

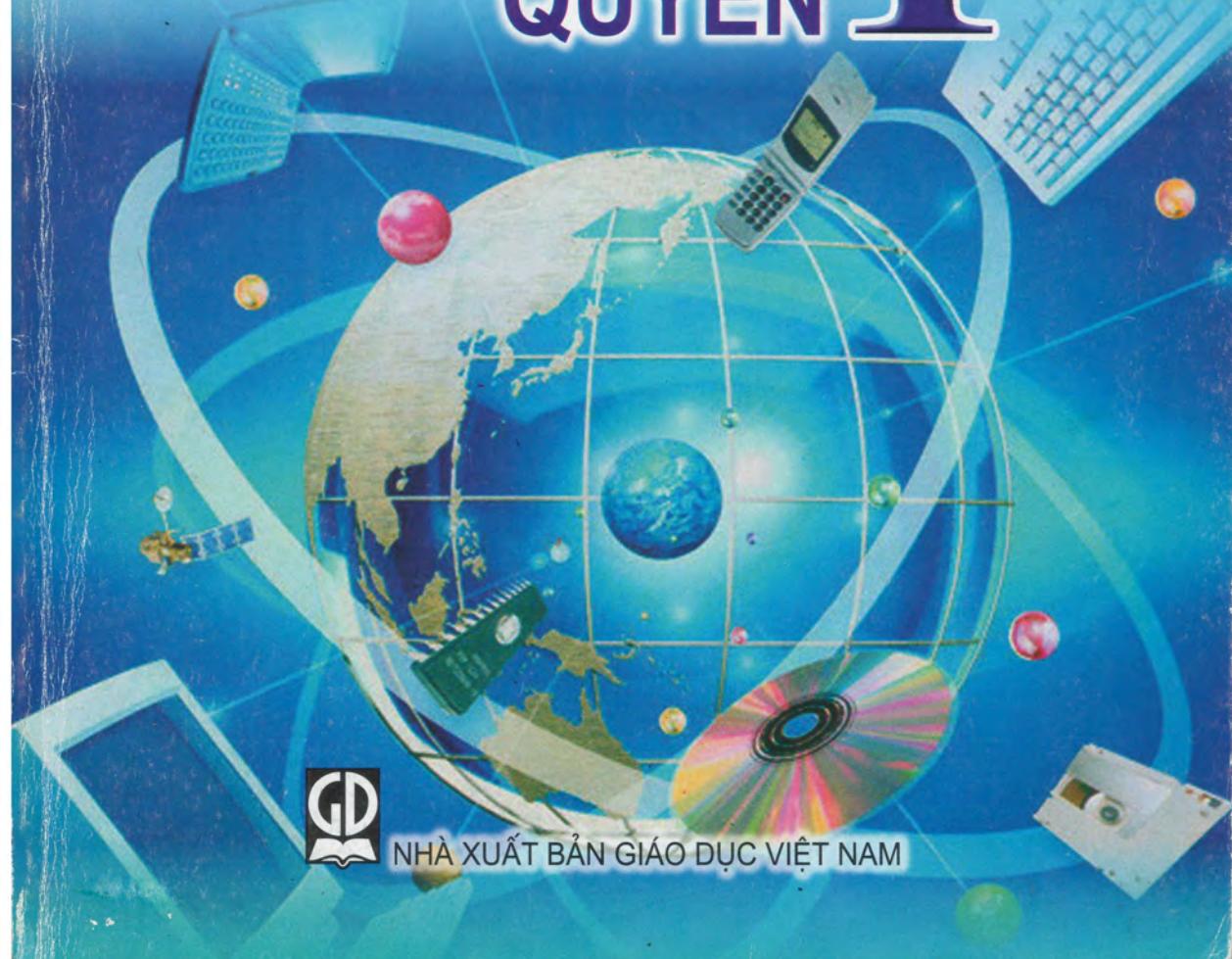
HỒ SĨ ĐÀM (Chủ biên)

ĐỖ ĐỨC ĐÔNG - LÊ MINH HOÀNG - NGUYỄN THANH HÙNG

# TÀI LIỆU CHUYÊN TIN HỌC

# BÀI TẬP

## QUYỂN 1



HỒ SĨ ĐÀM (Chủ biên)  
ĐỖ ĐỨC ĐÔNG - LÊ MINH HOÀNG - NGUYỄN THANH HÙNG

**TÀI LIỆU CHUYÊN TIN HỌC  
BÀI TẬP QUYỂN 1**

(Tái bản lần thứ nhất)

## LỜI NÓI ĐẦU

Bộ sách *Tài liệu chuyên Tin học - Bài tập Quyển 1, 2, 3* được viết kèm với bộ *Tài liệu chuyên Tin học - Quyển 1, 2, 3* tương ứng đã được xuất bản. Các tác giả tham gia biên soạn bộ sách là những thầy giáo đã và đang dạy ở các trường chuyên, lớp chọn hoặc tham gia các khóa bồi dưỡng thi tin học quốc tế, bồi dưỡng giáo viên tin cho các trường chuyên theo chương trình của Bộ Giáo dục và Đào tạo, mong muốn xây dựng được các tài liệu có tính hệ thống phục vụ tốt các đối tượng thuộc lĩnh vực chuyên tin học.

Các cuốn *Tài liệu chuyên Tin học - Bài tập* đều có cấu trúc như nhau, gồm hai phần:

**Phần I - Bài tập** bao gồm tất cả các bài tập trong những chuyên đề của sách *Tài liệu chuyên Tin học* tương ứng và các bài tập bổ sung, được sắp xếp từ dễ đến khó, từ đơn giản đến phức tạp.

**Phần II - Hướng dẫn giải bài tập** có thể là những hướng dẫn chi tiết để giúp bạn đọc tìm được lời giải hoặc chỉ là đoạn chương trình chính giúp bạn đọc hiểu và tìm được lời giải hoặc chương trình hoàn chỉnh để tham khảo. Đối với một số bài tập thì có thể chỉ là đáp án hay hướng dẫn ngắn gọn.

Hai bộ sách *Tài liệu chuyên Tin học* và *Tài liệu chuyên Tin học - Bài tập* tạo thành hệ thống tài liệu khá hoàn chỉnh theo định hướng Chương trình các chuyên đề chuyên tin học đã được Bộ Giáo dục và Đào tạo ban hành. Do vậy cùng với bộ sách *Tài liệu chuyên Tin học*, bộ sách *Tài liệu chuyên Tin học - Bài tập* sẽ là tài liệu thiết thực phục vụ cho giáo viên, học sinh các trường chuyên, lớp chọn cả Trung học phổ thông và Trung học cơ sở. Ngoài ra, bộ sách còn là tài liệu tham khảo bổ ích cho việc tập huấn sinh viên các trường Đại học, Cao đẳng tham gia các kì thi Olympic Tin học Sinh viên Toàn quốc và Kì thi lập trình viên Quốc tế.

**Lưu ý khi sử dụng bộ sách:** Các bài tập trong bộ sách này được đánh số như trong sách lí thuyết; các bài tập bổ sung được để ở mục riêng và đánh số tiếp theo.

Mặc dù các tác giả và Ban biên tập đã cố gắng hoàn thiện nhưng chắc chắn bộ sách còn nhiều thiếu sót, các tác giả mong nhận được nhiều ý kiến đóng góp để sách sẽ hoàn thiện hơn, phục vụ bạn đọc được hiệu quả hơn. Các góp ý xin gửi về:

Ban Toán-Tin, Công ty cổ phần Dịch vụ xuất bản Giáo dục Hà Nội-  
Nhà xuất bản Giáo dục Việt Nam, 187B, Giảng Võ, Hà Nội.

Các tác giả

P

1.

1.

1.

1.

# Phần I

## Bài tập

### CHUYÊN ĐỀ 1. THUẬT TOÁN VÀ PHÂN TÍCH THUẬT TOÁN

Phân tích thời gian thực hiện của đoạn chương trình ở các bài từ 1.1 đến 1.9.

1.1.

```
for i:=1 to n do
  if i mod 2=0 then c:=c+1;
```

1.2.

```
for i:=1 to n do
  if i mod 2=0 then c1:=c1+1
  else c2:=c2+1;
```

1.3.

```
for i:=1 to n do
  if i mod 2=0 then
    for j:=1 to n do c:=c+1
```

1.4.

```
a:=0;
b:=0;
c:=0;
for i:=1 to n do
begin
  a:=a + 1;
  b:=b + i;
  c:=c + i*i;
end;
```

1.5.

```
i:=n;  
d:=0;  
while i>0 do  
begin  
i:=i-1;  
d:=d + i;  
end;
```

1.6.

```
i:=0;  
d:=0;  
repeat  
i:=i+1;  
if i mod 3=0 then d:=d + i;  
until i>n;
```

1.7.

```
d:=0;  
for i:=1 to n-1 do  
for j:=i+1 to n do d:=d+1;
```

1.8.

```
d:=0;  
for i:=1 to n-2 do  
for j:=i+1 to n-1 do  
for k:=j+1 to n do d:=d+1;
```

1.9.

```
d:=0;  
while n>0 do  
begin  
n:=n div 2;  
d:=d+1;  
end;
```

1.10. Cho một dãy số gồm  $n$  số nguyên dương, xác định xem có tồn tại một dãy con liên tiếp có tổng bằng  $k$  hay không?

- a) Dưa ra thuật toán có thời gian thực hiện  $O(n^3)$ .
- b) Dưa ra thuật toán có thời gian thực hiện  $O(n^2)$ .
- c) Dưa ra thuật toán có thời gian thực hiện  $O(n)$ .

## CHUYÊN ĐỀ 2. CÁC KIẾN THỨC CƠ BẢN

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

Ví dụ:  $10101100_2 = AC_{16}$

$$101010111100000100100011_2 = ABC123_{16}$$

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

a) Phân tích  $N$  thành thừa số nguyên tố.

b) Đếm số ước của  $N$ .

c) Tính tổng các ước của  $N$ .

- 2.3. Đưa ra những số nhỏ hơn hoặc bằng  $10^6$  mà cách kiểm tra tính nguyên tố của Fermat bị sai.

- 2.4. Sử dụng sàng số nguyên tố liệt kê các số nguyên tố trong đoạn  $[L, R]$ .

- 2.5. Người ta định nghĩa một số nguyên dương  $N$  được gọi là số đẹp nếu  $N$  thỏa mãn *một trong hai* điều kiện sau:

-  $N$  bằng 9;

- Gọi  $f(N)$  là tổng các chữ số của  $N$  thì  $f(N)$  cũng là số đẹp.

Cho số nguyên dương  $N$  ( $N \leq 10^{100}$ ), hãy kiểm tra xem  $N$  có phải là số đẹp không?

- 2.6. Dùng cách biểu diễn số nguyên lớn bằng xâu kí tự và thêm thông tin dấu ( $sign = 1$  nếu số lớn là số không âm,  $sign = -1$  nếu số lớn là số âm) để xử lý số nguyên lớn có dấu như sau:

```
type bigNum = record
    sign : longint;
    num : string;
end;
```

Hãy xây dựng các hàm xử lí số nguyên lớn có dấu.

- 2.7. Dùng cách biểu diễn số nguyên lớn bằng mảng (mỗi phần tử của mảng là một nhóm các chữ số).

a) Hãy xây dựng các hàm xử lí số nguyên lớn.

- b) Sử dụng hàm nhân số nguyên lớn với số nhỏ để tính  $N!$  với  $N \leq 2000$ .
- 2.8.** Tìm  $K$  chữ số cuối cùng của  $M^N$  ( $0 < K \leq 9, 0 \leq M, N \leq 10^6$ ).
- Ví dụ,  $K = 2, M = 2, N = 10$ , ta có  $2^{10} = 1024$ , như vậy hai chữ số cuối cùng của  $2^{10}$  là 24.
- 2.9.** Cho  $N$  ( $N \leq 10$ ) nguyên dương  $a_1, a_2, \dots, a_N$  ( $a_i \leq 10^9$ ). Tìm ước số chung lớn nhất, bội số chung nhỏ nhất của  $N$  số trên (chú ý, BSCNN có thể rất lớn).
- 2.10.** Cho hai số nguyên không âm  $A, B$  ( $0 \leq A \leq B \leq 10^{200}$ ), tính số lượng số Fibonacci trong đoạn  $[A, B]$ .
- 2.11.** Cho số nguyên dương  $N$  ( $N \leq 10^{100}$ ), hãy tách  $N$  thành tổng các số Fibonacci đôi một khác nhau.
- Ví dụ,  $N = 16 = 1 + 5 + 13$ .
- 2.12.** Cho  $N$  là một số nguyên dương không vượt quá  $10^9$ . Hãy tìm số chữ số 0 tận cùng của  $N!$
- 2.13.** Cho  $s$  là một xâu mô tả số nguyên không âm ở hệ cơ số  $a$ , hãy chuyển số đó sang hệ cơ số  $b$  ( $1 < a, b \leq 16$ , độ dài xâu  $s$  không vượt quá 50).
- 2.14.** Xây dựng hàm kiểm tra số nguyên dương  $N$  có phải là số chính phương không? ( $N < 10^{100}$ ).
- 2.15.** Tính  $C_n^k$  ( $0 < k \leq n \leq 2000$ ).
- 2.16.** Tính số Catalan<sub>n</sub> ( $n \leq 2000$ ).
- 2.17.** Hãy đếm số cách đặt  $k$  quân xe lên bàn cờ  $n \times n$  sao cho không có quân nào ăn được nhau ( $1 \leq k \leq n \leq 100$ ).
- 2.18.** Giả thiết  $N$  là số nguyên dương. Số nguyên  $M$  là tổng của  $N$  với các chữ số của nó.  $N$  được gọi là nguồn của  $M$ .
- Ví dụ,  $N = 245$ , khi đó  $M = 245 + 2 + 4 + 5 = 256$ . Như vậy, nguồn của 256 là 245. Có những số không có nguồn và có số lại có nhiều nguồn. Ví dụ, số 216 có 2 nguồn là 198 và 207.
- Cho số nguyên  $M$  ( $M$  có không quá 100 chữ số) hãy tìm nguồn nhỏ nhất của nó. Nếu  $M$  không có nguồn thì đưa ra số 0.
- 2.19.** Tính số các ước và tổng các ước của  $N!$  ( $N \leq 100$ ).
- 2.20.** Cho một chiếc cân hai đĩa và các quả cân có khối lượng  $3^0, 3^1, 3^2, \dots$ . Hãy chọn các quả cân để có thể cân được vật có khối lượng  $N$  ( $N \leq 10^{100}$ ).

Ví dụ, cần cân vật có khối lượng  $N = 11$  ta cần sử dụng các quả cân sau:

- Đĩa cân bên trái: quả cân  $3^1$  và  $3^2$ .
- Đĩa cân bên phải: quả cân  $3^0$  và vật  $N = 11$ .

**2.21.** Đếm số lượng dãy nhị phân khác nhau độ dài  $n$  mà không có hai số 1 nào đứng cạnh nhau.

Ví dụ,  $n = 3$ , ta có 5 dãy 000, 001, 010, 100, 101.

**2.22.** Cho xâu  $s$  chỉ gồm kí tự từ ' $a'$  đến ' $z'$  (độ dài xâu  $s$  không vượt quá 100), hãy đếm số hoán vị khác nhau của xâu đó.

Ví dụ,  $s='aba'$ , ta có 3 hoán vị ' $bab$ ', ' $aba$ ', ' $baa$ '.

**2.23.** Nam quyết định đánh số trang cho quyển sách của anh ta từ 1 đến  $N$ . Hãy tính toán số lượng chữ số 0 cần dùng, số lượng chữ số 1 cần dùng,..., số lượng chữ số 9 cần dùng.

**Input:** Tệp *digits.inp* gồm một dòng duy nhất chứa một số  $N$  ( $N \leq 10^{100}$ ).

**Output:** Tệp *digits.out* có dạng gồm 10 dòng,

- Dòng thứ nhất là số lượng chữ số 0 cần dùng,
- Dòng thứ hai là số lượng chữ số 1 cần dùng,...,
- Dòng thứ 10 là số lượng chữ số 9 cần dùng.

**2.24. TAM GIÁC SỐ** (*Đề thi học sinh giỏi Hà Tây, năm 2006*)

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

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

Ví dụ, đường đi 7 8 1 4 6 có tích là  $S = 1344$ ,

đường đi 7 3 1 7 5 có tích là  $S = 735$ .

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

**Input:** Tệp văn bản TGS.INP:

- Dòng đầu tiên chứa số nguyên  $n$ , ( $0 < n < 101$ );

- $N$  dòng tiếp theo, từ dòng thứ 2 đến dòng thứ  $N + 1$ : dòng thứ  $i$  có  $(i - 1)$  số cách nhau bởi dấu cách (các số có giá trị tuyệt đối không vượt quá 100).

**Output:** Đưa ra tệp văn bản TGS.OUT có một số nguyên là tích lớn nhất tìm được.

**Ví dụ:**

TGS.INP	TGS.OUT
5	5880
7	
3 8	
8 1 0	
2 7 4 4	
4 5 -2 6 5	

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

Một cháu gái hằng ngày được mẹ giao nhiệm vụ đến thăm bà nội. Từ nhà mình đến nhà bà nội, cô bé phải đi qua một khu rừng có rất nhiều loại nấm. Trong số các loại nấm, có ba loại có thể ăn được. Cô bé đánh số ba loại nấm ăn được lần lượt là 1, 2 và 3. Là một người cháu hiếu thảo nên cô bé quyết định mỗi lần đến thăm bà, cô sẽ hái ít nhất hai loại nấm ăn được để nấu súp cho bà. Khu rừng mà cô bé đi qua được chia thành lưới ô vuông gồm  $m$  hàng và  $n$  cột. Các hàng của lưới được đánh số từ trên xuống dưới bắt đầu từ 1, còn các cột được đánh số từ trái sang phải, bắt đầu từ 1. Ô nấm giao của hàng  $i$  và cột  $j$  có toạ độ  $(i, j)$ . Trên mỗi ô vuông, trừ ô  $(1, 1)$  và ô  $(m, n)$  các ô còn lại hoặc có nấm độc và cô bé không dám đi vào (đánh dấu là  $-1$ ), hoặc là có đúng một loại nấm có thể ăn được (đánh dấu bằng số hiệu của loại nấm đó). Khi cô bé đi vào một ô vuông có nấm ăn được thì cô bé sẽ hái loại nấm mọc trên ô đó. Xuất phát từ ô  $(1, 1)$ , để đến được nhà bà nội ở ô  $(m, n)$  một cách nhanh nhất cô bé luôn đi theo hướng sang phải hoặc xuống dưới.

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

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

**Input:** Tệp văn bản MUSHROOM.INP:

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

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

**Output:** Tập văn bản MUSHROOM.OUT chứa một dòng ghi một số nguyên là kết quả bài toán.

**Ví dụ:**

MUSHROOM.INP	MUSHROOM.OUT
3 4 0 3 -1 2 3 3 3 3 3 1 3 0	3

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

Để trang trí cho lễ kỉ niệm 15 năm hội thi Tin học trẻ toàn quốc, ban tổ chức đã dùng một hệ thống đèn màu gồm  $n$  đèn đánh số từ 1 đến  $n$ . Mỗi đèn có khả năng sáng màu xanh hoặc màu đỏ. Các đèn được điều khiển theo quy tắc sau:

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

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

**Ví dụ:** với  $n = 10$ ;  $k = 3$ .

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

Sau chương trình 2	Xanh: 1	5	6	7				
	Đỏ : 2 3 4			8	9	10		
Sau chương trình 3	Xanh: 1	4	5	6	7	8		
	Đỏ : 2 3				9	10		

Vậy có 4 đèn đỏ sau chương trình thứ 3.

## Bài tập bổ sung

### 2.27. Số đẹp

Một số nguyên dương được gọi là **đẹp** nếu tổng bình phương các chữ số của nó (trong dạng biểu diễn thập phân) là một số nguyên tố.

Ví dụ, 12 là một số đẹp vì  $1^2 + 2^2 = 5$  là số nguyên tố.

Các số đẹp được đánh số theo thứ tự tăng dần của giá trị, bắt đầu từ 1.

**Yêu cầu:** Cho số nguyên  $n$  ( $1 \leq n \leq 1000000$ ). Hãy tìm số đẹp thứ  $n$ .

**Input:** Tệp văn bản BEAUTY.INP gồm nhiều dòng, mỗi dòng là một bộ kiểm thử chứa một số nguyên  $n$ .

**Output:** Tệp văn bản BEAUTY.OUT ghi kết quả của mỗi bộ kiểm thử, mỗi bộ được ghi trên một dòng.

**Ví dụ:**

BEAUTY.INP	BEAUTY.OUT
1	11
6	23

### 2.28. Đầu giá

Thành phố Hà Nội có một số linh vật được đánh số thứ tự từ  $A$  đến  $B$  ( $A, B$  là hai số nguyên dương  $A \leq B$ ). Lãnh đạo thành phố quyết định bán đấu giá những linh vật có số thứ tự đẹp để lấy tiền ủng hộ đồng bào lũ lụt miền Trung. Một số thứ tự được gọi là **đẹp** nếu nó thoả mãn các điều kiện sau:

- Là một số nguyên dương  $T$  mà  $A \leq T \leq B$ ;
- $T$  là một số nguyên tố;
- $T$  là một số đối xứng (đọc  $T$  từ trái qua phải giống như đọc  $T$  từ phải qua trái). Ví dụ 12321 là một số đối xứng.

**Yêu cầu:** Cho hai số nguyên dương  $A$  và  $B$  ( $A \leq B$ ), hãy tìm số linh vật có số thứ tự đẹp.

**Input:** Tệp văn bản AUCTION.INP gồm một dòng chứa hai số nguyên dương  $A$  và  $B$  ( $0 \leq A \leq B \leq 10^9$ ).

**Output:** Đưa ra tệp văn bản AUCTION.OUT gồm một số nguyên là số linh vật có số thứ tự đẹp.

**Ví dụ:**

AUCTION.INP
11111 22222

AUCTION.OUT
23

## 2.29. PAIRS

Cho số nguyên dương  $n$ , đếm số cách chia các số từ 1 đến  $2n$  thành  $n$  nhóm, mỗi nhóm gồm hai số mà giá trị tuyệt đối của hiệu hai số trong một nhóm bằng giá trị tuyệt đối của hiệu hai số trong nhóm khác.

**Input:** Tệp văn bản PAIRS.INP gồm nhiều dòng, mỗi dòng gồm một số  $n$  ( $n \leq 10^6$ ), có không quá  $10^6$  dòng.

**Output:** Tệp văn bản PAIRS.OUT gồm nhiều dòng, mỗi dòng là kết quả tương ứng với dữ liệu vào.

**Ví dụ:**

PAIRS.INP	PAIRS.OUT	Giải thích
1	1	
2	2	Có hai cách chia các số từ 1 đến 4 Cách 1: {1, 2}, {3, 4} Cách 2: {1, 3}, {2, 4}

## 2.30. FDP

Cho  $n$  và  $k$  ( $n \leq 10^8$ ;  $k \leq 10^{12}$ ). Tìm  $m$  lớn nhất sao cho  $n!$  chia hết cho  $k^m$ .

**Input:** Tệp văn bản FDP.INP:

- Dòng đầu ghi số  $T$  là số bộ kiểm thử ( $T \leq 100$ );
- $T$  dòng sau, mỗi dòng có hai số  $n$  và  $k$ .

**Output:** Tệp văn bản FDP.OUT gồm  $T$  dòng, mỗi dòng là kết quả tìm được tương ứng với dữ liệu vào.

**Ví dụ:**

FDP.INP	FDP.OUT
2	3
5 2	2
10 10	

### 2.31. Tổng hiệu

Tìm hai số nguyên  $A$  và  $B$  ( $|A|, |B| \leq 10^{100}$ ), biết tổng  $A + B$  và hiệu  $A - B$ .

**Yêu cầu:** Cho tổng và hiệu của hai số, tìm hai số đó.

**Input:** Tệp văn bản AB.INP

- Dòng 1: chứa tổng  $A + B$ ;
- Dòng 2: chứa hiệu  $A - B$ .

**Output:** Tệp văn bản AB.OUT

- Dòng 1: ghi  $A$ ;
- Dòng 2: ghi  $B$  ( $A \leq B$ ).

**Ví dụ:**

AB.INP	AB.OUT
0	-100
-200	100

### 2.32. Bội số chung nhỏ nhất

Cho số nguyên  $n$ , tìm bội số chung nhỏ nhất của các số  $1, 2, \dots, n$ .

**Input:** Tệp văn bản BSCNN.INP gồm một dòng chứa số  $n$  ( $n \leq 1000$ ).

**Output:** Tệp văn bản BSCNN.OUT ghi bội số chung nhỏ nhất của  $1, 2, \dots, n$ .

**Ví dụ:**

BSCNN.INP	BSCNN.OUT
3	6

### 2.33. Thẻ thông minh

Tập đoàn Smart IT quyết định ứng dụng thẻ thông minh trong việc quản lý an ninh ở nơi làm việc. Mỗi nhân viên của Smart IT được cấp một thẻ thông minh riêng, trong thẻ chứa một dãy số bí mật gồm  $m$  số nguyên dương  $\{k_1, k_2, \dots, k_m\}$ .

