuy nhiên có một số ô là đá nên không trồng rau được. Có một chú ốc sên tại ô (y, x) , y là vị trí dòng, x là vị trí cột. Từ một ô, chú ốc sên chỉ có thể di chuyển sang 4 ô liền kề $(y-1, x)$, $(y+1, x)$, $(y, x-1)$, $(y, x+1)$. Nếu gặp ô đá thì ốc sên không đi vào được.

			S	

Ốc sên đang rất đói. Bạn hãy xác định xem chú có thể ăn được số lượng rau nhiều nhất là bao nhiêu.

Dữ liệu vào: gồm các dòng sau:

- Dòng thứ nhất gồm bốn số nguyên n, m, y, x , mỗi số cách nhau một khoảng trắng ($1 \leq y \leq n \leq 100, 1 \leq x \leq m \leq 100$).

- Trong n dòng tiếp theo, mỗi dòng gồm m số nguyên 0 hoặc 1 biểu thị vườn rau, mỗi số cách nhau một khoảng trắng. Số 0 nghĩa là ô rau, còn số 1 nghĩa là ô đá.

(Dữ liệu cho đảm bảo ô (y, x) là ô rau)

Dữ liệu xuất:

- Là một số nguyên xác định số lượng ô lớn nhất mà ốc sên có thể di chuyển đến.

Ví dụ:

input
4 5 2 4
0 0 1 0 0
0 1 0 0 1
1 0 0 0 0
0 1 0 0 1
output
10

Bài 4. Đếm ao (BCLKCOUN)

Đề bài

Sau khi kết thúc OLP Tin Học SV, một số OLP-er quyết định đầu tư thuê đất để trồng rau. Mảnh đất thuê là một hình chữ nhật $N \times M$ ($1 \leq N \leq 100$; $1 \leq M \leq 100$) ô đất hình vuông. Nhưng chỉ sau vài ngày, trận lụt khủng khiếp đã diễn ra làm một số ô đất bị ngập. Mảnh đất biến thành một số các ào. Các OLP-er quyết định chuyển sang nuôi cá. Vấn đề lại nảy sinh, các OLP-er muốn biết mảnh đất chia thành bao nhiêu cái ào để có thể tính toán nuôi trồng hợp lý. Bạn hãy giúp các bạn ý nhé.

Chú ý: Ao là gồm một số ô đất bị ngập có chung đỉnh. Để nhận thấy là một ô đất có thể có tối đa 8 ô chung đỉnh.

Input:

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và M
- Dòng 2.. $N+1$: M kí tự liên tiếp nhau mỗi dòng đại diện cho 1 hàng các ô đất. Mỗi kí tự là 'W' hoặc '.' tương ứng với ô đất đã bị ngập và ô đất vẫn còn nguyên.

Output:

- 1 dòng chứa 1 số nguyên duy nhất là số ao tạo thành.

Ví dụ:

Input	Output
10 12 W.....WW. .WWW.....WWWWW...WW.WW.W.. ..W.....W.. .W.W.....WW. W.W.W.....W. .W.W.....W. ..W.....W.	3

Bài 5. Bảo vệ nông trang (NKGUARD)

Đề bài:

Nông trang có rất nhiều ngọn đồi núi, để bảo vệ nông trang nông dân John muốn đặt người canh gác trên các ngọn đồi này. Anh ta băn khoăn không biết sẽ cần bao nhiêu người canh

gác nếu như anh ta muốn đặt 1 người canh gác trên đỉnh của mỗi đồi. Anh ta có bản đồ của nông trang là một ma trận gồm N ($1 < N \leq 700$) hàng và M ($1 < M \leq 700$) cột. Mỗi phần tử của ma trận là độ cao H_{ij} so với mặt nước biển ($0 \leq H_{ij} \leq 10,000$) của ô (i, j) . Hãy giúp anh ta xác định số lượng đỉnh đồi trên bản đồ. Đỉnh đồi là 1 hoặc nhiều ô nằm kề nhau của ma trận có cùng độ cao được bao quanh bởi cạnh của bản đồ hoặc bởi các ô có độ cao nhỏ hơn. Hai ô gọi là kề nhau nếu độ chênh lệch giữa tọa độ X không quá 1 và chênh lệch tọa độ Y không quá 1.

Input:

- Dòng 1: Hai số nguyên cách nhau bởi dấu cách: N và M
- Dòng 2.. $N+1$: Dòng $i+1$ mô tả hàng i của ma trận với M số nguyên cách nhau bởi dấu cách: H_{ij}

Output:

- Dòng 1: Một số nguyên duy nhất là số lượng đỉnh đồi.

Ví dụ:

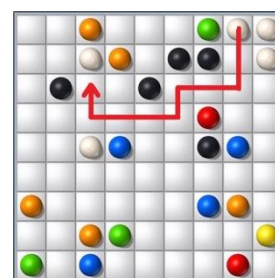
Input	Output
<pre> 8 7 4 3 2 2 1 0 1 3 3 3 2 1 0 1 2 2 2 2 1 0 0 2 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 2 2 1 1 0 0 1 1 1 2 1 0 </pre>	3

Bài 6. Trò chơi Lines (LINES)

Đề bài

Trò chơi Line là trò chơi di chuyển các viên bi trong một hình vuông 9×9 ô. Bi được ăn khi tạo thành các hàng, cột, đường chéo gồm 5 viên bi liên tiếp cùng màu.

Một **Thuật toán** được sử dụng trong trò chơi là tìm đường đi để di chuyển một viên bi. Giả sử trò chơi Line tổng quát có n dòng, n cột. Đánh số các dòng từ 1 đến n theo thứ tự từ trên xuống dưới, đánh số các cột từ 1 đến n theo thứ tự từ trái sang phải. Giả sử có một viên bi tại ô (y, x) - dòng y cột x , bi có thể di chuyển đến 1 trong 4 ô $(y+1, x)$, $(y-1, x)$, $(y, x+1)$, $(y, x-1)$, nếu ô đó đang trống. Cho vị trí bắt đầu và vị trí kết thúc của viên bi, hãy viết chương trình xác định xem có tồn tại đường đi để di chuyển viên bi hay không.



Input: gồm các dòng sau

- Dòng thứ nhất gồm năm số n , sy , sx , dy , dx , mỗi số cách nhau một khoảng trắng ($2 \leq n \leq 9$; $1 \leq sy, sx, dy, dx \leq n$). sy là chỉ số dòng, sx là chỉ số cột của viên bi cần di chuyển. dy là chỉ số dòng, dx là chỉ số cột của vị trí cần di chuyển viên bi đến.
- Trong n dòng tiếp theo, mỗi dòng gồm n số nguyên 0 hoặc 1, mỗi số cách nhau một khoảng trắng, biểu thị tình trạng của trò chơi. Số 1 nghĩa là vị trí ô đó có bi, số 0 nghĩa là vị trí ô đó đang trống.

(Dữ liệu cho bảo đảm tại ô (sy, sx) có giá trị là 1, tại ô (dy, dx) có giá trị là 0)

Output:

- Nếu tìm được đường đi, in ra YES.
- Nếu không tìm được đường đi, in ra NO.

Ví dụ:

Input	Output
2 1 1 2 2 1 0 1 0	YES
2 1 1 2 2 1 1 1 0	NO
3 1 1 3 3 1 0 1 1 0 0 1 1 0	YES

Bài 7. Kết nối (CONNECT) – Trại hè HV 2015 – K11

Đề bài

Lên LS tham dự trại hè HV, Sơn Tùng gặp lại cô bạn cùng ôn thi đội tuyển năm ngoái. Sau khi hàn huyên đủ thứ, cô bạn muốn Sơn Tùng trợ giúp về vấn đề đang gặp phải.

Tỉnh LS có N thành phố, được đánh số từ 1 đến N . Hai thành phố i và j ($1 \leq i, j \leq N$) có thể có nhiều nhất một con đường tỉnh lộ hai chiều nối với nhau. Ủy ban nhân dân tỉnh LS quyết định mở thêm một con đường mới nối trực tiếp giữa hai thành phố bất kỳ nào đó trong N thành phố và xây dựng một sân vận động tại một thành phố nào đó với tiêu chuẩn Olympic để tạo điều kiện cho nhân dân luyện tập và thi đấu thể thao.

Cô bạn nhờ Sơn Tùng tính toán xem sân vận động này có thể kết nối nhiều nhất là bao nhiêu thành phố với nhau, biết rằng thành phố định xây sân vận động và những thành phố khác đều có đường đi (trực tiếp hoặc gián tiếp) đến để luyện tập và thi đấu thể thao.

Dữ liệu:

- Dòng đầu ghi hai số nguyên N – số thành phố và M – số đường tỉnh lộ nối giữa hai thành phố với nhau.
- M dòng sau, mỗi dòng ghi hai số nguyên dương i và j thể hiện thành phố i có đường tỉnh lộ nối với thành phố j .

Kết quả: Ghi số nguyên dương là số thành phố lớn nhất mà người dân tại đó có thể tới luyện tập và thi đấu thể thao.

Ví dụ:

Input	output
10 6 1 2 5 4 6 7	7

10 8	
7 8	
3 4	

Ràng buộc: $1 \leq N \leq 1000$, $0 \leq M \leq 10000$, $1 \leq i, j \leq N$

Bài 8. Nước biển (BCISLAND)

Đề bài:

Trái đất nóng lên kéo theo mực nước biển dâng. Hòn đảo nhỏ Gonnàsinkà thuê bạn để dự báo trước hiểm họa này. Cho trước 1 lưới tọa độ thể hiện cao độ của đảo, hãy giúp họ tính toán xem nước biển dâng cao bao nhiêu thì hòn đảo sẽ bị chia cắt.

Input:

Input gồm nhiều bộ test, mỗi bộ test bao gồm:

- Dòng đầu ghi 2 số n, m là chiều dài và chiều rộng.
- Sau đó là n dòng, mỗi dòng gồm m số, mỗi số cho biết độ cao của ô đó, giá trị 0 chỉ mực nước biển. Những ô giá trị 0 dọc theo đường viền và những ô số 0 liền kề những ô này chỉ mặt biển. Những ô có giá trị 0 còn lại (được bao bọc bởi các số > 0) là đất liền bên trong đảo nhưng có độ cao ngang mặt nước biển. Hòn đảo lúc đầu chưa bị chia cắt. Số n và m không lớn hơn 100 và độ cao không lớn hơn 1000.
- Dòng cuối cùng của input chứa 2 số 0

Output:

Với mỗi bộ test, in ra:

Case n : Island splits when ocean raises f feet. (Đảo bị chia khi nước biển dâng cao f feet)

Hoặc:

Case n : Island never splits. (Đảo không bao giờ bị chia cắt)

Ví dụ:

Input	Output
5 5 3 4 3 0 0 3 5 5 4 3 2 5 4 4 3 1 3 0 0 0 1 2 1 0 0 5 5 5 5 5 5 7 4 1 1 1 4 4 1 2 1 3 7 1 0 0 4 7 3 4 4 4 0 0	Case 1: Island never splits. Case 2: Island splits when ocean rises 3 feet.

Bài 9. Tính toán lượng nước (PBCWATER)

Đề bài:

Nền phẳng của 1 công trình xây dựng được chia thành lưới ô vuông đơn vị kích thước $M \times N$ ô. Trên mỗi ô (i, j) của lưới, người ta dựng 1 cột bê tông hình hộp có đáy là ô (i, j) và chiều cao là $h[i, j]$ đơn vị. Sau khi dựng xong thì trời đổ mưa to và đủ lâu. Nhà thầu xây dựng muốn tính lượng nước đọng lại giữa các cột để có kế hoạch thi công tiếp theo. Giả thiết, nước ko thấm thấu qua các cột bê tông cũng như ko rò rỉ qua các đường ghép giữa chúng.

Nhiệm vụ của bạn là giúp nhà thầu tính toán lượng nước đọng lại giữa các cột.

Input:

- Dòng đầu tiên ghi 2 số nguyên dương M và N
- Dòng thứ i trong M dòng tiếp theo, ghi N số nguyên dương $h[i,1], h[i,2] \dots h[i,N]$.

Output

- 1 dòng duy nhất chứa số đơn vị khối nước đọng lại.

Ví dụ:

Input:

```
5 5
9 9 9 9 9
9 2 2 2 9
9 2 5 2 9
9 2 2 2 9
9 9 9 9 9
```

Output:

```
60
```

Giới hạn:

$1 \leq M, N \leq 100, 1 \leq H[i,j] \leq 1000$

V. Bài tập tự giải

Bài 10. Bãi cỏ ngon nhất (VBGRASS) vnoi.info

Bài 11. (Tham khảo) UVa 00260 - Il Gioco dell'X

Bài 12. (Tham khảo) UVa 00469 - Wetlands of Florida

Bài 13. (Tham khảo) UVa 00572 - Oil Deposits

Bài 14. (Tham khảo) UVa 00785 - Grid Colouring

Hướng dẫn làm bài:

a) **Bài 1. Chú bò hư hỏng (BCDAISY)**

Thuật toán

Dùng DFS liệt kê các đỉnh không thuộc thành phần liên thông với đỉnh có số hiệu 1

b) **Bài 2. Robin C11BC2**

Thuật toán:

– Chỉ đọc các cạnh 1 vào đồ thị, còn các cạnh 2 ta sẽ bỏ.

– Xây dựng mảng $DD[i]$ là chỉ số vùng liên thông của đỉnh i. Xây dựng mảng này bằng

Thuật toán BFS hoặc DFS.

– Trả lời các truy vấn: Nếu $DD[x] \neq DD[y]$ thì kết quả là YES (bởi vì khi ta bỏ cạnh 2 ra, mà ko thể đi từ x đến y thì dễ dàng suy ra đồ thị ban đầu nếu đi từ x đến y sẽ đi qua cạnh 2). Ngược lại là NO.

c) **Bài 3. Ốc sên ăn rau (OCSE)**

Thuật toán

Dùng DFS loang từ vị trí đứng của con sên, trong quá trình loang đếm số lượng ô là rau mà con sên đi qua.

d) Bài 4. Đếm ao (BCLKCOUN)

Thuật toán: Tiến hành dùng DFS để đếm số lượng các vùng chứa kí tự 'W'.

e) Bài 5. Bảo vệ nông trang (NKGUARD)

Thuật toán

Từ 1 ô chưa duyệt, ta đi đến những ô có cùng độ cao với nó và đánh dấu lại. Nếu gặp phải một ô có độ cao lớn hơn thì đánh dấu đây không phải là đỉnh đồi.

f) Bài 6. Trò chơi Lines (LINES)

Dùng DFS loang trên ma trận.

g) Bài 7. Kết nối

Dùng DFS đếm số lượng đỉnh của mỗi thành phần liên thông. Con đường mới sẽ nối hai thành phần liên thông có số lượng đỉnh nhiều nhất. Tổng hai thành phần liên thông nhiều nhất chính là kết quả cần tìm.

h) Bài 8. Nước biển (BCISLAND)

Dùng DFS đánh dấu các ô có độ cao $\leq L$ ta được mảng đánh dấu d; với mỗi độ cao x, đếm số lượng TPLT là các đỉnh có độ cao $> x$, nếu số lượng TLPT > 1 thì đảo bị chia cắt, lúc này in ra kết quả và thoát khỏi chương trình.

i) Bài 9. Tính toán lượng nước (PBCWATER)

NỘI DUNG 2

ĐƯỜNG ĐI VÀ CHU TRÌNH

I. Nhắc lại kiến thức

1. * Khái niệm “Đường đi và chu trình”

Một dãy các đỉnh $P = (p_0, p_1, \dots, p_k)$ sao cho $(P_{i-1}, P_i) \in E, \forall i: 1 \leq i \leq k$ được gọi là một đường đi.

Một đường đi là chu trình khi $p_0 = p_k$.

2. * Bài toán tìm đường đi

Cho đồ thị $G = (V, E)$ và hai đỉnh $s, t \in V$.

Một dãy các đỉnh $P = \langle s = p_0, p_1, \dots, p_k = t \rangle (P_{i-1}, P_i) \in E, \forall i: 1 \leq i \leq k$ được gọi là một đường đi từ s tới t , gồm $k+1$ đỉnh và k cạnh. Đỉnh s được gọi là đỉnh đầu, đỉnh t được gọi là đỉnh cuối của đường đi.

Nếu tồn tại một đường đi từ s đến t , ta nói s đến được t và t đến được s .

3. * Hướng giải quyết

Dùng **Thuật toán** tìm kiếm theo chiều sâu DFS hoặc tìm kiếm theo chiều rộng BFS.

II. Ví dụ “Tìm đường đi”

Cho đồ thị có hướng $G = (V, E)$, số đỉnh không quá 10^5 , số cung không quá 10^6 , các đỉnh được đánh số từ 1 đến n .

Input:

- Dòng 1 chứa số đỉnh n , đỉnh xuất phát s và đỉnh cần đến t .
- m dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cung (i, j) của đồ thị, ngoài ra có thêm một số 0 ở cuối dòng để báo hiệu kết thúc.

Output:

- Danh sách các đỉnh có thể đến được từ s .
- Đường đi từ s tới t nếu có (Nếu không có in ra -1)

Input	Output
8 1 5	Danh sach cac dinh den duoc tu 1: 1, 2, 3, 5, 4, 6
2 3 0	Duong di tu 1 toi 5 la: 1 – 2 – 3 – 5
3 4 0	
1 5 0	
6 0	
0	
2 0	
8 0	
0	

Hướng dẫn:

1. Sử dụng DFS/BFS từ đỉnh s để liệt kê các đỉnh đến được từ s .
2. Vì phải liệt kê các đỉnh trên đường đi từ s đến t nên ta phải lưu vết đường đi.

III. Ví dụ 2 “Kiểm tra tồn tại chu trình trong đồ thị”

Xét đồ thị có hướng G không chứa cạnh song song, không có khuyên. Cần xác định đồ thị có chứa chu trình hay không. Nếu tồn tại chu trình thì chỉ ra một chu trình tùy chọn.

Bài toán: Xét đồ thị có hướng n đỉnh và m cạnh, cạnh thứ i được xác định bởi cặp số nguyên (a_i, b_i) cho biết tồn tại cạnh $a_i \rightarrow b_i, i = 1 \div m$.

Hãy xác định đồ thị chứa chu trình hay không và đưa ra thông báo “NO” nếu không có chu trình. Nếu tồn tại chu trình đưa ra thông báo “YES” và ở dòng tiếp theo ghi ra dãy các đỉnh xác định một chu trình tùy chọn.

Input:

- Dòng đầu tiên chứa 2 số nguyên n và $m, n \leq 10^5, m \leq 2 \cdot 10^5$
- Dòng thứ i trong m dòng sau chứa hai số nguyên $a_i, b_i (a_i \neq b_i, 1 \leq a_i, b_i \leq n)$

Output:

- Đưa ra kết quả theo quy cách đã nêu ở trên.

Input	Output
10 10	YES
1 2	2 4 3 6 10 9 2
2 4	
4 3	
3 6	
3 8	
4 5	
4 7	
9 2	
6 10	
10 9	

Hướng dẫn:

1. Duyệt đồ thị theo chiều sâu DFS.
2. Khi tới một đỉnh, gán cho đỉnh đó màu 1, khi ra khỏi đỉnh đó gán cho nó màu 2.
3. Nếu trong quá trình duyệt gặp lại đỉnh màu 1 nghĩa là tồn tại chu trình.
4. Cần lưu trữ danh sách các đỉnh đã đi qua trong mỗi lần duyệt để dẫn xuất chu trình nếu có.

ĐƯỜNG ĐI NGẮN NHẤT TRÊN ĐỒ THỊ CÓ TRỌNG SỐ.

Bài toán:

Trong lý thuyết đồ thị, bài toán đường đi ngắn nhất giữa hai đỉnh cho trước là bài toán tìm một đường đi giữa chúng sao cho tổng các trọng số của các cạnh tạo nên đường đi đó là nhỏ nhất.

Định nghĩa một cách hình thức, cho trước một đồ thị có trọng số $G=(V,E,w)$. Cho trước một đỉnh u thuộc V , tìm một đường đi P từ u tới một đỉnh v thuộc V sao cho: $\sum_{p \in P} w(p)$ nhỏ nhất trong tất cả các đường đi từ u tới v .

Các **Thuật toán** thường được dùng để giải quyết:

Thuật toán Dijkstra - giải bài toán bài toán đường đi ngắn nhất giữa hai đỉnh cho trước nếu tất cả các trọng số đều không âm. **Thuật toán** này có thể tính toán tất cả các đường đi ngắn nhất từ một đỉnh xuất phát cho trước s tới mọi đỉnh khác mà không làm tăng thời gian chạy.

Thuật toán Bellman-Ford - giải bài toán bài toán đường đi ngắn nhất giữa hai đỉnh cho trước trong trường hợp trọng số có thể có giá trị âm.

Thuật toán Floyd-Warshall - giải bài toán đường đi ngắn nhất cho mọi cặp đỉnh.

Thuật toán Johnson - giải bài toán đường đi ngắn nhất cho mọi cặp đỉnh, có thể nhanh hơn **Thuật toán** Floyd-Warshall trên các đồ thị thưa.

Chú ý:

Ta có đường đi $P = \{v_1, v_2, \dots, v_k\}$ là một đường đi ngắn nhất từ v_1 tới v_k . Khi đó ta có nhận xét, đường đi từ v_i tới v_j qua $\{v_i, v_{i+1}, \dots, v_j\}$ với $i, j \in [1, k]$ là một đường đi ngắn nhất từ v_i tới v_j .

I. Thuật toán dijkstra

Trong trường hợp đồ thị có trọng số trên các cạnh không âm, ta có **Thuật toán** Dijkstra để tìm đường đi ngắn nhất từ đỉnh xuất phát s tới các đỉnh khác của đồ thị.

- **Bài toán:** Cho đồ thị n đỉnh và m cạnh. Đồ thị có thể có hướng hoặc không có hướng. Trọng số mỗi cạnh là không âm. Hãy xác định đường đi ngắn nhất từ đỉnh s cho trước tới mỗi đỉnh còn lại và độ dài của đường đi đó.

Input:

- Dòng đầu tiên chứa 2 số nguyên n và m .
- Mỗi dòng trong m dòng tiếp theo chứa 3 số nguyên a, b, r cho biết cạnh nối từ a tới b có trọng số là r . (không có hai dòng nào giống nhau)
- Dòng $m+2$ chứa số nguyên s
- Dòng cuối cùng chứa số nguyên k và sau đó là k số nguyên c_1, c_2, \dots, c_k cho biết phải dẫn xuất đường đi ngắn nhất từ s tới c_i

Output:

- Dòng đầu tiên chứa n số nguyên, số thứ i là độ dài đường đi ngắn nhất từ s đến đỉnh i . Độ dài bằng -1 nếu không tồn tại đường đi từ s đến đỉnh đó.
- Dòng thứ i trong k dòng sau chứa thông tin về đường đi ngắn nhất (theo quy cách đã nêu phía trên)

Input	Output
6 9	8 0 7 5 3 5
1 2 10	Duong đi tu 2 toi 1: 2 4 3 1
1 3 1	Duong đi tu 2 den 6: 2 5 6
2 4 5	
3 4 2	
2 5 3	
3 6 15	
4 6 10	
4 5 9	
5 6 2	
2	
2 1 6	

Tổ chức dữ liệu và **Thuật toán**:

Ta có mảng $kc[u]$ là khoảng cách ngắn nhất từ đỉnh s tới đỉnh u trên đồ thị. Ban đầu $kc[s] = 0$, các giá trị khác bằng dương vô cực. Mảng logic $cl[]$ để đánh dấu, $cl[u]$ cho biết đỉnh u đã được xét hay chưa. Ban đầu $cl[u] = \text{false}$ ($u = 1 \div n$).

Ở mỗi bước, ta sẽ lấy đỉnh u có $kc[u]$ nhỏ nhất trong các đỉnh chưa được đánh dấu vào thời điểm hiện tại:

$$kc[u] = \min\{kc[i] \mid cl[i] = \text{false}, i = 1 \div n\}$$

và sử dụng khoảng cách đó để cập nhật khoảng cách ngắn nhất của các đỉnh v xung quanh u .

Gọi lt là trọng số của cạnh (u,v) . Khi đó $kc[v]$ được cập nhật theo công thức sau:

$$kc[v] = \min(kc[v], kc[u] + lt)$$

Sau n bước, tất cả các đỉnh đều được đánh dấu, $kc[v]$ chính là đường đi ngắn nhất từ s đến v . Nếu không tồn tại đường đi từ s đến v thì $kc[v] = +\infty$.