Trong nhà điều hành của Smart IT có  $n$  căn phòng được đánh số từ 1 đến  $n$ . Ở cửa vào của căn phòng thứ  $i$  ( $1 \leq i \leq n$ ) có một đầu đọc thẻ. Khi cần mở cửa phòng, người nhân viên sẽ đưa thẻ vào đầu đọc thẻ. Nếu thẻ phù hợp với phòng thì cửa sẽ mở.

Trong đầu đọc thẻ ở phòng thứ  $i$  có lưu một dãy số nguyên dương  $\{x_{i1}, x_{i2}, \dots, x_{im}\}$ . Thẻ phù hợp với phòng thứ  $i$  nếu tích  $k_1 \times k_2 \times \dots \times k_m$  là bội số của tích  $x_{i1} \times x_{i2} \times \dots \times x_{im}$ .

**Yêu cầu:** Cho dãy số bí mật của một thẻ thông minh và dãy số trong đầu đọc thẻ của  $n$  căn phòng. Hãy cho biết thẻ thông minh này có thể dùng để mở được bao nhiêu phòng.

**Input:** Tệp văn bản SCARD.INP:

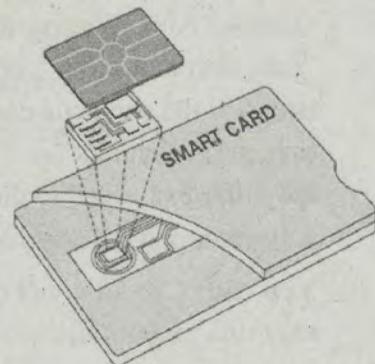
- Dòng đầu tiên chứa hai số nguyên dương  $m$  và  $n$  ( $1 \leq m, n \leq 100$ ).
- Dòng thứ hai chứa  $m$  số nguyên dương  $k_1, k_2, \dots, k_m$  là dãy số bí mật trên thẻ. Mỗi số có giá trị không quá  $10^{15}$ .
- Dòng thứ  $i$  trong số  $n$  dòng tiếp theo ( $1 \leq i \leq n$ ), mỗi dòng gồm  $m$  số nguyên dương  $x_{i1}, x_{i2}, \dots, x_{im}$  là dãy số trong đầu đọc thẻ tại phòng thứ  $i$ . Mỗi số có giá trị không quá  $10^{15}$ . Các số trên cùng một dòng được cách nhau bởi một dấu cách.

**Output:** Tệp văn bản SCARD.OUT:

- Dòng đầu tiên chứa số nguyên  $C$  là số lượng những phòng có thể mở cửa được.
- Dòng thứ hai chứa  $C$  số nguyên là số thứ tự (theo thứ tự tăng dần) của các phòng mà bạn có thể mở cửa được. Các số trên cùng một dòng được cách nhau bởi một dấu cách.

**Ví dụ:**

SCARD.INP
3 4
7 10 2011
1 3 5
2 2 7
7 2 5
14 1 2011



SCARD.OUT
2
3 4

### 2.34. Máy tính sinh học (Đề thi Olympic sinh viên, năm 2011)

Giáo sư Alex Waston đang nghiên cứu một loại máy tính mới có tên gọi “máy tính axít amin”, dựa trên nguyên lý hoạt động của các chuỗi axít amin trong cơ thể sống của các sinh vật. Hiện tại Alex Waston đã hiểu được thuộc tính của  $K$  loại axít amin khác nhau, cho nên tất cả con số và phép toán trong máy tính axít amin sẽ được biểu diễn và thực hiện trong hệ cơ số  $K$ . Xét số nguyên dương  $n$  trong máy tính axít amin, ta định nghĩa:  $n! = 1 \times 2 \times \dots \times n$ .

**Yêu cầu:** Cho số  $n$  viết ở hệ cơ số  $K$ , tìm số chữ số 0 cuối cùng của  $n!$  trong máy tính axít amin.

**Input:** Tệp văn bản BIOCOMP.INP:

- Dòng đầu chứa một số nguyên dương  $K$  ( $K \leq 10$ ).
- Dòng thứ hai chứa số  $n$  viết ở hệ cơ số  $K$  có không quá 1000 chữ số.

**Output:** Tệp văn bản BIOCOMP.OUT gồm một số nguyên (hệ cơ số  $K$ ) là số chữ số 0 cuối cùng của  $n!$  trong biểu diễn hệ cơ số  $K$ .

**Ví dụ:**

BIOCOMP.INP	BIOCOMP.OUT
8	
7	1

### 2.35. Xâu nhị phân gần đối xứng

Một xâu được gọi là đối xứng nếu đọc xâu đó từ trái qua phải giống như đọc từ phải qua trái. Một xâu nhị phân được gọi là gần đối xứng nếu sau khi sắp xếp lại các kí tự của nó ta thu được một xâu đối xứng.

**Ví dụ:** Các xâu nhị phân '110', '1010', '10000' là các xâu nhị phân gần đối xứng vì sau khi sắp xếp lại các kí tự của chúng, ta thu được các xâu tương ứng '101', '1001', '00100' là các xâu đối xứng.

**Yêu cầu:** Cho hai số nguyên  $n$ ,  $t$  và xâu nhị phân gần đối xứng  $s$  có độ dài  $n$ . Tìm xâu nhị phân gần đối xứng có độ dài  $n$  có thứ tự từ điển thứ  $t$  và tìm thứ tự từ điển của xâu  $s$ .

**Input:** Tệp văn bản NBS.INP:

- Dòng 1: chứa số nguyên dương  $n$  ( $n \leq 10^6$ );
- Dòng 2: chứa số nguyên dương  $t$  ( $t \leq 10^{100}$ );
- Dòng 3: chứa xâu  $s$ .

**Output:** Tệp văn bản NBS.OUT:

- Dòng 1: xâu nhị phân gần đối xứng độ dài  $n$  thứ  $t$ ;
- Dòng 2: số thứ tự của xâu nhị phân gần đối xứng  $s \bmod 111539786$ .

**Ví dụ:**

NBS.INP	NBS.OUT
2	11
2	1
00	

### 2.36. Hàng cây (Đề thi học sinh giỏi Quốc gia, năm 2011):

Một trang trại lớn có  $n$  cây cảnh với độ cao khác nhau tùng đôi. Các cây này được xếp theo một hàng dọc. Ông chủ trang trại là người có đầu óc thẩm mĩ nên bố trí hàng cây có tính chất không đơn điệu như sau: “Đi từ đầu hàng đến cuối hàng không có 3 cây nào (không nhất thiết phải liên tiếp) có chiều cao giảm dần”.

Một hôm, ông chủ mua thêm một cây cảnh mới có chiều cao lớn hơn chiều cao của tất cả các cây đã có. Ông ta muốn xếp cây cảnh mới vào một trong  $n+1$  vị trí có thể của hàng cây (vào vị trí đầu hàng, vị trí sau cây thứ nhất của hàng, vị trí sau cây thứ hai của hàng, ..., vị trí sau cây thứ  $n$  của hàng) sao cho được hàng cây vẫn thoả mãn yêu cầu về tính không đơn điệu nêu trên.

**Yêu cầu:**

- Hãy cho biết có bao nhiêu cách xếp cây cảnh cao nhất mua vào hàng cây sao cho vẫn đảm bảo điều kiện về tính không đơn điệu.
- Giả sử mỗi ngày ông chủ muốn xếp  $n+1$  cây đã có thành hàng cây đảm bảo yêu cầu về tính không đơn điệu và hai hàng cây của hai ngày khác nhau là không trùng nhau. Hãy giúp ông chủ tính xem việc đó có thể diễn ra nhiều nhất là bao nhiêu ngày.

**Input:** Tệp văn bản TREELINE.INP:

- Dòng thứ nhất chứa hai số nguyên dương  $n$  và  $h$  tương ứng là số lượng cây và chiều cao của cây cao nhất. Biết rằng  $n \leq 10^5$ ,  $h \leq 10^6$ .
- Dòng thứ hai chứa  $n$  số nguyên dương (mỗi số đều nhỏ hơn  $h$ ) tương ứng là dãy chiều cao của  $n$  cây được xếp ban đầu.

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

**Output:** Tệp văn bản TREELINE.OUT:

- Dòng thứ nhất ghi một số nguyên là số cách xếp cây cao nhất vào hàng.
- Dòng thứ hai ghi một số nguyên là phần dư trong phép chia số ngày lớn nhất tìm được cho  $10^9$ .

**Ví dụ:**

TREELINE.INP	TREELINE.OUT
2 2011	2
11 1	5

### CHUYÊN ĐỀ 3. SẮP XẾP

**3.1.** Cho một danh sách  $n$  học sinh ( $1 \leq n \leq 200$ ), mỗi học sinh có thông tin sau:

- Họ và tên: xâu kí tự độ dài không quá 30 (các từ cách nhau một dấu cách);
  - Điểm: số thực.
- a) Đưa ra danh sách họ và tên đã sắp xếp theo thứ tự abc (thứ tự ưu tiên tên, họ, đệm).
  - b) Có bao nhiêu tên khác nhau trong danh sách, liệt kê các tên đó.
  - c) Chọn những học sinh có thứ hạng 1, 2, 3 điểm cao nhất trong danh sách để trao học bổng, hãy cho biết tên những học sinh đó.

**Ví dụ:**

Dữ liệu vào	Kết quả câu a	Kết quả câu b	Kết quả câu c
6 Vu Anh Quan 8.9 Nguyen Van Chung 8.7 Hoang Trong Quynh 8.5 Dinh Quang Hoang 8.7 Dinh Quang Huy 8.8 Cong Hoang 8.0	Nguyen Van Chung Cong Hoang Dinh Quang Hoang Dinh Quang Huy Vu Anh Quan Hoang Trong Quynh	5 Chung Hoang Huy Quan Quynh	Vu Anh Quan Dinh Quang Huy Dinh Quang Hoang Nguyen Van Chung

3.2. Cho dãy số gồm  $N$  số nguyên  $a_1 \leq a_2 \leq \dots \leq a_N$ .

- a) Đưa ra thuật toán có độ phức tạp  $O(N \log N)$  để tìm hai chỉ số  $i, j$  mà  $i < j$  mà  $a_i + a_j = 0$ .
- b) Đưa ra thuật toán có độ phức tạp  $O(N^2 \log N)$  để tìm ba chỉ số  $i, j, k$  mà  $i < j < k$  và  $a_i + a_j + a_k = 0$ .
- c) Đưa ra thuật toán có độ phức tạp  $O(N)$  để tìm hai chỉ số  $i, j$  mà  $i < j$  và  $a_i + a_j = 0$ .
- d) Đưa ra thuật toán có độ phức tạp  $O(N^2)$  để tìm ba chỉ số  $i, j, k$  mà  $i < j < k$  và  $a_i + a_j + a_k = 0$ .

3.3. Cho một xâu  $s$  (độ dài không quá 200) chỉ gồm các kí tự 'a' đến 'z', đếm số lượng xâu con liên tiếp khác nhau nhận được từ xâu  $s$ .

Ví dụ,  $s = 'abab'$ , ta có các xâu con liên tiếp khác nhau là:

'a', 'b', 'ab', 'ba', 'aba', 'bab', 'abab',

số lượng xâu con liên tiếp khác nhau là 7.

3.4. Viết liên tiếp các số tự nhiên từ 1 đến  $N$  ta được một số nguyên  $M$ . Ví dụ  $N = 15$  ta có  $M = 123456789101112131415$ . Hãy tìm cách xoá đi  $K$  chữ số của số  $M$  để nhận được số  $M'$  là lớn nhất.

3.5. Xét tập  $F(N)$  tất cả các số hữu tỉ trong đoạn  $[0,1]$  với mẫu số không vượt quá  $N$  ( $1 < N \leq 100$ ).

Ví dụ, tập  $F(5)$ :

$$\frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}$$

Sắp xếp các phân số trong tập  $F(N)$  theo thứ tự tăng dần, đưa ra phân số thứ  $k$ .

- 3.6. Cho xâu  $s$  (độ dài không vượt quá  $10^6$ ) chỉ gồm các kí tự 'a' đến 'z',
  - a) Có bao nhiêu loại kí tự xuất hiện trong  $s$ ?
  - b) Đưa ra một kí tự xuất hiện nhiều nhất trong xâu  $s$  và số lần xuất hiện của kí tự đó.
- 3.7. Cho hai dãy  $a_1 \leq a_2 \leq \dots \leq a_n$  và  $b_1 \leq b_2 \leq \dots \leq b_m$ , hãy đưa ra thuật toán có độ phức tạp  $O(n + m)$  để có dãy  $c_1 \leq c_2 \leq \dots \leq c_{n+m}$  là dãy trộn của hai dãy trên.

- 3.8. Cho dãy số gồm  $n$  ( $n \leq 10000$ ) số nguyên  $a_1, a_2, \dots, a_n$  ( $|a_i| \leq 10^9$ ), tìm số nguyên  $X$  bất kì để  $S = |a_1 - X| + |a_2 - X| + \dots + |a_n - X|$  đạt giá trị nhỏ nhất. Có bao nhiêu giá trị nguyên khác nhau thoả mãn.

Ví dụ 1, dãy gồm 5 số 3, 1, 5, 4, 5, ta có duy nhất một giá trị  $X = 4$  để  $S$  đạt giá trị nhỏ nhất bằng 6.

Ví dụ 2, dãy gồm 6 số 3, 1, 7, 2, 5, 7 ta có ba giá trị nguyên của  $X$  là 3, 4, 5 để  $S$  đạt giá trị nhỏ nhất bằng 13.

- 3.9. Cho  $N$  ( $N \leq 10000$ ) điểm trên mặt phẳng  $Oxy$ , điểm thứ  $i$  có toạ độ là  $(x_i, y_i)$ . Ta định nghĩa khoảng cách giữa hai điểm  $P(x_P, y_P)$  và  $Q(x_Q, y_Q)$  bằng  $|x_P - x_Q| + |y_P - y_Q|$ . Hãy tìm điểm  $A$  có toạ độ nguyên mà tổng khoảng cách (theo cách định nghĩa trên) từ  $A$  tới  $N$  điểm đã cho là nhỏ nhất ( $|x_i|, |y_i|$  nguyên không vượt quá  $10^9$ ).

- 3.10. Cho  $N$  ( $N \leq 10000$ ) đoạn thẳng trên trục số với các điểm đầu  $x_i$  và độ dài  $d_i$  ( $|x_i|$ ,  $d_i$  là những số nguyên và không vượt quá  $10^9$ ). Tính tổng độ dài trên trục số bị phủ bởi  $N$  đoạn trên.

Ví dụ, có ba đoạn  $x_1 = 5, d_1 = 10; x_2 = 0, d_2 = 6; x_3 = -100, d_2 = 10$  thì tổng độ dài trên trục số bị phủ bởi ba đoạn trên là 21.

- 3.11. Cho  $N$  ( $N \leq 300$ ) điểm trên mặt phẳng  $Oxy$ , điểm thứ  $i$  có toạ độ là  $(x_i, y_i)$ . Hãy đếm số cách chọn bốn điểm trong  $N$  điểm trên mà bốn điểm đó là bốn đỉnh của một hình chữ nhật ( $|x_i|, |y_i|$  nguyên không vượt quá  $1000$ ).

Ví dụ, có 5 điểm  $(0, 0), (0, 1), (1, 0), (-1, 0), (0, -1)$  có duy nhất một cách chọn 4 điểm mà 4 điểm đó là 4 đỉnh của một hình chữ nhật.

- 3.12. Cho  $N$  ( $N \leq 10000$ ) đoạn số nguyên  $(a_i, b_i)$ . hãy tìm một số mà số đó thuộc nhiều đoạn số nguyên nhất.

Ví dụ, có 5 đoạn  $[0, 10], [2, 3], [4, 7], [3, 5], [5, 8]$ , ta chọn số 5 thuộc 4 đoạn  $[0, 10], [4, 7], [3, 5], [5, 8]$ .

- 3.13. Cho dãy gồm  $N$  ( $N \leq 10000$ ) số  $a_1, a_2, \dots, a_N$ . Hãy tìm dãy con liên tiếp dài nhất có tổng bằng 0 ( $|a_i| \leq 10^9$ ).

Ví dụ, dãy gồm 5 số 2, 1, -2, 3, -2 thì dãy con liên tiếp dài nhất có tổng bằng 0 là: 1, -2, 3, -2.

### 3.14. ESEQ

Cho dãy số nguyên  $A$  gồm  $N$  phần tử  $A_1, A_2, \dots, A_N$ . Tìm số cặp chỉ số  $i, j$  thoả mãn:

$$\sum_{p=1}^i A_p = \sum_{q=j}^N A_q \text{ với } 1 \leq i < j \leq N.$$

**Input:** Tệp văn bản ESEQ.INP có dạng:

- Dòng đầu là số nguyên dương  $N$  ( $2 \leq N \leq 10^5$ ).
- Dòng tiếp theo chứa  $N$  số nguyên  $A_1, A_2, \dots, A_N$  ( $|A_i| < 10^9$ ), các số cách nhau một dấu cách.

**Output:** Tệp văn bản ESEQ.OUT gồm một số là số cặp tìm được.

**Ví dụ:**

ESEQ.INP	ESEQ.OUT
3	3
1 0 1	

### 3.15. GHÉP SỐ

Cho  $n$  số nguyên dương  $a_1, a_2, \dots, a_n$  ( $1 < n \leq 100$ ), mỗi số không vượt quá  $10^9$ . Từ các số này người ta tạo ra một số nguyên mới bằng cách ghép tất cả các số đã cho, tức là viết liên tiếp các số đã cho với nhau. Ví dụ, với  $n = 4$  và các số 123, 124, 56, 90 ta có thể tạo ra các số mới sau: 1231245690, 1241235690, 5612312490, 9012312456, 9056124123,... Có thể dễ dàng thấy rằng, với  $n = 4$ , ta có thể tạo ra 24 số mới. Trong trường hợp này, số lớn nhất có thể tạo ra là 9056124123.

**Yêu cầu:** Cho  $n$  và các số  $a_1, a_2, \dots, a_n$ . Hãy xác định số lớn nhất có thể tạo ra khi ghép các số đã cho thành một số mới.

**Input:** Tệp văn bản NUMJOIN.INP:

- Dòng thứ nhất chứa số nguyên  $n$ ;
- Dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$ .

**Output:** Tệp văn bản NUMJOIN.OUT gồm một dòng là số lớn nhất có thể tạo ra khi ghép các số đã cho thành một số mới.

### 3.16. GIÁ TRỊ NHỎ NHẤT

Cho bảng số  $A$  gồm  $M \times N$  ô, mỗi ô chứa một số nguyên không âm ( $A_{ij}$ ) có giá trị không vượt quá  $10^9$ . Xét hàng  $i$  và hàng  $j$  của bảng, ta cần xác định  $X_{ij}$  nguyên để:

$$S_{ij} = \left( \sum_{k=1}^N |A_{ik} - X_{ij}| \right) + \left( \sum_{k=1}^N |A_{jk} - X_{ij}| \right) \text{ đạt giá trị nhỏ nhất.}$$

Tính  $W = \sum_{i=1}^{M-1} \sum_{j=i+1}^M S_{ij}$ .

**Input:** Tệp văn bản WMT.INP:

- Dòng đầu là hai số nguyên dương  $M, N (1 < M, N < 100)$ .
- $M$  dòng sau, mỗi dòng  $N$  số.

**Output:** Tệp văn bản WMT.OUT gồm một số  $W$ .

**Ví dụ:**

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

### 3.17. DECIPHERING THE MAYAN WRITING (IOI 2006)

Công việc giải mã chữ viết của người Maya là khó khăn hơn người ta tưởng nhiều. Trải qua hơn 200 năm mà người ta vẫn hiểu rất ít về các chữ viết này. Chỉ trong ba thập niên gần đây do công nghệ phát triển việc giải mã này mới có nhiều tiến bộ.

Chữ viết Maya dựa trên các kí hiệu nhỏ gọi là nét vẽ, mỗi nét vẽ tương ứng với một âm giọng nói. Mỗi từ trong chữ viết Maya sẽ bao gồm một tập hợp các nét vẽ như vậy kết hợp lại với nhiều kiểu dáng khác nhau. Mỗi nét vẽ có thể hiểu là một kí tự.

Một trong những vấn đề lớn khi giải mã chữ Maya là thứ tự đọc các nét vẽ. Do người Maya trình bày các nét vẽ này không theo thứ tự phát âm, mà theo cách thể hiện của chúng. Do vậy nhiều khi đã biết hết các nét vẽ của một từ rồi nhưng vẫn không thể tìm ra được chính xác cách ghi và đọc của từ này.

Các nhà khảo cổ đang đi tìm kiếm một từ đặc biệt  $W$ . Họ đã biết rõ tất cả các nét vẽ của từ này nhưng vẫn chưa biết các cách viết ra của từ này. Vì họ

có  
 $X_{ij}$   
biết có các thí sinh IOI'06 sẽ đến nên muốn sự trợ giúp của các thí sinh này. Họ sẽ đưa ra toàn bộ  $g$  nét vẽ của từ  $W$  và dãy  $S$  tất cả các nét vẽ có trong hang đá cổ. Bạn hãy giúp các nhà khảo cổ tính xem có bao nhiêu khả năng xuất hiện từ  $W$  trong hang đá.

**Yêu cầu:** Hãy viết chương trình, cho trước các kí tự của từ  $W$  và dãy  $S$  các nét vẽ trong hang đá, tính tổng số khả năng xuất hiện của từ  $W$  trong dãy  $S$ , nghĩa là số lần xuất hiện một hoán vị các kí tự của dãy  $g$  kí tự trong  $S$ .

#### Các ràng buộc

- $1 \leq g \leq 3\,000$ , số nét vẽ trong  $W$ .
- $g \leq |S| \leq 3\,000\,000$ ,  $|S|$  là số các nét vẽ của dãy  $S$ .

#### Input:

Tệp có dạng

- Dòng thứ nhất: chứa hai số  $g$  và  $|S|$  cách nhau bởi dấu cách.
- Dòng thứ hai: chứa  $g$  kí tự liền nhau là các nét vẽ của từ  $W$ . Các kí tự hợp lệ là ' $a'$ -' $z'$  và ' $A'$ -' $Z'$ . Các chữ in hoa và in thường là khác nhau.
- Dòng thứ ba: Chứa  $|S|$  kí tự là dãy các nét vẽ tìm thấy trong hang. Các kí tự hợp lệ là ' $a'$ -' $z'$  và ' $A'$ -' $Z'$ . Các chữ in hoa và in thường là khác nhau.

#### Output:

Tệp chứa đúng một số là khả năng xuất hiện của từ  $W$  trong dãy  $S$ .

#### Ví dụ:

Dữ liệu vào	Kết quả ra
4 11 cAda AbrAcadAbRa	2

### 3.18. TRÒ CHƠI VỚI DÃY SỐ (Đề thi học sinh giỏi quốc gia, 2007-2008)

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm  $n$  số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là:  $b_1, b_2, \dots, b_n$ , còn dãy số mà bạn thứ hai chọn là:  $c_1, c_2, \dots, c_n$ .

Mỗi lượt chơi, mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng  $b_i$  ( $1 \leq i \leq n$ ), còn bạn thứ hai đưa ra số hạng  $c_j$  ( $1 \leq j \leq n$ ) thì giá của lượt chơi đó sẽ là  $|b_i + c_j|$ .

Ví dụ, giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3),

(-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

**Yêu cầu:** Hãy xác định giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

**Input:** Tệp có dạng

- Dòng đầu tiên chứa số nguyên dương  $n$  ( $n \leq 10^5$ ).
- Dòng thứ hai chứa dãy số nguyên  $b_1, b_2, \dots, b_n$  ( $|b_i| \leq 10^9$ ,  $i = 1, 2, \dots, n$ ).
- Dòng thứ ba chứa dãy số nguyên  $c_1, c_2, \dots, c_n$  ( $|c_i| \leq 10^9$ ,  $i = 1, 2, \dots, n$ ).

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

**Output:** Tệp ghi ra giá nhỏ nhất tìm được.

**Ví dụ:**

Dữ liệu vào	Kết quả ra
2	0
1 -2	
2 3	

### 3.19. DÃY SỐ (Đề thi học sinh giỏi Hà Nội, 2008-2009)

Cho dãy số nguyên  $a_1, a_2, \dots, a_n$ . Số  $a_p$  ( $1 \leq p \leq n$ ) được gọi là một số trung bình cộng trong dãy nếu tồn tại ba chỉ số  $i, j, k$  ( $i, j, k \leq n$ ) đôi một khác nhau,

$$\text{saو cho } a_p = \frac{a_i + a_j + a_k}{3}.$$

**Yêu cầu:** Cho  $n$  và dãy số  $a_1, a_2, \dots, a_n$ . Hãy tìm số lượng các số trung bình cộng trong dãy.

**Input:** Tệp có dạng

- Dòng đầu ghi số nguyên dương  $n$  ( $3 \leq n \leq 1000$ ).
- Dòng thứ hai chứa  $n$  số nguyên  $a_i$  ( $a_i < 10^8$ ).

**Output:** Tệp ghi số lượng các số trung bình cộng trong dãy.

**Ví dụ:**

Dữ liệu vào	Kết quả ra
5	2
4 3 6 3 5	

### 3.20. ĐẾM SỐ TAM GIÁC (Đề thi Tin học trẻ, bảng B, năm 2009)

Cho ba số nguyên dương  $a, b, m$  và  $n$  đoạn thẳng đánh số từ 1 tới  $n$  ( $m, n \leq 10000$ ). Đoạn thẳng thứ  $i$  có độ dài  $d_i$  ( $1 \leq i \leq n$ ), ở đây các độ dài  $(d_1, d_2, \dots, d_n)$  được cho như sau:

$$d_i = \begin{cases} b & \text{nếu } i = 1 \\ (a \times d_{i-1} + b) \bmod m + 1 & \text{nếu } 1 < i \leq n \end{cases} \quad (*)$$

Hãy cho biết có bao nhiêu tam giác khác nhau có thể được tạo ra bằng cách lấy đúng ba đoạn trong số  $n$  đoạn thẳng đã cho làm ba cạnh (hai tam giác bằng nhau nếu chúng có ba cặp cạnh tương ứng bằng nhau, nếu không chúng được coi là khác nhau).

Ví dụ với  $a = 6; b = 3; m = 4; n = 5$ . Ta có 5 đoạn thẳng với độ dài của chúng tính theo công thức (\*) là  $(3, 2, 4, 4, 4)$ . Với 5 đoạn thẳng này có thể tạo ra được 4 tam giác với độ dài các cạnh được chỉ ra như sau:

Tam giác 1:  $(2, 3, 4)$

Tam giác 2:  $(2, 4, 4)$

Tam giác 3:  $(3, 4, 4)$

Tam giác 4:  $(4, 4, 4)$ .

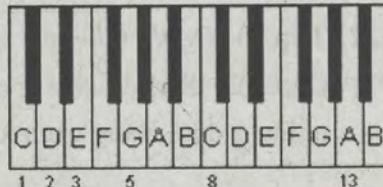
### Bài tập bổ sung

#### 3.21. Bản vanxơ Fibonacci (Đề thi học sinh giỏi Quốc gia, năm 2012)

Bản vanxơ Fibonacci là một bản nhạc mà giai điệu của nó bắt nguồn từ một trong những dãy số nổi tiếng nhất trong lí thuyết số - dãy số Fibonacci. Hai số đầu tiên của dãy là số 1 và số 2, các số tiếp theo được xác định bằng tổng của hai số liên tiếp ngay trước nó trong dãy.

Bản vanxơ Fibonacci thu được bằng việc chuyển dãy số Fibonacci thành dãy các nốt nhạc theo quy tắc chuyển một số nguyên dương thành nốt nhạc sau đây:

- số 1 tương ứng với nốt Đô (C),
- số 2 tương ứng với nốt Rê (D),
- số 3 tương ứng với nốt Mi (E),
- số 4 tương ứng với nốt Fa (F),



- số 5 tương ứng với nốt Sol (G),
- số 6 tương ứng với nốt La (A),
- số 7 tương ứng với nốt Si (B),
- số 8 tương ứng với nốt Đô (C),
- số 9 tương ứng với nốt Rê (D)

và cứ tiếp tục như vậy. Ví dụ, dãy gồm 6 số Fibonacci đầu tiên 1, 2, 3, 5, 8 và 13 tương ứng với dãy các nốt nhạc C, D, E, G, C và A.

Để xây dựng nhịp điệu vanxơ người ta đi tìm các đoạn nhạc có tính chu kì trong bản vanxơ Fibonacci. Đoạn nhạc được gọi là có tính chu kì nếu như có thể chia nó ra thành  $k$  đoạn giống hệt nhau ( $k \geq 2$ ). Ví dụ, đoạn nhạc GCAGCA là đoạn có tính chu kì, vì nó gồm hai đoạn giống nhau GCA.

**Yêu cầu:** Cho trước hai số nguyên dương  $u, v$  ( $u < v$ ). Hãy xác định độ dài đoạn nhạc dài nhất có tính chu kì của *bản nhạc* gồm dãy các nốt nhạc của bản vanxơ Fibonacci bắt đầu từ vị trí  $u$  kết thúc ở vị trí  $v$ .

**Input:** Tệp văn bản FIBVAL.INP

- Dòng thứ nhất chứa số nguyên dương  $k$  ( $k \leq 100$ ) là số lượng bộ kiểm thử;
- Dòng thứ  $i$  trong số  $k$  dòng tiếp theo chứa hai số nguyên dương  $u_i, v_i$  được ghi cách nhau bởi dấu cách ( $u_i < v_i \leq 10^9$ ) là vị trí bắt đầu và kết thúc của một bản nhạc;

**Output:** Tệp văn bản FIBVAL.OUT gồm  $k$  dòng, dòng thứ  $i$  chứa một số nguyên là độ dài đoạn nhạc tìm được tương ứng với bộ kiểm thử thứ  $i$ . Nếu không tìm được đoạn nào có tính chu kì thì ghi ra số  $-1$ .

**Ví dụ:**

FIBVAL.INP	FIBVAL.OUT
?	-1
1 3	2
4 10	

### 3.22. Tập số (Đề thi Olympic sinh viên, năm 2011)

Cho số  $n$  ở hệ cơ số 10, có không quá 20 chữ số và không chứa các số 0 vô nghĩa ở đầu. Bằng cách xoá một hoặc một vài chữ số liên tiếp của  $n$  (nhưng

không xoá hết tất cả các chữ số của  $n$ ) ta nhận được những số mới. Số mới được chuẩn hoá bằng cách xoá các chữ số 0 vô nghĩa nếu có. Tập số nguyên  $D$  được xây dựng bằng cách đưa vào số  $n$ , các số mới khác nhau đã chuẩn hoá và khác  $n$ . Ví dụ, với  $n = 1005$  ta có thể nhận được các số mới như sau:

- Bằng cách xoá một chữ số ta có các số: 5 (từ 005), 105, 105, 100;
- Bằng cách xoá hai chữ số ta có các số: 5 (từ 05), 15, 10;
- Bằng cách xoá 3 chữ số ta có các số: 5 và 1.

Tập  $D$  nhận được từ  $n$  chứa các số  $\{1005, 105, 100, 15, 10, 5, 1\}$ . Trong tập  $D$  này có 3 số chia hết cho 3, đó là các số 1005, 105 và 15.

**Yêu cầu:** Cho số nguyên  $n$ . Hãy xác định số lượng số chia hết cho 3 có mặt trong tập  $D$  được tạo thành từ  $n$ .

**Input:** Tập văn bản NUMSET.INP gồm một dòng chứa số nguyên  $n$ .

**Output:** Tập văn bản NUMSET.OUT một số nguyên - số lượng số chia hết cho 3 tìm được.

**Ví dụ:**

NUMSET.INP
1005

NUMSET.OUT
3

### 3.23. Đường thẳng

Cho  $n$  đường thẳng trên mặt phẳng ( $1 \leq n \leq 1000$ ), mỗi đường thẳng cho dưới dạng phương trình  $Ax + By + C = 0$ , trong đó các hệ số  $A, B, C$  đều là các số nguyên ( $|A|, |B|, |C| \leq 10^5, A^2 + B^2 \neq 0$ ).

**Yêu cầu:** Hãy xác định số lượng tối đa các đường thẳng, sao cho giữa chúng không có cặp đường thẳng nào song song (hai đường thẳng trùng nhau được coi là song song).

**Input:** Tập văn bản LINES.INP:

- Dòng đầu tiên chứa số nguyên  $n$ .
- Mỗi dòng trong  $n$  dòng tiếp theo chứa 3 số nguyên  $A, B, C$  xác định một đường thẳng.

**Output:** Tập văn bản LINES.OUT ghi số đường tối đa tìm được.

**Ví dụ:**

LINES.INP	LINES.OUT
3	2
1 1 4	
2 2 2	
3 1 0	

**3.24. Hình chữ nhật bốn màu** (*Đề thi học sinh giỏi Quốc gia, năm 2010*)

Trên mặt phẳng toạ độ *Oxy* cho  $n$  điểm phân biệt  $A_i(x_i, y_i)$ ,  $i = 1, 2, \dots, n$ . Mỗi điểm  $A_i$  được tô bởi màu  $c_i \in \{1, 2, 3, 4\}$ . Ta gọi hình chữ nhật bốn màu là hình chữ nhật thỏa mãn hai điều kiện sau:

- Bốn đỉnh của hình chữ nhật là bốn điểm trong  $n$  điểm đã cho và được tô bởi bốn màu khác nhau;
- Các cạnh của hình chữ nhật song song với trục toạ độ.

**Yêu cầu:** Cho biết toạ độ và màu của  $n$  điểm. Hãy đếm số lượng hình chữ nhật bốn màu.

**Input:** Tệp văn bản COLOREC.INP:

- Dòng đầu tiên chứa số nguyên dương  $n$  ( $4 \leq n \leq 105$ ) là số lượng điểm trên mặt phẳng.
- Dòng thứ  $i$  trong  $n$  dòng tiếp theo chứa ba số nguyên  $x_i, y_i, c_i$  ( $|x_i|, |y_i| \leq 200$ ) là thông tin về toạ độ và màu của điểm thứ  $i$ ,  $i = 1, 2, \dots, n$ .

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

**Output:** Tệp văn bản COLOREC.OUT gồm một dòng ghi số lượng hình chữ nhật đếm được.

**Ví dụ:**

COLOREC.INP	COLOREC.OUT
7	2
0 0 1	
0 1 4	
2 1 2	
2 -1 3	

0 -1 1	
-1 -1 4	
-1 1 1	

### 3.25. Phần tử chung

Cho  $k$  dãy số nguyên, các số trong dãy thuộc  $[-10^9..10^9]$ . Hãy viết chương trình tìm số xuất hiện trong  $k$  dãy. Nếu không có số nào xuất hiện trong  $k$  dãy thì ghi kí tự 'x', còn nếu có nhiều số cùng xuất hiện trong  $k$  dãy thì ghi số nhỏ nhất tìm được.

**Input:** Tệp văn bản PTC.INP:

- Dòng 1: chứa số nguyên dương  $k$  ( $k > 1$ );
- Dòng 2: gồm  $k$  số lần lượt là độ dài từng dãy;
- $k$  dòng sau, mỗi dòng mô tả một dãy số, biết rằng tổng các số trong  $k$  dãy không vượt quá 500000.

**Output:** Tệp văn bản PTC.OUT ghi số tìm được hoặc kí tự 'x'.

**Ví dụ:**

PTC.INP	PTC.OUT
2	2
3 4	
1 2 3	
4 3 2 -1	
3	3
5 6 7	
2 1 3 4 3	
4 5 -1 0 0 3	
11 4 7 8 9 0 3	

### 3.26. Nhà mạng XYZ

Trong dịp kỉ niệm 100 năm thành lập, nhà mạng XYZ triển khai chương trình “Thuê bao vàng” như sau: Mỗi ngày, kể từ thời điểm 0 giờ 0 phút 0 giây đến thời điểm 23 giờ 59 phút 59 giây, nhà mạng sẽ thông kê tất cả cuộc gọi để chọn ra thuê bao tích cực nhất trong ngày. Độ tích cực của một thuê

bao tính theo công thức: tổng số giây trong các cuộc gọi đi của thuê bao nhân với 2, cộng với tổng số giây mà thuê bao nhận các cuộc gọi. Thuê bao tích cực nhất là thuê bao có độ tích cực lớn nhất. Những thuê bao này sẽ được nhận các chương trình ưu đãi của nhà mạng.

**Yêu cầu:** Cho thông tin các cuộc gọi trong ngày. Hãy tính độ tích cực của thuê bao tích cực nhất.

**Input:** Tệp văn bản XYZ.INP:

- Dòng đầu ghi số  $n$  ( $0 < n \leq 30000$ ) là số cuộc gọi được thực hiện trong ngày;
- $n$  dòng sau, mỗi dòng chứa một xâu mô tả về một cuộc gọi, cụ thể:
  - 10 kí tự số đầu tiên của xâu mô tả số của thuê bao thực hiện cuộc gọi;
  - Tiếp theo là một dấu cách;
  - 10 kí tự số tiếp theo của xâu mô tả số của thuê bao nhận cuộc gọi;
  - Tiếp theo là một dấu cách;
  - 6 kí tự số tiếp theo của xâu mô tả thời điểm bắt đầu cuộc gọi: hai kí tự đầu mô tả giờ (từ 00 đến 23), hai kí tự sau mô tả phút (từ 00 đến 59), hai kí tự cuối mô tả giây (từ 00 đến 59).
  - Tiếp theo là một dấu cách;
  - 6 kí tự số cuối cùng của xâu mô tả thời điểm kết thúc cuộc gọi: hai kí tự đầu mô tả giờ (từ 00 đến 23), hai kí tự sau mô tả phút (từ 00 đến 59), hai kí tự cuối mô tả giây (từ 00 đến 59).

**Output:** Tệp văn bản XYZ.OUT gồm một dòng ghi một số nguyên là độ tích cực của thuê bao tích cực nhất.

**Ví dụ:**

XYZ.INP	XYZ.OUT
3 0123456789 1234567890 015915 015945 8888888888 0123456789 015949 020049 9999999999 6666666666 225915 230000	120

## CHUYÊN ĐỀ 4. THIẾT KẾ THUẬT TOÁN

- 4.1. Cho danh sách tên của  $n$  học sinh (các tên đôi một khác nhau,  $n \leq 10$ ) và một số nguyên dương  $k$  ( $k \leq n$ ). Hãy liệt kê tất cả các cách chọn  $k$  học sinh trong  $n$  học sinh.

Ví dụ:

Dữ liệu vào	Kết quả ra
$n = 4$ , $k = 2$ , danh sách tên học sinh như sau: An Binh Hong Minh	Có 6 cách chọn 2 học sinh trong 4 học sinh: 1. An Binh 2. An Hong 3. An Minh 4. Binh Hong 5. Binh Minh 6. Hong Minh

- 4.2. Một dãy nhị phân độ dài  $n$  ( $n \leq 10$ ) là một dãy  $x = x_1x_2\dots x_n$  trong đó  $x_i \in \{0, 1\}$ ,  $i = 1, 2, \dots, n$ . Hãy liệt kê tất cả các dãy nhị phân có độ dài  $n$ .

Ví dụ:

Dữ liệu vào	Kết quả ra
$n = 3$	Có 8 dãy nhị phân độ dài 3 1. 000 2. 001 3. 010 4. 011 5. 100 6. 101 7. 110 8. 111

- 4.3. Cho xâu  $S$  (độ dài không vượt quá 10) chỉ gồm các kí tự 'A' đến 'Z' (các kí tự trong xâu  $S$  đôi một khác nhau). Hãy liệt kê tất cả các hoán vị khác nhau của xâu  $S$ .

Ví dụ:

Dữ liệu vào	Kết quả ra
S='XYZ'	Có 6 hoán vị khác nhau của 'XYZ' 1. XYZ 2. XZY 3. YXZ 4. YZX 5. ZXY 6. ZYX

- 4.4. Cho số nguyên dương  $n$  ( $n \leq 20$ ). Hãy liệt kê tất cả các xâu độ dài  $n$  chỉ gồm hai kí tự 'A' hoặc 'B' mà không có hai kí tự 'B' nào đứng cạnh nhau.

Ví dụ:

Dữ liệu vào	Kết quả ra
$n = 4$	Có 8 xâu độ dài 4 1. AAAA 2. AAAB 3. AABA 4. ABAA 5. ABAB 6. BAAA 7. BAAB 8. BABA

- 4.5. Cho dãy số  $A$  gồm  $N$  ( $N \leq 10$ ) số nguyên  $a_1, a_2, \dots, a_N$  và một số nguyên dương  $K$  ( $1 < K < N$ ). Hãy đưa ra một cách chia dãy số thành  $K$  nhóm mà các nhóm có tổng bằng nhau.

Ví dụ:

Dữ liệu vào	Kết quả ra
$N=5, K=3$	nhóm 1: 4, 6
Dãy số A: 1, 4, 6, 9, 10	nhóm 2: 1, 9 nhóm 3: 10

- 4.6. Một xâu  $X = x_1x_2\dots x_M$  được gọi là xâu con của xâu  $Y = y_1y_2\dots y_N$  nếu ta có thể nhận được xâu  $X$  từ xâu  $Y$  bằng cách xoá đi một số kí tự, tức là tồn tại một dãy các chỉ số:

$$1 \leq i_1 < i_2 < \dots < i_M \leq N \text{ để } x_1 = y_{i1}, x_2 = y_{i2}, x_M = y_{iM}.$$

Ví dụ,  $X = 'adz'$  là xâu con của xâu  $Y = 'baczdtz'$ ;

$$i_1 = 2 < i_2 = 5 < i_3 = 7$$

Nhập vào một xâu  $S$  (độ dài không quá 15, chỉ gồm các kí tự 'a' đến 'z'). Hãy liệt kê tất cả các xâu con khác nhau của xâu  $S$ .

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$S = 'aba'$	Có 6 xâu con khác nhau của 'aba' 1. a 2. b 3. aa 4. ab 5. ba 6. aba

- 4.7. Cho số nguyên dương  $n$  ( $n \leq 10$ ). Liệt kê tất cả các cách khác nhau đặt  $n$  dấu ngoặc mở và  $n$  dấu ngoặc đóng đúng đắn?

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$n = 3$	Có 5 cách $(( ))$ , $(( )( ))$ , $(( )( ))$ , $( )( ( ))$ , $( )( )( )$

- 4.8. Cho  $n$  ( $n \leq 10$ ), số nguyên dương  $a_1, a_2, \dots, a_n$  ( $a_i \leq 10^9$ ). Tìm số nguyên dương  $m$  nhỏ nhất sao cho  $m$  không phân tích được dưới dạng tổng của một số các số (mỗi số sử dụng không quá một lần) thuộc  $n$  số trên.

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$n = 4$ Dãy số a: 1, 2, 3, 6	13

- 4.9. Cho xâu  $S$  (độ dài không vượt quá 10) chỉ gồm các kí tự 'A' đến 'Z' (các kí tự trong xâu  $S$  không nhất thiết phải khác nhau). Hãy liệt kê tất cả các hoán vị khác nhau của xâu  $S$ .

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$S = 'ABA'$	Có 3 hoán vị khác nhau của 'ABA' 1. AAB 2. ABA 3. BAA

#### 4.10. Bài toán mã đi tuần

Cho bàn cờ  $n \times n$  ô, tìm cách di chuyển một quân mã (mã di chuyển theo luật cờ vua) trên bàn cờ xuất phát từ ô  $(1,1)$  đi qua tất cả các ô, mỗi ô qua đúng một lần.

**Ví dụ,  $n = 5$**

1	24	13	18	7
14	19	8	23	12
9	2	25	6	17
20	15	4	11	22
3	10	21	16	5

- 4.11. Số siêu nguyên tố là số nguyên tố mà khi bỏ một số tuỳ ý các chữ số bên phải của nó thì phần còn lại vẫn là một số nguyên tố.

Ví dụ, 2333 là một số siêu nguyên tố có 4 chữ số vì 233, 23, 2 cũng là các số nguyên tố.

Cho số nguyên dương  $N$  ( $0 < N < 10$ ), đưa ra các số siêu nguyên tố có  $N$  chữ số cùng số các số đó.

**Ví dụ, với  $N = 4$**

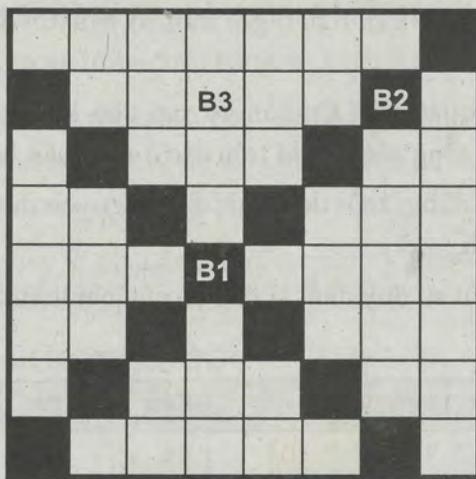
Có 16 số:

2333	2339	2393	2399	2939	3119	3137	3733
3739	3793	3797	5939	7193	7331	7333	7393

- 4.12. Cho một xâu  $S$  (chỉ gồm các kí tự '0' đến '9', độ dài nhỏ hơn 10) và số nguyên  $M$ , hãy đưa ra một cách chèn vào  $S$  các dấu '+' hoặc '-' để thu được số  $M$  cho trước (nếu có thể).

Ví dụ,  $M = 8$ ,  $S = '123456789'$  một cách chèn: ' $-1+2-3+4+5-6+7$ ';

- 4.13. Trong cờ vua quân tượng chỉ có thể di chuyển theo đường chéo và hai quân tượng có thể chiếu nhau nếu chúng nằm trên đường di chuyển của nhau. Trong hình sau, hình vuông tô đậm thể hiện các vị trí mà quân tượng  $B_1$  có thể đi tới được, quân tượng  $B_1$  và  $B_2$  chiếu nhau, quân  $B_1$  và  $B_3$  không chiếu nhau. Cho kích thước  $N$  của bàn cờ và  $K$  quân tượng, hỏi có bao nhiêu cách đặt các quân tượng vào bàn cờ mà các quân tượng không chiếu nhau.



**Input:** Tệp văn bản BISHOP.INP:

- Dòng đầu là số  $t$  là số test ( $t \leq 10$ )
- $t$  dòng sau mỗi dòng chứa 2 số nguyên dương  $N, K$  ( $2 \leq N \leq 10, 0 < K \leq N^2$ )

**Output:** Tệp văn bản BISHOP.OUT gồm  $t$  dòng, mỗi dòng chứa một số duy nhất là số cách đặt các quân tượng vào bàn cờ tương ứng với dữ liệu vào.

- 4.14.  $N$ -mino là hình thu được từ  $N$  hình vuông  $1 \times 1$  ghép lại (cạnh kề cạnh). Hai  $n$ -mino được gọi là đồng nhất nếu chúng có thể đặt chồng khít lên nhau. Cho số nguyên dương  $N$  ( $1 < N < 8$ ), tính và vẽ ra tất cả các  $N$ -mino trên màn hình.

Ví dụ, với  $N = 3$  chỉ có hai loại  $N$ -mino sau đây:



3-mino thẳng 3-mino hình thước thợ

- 4.15. Trong mục II.2 trong sách Tài liệu chuyên Tin học Quyền 1, lời giải bài toán người du lịch (*Travelling Salesman Problem - TSP*) là một giải pháp nhánh cận rất thô sơ. Hãy thử chạy chương trình với trường hợp như sau:

Số thành phố  $n = 20$ , khoảng cách giữa các thành phố bằng 1 (nghĩa là  $C[i, j] = 1$  với  $i \neq j$ ). Hãy rút ra nhận xét và có thể đánh giá nhánh cận chặt hơn nữa làm tăng hiệu quả của chương trình.

- 4.16. Cho bàn cờ quốc tế  $8 \times 8$  ô, mỗi ô ghi một số nguyên dương không vượt quá 32000.

**Yêu cầu:** Xếp 8 quân hậu lên bàn cờ sao cho không quân nào không chế được quân nào và tổng các số ghi trên các ô mà quân hậu đứng là lớn nhất.

**Input:** Tập gồm 8 dòng, mỗi dòng ghi 8 số nguyên dương, giữa các số cách nhau một dấu cách.

**Output:** Tập có một số duy nhất là đáp số của bài toán.

**Ví dụ:**

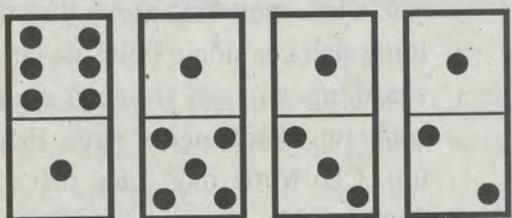
Dữ liệu vào	Kết quả ra
1 2 4 9 3 2 1 4	66
6 9 5 4 2 3 1 4	
3 6 2 3 4 1 8 3	
2 3 7 3 2 1 4 2	
1 2 3 2 3 9 2 1	
2 1 3 4 2 4 2 8	
2 1 3 2 8 4 2 1	
8 2 3 4 2 3 1 2	

- 4.17. Một chiếc ba lô có thể chứa được một khối lượng  $w$ . Có  $n$  đồ vật được đánh số 1, 2, ...,  $n$  ( $n \leq 20$ ). Đồ vật  $i$  có khối lượng  $a_i$  và có giá trị  $c_i$ . Cần chọn các đồ vật cho vào ba lô để tổng giá trị các đồ vật là lớn nhất.

#### 4.18. Đôminô

Có  $N$  quân đôminô xếp thành một hàng như hình vẽ.

Mỗi quân đôminô được chia làm hai phần, phần trên và phần dưới. Trên mặt mỗi phần có các chấm, số chấm từ 1 đến 6.



Ta nhận thấy rằng:

Tổng số chấm ở phần trên của  $N$  quân đôminô bằng:  $6 + 1 + 1 + 1 = 9$ , tổng số chấm ở phần dưới của  $N$  quân đôminô bằng  $1 + 5 + 3 + 2 = 11$ , độ chênh lệch giữa tổng trên và tổng dưới bằng  $|9 - 11| = 2$ .

Với mỗi quân, bạn có thể quay  $180^\circ$  để phần trên trở thành phần dưới, phần dưới trở thành phần trên, khi đó độ chênh lệch có thể được thay đổi. Ví dụ, ta quay quân đôminô cuối cùng của hình trên thì độ chênh lệch bằng 0.