Để khôi phục đường đi có độ dài ngắn nhất cần tổ chức mảng $P = (p_1, p_2, \dots, p_n)$ trong đó $p[v]$ lưu đỉnh cuối cùng trước đỉnh v trong đường đi ngắn nhất từ s đến v . Mỗi lần $kc[v]$ thay đổi giá trị thì đỉnh đạt min: $p[v] = u$.

Lưu ý:

Việc tìm $kc[u]$ nhỏ nhất được thực hiện dễ dàng thông qua hàng đợi ưu tiên `Priority_queue` trong thư viện STL của C++.

Để sử dụng được `Priority_queue`, mỗi đơn vị dữ liệu input cần được tổ chức dưới dạng một cặp số nguyên (`pair<int,int>`), phần tử thứ nhất là trọng số và phần tử thứ 2 là đỉnh của cạnh.

Khai báo để lấy giá trị max ở đầu hàng đợi ưu tiên:

```
priority_queue < pair < int, int > > q;
```

Khai báo để lấy giá trị min ở đầu hàng đợi ưu tiên:

```
typedef pair <int, int> ii;
```

```
priority_queue <ii, vector<ii>, greater<ii> > q;
```

Code tham khảo:

```

1  #include <bits/stdc++.h>
2  #define nm 100005
3  #define vc 100000000
4
5  using namespace std;
6  typedef pair<int, int> ii;
7  int m, n, k, s;
8  vector<ii> g[nm];
9  priority_queue<ii, vector<ii>, greater<ii>> q;
10 vector<int> kc(nm+1,vc), p(nm+1);
11 ii x;
12
13 void nhap()
14 {
15     cin >> n >> m;
16     for(int i =1; i<=m; i++)
17     {
18         int u, v, w;
19         cin >> u >> v >> w;
20         x.first = w;
21         x.second = v;
22         g[u].push_back(x);
23         x.second = u;
24         g[v].push_back(x);
25     }
26     cin >> s;
27 }
28
29 void dijsktra()
30 {
31     kc[s] = 0;
32     q.push(make_pair(0,s));
33     while (!q.empty())
34     {
35         int u = q.top().second;
36         int kc_cur = q.top().first;
37         q.pop();
38         if (kc[u] < kc_cur)
39             continue;
40         for(int i = 0; i<g[u].size(); i++)
41         {
42             int v = g[u][i].second;
43             int len = g[u][i].first;
44             if (kc[v] > kc[u] + len)
45             {
46                 kc[v] = kc[u] + len;
47                 p[v] = u;
48                 q.push(make_pair(kc[v],v));
49             }
50         }
51     }
52 }
53

```

```

51 void inkq()
52 {
53     for(int i = 1; i<=n; i++)
54         if (kc[i] != vc)
55             cout << kc[i] << " ";
56         else
57             cout << "-1" << " ";
58     cout << endl;
59
60     cin >> k;
61     vector <int> path;
62     for(int i = 1; i<=k; i++)
63     {
64         int t;
65         cin >> t;
66         cout << "Duong di tu " << s << " toi " << t << ": ";
67         for(int j = t; j!=s; j= p[j])
68             path.push_back(j);
69         path.push_back(s);
70         reverse(path.begin(), path.end());
71         for(int j = 0; j<path.size(); j++)
72             cout << path[j] << " ";
73         cout << endl;
74     }
75 }
76
77 int main()
78 {
79     freopen("dijsktra.inp", "r", stdin);
80     freopen("dijsktra.out", "w", stdout);
81     nhap();
82     dijsktra();
83     inkq();
84     return 0;
85 }

```

II. Thuật toán Floyd – Warshall

Dùng để tìm đường đi ngắn nhất giữa mọi cặp cạnh của đồ thị.

Bài toán: Xét đồ thị vô hướng không chứa chu trình âm, n đỉnh và m cạnh. Cạnh i nối trực tiếp 2 đỉnh a_i và b_i với trọng số t_i , $i = 1 \div m$.

Cho q truy vấn, mỗi truy vấn yêu cầu tìm độ dài đường đi ngắn nhất từ x tới y ($1 \leq x, y \leq n$) và chỉ ra đường đi trong trường hợp tồn tại. Nếu không có đường đi thì đưa ra số -1.

Input:

- Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n \leq 500$, $0 \leq m \leq n(n-1)/2$)
- Dòng thứ i trong m dòng sau chứa 3 số nguyên a_i , b_i và t_i ($1 \leq a_i, b_i \leq n$, $0 \leq t_i \leq 10000$)
- Dòng tiếp theo chứa số nguyên q ($1 \leq q \leq 10000$)
- Mỗi dòng trong q dòng tiếp theo chứa 2 số nguyên x và y .

Output:

Với mỗi truy vấn đưa ra số -1 nếu không có đường đi hoặc 2 dòng thông tin:

- Dòng đầu tiên chứa một số nguyên – độ dài đường đi ngắn nhất.
- Dòng thứ 2 chứa các số nguyên xác định đường đi từ x đến y .

Ví dụ:

Input	Output
6 10	8
1 2 2	1 3 6 4
1 3 4	7
1 6 6	5 6 4
2 4 10	8
2 3 8	4 6 3 1
6 3 1	
3 4 20	
6 4 3	
5 6 4	
5 4 9	
3	
1 4	
5 4	
4 1	

Tổ chức dữ liệu và giải thuật:

Kích thước đồ thị không quá lớn và số truy vấn có thể rất nhiều vì vậy có thể tìm đường đi ngắn nhất giữa 2 đỉnh bất kì, từ đó việc xử lý truy vấn đơn thuần chỉ là tra cứu và dẫn xuất độ dài cũng như đường đi ngắn nhất cần tìm.

Khởi tạo: $d_{ij} = a_{ij}$ là độ dài cạnh nối trực tiếp đỉnh i với đỉnh j .
 $d_{ij} = +\infty$, nếu không có đường đi trực tiếp từ i tới j .
 $d_{ii} = 0$ với mọi i .

Xét mọi đỉnh k ($k = 1 \div n$). Với mỗi đỉnh k được xét, **Thuật toán** lại xét tiếp tất cả các cặp đỉnh (i, j) : nếu đồ thị có cạnh (i, k) và cạnh (k, j) thì ta cập nhật lại đường đi từ i tới j qua cạnh trung gian k nếu cần.

$$d_{ij} = \min(d_{ij}, d_{ik} + d_{kj});$$

Tức là đường đi mới từ i tới j tốt hơn đường đi cũ nếu có một đỉnh trung gian cho phép cải tiến đường đi.

Để khôi phục đường đi ngắn nhất ta phải giữ lại giá trị r_{ij} – lưu trữ đỉnh trung gian tốt nhất cần qua trên đường đi từ i tới j .

Code tham khảo.

```
1  #include <bits/stdc++.h>
2  #define nm 105
3  #define oo 100000
4
5  using namespace std;
6
7  int g[nm][nm], n, m, t, trace[nm][nm];
8  vector<int> res;
9
10 void nhap()
11 {
12     cin >> n >> m;
13     for(int i = 1; i<=n; i++)
14         for(int j = 1; j<=n; j++)
15             g[i][j] = oo;
16     for(int i = 1; i<=n; i++)
17         g[i][i] = 0;
18     for(int i = 1; i<=m; i++)
19     {
20         int u, v, w;
21         cin >> u >> v >> w;
22         g[u][v] = w;
23         g[v][u] = w;
24     }
25     cin >> t;
26 }
27
28 void floy()
29 {
30     for(int k = 1; k<=n; k++)
31         for(int i = 1; i<=n; i++)
32             for(int j = 1; j<=n; j++)
33                 if(g[i][j] > g[i][k] + g[k][j])
34                 {
35                     g[i][j] = g[i][k] + g[k][j];
36                     trace[i][j] = k;
37                 }
38 }
39
40 void ddi(int i, int j)
41 {
42     if(trace[i][j] == 0)
43         res.push_back(j);
44     else
45     {
46         ddi(i, trace[i][j]);
47         ddi(trace[i][j], j);
48     }
49 }
50
```

```

52 void xuly()
53 {
54     int u, v;
55     for(int k = 1; k<=t; k++)
56     {
57         cin >> u >> v;
58         if(g[u][v]==oo)
59             cout << -1 << endl;
60         else
61         {
62             res.push_back(u);
63             ddi(u,v);
64             cout << g[u][v] << endl;
65             for(int i= 0; i<res.size(); i++)
66                 cout << res[i] << " ";
67             cout << endl;
68         }
69         res.clear();
70     }
71 }
72
73 int main()
74 {
75     freopen("floy.inp", "r", stdin);
76     freopen("floy.out", "w", stdout);
77     nhap();
78     floy();
79     xuly();
80     return 0;
81 }

```

III. BÀI TẬP

Bài 1. <https://vn.spoj.com/problems/FLOYD/>

FLOYD - Floyd hoặc Dijkstra (Cơ bản)

Cho đơn đồ thị vô hướng N đỉnh và M cạnh, trọng số các cạnh đều nguyên dương. Có 2 loại câu hỏi :

0 $u\ v$: Cho biết đường đi ngắn nhất từ u tới v có độ dài là bao nhiêu.

1 $u\ v$: Hãy chỉ ra 1 đường đi ngắn nhất từ $u \Rightarrow v$

Input

Dòng 1 : 3 số nguyên N, M, K . ($1 \leq N \leq 100, 1 \leq M \leq N*(N-1)/2, 1 \leq K \leq 1000$)

M dòng tiếp theo , dòng thứ i gồm 3 số nguyên dương u, v, c cho biết cạnh (u,v) có trọng số là c ($1 \leq c \leq 10000$)

K dòng tiếp theo là K câu hỏi , dòng thứ j sẽ có định dạng như đã nêu ở trên.

Output

Ứng với mỗi câu hỏi trong K câu hỏi thì ta phải trả lời trên mỗi dòng như sau .

Câu hỏi 0 $u\ v$: Ghi ra 1 số nguyên duy nhất là độ dài đường đi ngắn nhất từ u đến v .

Câu hỏi 1 $u\ v$: Ghi ra số đầu tiên là số X là số đỉnh trên đường đi ngắn nhất này, tiếp đó ghi ra X số là chỉ số các đỉnh theo thứ tự xuất hiện trên hành trình.

Example

Input	Output
3 3 2	3
1 2 3	3 1 2 3
2 3 1	
1 3 5	
0 1 2	
1 1 3	

Bài 2. <https://vn.spoj.com/problems/QBSCHOOL/>

QBSCHOOL - Đến trường

Ngày 27/11 tới là ngày tổ chức thi học kỳ I ở trường ĐH BK. Là sinh viên năm thứ nhất, Hiếu không muốn vì đi muộn mà gặp trục trặc ở phòng thi nên đã chuẩn bị khá kỹ càng. Chỉ còn lại một công việc khá gay go là Hiếu không biết đi đường nào tới trường là nhanh nhất.

Thường ngày Hiếu không quan tâm tới vấn đề này lắm cho nên bây giờ Hiếu không biết phải làm sao cả . Bản đồ thành phố là gồm có N nút giao thông và M con đường nối các nút giao thông này. Có 2 loại con đường là đường 1 chiều và đường 2 chiều. Độ dài của mỗi con đường là một số nguyên dương.

Nhà Hiếu ở nút giao thông 1 còn trường ĐH BK ở nút giao thông N . Vì một lộ trình đường đi từ nhà Hiếu tới trường có thể gặp nhiều yếu tố khác như là gặp nhiều đèn đỏ, đi qua công trường xây dựng, ... phải giảm tốc độ cho nên Hiếu muốn biết là có tất cả bao nhiêu lộ trình ngắn nhất đi từ nhà tới trường. Bạn hãy lập trình giúp Hiếu giải quyết bài toán khó này.

Input

Dòng thứ nhất ghi hai số nguyên N và M .

M dòng tiếp theo, mỗi dòng ghi 4 số nguyên dương K, U, V, L. Trong đó:

K = 1 có nghĩa là có đường đi một chiều từ U đến V với độ dài L.

K = 2 có nghĩa là có đường đi hai chiều giữa U và V với độ dài L.

Output

Ghi hai số là độ dài đường đi ngắn nhất và số lượng đường đi ngắn nhất. Biết rằng số lượng đường đi ngắn nhất không vượt quá phạm vi int64 trong pascal hay long long trong C++.

Giới hạn:

$$1 \leq N \leq 5000$$

$$1 \leq M \leq 20000$$

$$\text{Độ dài các con đường} \leq 32000$$

Example

Input	Output
3 2	4 1
1 1 2 3	
2 2 3 1	

Bài 3. <https://vn.spoj.com/problems/VDANGER/>

VDANGER - Nguy hiểm rõ ràng trước mắt

Nông dân John đang ở trên một con thuyền nhỏ và đang tìm kiếm kho báu ở 1 trong số N ($1 \leq N \leq 100$) hòn đảo (đánh số từ 1..N) ở vùng biển Ca-ri-bô.

Bản đồ kho báu cho John biết John cần phải thực hiện 1 hành trình đi qua đảo A_1, A_2, ... A_M ($2 \leq M \leq 10,000$), bắt đầu từ đảo 1 và kết thúc ở đảo N trước khi kho báu biến mất. Anh ta có thể đến thăm các đảo khác và thăm bao nhiêu lần tùy thích, miễn là hành trình của ông ta phải chứa dãy A_1,...A_M là 1 dãy con (không nhất thiết phải liên tiếp nhau).

John muốn tránh đụng độ cướp biển và biết được mức-độ-bị-cướp ($0 \leq \text{mức-độ-bị-cướp} \leq 100,000$) khi đi lại giữa 2 hòn đảo với nhau. Độ nguy hiểm của hành trình của John sẽ là tổng các mức-độ-bị-cướp trên các tuyến đường mà John đi qua.

Hãy giúp John tìm được 1 hành trình ít nguy hiểm nhất để có thể lấy được kho báu.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và M
- Dòng 2..M+1: Dòng i+1 mô tả chứa 1 số nguyên là đảo thứ i mà John cần phải tới: A_i
- Dòng M+2..N+M+1: Dòng i+M+1 chứa N số nguyên cách nhau bởi dấu cách tương ứng là mức-độ-bị-cướp trên tuyến đường đi giữa đảo i và đảo 1, 2,...N; đảm bảo số nguyên thứ i luôn là số 0.

Kết quả

- Dòng 1: Độ nguy hiểm nhỏ nhất của hành trình của John.

Ví dụ

Input	Output	Giải thích
3 4 1 2	7	Có 3 hòn đảo và bản đồ kho báu yêu cầu John phải thực hiện 1 hành trình tới các đảo như sau: từ đảo 1 tới đảo 2, quay lại đảo 1 và cuối cùng là tới đảo 3. Mức-độ-bị-cướp trên các tuyến đường

1 3 0 5 1 5 0 2 1 2 0		<p>đã được cho: (1, 2); (2, 3); (3, 1) có độ lớn tương ứng là 5, 2 và 1. Hành trình có độ nguy hiểm nhỏ nhất là 7. John sẽ đi như sau: 1, 3, 2, 3, 1, and 3. Yêu cầu của bản đồ là phải chứa dãy (1, 2, 1, và 3) và hành trình này thỏa mãn yêu cầu.</p> <p>Chúng ta sẽ tránh đi trên đường nối giữa 2 đảo 1 và 2 vì nó có mức-độ-bị-cướp lớn.</p>
-----------------------------------	--	--

Bài 4. <https://vn.spoj.com/problems/TTRIP/>

TTRIP - Tham quan Thành Cổ

Trong kì thi IOI tại Thái Lan vừa qua, sau 2 ngày làm bài đầy căng thẳng, Tuệ cùng các thí sinh khác được đi tham quan Thành Cổ (Ancient City), 1 địa danh du lịch khá nổi tiếng nơi đây. Thành Cổ ngoài lối vào (được đánh số 1) và lối ra (được đánh số N), được chia ra làm N-2 khu vực khác nhau (được đánh số từ 2 đến N-1), mỗi khu vực được xây dựng theo 1 lối kiến trúc riêng vô cùng độc đáo. Giữa các khu vực này có thể có các lối đi, được biểu diễn bằng ma trận A.

Hành trình của Tuệ sẽ bắt đầu từ lối vào, tham quan các khu vực trong Thành Cổ và kết thúc ở lối ra. Là 1 người yêu thích chụp ảnh, Tuệ chắc chắn sẽ không bỏ qua 1 khu vực nào nếu cậu ta có thể đến được nó thông qua các con đường. Tại mỗi địa điểm, nếu còn ít nhất 1 khu vực Tuệ có thể đến được nhưng vẫn chưa đến tham quan, cậu ta sẽ chọn khu vực gần nhất so với vị trí hiện tại của cậu ta (có thể di chuyển qua các khu vực đã tham quan rồi hoặc lối vào, lối ra). Nếu có nhiều hơn 1 khu vực thỏa yêu cầu, Tuệ sẽ chọn khu vực có số thứ tự nhỏ nhất.

Hãy tính tổng độ dài đường đi trong chuyến tham quan của Tuệ. Luôn đảm bảo có ít nhất 1 cách để Tuệ di chuyển từ lối vào đến lối ra.

Input

Dòng 1: số nguyên N.

Dòng 2...N+1: dòng thứ i+1 chứa N số nguyên $A_{i,1} A_{i,2} \dots A_{i,n}$; trong đó $A_{i,j} > 0$ nếu có lối đi và $A_{i,j} = 0$ nếu không có (với mọi i khác j, luôn đảm bảo $A_{i,j} = A_{j,i}$ và $A_{i,i} = 0$).

Output

Tổng độ dài chuyến tham quan của Tuệ.

Constraints

$$2 \leq N \leq 100.$$

$$0 \leq A \leq 10^6.$$

Example

Input	Output	Giải thích
5 0 3 2 0 0 3 0 2 4 5 2 2 0 1 0 0 4 1 0 2 0 5 0 2 0	11	<p>Giải thích: Thứ tự các khu vực tham quan là 3, 4, 2. Hành trình cụ thể: $1 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5$.</p>

Bài 5. <https://vn.spoj.com/problems/QBBUILD/>

QBBUILD - Xây dựng đường

Vua Peaceful vừa khai hoang một vùng đất để lập ra đất nước Peace, lúc đầu chỉ có N thành phố (được đánh số từ 1 đến N) và không có con đường nào.

Vua Peace chọn ra 4 thành phố đặc biệt để làm trung tâm kinh tế và 4 thành phố này phải được liên thông với nhau. Chi phí xây dựng các con đường không phải nhỏ vì thế nhà vua muốn sử dụng chi phí ít nhất để xây dựng các con đường sao cho 4 thành phố đặc biệt đó vẫn liên thông.

Bạn được biết chi phí ước tính để xây dựng một số con đường và bạn hãy chọn một số con đường để xây dựng để theo đúng ý nhà vua biết rằng luôn tồn tại ít nhất một phương án xây dựng đường sao cho 4 thành phố đặc biệt liên thông.

Input

Dòng đầu tiên ghi số nguyên dương N là số lượng các thành phố. ($1 \leq N \leq 100$)

Dòng thứ hai ghi 4 số nguyên là số hiệu của 4 thành phố đặc biệt.

Trong một số dòng tiếp theo, mỗi dòng ghi 3 số nguyên u, v và c với ý nghĩa muốn xây dựng một con đường hai chiều nối trực tiếp giữa 2 thành phố u và v thì chi phí là c. ($1 \leq c \leq 5000$)

Output

Gồm 1 dòng duy nhất là tổng chi phí nhỏ nhất để xây dựng hệ thống đường.

Example

Input	Output
5 2 3 4 1 1 2 10 1 5 1 5 2 1 1 4 1 4 3 3 3 2 2	5

Bài 6. <https://vn.spoj.com/problems/CENTRE28/>

CENTRE28 - CENTRE

Theo thống kê cho biết mức độ tăng trưởng kinh tế của nước Peace trong năm 2006 rất đáng khả quan. Cả nước có tổng cộng N thành phố lớn nhỏ được đánh số tuần tự từ 1 đến N phát triển khá đồng đều. Giữa N thành phố này là một mạng lưới gồm M đường đi hai chiều, mỗi tuyến đường nối 2 trong N thành phố sao cho không có 2 thành phố nào được nối bởi quá 1 tuyến đường. Trong N thành phố này thì thành phố 1 và thành phố N là 2 trung tâm kinh tế lớn nhất nước và hệ thống đường đảm bảo luôn có ít nhất một cách đi từ thành phố 1 đến thành phố N.

Tuy nhiên, cả 2 trung tâm này đều có dấu hiệu quá tải về mật độ dân số. Vì vậy, đức vua Peaceful quyết định chọn ra thêm một thành phố nữa để đầu tư thành một trung tâm kinh tế thứ ba. Thành phố này sẽ tạm ngưng mọi hoạt động thường nhật, cũng như mọi luồng lưu thông ra vào để tiến hành nâng cấp cơ sở hạ tầng. Nhưng trong thời gian sửa chữa ấy, phải bảo đảm đường đi ngắn nhất từ thành phố 1 đến thành phố N không bị thay đổi, nếu không nền kinh tế quốc gia sẽ bị trì trệ.

Vị trí và đường nối giữa N thành phố được mô tả như một đồ thị N đỉnh M cạnh. Hãy giúp nhà vua đếm số lượng thành phố có thể chọn làm trung tâm kinh tế thứ ba sao cho thành phố được chọn thỏa mãn các điều kiện ở trên

Input

Dòng đầu tiên ghi 2 số nguyên dương N và M là số thành phố và số tuyến đường.

Dòng thứ i trong số M dòng tiếp theo ghi 3 số nguyên dương x_i , y_i và d_i với ý nghĩa tuyến đường thứ i có độ dài d_i và nối giữa 2 thành phố x_i , y_i .

Output

Dòng đầu tiên ghi số tự nhiên S là số lượng các thành phố có thể chọn làm trung tâm kinh tế thứ ba.

S dòng tiếp theo, mỗi dòng ghi 1 số nguyên dương là số thứ tự của thành phố được chọn (In ra theo thứ tự tăng dần)

Giới hạn

- $2 \leq N \leq 30000$
- $1 \leq M \leq 100000$
- $1 \leq d_i \leq 1000$

Example

Input	Output
6 6	2
1 2 1	4
2 3 1	5
3 6 1	
1 4 100	
4 5 100	
5 6 100	

Bài 7. <https://vn.spoj.com/problems/NETACCEL/>

NETACCEL - Tăng tốc mạng máy tính

Cho mạng máy tính gồm N máy và M liên kết hai chiều giữa các máy. Các máy được đánh số từ 1 đến N. Máy của Bờm là máy 1 còn máy của Cuội là máy N. Mỗi đường nối cần tốn một giá trị thời gian khác nhau để dữ liệu truyền qua. Tốc độ kết nối giữa hai máy là độ dài đường truyền dữ liệu ngắn nhất giữa hai máy đó.

Tốc độ kết nối của mạng khá chậm khiến Bờm và Cuội không thể chơi Dota được, do đó Bờm quyết định mua K thiết bị tăng tốc mạng. Thiết bị tăng tốc mạng được gắn vào các đường

truyền dữ liệu giữa hai máy. Mỗi thiết bị sẽ làm giảm thời gian truyền dữ liệu của đường truyền đi một nửa.

Hãy giúp Bờm đặt các thiết bị tăng tốc sao cho tốc độ kết nối giữa máy của Bờm và Cuội là nhanh nhất có thể để hai bạn có thể chơi Dota mà không bị lag!

Dữ liệu

Dòng đầu chứa 3 số N, M, K .

M dòng tiếp theo, mỗi dòng chứa 3 số x, y, c mô tả một đường truyền dữ liệu: x, y là số hiệu của hai máy tính, còn c là thời gian truyền dữ liệu.

Giới hạn

- $1 \leq N \leq 1000$
- $1 \leq M \leq 100,000$
- $1 \leq K \leq 10$
- $1 \leq c \leq 1,000,000$

Kết quả

In ra 1 số duy nhất là tốc độ kết nối nhanh nhất có thể sau khi đã lắp đặt các thiết bị tăng tốc, làm tròn đến 2 chữ số thập phân.

Ví dụ

Input	Output	Giải thích
5 5 2 1 2 1 2 3 9 3 5 1 1 4 5 4 5 5	4.25	Bờm lắp cả 2 thiết bị tăng tốc lên đường nối giữa máy 2 và máy 3.

Hướng dẫn làm bài:

Bài 1.

Floy hoặc Dijkstra cơ bản

Bài 2.

Dijkstra + Quy hoạch động

Gọi $kc[i]$ là độ dài đường đi ngắn nhất từ đỉnh 1 tới đỉnh i .

Để tìm số đường đi có độ dài ngắn nhất:

- Gọi $f[i]$ là số đường đi ngắn nhất từ đỉnh 1 tới đỉnh i .
- Khởi tạo: $f[1] = 1$; $f[i] = 0$ ($i = 2 \div n$)
- Đi từ u đến v :
 - Nếu $kc[v] = kc[u] + w(u,v)$ (Đã có con đường khác tới v)
 $f[v] = f[v] + f[u]$;
 - Nếu $kc[v] > kc[u] + w(u,v)$ (Phải đi qua u)
 $f[v] = f[u]$;

Bài 3.

Sử dụng Dijkstra hoặc Floyd để tìm đường đi ngắn nhất giữa các cặp đỉnh, với trọng số của mỗi cạnh sẽ là mức độ nguy hiểm.

Kết quả của bài toán sẽ là tổng:

$$kc[1][x[1]] + kc[x[2]][x[3]] + \dots + kc[x[m]][n];$$

Bài 4. Dùng Floyd tính độ dài đường đi giữa mọi cặp đỉnh. Ta bắt đầu từ đỉnh 1, tiến hành chọn ra đỉnh gần đỉnh đang đứng ở hiện tại nhất và có số thứ tự nhỏ nhất. Cộng vào độ dài đường đi. Di chuyển đến đỉnh đó và tiến hành chọn tiếp cho đến khi không còn đỉnh nào để chọn.

Bài 5.

Trước tiên sử dụng Floyd tìm đường đi ngắn nhất giữa mọi cặp đỉnh. Chọn ra 2 thành phố bất kì làm trung gian để nối các thành phố đặc biệt lại (các thành phố đặc biệt cũng có thể được chọn làm thành phố trung gian). Trong trường hợp chung nhất, ta tưởng tượng 2 thành phố được chọn ra nằm ở giữa nối với nhau bởi đoạn đường đã được tính. Mỗi đỉnh lại nối với 2 thành phố đặc biệt, 2 đỉnh là 4 thành phố đặc biệt. Trong một số trường hợp, như đã nói ở trên, các thành phố đặc biệt cũng có thể tham gia làm thành phố trung gian.

```
for(int u = 1 ; u ≤ n; u++)
    for(int v = u; v ≤ n; v++)
        for(int i = 1; i ≤ 3; i++)
            for(int j = i + 1; j ≤ 4; j++)
            {
                int ans = a[f[i]][u] + a[f[j]][u] + a[u][v];
                for(int k = 1; k ≤ 4; k++)
                    if(k != i && k != j) ans += a[f[k]][v];
                res = min(res, ans);
            }
```

Bài 6.

Gọi $d1(u)$ là độ dài đường đi ngắn nhất từ 1 đến u , $f1(u)$ là số đường đi ngắn nhất từ 1 đến u .

Gọi $dn(u)$ là độ dài đường đi ngắn nhất từ u đến N , $fn(u)$ là số đường đi ngắn nhất từ u đến N .

Đỉnh $u(1 < u < N)$ có thể chọn làm trung tâm kinh tế mới nếu đường đi ngắn nhất từ 1 đến N không đi qua u , hoặc khi bỏ u vẫn còn ít nhất một đường đi ngắn nhất khác từ 1 đến N . Chuyển điều kiện trên thành công thức:

$$d1(u) + dn(u) > d1(n) \text{ hoặc}$$

$$f1(u) \times fn(u) < f1(n)$$

Mới: hàm tie() trong C++

`tie(n,m) = make_tuple(10,20);` // gán $n = 10$, $m = 20$

`tie(n,m) = nhap();` // gán n và m bằng cặp giá trị trả về từ hàm nhap()

Bài 7.

Gọi $d(u, i)$ là tốc độ truyền nhanh nhất từ 1 $\rightarrow u$ sử dụng i thiết bị tăng tốc
 $d(u, i) = \min(d(v, i - 1) + a[u][v] / 2, d(v, i) + a[u][v])$

=> Ta dùng dijkstra để giải bài toán này.

Code tham khảo

Bài 1.

```
1  #include <bits/stdc++.h>
2  #define nm 105
3  #define oo 100000
4
5  using namespace std;
6
7  int g[nm][nm], n, m, t, trace[nm][nm];
8  vector<int> res;
9
10 void nhap()
11 {
12     cin >> n >> m >> t;
13     for(int i = 1; i<=n; i++)
14         for(int j = 1; j<=n; j++)
15             g[i][j] = oo;
16     for(int i = 1; i<=n; i++)
17         g[i][i] = 0;
18     for(int i = 1; i<=m; i++)
19     {
20         int u, v, w;
21         cin >> u >> v >> w;
22         g[u][v] = w;
23         g[v][u] = w;
24     }
25 }
26
27 void floy()
28 {
29     for(int k = 1; k<=n; k++)
30         for(int i = 1; i<=n; i++)
31             for(int j = 1; j<=n; j++)
32                 if(g[i][j] > g[i][k] + g[k][j])
33                 {
34                     g[i][j] = g[i][k] + g[k][j];
35                     trace[i][j] = k;
36                 }
37 }
38
39 void ddi(int i, int j)
40 {
41     if(trace[i][j] == 0)
42         res.push_back(j);
43     else
44     {
45         ddi(i, trace[i][j]);
46         ddi(trace[i][j], j);
47     }
48 }
49
```

```

51 void xuly()
52 {
53     int q, u, v;
54     for(int k = 1; k<=t; k++)
55     {
56         cin >> q >> u >> v;
57         if(q==0)
58             cout << g[u][v] << endl;
59         else
60         {
61             res.push_back(u);
62             ddi(u,v);
63             cout << res.size() << " ";
64             for(int i= 0; i<res.size(); i++)
65                 cout << res[i] << " ";
66             cout << endl;
67         }
68         res.clear();
69     }
70 }
71
72 int main()
73 {
74     freopen("floy.inp", "r", stdin);
75     freopen("floy.out", "w", stdout);
76     nhap();
77     floy();
78     xuly();
79     return 0;
80 }
81

```

Bài 2.


```

1  #include <bits/stdc++.h>
2  #define nm 5005
3  #define mm 20005
4  #define oo INT_MAX
5  using namespace std;
6  typedef pair <int, int > ii;
7  vector <ii> g[nm];
8  vector <int> kc(nm, oo);
9  vector <long long> f(nm, 0ll);
10 int n, m;
11
12 void nhap()
13 {
14     cin >> n >> m;
15     for(int i = 1; i<=m; i++)
16     {
17         int k, u, v, l;
18         cin >> k >> u >> v >> l;
19         g[u].push_back({l, v});
20         if (k==2)
21             g[v].push_back({l, u});
22     }
23 }
24
25 void dijsktra()
26 {
27     priority_queue <ii, vector<ii>, greater<ii> > q;
28     kc[1] = 0;
29     q.push({0, 1});
30     f[1] = 1;
31     while(!q.empty())
32     {
33         int u = q.top().second;
34         int kc_cur = q.top().first;
35         q.pop();
36         if (kc[u] < kc_cur)
37             continue;
38         for(int i = 0; i< g[u].size(); i++)
39         {
40             int v = g[u][i].second;
41             int w = g[u][i].first;
42
43             if(kc[v] == kc[u] + w)
44                 f[v] = f[u] + f[v];
45             else
46                 if (kc[v] > kc[u] + w)
47                 {
48                     kc[v] = kc[u] + w;
49                     f[v] = f[u];
50                     q.push({kc[v], v});
51                 }
52         }
53     }
54 }
55

```

```

57     int main()
58     {
59         freopen("gbschool.inp", "r", stdin);
60         freopen("gbschool.out", "w", stdout);
61         nhap();
62         dijsktra();
63         cout << kc[n] << " " << f[n];
64         return 0;
65     }

```

Chuyên đề 9:
TUYỂN TẬP ĐỀ CODEFORCE

I. A problem

1. <https://codeforces.com/problemset/problem/469/A>

Tóm tắt: X và Y muốn chơi “I Wanna Be The Guy”. Mỗi người có thể chơi được p trong số n màn chơi. Hỏi xem nếu hai người chơi cùng nhau liệu có hoàn thành được game (qua hết tất cả các màn) hay không?

Thuật toán:

Thử xem tất cả các màn đã được qua chưa bằng cách đánh dấu chúng nếu X hoặc Y qua được. Game sẽ hoàn tất nếu tất cả các phần tử đều được đánh dấu.

Độ phức tạp: nếu gọi X qua được p_1 màn, Y qua được p_2 màn thì $O(\max(p_1, p_2))$.

Code:

```
#include <bits/stdc++.h>
using namespace std;
int n, p, a;
map <int, bool> lv;
bool flag = 1;
int main();
{
    cin >> n;
    for (int i = 0; i < 2; i++)
    {
        cin >> p;
        while (p--)
        {
            cin >> a;
            lv[a] = 1;
        }
    }
    for (int i = 1; i ≤ n; i++)
        if (!lv[i])
            flag = 0;
    (flag)? cout << "I become the guy.": cout << "Oh, my keyboard!";
    return 0;
}
```

2. <https://codeforces.com/problemset/problem/41/A>

Tóm tắt: Kiểm tra xem một xâu đã được “dịch” (đảo xâu) hay chưa.

Thuật toán: Kiểm tra xâu đảo.

Code:

```
#include <bits/stdc++.h>
#define gl(s) getline(cin, s)
using namespace std;
string s1, s2;
int main()
{
    gl(s1); gl(s2);
    reverse(s1.begin(), s1.end());
    (s1 == s2)? cout << "YES": cout << "NO";
    return 0;
}
```

3. <https://codeforces.com/contest/339/problem/A>

Tóm tắt: “Sort” một phép toán cộng theo thứ tự tăng dần.

VD: $1 + 3 + 2 + 1 \rightarrow 1 + 1 + 2 + 3$.

Thuật toán: Ta sẽ tách từng số ra và sắp xếp chúng lại.

Code:

```
#include <bits/stdc++.h>
#define gl(s) getline(cin, s)
using namespace std;
string s;
vector <char> p;
int main()
{
    gl(s);
    for (int i = 0; i < s.size(); i += 2)
        p.push_back(s[i]);
    sort(p.begin(), p.end());
    for (int i = 0; i < p.size(); i++)
    {
        cout << p[i];
        if (i != p.size() - 1) cout << '+';
    }
    return 0;
}
```

4. <https://codeforces.com/problemset/problem/705/A>

Thuật toán:

Code:

```
#include <bits/stdc++.h>
```

```

using namespace std;
int n;
string p[] = {"I hate ", "I love "};
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << p[i%2];
        if (i != n - 1) cout << "that ";
    }
    cout << "it";
    return 0;
}

```

5. <https://codeforces.com/contest/1325/problem/A>

Thuật toán: cặp 1 và $n-1$ là một nghiệm của bài toán. Thật vậy, $\text{GCD}(1, n-1) = 1$ và $\text{LCM}(1, n-1) = n-1$.

Code

```

#include <bits/stdc++.h>
using namespace std;
int T, n;
int main()
{
    cin >> T;
    while (T--)
    {
        cin >> n;
        cout << 1 << " " << n - 1 << endl;
    }
    return 0;
}

```

II. B problem

6. <https://codeforces.com/problemset/problem/519/B>

Tóm tắt: tìm ra hai số bị loại ra khỏi mảng.

Thuật toán: gọi tổng mảng đầu tiên là a , tổng mảng hai là b và tổng mảng ba là c , thì ta có $a - b$ và $b - c$ là hai số cần tìm.

Code:

```

#include <bits/stdc++.h>
using namespace std;
int n, a, ls;

```

```

int main()
{
    cin >> n;
    for (int i = 0; i < 3; i++)
    {
        int s = 0;
        for (int j = 0; j < n; j++)
        {
            cin >> a;
            s += a;
        }
        if (i) cout << ls - s << endl;
        ls = s;
        n--;
    }
    return 0;
}

```

7. <https://codeforces.com/contest/1263/problem/B>

Tóm tắt: tạo ra các mật khẩu khác nhau với số lần thay ít nhất.

Thuật toán: do ta có tối đa 10 mật khẩu, nên chúng ta chỉ cần nhiều nhất 9 lần đổi để tối ưu bài toán, tức là với mỗi số, ta chỉ cần đổi một ký tự, và phải cùng hàng. Ở đây Mino chọn hàng đơn vị để thay đổi mật khẩu.

Code:

```

#include <bits/stdc++.h>
using namespace std;
int T, n, res;
string a[11];
map <char, bool> k;
bool check(int j)
{
    for (int i = 0; i < j; i++)
        if (a[i] == a[j])
            return 0;
    return 1;
}
int main()
{
    cin >> T;
    while (T--)
    {

```

```

        k.clear();
        res = 0;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
        k[a[i][3]] = 1;
    }
    for (int i = 1; i < n; i++)
    {
        if (!check(i))
        {
            res++;
            char m = '0';
            while (k[m]) m++;
            a[i].pop_back();
            a[i].push_back(m);
            k[m] = 1;
        }
    }
    cout << res << endl;
    for (int i = 0; i < n; i++)
        cout << a[i] << endl;
}

return 0;
}

```

8. <https://codeforces.com/contest/1293/problem/B>

Thuật toán: Để đạt được tiền thưởng tối đa, hãy phải để Joe loại từng người một.

Code:

```

#include <bits/stdc++.h>
using namespace std;
int n;
double s = 0;
int main()
{
    cin >> n;
    while (n)
        s += 1.0/(n--);
    cout << setiosflags(ios::showpoint);
    cout << setprecision(13) << s;
}

```

```
    return 0;
}
```

9. <https://codeforces.com/contest/1300/problem/B>

Thuật toán: Để chia được hai lớp học “cân bằng”, ta chỉ cần để ý sao hai học sinh nằm giữa phải học khác lớp. Khi đó độ chênh lệch sẽ đạt tối thiểu. Có thể hiểu hai lớp sẽ được chia so le: HS 1 (học kém nhất) sẽ vào lớp A, HS 2 sẽ vào lớp B, HS 3 sẽ vào lớp A,... cứ như thế cho đến hết, ta được một cách tối ưu.

Chứng minh đầy đủ: <https://codeforces.com/blog/entry/73763>.

Code:

```
#include <bits/stdc++.h>
using namespace std;
int T, n;
vector <int> p;
int main()
{
    cin >> T;
    while (T--)
    {
        p.clear();
        cin >> n;
        p.resize(2*n);
        for (int i = 0; i < 2*n; i++)
            cin >> p[i];
        sort(p.begin(), p.end());
        cout << p[n] - p[n - 1] << endl;
    }
    return 0;
}
```

10. <https://codeforces.com/problemset/problem/1213/B>

Thuật toán: Ta duyệt từ cuối về đầu, nếu có một số nào đó lớn hơn min của đoạn sau về cuối dãy thì đó là một số cần tìm, ta tăng res, rồi cập nhật lại min. Tiếp tục thực hiện đến khi về đầu dãy, ta được res là đáp án cần tìm.

Độ phức tạp: $O(n)$.

Code:

```
#include <bits/stdc++.h>
using namespace std;
int T, n;
vector <int> a;
int main()
{
```



```
cin >> T;
while (T--)
{
    a.clear();
    cin >> n;
    a.resize(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    int minn = a.back(), res = 0;
    for (int i = n - 2; i >= 0; i--)
        if (a[i] > minn) res++;
        else minn = a[i];
    cout << res << endl;
}
return 0;
}
```

Chuyên đề 10
TUYỂN TẬP ĐỀ THI HSG

I. ĐỀ I

TỔNG QUAN VỀ ĐỀ THI

	Tên bài	File chương trình	File dữ liệu vào	File kết quả
Câu 1	Tích các chữ số lớn nhất	CALC.*	CALC.INP	CALC.OUT
Câu 2	Số chính phương	SQUA.*	SQUA.INP	SQUA.OUT
Câu 3	Vị trí đẹp 2	PLUCKY2.*	PLUCKY2.INP	PLUCKY2.OUT
Câu 4	Ước nguyên tố	PRIDIV.*	PRIDIV.INP	PRIDIV.OUT

Dấu * được thay bằng PAS hoặc CPP của ngôn ngữ lập trình được sử dụng là Pascal hoặc C++.

Hãy lập trình giải các bài toán sau:

Câu 1. (7,0 điểm) Tích các chữ số lớn nhất

Cho số nguyên dương N, M và P là các số có 3 chữ số. Tìm số có tích các chữ số lớn nhất. Đưa tích lớn nhất ra.

Ví dụ: N = 234 → tích1 = 24; M = 123 → tích2 = 6; P = 321 → tích3=6.

Kết quả: Tích lớn nhất là 24.

Dữ liệu: Vào từ tệp CALC.INP gồm một dòng duy nhất chứa số nguyên N, M và P là các số có 3 chữ số. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp CALC.OUT số nguyên duy nhất là tích lớn nhất tìm được.

Ví dụ:

CALC.INP	CALC.OUT
234 123 321	24

Câu 2. (6,0 điểm) Số chính phương

Số chính phương (hay còn gọi là *số hình vuông*) là số tự nhiên có căn bậc 2 là một số tự nhiên.

Các số chính phương đầu tiên: 1, 4, 9, 16, 25....

Cho số nguyên dương N, hãy tìm số chính phương bé nhất lớn hơn N.

Ví dụ: N = 10 → Kết quả: 16.

Dữ liệu: Vào từ tệp SQUA.INP số nguyên N ($N \leq 10^9$).

Kết quả: Ghi ra tệp SQUA.OUT số chính phương bé nhất lớn hơn N.

Ví dụ:

SQUA.INP	SQUA.OUT
10	16

Câu 3. (4,0 điểm) Vị trí đẹp 2

Cho N và dãy a_1, a_2, \dots, a_N .

Vị trí i gọi là vị trí đẹp nếu i chia dãy số thành 2 đoạn mà tổng các phần tử của đoạn đầu gấp đôi tổng các phần tử của đoạn sau.

VD: $N = 6$; **1 7 4 2 3 4** \rightarrow Vị trí đẹp là: 4

Giải thích: $1 + 7 + 4 + 2 = 14 = 2 \cdot (3 + 4)$

Dữ liệu: Vào từ tệp PLUCKY2.INP gồm:

- Dòng 1: Ghi số nguyên dương N ($N \leq 10^5$).
- Dòng 2: Ghi N số nguyên a_1, a_2, \dots, a_N ($a_i \leq 10^9$).

Kết quả: Ghi ra tệp PLUCKY2.OUT các vị trí đẹp. Mỗi số cách nhau một dấu cách. Nếu không có thì ghi ra -1.

PLUCKY2.INP	PLUCKY2.OUT
6 1 7 4 2 3 4	4
3 1 2 4	-1

Câu 4. (3,0 điểm) Ước nguyên tố

Cho số nguyên dương N , hãy tìm ước nguyên tố lớn nhất của N .

Ví dụ: $N = 28$; N chia hết cho 1, 2, 4, 7 và 28; \rightarrow Kết quả: 7 là số nguyên tố lớn nhất là ước của 28.

Dữ liệu: Vào từ tệp PRIDIV.INP chứa số nguyên N ($N \leq 10^{12}$).

Kết quả: Ghi ra tệp PRIDIV.OUT ước nguyên tố lớn nhất của N .

Ví dụ:

PRIDIV.INP	PRIDIV.OUT
28	7
17	17

___Hết___

II. ĐỀ II

TỔNG QUAN VỀ ĐỀ THI

	Tên bài	File chương trình	File dữ liệu vào	File kết quả
Câu 1	Chữ số lẻ lớn nhất	NODD.*	NODD.INP	NODD.OUT
Câu 2	Ước chung lớn nhất	THREEGCD.*	THREEGCD.INP	THREEGCD.OUT
Câu 3	Giải Nhì	SECOND.*	SECOND.INP	SECOND.OUT
Câu 4	Tìm K	KMIN1.*	KMIN1.INP	KMIN1.OUT

Dấu * được thay bằng PAS hoặc CPP của ngôn ngữ lập trình được sử dụng là Pascal hoặc C++.

Hãy lập trình giải các bài toán sau:

Câu 1. (7,0 điểm) Chữ số lẻ lớn nhất

Cho số nguyên dương N , M và P là các số có 4 chữ số. Tìm số lẻ lớn nhất có mặt trong các số N hoặc M hoặc P .

Ví dụ: $N = 3121$; $M = 3530$; $P = 6572 \rightarrow \text{res} = 7$.

Dữ liệu: Vào từ tệp NODD.INP gồm một dòng duy nhất chứa số nguyên N , M và P là các số có 4 chữ số. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp NODD.OUT số nguyên duy nhất là số lẻ lớn nhất có mặt trong các số N hoặc M hoặc P . Nếu không có chữ số lẻ nào ghi ra -1.

Ví dụ:

NODD.INP	NODD.OUT
3121 3530 6572	7
2222 4000 6600	-1

Câu 2. (6,0 điểm) Ước chung lớn nhất

Cho các số nguyên dương M , N và P .

Hãy tìm ước chung lớn nhất của M , N và P .

Ví dụ: $M = 10$; $N = 25$; $P = 50 \rightarrow \text{res} = 5$

Dữ liệu: Vào từ tệp THREEGCD.INP số nguyên dương M , N và P ($M, N, P \leq 10^9$). Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp THREEGCD.OUT ước chung lớn nhất của M , N và P .

Ví dụ:

THREEGCD.INP	THREEGCD.OUT
10 25 50	5

Câu 3. (4,0 điểm) Giải Nhì

Contest#1 “Làm giàu không khó” có N thí sinh tham dự, thí sinh thứ i đạt điểm $a[i]$. Ban tổ chức muốn biết số lượng thí sinh đạt Giải Nhì trong Contest1.

Dữ liệu: Vào từ tệp SECOND.INP gồm:

- Dòng 1: Ghi số nguyên dương N ($N \leq 10^6$).

- Dòng 2: Ghi N số nguyên dương a_1, a_2, \dots, a_N ($a_i \leq 10^9$). Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp SECOND.OUT số lượng thí sinh đạt Giải Nhì. Nếu không có thì ghi ra -1.

SECOND.INP	SECOND.OUT
6 1 7 4 2 3 4	2
3 4 4 4	-1

Câu 4. (3,0 điểm) Tìm K

Cho số nguyên dương M. Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M.

Ví dụ: $M = 30 \rightarrow K = 56$

Dữ liệu: Vào từ tệp KMIN1.INP chứa số nguyên M ($M \leq 10^9$).

Kết quả: Ghi ra tệp KMIN1.OUT giá trị K nhỏ nhất tìm được. Nếu không tồn tại K, đưa ra -1.

Ví dụ:

KMIN1.INP	KMIN1.OUT
30	56
17	-1

---Hết đề 2---

III. ĐỀ III

TỔNG QUAN VỀ ĐỀ THI

	Tên bài	File chương trình	File dữ liệu vào	File kết quả
Câu 1	Ba anh em siêu nhân	GBOYS.*	GBOYS.INP	GBOYS.OUT
Câu 2	Phép tính với phân số	MINIFRAC.*	MINIFRAC.INP	MINIFRAC.OUT
Câu 3	Bài toán đếm	GCDCOUNT.*	GCDCOUNT.INP	GCDCOUNT.OUT
Câu 4	Nhầm lẫn	MISTAKE.*	MISTAKE.INP	MISTAKE.OUT

Dấu * được thay bằng PAS hoặc CPP của ngôn ngữ lập trình được sử dụng là Pascal hoặc C++.

Hãy lập trình giải các bài toán sau:

Câu 1. (7,0 điểm) Ba anh em siêu nhân

Lân_eZ, Hải_vOI và Quang_Quác tham dự Contest#3 với quyết tâm giành giải “*Thí sinh nộp bài sớm nhất*”. Nên ba anh em làm bài với tốc độ thần thánh và có tổng thời gian làm bài của từng người lần lượt là X, Y, Z (giây).

Thầy MrT bắt đầu tính giờ làm Contest#3 vào lúc A (giờ) B (phút) C(giây).

Hãy cho biết, thầy MrT nhận được bài nộp sớm nhất lúc nào? Giả thiết thao tác gửi bài của mỗi thí sinh là không đáng kể.

Dữ liệu: Vào từ tệp GBOYS.INP gồm:

- Dòng 1: Chứa số nguyên dương X, Y và Z ($X, Y, Z \leq 9000$).
- Dòng 2: Chứa số nguyên dương A, B, C ($A \leq 15$, C và $B \leq 59$).

Kết quả: Ghi ra tệp GBOYS.OUT thời điểm thầy MrT nhận được bài sớm nhất. Mỗi số cách nhau một dấu cách.