**Yêu cầu:** Cần quay ít nhất bao nhiêu quân đôminô để độ chênh lệch giữa phần trên và phần dưới là nhỏ nhất?

**Input:** Tệp văn bản DOMINO.INP:

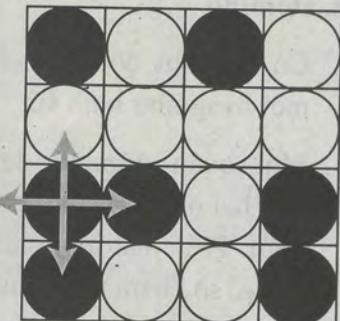
- Dòng đầu là số nguyên dương  $N$  ( $1 \leq N \leq 20$ ).
- $N$  dòng sau, mỗi dòng là hai số  $a_i, b_i$  tương ứng với số chấm của phần trên, phần dưới của quân đôminô thứ  $i$  ( $1 \leq a_i, b_i \leq 6$ ).

**Output:** Tệp văn bản DOMINO.OUT: Gồm một dòng duy nhất chứa hai số nguyên cách nhau một dấu cách là độ chênh lệch nhỏ nhất và số quân đôminô cần quay ít nhất để được độ chênh lệch đó.

**4.19.** Cho một lưới  $M \times N$  ô ( $M, N \leq 10$ ), trên mỗi ô đặt một bóng đèn bật hoặc tắt. Trên mỗi dòng và mỗi cột có một công tắc. Nếu tác động vào công tắc dòng  $i$  ( $i = 1..M$ ) hoặc công tắc cột  $j$  ( $j = 1..N$ ) thì tắt cả các bóng đèn trên dòng  $i$  hoặc cột  $j$  sẽ thay đổi trạng thái. Hãy tìm cách tác động vào các công tắc để được nhiều đèn sáng nhất.

**4.20.** Có 16 đồng xu xếp thành bảng  $4 \times 4$ , mỗi đồng xu có thể úp hoặc ngửa như hình vẽ sau:

Tại mỗi bước ta có phép biến đổi sau: Chọn một đồng xu và thay đổi trạng thái của đồng xu đó và tất cả các đồng xu nằm ở các ô chung cạnh (úp thành ngửa, ngửa thành úp). Cho trước một trạng thái các đồng xu, hãy lập trình tìm số phép biến đổi ít nhất để đưa về trạng thái tất cả các đồng xu hoặc đều úp hoặc đều ngửa.



Màu đen thể hiện đồng xu úp, màu trắng thể hiện đồng xu ngửa.

**Input:** Tệp văn bản COIN.INP: Gồm bốn dòng, mỗi dòng bốn ký tự 'w' - mô tả trạng thái ngửa hoặc 'b' - mô tả trạng thái úp.

**Output:** Tệp văn bản COIN.OUT: Nếu có thể biến đổi được ghi số phép biến đổi ít nhất nếu không ghi "Impossible"

**Ví dụ:**

COIN.INP	COIN.OUT	COIN.INP	COIN.OUT
bwwb	<b>Impossible</b>	bwwb	
wwww		bbwb	
bbwb		bwwb	
bwwb		bwww	

4.21. Có  $N$  tệp chương trình với dung lượng  $S_1, S_2, \dots, S_N$  và loại đĩa CD có dung lượng  $D$ . Hỏi cần ít nhất bao nhiêu đĩa CD để có thể sao chép đủ tất cả các tệp chương trình (mỗi tệp chương trình chỉ nằm trong một đĩa CD)?

- a) Giải bài toán bằng phương pháp nhánh cận với  $N \leq 10$ .
- b) Giải bài toán bằng một thuật toán tham lam với  $N \leq 100$ .

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$N = 5, D = 700$ 320, 100, 300, 560, 50	Cần ít nhất 2 đĩa CD Đĩa 1: 320, 300, 50 Đĩa 2: 100, 560

- 4.22. Chương trình giải bài toán lập lịch giảm thiểu trễ hạn (ở mục III.4, trong Tài liệu chuyên Tin học Quyển 1) có độ phức tạp  $O(N^3)$ . Hãy cải tiến hàm *check* để nhận được chương trình với độ phức tạp  $O(N^2)$
- 4.23. Cho một xâu  $S$  (độ dài không quá 200) chỉ gồm ba loại kí tự 'A', 'B', 'C'. Ta có phép đổi chỗ hai kí tự bất kì trong xâu. Hãy tìm cách biến đổi ít bước nhất để được xâu theo thứ tự tăng dần.

**Ví dụ:**

Dữ liệu vào	Kết quả ra
$S = 'CBABA'$	Cần ít nhất 2 phép biến đổi $CBABA \rightarrow ABABC \rightarrow AABBC$

- 4.24. Cho  $N$  ( $N \leq 1000$ ) đoạn số nguyên  $[a_i, b_i]$ , hãy chọn một tập gồm ít số nhất mà mỗi đoạn số nguyên trên đều có ít nhất hai số thuộc tập, trong đó:  $(|a_i|, |b_i| \leq 10^9)$ .

Ví dụ, có năm đoạn  $[0, 10], [2, 3], [4, 7], [3, 5], [5, 8]$ , ta chọn tập gồm bốn số  $\{2, 3, 5, 7\}$ .

- 4.25. Cho phân số  $M/N$  ( $0 < M < N, M, N$  nguyên). Hãy phân tích phân số này thành tổng các phân số có tử số bằng 1, càng ít số hạng càng tốt.

**Input:** Tệp văn bản PS.INP chứa hai số  $M, N$ .

**Output:** Tệp văn bản PS.OUT:

- Dòng đầu là số lượng phân số tách;
- Các dòng sau, mỗi dòng chứa mẫu số của các số hạng.

**Ví dụ:**

Dữ liệu vào	Kết quả ra
5 6	2
	2 3

- 4.26. Cho một số tự nhiên  $N$ . Hãy tìm cách phân tích số  $N$  thành các số nguyên dương  $p_1, p_2, \dots, p_k$  (với  $k > 1$ ) sao cho:

- $p_1, p_2, \dots, p_k$  đôi một khác nhau;
- $p_1 + p_2 + \dots + p_k = N$ ;
- $S = p_1 * p_2 * \dots * p_k$  đạt giá trị lớn nhất.

**Input:** Tệp văn bản PT.INP: Gồm nhiều bộ kiểm thử, mỗi dòng là một bộ kiểm thử chứa một số  $N$  ( $5 \leq N \leq 1000$ ).

**Output:** Tệp văn bản PT.OUT: Gồm nhiều dòng, mỗi dòng là tích lớn nhất đạt được (số  $S$ ) cho bộ kiểm thử đó:

**Ví dụ:**

Dữ liệu vào	Kết quả ra
5	6
7	12

- 4.27. Cho hai phép toán  $*2$  (nhân với 2) và  $/3$  (chia nguyên cho 3). Cho trước số 1, bằng cách sử dụng hai phép toán trên. Hãy xây dựng được biểu thức có giá trị bằng  $N$ .

Ví dụ,  $N = 6$  thì  $1 * 2 * 2 * 2 * 2 / 3 / 3 * 2 = 6$  (thực hiện từ trái qua phải).

**Input:** Tệp văn bản BT.INP chứa số  $N$  ( $N$  có không quá 100 chữ số).

**Output:** Tệp văn bản BT.OUT chứa biểu thức ngắn nhất có thể.

- 4.28. Cho số nguyên dương  $N$  ( $N \leq 10^{100}$ ). Hãy tách  $N$  thành tổng ít các số Fibonacci nhất.

**Ví dụ,**  $N=16 = 1 + 5 + 13$ .

- 4.29. Cần phải tổ chức việc thực hiện  $N$  chương trình đánh số từ 1 đến  $N$  trên một máy tính. Mỗi chương trình thứ  $i$  đòi hỏi thời gian tính là 1 giờ và nếu nó được hoàn thành trước thời điểm  $d[i]$  (giả sử thời điểm bắt đầu thực hiện các chương trình là 0) thì người chủ máy tính sẽ được trả tiền công là  $w[i]$  ( $i = 1, 2, \dots, N$ ). Việc thực hiện mỗi chương trình phải được tiến hành liên tục từ lúc bắt đầu cho đến khi kết thúc không cho phép ngắt quãng, đồng thời tại mỗi thời điểm máy chỉ có thể thực hiện một chương trình).

Hãy tìm trình tự thực hiện các chương trình sao cho tổng tiền công nhận được là lớn nhất.

**Input:** Tệp văn bản JOB.INP:

- Dòng đầu tiên chứa số  $N$  ( $N \leq 5000$ ),
- Dòng thứ  $i$  trong  $N$  dòng tiếp theo chứa 2 số  $d[i]$ ,  $w[i]$  được ghi cách nhau ít nhất một dấu cách.

**Output:** Tệp văn bản JOB.OUT:

- Dòng đầu tiên chứa tổng tiền công nhận được theo trình tự tìm được.
- Dòng tiếp theo ghi trình tự thực hiện các chương trình.

ột bộ

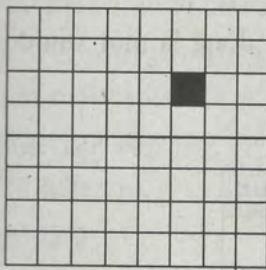
4.30. Tim  $K$  chữ số cuối cùng của  $M^N$  ( $0 < K \leq 9$ ,  $0 \leq M, N \leq 10^9$ ).

Ví dụ,  $K = 2$ ,  $M = 2$ ,  $N = 10$ , ta có  $2^{10} = 1024$ , như vậy hai chữ số cuối cùng nhất của  $2^{10}$  là 24.

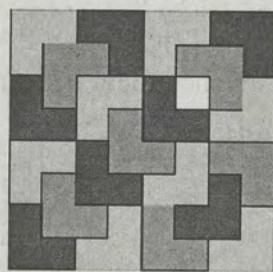
4.31. Viết hàm kiểm tra tính nguyên tố của số  $N$  ( $N \leq 10^9$ ) theo Fermat.

4.32. Lát gạch

Cho một nền nhà hình vuông có kích thước  $2^k$  bị khuyết một ô. Hãy tìm cách lát nền nhà bằng loại gạch hình thoi (tạo bởi 3 hình vuông đơn vị).



nền nhà (ô màu đen là ô khuyết)



một cách lát nền

4.33. Chỗ dãy  $a_1, a_2, \dots, a_n$ , các số đôi một khác nhau và số nguyên dương  $k$  ( $1 \leq k \leq n$ ). Hãy đưa ra giá trị nhỏ thứ  $k$  trong dãy.

Ví dụ, dãy gồm 5 phần tử: 5, 7, 1, 3, 4 và  $k = 3$  thì giá trị nhỏ thứ  $k$  là 4.

4.34. Dãy con lòi

Dãy số nguyên  $A_1, A_2, \dots, A_N$  được gọi là lòi, nếu nó giảm dần từ  $A_1$  đến một  $A_i$  nào đó, rồi tăng dần tới  $A_N$ .

Ví dụ dãy lòi: 10 5 4 2 -1 4 6 8 12

**Yêu cầu:** Cho một dãy số nguyên, bằng cách xoá bớt một số phần tử của dãy và giữ nguyên trình tự các phần tử còn lại, ta nhận được dãy con lòi dài nhất.

**Input:** Tệp văn bản DS.INP:

- Dòng đầu là  $N$  ( $N \leq 10000$ );
- Các dòng sau là  $N$  số nguyên của dãy số (các số kiểu longint).

**Output:** Tệp văn bản DS.OUT:

- Ghi số phần tử của dãy con tìm được;
- Các dòng tiếp theo ghi các số thuộc dãy con.

### 4.35. Palindrome

Một xâu được gọi là xâu đối xứng nếu đọc từ trái qua phải cũng giống như đọc từ phải qua trái. Ví dụ xâu “madam” là một xâu đối xứng. Bài toán đ<sup>ă</sup>ra là cho một xâu  $S$  gồm các kí tự thuộc tập  $['a'..'z']$ , hãy tìm cách chèn vào xâu  $S$  ít các kí tự nhất để xâu  $S$  thành xâu đối xứng.

Ví dụ, với xâu “adbhbca” ta s<sup>e</sup> chèn thêm hai kí tự ( $c$  và  $d$ ) để được xâu đối xứng “adcbhbcda”.

**Input:** Tệp văn bản PALIN.INP: Gồm một dòng chứa xâu  $S$  (độ dài xâu không vượt quá 200).

**Output:** Tệp văn bản PALIN.OUT: Gồm một dòng là một xâu đối xứng sa<sup>t</sup> khi đã chèn thêm ít kí tự nhất vào xâu  $S$ .

**Ví dụ:**

Palin.inp	Palin.out
acbcd	ad <u>c</u> bc <u>d</u> a

### 4.36. Stones

Có  $N$  đống sỏi xếp thành một hàng, đống thứ  $i$  có  $A_i$  viên sỏi. Ta có thể ghép hai đống sỏi kề nhau thành một đống và mất một chi phí bằng tổng hai đống sỏi đó.

**Yêu cầu:** Hãy tìm cách ghép  $N$  đống sỏi này thành một đống với chi phí là nhỏ nhất.

Ví dụ, có 5 đống sỏi

$$\begin{array}{ccccccc} 4 & & \underline{1} & \underline{2} & 7 & 5 \\ & \underline{4} & & 3 & & 7 & 5 \\ & & 7 & & \underline{7} & \underline{5} & \\ & & & 7 & & 12 & \\ & & & & 19 & & \end{array}$$

$$\text{Chi phí} = 3 + 7 + 12 + 19 = 41$$

**Input:** Tệp văn bản STONES.INP:

- Dòng đầu là số  $N$  ( $N < 101$ ) là số đống sỏi;
- Dòng thứ hai gồm  $N$  số nguyên là số sỏi của  $N$  đống sỏi ( $0 < A_i < 1001$ ).

**Output:** Tệp văn bản STONES.OUT: gồm một số là chi phí nhỏ nhất để ghép  $N$  đống thành một đống.

**Ví dụ:**

STONES.INP	STONES.OUT
5	41
4 1 2 7 5	

#### 4.37. Cắt hình 1

Có một hình chữ nhật  $M \times N$  ô, mỗi lần ta được phép cắt một hình chữ nhật thành hai hình chữ nhật con theo chiều ngang hoặc chiều dọc và lại tiếp tục cắt các hình chữ nhật con cho đến khi được hình vuông thì dừng.

Hỏi có thể cắt hình chữ nhật  $M \times N$  thành ít nhất bao nhiêu hình vuông.

**Input:** Tệp văn bản HCN.INP: Gồm một số dòng, mỗi dòng là một bộ kiểm thử ghi một cặp số  $M, N$  ( $1 \leq M, N \leq 100$ ).

**Output:** Tệp văn bản HCN.OUT: Gồm một số dòng ghi kết quả tương ứng với dữ liệu vào.

#### 4.38. Cắt hình 2

Cho một bảng số  $A$  gồm  $M$  dòng,  $N$  cột, các giá trị của bảng  $A$  chỉ là 0 hoặc 1. Ta muốn cắt bảng  $A$  thành các hình chữ nhật con sao cho các hình chữ nhật con có các giá trị toàn bằng 1 hay toàn bằng 0. Một lần cắt là một nhát cắt thẳng theo dòng hoặc theo cột của một hình chữ nhật thành hai hình chữ nhật riêng biệt. Cứ tiếp tục cắt cho đến khi hình chữ nhật có các giá trị toàn bằng 1 hay toàn bằng 0. Hãy tìm cách cắt để số hình chữ nhật con nhận được, có giá trị toàn bằng 1 hay toàn bằng 0, là nhỏ nhất.

**Ví dụ:** bảng số  $5 \times 5$  sau được chia thành 8 hình chữ nhật con.

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

**Input:** Tệp văn bản HCN2.INP:

- Dòng đầu là 2 số nguyên dương  $M, N$  ( $M, N \leq 30$ ).

- $M$  dòng tiếp theo, mỗi dòng  $N$  số chỉ gồm 0 hoặc 1 thể hiện bảng số  $A$ .

**Output:** Tệp văn bản HCN2.OUT: Gồm một dòng duy nhất chứa một số duy nhất là số hình chữ nhật ít nhất.

### 4.39. TKSEQ

Cho dãy số  $A$  gồm  $N$  số nguyên và số nguyên  $K$ . Tìm dãy chỉ số  $1 \leq i_1 < i_2 < \dots < i_{3K} \leq N$  sao cho:

$$S = (a_{i1} - a_{i2} + a_{i3}) + (a_{i4} - a_{i5} + a_{i6}) + \dots + (a_{i3k-2} - a_{i3k-1} + a_{i3k})$$

đạt giá trị lớn nhất.

4.4

**Input:** Tệp văn bản TKSEQ.INP:

- Dòng đầu là gồm hai số nguyên  $N, K$  ( $0 < 3K \leq N \leq 500$ );
- Dòng thứ hai gồm  $N$  số nguyên  $a_1, a_2, \dots, a_N$  ( $|a_i| < 10^9$ ).

**Output:** Tệp văn bản TKSEQ.OUT gồm một số duy nhất  $S$  lớn nhất tìm được.

**Ví dụ:**

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

### 4.40. Least-Squares Segmentation

Ta định nghĩa trọng số của đoạn số từ số ở vị trí thứ  $i$  đến vị trí thứ  $j$  của dãy số nguyên  $A[1], A[2], \dots, A[N]$  là:

$$\sum_{k=i}^j (A[k] - mean)^2 \text{ trong đó } mean = \frac{\sum_{k=1}^N A[k]}{j-i+1}$$

**Yêu cầu:** Cho dãy số nguyên  $A$  gồm  $N$  số  $A[1], A[2], \dots, A[N]$  và số nguyên dương  $G$  ( $1 < G^2 < N$ ). Hãy chia dãy  $A$  thành đúng  $G$  đoạn để tổng trọng số là nhỏ nhất.

4.41

**Input:** Tệp văn bản LSS.INP:

- Dòng đầu gồm hai số  $N$  và  $G$  ( $1 < G^2 < N < 1001$ );
- $N$  dòng tiếp theo, mỗi dòng một số nguyên mô tả dãy số  $A$  ( $0 < A[i] < 10^6$ ).

**Output:** Tệp văn bản LSS.OUT: gồm một dòng chứa một số thực duy nhất là đáp án của bài toán (đưa ra theo quy cách :0:2).

#### Ví dụ:

LSS. INP	LSS. OUT
5 2	0.50
3	
3	
3	
4	
5	

#### 4.41. Phân trang (Đề thi chọn đội tuyển quốc gia, 1999)

Văn bản là một dãy gồm  $N$  từ được đánh số từ 1 đến  $N$ . Từ thứ  $i$  có độ dài là  $w_i$  ( $i = 1, 2, \dots, N$ ). Phân trang là một cách xếp lần lượt các từ của văn bản vào dãy các dòng, mỗi dòng có độ dài  $L$ , sao cho tổng độ dài của các từ trên cùng một dòng không vượt quá  $L$ . Ta gọi hệ số phạt của mỗi dòng trong cách phân trang là hiệu số  $(L - S)$ , trong đó  $S$  là tổng độ dài của các từ xếp trên dòng đó. Hệ số phạt của cách phân trang là giá trị lớn nhất trong số các hệ số phạt của các dòng.

**Yêu cầu:** Tìm cách phân trang với hệ số phạt nhỏ nhất.

**Input:** Tệp văn bản PTRANG.INP:

- Dòng 1 chứa hai số nguyên dương  $N, L$  ( $N \leq 4000, L \leq 70$ )
- Dòng thứ  $i$  trong số  $N$  dòng tiếp theo chứa số nguyên dương  $w_i$  ( $w_i \leq L$ ),  $i = 1, 2, \dots, N$ .

**Output:** Tệp văn bản PTRANG.OUT:

- Dòng đầu ghi hai số  $P, Q$  theo thứ tự là hệ số phạt và số dòng theo cách phân trang tìm được.
- Dòng thứ  $i$  trong số  $Q$  dòng tiếp theo ghi chỉ số của các từ trong dòng thứ  $i$  của cách phân trang.

#### 4.42. Chọn số

Cho mảng  $A$  có kích thước  $N \times N$ , gồm các số nguyên không âm. Hãy chọn ra  $K$  số sao cho mỗi dòng có nhiều nhất 1 số được chọn, mỗi cột có nhiều nhất 1 số được chọn để tổng  $K$  số là lớn nhất.

**Input:** Tệp văn bản SELECT.INP:

- Dòng thứ nhất gồm 2 số  $N$  và  $K$  ( $K \leq N \leq 15$ );
- $N$  dòng sau, mỗi dòng là  $N$  số nguyên không âm  $A_{ij} < 10000$  ( $1 \leq i, j \leq N$ ).

**Output:** Tệp văn bản SELECT.OUT: Tổng lớn nhất chọn được và số cách chọn (cách nhau đúng một dấu cách).

Ví dụ:

Select.inp	Select.out
3 2	6 3
1 2 3	
2 3 1	
3 1 2	

#### 4.43. Puzzle of numbers

Khi một số các chữ số trong phép toán đúng của hai số nguyên bị mất (được thay bởi các dấu sao “\*”). Có một câu đố là: Hãy thay các dấu sao bởi các chữ số để được phép toán đúng.

Ví dụ, bắt đầu phép cộng sau:

9334

789

-----

10123       $(9334 + 789 = 10123)$

Các chữ số bị mất được thay bằng các dấu sao như sau:

\*3\*4 hay \*\*\*\*

78\*           \*\*\*

10123        \*\*\*\*\*

Nhiệm vụ của bạn là viết chương trình thay các dấu sao thành các chữ số để được một phép toán đúng. Nếu có nhiều lời giải thì đưa ra một trong số đó. Nếu không có thì đưa ra thông báo: “No Solution”.

Chú ý các chữ số ở đầu mỗi số phải khác 0.

**Input:** Tệp văn bản REBUSS.INP: gồm ba dòng, mỗi dòng là một xâu kí tự gồm các chữ số hoặc kí tự “\*”. Độ dài mỗi xâu không quá 50 kí tự.

Dòng thứ nhất, dòng thứ hai biểu diễn hai số hạng, dòng thứ ba biểu diễn tổng hai số.

**Output:** Tệp văn bản REBUSS.OUT: Nếu có lời giải thì tệp kết quả gồm ba dòng tương ứng với tệp dữ liệu vào, nếu không thì thông báo “No Solution”.

cách

### Ví dụ:

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

#### 4.44. Xếp lịch giảng

Một giáo viên cần giảng  $n$  vấn đề được đánh số từ 1 đến  $n$  ( $n \leq 10000$ ). Mỗi một vấn đề  $i$  cần có thời gian là  $t_i$  ( $i = 1..n$ ). Để giảng  $n$  vấn đề đó thì giáo viên có các buổi đã được phân có độ dài là  $L$  ( $L \leq 500$ ).

- Một vấn đề thì phải giải quyết trong một buổi.
- Vấn đề  $i$  phải được giảng trước vấn  $i + 1$  với mọi  $i = 1..(n - 1)$ .

Học sinh có thể ra về sớm nếu như buổi giảng đã kết thúc, tuy nhiên nếu thời gian ra về đó quá sớm so với buổi giảng thì thật là phí. Chính vì thế người ta đánh giá buổi lên lớp bằng giá trị  $DI$  như sau :

$$DI = \begin{cases} 0 & \text{nếu } t = 0 \\ -C & \text{nếu } 1 \leq t \leq 10 \\ (t-10)^2 & \text{nếu } t > 10 \end{cases}$$

trong đó,  $t$  là thời gian thừa của buổi lên lớp đó,  $C$  là một hằng số.

**Yêu cầu:** Hãy xếp lịch giảng sao cho tổng số các buổi là ít nhất có thể được. Trong các lịch giảng đó, hãy tìm lịch giảng sao cho tổng số  $DI$  là nhỏ nhất.

**Input:** Tệp văn bản SCHEDULING.INP:

- Dòng đầu là số  $n$  (số vấn đề cần giảng);
- Dòng tiếp theo là  $L$  và  $C$ ;
- Dòng cuối cùng là  $N$  số thể hiện cho  $t_1, t_2, \dots, t_n$ .

**Output:** Tệp văn bản SCHEDULING.OUT:

- Dòng đầu tiên là số buổi.
- Dòng tiếp theo là tổng  $DI$  nhỏ nhất đạt được.

**Ví dụ:**

SCHEDULING . INP	SCHEDULING . OUT
10	6
120 10	2700
80 80 10 50 30 20 40 30 120 100	

**4.45. Khu vườn (IOI 2008)**

Ramsesses II thắng trận trở về. Để ghi nhận chiến tích của mình ông quyết định xây một khu vườn tráng lệ. Khu vườn phải có một hàng cây chạy dài từ cung điện của ông tại Luxor tới thánh đường Karnak. Hàng cây này chỉ chứa hai loại cây là sen và cói giấy, bởi vì chúng tương ứng là biểu tượng của miền Thượng Ai Cập và Hạ Ai Cập.

Vườn phải có đúng  $N$  cây. Ngoài ra, phải có sự cân bằng, ở mọi đoạn cây liên tiếp của vườn, số lượng sen và số lượng cói giấy phải không lệch nhau quá 2.

Vườn cây được biểu diễn dưới dạng xâu các kí tự 'L' (lotus – sen) và 'P' (papyrus – cói giấy).