Ví dụ:

GBOYS.INP	GBOYS.OUT	Giải thích
4500 3365 8700 14 0 0	14 56 5	Thời gian sớm nhất là: 14 giờ, 56 phút, 5 giây

Câu 2. (6,0 điểm) Phép tính với phân số

MrT đang học các phép toán với phân số. Các phép toán cộng, trừ hai phân số được mô tả như sau:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd} \text{ và } \frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$$

Cho a, b, c và d. Hãy giúp MrT tính tổng và hiệu các phân số. Kết quả đưa ra dạng phân số tối giản.

Ví dụ:

$$\frac{3}{8} + \frac{2}{4} = \frac{3.4 + 2.8}{8.4} = \frac{28}{32} = \frac{7}{8}$$

$$\frac{3}{8} - \frac{2}{4} = \frac{3.4 - 2.8}{8.4} = \frac{-4}{32} = \frac{-1}{8}$$

Dữ liệu: Vào từ tệp MINIFRAC.INP số nguyên a, b, c, d ($|a|, |b|, |c|, |d| \leq 10^9$, $b \neq 0$, $d \neq 0$).

Kết quả: Ghi ra tệp MINIFRAC.OUT tổng và hiệu của hai phân số dạng: tử số, dấu cách, mẫu số.

Ví dụ:

MINIFRAC.INP	MINIFRAC.OUT
3 8 2 4	7 8 -1 8
3 2 3 2	3 1 0 2

Câu 3. (4,0 điểm) Bài toán đếm

Cho N , K và dãy a_1, a_2, \dots, a_N .

Hãy đếm số cặp (i, j) sao cho ước chung lớn nhất của a_i và a_j bằng K .

VD: $N = 6, K = 2$; dãy 1 7 4 2 3 4 \rightarrow Res = 2.

Giải thích: Có 2 cặp (i, j) có các phân tử có ước chung lớn nhất bằng 2 là cặp $(3, 4); (4, 6)$.

Dữ liệu: Vào từ tệp GCDCOUNT.INP gồm:

- Dòng 1: Ghi số nguyên dương N và K ($N \leq 3000, K \leq 10^9$).
- Dòng 2: Ghi N số nguyên dương a_1, a_2, \dots, a_N ($a_i \leq 10^9$).

Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp GCDCOUNT.OUT số cặp (i, j) tìm được. Nếu không có cặp nào thỏa mãn thì ghi ra -1.

GCDCOUNT.INP	GCDCOUNT.OUT
6 2 1 7 4 2 3 4	2
3 3 1 2 4	-1

Câu 4. (3,0 điểm) Nhầm lẫn

MrT làm các phép toán cộng, phép toán trừ rất tốt, chỉ có một tội: rất hay chép nhầm đầu bài và cũng chỉ nhầm ở hai số 5, 6: đôi khi anh ấy chép nhầm 5 thành 6 hoặc ngược lại. Ví dụ số 25 có thể ghi thành số 26.

Như vậy, nếu đầu bài cho là $11 + 25$ thì MrT có thể có kết quả là 36 hoặc 37.

Yêu cầu: Cho 2 số nguyên dương a và b ($1 \leq a, b \leq 10^{12}$). Hãy xác định kết quả nhỏ nhất và lớn nhất mà MrT có được.

Dữ liệu: Vào từ tệp MISTAKE.INP gồm một dòng chứa 2 số nguyên dương a và b .

Kết quả: Đưa ra file MISTAKE.OUT trên một dòng 2 số nhỏ nhất và lớn nhất có thể nhận được.

Ví dụ:

MISTAKE.INP	MISTAKE.OUT
11 25	36 37

IV. ĐỀ IV

TỔNG QUAN VỀ ĐỀ THI

	Tên bài	File chương trình	File dữ liệu vào	File kết quả
Câu 1	Đàn cừu	SHEEPS.*	SHEEPS.INP	SHEEPS.OUT
Câu 2	Chẵn hay lẻ	EVENODD.*	EVENODD.INP	EVENODD.OUT
Câu 3	Hàng cây	TREELINE.*	TREELINE.INP	TREELINE.OUT
Câu 4	Truy tìm kho báu	TREELINE.*	TREELINE.INP	TREELINE.OUT

Dấu * được thay bằng PAS hoặc CPP của ngôn ngữ lập trình được sử dụng là Pascal hoặc C++.

Hãy lập trình giải các bài toán sau:

Câu 1. (7,0 điểm) Đàn cừu

Thắng vừa chuyển đến làm việc cho một nông trại. Thắng được giao nhiệm vụ quản lý đàn cừu của nông trại. Thắng nghĩ cần phải gán cho mỗi chú cừu một số thứ tự để việc theo dõi và chăm sóc đàn cừu được tốt hơn.

Đàn cừu có n con, mỗi con sẽ được Thắng gán cho một số thứ tự trong đoạn từ 1 đến n , hai chú cừu khác nhau được gán bởi hai số thứ tự khác nhau.

Yêu cầu:

Thắng cần biết khi gán số thứ tự cho đàn cừu thì phải viết bao nhiêu chữ số.

Dữ liệu vào: Từ tệp SHEEPS.INP có một dòng ghi số n ($1 \leq n \leq 10^9$).

Kết quả: Ghi ra tệp SHEEPS.OUT số chữ số mà Thắng phải viết.

Ví dụ:

SHEEPS.inp	SHEEPS.out
3	3
100	192
1000000	5888896

Ràng buộc:

- Subtask 1: 60% test có $N \leq 10^6$
- Subtask 2: 40% test có $N \leq 10^9$

Câu 2. (6,0 điểm) Chẵn hay lẻ

Cho hai số nguyên a và b . Nhiệm vụ của bạn là trả lời câu hỏi: Ước chung lớn nhất của chúng là số chẵn hay số lẻ?

- Nếu là số chẵn, in ra chuỗi **even**
- Nếu là số lẻ, in ra chuỗi **odd**

Dữ liệu: Vào từ tệp EVENODD.INP gồm:

- Dòng đầu tiên ghi số nguyên dương T ($T \leq 10^6$) - Số bộ dữ liệu.
- Dòng thứ 2 đến dòng $T+1$: mỗi dòng ghi số nguyên dương a, b ($a, b \leq 10^6$).

Kết quả: Ghi ra tệp EVENODD.OUT:

- Gồm T dòng, mỗi dòng ghi kết quả tìm được ứng với bộ dữ liệu đầu vào.

Ví dụ:

EVENODD.INP	EVENODD.OUT
3	odd
2 3	odd
1 1	even
4 4	

Ràng buộc:

- Có 20% số test ứng với 20% số điểm: $1 \leq T \leq 100$, $a, b \leq 100$.
- Có 20% số test khác ứng với 20% số điểm: $100 \leq T \leq 1000$, $a, b \leq 1000$.
- Có 60% số test còn lại không có ràng buộc gì thêm.

Câu 3. (4,0 điểm) Hàng cây

Lão phù thủy giam công chúa xứ sở thần tiên trong một lâu đài, biết bao hiệp sĩ đến cứu nàng nhưng đều có dễ. Chỉ có một con đường duy nhất vào lâu đài, trên con đường ấy có một hàng n cây, cây thứ i có độ cao là a_i .

Lão ta thích **Thuật toán** nên nghĩ ra một số nguyên k và thách thức các hiệp sĩ tìm ra dãy liên tục các cây có độ cao trung bình là k trong khoảng 1 giây. Nếu ai giải được thì người đó có thể cứu công chúa.



Yêu cầu: Cho n , k và dãy a_i . Hãy xác định dãy liên tục các cây *có độ cao trung bình là k và dài nhất*. Đưa độ dài của dãy cây tìm được ra. Nếu không tồn tại dãy cây có độ cao trung bình là k thì đưa ra một số 0.

Dữ liệu: Vào từ tệp TREELINE.INP gồm:

- Dòng đầu tiên chứa 2 số nguyên dương n và k ($n \leq 10^6$, $k \leq 10^9$).
- Dòng thứ 2 chứa n số nguyên dương a_1, a_2, \dots, a_n ($a_i \leq 10^9$).

Mỗi số ghi cách nhau ít nhất một dấu cách.

Kết quả: Đưa ra tệp TREELINE.OUT độ dài của dãy tìm được hoặc ghi ra 0 nếu không tồn tại dãy.

Ví dụ:

TREELINE.INP	TREELINE.OUT
7 4 1 3 4 5 6 8 7	3

Ràng buộc: Có 50% số test ứng với 50% số điểm của bài có $n \leq 10^3$.

Câu 4. (3,0 điểm) Truy tìm kho báu

Dante cùng nhóm bạn chơi trò chơi truy tìm kho báu. Bản đồ kho báu dẫn Dante đến một đầm lầy, theo đúng như bản đồ thì để tìm được kho báu thì Dante phải đi qua đầm lầy, biết rằng đầm lầy qua đầm lầy có một cây cầu gỗ gồm n nhịp. Các nhịp cầu được đánh số từ 1 đến n từ trái qua phải. Dante có thể bước một bước, hoặc bước hai bước. Tuy nhiên một số nhịp cầu đã bị thủng do cũ kỹ và Dante không thể bước chân lên được. Dante đứng ở một bên đầu cầu một và muốn biết có bao nhiêu cách để qua cầu. Dante nhờ bạn lập trình trả lời câu hỏi trên.

Dữ liệu vào: Từ tệp TREASURE.INP gồm:

- Dòng đầu tiên gồm 2 số nguyên dương n và k , là số nhịp của cây cầu và số nhịp cầu bị hỏng ($k < n \leq 10^5$).
- Dòng thứ hai gồm k số nguyên cho biết chỉ số của các nhịp cầu bị hỏng theo thứ tự tăng dần.

Kết quả: Ghi ra tệp TREASURE.OUT là phần dư của số cách Dante qua cầu khi chia cho 132020.

TREASURE.INP		TREASURE.OUT
4 2 3	2	0
90000 49000	1	127715

---Hết đề 4---

V. ĐỀ V

CONTEST#5: NINJA VÀ HOÀNG TỬ HIKARU GENJI

Khoảng những năm đầu của thế kỷ thứ VIII, tại Nhật Bản đã xuất hiện từ “Ninja” để chỉ một hoặc một nhóm người được đào tạo bài bản hoạt động bí mật. Họ có những kỹ năng chiến đấu đặc biệt, khả năng phi tiêu, khả năng ẩn mình trong thiên nhiên và đi trên nước...

Chữ Ninja trong tiếng Nhật là:

忍者

Ảnh: NINJA VÀ HOÀNG TỬ HIKARU GENJI

Câu 1. (7,0 điểm) Mật mã

Những Ninja thực hiện trao đổi thông tin với nhau theo cách riêng của họ. Trước khi trao đổi thông tin, họ sẽ dùng những mật mã để xác nhận đảm bảo người đối diện là Ninja cùng phe phái. Những mật mã này sẽ được tạo mới từng ngày bởi Shinobi – một Ninja đam mê số học.

Shinobi tạo ra mật mã bằng cách lấy tích 3 chữ số của cùng của số N^T (N , T tương ứng là ngày, tháng tạo mật mã).

Ví dụ: Hôm nay là ngày 19/3, $19^3 = 6859$ và tích 3 chữ số cuối cùng là 360. Vậy mật mã ngày 19/3 Shinobi tạo ra là: **360**.

Bạn hãy cho biết vào ngày N tháng T , thì mật mã Shinobi tạo ra là bao nhiêu?

Dữ liệu: Vào từ tệp NJCODE.INP gồm

một dòng duy nhất ghi số nguyên dương N và T ($N \leq 31$, $T \leq 12$).

Kết quả: Ghi ra tệp NJCODE.OUT mật mã Shinobi tạo ra trong ngày N tháng T .

Ví dụ:

NJCODE.INP	NJCODE.OUT	Giải thích
19 3	360	$19^3 = 6859$ Tích 3 số cuối là: $8 \cdot 5 \cdot 9 = 360$.

Câu 2. (6,0 điểm) Vinyl và những chiếc tiêu

Vinyl là ninja có kỹ năng phi tiêu thượng thừa. Ngoài những tiêu làm bằng kim loại hình sao nhiều cạnh, Vinyl còn có những chiếc tiêu làm từ cây trúc.

Trong những lúc rảnh rỗi, Vinyl vào rừng chọn các cây trúc để làm tiêu. Những chiếc tiêu của Vinyl có độ dài L . Trước mặt anh bây giờ có N cây trúc, cây thứ i có độ cao là $a[i]$.

Như vậy, có những cây trúc có thể tạo thành nhiều tiêu và không có các mẫu thừa. Nhưng cũng có các cây trúc khác sau khi tạo thành tiêu sẽ có các mẫu thừa.

Tất nhiên, Vinyl không muốn có các mẫu thừa này và anh ấy chỉ chặt các cây mà không tạo ra mẫu thừa.

Ví dụ: Có 3 cây trúc độ dài là 10, 11, 12 và độ dài tiêu $L = 2$.

Vinyl sẽ chặt cây 2 cây: cây 1 và cây 3. Tạo ra được 11 cây tiêu.

Yêu cầu: Hãy tính số tiêu mà Vinyl tạo được theo cách chặt cây của anh ấy.

Dữ liệu: Vào từ tệp DARTS.INP gồm:

- Dòng 1: Ghi số nguyên dương N và L ($N, L \leq 10^5$).
- Dòng 2: Ghi N số nguyên dương $a[i]$ ($a[i] \leq 10^9$).

Kết quả: Ghi ra tệp DARTS.OUT số tiêu Vinyl có được theo cách chặt cây của anh ấy.

Ví dụ:

DARTS.INP	DARTS.OUT
3 2 10 11 12	11

Câu 3. (4,0 điểm) Phần thưởng

Sau một nhiệm vụ đặc biệt thành công, Shinobi và các đồng đội được hoàng tử Hikaru Genji gọi đến trao thưởng. Mỗi ninja được tặng thưởng nhiều đồng vàng và tham dự trò chơi trúng thưởng.

Ở trò chơi này, hoàng tử có N chiếc hòm và chiếc thứ i ghi số nguyên $a[i]$. Mỗi ninja được chọn 3 chiếc hòm và điểm của lượt chơi là tích các chữ số ghi trên các hòm được chọn. Tất nhiên, ninja nào cũng muốn có điểm lớn nhất để nhận thêm phần thưởng của hoàng tử.

Hãy cho biết điểm lớn nhất của các ninja có được là bao nhiêu?

Ví dụ: $N = 5$, dãy số -1 2 3 4 5 \rightarrow diemmax = 60.

Dữ liệu: Vào từ tệp NJBONUS.INP gồm:

- Dòng 1: Ghi số nguyên dương N ($3 \leq N \leq 10^5$).
- Dòng 2: Ghi N số nguyên a_1, a_2, \dots, a_N ($a_i \leq 10^5$).

Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp NJBONUS.OUT điểm lớn nhất của các ninja có được.

NJBONUS.INP	NJBONUS.OUT
5 -1 2 3 4 5	60

Câu 4. (3,0 điểm) Hoàng tử Hikaru Genji

Có thông tin cho rằng hoàng tử Hikaru Genji cũng là một ninja giỏi chiến thuật. Việc lựa chọn các ninja của hoàng tử luôn được đánh giá rất chính xác và hiệu quả. Sa bàn đưa ra có dạng một ma trận hình chữ nhật kích thước $m \times n$, tại ô (i, j) có một ninja có chỉ số sức mạnh là $a[i, j]$.

Hoàng tử Hikaru Genji được phép chọn một hình vuông kích thước K trên ma trận và điểm của hoàng tử có được là tổng các chỉ số sức mạnh của các ninja được chọn.

Yêu cầu: Hãy cho biết điểm lớn nhất mà hoàng tử có được là bao nhiêu?

Dữ liệu: Vào từ tệp NJHIKARU.INP gồm:

- Dòng 1: Ghi số nguyên dương M, N và K ($M.N.K \leq 10^6, K \leq \min(M, N)$).
- M dòng sau, mỗi dòng ghi N số nguyên $a[i, j]$ ($a_{i, j} \leq 10^5$).

Mỗi số cách nhau một dấu cách.

Kết quả: Đưa ra file NJHIKARU.OUT điểm lớn nhất mà hoàng tử có được.

Ví dụ:

NJHIKARU.INP	NJHIKARU.OUT
4 3 2 1 3 2 1 1 1 0 1 2 1 0 1	7

---HẾT ĐỀ 5---

VI. ĐỀ VI

Câu 1. (7,0 điểm) Quân cờ may mắn

Hikaru có N quân cờ vây, quân cờ vây thứ i ghi số i . Ở trong các ván cờ, Hikaru lần lượt sử dụng K quân cờ may mắn được chọn sẵn.

Với tài năng của mình, Hikaru dễ dàng thắng đối thủ sau K bước đi và anh ấy muốn biết số ghi ở các quân cờ còn lại chưa được sử dụng.

Ví dụ: $N = 5$, dãy chỉ số ghi trên quân cờ là: 1, 2, 3, 4, 5.

$K = 3$, dãy các quân cờ được sử dụng: 1, 4, 5.

Các quân cờ chưa được sử dụng là: 2, 3.

Dữ liệu: Vào từ tệp GGAME.INP gồm

- Dòng 1: Ghi số nguyên dương N và K ($K < N \leq 10^6$).
- Dòng 2: Ghi K số nguyên dương a_1, a_2, \dots, a_K ($a_i \leq N$)

Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp GGAME.OUT dãy các quân cờ vây chưa được sử dụng theo thứ tự tăng.

Ví dụ:

GGAME.INP	GGAME.OUT
5 3	2 3
1 4 5	

Câu 2. (6,0 điểm) Hikaru tìm quân cờ

Hikaru có N quân cờ vây, quân cờ vây thứ i ghi số $a[i]$ đôi một khác nhau. Ở trong các ván cờ, Hikaru sử dụng K quân cờ may mắn được chọn sẵn.

Với tài năng của mình, Hikaru dễ dàng thắng đối thủ sau K bước đi. Tuy nhiên, do các quân cờ vây để úp mặt ghi số xuống dưới, nên anh ấy lật lần lượt các quân cờ để tìm được quân cờ cần lấy mất rất nhiều thời gian.

Bạn hãy lập trình giúp anh ấy tìm vị trí của K quân cờ may mắn nhé.

Ví dụ: $N = 5$, dãy 1, 3, 5, 6, 8.

$K = 3$, dãy các quân cờ may mắn 3, 6, 8.

Thứ tự các quân cờ may mắn trong dãy a là: 2, 4, 5.

Dữ liệu: Vào từ tệp GSEARCH.INP gồm:

- Dòng 1: Ghi số nguyên dương N và K ($K \leq N \leq 10^5$).
- Dòng 2: Ghi N số nguyên dương $a[i]$ đôi một khác nhau ($a[i] \leq 10^9$, $a[i] < a[i+1]$).
- Dòng 3: Ghi K số nguyên dương $b[j]$ ($b[j] \leq 10^9$) – số ghi trên các quân cờ may mắn.

Dữ liệu đảm bảo có nghiệm.

Kết quả: Ghi ra tệp GSEARCH.OUT số thứ tự của các quân cờ may mắn theo thứ tự đầu vào.

Ví dụ:

GSEARCH.INP	GSEARCH.OUT
5 3	2 4 5
1 3 5 6 8	
3 6 8	

Câu 3. (4,0 điểm) Hikaru chơi cờ vây

Vào một ngày không đẹp trời, Hikaru ngồi luyện cờ vây với sư phụ của mình. Không dễ để thắng được sư phụ của mình, nên Hikaru phải dùng chiến thuật chơi cầm hoà. Tức là khi bàn cờ không còn chỗ đặt quân cờ vây nữa thì sẽ ván đấu sẽ hoà. Tất nhiên sư phụ của anh ấy không nghĩ đơn giản như vậy. Khi quân cờ đã kín bàn, sư phụ yêu cầu anh ấy chọn một ô trên bàn cờ, phải nhặt hết quân cờ theo hàng và cột của ô được chọn. Sau đó hai thầy trò lại chơi tiếp.

Ngoài trời mưa vẫn rơi, Hikaru muốn kéo dài thời gian chơi cờ với sư phụ, nên anh ấy sẽ chọn ô để nhặt cờ sao cho các ô được nhặt có số quân đen nhiều nhất.

Hãy cho biết Hikaru nhặt nhiều nhất được bao nhiêu quân đen.

Dữ liệu: Vào từ tệp GREMOVE.INP gồm:

- Dòng 1: Ghi số nguyên dương N – kích thước của bàn cờ vây ($N \leq 3000$).
- Dòng N dòng tiếp theo, dòng thứ i ghi N số nguyên $a[i,j]$. $a[i,j] = -1$ là quân cờ trắng, $a[i,j] = 1$ là quân cờ đen.

Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp GREMOVE.OUT số cách chọn ô để nhặt được các quân cờ theo cách của Hikaru.

GREMOVE.INP	GREMOVE.OUT	Giải thích
4 1 -1 1 <u>1</u> -1 1 -1 -1 1 -1 -1 1 -1 -1 1 1	1	Chọn ô (1,4) có tổng hàng, tổng cột lớn nhất. Và chỉ có một cách chọn.

Câu 4. (3,0 điểm) Hikaru

Trên bàn cờ vây của Hikaru giờ chỉ còn có hai hàng phủ kín quân cờ đen và trắng. Hikaru chọn một số quân cờ liên tiếp ở hàng thứ nhất và một số quân cờ liên tiếp của hàng thứ hai sao cho hai dãy quân cờ được chọn giống nhau.

Ví dụ: Quân cờ vây đen ký hiệu là 1, quân cờ vây trắng ký hiệu -1.

Dãy 1: 1 -1 -1 1 1 1 -1 -1

Dãy 2: 1 1 -1 1 1 1 1 -1

Dãy quân cờ giống nhau dài nhất có độ dài 4 được chọn là: -1 1 1 1

Yêu cầu: Hãy cho biết độ dài dãy quân cờ giống nhau dài nhất mà Hikaru có được.

Dữ liệu: Vào từ tệp GSAME.INP gồm:

- Dòng 1: Ghi số nguyên dương N – kích thước của bàn cờ ($N \leq 3000$).
- Dòng 2 ghi N số nguyên $a[i]$.
- Dòng 3 ghi N số nguyên $b[j]$.

Mỗi số cách nhau một dấu cách, $a[i], b[j] \in \{-1; 1\}$.

Kết quả: Đưa ra file GSAME.OUT độ dài dãy quân cờ giống nhau dài nhất mà Hikaru có được.

Ví dụ:

GSAME.INP	GSAME.OUT
8 1 -1 <u>-1 1 1 1</u> -1 -1 1 1 <u>-1 1 1 1 1</u> 1 -1	4

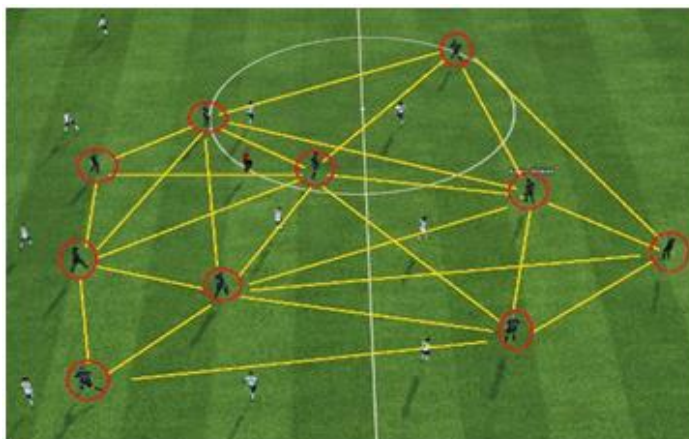
VII. ĐỀ VII

Câu 1. (7,0 điểm) Tiqui-Taca

Tiqui-Taca (tiếng Tây Ban Nha, phiên âm: Ti-ki Ta-ca) hoặc Tiki-taca là một khái niệm trong môn bóng đá. Đây là một loại hình chiến thuật thi đấu trên sân và cũng được xem là một trường phái bóng đá riêng với đặc trưng là lối chơi ưu tiên việc kiểm soát bóng và chuyền ngắn, kết hợp với di chuyển.

Về nguyên tắc, có thể diễn giải Tiqui-Taca là lối chơi kết hợp giữa "chuyền" (Tiqui) và "chạy" (Taca). Những đường chuyền của Tiqui-Taca đa phần ở cự ly trung bình - ngắn và tần số di chuyển không bóng của cầu thủ ở mức cao.