Ví dụ, với  $N = 5$  có tất cả 14 vườn đảm bảo cân bằng. Theo thứ tự từ điển, các vườn đó là: *LLPLP, LLPPL, PLLLP, LPLPL, LPLPP, LPPLL, LPPLP, PLLPL, PLLPP, PLPLL, PLPPL, PPPLP* và *PPLPL*.

Các vườn cân bằng với độ dài xác định cho trước được sắp xếp theo thứ tự từ điển và được đánh số từ 1 trở đi. Ví dụ, với  $N = 5$ , vườn số 12 sẽ là vườn *PLPPL*.

**Yêu cầu:** Cho số cây  $N$  và xâu biểu diễn một vườn cân bằng, hãy lập trình tính số thứ tự của vườn này theo môđun  $M$ , trong đó  $M$  là số nguyên cho trước.

Lưu ý rằng giá trị của  $M$  không đóng vai trò quan trọng trong việc giải bài toán, nó chỉ làm cho việc tính toán trở nên đơn giản.

*Hạn chế:*

$$1 \leq N \leq 1\ 000\ 000;$$

$$7 \leq M \leq 10\ 000\ 000.$$

**Chấm điểm:** Có 40 điểm dành cho các dữ liệu vào với  $N$  không vượt quá 40.

**Input:** Tệp văn bản GARDEN.INP:

- Dòng 1 chứa số nguyên  $N$ , số cây trong vườn,
- Dòng 2 chứa số nguyên  $M$ ,
- Dòng 3 chứa xâu gồm  $N$  kí tự 'L' (sen) hoặc 'P' (cói giấy) biểu diễn vườn cân bằng.

**Output:** Tệp văn bản GARDEN.OUT có một dòng chứa một số nguyên trong phạm vi từ 0 đến  $M - 1$ , là số thứ tự theo môđun  $M$  của vườn được mô tả trong đầu vào.

**Ví dụ:**

Input	Output	Giải thích
5	5	Số thứ tự của PLPPL là 12.
7		Như vậy output là 12 theo môđun 7, tức là 5.
PLPPL		

Input	Output
12	39
10000	
LPLLPLPPLPLL	

#### 4.46. Số rõ ràng

Bờm mới tìm được một tài liệu định nghĩa số rõ ràng như sau: Với số nguyên dương  $n$ , ta tạo số mới bằng cách lấy tổng bình phương các chữ số của nó, với số mới này ta lại lặp lại công việc trên. Nếu trong quá trình đó, ta nhận được số mới là 1, thì số  $n$  ban đầu được gọi là số rõ ràng. Ví dụ, với  $n = 19$ , ta có:

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

Như vậy, 19 là số rõ ràng.

Không phải mọi số đều rõ ràng. Ví dụ, với  $n = 12$ , ta có:

$$\begin{aligned} 12 &\rightarrow 5 \rightarrow 25 \rightarrow 29 \rightarrow 85 \rightarrow 89 \rightarrow 145 \rightarrow 42 \rightarrow 20 \rightarrow 4 \rightarrow 16 \rightarrow 37 \rightarrow 58 \\ &\rightarrow 89 \rightarrow 145 \end{aligned}$$

Bờm rất thích thú với định nghĩa số rõ ràng này và thách đố phú ông:

Cho một số nguyên dương  $n$ , tìm số  $S(n)$  là số rõ ràng liền sau số  $n$ , tức là  $S(n)$  là số rõ ràng nhỏ nhất lớn hơn  $n$ .

Tuy nhiên, câu hỏi đó quá dễ với phú ông và phú ông đã đố lại Bờm:  
 Cho hai số nguyên dương  $n$  và  $k$  ( $1 \leq n, k \leq 10^{15}$ ), hãy tìm số  
 $S^k(n) = S(S(\dots(S(n))))$  là số rõ ràng liền sau thứ  $k$  của  $n$ .

Bạn hãy giúp Bờm giải câu đố này nhé!

**Input:** Tệp văn bản CLEAR.INP :

- Dòng đầu là số  $t$  ( $0 < t \leq 20$ );
- $t$  dòng sau, mỗi dòng chứa 2 số nguyên  $n$  và  $k$ .

**Output:** Tệp văn bản CLEAR.OUT gồm  $t$  dòng, mỗi dòng là kết quả tương ứng với dữ liệu vào.

**Ví dụ:**

CLEAR.INP	CLEAR.OUT
2	19
18 1	1000000000
1 145674807	

#### 4.47. Háí nấm

Bé Bông đi hái nấm trong  $N$  khu rừng đánh số từ 1 đến  $N$ , nhưng chỉ có  $M$  khu rừng có nấm. Việc di chuyển từ khu rừng thứ  $i$  sang khu rừng thứ  $j$  tốn  $t_{ij}$  đơn vị thời gian. Đến khu rừng  $i$  có nấm, cô bé có thể dừng lại để hái nấm. Nếu tổng số đơn vị thời gian cô bé dừng lại ở khu rừng thứ  $i$  là  $d_i$  ( $d_i > 0$ ), thì cô bé hái được:  $\left[ \frac{S_i}{2} \right] + \left[ \frac{S_i}{4} \right] + \dots + \left[ \frac{S_i}{2^{d_i}} \right]$  cây nấm tại khu rừng đó (trong đó  $S_i$  là số lượng nấm có tại khu rừng  $i$ ,  $[x]$  là phần nguyên của  $x$ ). Giả thiết rằng ban đầu cô bé ở khu rừng thứ nhất và đi hái nấm trong thời gian không quá  $P$  đơn vị.

**Yêu cầu:** Hãy tính số lượng cây nấm nhiều nhất mà cô bé có thể hái được.

**Input:** Tệp văn bản MUSHROOM.INP :

- Dòng đầu tiên chứa ba số nguyên dương  $M$  ( $M \leq 10$ ),  $N$  ( $M \leq N \leq 100$ ) và  $P$  ( $P \leq 10000$ );
- $M$  dòng tiếp theo, mỗi dòng chứa hai số nguyên dương  $r$  và  $S_r$  nghĩa là khu rừng  $r$  có  $S_r$  nấm ( $S_r \leq 10^9$ );
- Dòng thứ  $i$  trong  $N$  dòng cuối cùng chứa  $N$  số nguyên dương  $t_{ij}$  ( $t_{ij} \leq 10000$ ), ( $i, j = 1, \dots, N$ ).

**Output:** Tệp văn bản MUSHROOM.OUT: số lượng cây nấm nhiều nhất bé Bông có thể hái được.

**Ví dụ:**

MUSHROOM.INP	MUSHROOM.OUT
2 2 2	3
1 5	
2 10	
0 3	
3 0	

### Bài tập bổ sung

#### 4.48. Mật khẩu

Một xâu kí tự được gọi là mật khẩu “an toàn” nếu xâu có độ dài ít nhất bằng 6 và xâu chứa ít nhất một chữ cái in hoa ('A'..'Z'), một chữ cái thường ('a','z'), một chữ số ('0'.. '9').

Ví dụ, 'a1B2C3', 'tinHoc6' là hai mật khẩu “an toàn”, còn 'a1B2C', 'alb2c3', 'A1B2C3', 'tinHoc' đều không là mật khẩu “an toàn”.

Một lần, Bình nhìn thấy một xâu  $S$ , chỉ gồm các loại kí tự: chữ cái in hoa, chữ cái thường và chữ số. Bình muốn tự kiểm tra khả năng đoán nhận mật khẩu bằng cách đếm xem có bao nhiêu cặp chỉ số  $(i, j)$  thoả mãn điều kiện:  $1 \leq i < j \leq \text{length}(S)$  và xâu con gồm các kí tự liên tiếp từ  $i$  đến  $j$  của  $S$  là mật khẩu “an toàn”.

**Yêu cầu:** Cho xâu  $S$ , tính số lượng cặp chỉ số  $(i, j)$  thoả mãn điều kiện nêu trên.

**Input:** Tệp văn bản MATKHAU.INP gồm một dòng chứa xâu  $S$ .

**Output:** Tệp văn bản MATKHAU.OUT một số nguyên là số lượng cặp chỉ số  $(i, j)$  tính được.

**Ví dụ:**

MATKHAU.INP	MATKHAU.OUT
abc3456789PQ	6

MATKHAU.INP	MATKHAU.OUT
abc123	0

*Subtask 1: Xâu  $S$  có độ dài không quá 300.*

*Subtask 2:* Xâu  $S$  có độ dài không quá 3000.

*Subtask 3:* Xâu  $S$  có độ dài không quá  $10^6$ .

#### 4.49. Dây dẫn

Cho  $n$  đoạn dây điện, đoạn dây thứ  $i$  có độ dài  $l_i$  cm. Cần phải cắt các đoạn đã cho thành các đoạn sao cho có được  $k$  đoạn dây bằng nhau có độ dài nguyên. Có thể không cần cắt hết các đoạn dây đã cho. Mỗi đoạn dây bị cắt có thể có phần còn thừa khác 0.

**Yêu cầu:** Xác định độ dài lớn nhất của đoạn dây có thể nhận được. Nếu không có cách cắt thì đưa ra số 0.

**Input:** Tệp văn bản WIRES.INP :

- Dòng đầu tiên chứa hai số nguyên  $n, k$ .
- Dòng thứ  $i$  trong  $n$  dòng sau chứa số nguyên  $l_i$ .

**Output:** Tệp văn bản WIRES.OUT, ghi kết quả trên một dòng dưới dạng số nguyên.

**Ví dụ:**

WIRES.INP	WIRES.OUT
4 11	200
802	
743	
547	
539	

#### 4.50. Lucky Numbers (Olympic sinh viên)

Trong một số nước châu Á, các số 8 và 6 được coi là những chữ số may mắn. Bất cứ số nguyên nào chỉ chứa chữ số 8 và 6 được coi là số may mắn, ví dụ 6, 8, 66, 668, 88, 886, ...

Nguyên là một học sinh rất thích toán. Nguyên thích các số may mắn nhưng chỉ thích các số có dạng  $S = 8..86..6$ , trong đó  $S$  có ít nhất một chữ số 8 và một chữ số 6 và không nhất thiết phải đồng thời xuất hiện. Ví dụ, 8, 88, 6, 66, 86, 886, 8866, ... là các số có dạng  $S$ .

Cho trước một số nguyên dương  $X$  ( $1 < X < 10\ 000$ ), Nguyên muốn tìm số may mắn nhỏ nhất dạng  $S$ , có không quá 200 chữ số và chia hết cho  $X$ .

Nhiệm vụ của bạn là viết một chương trình tìm số đó cho Nguyên.

**Input:** Tệp văn bản NUM86.INP :

Dòng đầu tiên chứa một số nguyên dương không lớn hơn 20 là số lượng các bộ dữ liệu.

Trên mỗi dòng tiếp theo chứa một số nguyên  $X$  tương ứng với một bộ dữ liệu.

**Output:** Tệp văn bản NUM86.OUT, ghi ra trên một dòng số may mắn dạng  $S$  nhỏ nhất chia hết cho  $X$  với mỗi bộ dữ liệu. Trường hợp không tồn tại số  $S$  có không quá 200 chữ số như vậy, ghi -1.

**Ví dụ:**

NUM86.inp	NUM86.out
4	6
6	8
8	86
43	-1
5	

#### 4.51. ACM

Vào hè, trường Đại học Công nghệ dự định tổ chức kì thi theo mô hình ACM cho các trường phổ thông. Mỗi trường sẽ chọn ra một đội gồm ba thí sinh để thi đấu. Để chuẩn bị tốt cho kì thi, trường  $XYZ$  đã có kế hoạch tập huấn cho  $n$  học sinh với 11 chủ đề sau:

- 1) Lí thuyết độ phức tạp tính toán;
- 2) Tổ hợp và số học;
- 3) Sắp xếp, tìm kiếm nâng cao;
- 4) Xử lí xâu;
- 5) Quy hoạch động;
- 6) Duyệt toàn bộ và nhánh cận;
- 7) Các thuật toán đồ thị;

- 8) Các thuật toán xấp xỉ;
- 9) Các thuật toán hình học;
- 10) Lý thuyết trò chơi;
- 11) Một số cấu trúc dữ liệu nâng cao.

Kết thúc khoá tập huấn, Ban giám hiệu đã thống kê khả năng của từng học sinh và muốn chọn ra ba học sinh để lập thành đội thi với hi vọng đạt kết quả cao nhất. Giả sử  $s_{ij}$  là điểm đánh giá khả năng của học sinh  $i$  với chủ đề  $j$  thì việc đánh giá khả năng đạt kết quả cao của đội gồm ba thí sinh  $x, y, z$  bằng  $\sum_{j=1}^{11} \text{Max}(s_{xj}, s_{yj}, s_{zj})$ .

**Yêu cầu:** Cho  $n$  học sinh và  $s_{ij}$  là khả năng của học sinh  $i$  với chủ đề  $j$ . Hãy giúp Ban giám hiệu nhà trường chọn ra ba học sinh thành một đội thi đấu có khả năng đạt kết quả cao nhất.

**Input:** Tệp văn bản ACM.INP, trong đó:

- Dòng đầu tiên chứa số nguyên  $n$  ( $n \leq 10^4$ ).
- $n$  dòng tiếp theo, mỗi dòng chứa 11 số nguyên không âm  $s_{ij}$  ( $s_{ij} \leq 10^9$ ).

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

**Output:** Tệp văn bản ACM.OUT ghi khả năng đạt kết quả cao nhất của đội có ba thí sinh được chọn.

**Ví dụ:**

ACM. INP	ACM. OUT
4	9
2 2 2 0 0 0 0 0 0 0 0	
3 1 1 0 0 0 0 0 0 0 0	
1 3 1 0 0 0 0 0 0 0 0	
1 1 3 0 0 0 0 0 0 0 0	

## 4.52. Phần thưởng

Tuấn là người thắng cuộc trong một cuộc thi “Tìm hiểu kiến thức vũ trụ” và được nhận các phần thưởng do công ty XYZ tài trợ. Các phần thưởng được bố trí trên một bảng vuông kích thước  $n \times n$  có dạng một lưới ô vuông kích thước đơn vị. Các dòng của bảng được đánh số từ 1 đến  $n$ , từ trên xuống dưới và các cột của bảng được đánh số từ 1 đến  $n$ , từ trái qua phải. Ô nằm trên giao của dòng  $i$  và cột  $j$  được gọi là ô  $(i, j)$  và trên ô đó chứa một món quà có giá trị là  $a_{ij}$  ( $1 \leq i, j \leq n$ ).

Để nhận phần thưởng, Tuấn được phép chọn đúng hai hình vuông không giao nhau (có thể tiếp xúc) kích thước  $k \times k$  chiếm trọn một số ô của bảng và nhận tất cả các phần quà trong các ô nằm trong hai hình vuông đó.

**Yêu cầu:** Hãy xác định tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

**Input:** Tệp văn bản BONUS2.INP :

- Dòng thứ nhất chứa hai số nguyên dương  $n, k$  ( $n \leq 1000; \frac{n}{3} \leq k \leq \frac{n}{2}$ ).
- Dòng thứ  $i$  trong số  $n$  dòng tiếp theo chứa  $n$  số nguyên không âm,  $a_{ij}$  ( $a_{ij} \leq 1000$ ).

Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

**Output:** Tệp văn bản BONUS2.OUT một số nguyên duy nhất là tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

**Ví dụ:**

BONUS2.INP	BONUS2.OUT
4 2	68
9 9 1 1	
9 9 1 1	
1 8 8 1	
1 8 8 1	

### 4.53. Bốc sỏi

Có  $n$  đồng sỏi xếp thành một vòng tròn, đồng thứ  $i$  có  $a_i$  viên sỏi. Ta có hai cách bốc sỏi như sau:

- Chọn hai đồng liền nhau có số lượng sỏi đều là lẻ. Sau đó bốc từ hai đồng đó, mỗi đồng một viên sỏi;
- Chọn hai đồng liền nhau vẫn còn sỏi. Sau đó lần lượt bốc từ hai đồng đó, để sau khi bốc số sỏi còn lại của mỗi đồng bằng số sỏi ban đầu của đồng đó div 2.

**Yêu cầu:** Cho số lượng sỏi ban đầu của mỗi đồng  $a_1, a_2, \dots, a_n$ , hãy tính số cách khác nhau để bốc được hết tất cả các sỏi.

**Input:** Tệp văn bản BOCSOI.INP

- Dòng đầu là số đồng sỏi  $n$  ( $3 \leq n \leq 5$ );
- Dòng thứ hai gồm  $n$  số mô tả số lượng sỏi của  $n$  đồng.

**Output:** Tệp văn bản BOCSOI.OUT, gồm một dòng là số lượng cách khác nhau mod 111539786.

**Ví dụ:**

BOCSOI.INP	BOCSOI.OUT
3 1 2 3	2
3 1 1 1	0

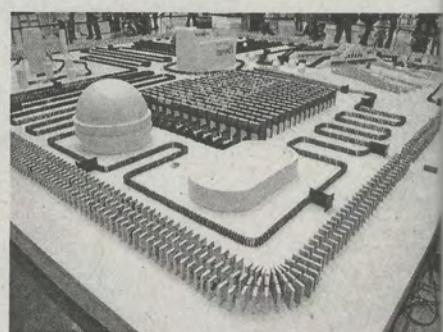
*Subtask 1:  $n = 3$  và  $a_i \leq 30$ .*

*Subtask 2:  $a_i \leq 30$ .*

*Subtask 3: Dự đoán giới hạn của  $a_i$  và giải quyết.*

### 4.54. Kỉ lục đố domino

Một kỉ lục thế giới về xếp domino đố đã được ghi nhận vào ngày 17/11/2006. Kỉ lục này thuộc về Hà Lan khi 4 079 381 quân domino đã lần lượt đố xuống theo phản ứng dây chuyền trong tiếng vỗ tay reo hò của các cổ động viên. Những người tổ chức sự kiện ngày Đomino ở



Hà Lan cho biết, 4 079 381 quân domino đã lần lượt đổ xuống trong vòng 2 giờ đồng hồ.

Những quân domino đã di động uyển chuyển trên nền những điệu nhạc cổ điển và đương đại là nét đặc biệt nhất của màn trình diễn domino. Tác giả Robin Paul Weijers nói: "Hơn 4 triệu quân domino, điều này chưa bao giờ xảy ra. Chúng tôi còn thành công trong việc khiến cho những quân bài domino nhảy múa trong tiếng nhạc. Tôi rất hạnh phúc vì đã thành công."

Với màn trình diễn tuyệt vời này, những kỉ lục gia domino Hà Lan đã phá vỡ kỉ lục của chính họ lập được năm 2005 với 4.002.136 quân bài domino.

Sắp tới, Bờm dự định xây dựng một công trình lớn hơn để phá kỉ lục của người Hà Lan. Công trình sẽ bao gồm hai công đoạn chính:

Công đoạn 1: Xếp  $M \times N - T$  quân domino vào các ô còn trống trên hình chữ nhật kích thước  $M \times N$  ( $M, N \leq 16$ ), trong hình chữ nhật đó có  $T$  ô đã được đặt trước  $T$  vật trang trí.

Công đoạn 2: Xếp  $R \times L$  quân domino thành một dãy độ dài  $L$  ( $L \leq 10^6$ ), mỗi hàng có đúng  $R$  ( $R \leq 8$ ) quân (có thể được hiểu như xếp vào hình chữ nhật kích thước  $R \times L$ ).

Điểm độc đáo trong công trình này là sự phối màu giữa các quân domino lân cận chung cạnh. Các quân domino được xếp bằng hai loại domino, loại 1 có màu xanh nhạt và loại 2 có màu xanh đậm. Quân domino ở vị trí ô  $(i, j)$  sẽ phải thoả mãn điều kiện: nếu  $i + j$  lẻ thì màu quân domino này sẽ phải có màu không nhạt hơn các quân ở các ô chung cạnh (nếu có), nếu  $i + j$  chẵn thì màu quân domino này sẽ phải có màu không đậm hơn các quân ở các ô chung cạnh (nếu có).

Để có những thông tin thú vị khi giới thiệu về công trình, Bờm muốn biết số lượng cách xếp khác nhau của công đoạn 1 và công đoạn 2. Hai cách xếp được gọi là khác nhau nếu khi chồng khít hai cách lên nhau (không xoay hoặc lật) có ít nhất một quân khác màu.

**Input:** Tệp văn bản DOMINO.INP:

- Dòng 1: gồm một số nguyên dương  $K$  ( $K \leq 10^9$ ), các kết quả tìm được sẽ mod cho  $K$ ;

- Dòng 2: bắt đầu là ba số nguyên dương  $M, N, T$  ( $M, N \leq 16; T < M \times N$ ), trong đó  $M, N$  là kích thước hình chữ nhật trong công đoạn 1,  $T$  là số lượng ô trong hình chữ nhật đã đặt vật trang trí, tiếp theo là  $T$  cặp số, cặp số  $i, j$  là tọa độ ô đã đặt vật trang trí;
- Dòng 3: gồm hai số nguyên dương  $R, L$  ( $R \leq 8; L \leq 10^6$ ) là kích thước hình của công đoạn 2.

**Output:** Tệp văn bản DOMINO.OUT:

- Dòng 1: số cách xếp công đoạn 1 khác nhau mod  $K$ ;
- Dòng 2: số cách xếp công đoạn 2 khác nhau mod  $K$ .

**Ví dụ:**

DOMINO.INP	DOMINO.OUT
1000	240
5 5 1 3 3	593
3 10000	

#### 4.55. VACCINE (ACM Việt Nam 2010)

Amino acid là thành phần cơ bản của sự sống. Amino acid gồm 20 loại khác nhau {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}. Protein là một chuỗi amino acid và được biểu diễn bởi một xâu kí tự  $P$ , trong đó  $P(t)$  là amino acid ở vị trí thứ  $t$  trong  $P$ . Các chuỗi amino acid khác nhau sẽ tạo ra các protein khác nhau. M-vaccine là một loại protein có độ dài không quá 500 amino acid được dùng để tiêu diệt các loại virus.

Các nhà khoa học đang tập trung nghiên cứu protein của virus POKTE nhằm sản xuất ra các loại M-vaccine để tiêu diệt loại virus này.

Ta nói, M-vaccine  $V$  có thể tiêu diệt được  $k$  amino acid trong protein  $P$  của virus POKTE nếu tồn tại hai dãy chỉ số  $(X_1 < X_2 < \dots < X_k)$  và  $(Y_1 < Y_2 < \dots < Y_k)$  sao cho  $V(X_i) = P(Y_i)$ , với  $i = 1 \dots k$ . Mức độ tiêu diệt virus POKTE của M-vaccine  $V$  được xác định bằng giá trị  $k$  lớn nhất.

Cho protein  $P$  của virus POKTE và  $N$  loại M-vaccine, nhiệm vụ của bạn là tìm loại M-vaccine trong  $N$  loại M-vaccine có mức độ tiêu diệt virus POKTE lớn nhất.

**Input:** Tệp văn bản VACCINE.INP:

Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều bộ kiểm thử. Dòng đầu tiên chứa một số nguyên dương không lớn hơn 20 là số lượng các bộ dữ liệu. Các dòng tiếp theo chứa các bộ dữ liệu.

Với mỗi bộ dữ liệu, dòng đầu tiên chứa số nguyên  $N$  ( $1 \leq N \leq 200$ ) là số lượng loại M-vaccine. Dòng thứ hai chứa một xâu độ dài không vượt quá 10000 là biểu diễn protein P của virus POKTE.  $N$  dòng tiếp theo, mỗi dòng chứa một xâu biểu diễn cho một loại M-vaccine.

**Output:** Tệp văn bản VACCINE.OUT:

Với mỗi bộ dữ liệu, ghi ra trên một dòng một số nguyên là mức độ tiêu diệt virus POKTE lớn nhất của loại M-vaccine.

**Ví dụ:**

VACCINE.INP	VACCINE.OUT
1	2
5 1	
ENRPPNVPES	
TEV	
LNRC	
HKVR	
FWW	
PWP	

#### 4.56. Cổ phiếu

Hiện nay thị trường chứng khoán Việt Nam là một kênh huy động vốn hiệu quả cho các doanh nghiệp cũng như là một kênh đầu tư tài chính tiềm năng cho các nhà đầu tư. Một nhà đầu tư muốn thực hiện việc mua và bán cổ phiếu của  $n$  công ty có niêm yết cổ phiếu trên thị trường trong giai đoạn gồm  $D$  ngày.

Sau nhiều ngày phân tích, nhà đầu tư rút ra được các dự báo sau:

- 1) Số lượng cổ phiếu của công ty  $i$  ( $1 \leq i \leq n$ ) trên thị trường là  $Q_i$ ;
- 2) Giá một cổ phiếu của công ty  $i$  trong ngày  $t$  là  $P_{ti}$  ( $0 < P_{ti} \leq 10^6$ );
- 3) Phí giao dịch mua bán một cổ phiếu là 1 đồng.

Luật đầu tư như sau: Tại một ngày  $t$  ( $1 \leq t \leq D$ ) nhà đầu tư có thể tiến hành hai bước:

*Bước 1 – Bán cổ phiếu:* Nếu có  $m_i$  cổ phiếu của công ty  $i$  trong tài khoản, nhà đầu tư có thể tiến hành bán  $s_i$  ( $0 \leq s_i \leq m_i$ ) cổ phiếu. Số tiền bán cổ phiếu trừ đi phí giao dịch ( $s_i \times P_{ti} - s_i$ ) sẽ được chuyển vào tài khoản của nhà đầu tư ngay lập tức. Số lượng cổ phiếu của công ty  $i$  trong tài khoản của nhà đầu tư còn lại là  $(m_i - s_i)$ . Với mỗi loại cổ phiếu, nhà đầu tư có thể bán hoặc không bán cổ phiếu.

*Bước 2 – Mua cổ phiếu :* Nếu muốn mua  $s_i$  cổ phiếu của công ty  $i$ , nhà đầu tư phải trả ngay lập tức tiền mua cổ phiếu và phí giao dịch (vì thế muốn thực hiện giao dịch này, số tiền trong tài khoản phải lớn hơn hoặc bằng số tiền phải trả). Số tiền trong tài khoản của nhà đầu tư ngay lập tức sẽ bị trừ đi một lượng là  $(s_i \times P_{ti} + s_i)$  đồng. Số lượng cổ phiếu mua ngay lập tức sẽ được chuyển vào tài khoản của nhà đầu tư. Chú ý: tổng số cổ phiếu của công ty trong tài khoản của nhà đầu tư không thể lớn hơn  $Q_i$ . Với mỗi loại cổ phiếu nhà đầu tư có thể mua hoặc không mua cổ phiếu.

**Yêu cầu:** Tại thời điểm đầu tiên, nhà đầu tư có  $k$  đồng trong tài khoản, và không có cổ phiếu của bất cứ công ty nào. Dựa vào dự báo nêu trên, hãy giúp nhà đầu tư tìm cách mua bán cổ phiếu từ ngày thứ nhất cho đến hết ngày thứ  $D$ , để tổng lượng tiền thu được là lớn nhất.

**Input:** Tệp văn bản STOCK.INP:

- Dòng đầu chứa ba số nguyên dương  $n, k, D$  ( $n \leq 3, k \leq 10^9, D \leq 30$ );
- Dòng thứ hai chứa  $n$  số nguyên dương  $Q_1, Q_2, \dots, Q_n$  cho biết số lượng cổ phiếu của các công ty niêm yết trên thị trường ( $Q_i \leq 100, i = 1, 2, \dots, n$ );
- Dòng thứ  $t$  trong  $D$  dòng tiếp theo chứa  $n$  số nguyên dương cho biết giá cổ phiếu của  $n$  công ty tại ngày  $t$ .

Các số trên cùng một dòng được ghi cách nhau bởi ít nhất một dấu cách.

**Output:** Tệp văn bản STOCK.OUT gồm một số nguyên dương duy nhất là tổng lượng tiền lớn nhất nhà đầu tư có được sau  $D$  ngày theo cách mua bán cổ phiếu tìm được.

Ví dụ:

STOCK.INP	STOCK.OUT
2 100 5	255
15 20	
4 5	
5 4	
7 3	
6 10	
5 12	

#### 4.57. Đồ chơi XYZ

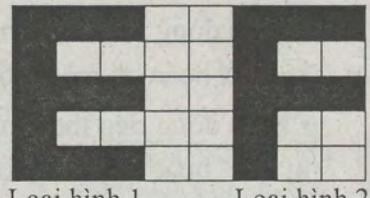
Mới đây hãng sản xuất đồ chơi có sáng kiến cho ra đời bộ đồ chơi *XYZ* “đoán hình” như sau: đồ chơi là một bảng điện tử có kích thước  $5 \times N$  điểm sáng để phân biệt được hai loại hình có kích thước  $5 \times L$  ( $L \leq N$ ).

Ví dụ:

Bảng điện tử có kích thước  $5 \times 8$  ( $N = 8$ )

5	{	1	2	3	4	5	6	7	8

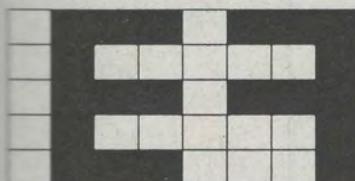
Hai loại hình kích thước  $5 \times 3$  ( $L = 3$ )



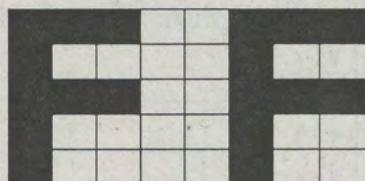
Loại hình 1

Loại hình 2

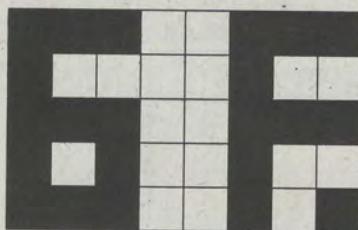
Bảng điện tử hiển thị được gọi là chứa loại hình  $i$  ( $i = 1$  hoặc  $2$ ) nếu tồn tại vị trí từ cột thứ  $k$  đến cột ( $k + L - 1$ ) trên bảng điện tử hiển thị đúng loại hình  $i$ .



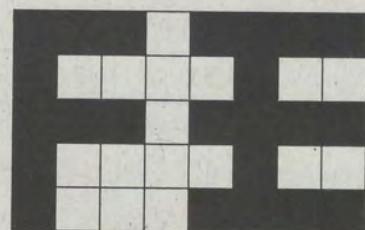
Bảng điện tử  $5 \times 8$  chỉ chứa loại hình 1



Bảng điện tử  $5 \times 8$  chỉ chứa loại hình 2



Bảng điện tử  $5 \times 8$   
không chứa loại hình 1 và hình 2



Bảng điện tử  $5 \times 8$   
chứa cả loại hình 1 và hình 2

Mỗi lần bảng điện tử sẽ hiển thị và trẻ phải trả lời câu hỏi: bảng điện tử hiển thị có chứa loại hình 1 hay loại hình 2. Do đó, người ta muốn khảo sát xem có bao nhiêu cách hiển thị bảng điện tử khác nhau để bảng điện tử luôn chỉ chứa loại 1 hoặc chỉ chứa loại 2.

**Yêu cầu:** Cho  $N$  và hai loại hình. Gọi  $k$  là số cách hiển thị bảng điện tử khác nhau để bảng điện tử luôn chỉ chứa loại 1 hoặc chỉ chứa loại 2. Hãy tính giá trị  $k \bmod 10^6$ .

**Input:** Tệp văn bản XYZ.INP có dạng:

- Dòng đầu tiên là hai số  $N$  và  $L$  ( $0 < L \leq N \leq 30$ ).
- Năm dòng tiếp theo, mỗi dòng là một xâu kí tự độ dài  $L$  chỉ gồm hai loại kí tự '.' hoặc '#' mô tả loại hình 1.
- Năm dòng tiếp theo, mỗi dòng là một xâu kí tự độ dài  $L$  chỉ gồm hai loại kí tự '.' hoặc '#' mô tả loại hình 2 (loại hình 1 khác loại hình 2).

**Output:** Tệp văn bản XYZ.OUT chứa một dòng ghi một số nguyên  $k \bmod 10^6$ .

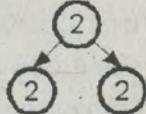
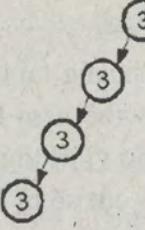
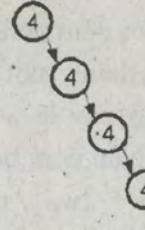
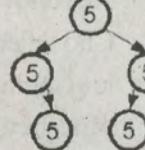
Ví dụ 1	
XYZ.INP	XYZ.OUT
3 3	2
# ##	
# ..	
# ##	
# ..	
# ##	
# ##	
# ..	
# ##	
# ..	
# ..	

Ví dụ 2	
XYZ.INP	XYZ.OUT
4 2	6138
. #	
# #	
. #	
. #	
# #	
. #	
# #	
# .	
# #	

### 4.58. Tree Tiles

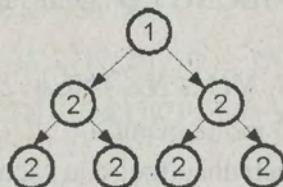
Bắc và Long cùng chơi trò chơi “Tree Tiles” như sau:

Long sẽ chọn ra một tập con các loại hình của tập gồm 5 loại hình sau:

①				
Loại 1	Loại 2	Loại 3	Loại 4	Loại 5

Bắc sẽ đếm số lượng cách xếp các loại hình mà Long đã chọn để phủ kín cây nhị phân đầy đủ độ sâu  $K$ .

Ví dụ một cách phủ bằng loại hình 1 và loại hình 2



**Yêu cầu:** Cho biết các loại hình mà Long chọn và số  $K$ , hãy giúp Bắc đếm số lượng cách xếp các loại hình mà Long đã chọn để phủ kín cây nhị phân đầy đủ độ sâu  $K$ .

**Input:** Tệp văn bản TILES.INP :

- Dòng 1: số nguyên dương  $K$  ( $1 \leq K \leq 1000$ ).
- Dòng  $i$  ( $i = 2..6$ ): là số 0 – không chọn loại hình  $i$  hoặc số 1 – chọn loại hình  $i$ .

**Output:** Tệp văn bản TILES.OUT: ghi số cách xếp mod  $10^{80}$ .

**Ví dụ :**

TILES.INP	TILES.OUT
3	
1	
1	

0	
0	
0	

### 4.59. Ném bóng

Xét chơi trò chơi ném bóng như sau:

Cho  $N$  cột xếp thành một hàng ngang tại các vị trí  $1, 2, \dots, N$ . Chúng ta có loại bóng có độ nảy là  $w$  và có thể ném hướng sang bên trái hoặc hướng sang bên phải. Nếu ném bóng vào vị trí thứ  $i$  và hướng sang bên phải thì các vị trí  $i, i + w, i + 2w, \dots$  nếu có cột sẽ bị đổ, nếu ném bóng vào vị trí  $i$  và hướng sang bên trái thì các vị trí  $i, i - w, i - 2w, \dots$  nếu có cột sẽ bị đổ. Nếu ném đổ cột  $i$  sẽ được số điểm là  $A_i$  điểm và cột đó sẽ bị loại bỏ khỏi hàng.

**Yêu cầu:** Hãy tìm cách ném bóng không vượt quá  $B$  lần ném bóng để được nhiều điểm nhất. Nếu có nhiều cách ném thì tìm cách ném với số lần ném là ít nhất.

**Input:** Tệp văn bản NEMBONG.INP gồm nhiều bộ kiểm thử: Mỗi bộ có dạng:

- Dòng đầu là ba số  $w, B, N$  ( $1 \leq N \leq 100; w, B > 0$ ).
- Dòng thứ hai gồm  $N$  số mô tả mảng  $A$  ( $|A[i]| \leq 10^6$ ).

Kết thúc tệp là ba số 0 cách nhau một dấu cách.

**Output:** Tệp văn bản NEMBONG.OUT gồm nhiều dòng, mỗi dòng là đáp án của từng bộ kiểm thử tương ứng với tệp vào có dạng: số điểm lớn nhất đạt được và số lần ném bóng.

**Ví dụ:**

NEMBONG.INP	NEMBONG.OUT
2 3 10	15 2
-1 3 2 5 1 -2 0 5 1 -3	10 3
2 3 14	0 0
-1 3 2 5 -5 -5 1 -2 0 5 -5 -5 1 -3	18 1
3 3 5	
-1 -2 -3 -4 -5	
1 2 6	
-1 -1 10 10 -1 -1	
0 0 0	

#### 4.60. Nối điểm

Trên hai đường thẳng song song  $L_1$  và  $L_2$ , người ta đánh dấu trên mỗi đường  $N$  điểm. Các điểm trên đường thẳng  $L_1$  được đánh số  $1, 2, \dots, N$  từ trái qua phải, còn các điểm trên đường  $L_2$  được đánh số bởi  $D[1], D[2], \dots, D[N]$  là một hoán vị của  $N$ , cũng được đánh dấu từ trái qua phải (dưới đây cho một ví dụ khi  $N=9$ )

1-----2-----3-----4-----5-----6-----7-----8-----9       $L_1$

2-----5-----3-----8-----7-----4-----6-----9-----1       $L_2$

Ta được phép nối điểm thứ  $i$  trên  $L_1$  với điểm thứ  $j$  trên  $L_2$  nếu  $ABS(i - D[j]) \leq 1$ .

**Yêu cầu:** Tìm cách nối được nhiều cặp điểm nhất với điều kiện các đoạn nối không được cắt nhau.

**Input:** Tệp văn bản NOIDIEM.INP:

- Dòng đầu tiên chứa số nguyên  $N$ .
- Dòng thứ hai chứa các số  $D[1], D[2], \dots, D[N]$ .

**Output:** Tệp văn bản NOIDIEM.OUT chứa số  $K$  là số lượng cặp điểm nối tìm được.

**Ví dụ:**

noidiem.inp	noidiem.out
3	2
3 2 1	

#### 4.61. Xâu gần nhất

Cho ba xâu  $X, Y, Z$  có độ dài cùng bằng  $n$ , chứa các kí tự thuộc tập 'A' ... 'Z'. Ta định nghĩa khoảng cách  $D(X, Y)$  giữa hai xâu  $X, Y$  là tổng số cặp kí tự tương ứng khác nhau trong hai xâu, cụ thể:

$$D(X, Y) = \sum_{i=1}^n D_i(x_i, y_i) \text{ trong đó } D(x_i, y_i) = 0 \text{ nếu } x_i = y_i; D(x_i, y_i) = 1 \text{ nếu } x_i \neq y_i.$$

Ví dụ:  $X = 'ABAB'$ ,  $Y = 'AAAB'$ ,  $Z = 'BBBB'$  khoảng cách hai xâu  $X$  và  $Y$  là 1, khoảng cách hai xâu  $Y$  và  $Z$  là 3.

**Yêu cầu:** Tìm xâu  $T$  sao cho khoảng cách lớn nhất của  $T$  với các xâu  $X, Y, Z$  là nhỏ nhất.

**Input:** Tệp văn bản CSTR.INP gồm ba dòng, mỗi dòng chứa một xâu, độ dài không vượt quá 100.

**Output:** Tệp văn bản CSTR.OUT chứa xâu cần tìm. Nếu có nhiều kết quả đưa ra xâu có thứ tự từ điển nhỏ nhất.

**Ví dụ:**

CSTR.INP	CSTR.OUT
ABAB	AABB
AAAA	
BBBB	

#### 4.62. Password

Bờm có một danh sách gồm  $n$  số yêu thích, đó là các con số ngày-tháng-năm sinh, đó là biển số xe, đó là số chứng minh thư,... Sau khi lập một hòm thư điện tử, Bờm quyết định đặt mật khẩu cho hòm thư điện tử từ những số yêu thích như sau: chọn một số nguyên dương  $k$  ( $1 \leq k < n$ ) rồi tìm số  $P$  lớn nhất thỏa mãn:

- Số  $P$  được tạo bởi ghép của  $k$  số.
- Số  $P$  chia hết cho 9.

Nhiệm vụ của bạn là viết một chương trình để giúp Bờm tìm số  $P$  làm mật khẩu.

**Input:** Tệp văn bản PASSWORD.INP:

- Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu tiên chứa số nguyên dương  $T$  là số lượng bộ dữ liệu. Các dòng tiếp theo chứa các bộ dữ liệu.
  - Với mỗi bộ dữ liệu, dòng đầu tiên chứa 2 số nguyên dương  $n, k$  là số lượng số yêu thích và số  $k$  mà Bờm chọn. Dòng thứ  $i$  trong  $n$  dòng tiếp theo chứa một số nguyên dương.

**Output:** Tệp văn bản PASSWORD.OUT: Với mỗi bộ dữ liệu, ghi ra trên một dòng số  $P$  tìm được hoặc số  $-1$  nếu không tồn tại số  $P$  nào thỏa mãn.

**Giới hạn:**  $T \leq 50$ ;  $1 \leq k < n \leq 100$ ; Các số yêu thích không vượt quá  $10^6$ .

Ví dụ:

password.inp	password.out
2	-1
3 2	54
1	
2	
3	
5 2	
1	
2	
3	
4	
5	

#### 4.63. Quản lí kho

Công ty XYZ có  $n$  kho và  $m$  nhân viên làm nhiệm vụ quản lí các kho. Cho biết các thông tin sau:

- Nhân viên thứ  $i$  có năng lực  $1 \leq P_i \leq 10^6$ .
- Các kho đều giống nhau và mỗi kho chỉ do một nhân viên quản lí, nhưng một nhân viên có thể quản lí nhiều kho. Nếu nhân viên  $i$  quản lí  $k$  kho thì độ an toàn của các kho đó là  $S = P_i \text{ div } k$ . Nếu một kho không có ai quản lí thì độ an toàn bằng 0.
- Độ an toàn của tất cả các kho là  $L$  bằng độ an toàn nhỏ nhất trong  $n$  kho.
- Mỗi tháng công ty sẽ trả lương cho các nhân viên, nếu nhân viên  $i$  được chọn thì sẽ phải trả  $P_i$  đồng. Tổng số tiền phải trả cho các nhân viên được chọn là  $Y$ .

**Yêu cầu:** Chọn và phân công các nhân viên quản lí các kho để độ an toàn của tất cả các kho ( $L$ ) là lớn nhất, nếu có nhiều cách phân công thì chọn cách hết ít tiền nhất ( $Y$ ).

**Input:** Tệp văn bản QLKHO.INP gồm nhiều bộ dữ liệu (có không quá 10 bộ), mỗi bộ dữ liệu có dạng:

- Dòng 1: gồm hai số  $n, m$  ( $1 \leq n, m \leq 300$ ).
- Dòng 2: gồm  $m$  số  $P_i$ .
- Kết thúc tệp  $m = n = 0$ .

**Output:** Tệp văn bản QLKHO.OUT gồm nhiều dòng, mỗi dòng gồm hai số  $L$  và  $Y$  là kết quả tương ứng với dữ liệu vào.

**Ví dụ:**

QLKHO.INP	QLKHO.OUT
2 1	3 7
7	10 10
1 2	8 18
10 9	0 0
2 5	
10 8 6 4 1	
5 4	
1 1 1 1	
0 0	

#### 4.64. COMPUTER

Công ty phần mềm XYZ mới mua  $x$  máy tính để bàn và  $y$  máy tính xách tay. Giá một chiếc máy tính để bàn là  $a$  USD, còn giá một chiếc máy tính xách tay là  $b$  USD. Để tránh sự thắc mắc giữa các phòng ban, Tổng giám đốc đã đưa ra cách phân bổ các máy tính này về  $n$  phòng ban như sau:

- Sắp xếp  $n$  phòng ban theo thứ tự về mức độ quan trọng của các phòng ban.
- Tiến hành phân bổ các máy tính cho các phòng ban, bảo đảm nếu phòng ban  $i$  có mức độ quan trọng nhỏ hơn mức độ quan trọng của phòng ban  $j$  thì tổng giá trị máy tính được phân bổ cho phòng ban  $i$  không được vượt quá tổng giá trị máy tính được phân bổ cho phòng ban  $j$ .
- Số lớn nhất các phòng ban nhận được tổng giá trị máy tính nhỏ nhất.

Là một lập trình viên giỏi nhưng lại thuộc phòng ban có mức độ quan trọng nhỏ nhất, Khanh muốn chứng tỏ tay nghề của mình với đồng nghiệp nên đã lập trình tính ra ngay được tổng giá trị máy tính mà phòng ban mình nhận được rồi mời bạn tính lại thử xem!

**Yêu cầu:** Cho  $x, a, y, b, n$ . Hãy tính tổng giá trị máy tính mà phòng Khanh nhận được.

**Input:** Gồm hai bộ dữ liệu, mỗi bộ trên một dòng, mỗi dòng chứa năm số nguyên dương  $x, a, y, b, n$  (các số có giá trị không vượt quá 1000).

**Output:** Gồm hai dòng là mỗi dòng là đáp án tương ứng với một bộ dữ liệu vào.

hai số

Ví dụ:

COMPUTER.INP	COMPUTER.OUT
3 300 2 500 2	900
4 300 3 500 2	1300

#### 4.65. Đường đi đối xứng trên lưới

Cho một lưới ô vuông gồm  $m$  dòng và  $n$  cột. Các dòng được đánh số từ 1 đến  $m$  từ trên xuống dưới, các cột được đánh số từ 1 đến  $n$  từ trái qua phải. Ô nằm ở vị trí dòng  $i$  và cột  $j$  của lưới được gọi là ô  $(i, j)$  và có ghi kí tự  $c_{ij}$ . Trên lưới đã cho, từ ô  $(i, j)$  ta có thể di chuyển đến ô  $(p, q)$  nếu:  $i \rightarrow p$ ;  $j \rightarrow q$  và  $i + j < p + q$ .

**Yêu cầu:** Xuất phát từ một ô bất kì, ta có thể di chuyển qua các ô của lưới tuân theo quy tắc di chuyển đã nêu. Tìm cách di chuyển để các kí tự nhận được lần lượt trên các ô đi qua tạo thành một xâu đối xứng dài nhất.

**Input:** Tệp văn bản PALPATH.INP:

- Dòng đầu tiên ghi hai số nguyên dương  $m, n$ .
- Dòng thứ  $i$  trong số  $m$  dòng tiếp là một xâu độ dài  $n$  mô tả lưới.

**Output:** Tệp văn bản PALPATH.OUT

- Số nguyên  $k$  là số bước di chuyển nhiều nhất có thể trên lưới.
- $k$  dòng sau, mỗi dòng là tọa độ lần lượt các bước di chuyển.

Ví dụ:

PALPATH.INP	PALPATH.OUT
2 7	7
ioivnoi	1 1
vnoiioi	1 2
	1 3
	2 4
	2 5
	2 6
	2 7

Giới hạn:  $1 \leq m, n \leq 50$ ;  $a_{ij} \in ['a'..'z']$ .

#### 4.66. Tuỳ chọn (Olympic sinh viên 2009)

Khi các hình thức khuyến mãi thông thường đã phần nào trở thành nhảm chán, không thu hút khách hàng. Chẳng hạn, ở nhà bạn đã có một rổ USB đủ các loại, vậy mà khi mua một máy tính xách tay Macbook bạn được mời nhận khuyến mãi thêm một USB 4GB.

Để đổi mới hình thức khuyến mại, siêu thị máy tính CMA (*Computer Machine for All – Máy tính cho tất cả mọi người*) đã đưa ra một phương thức khuyến mãi mới có sức thu hút lớn, đặc biệt là đối với giới trẻ sinh viên.

Nếu bạn mua một máy tính ở CMA giá từ 5 triệu bạn sẽ được cấp một mã khoá vạn năng  $P$  sử dụng một lần và một số nguyên dương  $k$ . Bạn được quyền truy cập vào trang web *CMA.Soft.com* của siêu thị. Trang web này chứa  $n$  phần mềm, đánh số từ 1 đến  $n$ . Mỗi phần mềm được lưu trữ dưới dạng một tệp ZIP và được bảo vệ bằng một khoá riêng. Khoá này vừa dùng để mở nén tệp vừa dùng để cài đặt phần mềm và đăng ký bản quyền sử dụng. Khoá thuộc loại sử dụng một lần: sau khi được dùng để mở tệp và cài đặt khoá sẽ bị vô hiệu hoá. Trong một vài tệp ZIP còn chứa tệp DOC lưu khoá truy cập tệp ZIP khác.

Thông tin trên trang web cho biết giá của mỗi phần mềm và khoá truy cập của phần mềm này được giữ ở tệp ZIP nào. Bạn được quyền mở không quá  $k$  tệp ZIP, cài đặt phần mềm mở được và sử dụng khoá hoặc những khoá lưu trữ ở tệp này để truy cập tới các tệp khác. Bạn không nhất thiết phải sử dụng hết các khoá nhận được. Ban đầu với khoá vạn năng  $P$  bạn có thể mở một tệp ZIP tùy chọn bất kì, cài đặt phần mềm đó vào máy của mình và dùng các khoá lưu trữ trong tệp này để truy cập tới các tệp khác. Giá trị máy của bạn sẽ tăng thêm một lượng đúng bằng giá trị phần mềm được cài đặt thêm. Nếu chọn cách sử dụng khoá thích hợp, giá trị máy tính của bạn có thể tăng lên gấp đôi hay gấp ba.

**Ví dụ**, với  $n = 6$ ,  $k = 3$  và thông tin về các tệp ZIP như sau:

Tệp	Giá trị	Khoá truy cập tới các tệp
1	400	4
2	400	3 và 5
3	100	1
4	1000	
5	150	2
6	750	

Nếu dùng khoá vạn năng truy cập vào tệp 2, bạn có thể cài đặt phần mềm 2. Dùng khoá 3 nhận được để truy cập và cài đặt phần mềm 3. Sau đó dùng khoá 1 để truy cập và cài đặt phần mềm 1. Tổng giá trị phần mềm cài đặt được là  $400 + 100 + 400 = 900$ . Nhưng nếu lúc đầu bạn truy cập vào tệp 1, cài đặt và truy cập tiếp đến tệp 4. Bạn chỉ cài được hai phần mềm, nhưng tổng giá trị của chúng là 1400. Song đó vẫn chưa phải là cách tốt nhất.

**Yêu cầu:** Cho  $n, k$ , giá trị của từng phần mềm và khoá kèm theo tới các tệp khác (nếu có). Khoá truy cập tới mỗi tệp được lưu ở không quá một nơi. Hãy xác định tổng giá trị lớn nhất của các phần mềm bạn có thể cài đặt vào máy của mình.