Cơ bản 2 yếu tố này đan xen với nhau, làm cho đội chơi Tiqui-Taca luôn kiểm soát được bóng và có cơ hội xuyên phá hàng phòng ngự đối phương.



Đội bóng Seiga đang luyện tập chuẩn bị cho mùa giải mới. Jindo muốn đội bóng của mình luyện tập lối chơi Tiqui-Taca nên đã chuẩn bị N đoạn dây, đoạn dây thứ i có độ dài $a[i]$.

Ở bài tập đầu tiên, mỗi nhóm 3 cầu thủ sẽ chọn 1 đoạn dây sao cho có thể gấp dây này vừa đủ thành một tam giác đều cạnh nguyên và các cầu thủ sẽ đứng ở các đỉnh của tam giác này chuyền bóng qua lại cho nhau.

Jindo muốn biết có bao nhiêu đoạn dây không được các cầu thủ chọn.

Ví dụ: $N = 5$, độ dài các đoạn giây là: 1, 3, 6, 9, 10. Có đoạn dây số 2, 3 và 4 được chọn.

Dữ liệu: Vào từ tệp TIKITAKA.INP gồm

- Dòng 1: Ghi số nguyên dương N ($N \leq 10^6$).
- Dòng 2: Ghi N số nguyên dương a_1, a_2, \dots, a_N ($a_i \leq 10^9$)

Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp TIKITAKA.OUT số lượng đoạn dây không được chọn.

Ví dụ:

TIKITAKA.INP	TIKITAKA.OUT
5	2
1 3 6 9 10	

Câu 2. (6,0 điểm) Số áo tập

Trên sân Seiga, trước khi vào buổi tập các cầu thủ sẽ chọn cho mình số áo tập. Do đội bóng Seiga còn khó khăn về kinh tế, nên trên sân tập chỉ có các số từ 0 đến 9 được cắt bằng vải màu, mỗi chữ số có nhiều chiếc. Mỗi cầu thủ sẽ chọn các chữ số này để ghép thành số áo của mình.

Để tránh lãng phí, Jindo cùng các bạn của mình lập bản đăng ký số áo tập. Mọi người có số áo khác nhau và có giá trị nhỏ hơn 100.

Ban quản lý sân muốn biết có bao nhiêu chữ số các loại được sử dụng theo bản đăng ký số áo của các cầu thủ. Ví dụ: $N = 7$; Các số áo đăng ký 1, 3, 4, 5, 9, 10, 18.

Số 0, 3, 4, 5, 8, 9 cần mỗi số 1 chiếc và số 1 cần 3 chiếc.

Dữ liệu: Vào từ tệp CLNUMBER.INP gồm:

- Dòng 1: Ghi số nguyên dương N ($N \leq 10^5$).
- Dòng 2: Ghi N số nguyên dương $a[i]$ đôi một khác nhau là số áo đăng ký của các cầu thủ ($a[i] < 100$). Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp CLNUMBER.OUT loại số và số lượng loại số cần chuẩn bị theo thứ tự tăng về loại số.

Ví dụ:

CLNUMBER.INP	CLNUMBER.OUT	Giải thích
7 1 3 4 5 9 10 18	0 1 1 3 3 1 4 1 5 1 8 1 9 1	Loại số 0 cần 1 số Loại số 1 cần 3 số

Câu 3. (4,0 điểm) Số áo may mắn

Để tự tin hơn cho các cầu thủ trong mùa giải mới, Jindo cùng các bạn được đăng ký mỗi người hai số áo thi đấu gồm: số áo chính thức và số áo dự bị. Tất nhiên số áo chính thức của các cầu thủ không được trùng nhau và số áo dự bị cũng vậy.

Huấn luyện viên đội Seiga cho rằng, độ may mắn theo số áo của cả đội là ước chung lớn nhất các số áo được chọn. Vì thế HLV sẽ quyết định chọn số áo chính thức hoặc số áo dự bị để cho trận đấu mở màn nếu độ may mắn theo số áo loại nào lớn hơn.

Ví dụ: Có $N = 5$ cầu thủ, số áo chính thức: 12, 3, 6, 15, 9, số áo dự bị: 20, 4, 8, 16, 18.

Thì bảng số áo dự bị sẽ được chọn thi đấu trận mở màn vì có ước chung lớn nhất là: 4. Trong khi bảng số áo chính thức có ước chung lớn nhất chỉ là 3.

Dữ liệu: Vào từ tệp CHNUMBER.INP gồm:

- Dòng 1: Ghi số nguyên dương N – số cầu thủ đội Seiga ($N \leq 10^5$).
- Dòng 2: Ghi N số nguyên $a[i]$ đôi một khác nhau là số áo đăng ký chính thức của cầu thủ thứ i .
- Dòng 3: Ghi N số nguyên $b[i]$ đôi một khác nhau là số áo đăng ký dự bị của cầu thủ thứ i .

Mỗi số cách nhau một dấu cách, các số áo nguyên dương.

Kết quả: Ghi ra tệp CHNUMBER.OUT chỉ số của đội được chọn và độ may mắn lớn nhất tìm được.

Qui ước, đội chính thức kí hiệu số 1, đội dự bị kí hiệu số 2.

CHNUMBER.INP	CHNUMBER.OUT	Giải thích
5 12 3 6 15 9 20 4 8 16 18	1 3	Đội chính thức được chọn và độ may mắn là 3.

Câu 4. (3,0 điểm) Mật mã phòng họp

Cả thế giới đang chao đảo vì dịch COVID-19, đội Seiga cũng vậy. Hàng ngày, HLV sẽ họp chiến thuật với các cầu thủ qua ứng dụng Zoom. Seiga và Meihou là hai địch thủ với nhau, để không bị lộ chiến thuật cho trận đấu với đội Meihou, HLV sẽ đặt mật khẩu đăng nhập vào phòng họp.

HLV có tư duy logic rất giỏi vì vậy mật khẩu của phòng họp do ông đặt ra cũng thật thú vị. Dựa trên danh sách số áo của các cầu thủ trong đội, ông sẽ chọn K số áo, sau đó ghép thành một số nguyên lớn nhất có thể. Đó sẽ là mật khẩu của phòng họp.

Jindo nhanh chóng nhờ các bạn của mình lập trình tìm giúp mật khẩu mà HLV đã đặt cho phòng họp.

Ví dụ: Có 7 cầu thủ với số áo là: 1, 3, 6, 8, 10, 18, 22 và $K = 3$

Các cầu thủ được chọn là: 10, 18, 22

Mật khẩu tạo được là: 221810

Yêu cầu: Hãy tìm mật khẩu mà HLV đã đặt cho phòng họp.

Dữ liệu: Vào từ tệp MRPASS.INP gồm:

- Dòng 1: Ghi số nguyên dương N, K lần lượt là số lượng cầu thủ của đội Seiga và số lượng số áo được chọn ($K \leq N \leq 10^5$).
- Dòng 2: Ghi N số nguyên dương $a[i]$ - số áo cầu thủ thứ i, đôi một khác nhau ($a[i] < 10^5$).

Mỗi số cách nhau một dấu cách.

Kết quả: Đưa ra file MRPASS.OUT mật khẩu HLV đã đặt cho phòng họp.

Ví dụ:

MRPASS.INP	MRPASS.OUT
7 3 1 3 6 8 10 18 22	221810

---Hết ĐỀ 7---

VIII. ĐỀ VIII

Câu 1. (7,0 điểm) Bộ số tam giác

Cho dãy số A gồm N số nguyên dương a_1, a_2, \dots, a_N . Một bộ ba số được gọi là bộ số tam giác, nếu ba số này là độ dài ba cạnh của một tam giác (tức là tổng hai số lớn hơn số còn lại).

Yêu cầu: Hãy đếm xem trong dãy A có bao nhiêu bộ số tam giác (a_i, a_j, a_k) với i, j, k đôi một khác nhau.

Dữ liệu: Vào từ tệp văn bản BSTG.INP gồm:

- Dòng đầu chứa số nguyên N ($3 \leq N \leq 1000$).
- Dòng tiếp theo chứa N số nguyên dương $a[i]$ ($a_i \leq 10^9$).

Kết quả: Ghi ra tệp văn bản BSTG.OUT số lượng bộ số tam giác trong dãy.

Ví dụ:

BSTG.INP	BSTG.OUT	Giải thích
5 4 3 1 5 7	3	Có 3 bộ số tam giác là (4, 3, 5), (4, 5, 7) và (3, 5, 7).

Giới hạn: Có 60% điểm của bài tương ứng với $n \leq 500$.

Câu 2. (6,0 điểm) Lát cắt cực đại

Jindo ghi N số nguyên a_1, a_2, \dots, a_N trên một vòng tròn (hình vẽ). Lát cắt vị trí K sẽ chia vòng tròn thành một dãy số mới. Với dãy số mới này, Jindo tính giá trị của lát cắt là tích lớn nhất của ba phần tử liên tiếp.

Ví dụ: Cho N = 11, dãy 6 -2 4 -1 -8 1 2 5 -3 -2 3

- Lát cắt K = 2, dãy mới là 4 -1 -8 1 2 5 -3 -2 3 6 -2

Giá trị của lát cắt này là $4 * (-1) * (-8) = 32$

- Lát cắt K = 4, dãy mới là -8 1 2 5 -3 -2 3 6 -2 4 -1

Giá trị của lát cắt này là $5 * (-3) * (-2) = 30$

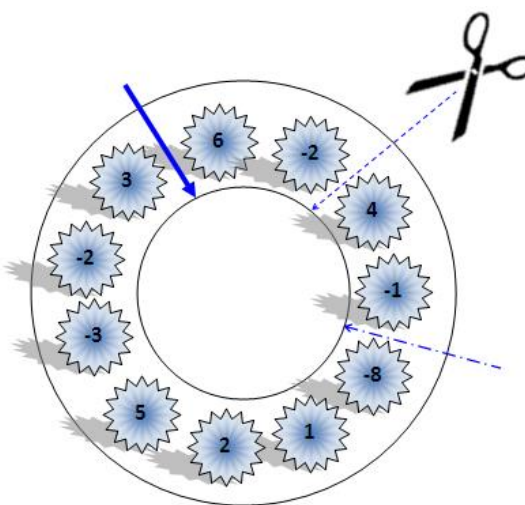
Yêu cầu: Jindo thử nghiệm m lần cắt, lần cắt thứ i có vị trí k_i . Hãy tìm lát cắt có giá trị lớn nhất.

Đưa giá trị lớn nhất đó ra.

Dữ liệu: Vào từ tệp văn bản ARRAYCUT.INP:

- Dòng đầu tiên chứa số nguyên dương N, M ($N * M \leq 10^6$).
- Dòng thứ 2 chứa N số nguyên a_1, a_2, \dots, a_N ($|a_i| \leq 1000$).
- Dòng thứ 3 chứa M số nguyên dương k_1, k_2, \dots, k_M ($k_i \leq N$).

Kết quả: Đưa ra tệp văn bản ARRAYCUT.OUT một số nguyên là giá trị lớn nhất trong các lát cắt.



ARRAYCUT.INP
11 2
6 -2 4 -1 -8 1 2 5 -3 -2 3
2 4

ARRAYCUT.OUT
32

Câu 3. (4,0 điểm) Trò chơi đập ếch

Sau buổi học đội tuyển Tin trên Zoom, các bạn học sinh rủ nhau ra Vincom chơi trò đập ếch. Màn hình trò chơi là một bảng lưới ô vuông hình chữ nhật kích thước $M \times N$. Trong mỗi ô của bảng có một chú ếch, trên lưng có in một số nguyên dương.

Khi người chơi cầm búa đập vào chú ếch thì tất cả các chú ếch có cùng số hiệu với chú ếch bị đập sẽ biến mất (kể cả chú ếch bị đập) và người chơi nhận được số điểm bằng số lượng ếch vừa bị biến mất.

Tất nhiên, khi cách chú ếch bị biến mất hết thì người chơi sẽ dừng lại.

Yêu cầu: Cho K , hãy tìm cách đập ếch để có tổng điểm lớn nhất sau K lần đập.

Dữ liệu: Vào từ tệp văn bản DAPECH.INP gồm:

- Dòng đầu ghi ba số nguyên dương M , N và K .
- M dòng tiếp theo, dòng thứ i ghi N số nguyên dương $a[i, j]$ là số ghi trên chú ếch ở ô (i, j) .

Các số cách nhau một dấu cách.

Kết quả: Ghi ra tệp văn bản DAPECH.OUT tổng điểm lớn nhất sau K lần đập ếch.

Ví dụ:

DAPECH.INP	DAPECH.OUT	Giải thích
4 6 2 1 4 3 3 2 4 2 4 2 1 4 1 2 3 4 4 1 1 1 1 2 3 4 4	15	- Lần 1: đập ô có số hiệu 4, đạt 8 điểm. - Lần 2: đập ô có số hiệu 1, đạt 7 điểm. Tổng điểm: $8 + 7 = 15$ điểm.

Ràng buộc: $1 \leq M, N \leq 3000$; $1 \leq K \leq M \times N$; Số nguyên dương ghi trên các chú ếch không vượt quá 10^5 .

Câu 4. (3,0 điểm) Hình vuông

Cho mảng A kích thước $N \times N$. Người ta định nghĩa:

- Đường chéo chính của mảng A là đường chéo đi qua các ô A_{ij} với $i = j$
- Đường chéo phụ của mảng A là đường chéo đi qua các ô A_{ij} với $i + j = N + 1$

Yêu cầu: Cho K , hãy tìm hình vuông con kích thước $K \times K$ có tổng các số trên đường chéo chính và đường chéo phụ là lớn nhất.

Dữ liệu: Vào từ tệp văn bản HINHVUONG.INP gồm:

- Dòng đầu tiên ghi số nguyên dương N và K ($N \leq 3000$, $K \leq N$).
- N dòng sau, mỗi dòng ghi N số nguyên A_{ij} ($|A_{ij}| \leq 10^6$).

Kết quả: Ghi ra tệp văn bản HINHVUONG.OUT tổng lớn nhất tìm được.

5	-2	-30	-20
10	6	4	1
-7	5	4	2
20	2	-3	2

Ví dụ:

HINHVUONG.INP	HINHVUONG.OUT
4 3 5 -2 -30 -20 10 6 4 1 -7 5 4 2 20 2 -3 2	36

Ràng buộc:

Có 30% điểm của bài tương ứng với $K = 1$;

Có 30% điểm của bài tương ứng với $N \leq 100$.

---Hết---

HƯỚNG DẪN GIẢI BỘ ĐỀ

I. ĐỀ I

Contest 1 _ Solve

Câu 1. (7,0 điểm) Tích các chữ số lớn nhất

Cho số nguyên dương N, M và P là các số có 3 chữ số. Tìm số có tích các chữ số lớn nhất. Đưa tích lớn nhất ra.

Thuật toán: Bài toán tính tích các chữ số của một số và tìm max của 3 số nguyên.

```
[ ]
Const
    fi = 'CALC.INP';
    fo = 'CALC.OUT';
Var
    m, n, p, res, x, y, z: longint;
Function tich(u: longint) : longint;
Var temp: longint;
Begin
    temp := 1;
    while u > 0 do
    begin
        temp := temp * (u mod 10);
        u := u div 10;
    end;
    exit(temp);
End;
Begin
    Assign(input, fi); reset(input);
    Assign(output, fo); rewrite(output);
    Readln(m, n, p);
    x := tich(m);
    y := tich(n);
    z := tich(p);
    res := x;
    if res < y then res := y;
    if res < z then res := z;
    write(res);
    close(input); close(output);
End.
```

Câu 2. (6,0 điểm) Số chính phương

Số chính phương (hay còn gọi là *số hình vuông*) là số tự nhiên có căn bậc 2 là một số tự nhiên.

Các số chính phương đầu tiên: 1, 4, 9, 16, 25....

Cho số nguyên dương N, hãy tìm số chính phương bé nhất lớn hơn N.

Ví dụ: N = 10 → Kết quả: 16.

Thuật toán:

N là số chính phương khi và chỉ khi căn bậc 2 của N là một số nguyên.

Hay $\text{sqrt}(N) = \text{trunc}(\text{sqrt}(N))$

Ta khởi tạo $\text{res} = N + 1$;

Chừng nào res không phải là số chính phương thì tăng res lên.

Lưu ý: Ta nên tổ chức thành chương trình con dạng hàm để kiểm tra số U có là số chính phương.

Hàm $\text{trunc}(x)$ là hàm lấy phần nguyên của x.

```
[ ]
Const
    fi = 'SQUA.INP';
    fo = 'SQUA.OUT';
Var
    n, res: longint;
Function Chinhphuong(u: longint) : boolean;
Begin
    exit(sqrt(u) = trunc(sqrt(u)));
End;
Begin
    Assign(input, fi); reset(input);
    Assign(output, fo); rewrite(output);
    Readln(n);
    res := n + 1;
    while Chinhphuong(res) = False do
        inc(res);
    end;
    write(res);
    close(input); close(output);
End.
```

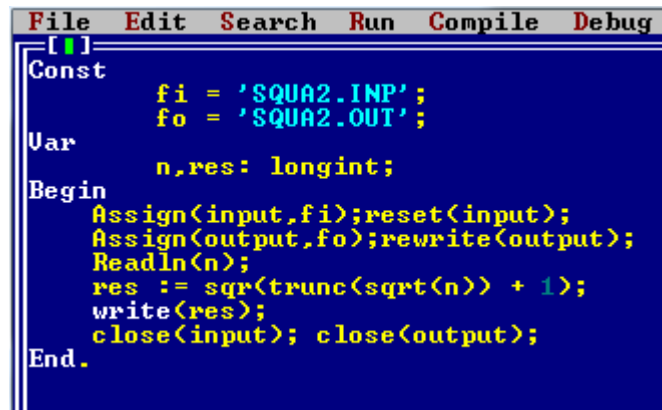
Thủ tục inc(res); tương đương lệnh $res := res + 1$;

Cách 2: Ta có nhận xét:

Với $N = 10$, phần nguyên của căn 10 bằng 3.

Ta nhận thấy $(3+1)^2 = 16$, từ đó ta tìm được công thức:

$Res := \text{sqr}(\text{trunc}(\text{sqr}(n))+1)$;



```
File Edit Search Run Compile Debug
[ ]
Const
    fi = 'SQUA2.INP';
    fo = 'SQUA2.OUT';
Var
    n, res: longint;
Begin
    Assign<input, fi>; reset<input>;
    Assign<output, fo>; rewrite<output>;
    Readln<n>;
    res := sqr(trunc(sqr(n)) + 1);
    write<res>;
    close<input>; close<output>;
End.
```

Câu 3. (4,0 điểm) Vị trí đẹp 2

Cho N và dãy a_1, a_2, \dots, a_N .

Vị trí i gọi là vị trí đẹp nếu i chia dãy số thành 2 đoạn mà tổng các phần tử của đoạn đầu gấp đôi tổng các phần tử của đoạn sau.

VD: $N = 6$; 1 7 4 2 3 4 \rightarrow Vị trí đẹp là: 4

Giải thích: $1 + 7 + 4 + 2 = 14 = 2 \cdot (3 + 4)$

Thuật toán:

Gọi $s[i]$ là tổng các phần tử từ 1 tới i .

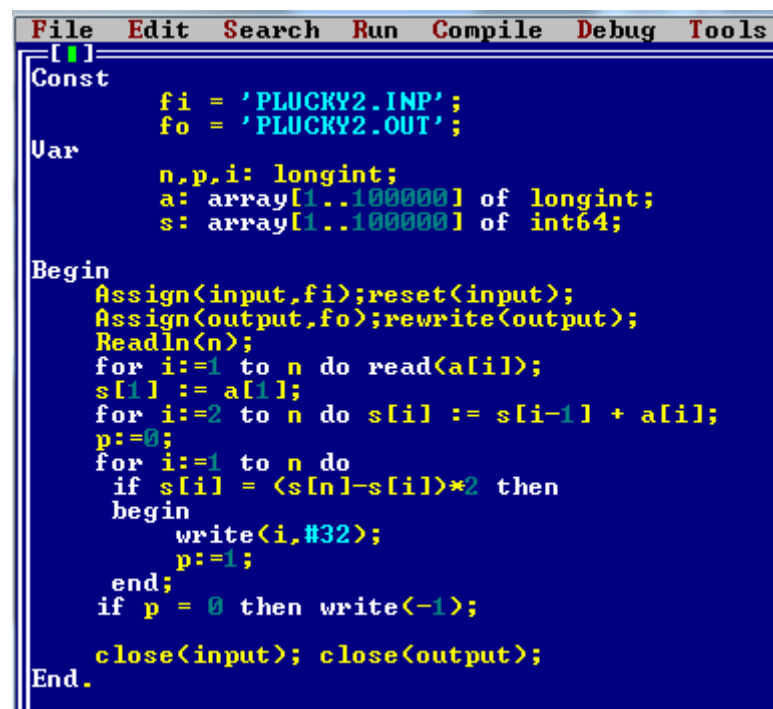
Khi đó, $s[1] = 1$; $s[i] = s[i-1] + a[i]$; với $i \geq 3$.

$S[i]$ được gọi là tổng tích lũy dùng để tính trước tổng của một đoạn của một mảng.

i là vị trí đẹp khi $s[i] = (s[n]-s[i])*2$;

Ta sử dụng biến p khởi tạo $p = 0$; khi phát hiện một nghiệm ta bật $p = 1$;

Sau khi duyệt qua tất cả các đoạn con, nếu p không bị bật lần nào chứng tỏ không có i nào thỏa mãn. Ta kết luận vô nghiệm và ghi ra -1.



```
File Edit Search Run Compile Debug Tools
[ ]
Const
    fi = 'PLUCKY2.INP';
    fo = 'PLUCKY2.OUT';
Var
    n, p, i: longint;
    a: array[1..100000] of longint;
    s: array[1..100000] of int64;
Begin
    Assign<input, fi>; reset<input>;
    Assign<output, fo>; rewrite<output>;
    Readln<n>;
    for i:=1 to n do read<a[i]>;
    s[1] := a[1];
    for i:=2 to n do s[i] := s[i-1] + a[i];
    p:=0;
    for i:=1 to n do
        if s[i] = (s[n]-s[i])*2 then
            begin
                write<i, #32>;
                p:=1;
            end;
    if p = 0 then write<-1>;
    close<input>; close<output>;
End.
```

Câu 4. (3,0 điểm) Ước nguyên tố

Cho số nguyên dương N, hãy tìm ước nguyên tố lớn nhất của N.

Ví dụ: N = 28; N chia hết cho 1, 2, 4, 7 và 28; → Kết quả: 7 là số nguyên tố lớn nhất là ước của 28.

Thuật toán :

Cách 1:

Duyệt qua tất cả các ước của N, nếu ước nào là số nguyên tố thì ta tìm max.

Khi i là ước của N thì N div i cũng là ước của N.

Ví dụ: 100 chia hết cho 2, thì 100 cũng chia hết cho phần còn lại là 50.

Vậy ta chỉ cần duyệt từ [1, căn N] nếu i là ước của N thì:

Nếu i nguyên tố thì so sánh và cập nhật lại res.

Nếu N div i nguyên tố thì so sánh và cập nhật lại res.

Lưu ý: $N \leq 10^{12}$ nên N phải là kiểu int64;

Cách 2: Ta phân tích N thành các thừa số nguyên tố. Khi đó thừa số nguyên tố cuối cùng chính là ước nguyên tố lớn nhất của N. Độ phức tạp **Thuật toán** $O(\sqrt{N})$.

```
Const
    fi = 'PRIDIU.IMP';
    fo = 'PRIDIU.OUT';

Var
    n, res: int64;
    i: longint;
Function nguyento(u: int64): boolean;
Var j: longint;
Begin
    if <u>=1 then exit<false>;
    if <u>=2 or <u>=3 then exit<true>;
    for j:=2 to trunc(sqrt(u)) do
        if u mod j = 0 then exit<false>;
    exit<true>;
End;
Begin
    Assign<input, fi>; reset<input>;
    Assign<output, fo>; rewrite<output>;
    Readln(n);
    res := low<longint>;
    for i:=1 to trunc(sqrt(n)) do
        if n mod i = 0 then
            begin
                if nguyento(i) and <res < i> then res := i;
                if nguyento(n div i) and <res < n div i> then res := n div i;
            end;
        write(res);
        close<input>; close<output>;
    End.
```

---Hết---

II. ĐỀ II

CONTEST2_SOLVE

Câu 1. (7,0 điểm) Chữ số lẻ lớn nhất

Cho số nguyên dương N, M và P là các số có 4 chữ số. Tìm số lẻ lớn nhất có mặt trong các số N hoặc M hoặc P.