**Input:** Tệp văn bản OPTION.INP:

- Dòng đầu tiên chứa hai số nguyên  $n$  và  $k$  ( $1 \leq k \leq n \leq 100$ ),
- Dòng thứ  $i$  trong  $n$  dòng sau chứa hai số nguyên không âm  $v_i$  và  $m_i$ , trong đó  $v_i$  ( $v_i \leq 10^6$ ) - giá trị của phần mềm thứ  $i$ ,  $m_i$  - số lượng khoá lưu trữ trong tệp thứ  $i$ . Nếu  $m_i > 0$  thì sau đó là  $m_i$  số nguyên dương khác nhau từng đôi một, mỗi số có giá trị không vượt quá  $n$  – là các chỉ số của các tệp có khoá truy cập được lưu trong tệp thứ  $i$ .

Các số trên một dòng cách nhau một dấu cách.

**Output:** Tệp văn bản OPTION.OUT ghi một số nguyên là tổng giá trị lớn nhất của các phần mềm có thể cài đặt.

**Ví dụ:**

OPTION.INP
6 3
400 1 4
400 2 3 5
100 1 1
1000 0
150 1 2
750 0

OPTION.OUT
1500

#### 4.67. Đa giác không tự cắt

Cho  $n$  điểm trên mặt phẳng, trong đó có ít nhất ba điểm không thẳng hàng. Từ  $n$  điểm trên ta có thể dựng được rất nhiều đa giác không tự cắt. Trong bài toán này ta sẽ quan tâm đến các đa giác không tự cắt và diện tích của chúng.

**Yêu cầu:** Cho  $n$  điểm và số nguyên  $k$ . Hãy tìm ít nhất ba điểm và không quá  $k$  điểm trong  $n$  điểm trên, rồi dựng một đa giác không tự cắt từ các điểm được chọn để đa giác đó có diện tích lớn nhất.

**Input:** Tệp văn bản POLY.INP :

- Dòng đầu chứa hai số nguyên dương  $n, k$  ( $3 \leq k \leq n \leq 200$ );
- $n$  dòng sau, dòng thứ  $i$  gồm hai số nguyên  $x_i, y_i$  ( $|x_i|, |y_i| \leq 10^6$ ) là tọa độ điểm thứ  $i$  ( $i = 1, 2, \dots, n$ ).

**Output:** Tệp văn bản POLY.OUT chứa một số thực với hai chữ số sau dấu chấm thập phân là diện tích đa giác lớn nhất dựng được.

**Ví dụ:**

POLY.INP	POLY.OUT
4 3	3.00
0 0	
2 0	
0 3	
2 2	

#### 4.68. Đầu bếp giỏi

*Đầu bếp giỏi 2012* là cuộc thi đầu bếp giỏi do Liên hiệp hội các khách sạn du lịch tổ chức. Cuộc thi được diễn ra trong khoảng thời gian từ thời điểm  $A$  đến hết thời điểm  $B$ . Trong khoảng thời gian này, mỗi thí sinh tham gia cuộc thi phải nấu đúng  $n$  món ăn, được đánh số từ 1 đến  $n$ . Thanh là một đầu bếp kỉ cựu và quyết tâm giành giải cao nhất của cuộc thi này. Để chuẩn bị tốt nhất cho cuộc thi, Thanh đã tính được thời gian cần thiết để nấu món thứ  $i$  là  $t_i$  đơn vị thời gian và đánh giá món ăn đó có mức độ quan trọng là  $w_i$ . Dự đoán là Ban giám khảo sẽ đi chấm bài thi của mình vào khoảng thời điểm  $D$ , Thanh đã đánh giá rằng nếu món ăn thứ  $i$  nấu xong ở thời điểm  $f_i$  thì độ hấp dẫn của nó sẽ bị giảm một lượng:  $|f_i - D| \times w_i$ . Để đạt

được kết quả cao, Thanh cần xây dựng kế hoạch nấu  $n$  món ăn sao cho tổng lượng giảm độ hấp dẫn của tất cả  $n$  món ăn là nhỏ nhất. Biết rằng, tại mỗi thời điểm Thanh chỉ có thể nấu một món ăn và mỗi món ăn phải được nấu liên tục từ lúc bắt đầu cho đến khi nấu xong.

**Yêu cầu:** Cho biết số lượng các món ăn  $n$ , thời gian thực hiện nấu  $n$  món ăn  $t_1, t_2, \dots, t_n$  và các thời điểm  $A, B, D$ ; hãy xây dựng kế hoạch nấu  $n$  món ăn sao cho tổng lượng giảm độ hấp dẫn của  $n$  món ăn là nhỏ nhất.

**Input:** Tệp văn bản DBG2012.INP:

- Dòng đầu tiên chứa bốn số nguyên  $n, A, B$  và  $D$  ( $1 \leq n \leq 1000$ ,  $0 \leq A \leq D \leq B \leq 10000$ );
- Dòng thứ  $i$  trong số  $n$  dòng tiếp theo chứa 2 số nguyên  $t_i$  và  $w_i$  là thời gian cần thiết để nấu món ăn  $i$  và mức độ quan trọng của nó ( $0 < t_i, w_i \leq 100$ ,  $i = 1, 2, \dots, n$ ;  $t_1 + t_2 + \dots + t_n \leq B - A$ ).

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

**Output:** Tệp văn bản DBG2012.OUT tổng lượng giảm độ hấp dẫn của  $n$  món ăn theo kế hoạch tìm được.

**Ví dụ:**

DBG2012.INP	DBG2012.OUT	Hình vẽ minh họa
3 0 100 50	130	
20 2		
40 3		
30 2		

#### 4.69. Bảng số

Giả sử  $A$  là lưới ô vuông gồm  $m$  dòng và  $n$  cột. Các dòng của lưới được đánh số từ 1 đến  $m$ , từ trên xuống dưới. Các cột của lưới được đánh số từ 1 đến  $n$ , từ trái sang phải. Ô nằm trên giao của dòng  $i$  và cột  $j$  của lưới gọi là ô  $(i, j)$ .

Với số nguyên dương  $x$ , gọi  $f(x)$  là số lượng số nguyên dương không vượt quá  $x$  mà trong biểu diễn nhị phân có hai bit 1 đứng cạnh nhau. Ví dụ,  $f(5) = 1$  vì trong các số nguyên dương bé hơn hoặc bằng 5 chỉ có số 3 có biểu diễn nhị phân với hai bit 1 đứng cạnh nhau.

Cho dãy số nguyên dương gồm  $m \times n$  số  $b_1, b_2, \dots, b_{m \times n}$ . Ta sẽ lần lượt điền các số hạng của dãy  $f(b_1) \bmod 3, f(b_2) \bmod 3, \dots, f(b_{m \times n}) \bmod 3$  vào các ô của lưới  $A$  theo thứ tự từ trên xuống dưới từ trái qua phải. Gọi bảng số thu được là  $B$ .

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

**Yêu cầu:** Cho  $m, n$ , dãy số  $b_1, b_2, \dots, b_{m \times n}$  và  $k$  bộ  $p_i, q_i$  ( $i = 1, 2, \dots, k$ ) tương ứng với  $k$  truy vấn. Hãy đưa ra các câu trả lời cho  $k$  truy vấn.

**Input:** Tệp văn bản NUMTAB.INP:

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

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

**Output:** Tệp văn bản NUMTAB.OUT gồm  $k$  dòng, mỗi dòng chứa một số là câu trả lời cho truy vấn theo thứ tự xuất hiện trong dữ liệu vào.

**Ví dụ:**

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

## CHUYÊN ĐỀ 5. CÁC THUẬT TOÁN TRÊN ĐỒ THỊ

5.1. Cho một đồ thị có hướng  $n$  đỉnh,  $m$  cạnh được biểu diễn bằng danh sách kè, trong đó mỗi đỉnh  $u$  sẽ được cho tương ứng với một danh sách các đỉnh nối từ  $u$ . Cho một đỉnh  $v$ , hãy tìm thuật toán tính bán bậc ra và bán bậc vào của  $v$ . Xác định độ phức tạp tính toán của thuật toán.

5.2. Đồ thị chuyển vị của đồ thị có hướng  $G = (V, E)$  là đồ thị  $G^T = (V, E^T)$ , trong đó  $E^T = \{(u, v) : (v, u) \in E\}$ .

Hãy tìm thuật toán xây dựng  $G^T$  từ  $G$  trong hai trường hợp:  $G$  và  $G^T$  được biểu diễn bằng ma trận kè;  $G$  và  $G^T$  được biểu diễn bằng danh sách kè.

5.3. Cho đa đồ thị vô hướng  $G = (V, E)$  được biểu diễn bằng danh sách kè, hãy tìm thuật toán  $O(|V| + |E|)$  để xây dựng đơn đồ thị  $G' = (V, E')$  và biểu diễn  $G'$  bằng danh sách kè, biết rằng đồ thị  $G'$  gồm tất cả các đỉnh của đồ thị  $G$  và các cạnh song song trên  $G$  được thay thế bằng duy nhất một cạnh trong  $G'$ .

5.4. Cho đa đồ thị  $G$  được biểu diễn bằng ma trận kè  $A = \{a_{ij}\}$  trong đó  $a_{ij}$  là số cạnh nối từ đỉnh  $i$  tới đỉnh  $j$ . Hãy chứng minh rằng  $A^k$  là ma trận  $B = \{b_{ij}\}$  trong đó  $b_{ij}$  là số đường đi từ đỉnh  $i$  tới đỉnh  $j$  qua đúng  $k$  cạnh. Gợi ý: Sử dụng chứng minh quy nạp.

5.5. Cho đơn đồ thị  $G = (V, E)$ , ta gọi bình phương của một đồ thị  $G$  là đơn đồ thị  $G^2 = (V, E^2)$  sao cho  $(u, v) \in E^2$  nếu và chỉ nếu tồn tại một đỉnh  $w \in V$  sao cho  $(u, w)$  và  $(w, v)$  đều thuộc  $E$ . Hãy tìm thuật toán  $O(|V|^3)$  để xây dựng  $G^2$  từ  $G$  trong trường hợp cả  $G$  và  $G^2$  được biểu diễn bằng ma trận kè, tìm thuật toán  $O(|E| + |V|^2)$  để xây dựng  $G^2$  từ  $G$  trong trường hợp cả  $G$  và  $G^2$  được biểu diễn bằng danh sách kè.

5.6. Xây dựng cấu trúc dữ liệu để biểu diễn đồ thị vô hướng và các thao tác:

- Liệt kê các đỉnh kè với một đỉnh cho trước trong thời gian  $O(|E|)$ ;
- Kiểm tra hai đỉnh có kè nhau hay không trong thời gian  $O(1)$ ;
- Loại bỏ một cạnh trong thời gian  $O(1)$ .

5.7. Với đồ thị  $G = (V, E)$ , được biểu diễn bằng ma trận kè, đa số các thuật toán trên đồ thị sẽ có độ phức tạp tính toán  $\Omega(|V|^2)$ , tuy nhiên không phải không

có ngoại lệ. Chẳng hạn bài toán tìm “bồn chứa” (universal sink) trong đồ thị bồn chứa trong đồ thị có hướng là một đỉnh nối từ tất cả các đỉnh khác và không có cung đi ra. Hãy tìm thuật toán  $O(|V|)$  để xác định sự tồn tại và chỉ ra bồn chứa trong đồ thị có hướng.

- 5.8.** Người ta còn có thể biểu diễn đồ thị bằng *ma trận liên thuộc* (incidence matrix): Với đồ thị có hướng  $G = (V, E)$ , có  $n$  đỉnh và  $m$  cung, ma trận liên thuộc  $B = \{b_{ij}\}$  của  $G$  kích thước  $m \times n$ , trong đó:

$$b_{ij} = \begin{cases} -1, & \text{nếu cung thứ } j \text{ ra khỏi đỉnh } i \\ 1, & \text{nếu cung thứ } j \text{ đi vào đỉnh } i \\ 0, & \text{nếu cung thứ } j \text{ không liên thuộc với đỉnh } i. \end{cases}$$

Xét  $B^T$  là ma trận chuyển vị của ma trận  $B$ , hãy cho biết ý nghĩa của ma trận tích  $BB^T$ .

- 5.9.** Viết chương trình cài đặt thuật toán DFS không đệ quy.  
**5.10.** Xét đồ thị có hướng  $G = (V, E)$ , dùng thuật toán DFS duyệt đồ thị  $G$ . Cho một phản ví dụ để chứng minh giả thuyết sau là sai: Nếu từ đỉnh  $u$  có đường đi tới đỉnh  $v$  và  $u$  được duyệt đến trước  $v$ , thì  $v$  nằm trong nhánh DFS gốc  $u$ .  
**5.11.** Cho đồ thị vô hướng  $G = (V, E)$ , tìm thuật toán  $O(|V|)$  để phát hiện một chu trình đơn trong  $G$ .  
**5.12.** Cho đồ thị có hướng  $G = (V, E)$  có  $n$  đỉnh, và mỗi đỉnh  $i$  được gán một nhãn là số nguyên  $a_i$ , tập cung  $E$  của đồ thị được định nghĩa là

$$(u, v) \in E \Leftrightarrow a_u \geq a_v.$$

Giả sử rằng thuật toán DFS được sử dụng để duyệt đồ thị, hãy khảo sát tính chất của dãy các nhãn nếu ta xếp các đỉnh theo thứ tự từ đỉnh duyệt xong đầu tiên đến đỉnh duyệt xong sau cùng.

- 5.13.** Mê cung hình chữ nhật kích thước  $m \times n$  gồm các ô vuông đơn vị ( $m, n \leq 1000$ ). Trên mỗi ô ghi một trong bốn ký tự:
- $O$ : Nếu ô đó an toàn;
  - $X$ : Nếu ô đó có cạm bẫy;
  - $E$ : Nếu là ô có một nhà thám hiểm đang đứng.

Duy nhất chỉ có 1 ô ghi chữ  $E$ . Nhà thám hiểm có thể từ một ô đi sang một trong số các ô chung cạnh với ô đang đứng. Một cách đi thoát khỏi mê cung

là một hành trình đi qua các ô an toàn ra một ô biên. Hãy chỉ giúp cho nhà thám hiểm một hành trình thoát ra khỏi mê cung đi qua ít ô nhất.

**5.14.** Chứng minh rằng đồ thị có hướng  $G = (V, E)$  là không có chu trình nếu và chỉ nếu quá trình thực hiện thuật toán tìm kiếm theo chiều sâu trên  $G$  không có cung ngược.

**5.15.** Cho đồ thị có hướng không có chu trình  $G = (V, E)$  và hai đỉnh  $s, t$ . Hãy tìm thuật toán đếm số đường đi từ  $s$  tới  $t$  (chỉ cần đếm số lượng, không cần liệt kê các đường).

**5.16.** Trên mặt phẳng với hệ toạ độ Đè-các vuông góc cho  $n$  đường tròn, mỗi đường tròn xác định bởi bộ ba số thực  $(x, y, r)$  ở đây  $(x, y)$  là toạ độ tâm và  $r$  là bán kính. Hai đường tròn gọi là thông nhau nếu chúng có điểm chung. Hãy chia các đường tròn thành một số tối thiểu các nhóm sao cho hai đường tròn bất kì trong một nhóm bắt kì có thể đi được sang nhau sau một số hữu hạn các bước di chuyển giữa hai đường tròn thông nhau.

**5.17.** Cho một lưới ô vuông kích thước  $m \times n$  gồm các số nhị phân thuộc  $\{0, 1\}$  ( $m, n \leq 1000$ ). Ta định nghĩa một hình là một miền liên thông các ô kề cạnh mang số 1. Hai hình được gọi là giống nhau nếu hai miền liên thông tương ứng có thể đặt chồng khít lên nhau qua một phép dời hình. Hãy phân loại các hình trong lưới ra thành một số các nhóm thỏa mãn: Mỗi nhóm gồm các hình giống nhau và hai hình bất kì thuộc hai nhóm khác nhau thì không giống nhau:

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

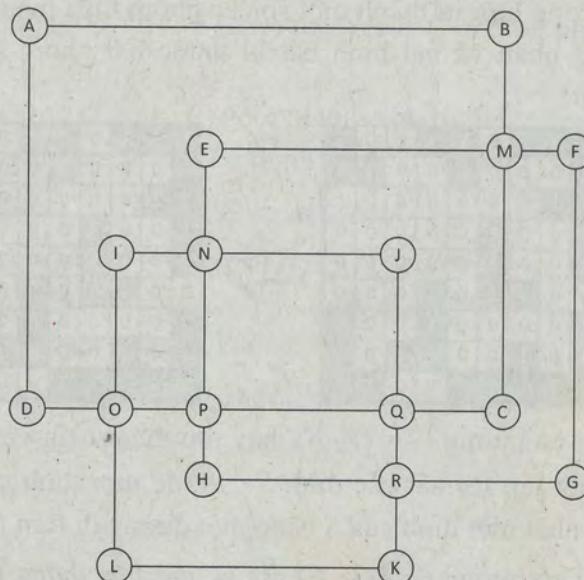


1	1	1	0	2	2	0	0	2
1	0	0	0	2	0	0	2	2
1	1	0	0	0	0	0	0	0
1	0	0	3	0	0	0	0	0
1	0	0	3	0	0	0	0	0
0	0	3	3	0	3	0	0	0
1	0	0	0	0	3	0	0	3
1	0	1	0	0	3	3	0	3
1	1	1	1	1	0	0	3	3

**5.18.** Cho đồ thị có hướng  $G = (V, E)$ , hãy tìm thuật toán và viết chương trình để chọn ra một tập ít nhất các đỉnh  $S \subseteq V$  để mọi đỉnh của  $V$  đều có thể đến được từ ít nhất một đỉnh của  $S$  bằng một đường đi trên  $G$ .

**5.19.** Một đồ thị có hướng  $G = (V, E)$  gọi là *nửa liên thông* (semi-connected) nếu với mọi cặp đỉnh  $u, v \in V$  thì hoặc  $u$  có đường đi đến  $v$ , hoặc  $v$  có đường đi đến  $u$ .

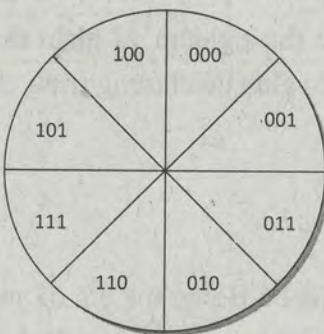
- a) Chứng minh rằng đồ thị có hướng  $G = (V, E)$  là nửa liên thông nếu và chỉ nếu trên  $G$  tồn tại đường đi qua tất cả các đỉnh (không nhất thiết phải là đường đi đơn).
- b) Tìm thuật toán và viết chương trình kiểm tra tính nửa liên thông của đồ thị.
- 5.20.** Hãy sửa đổi thuật toán liệt kê khớp và cầu của đồ thị, sửa đổi thuật toán liệt kê các thành phần song liên thông sao cho không cần phải thực hiện việc định chiều đồ thị nữa (Bởi vì việc định chiều một đồ thị tỏ ra khá cồng kềnh và không hiệu quả nếu đồ thị được biểu diễn bằng danh sách kề hay danh sách cạnh).
- 5.21.** Tìm thuật toán đếm số cây khung của đồ thị (hai cây khung gọi là khác nhau nếu chúng có ít nhất một cạnh khác nhau).
- 5.22.** Trên mặt phẳng cho  $n$  hình chữ nhật có các cạnh song song với các trục toạ độ. Hãy chỉ ra một chu trình:
- Chỉ đi trên cạnh của các hình chữ nhật.
  - Trên cạnh của mỗi hình chữ nhật, ngoại trừ những giao điểm với cạnh của hình chữ nhật khác có thể qua nhiều lần, những điểm còn lại chỉ được quét đúng một lần.



ABMFGRHPNEMCQRKLOINJQPODA

5.23. Trong đám cưới của Persée và Andromède có  $2n$  hiệp sĩ. Mỗi hiệp sĩ có không quá  $n - 1$  kẻ thù. Hãy giúp Cassiopé, mẹ của Andromède xếp  $2n$  hiệp sĩ ngồi quanh một bàn tròn sao cho không có hiệp sĩ nào phải ngồi cạnh kẻ thù của mình. Mỗi hiệp sĩ sẽ cho biết những kẻ thù của mình khi họ đến sân rồng.

5.24. **Gray code:** Một hình tròn được chia thành  $2n$  hình quạt đồng tâm. Hãy xếp tất cả các xâu nhị phân độ dài  $n$  vào các hình quạt, mỗi xâu vào một hình quạt sao cho bất cứ hai xâu nào ở hai hình quạt cạnh nhau đều chỉ khác nhau đúng 1 bit. Ví dụ với  $n = 3$ :



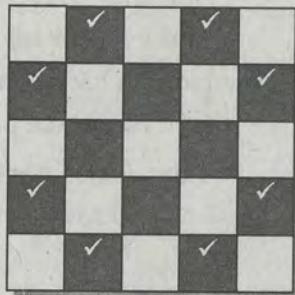
5.25. **Bài toán mã đi tuần:** Trên bàn cờ tổng quát kích thước  $m \times n$  ô vuông ( $5 \leq m, n \leq 1000$ ). Một quân mã đang ở ô  $(x_1, y_1)$  có thể di chuyển sang ô  $(x_2, y_2)$  nếu  $|x_1 - x_2|, |y_1 - y_2| = 2$  (xem hình).

Hãy tìm hành trình của quân mã từ ô xuất phát từ một ô tùy chọn, đi qua tất cả các ô của bàn cờ, mỗi ô đúng 1 lần.

Ví dụ với  $n = 8$ .

*Hướng dẫn:* Nếu coi các ô của bàn cờ là các đỉnh của đồ thị và các cạnh là nối giữa hai đỉnh tương ứng với hai ô mã giao nhau thì dễ thấy rằng hành trình của quân mã cần tìm sẽ là một đường đi Hamilton. Tuy vậy thuật toán duyệt thuần túy là bất khả thi với dữ liệu lớn, bạn có thể thử cài đặt.

Để giải quyết bài toán mã đi tuần, có một mẹo nhỏ



15	26	39	58	17	28	37	50
40	59	16	27	38	51	18	29
25	14	47	52	57	30	49	36
46	41	60	31	48	53	56	19
13	24	45	62	1	20	35	54
42	61	10	23	32	55	2	5
9	12	63	44	7	4	21	34
64	43	8	11	22	33	6	3

được Warnsdorff đưa ra cách đây gần 2 thế kỉ (1823). Mẹo này không áp dụng được vào bài toán mã đi tuần mà còn có thể kết hợp vào thuật toán duyệt để tìm đường đi Hamilton trên đồ thị bất kì nếu biết chắc đường đi tồn tại (duyệt tham phối hợp).

Với mỗi ô  $(x, y)$  ta gọi bậc của ô đó,  $\deg(x, y)$  là số ô kề với ô  $(x, y)$  chưa được thăm (kề ở đây theo nghĩa định kè chứ không phải là ô kề cạnh). Đang nhiên quân mã vào ô  $(x, y)$  nào đó và cứ di chuyển quân mã sang ô có bậc nhỏ nhất. Nếu đi được hết bàn cờ thì xong, nếu không ta đặt ngẫu nhiên quân mã vào một ô xuất phát khác và làm lại.

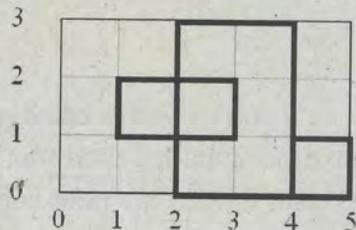
Thuật toán này đã được thử nghiệm và nhận thấy rằng việc tìm ra một bm, n:  $5 \leq m, n \leq 1000$  thời gian để chương trình chạy lớn hơn 10 giây cũng một chuyện bất khả thi.

## Bài tập bổ sung

### 5.26. Cắt bánh

Trong ngày sinh nhật của bé Bông, mẹ em đã mua tặng em một chiếc bánh hình chữ nhật  $M \times N$  mảnh bánh con. Bông đã đưa ra một cách cắt bánh như sau: Mỗi người sẽ chọn một mảnh bánh hình chữ nhật của mình (có cạnh song song với chiếc bánh ban đầu, chứa nguyên các mảnh bánh con) sau đó sẽ cắt mảnh bánh đó nhưng chưa lấy bánh ra. Sau khi tất cả mọi người đã cắt bánh xong, Bông muốn biết là mảnh bánh ban đầu đã được chia thành bao nhiêu phần.

**Ví dụ:** Chiếc bánh có kích thước  $3 \times 5$ , có ba người sẽ cắt bánh và kết quả sẽ nhận được 6 phần bánh.



**Yêu cầu:** Cho kích thước bánh ban đầu và các mảnh bánh sẽ được cắt. Hãy lập trình giúp bé Bông đếm xem sẽ nhận được bao nhiêu phần bánh.

**Input:** Tệp văn bản CAKE.INP gồm nhiều bộ kiểm thử, mỗi bộ có dạng:

- Dòng đầu gồm hai số nguyên dương  $M, N$  ( $0 < M, N < 106$ );

không c  
thuật to  
trường đi  
( $x, y$ ) ch  
cạnh). Đ  
sang ô  
đặt ngà  
a một b  
y cũng l  
iếc bán  
ánh nh  
(có ca  
nh con  
cả m  
ược c  
**5.27. ANT**

- Dòng tiếp theo là một số nguyên dương  $K$  ( $0 < K < 51$ ), là số người tham gia cắt bánh.