Ví dụ: N = 3121; M = 3530; P = 6572 → res = 7.

Ta tìm số lẻ lớn nhất của mỗi số. Kết quả của bài toán là giá trị lớn nhất của các số lẻ tìm được.


```

[ ] NODD.pas
Uses math; //thu vien toan hoc
Const
    fi = 'NODD.INP';
    fo = 'NODD.OUT';
Var
    n,p,m,q,r2,r3,r4,res: longint;
Function Lemax(u: longint): byte; // Ham tra ve so le lon nhat cua u
Var
    temp: byte:=low(byte);
    z:byte;
Begin
    while u > 0 do
    begin
        z:= u mod 10;
        if z mod 2 = 1 then temp:= max(temp,z);
        u := u div 10;
    end;
    exit(temp);
End;
Begin
    Assign(input,fi);reset(input);
    Assign(output,fo);rewrite(output);
    Readln(m,n,p);
    r2:= Lemax(m);
    r3:= Lemax(n);
    r4:= Lemax(p);
    res := max(r2,max(r3,r4));
    if res = 0 then
        write(-1)
    else
        write(res);
    close(input); close(output);
End.

```

Câu 2. (6,0 điểm) Ước chung lớn nhất

Cho các số nguyên dương M, N và P.

Hãy tìm ước chung lớn nhất của M, N và P.

Ví dụ: M = 10; N = 25; P = 50 → res = 5

Thuật toán:

Sub1: Độ phức tạp **Thuật toán** $O(\min(m,n,p))$

Duyệt các i thuộc đoạn $[1, \min(m,n,p)]$, nếu m, n và p cùng chia hết cho i thì i cuối cùng là nghiệm của bài.

Sub2: $O(\log M + \log N + \log P)$

Ta tổ chức hàm tìm UCLN của hai số u và v.

Khi đó, ước chung lớn nhất của M, N và P bằng: $\text{UCLN}(m, \text{UCLN}(n,p))$.

```

File Edit Search Run Compile Debug Tools Options Window Help
[ ]
Const
    fi = 'THREEGCD.INP';
    fo = 'THREEGCD.OUT';
Var
    n,p,m,res: longint;
Function GCD(u, v: longint): longint; // Ham tra ve UCLN cua u va v
Var
    r: longint;
Begin
    while v > 0 do
        begin
            r := u mod v;
            u := v;
            v := r;
        end;
    exit(u);
End;
Begin
    Assign(input, fi); reset(input);
    Assign(output, fo); rewrite(output);
    Readln(m, n, p);
    res := GCD(m, GCD(n, p));
    write(res);
    close(input); close(output);
End.

```

Câu 3. (4,0 điểm) Giải Nhì

Bài toán: Tìm giá trị lớn nhì của dãy số.

Thuật toán:

Sub1: Sắp xếp dãy số thành dãy không giảm. Ta đi từ cuối dãy về đầu dãy, phần tử đầu tiên khác $a[n]$ là giá trị lớn thứ nhì.

Duyệt từ đầu dãy số đến cuối dãy số, nếu $a[i] = \text{resmax2}$ thì tăng res lên.

Đưa res ra.

Độ phức tạp **Thuật toán** $O(N \cdot \log N)$.

Sub2: Tìm resmax (giá trị lớn nhất của dãy số).

Tìm resmax2 bằng cách, duyệt từ đầu dãy đến cuối dãy, nếu $a[i] > \text{resmax2}$ và $a[i] \neq \text{resmax}$ thì cập nhập lại resmax2.

Lưu ý trường hợp không có giá trị lớn nhì của dãy phải ghi ra -1.

Độ phức tạp **Thuật toán**: $O(N)$.

```

File Edit Search Run Compile Debug Tools Options Window
[ ]
Uses math;
Const
    fi = 'SECOND.INP';
    fo = 'SECOND.OUT';
Var
    n, resmax, resmax2, i, p, res: longint;
    a: array[1..1000000] of longint;
Begin
    Assign(input, fi); reset(input);
    Assign(output, fo); rewrite(output);
    Readln(n);
    resmax := low(longint); //resmax bang mot gia tri cuc be'
    for i:=1 to n do
    begin
        read(a[i]);
        resmax := max(resmax, a[i]);
    end;
    resmax2 := low(longint);
    p := 0;
    for i:=1 to n do
        if (a[i] > resmax2) and (a[i] <> resmax) then
        Begin
            resmax2 := a[i];
            p := 1;
        End;
    if p = 0 then write(-1)
    else
    begin
        res := 0;
        for i:=1 to n do
            if resmax2 = a[i] then inc(res);
        write(res);
    end;
    close(input); close(output);
End.

```

Câu 4. (3,0 điểm) Tìm K

Cho số nguyên dương M. Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M.

Ví dụ: $M = 30 \rightarrow K = 56$

Thuật toán:

Sub1: Bắt đầu từ $K = 1$, chừng nào tích các chữ số của K khác M thì tăng K lên 1 đơn vị.

Nếu tíchchucso(k)=M thì đưa K ra, ngược lại ghi ra -1.

Thuật toán này chỉ đúng một phần.

Sub2:

Ta phân tích M thành các thừa số nguyên tố.

Khi đó, nếu có một thừa số nguyên tố có giá trị ≥ 10 thì đây là một trường hợp vô nghiệm. Vì tích các chữ số của K phải là tích của các số < 10 .

Ví dụ: $M = 26 = 2.13$

Trường hợp này ta ghi ra -1.

Hay nói cách khác, khi phân tích M thành thừa số nguyên tố thì chỉ gồm các số 2, 3, 5 và 7. Với các thừa số thu được ta lưu vào mảng C.

Khi đó ta tạo được mảng D từ mảng C bằng cách nhân dồn các phần tử của mảng C lại sao cho tích của các phần tử nhân được nhỏ hơn 10,

Sắp xếp mảng D thành dãy không giảm, và đưa mảng D ra là nghiệm của bài.

Ví dụ: $M = 7560 = 2.2.2.3.3.3.5.7 = 8.9.3.5.7 \rightarrow 35789$.

$M=30 = 2.3.5 = 6.5 = 5.6 \rightarrow 56$.

Bổ sung: Nếu $M = 1$ thì $K = 1$.

```
KMIN1.pas
Uses math; //thu vien toan hoc
Const
    fi = 'KMIN1.INP';
    fo = 'KMIN1.OUT';
Var
    m,nc,nd,i,p: longint; //nd, nc la so luong phan tu mang d va mang c
    c,d: array[1..1000000] of longint;
Procedure phantich(u: longint);
Var j:longint;
Begin
    for j:=2 to trunc(sqrt(u)) do
    begin
        while (u mod j = 0) and (u>0) do
        begin
            inc(nc);
            c[nc]:=j;
            u := u div j;
        end;
    end;
    if u>1 then
    begin
        inc(nc);
        c[nc]:=u;
    end;
End;
Procedure taomangd();
var j,z,temp:longint;
Begin
    j:=1;
    while j<=nc do
    begin
        temp:=1;
        while (temp*c[j] < 10) and (j<=nc) do
        begin
            temp:= temp*c[j];
            inc(j);
        end;
        inc(nd);
        d[nd]:=temp;
    end;
End;
```

```

Procedure sapxep();
var i,j, tem: longint;
Begin
  for i:=1 to nd-1 do
    for j:=i+1 to nd do
      if d[i]>d[j] then
        begin
          tem:=d[i];
          d[i]:=d[j];
          d[j]:=tem;
        end;
    end;
End;
Begin
  Assign(input,fi);reset(input);
  Assign(output,fo);rewrite(output);
  Readln(m);
  nc:=0;
  phantich(m);
  p:=1;
  for i:=1 to nc do
    if c[i]>=10 then
      begin
        p:=0;
        break;
      end;
  if p = 0 then write(-1)
  else
    Begin
      nd:=0;
      taomangd();
      sapxep();
      for i:=1 to nd do write(d[i]);
    end;
  close(input); close(output);
End.

```

80:18

---Hết---

III. ĐỀ III

CONTEST#3_SOLVE

Câu 1. (7,0 điểm) Ba anh em siêu nhân

Thuật toán:

Đổi thời điểm tính giờ A:B:C thành đơn vị giây: T giây.

Tìm min của x, y, z.

Thời điểm sớm nhất nhận bài: $time = T + \min(x,y,z)$ (giây)

Ta đổi time (giây) sang dạng giờ:phút:giây.

```

File Edit Search Run Compile Debug Tools Options Window Help
[ ]
uses math;
Const
    fi = 'GBOYS.INP';
    fo = 'GBOYS.OUT';
Var
    A,B,C, X, Y, Z,tmin,time: LONGINT;
function htos(u,v,t:longint): longint;    //doi gio:phut:giay sang giay
begin
    exit (u*3600+v*60+t);
end;
procedure stoh(var p, u, v, t: longint);    // doi p giay sang gio:phut:giay
begin
    u:= p div 3600;
    p:= p mod 3600;
    v:= p div 60;
    t:= p mod 60;
end;
Begin
    Assign(input,fi);reset(input);
    Assign(output,fo);rewrite(output);
    Readln(X,Y,Z);
    Readln(A,B,C);
    tmin := min(X,min(Y,Z));
    time := htos(A,B,C) + tmin;
    stoh(time,A,B,C);
    write(A,#32,B,#32,C);
    close(input); close(output);
End.

```

Câu 2. (6,0 điểm) Phép tính với phân số

Thuật toán:

Tổ chức hàm tìm ước chung lớn nhất của hai số nguyên dương u, v.

Khi tính tổng và hiệu hai phân số, ta cần quan tâm đến dấu của biểu thức ts và ms trước khi gọi hàm tìm UCLN.

Để rút gọn phân số ta có

$$ts = \frac{ts}{ucln(ts,ms)}; ms = \frac{ms}{ucln(ts,ms)}$$

Kết quả đưa ra: writeln(ts*dấu,#32,ms);

```

File Edit Search Run Compile Debug Tools Options Window Help
[ ]
uses math;
Const
    fi = 'MINIFRAC.INP';
    fo = 'MINIFRAC.OUT';
Var
    a,b,c,d,ts,ms,dau: longint;
Function gcd(u, v: longint): longint;
Var r: longint;
Begin
    while v > 0 do
        Begin
            r := u mod v;
            u := v;
            v := r;
        End;
    exit(u);
End;
procedure sum(u,v,x,y: longint; var ts, ms: longint);
    //ham tinh tong phan so u/v + x/y luu vao ts/ms
var p: longint;
begin
    ts := u*y + v*x;
    ms := v*y;
    dau:= 1;
    if ts*ms < 0 then dau:=-1;
    if (ts < 0) or (ms < 0) then
        begin
            ts:=abs(ts);
            ms:=abs(ms);
        end;
    p := gcd(ts,ms);
    ts := ts div p;
    ms := ms div p;
end;

procedure sub(u,v,x,y: longint; var ts, ms: longint);
    //ham tinh hieu phan so u/v - x/y luu vao ts/
var p: longint;
begin
    ts := u*y - v*x;
    ms := v*y;
    dau:= 1;
    if ts*ms < 0 then dau:=-1;
    if (ts < 0) or (ms < 0) then
        begin
            ts:=abs(ts);
            ms:=abs(ms);
        end;
    p := gcd(ts,ms);
    ts := dau*ts div p;
    ms := ms div p;
end;
Begin
    Assign(input,fi);reset(input);
    Assign(output,fo);rewrite(output);
    readln(a,b,c,d);
    sum(a,b,c,d,ts,ms);
    writeln(ts,#32,ms);
    sub(a,b,c,d,ts,ms);
    write(ts,#32,ms);
    close(input); close(output);
End.

```

Câu 3. (4,0 điểm) Bài toán đếm

Cho N, K và dãy a_1, a_2, \dots, a_N .

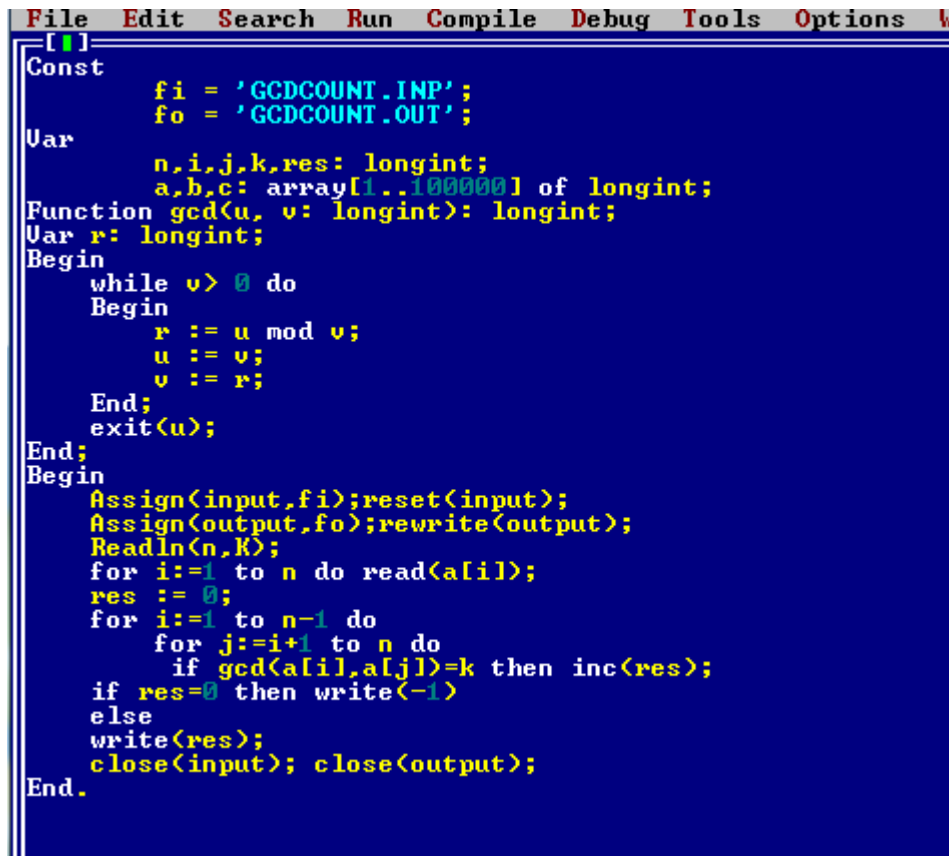
Hãy đếm số cặp (i, j) đôi một khác nhau sao cho ước chung lớn nhất của a_i và a_j bằng K .

Thuật toán:

Duyệt qua tất cả các cặp (i, j) , nếu $\gcd(a[i], a[j]) = k$ thì tăng res lên.

Nếu $res = 0$ thì ghi ra -1.

Độ phức tạp **Thuật toán:** $O(N^2)$.



```
File Edit Search Run Compile Debug Tools Options W
Const
    fi = 'GCDCOUNT.INP';
    fo = 'GCDCOUNT.OUT';
Var
    n, i, j, k, res: longint;
    a, b, c: array[1..100000] of longint;
Function gcd(u, v: longint): longint;
Var r: longint;
Begin
    while v > 0 do
        Begin
            r := u mod v;
            u := v;
            v := r;
        End;
    exit(u);
End;
Begin
    Assign(input, fi); reset(input);
    Assign(output, fo); rewrite(output);
    Readln(n, K);
    for i:=1 to n do read(a[i]);
    res := 0;
    for i:=1 to n-1 do
        for j:=i+1 to n do
            if gcd(a[i], a[j])=k then inc(res);
    if res=0 then write(-1)
    else
        write(res);
    close(input); close(output);
End.
```

Câu 4. (3,0 điểm) Nhầm lẫn

Thuật toán:

Ta tổ chức hàm thay thế các số 5 (nếu có) thành số 6 trong số $u \rightarrow$ ta thu được số u max.

Ta tổ chức hàm thay thế các số 6 (nếu có) thành số 5 trong số $u \rightarrow$ ta thu được số u min.

Khi đó, tổng nhỏ nhất là $\text{timmin}(a) + \text{timmin}(b)$;

Tổng lớn nhất là $\text{timmax}(a) + \text{timmax}(b)$.

Bài này ta có thể xử lý kiểu xâu.


```

File Edit Search Run Compile Debug Tools Options Window Help
[ ] MIS
Const
    fi = 'MISTAKE.INP';
    fo = 'MISTAKE.OUT';
Var
    a,b: longint;
Function timmax(u: longint): longint;    //thay so 5 bang so 6 trong u
var temp:longint=0;
    t:longint=1;
    p:longint;
Begin
    while u>0 do
        begin
            p:= u mod 10;
            if p=5 then p:=6;
            temp:=temp+p*t;
            t:=t*10;
            u := u div 10;
        end;
    exit(temp);
End;
Function timmin(u: longint): longint;    //thay so 6 bang so 5 trong u
var temp:longint=0;
    t:longint=1;
    p:longint;
Begin
    while u>0 do
        begin
            p:= u mod 10;
            if p=6 then p:=5;
            temp:=temp+p*t;
            t:=t*10;
            u := u div 10;
        end;
    exit(temp);
End;
Begin
    Assign(input,fi);reset(input);
    Assign(output,fo);rewrite(output);
    Readln(a,b);
    write(timmin(a)+timmin(b),#32,timmax(a)+timmax(b));
    close(input); close(output);
End.

```

---Hết---

IV. ĐỀ V

SOLVE_CONTEST#5

忍者

Câu 1. (7,0 điểm) Mật mã

Shinobi tạo ra mật mã bằng cách lấy tích 3 chữ số của cùng của số N^T (N, T tương ứng là ngày, tháng tạo mật mã).

Ví dụ: Hôm nay là ngày 19/3, $19^3 = 6859$ và tích 3 chữ số cuối cùng là 360. Vậy mật mã ngày 19/3 Shinobi tạo ra là: 360.

Bạn hãy cho biết vào ngày N tháng T, thì mật mã Shinobi tạo ra là bao nhiêu?

Thuật toán:

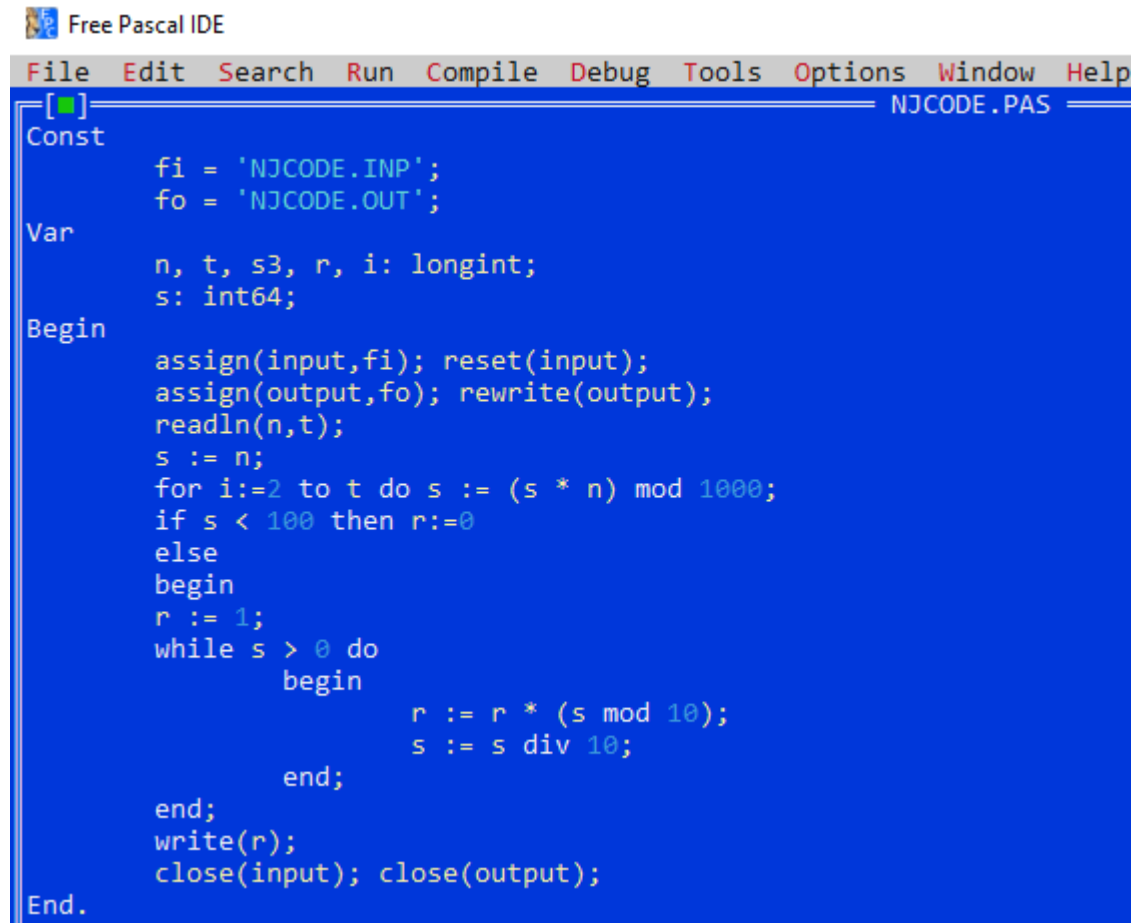
Ta tính $Res = N^T \bmod 1000$;

Lưu ý, khi tính N^T giá trị có thể vượt kiểu longint (int). Vì thế vừa tính N^T ta vừa mod cho 1000.

Còn trường hợp nữa, $N^T = 12045$ thì phép toán $N^T \bmod 1000 = 045 = 45$.

Do đó trước khi tính tích các chữ số của res ta cần xét, nếu $res < 100$ thì $kq = 0$ ngược lại ta đi tính các chữ số của res, lưu vào kq.

Đưa kq ra.



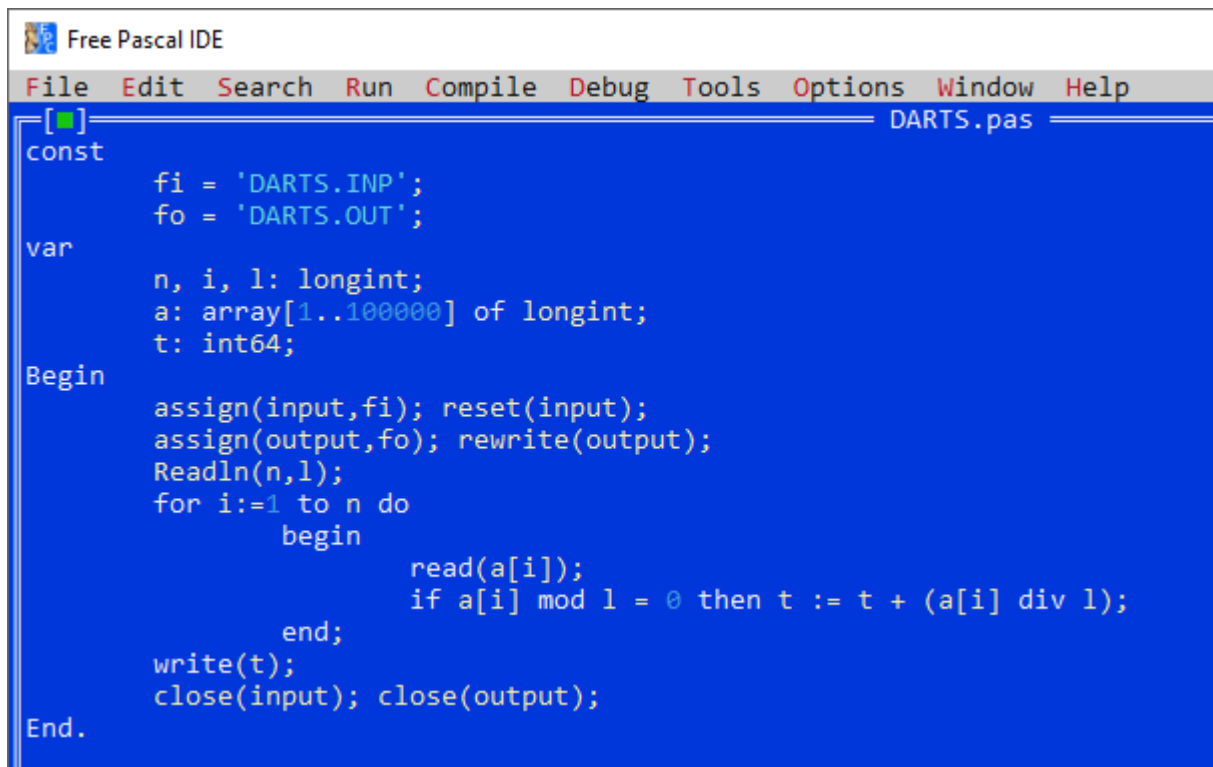
```
Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
NJCODE.PAS
Const
    fi = 'NJCODE.INP';
    fo = 'NJCODE.OUT';
Var
    n, t, s3, r, i: longint;
    s: int64;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n, t);
    s := n;
    for i:=2 to t do s := (s * n) mod 1000;
    if s < 100 then r:=0
    else
    begin
        r := 1;
        while s > 0 do
            begin
                r := r * (s mod 10);
                s := s div 10;
            end;
        end;
    end;
    write(r);
    close(input); close(output);
End.
```

Câu 2. (6,0 điểm) Vinyl và những chiếc tiêu

Thuật toán: Vinyl sẽ chọn các cây có độ cao chia hết cho L.

Do đó, kết quả của bài sẽ là tổng độ cao các cây được chặt chia cho L.

Lưu ý, kết quả của biến tổng S có thể vượt qua kiểu longint (int) do đó phải chọn kiểu dữ liệu lớn hơn cho S.



```
Free Pascal IDE
File Edit Search Run Compile Debug Tools Options Window Help
DARTS.pas
const
    fi = 'DARTS.INP';
    fo = 'DARTS.OUT';
var
    n, i, l: longint;
    a: array[1..100000] of longint;
    t: int64;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n, l);
    for i:=1 to n do
        begin
            read(a[i]);
            if a[i] mod l = 0 then t := t + (a[i] div l);
        end;
    write(t);
    close(input); close(output);
End.
```

Câu 3. (4,0 điểm) Phần thưởng

Thuật toán:

Để chọn được 3 phần tử có tích lớn nhất, ta sắp xếp dãy thành dãy không giảm bằng **Thuật toán** sắp xếp nhánh Quicksort. Độ phức tạp $O(n \log n)$

Ta dễ dàng chứng minh được kết quả của bài toán là giá trị lớn nhất $(a[1] * a[2] * a[n], a[n], a[n-1], a[n-2])$.

Lưu ý, tích của 3 số này có thể vượt kiểu longint (int) nên ta chọn biến kết quả kiểu dữ liệu lớn hơn.

```

[■] NJBONUS.PAS
Uses math;
Const
    fi = 'NJBONUS.INP';
    fo = 'NJBONUS.OUT';
    nmax = 100000+7;
Var
    a: array[1..nmax] of longint;
    n, i: longint;
    res11, res12, res: real;
Procedure Quicksort(L, H: Longint);
Var i, j, mid, temp: longint;
Begin
    if L >= H then exit;
    i := L;
    j := H;
    mid := (i+j) div 2;
    while i <= j do
    begin
        while a[i] < a[mid] do inc(i);
        while a[j] > a[mid] do dec(j);
        if i <= j then
        begin
            temp := a[i];
            a[i] := a[j];
            a[j] := temp;
            inc(i);
        end;
    end;
    if L <= j then quicksort(L, j);
    if i <= H then quicksort(i, H);
End;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    readln(n);
    for i:=1 to n do read(a[i]);
    quicksort(1, n);
    res11 := 1.0*a[1]*a[2]*a[n];
    res12 := 1.0*a[n-2]*a[n-1]*a[n];
    if res11 > res12 then res := res11
    else res := res12;
    write(res:0:0);
    close(input); close(output);
End.
47:58

```

Câu 4. (3,0 điểm) Hoàng tử Hikaru Genji

Thuật toán: Bài toán tìm hình vuông cạnh K có tổng lớn nhất.

Ta gọi $S[i,j]$ là tổng các ô từ ô (1, 1) đến ô (i, j).

$S[0,0] = 0$;

Ta dễ dàng chứng minh được $s[i,j] := s[i-1,j] + s[i,j-1] - s[i-1,j-1] + a[i,j]$;

Ta duyệt qua tất cả các hình vuông thuộc K, có góc trái trên (i, j), thì tổng các ô trong hình vuông này $temp := s[i+k-1,j+k-1] - s[i-1,j+k-1] - s[i+k-1,j-1] + s[i-1,j-1]$;

Khi đó kết quả $res := \max(res, temp)$;

Lưu ý, vì $s[i,j]$, res , $temp$ có thể rất lớn, nên ta chọn kiểu dữ liệu lớn hơn cho các biến này.
Độ phức tạp $O(M.N)$.

```
File Edit Search Run Compile Debug Tools Options Window Help
NJHIKARU.PAS
[■]
Var
    a: array[1..nmax,1..nmax] of longint;
    s: array[0..nmax,0..nmax] of int64;
    n, i,m,k,j: longint;
    res,temp: int64;
Begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(m,n,k);
    for i:=1 to m do
        for j:=1 to n do read(a[i,j]);
    s[0,0] := 0;
    for i:= 1 to m do
        for j:=1 to n do
            s[i,j] := s[i-1,j] + s[i,j-1] -s[i-1,j-1] + a[i,j];
    temp:=low(int64);
    for i:=1 to m-k+1 do
        for j:=1 to n-k+1 do
            begin
                temp := s[i+k-1,j+k-1]-s[i-1,j+k-1]-s[i+k-1,j-1]+s[i-1,j-1];
                res := max(res, temp);
            end;
    write(res);
    close(input); close(output);
End.
* 27:44
```

---Hết---

V. ĐỀ VI

SOLVE_CONTEST#6

Câu 1. (7,0 điểm) Quân cờ may mắn

Thuật toán: Ta sử dụng kỹ thuật mảng đánh dấu.

Dãy a là dãy các phần tử liên tiếp từ 1 tới N : 1, 2, 3, 4, 5, ..., N .

Gọi $F[i]$ = false/true tương ứng với i chưa được chọn/ đã được chọn.

Ban đầu ta phủ mảng F bởi giá trị False: `fillchar(f,sizeof(f),false);`

Duyệt từ đầu mảng b (mảng các quân cờ may mắn) đến cuối mảng, ta gán $f[b[j]]$ = true;

Sau đó ta duyệt từ 1 tới N , nếu $f[i]$ = false thì đưa i ra.

```

[■] GGAME.PAS
const
    fi = 'GGAME.INP';
    fo = 'GGAME.OUT';
var
    n, i, k: longint;
    b: array[1..100000] of longint;
    F: array[1..100000] of boolean;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n, k);
    for i:=1 to k do read(b[i]);
    fillchar(f, n+1, false);
    for i:=1 to k do
        f[b[i]] := true;
    for i:=1 to n do
        if f[i] = false then write(i, #32);
    close(input); close(output);
End.

```

Câu 2. (6,0 điểm) Hikaru tìm quân cờ

Hikaru có N quân cờ vây, quân cờ vây thứ i ghi số $a[i]$ đôi một khác nhau. Ở trong các ván cờ, Hikaru sử dụng K quân cờ may mắn được chọn sẵn.

Thuật toán: Ta sử dụng **Thuật toán** tìm kiếm nhị phân

Đánh giá: Dãy $a[i]$ là dãy tăng ngặt, vì thế để nhanh chóng tìm được vị trí của một phần tử trong dãy ta sử dụng **Thuật toán** tìm kiếm nhị phân.

Duyệt từ đầu mảng b đến cuối mảng b, với mỗi phần tử ta $\text{tknp}(b[j])$ trong dãy a. Nếu tìm thấy thì trả về vị trí của phần tử đó.

Vì dữ liệu đảm bảo có nghiệm nên ta không cần xét trường hợp suy biến là toàn bộ $b[j]$ không có trong dãy a.

Độ phức tạp **Thuật toán:** $O(N \log N)$

```

[■] GSEARCH.PAS
Var L, H, mid: longint;
Begin
    L:= 1;
    H:= n;
    while L<=H do
    begin
        mid:= (L+H) div 2;
        if a[mid] = u then exit(mid);
        if a[mid] > u then H := mid - 1
        else L:= mid + 1;
    end;
    if L > H then exit(-1);
End;
Begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    Readln(n,k);
    for i:=1 to n do read(a[i]);
    for i:=1 to k do read(b[i]);
    for i:=1 to k do
        write(tkn(b[i]),#32);
    close(input); close(output);
End.

```

Câu 3. (4,0 điểm) Hikaru chơi cờ vây

Thuật toán: Bài toán tìm hàng và cột có tổng lớn nhất.

Gọi $h[i]$ là tổng của hàng i .

$C[j]$ là tổng của cột j .

Duyệt qua tất cả các ô (i,j) trên a , với mỗi ô ta đi tính $temp = h[i] + c[j] - a[i,j]$. Ta phải trừ $a[i,j]$ bởi vì đây là giao của hàng và cột, nên nó đã bị tính 2 lần.

Vì quân đen ký hiệu bằng 1, quân trắng ký hiệu -1. Để chọn được nhiều quân đen nhất thì ta chọn ô có $temp$ lớn nhất.

Với mỗi tổng tính được ta cập nhật với $res = \max(res, temp)$;

Ta duyệt lại một lần với các ô (i, j) , nếu $h[i]+c[j]-a[i,j] = res$ thì ta tăng biến đếm số lượng lên.

Đưa biến đếm số lượng ra.

Độ phức tạp $O(N^2)$.

```

uses math;
const
    fi = 'GREMOVE.INP';
    fo = 'GREMOVE.OUT';
var
    n, i, j, res, temp, resmax: longint;
    a: array[1..3000, 1..3000] of longint;
    h, c: array[0..3000] of longint;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n);
    for i:=1 to n do
        for j:=1 to n do read(a[i, j]);

    for i:=1 to n do
        begin
            temp:=0;
            for j:=1 to n do temp:=temp+a[i, j];
            h[i]:=temp;
        end;
    for j:=1 to n do
        begin
            temp:=0;
            for i:=1 to n do temp:=temp+a[i, j];
            c[j]:=temp;
        end;
    resmax := low(longint);
    for i:=1 to n do
        for j:=1 to n do
            resmax := max(resmax, h[i]+c[j]-a[i, j]);
    res := 0;
    for i:=1 to n do
        for j:=1 to n do
            if h[i]+c[j]-a[i, j]=resmax then inc(res);
    write(res);
    close(input); close(output);
End.

```

Câu 4. (3,0 điểm) Hikaru

Thuật toán: Tìm dãy con chung liên tiếp dài nhất của dãy a và dãy b.

Gọi $F[i, j]$ là dãy con chung liên tiếp dài nhất kết thúc tại i của dãy a và kết thúc tại j của dãy b.

Khi đó, $f[0, 0] = 0$ và $f[i, j] = f[i-1, j-1] + 1$ nếu $a[i] = b[j]$, ngược lại thì $f[i, j] = 0$.

Kết quả của bài toán là $\max(f[i, j])$.


```

[■]
uses math;
const
    fi = 'GSAME.INP';
    fo = 'GSAME.OUT';
var
    n, i, j, res: longint;
    a, b: array[1..100000] of longint;
    f: array[0..3000, 0..3000] of longint;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n);
    for i:=1 to n do read(a[i]);
    for i:=1 to N do read(b[i]);
    f[0,0] := 0;
    for i:=1 to n do
        for j:=1 to n do
            if a[i]=b[j] then
                f[i,j] := f[i-1,j-1] + 1
            else
                f[i,j] := 0;
        for i:=1 to n do
            for j:=1 to n do
                res := max(res, f[i,j]);
    write(res);
    close(input); close(output);
End.

```

Hoặc Code C++ có truy vết:

```

#include <bits/stdc++.h>
#define ll long long
#define Pii pair<int, int>
#define Piii pair<int, Pii>
#define _mp make_pair
#define _pb push_back
#define st first
#define nd second
#define whole(x) x.begin(), x.end()
#define Reset(x) memset(x, 0, sizeof(x))
#define PI acos(-1)
#define TASK "GSAME"
const int M = 3008;
const int oo = 1e9+7;
using namespace std;

// mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());

int n, a[M], b[M], Res = 0, f[M][M], Trace = 0;

```

```

int main(int argc, char const *argv[]){
//  freopen("test.txt", "r", stdin);
    freopen(TASK".inp", "r", stdin);
    freopen(TASK".out", "w", stdout);
    ios_base::sync_with_stdio(false); cin.tie(0);
    cin >> n;
    for(int i = 1; i ≤ n; i++) cin >> a[i];
    for(int i = 1; i ≤ n; i++) cin >> b[i];

    f[0][0] = 0;
    for(int i = 1; i ≤ n; i++)
        for(int j = 1; j ≤ n; j++){
            f[i][j] = 0;
            if(a[i] == b[j])
                f[i][j] = f[i-1][j-1] + 1;
//            Res = max(Res, f[i][j]);
            if(f[i][j] > Res) Res = f[i][j], Trace = i;
        }
    cout << Res << "\n";
    for(int i = Res-1; i ≥ 0; i--) cout << a[Trace - i] << ' ';
    return 0;
}

```

---Hết---

VI. ĐỀ VII

SOLVE_CONTEST#7

Câu 1. (7,0 điểm) Tiqui-Taca

Thuật toán: Đếm số phần tử chia hết cho 3, lưu vào res. Kết quả là N-res.

Độ phức tạp: $O(N)$.

```
File Edit Search Run Compile Debug Tools Opti
[■]
const
    fi = 'TIKITAKA.INP';
    fo = 'TIKITAKA.OUT';
var
    n, i, res: longint;
    A: array[1..1000000] of longint;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n);
    for i:=1 to n do read(a[i]);
    res := 0;
    for i:=1 to n do
        if a[i] mod 3 = 0 then inc(res);
    write(n-res);
    close(input); close(output);
End.
```

Câu 2. (6,0 điểm) Số áo tập

Thuật toán: Đếm phân phối các chữ số của a[i].

Ta dùng mảng F[i] lưu số lần xuất hiện số i.

Ta duyệt các phần tử, đến phần tử thứ i ta đếm phân phối (a[i]).

Khi đó, các phần tử có F[i] khác 0 là loại i và số lượng số f[i] cần chuẩn bị.

```
File Edit Search Run Compile Debug Tools Options W
[■]
const
    fi = 'CLNUMBER.INP';
    fo = 'CLNUMBER.OUT';
var
    n, i, res: longint;
    A: array[1..1000000] of longint;
    F: array[0..9] of longint;
Procedure phantich(u: longint);
Begin
    while u > 0 do
        begin
            inc(F[u mod 10]);
            u := u div 10;
        end;
End;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n);
    for i:=1 to n do read(a[i]);
    fillchar(f, sizeof(f), 0);
    for i:=1 to n do
        phantich(a[i]);
    for i:=0 to 9 do
        if f[i] <> 0 then writeln(i, #32, f[i]);
    close(input); close(output);
End.
```

Câu 3. (4,0 điểm) Số áo may mắn

Thuật toán:

Để tìm UCLN của số u và v ta sử dụng **Thuật toán** O'c lit.

Với dãy $a[i]$ ta tìm UCLN của dãy này. Tương tự, ta tìm được UCLN của dãy $b[i]$.

Khi đó, nếu $res1 \geq res2$ thì đưa (1, $res1$) ra; ngược lại đưa (2, $res2$) ra.

Độ phức tạp **Thuật toán**: $O(N \cdot \log \log N)$

```
File Edit Search Run Compile Debug Tools Options Wi
[■]
const
    fi = 'CHNUMBER.INP';
    fo = 'CHNUMBER.OUT';
var
    n, i: longint;
    res1, res2: int64;
    A, B: array[1..100000] of longint;
Function gcd(u, v: int64): int64;
Var r: int64;
Begin
    while v > 0 do
    begin
        r := u mod v;
        u := v;
        v := r;
    end;
    exit(u);
End;
Begin
    assign(input, fi); reset(input);
    assign(output, fo); rewrite(output);
    Readln(n);
    for i:=1 to n do read(a[i]);
    for i:=1 to n do read(b[i]);
    res1 := a[1];
    for i:=2 to n do
        res1 := gcd(res1, a[i]);
    res2 := b[1];
    for i:=2 to n do
        res2 := gcd(res2, b[i]);
    if res1 >= res2 then write(1, #32, res1)
    else write(2, #32, res2);
    close(input); close(output);
End.
```

Câu 4. (3,0 điểm) Mật mã phòng họp

Thuật toán: Để thu được số có giá trị lớn nhất ta sẽ chọn K phần tử lớn nhất bằng cách sắp xếp dãy a thành dãy không giảm.

Sau đó sắp xếp K phần tử này theo *thứ tự từ điển* giảm dần.

Dãy K phần tử sau khi sắp xếp là nghiệm của bài này.

Lưu ý: Do $N \leq 10^5$, nên để sắp xếp các phần tử ta sử dụng **Thuật toán** Quicksort với độ phức tạp

Thuật toán là $O(n \log n)$.

```

[■]
const
    fi = 'MRPASS.INP';
    fo = 'MRPASS.OUT';
var
    n, i, K: longint;
    A: array[1..100000] of longint;
Procedure Quicksort1(L, H: Longint);
Var i, j, mid, temp: longint;
Begin
    if L >= H then exit;
    i := L;
    j := H;
    mid := (i+j) div 2;
    while i <= j do
    begin
        while a[i] < a[mid] do inc(i);
        while a[j] > a[mid] do dec(j);
        if i <= j then
        begin
            temp := a[i];
            a[i] := a[j];
            a[j] := temp;
            inc(i);
            dec(j);
        end;
    end;
    if L <= j then quicksort1(L, j);
    if i <= H then quicksort1(i, H);
End;

```

```

[■]
Function sosanh(u, v: longint): boolean;
Var s1,s2: string;
Begin
    str(u,s1); str(v,s2);
    exit(s1<s2);
End;
Procedure Quicksort2(L, H: Longint);
Var i,j, mid,temp: longint;
Begin
    if L>=H then exit;
    i:= L;
    j:= H;
    mid:= (i+j) div 2;
    while i<=j do
    begin
        while sosanh(a[i],a[mid]) do inc(i);
        while sosanh(a[mid],a[j]) do dec(j);
        if i <=j then
        begin
            temp := a[i];
            a[i] := a[j];
            a[j] := temp;
            inc(i);
            dec(j);
        end;
    end;
    if L <= j then quicksort2(L, j);
    if i <= H then quicksort2(i,H);
End;

```

```

Begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    Readln(n,K);
    for i:=1 to n do read(a[i]);
    Quicksort1(1,N);
    Quicksort2(n-k+1,N);
    for i:=n downto n-k+1 do
        write(a[i]);
    close(input); close(output);
End.

```

Hoặc code c++

```

#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define Pii pair < int, int >
#define Piii pair < int, Pii >
#define Vi vector < int >
#define Vii vector < Pii >
#define pb push_back

```

```

#define mp make_pair
#define reset(x) memset(x, 0, sizeof(x))
#define fori(i, a, b) for(int i = a; i ≤ b; i++)
#define ford(i, a, b) for(int i = a; i ≥ b; i--)
#define fora(xx, gg) for(auto xx : gg)
#define fi first
#define se second
#define Kurumi "MRPASS"
#define nmax 100005
const ll OO = 1e9+7 ;

```

```

using namespace std;
int n,k,a[nmax];
string shido = "";
string CC[nmax];
int read(){
    int res = 0;
    char c = getchar();
    bool Neg = false;
    while(c == ' ' || c == '\n') c = getchar();
    if(c == '-'){
        Neg = true;
        c = getchar();
    }
    while('0' ≤ c && c ≤ '9'){
        res = res * 10 + c - '0';
        c = getchar();
    }
    if(Neg) return -res;
    return res;
}

```

```

string solve(int a){
    string xx = "";
    while(a > 0){
        string x;
        x = a % 10 + 48;
        xx = x + xx;
        a /= 10;
    }
}

```

```

    return xx;
}
void kurumi(){
    cin >> n >> k;
    for(int i = 1; i ≤ n; i++){
        cin >> a[i];
        sort(a + 1, a + n + 1);
        int d = 1;
        for(int i = n; i > n - k; i--){
            string s;
            s = solve(a[i]);
            CC[d] = s;
            d++;
        }
        sort(CC + 1, CC + d + 1);
        for(int i = d; i >= 1; i--){
            shido += CC[i];
        }
        cout << shido << '\n';
    }

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
    freopen(Kurumi".inp","r",stdin);
    freopen(Kurumi".out","w",stdout);
    kurumi();
    return 0;
}

```

---Hết---

VII. ĐỀ VIII

SOLVE_CONTEST#8

Câu 1. (7,0 điểm) Bộ số tam giác

Thuật toán: Dùng 3 vòng for để tìm các cặp giá trị $a[i]$ khác nhau thỏa mãn điều kiện của tam giác.

Độ phức tạp: $O(N^3)$.

```

#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long

```



```

#define Pii pair < int, int >
#define Piii pair < int, Pii >
#define Vi vector < int >
#define Vii vector < Pii >
#define pb push_back
#define mp make_pair
#define reset(x) memset(x, 0, sizeof(x))
#define fori(i, a, b) for(int i = a; i ≤ b; i++)
#define ford(i, a, b) for(int i = a; i ≥ b; i--)
#define fora(xx, gg) for(auto xx : gg)
#define fi first
#define se second
#define Kurumi "BSTG"
#define nmax 1005
const ll OO = 1e9+7 ;

using namespace std;
int n,a[nmax];
int shido = 0;
int read(){
    int res = 0;
    char c = getchar();
    bool Neg = false;
    while(c == ' ' || c == '\n') c = getchar();
    if(c == '-'){
        Neg = true;
        c = getchar();
    }
    while('0' ≤ c && c ≤ '9'){
        res = res * 10 + c - '0';
        c = getchar();
    }
    if(Neg) return -res;
    return res;
}
int random(int minn, int maxx){
    return minn + rand() % (maxx - minn + 1);
}

void kurumi(){

```

```

    cin >> n;
    for(int i = 1; i ≤ n; i++)
        cin >> a[i];
    sort(a + 1, a + n + 1);
    for(int i = 1; i ≤ n - 2; i++){
        for(int j = i + 1; j ≤ n - 1; j++){
            for(int k = j + 1; k ≤ n; k++){
                if(a[i] + a[j] > a[k] && abs(a[i] - a[j]) < a[k]){
                    shido++;
                }
            }
        }
    }
    cout << shido << '\n';
}

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
    freopen(Kurumi".inp", "r", stdin);
    freopen(Kurumi".out", "w", stdout);
    kurumi();
    return 0;
}

```

Câu 2. (6,0 điểm) Lát cắt cực đại

Thuật toán: Với mỗi lần xét k ta sẽ cắt mảng a[i] và tìm tích lớn nhất của ba phần tử liên tiếp trong mỗi lần cắt, kết quả cuối cùng là giá trị lớn nhất trong các giá trị tích tìm được.

```

#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define Pii pair < int, int >
#define Piii pair < int, Pii >
#define Vi vector < int >
#define Vii vector < Pii >
#define pb push_back
#define mp make_pair
#define reset(x) memset(x, 0, sizeof(x))
#define fori(i, a, b) for(int i = a; i ≤ b; i++)
#define ford(i, a, b) for(int i = a; i ≥ b; i--)
#define fora(xx, gg) for(auto xx : gg)

```

```

#define fi first
#define se second
#define Kurumi "ARRAYCUT"
#define nmax 1000007
const ll OO = 1e9+7 ;

using namespace std;
int n,m,a[nmax],b[nmax];
int c[nmax];
int shido = 0;
int read(){
    int res = 0;
    char c = getchar();
    bool Neg = false;
    while(c == ' ' || c == '\n') c = getchar();
    if(c == '-'){
        Neg = true;
        c = getchar();
    }
    while('0' ≤ c && c ≤ '9'){
        res = res * 10 + c - '0';
        c = getchar();
    }
    if(Neg) return -res;
    return res;
}

void kurumi(){
    cin >> n >> m;
    for(int i = 1; i ≤ n; i++){
        cin >> a[i];
    }
    for(int i = 1; i ≤ m; i++){
        cin >> b[i];
    }
    for(int j = 1; j ≤ m; j++){
        int x = b[j];
        for(int i = 1; i ≤ n; i++){
            c[i] = a[i];
        }
    }
}

```

```

    for(int i = 1; i ≤ n - x; i++){
        c[i] = a[i + x];
    }
    for(int i = n - x + 1; i ≤ n; i++){
        c[i] = a[i + x - n];
    }
    int CC = 0;
    for(int i = 1; i ≤ n - 2; i++){
        CC = max(CC, c[i] * c[i + 1] * c[i + 2]);
    }
    shido = max(shido, CC);
}
cout << shido << '\n';
}

int main()
{
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
    freopen("Kurumi".inp, "r", stdin);
    freopen("Kurumi".out, "w", stdout);
    kurumi();
    return 0;
}

```

Câu 3. (4,0 điểm) Trò chơi đập ếch

Thuật toán: Ta sẽ đếm xem với mỗi giá trị sẽ xuất hiện bao nhiêu lần, sau đó sắp xếp các giá trị này theo 1 thứ tự. Kết quả cần tìm là tổng của K giá trị xuất hiện nhiều nhất vừa tìm được.

```

#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define Pii pair < int, int >
#define Piii pair < int, Pii >
#define Vi vector < int >
#define Vii vector < Pii >
#define pb push_back
#define mp make_pair
#define reset(x) memset(x, 0, sizeof(x))
#define fori(i, a, b) for(int i = a; i ≤ b; i++)
#define ford(i, a, b) for(int i = a; i ≥ b; i--)

```

```

#define fora(xx, gg) for(auto xx : gg)
#define fi first
#define se second
#define Kurumi "DAPECH"
#define nmax 3005
#define mmax 100005
const ll OO = 1e9+7 ;

using namespace std;
int n,m,k,a[nmax][nmax],f[mmax];
int read(){
    int res = 0;
    char c = getchar();
    bool Neg = false;
    while(c == ' ' || c == '\n') c = getchar();
    if(c == '-'){
        Neg = true;
        c = getchar();
    }
    while('0' ≤ c && c ≤ '9'){
        res = res * 10 + c - '0';
        c = getchar();
    }
    if(Neg) return -res;
    return res;
}

void kurumi(){
    cin >> n >> m >> k;
    int ma = 0;
    for(int i = 1; i ≤ n; i++){
        for(int j = 1; j ≤ m; j++){
            cin >> a[i][j];
            f[a[i][j]]++;
            ma = max(ma,a[i][j]);
        }
    }
    int shido = 0;
    sort(f + 1, f + ma + 1);
    for(int i = 1; i ≤ k; i++){

```

```

        if(ma - i + 1 < 1) break;
        shido += f[ma - i + 1];
    }

    cout << shido << '\n';

}

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
    freopen(Kurumi".inp", "r", stdin);
    freopen(Kurumi".out", "w", stdout);
    kurumi();
    return 0;
}

```

Câu 4. (3,0 điểm) Hình vuông.

Thuật toán: Ta sẽ làm mảng tính tổng tích lũy đường chéo chính và đường chéo phụ. Sau đó ta sẽ dùng 2 vòng for để tìm hình vuông con kích thước $K \times K$ với tổng các số trên đường chéo chính và đường chéo phụ là lớn nhất.

```

#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define Pii pair < int, int >
#define Piii pair < int, Pii >
#define Vi vector < int >
#define Vii vector < Pii >
#define pb push_back
#define mp make_pair
#define reset(x) memset(x, 0, sizeof(x))
#define fori(i, a, b) for(int i = a; i ≤ b; i++)
#define ford(i, a, b) for(int i = a; i ≥ b; i--)
#define fora(xx, gg) for(auto xx : gg)
#define fi first
#define se second
#define Kurumi "HINHVIUONG"
#define nmax 3005
const ll OO = 1e9+7 ;

using namespace std;

```

```

int n,k;
int a[nmax][nmax];
Pii f[nmax][nmax];
int ma = -OO;
int read(){
    int res = 0;
    char c = getchar();
    bool Neg = false;
    while(c == ' ' || c == '\n') c = getchar();
    if(c == '-'){
        Neg = true;
        c = getchar();
    }
    while('0' ≤ c && c ≤ '9'){
        res = res * 10 + c - '0';
        c = getchar();
    }
    if(Neg) return -res;
    return res;
}

void kurumi(){
    cin >> n >> k;
    for(int i = 1; i ≤ n; i++){
        for(int j = 1; j ≤ n; j++){
            cin >> a[i][j];
            f[i][j].fi = a[i][j];
            f[i][j].se = a[i][j];
            f[i][j].fi += f[i - 1][j - 1].fi;
            f[i][j].se += f[i - 1][j + 1].se;
            ma = max(ma,a[i][j]);
        }
    }

    if(k == 1){
        cout << ma << '\n';
        return;
    }

    int shido = -OO;

```

```

for(int i = 1; i ≤ n - k + 1; i++){
    for(int j = 1; j ≤ n - k + 1; j++){
        int CC;
        CC = f[i + k - 1][j + k - 1].fi + f[i + k - 1][j].se - f[i - 1][j - 1].fi - f[i - 1][j + k].se;
        if(k % 2 != 0) CC -= a[i + k / 2][j + k / 2];
        shido = max(shido, CC);
    }
}
cout << shido << '\n';

}

int main()
{
    ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
    freopen(Kurumi".inp", "r", stdin);
    freopen(Kurumi".out", "w", stdout);
    kurumi();
    return 0;
}

```

---Hét---

LỜI NÓI ĐẦU.....	2
Chuyên đề 1.....	3
BÀI TOÁN VÀ THUẬT TOÁN	3
I. Bài tập và lời giải chuyên đề 1	3
1. <i>Nêu khái niệm bài toán và khái niệm Thuật toán?</i>	3
2. <i>Nêu các bước giải bài toán trên máy tính?</i>	3
3. <i>Nêu khái niệm chương trình dịch, thông dịch và biên dịch?</i>	3
4. <i>Trình bày Thuật toán giải các bài toán</i>	3
5. <i>Lời giải bài 4.</i>	4
II. Bài tập rèn luyện 1	5
Chuyên đề 2.....	6
CẤU TRÚC ĐIỀU KHIỂN RỄ NHÁNH	6
I. Bài tập và lời giải về cấu trúc rẽ nhánh	6
1. <i>Cấu trúc rẽ nhánh</i>	6
2. Viết chương trình giải các bài toán	6
a) Chẵn lẻ.....	6
b) Số chính phương.....	6
c) Tìm max2.....	6
d) Tìm max3.....	6
e) Tìm max4.....	6
f) Ghép số	6
g) Phương trình bậc nhất	6
h) Phương trình bậc hai	6
i) Hệ phương trình.....	6
j) Tìm X.....	6
3. Lời giải bài tập mục 2	6
a) Chẵn lẻ.....	6
b) Số chính phương.....	7
c) Tìm max2.....	7
d) Tìm max3.....	7
e) Tìm max4.....	8
f) Ghép số	8

g) Phương trình bậc nhất	8
h) Phương trình bậc hai	8
i) Hệ phương trình.....	8
j) Tìm X.....	8
4. Bài tập rèn luyện 2.....	9
a) Tổng các chữ số.....	9
b) Tích các chữ số.....	9
c) Tìm S	9
d) Đổi thời gian.....	9
e) Đổi tiền 1	9
f) Tìm P	9
g) Khoảng cách	9
h) Tam giác 1	9
i) Hình chữ nhật 1	9
j) Hình chữ nhật 2	9
k) Hình chữ nhật 3	9
l) Hình chữ nhật 4	10
m) Hình chữ nhật 5	10
n) Tìm X, Y.....	10
o) Tìm Q.....	10
p) Xóa số.....	10
q) Đổi tiền 2	10
r) Số trung bình cộng.....	10
s) Giải phương trình	10
t) Tam giác	11
u) Điểm và đường tròn 2.....	11
v) Vị trí tương đối của hai đường tròn.....	11
w) Giải phương trình	11
x) Số đẹp X	11
y) Hình chữ nhật	11
Chuyên đề 3.....	12
CẤU TRÚC ĐIỀU KHIỂN LẶP	12

I. Bài tập và lời giải về cấu trúc lặp For.....	12
1. Cấu trúc lặp For.....	12
2. Viết chương trình giải các bài toán.....	12
a) Liệt kê 1	12
b) Liệt kê 2.....	12
c) Chia hết.....	12
d) Tính S1	12
e) Tính S2	12
f) Tính S3	12
g) Tính S4	12
h) Tính S5	12
i) Tìm ước.....	12
j) Tính tổng ước	12
k) Số nguyên tố 1	12
3. Lời giải bài tập mục 2.....	12
a) Liệt kê 1	12
b) Liệt kê 2.....	13
c) Chia hết.....	13
d) Tính S1	13
e) Tính S2	13
f) Tính S3	14
g) Tính S4	14
h) Tính S5	14
i) Tìm ước.....	15
j) Tính tổng ước	15
k) Số nguyên tố 1	15
4. Bài tập rèn luyện 3.....	16
a) Liệt kê số nguyên tố	16
b) Nguyên tố 2	16
c) Tìm số 1	16
d) Tìm số 2.....	16
e) Số hoàn thiện	16

f) Phân tích số.....	16
g) Tính P	16
h) Số nguyên tố 2.....	17
i) Số Fibonacci	17
j) Số chính phương.....	17
II. Bài tập và lời giải về cấu trúc lặp While	17
1. Cấu trúc lặp While.....	17
2. Viết chương trình giải các bài toán.....	17
a) Tính tổng 1.....	17
b) Tính P	17
c) Tính tổng 2.....	17
d) Căn bậc 3	17
e) Tìm ước chung lớn nhất	17
f) Tìm bội chung nhỏ nhất.....	17
g) Phân tích thừa số nguyên tố	18
h) Tìm số nguyên tố.....	18
3. Lời giải bài tập mục 2.....	18
a) Tính tổng 1.....	18
b) Tính P	18
c) Tính tổng 2.....	19
d) Căn bậc 3	19
e) Tìm ước chung lớn nhất	20
f) Tìm bội chung nhỏ nhất.....	20
g) Phân tích thừa số nguyên tố	20
h) Tìm số nguyên tố.....	21
4. Bài tập rèn luyện 4.....	22
i) Số Mersenne	22
j) Ốc sên	22
k) Tìm số.....	22
l) Số đẹp 4	22
m) Cát số	22
n) Số Palindrom	22

o) Căn bậc hai	22
p) Gấp giấy.....	22
q) Vòng tay (ntucoder).....	23
r) Chữ số tận cùng	23
Chuyên đề 4.....	25
Kiểu dữ liệu mảng	25
I. Bài tập và lời giải về mảng một chiều	25
1. Kiểu dữ liệu mảng một chiều	25
2. Viết chương trình giải các bài toán	25
a) In mảng	25
b) Sắp xếp nổi bọt	25
c) Trung bình cộng.....	25
d) Bài toán đếm 1	25
e) Vị trí đẹp 1	25
f) Tìm min, max	25
g) Vị trí đạt giá trị nhỏ nhất	25
3. Lời giải các bài tập mục 2	25
a) In mảng	25
b) Sắp xếp nổi bọt	26
c) Trung bình cộng.....	26
d) Bài toán đếm 1.....	27
e) Vị trí đẹp 1	27
f) Tìm min, max	28
g) Vị trí đạt giá trị nhỏ nhất	28
4. Bài tập rèn luyện	29
a) Giá trị nhỏ nhì.....	29
b) Giá trị lớn thứ K	29
c) Tổng 3 phân tử.....	29
d) Tìm kiếm tuần tự	29
e) Dãy không giảm.....	29
f) Số Fibonacci	29
g) Bài toán đếm 2.....	29

h)	Dãy không giảm 2	29
i)	Mảng chẵn lẻ	30
j)	Tạo mảng C	30
k)	Tìm tổng nhỏ nhất	30
l)	Xóa phần tử thứ K	30
m)	Xóa phần tử chia hết cho K.....	30
n)	Chèn phần tử sau vị trí K.....	30
o)	Xóa phần tử giống nhau trong dãy số	30
p)	Bài toán đếm 2.....	30
q)	Đảo ngược dãy số	31
r)	Dãy liên tiếp không giảm dài nhất.....	31
s)	Dãy con liên tiếp có tổng lớn nhất.....	31
t)	Dãy bằng nhau liên tiếp dài nhất	31
u)	Tìm kiếm tuần tự 2	31
v)	Tìm kiếm nhị phân	31
w)	Nhân đôi mảng	31
x)	Vòng tròn.....	31
II.	Bài tập và lời giải về mảng hai chiều	31
1.	Kiểu dữ liệu mảng hai chiều	31
2.	Viết chương trình giải các bài toán.....	31
a)	In mảng	31
b)	Tổng số chẵn	31
c)	Đếm K.....	32
d)	Tìm vị trí chia hết cho K	32
e)	Tổng hàng, tổng cột.....	32
f)	Tìm max.....	32
g)	Tìm max hàng.....	32
h)	Tìm giá trị lớn nhì.....	32
i)	Ma trận đảo.....	32
3.	Lời giải bài tập mục 2.....	32
a)	In mảng	32
b)	Tổng số chẵn	32

c) Đếm K.....	33
d) Tìm vị trí chia hết cho K	34
e) Tổng hàng, tổng cột.....	34
f) Tìm max.....	35
g) Tìm max hàng.....	35
h) Tìm giá trị lớn nhì.....	36
i) Ma trận đảo.....	36
4. Bài tập rèn luyện.....	37
a) Tìm giá trị lớn thứ K.....	37
b) Sắp xếp hàng	37
c) Vị trí phân tử nguyên tố.....	37
d) Tìm chỉ số	37
e) Hình vuông cạnh K 1.....	37
f) Hình vuông cạnh K 2.....	37
g) Hình vuông có tổng bằng S.....	37
h) Hình vuông lớn nhất.....	37
i) Đếm bộ chỉ số.....	37
j) Hình chữ nhật có tổng bằng S.	37
k) Đếm hình chữ nhật	37
l) Hình chữ nhật toàn số 1	37
m) Xây kè.....	37
n) Điểm yên ngựa	38
o) Quân xe 1.....	38
p) Con kiến 1.....	38
q) Quân mã.....	38
r) Chia hết cho 11	38
s) Con kiến 2.....	38
Chuyên đề 5.....	39
Kiểu dữ liệu XÂU.....	39
I. Bài tập và lời giải về xâu	39
1. Kiểu dữ liệu xâu	39
2. Viết chương trình giải các bài toán.....	39

a) Độ dài xâu.....	39
b) Vị trí S1 trong S.....	39
c) Đếm S1 trong S.....	39
d) Nối xâu	40
e) Đếm kí tự số.....	40
f) Xâu đảo	41
g) Xâu đối xứng	41
h) Xâu chữ hoa.....	41
i) Xâu chữ thường	42
j) Chuẩn hóa xâu	42
k) Đếm từ	42
l) Kí tự đầu tiên các từ in hoa.....	43
m) Đếm phân phối	43
3. Bài tập rèn luyện.....	44
a) Tách số trong xâu	44
b) Số lớn nhất trong xâu	44
c) Sắp xếp kí tự	44
d) Đếm kí tự 1	44
e) Nén xâu.....	44
f) Giải nén xâu.....	44
g) Xóa S1 trong S	44
h) Xâu Fibonacci 1.....	44
i) Xâu Fibonacci 2.....	44
j) Xâu con palidrom	44
k) Sắp xếp số trong xâu	44
l) Mã hóa xâu	44
m) Giải mã xâu.....	45
n) Ghép xâu số	45
o) Xâu con chung dài nhất.....	45
Chuyên đề 6.....	46
BÀI TOÁN LIỆT KÊ.....	46
I. Bài tập và lời giải chuyên đề 5	46

1. Thuật toán Greedy (Tham)	46
4. Bài tập áp dụng	46
1. Liệt kê dãy nhị phân độ dài N	46
p) Liệt kê 2.....	46
q) Liệt kê 3.....	46
r) Liệt kê 4.....	46
s) Liệt kê 5	46
t) Dãy con có tổng bằng K.....	46
u) Dãy con có nhiều phần tử nhất có tổng $\leq K$	46
v) Chia nhóm 1	46
w) Dãy có tổng chia hết cho K.....	46
x) Chia nhóm có độ chênh lệch nhỏ nhất	46
y) Dãy con 1.....	46
z) Dãy con 2.....	46
aa) Dãy con 3.....	47
bb) Dãy con 4.....	47
cc) Dãy con 5.....	47
dd) Dãy con 6.....	47
ee) Dãy con 7.....	47
LỜI GIẢI BÀI TẬP ĐỀ XUẤT	48
a) Tổng các chữ số.....	48
b) Tích các chữ số.....	48
c) Đổi thời gian	48
d) Đổi tiền 1	49
e) Khoảng cách	50
f) Tam giác 1	51
g) Hình chữ nhật 1	51
h) Hình chữ nhật 2	52
i) Hình chữ nhật 3	52
j) Hình chữ nhật 4	52
k) Hình chữ nhật 5	52
l) Tìm X, Y	54

m)	Tìm Q.....	54
n)	Xóa số.....	55
o)	Đổi tiền 2	55
p)	Số trung bình cộng	56
q)	Giải phương trình	56
r)	Tam giác	56
s)	Điểm và đường tròn.....	56
t)	Vị trí tương đối của hai đường tròn.....	56
u)	Giải phương trình	56
Chuyên đề 7.....		57
QUY HOẠCH ĐỘNG		57
a)	Tính giai thừa của số tự nhiên N (kí hiệu: N!)	57
b)	Số Fibonacci.....	57
c)	Tính tổng của dãy số.....	57
d)	Hiệu số.....	58
e)	Lưới ô	58
f)	TRIANGLE PASCAL (Tam giác Pascal).....	59
g)	TRIANGLE NUMBER (Tam giác số)	59
1.	3 4 9 8 1	59
h)	h) Dãy con đơn điệu tăng dài nhất	60
i)	Di chuyển từ tây sang đông	60
HƯỚNG DẪN GIẢI CHUYÊN ĐỀ QUY HOẠCH ĐỘNG		61
a)	Tính giai thừa của số tự nhiên N (kí hiệu: N!) GT.PAS	61
b)	Số Fibonacci FIBO.PAS	61
c)	Tính tổng của dãy số SUM.PAS	62
d)	Hiệu số HIEUSO.PAS.....	63
e)	Lưới ô GRID.PAS	64
f)	TRIANGLE PASCAL (Tam giác Pascal) TRIANPAS.PAS.....	66
g)	TRIANGLE NUMBER (Tam giác số) TRIANNUM.PAS	67
h)	DÃY CON ĐƠN ĐIỆU TĂNG DÀI NHẤT SEQ.PAS	69
i)	DI CHUYỂN TỪ TÂY SANG ĐÔNG TAYDONG.PAS.....	71
Chuyên đề 8.....		74

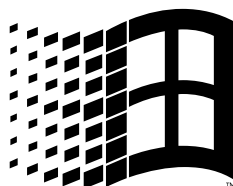
ĐỒ THỊ.....	74
I. Sơ lược các khái niệm cơ bản về đồ thị	74
II. Biểu diễn đồ thị trên máy tính:	75
1. Biểu diễn đồ thị theo ma trận kề:	75
2. Biểu diễn đồ thị theo danh sách cạnh:.....	77
3. Biểu diễn đồ thị theo danh sách kề:	77
III. Các Thuật toán tìm kiếm trên đồ thị.....	78
1. Thuật toán tìm kiếm theo chiều sâu (DFS).....	79
2. Thuật toán tìm kiếm theo chiều rộng (BFS).....	79
IV. Tính liên thông của đồ thị	82
II. Đồ thị liên thông:	82
III. Thành phần liên thông:	82
IV. Xác định các thành phần liên thông trên đồ thị vô hướng.....	83
BÀI TẬP:	85
Bài 1. Chú bò hư hỏng (BCDAISY)	85
Bài 2. Robin C11BC2	86
Bài 3. Ốc sên ăn rau (OCSE)	87
Bài 4. Đếm ao (BCLKCOUN).....	88
Bài 5. Bảo vệ nông trang (NKGUARD)	88
Bài 6. Trò chơi Lines (LINES)	89
Bài 7. Kết nối (CONNECT) – Trại hè HV 2015 – K11	90
Bài 8. Nước biển (BCISLAND).....	91
Bài 9. Tính toán lượng nước (PBCWATER).....	91
V. Bài tập tự giải.....	92
Hướng dẫn làm bài:	92
a) Bài 1. Chú bò hư hỏng (BCDAISY)	92
b) Bài 2. Robin C11BC2.....	92
c) Bài 3. Ốc sên ăn rau (OCSE).....	92
d) Bài 4. Đếm ao (BCLKCOUN)	93
e) Bài 5. Bảo vệ nông trang (NKGUARD)	93
f) Bài 6. Trò chơi Lines (LINES).....	93
g) Bài 7. Kết nối.....	93

h) Bài 8. Nước biển (BCISLAND).....	93
i) Bài 9. Tính toán lượng nước (PBCWATER)	93
I. Nhắc lại kiến thức	94
1. * Khái niệm “Đường đi và chu trình”	94
2. * Bài toán tìm đường đi.....	94
3. * Hướng giải quyết.....	94
II. Ví dụ “Tìm đường đi”	94
III. Ví dụ 2 “Kiểm tra tồn tại chu trình trong đồ thị”	95
ĐƯỜNG ĐI NGẮN NHẤT TRÊN ĐỒ THỊ CÓ TRỌNG SỐ.....	96
I. Thuật toán dijkstra	96
II. Thuật toán Floyd – Warshall	100
III. BÀI TẬP	103
Chuyên đề 9:.....	115
TUYỂN TẬP ĐỀ CODEFORCE.....	115
I. A problem	115
1. https://codeforces.com/problemset/problem/469/A	115
2. https://codeforces.com/problemset/problem/41/A	115
3. https://codeforces.com/contest/339/problem/A	116
4. https://codeforces.com/problemset/problem/705/A	116
5. https://codeforces.com/contest/1325/problem/A	117
II. B problem	117
6. https://codeforces.com/problemset/problem/519/B	117
7. https://codeforces.com/contest/1263/problem/B	118
8. https://codeforces.com/contest/1293/problem/B	119
9. https://codeforces.com/contest/1300/problem/B	120
10. https://codeforces.com/problemset/problem/1213/B	120
Chuyên đề 10.....	122
TUYỂN TẬP ĐỀ THI HSG.....	122
I. ĐỀ I.....	122
Câu 1. (7,0 điểm) Tích các chữ số lớn nhất.....	122
Câu 2. (6,0 điểm) Số chính phương.....	122
Câu 3. (4,0 điểm) Vị trí đẹp 2.....	122

Câu 4. (3,0 điểm) Ước nguyên tố	123
II. ĐỀ II	124
Câu 1. (7,0 điểm) Chữ số lẻ lớn nhất.....	124
Câu 2. (6,0 điểm) Ước chung lớn nhất	124
Câu 3. (4,0 điểm) Giải Nhì	124
Câu 4. (3,0 điểm) Tìm K.....	125
III. ĐỀ III	126
Câu 1. (7,0 điểm) Ba anh em siêu nhân.....	126
Câu 2. (6,0 điểm) Phép tính với phân số	126
Câu 3. (4,0 điểm) Bài toán đếm.....	127
Câu 4. (3,0 điểm) Nhầm lẫn.....	127
IV. ĐỀ IV	128
V. ĐỀ V	131
Câu 1. (7,0 điểm) Mật mã.....	131
Câu 2. (6,0 điểm) Vinyl và những chiếc tiêu.....	131
Câu 3. (4,0 điểm) Phần thưởng	132
Câu 4. (3,0 điểm) Hoàng tử Hikaru Genji	132
VI. ĐỀ VI.....	134
Câu 1. (7,0 điểm) Quân cờ may mắn.....	134
Câu 2. (6,0 điểm) Hikaru tìm quân cờ	134
Câu 3. (4,0 điểm) Hikaru chơi cờ vây	135
Câu 4. (3,0 điểm) Hikaru	135
VII. ĐỀ VII	136
Câu 1. (7,0 điểm) Tiqui-Taca.....	136
Câu 2. (6,0 điểm) Số áo tập	136
Câu 3. (4,0 điểm) Số áo may mắn	137
Câu 4. (3,0 điểm) Mật mã phòng họp.....	138
VIII. ĐỀ VIII.....	139
Câu 1. (7,0 điểm) Bộ số tam giác	139
Câu 2. (6,0 điểm) Lát cắt cực đại.....	139
Câu 3. (4,0 điểm) Trò chơi đập ếch	140
Câu 4. (3,0 điểm) Hình vuông	140

HƯỚNG DẪN GIẢI BỘ ĐỀ	142
I. ĐỀ I.....	142
II. ĐỀ II	144
III. ĐỀ III.....	149
IV. ĐỀ V	153
V. ĐỀ VI.....	157
VI. ĐỀ VII.....	163
VII. ĐỀ VIII.....	168

"Mỗi ngày tôi chọn một niềm vui,
Chọn những bông hoa, chọn những nụ cười..."



Các bạn tìm đọc tài liệu của cùng tác giả:



Hỗ trợ qua zalo: 0854518333.