- $K$  dòng sau, mỗi dòng bốn số  $x_1, y_1, x_2, y_2$  mô tả các mảnh bánh của từng người ( $0 \leq x_1, x_2 \leq N$  và  $0 \leq y_1, y_2 \leq M$ ).

Kết thúc tệp khi  $M, N$  bằng 0.

**Output:** Tệp văn bản CAKE.OUT gồm nhiều dòng, mỗi dòng có một số là số phần bánh nhận được tương ứng với dữ liệu vào.

**Ví dụ:**

Cake.inp	Cake.out
3 5	6
3	3
1 1 3 2	
4 0 2 3	
4 0 5 1	
6 6	
2	
2 0 5 3	
3 1 4 2	
0 0	

Một khu vườn được xem như là một lưới các ô vuông, có một tổ kiến ở ô có toạ độ  $(0,0)$  và có một số ô trên lưới có vật cản. Một chú kiến muốn đi tìm thức ăn, kiến sẽ đi theo quy tắc sau:

- Từ một ô kiến có thể đi sang được 4 ô chung cạnh.
- Kiến không đi vào ô có vật cản.
- Kiến không đi xa tổ quá  $S$  bước.

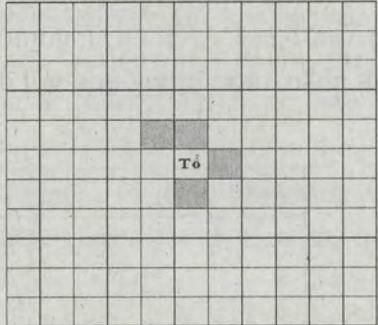
**Yêu cầu:** Cho biết toạ độ các ô có vật cản và số  $S$ , hỏi kiến có thể đến được tất cả bao nhiêu ô?

**Input:** Tệp văn bản ANT.INP:

- Dòng đầu là hai số  $C$  (số ô có vật cản) và  $S$  ( $0 \leq C \leq 10000$ ;  $1 \leq S \leq 10^7$ ).
- $C$  dòng sau, mỗi dòng hai số nguyên  $x_i, y_i$  là toạ độ của các ô chứa vật cản ( $|x_i|, |y_i| < 1001$ ).

**Output:** Tập văn bản ANT.OUT gồm một dòng duy nhất là số ô mà kiến có thể đến được.

**Ví dụ:**

ANT.INP	ANT.OUT	Hình mô tả cho ví dụ
4 5 -1 1 0 -1 0 1 1 0	26	

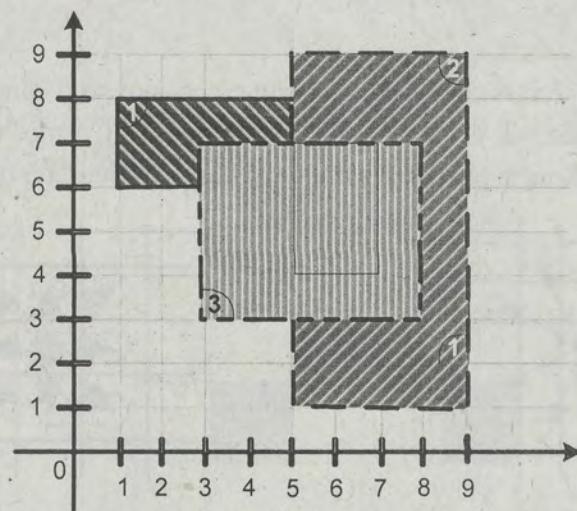
### 5.28. Virus

Virus cúm gia cầm H5N1 đang lan rộng trên thế giới cũng như ở nước ta. Loại virus này có khả năng biến đổi rất nhanh để tạo ra các biến thể khác nhau. Để kiểm soát sự lây nhiễm, người ta tiến hành khảo sát trên một vùng diện tích hình vuông được chia thành một lưới ô vuông có toạ độ hai đỉnh đối nhau là  $(0, 0)$  và  $(10^6, 10^6)$ . Phân tích ADN cho thấy, hiện tại đã tồn tại  $m$  biến thể khác nhau được đánh số hiệu từ 1 đến  $m$ . Tại một ô vuông có thể tồn tại nhiều biến thể khác nhau.

Khảo sát cho thấy mỗi biến thể lan truyền trên một hoặc một số vùng (gọi là *vùng bệnh*). Mỗi vùng bệnh có dạng một hình đa giác không tự cắt trên mặt phẳng toạ độ có các cạnh song song với trục toạ độ (các đỉnh của đa giác có toạ độ nguyên). Hai vùng bệnh cùng nhiễm một biến thể thì không có cạnh chung. Hai vùng bệnh ứng với hai biến thể khác nhau có thể có diện tích chung khác 0. Có  $n$  vùng bệnh được đánh số từ 1 đến  $n$ .

Một hiện tượng phổ biến và nguy hiểm là các biến thể khác nhau của virus có thể kết hợp với nhau để tạo ra một biến thể virus mới nguy hiểm và khó kiểm soát hơn. Theo tính toán, nếu  $K$  biến thể của virus *cùng tồn tại* trong một miền gồm các ô vuông liên thông theo cạnh có diện tích là  $S$  thì khả năng chúng kết hợp để tạo ra một biến thể mới là  $S*K^2$ .

kiến có



Hình vẽ mô tả 4 vùng bệnh.

Số hiệu của biến thể virus trong vùng bệnh được ghi ở góc của đa giác.

**Yêu cầu:** Hãy tính khả năng lớn nhất một biến thể mới của virus cúm H5N1 được tạo ra.

**Input:** Tệp văn bản VIRUS.INP:

- Dòng đầu chứa hai số nguyên dương  $n, m$  ( $n \leq 50; m \leq 10$ );
- Dòng thứ  $i$  trong  $n$  dòng tiếp theo chứa các số nguyên  $v, t, x_{i1}, y_{i1}, \dots, x_{it}, y_{it}$  cho biết vùng bệnh thứ  $i$  có biến thể virus  $v$  và được mô tả bởi đa giác có  $t$  đỉnh ( $t \leq 10; 0 \leq x_{it}, y_{it} \leq 10^6$ ). Các đỉnh của đa giác được liệt kê liên tiếp theo một chiều đi vòng quanh đa giác.

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

**Output:** Tệp văn bản VIRUS.OUT gồm một dòng chứa khả năng lớn nhất một biến thể mới của virus cúm H5N1 được tạo ra.

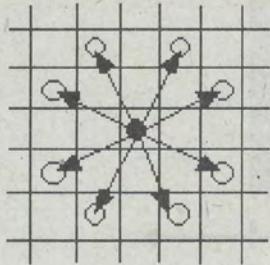
**Ví dụ:**

VIRUS.INP
4 3
1 6 1 6 1 8 7 8 7 4 4 4 4 6
2 4 3 3 3 7 8 7 8 3
3 4 5 1 5 9 9 9 9 1
1 4 9 2 7 2 7 4 9 4

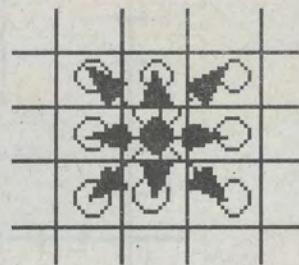
VIRUS.OUT
54

### 5.29. Mã và tốt

Cho bàn cờ có kích thước  $N \times N$ , trên bàn cờ có một số ô cấm những ô còn lại là những ô tự do - ô có thể di chuyển vào được. Có một quân mã và một quân tốt đang đứng ở hai ô khác nhau, cả hai ô đó đều là ô tự do.



*Quy tắc di chuyển của quân mã*



*Quy tắc di chuyển của quân tốt*

Tại mỗi bước cả hai quân đều phải di chuyển theo quy tắc và không được đi vào ô cấm, hãy tìm cách di chuyển quân mã và tốt để chúng gặp nhau nhanh nhất.

**Input:** Tệp văn bản MATOT.INP:

- Dòng đầu là số  $N$  ( $2 < N \leq 100$ ).
- $N$  dòng tiếp theo, mỗi dòng là một xâu  $N$  kí tự, gồm các kí tự “.” thể hiện ô tự do, kí tự ‘#’ thể hiện ô cấm không được phép đi vào, duy nhất một kí tự ‘T’ thể hiện vị trí tốt đang đứng, duy nhất một kí tự ‘M’ thể hiện vị trí quân mã.

**Output:** Tệp văn bản MATOT.OUT gồm một số là số bước ít nhất để quân tốt và quân mã gặp nhau, nếu không thể gặp được nhau ghi -1.

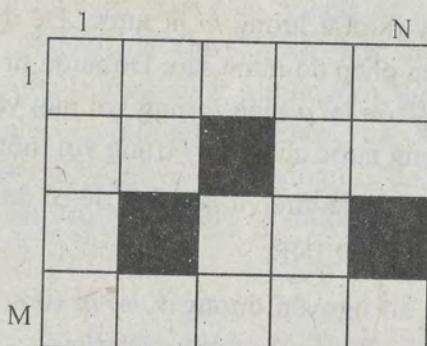
**Ví dụ:**

MATOT.INP	MATOT.OUT
5 M.... ..... .#... .#..# ...#T	2

### 5.30. PATH

còn lại  
và một

An và Bình chơi một trò chơi trên lưới ô vuông gồm  $M$  dòng,  $N$  cột, đã có một số ô đã được tô màu như hình vẽ sau:



An cố gắng tìm cách di chuyển một quân tốt từ ô  $(1, 1)$  đến ô  $(M, N)$ . Bình là người cố gắng tìm cách tô màu một số ô của lưới nhằm mục đích ngăn cản An không thể thực hiện được ý định của mình (quân tốt chỉ di chuyển vào ô màu trắng và sang các ô chung cạnh).

**Yêu cầu:** Bình chỉ được tô màu các ô trên các dòng 2 đến dòng  $M - 1$ , tìm cách tô ít ô nhất để An không thể di chuyển tốt từ ô  $(1, 1)$  đến ô  $(M, N)$ .

**Input:** Tệp văn bản PATH.INP

- Dòng đầu gồm hai số  $M, N$  ( $2 < M, N < 1001$ ).
- $M$  dòng sau, mỗi dòng một xâu gồm  $N$  kí tự '.' (ô màu trắng) hoặc '#' (ô màu đen). Chú ý rằng dòng đầu và dòng cuối không có ô màu đen.

**Output:** Tệp văn bản PATH.OUT gồm một số duy nhất là số ô mà Bình phải tô màu.

**Ví dụ:**

Path.inp	Path.out	Hình minh họa
4 5 ..... . # ... . # ... # .....	2	

### 5.31. Đỗ nước

Có ba bình nước giống nhau, trên mỗi bình có  $n$  vạch ( $p_1 < p_2 < \dots < p_n$ ,  $p_i$  nguyên dương). Ban đầu ta có  $v$  lít nước ở bình 1, còn bình 2 và 3 không chứa nước. Cần lấy ra một lượng  $m$  lít nước. Để đạt được mục đích đó, ta được phép thực hiện phép đổ nước sau: Đỗ nước từ bình  $i$  sang bình  $j$  ( $i \neq j$ ) sao cho lượng nước còn lại ở bình  $i$  trùng với một vạch trên bình  $i$  hoặc bình  $i$  hết nước hoặc lượng nước của bình  $j$  trùng với một vạch trên bình  $j$ .

**Yêu cầu:** Hãy tìm cách đổ nước ít lần nhất để có thể lấy ra được  $m$  lít nước.

**Input:** Tệp văn bản POUR.INP:

- Dòng đầu là các số nguyên dương  $v, n, m$  ( $0 < v \leq 30000, 0 < n \leq 20, 0 < m \leq v$ ).

- Dòng thứ hai là  $n$  số nguyên dương  $p_1, p_2, \dots, p_n$  ( $0 < p_1 < p_2 < \dots < p_n \leq v$ ).

**Output:** Tệp văn bản POUR.OUT: Gồm một số là số lần đổ ít nhất (nếu không có cách đổ ghi -1).

**Ví dụ:**

POUR.INP	POUR.OUT	Giải thích
5 1 3 1	2	ban đầu: 5 0 0 đỗ 1 sang 2: 4 1 0 đỗ 1 sang 3: 3 1 1
5 1 2 1	3	ban đầu: 5 0 0 đỗ 1 sang 2: 4 1 0 đỗ 1 sang 3: 3 1 1 đỗ 2 sang 3: 3 0 2
6 1 1 2	-1	

### 5.32. Đến trường (Olympic sinh viên 2010)

Gia đình Tuấn sống ở thành phố XYZ. Hằng ngày, mẹ đi ô tô đến cơ quan làm việc còn Tuấn đi bộ đến trường học. Thành phố XYZ có  $N$  nút giao thông được đánh số từ 1 đến  $N$ . Nhà Tuấn nằm ở nút giao thông 1, trường của Tuấn nằm ở nút giao thông K, cơ quan của mẹ nằm ở nút giao thông M. Từ nút  $i$  đến nút  $j$  có không quá một đường đi một chiều, tất nhiên, có thể

đường đi một chiều khác đi từ nút  $j$  đến nút  $i$ . Nếu từ nút  $i$  đến nút  $j$  có đường đi thì thời gian đi bộ từ nút  $i$  đến nút  $j$  hết  $a_{ij}$  phút, còn đi ô tô hết  $b_{ij}$  ( $0 < b_{ij} \leq a_{ij}$ ) phút.

Hôm nay, mẹ và Tuấn cùng xuất phát từ nhà lúc 7 giờ. Tuấn phải có mặt tại trường lúc 7 giờ 59 phút để kịp vào lớp học lúc 8 giờ. Tuấn băn khoăn không biết có thể đến trường đúng giờ hay không, nếu không Tuấn sẽ phải nhờ mẹ đưa đi từ nhà đến một nút giao thông nào đó.

**Yêu cầu:** Cho biết thông tin về các đường đi của thành phố XYZ. Hãy tìm cách đi để Tuấn đến trường không bị muộn giờ còn mẹ đến cơ quan làm việc sớm nhất.

**Input:** Tệp văn bản SCHOOL.INP:

- Dòng đầu ghi ba số nguyên dương  $N, M, K$  ( $3 \leq N \leq 10000 ; M \leq 10^5$ ;  $1 < K < N$ ), trong đó  $N$  là số nút giao thông,  $M$  là số đường đi một chiều,  $K$  là nút giao thông, trường của Tuấn.
- $M$  dòng tiếp theo, mỗi dòng chứa bốn số nguyên dương  $i, j, a_{ij}, b_{ij}$  ( $1 \leq i, j \leq N$ ,  $b_{ij} \leq a_{ij} \leq 60$ ) mô tả thông tin đường đi một chiều từ  $i$  đến  $j$ .

Hai số liên tiếp trên một dòng cách nhau một dấu cách. Dữ liệu bảo đảm luôn có nghiệm.

**Output:** Tệp văn bản SCHOOL.OUT gồm một dòng chứa một số nguyên là thời gian sớm nhất mẹ Tuấn đến được cơ quan còn Tuấn thì không bị muộn học.

**Ví dụ:**

SCHOOL.INP
5 6 3
1 4 60 40
1 2 60 30
2 3 60 30
4 5 30 15
4 3 19 10
3 5 20 10

SCHOOL.OUT
55

### 5.33. Vé xe miễn phí

An sống ở thành phố XYZ, hàng ngày anh phải đi làm từ nhà tới cơ quan bằng xe buýt. Thành phố XYZ có  $n$  nút giao thông được đánh số từ 1 đến  $n$  và  $m$  tuyến xe buýt hai chiều. Mỗi cặp nút giao thông  $i, j$  có không quá một tuyến xe buýt hai chiều, nếu có thì để đi từ nút  $i$  đến nút  $j$  (hoặc từ nút  $j$  đến

nút  $i$ ) với giá vé là  $c_{ij} = c_{ji}$  đồng. Vị trí nhà An nằm ở nút giao thông 1 còn cơ quan nằm ở nút giao thông  $n$ . Để lựa chọn đường đi từ nhà đến cơ quan An luôn chọn theo đường đi với chi phí ít nhất.

**Ví dụ:** Thành phố có 5 nút giao thông và 6 tuyến xe buýt:

Tuyến 1: 1-2 giá vé 10 đồng;

Tuyến 2: 2-5 giá vé 10 đồng,

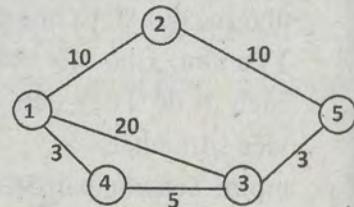
Tuyến 3: 1-4 giá vé 3 đồng;

Tuyến 4: 3-4 giá vé 5 đồng,

Tuyến 5: 3-5 giá vé 3 đồng;

Tuyến 6: 1-3 giá vé 20 đồng.

Đường đi  $1 \rightarrow 4 \rightarrow 3 \rightarrow 5$  hết 11 đồng là ít nhất.



Vừa qua An nhận được một vé đi xe buýt miễn phí. Vé có thể dùng để đi xe buýt miễn phí một lần trên một tuyến bất kì. Với vé xe miễn phí này An muốn biết chi phí ít nhất để đi từ nhà đến cơ quan là bao nhiêu.

Với ví dụ trên, đường đi  $1 \rightarrow 3 \rightarrow 5$  có sử dụng vé xe miễn phí (tại tuyến 1-3) hết 3 đồng là ít nhất.

**Yêu cầu:** Cho biết các tuyến xe buýt và giá vé tương ứng. Hãy tìm chi phí ít nhất để đi từ nhà (nút giao thông 1) đến cơ quan (nút giao thông  $n$ ) với vé xe miễn phí mà An có.

**Input:** Tệp văn bản FT.INP :

- Dòng đầu tiên ghi hai số nguyên dương  $n$  và  $m$  ( $3 \leq n \leq 5000$ ,  $m \leq 30000$ ).
- $m$  dòng sau, mỗi dòng ba số nguyên  $i, j, c_{ij}$  ( $1 \leq i, j \leq n$ ,  $0 \leq c_{ij} \leq 30000$ ) mô tả có tuyến xe buýt  $i - j$  hết  $c_{ij}$  đồng.

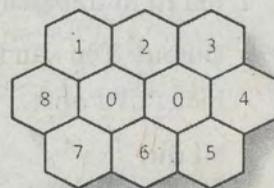
Hai số liên tiếp trên một dòng cách nhau một dấu cách. Dữ liệu bảo đảm luôn có đường đi từ 1 đến  $n$ .

**Output:** Tệp văn bản FT.OUT là một số duy nhất là chi phí ít nhất để đi từ nhà (nút giao thông 1) đến cơ quan (nút giao thông  $n$ ) với vé xe miễn phí.

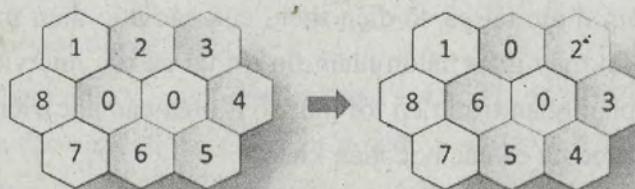
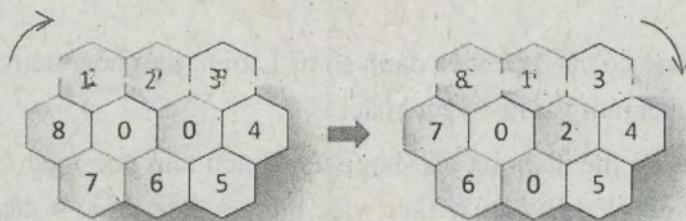
ví dụ 1		ví dụ 2	
FT.INP	FT.OUT	FT.INP	FT.OUT
5 6	3	5 5	6
1 2 10		1 2 10	
2 5 10		2 5 10	
1 4 3		1 4 3	
3 4 5		4 3 5	
3 5 3		3 5 3	
1 3 20			

### 5.34. Hexgame (Olympic sinh viên 2011)

HEXGAME là một trò chơi xếp hình gồm 10 miếng ghép hình lục giác đều. Trên mỗi miếng ghép được điền một số nguyên, có 8 miếng được điền số từ 1 đến 8 và có hai miếng điền số 0. Các miếng liên kết với nhau tạo thành lưới tổ ong. Ban đầu các miếng ghép ở vị trí như hình bên.



Tại mỗi bước, chọn một miếng ghép có đúng 6 miếng ghép kề cạnh làm tâm, rồi xoay một nắc 6 miếng ghép kề cạnh đó theo chiều kim đồng hồ. Như vậy chỉ có hai cách chọn tâm. Ví dụ với trạng thái ban đầu nêu trên thì nhận được một trong hai trạng thái dưới đây ứng với cách chọn sau khi xoay một nắc.



**Yêu cầu:** Cho một trạng thái của trò chơi (nhận được sau một dãy biến đổi từ trạng thái ban đầu), hãy tính số phép biến đổi ít nhất để đưa về trạng thái ban đầu.

**Input:** Tệp văn bản HEXGAME.INP có dạng:

- Dòng 1: chứa ba số ghi trên ba miếng ghép ở dòng thứ nhất của lưới thứ tự từ trái qua phải;
- Dòng 2: chứa bốn số ghi trên bốn miếng ghép ở dòng thứ hai của lưới theo thứ tự từ trái qua phải;
- Dòng 3: chứa ba số ghi trên ba miếng ghép ở dòng thứ ba của lưới thứ tự từ trái qua phải.

**Output:** Tệp văn bản HEXGAME.OUT gồm một dòng ghi một số là số phép biến đổi ít nhất.

**Ví dụ:**

HEXGAME.INP	HEXGAME.OUT
1 0 2 8 6 0 3 7 5 4	5

### 5.35. Nhắn tin

Một khoá học có các học viên đánh số từ 1 tới  $n$ , mỗi học viên có thể biết số điện thoại của một vài học viên khác.

Học viên A có thể nhắn tin cho học viên B nếu như học viên A biết số điện thoại của học viên B. Lưu ý rằng việc biết số điện thoại ở đây không phải quan hệ đối xứng: Có thể học viên A biết số điện thoại của học viên B nhưng học viên B hoàn toàn không biết số điện thoại của học viên A.

Thầy giáo năm được tất cả số điện thoại của các học viên trong hồ sơ trường. Hỏi khi thầy giáo muốn nhắn tin tới tất cả các học viên trong lớp, thầy giáo sẽ phải nhắn trực tiếp tới một số ít nhất các học viên nào để thông điệp đó đến được tất cả các học viên khác?

**Input:** Tệp văn bản MESSAGE.INP:

- Dòng 1 chứa hai số nguyên dương  $n, m \leq 10^5$ .

- $m$  dòng tiếp theo, mỗi dòng chứa hai số nguyên dương  $x, y$  cho ta thông tin: học viên  $x$  biết số điện thoại của học viên  $y$ .

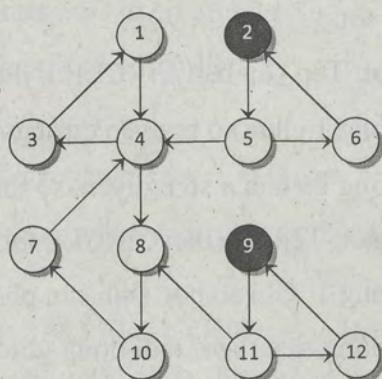
**Output:** Tập văn bản MESSAGE.OUT:

- Dòng 1: Ghi số  $k$  là số học sinh được thầy giáo nhắn tin trực tiếp khi cần;
- Dòng 2: Ghi  $k$  số hiệu của các học sinh được thầy giáo nhắn tin trực tiếp.

Các số trên một dòng phải ghi cách nhau ít nhất một dấu cách.

**Ví dụ:**

MESSAGE.INP	MESSAGE.OUT
12 15	2
1 4	9 2
2 5	
3 1	
4 3	
4 8	
5 4	
5 6	
6 2	
7 4	
8 10	
9 11	
10 7	
11 8	
11 12	
12 9	



### 5.36. Dây chuyền thông báo

Có  $n$  học sinh trong lớp đánh số từ 1 tới  $n$ . Trước kì nghỉ hè, mỗi học sinh tự chọn cho mình đúng một người khác  $\varphi(s)$  gọi là người truyền tin. Mỗi khi nhận được thông điệp, học sinh  $s$  sẽ lập tức chuyển thông điệp đó cho người truyền tin  $\varphi(s)$  của mình.

Dây chuyền thông báo được gọi là tốt nếu nó thoả mãn điều kiện: Khi học sinh  $s_1$  gửi thông điệp nào đó cho người truyền tin  $s_2 = \varphi(s_1)$ , học sinh  $s_2$  sẽ lại gửi tiếp cho học sinh  $s_3 = \varphi(s_2)$ , ... cứ như vậy thông điệp sẽ đến được mọi người trong lớp và cuối cùng quay trở về người ban đầu  $s_1$ . Có nghĩa là khi một người gửi đi một thông điệp, anh ta chỉ cần đợi tới khi thông điệp đó quay trở về là yên tâm thông điệp đó đã đến với tất cả mọi người trong lớp.

Tuy nhiên không phải dây chuyền thông báo nào cũng là tốt. Bài toán đặt là cho trước một dây chuyền thông báo, hãy tìm cách yêu cầu một số ít học sinh thay đổi người truyền tin của mình để được một dây chuyền thông báo tốt.

**Input:** Tệp văn bản CIRCLE.INP:

- Dòng 1 chứa số nguyên dương  $n \leq 10^6$ .
- Dòng 2 chứa  $n$  số nguyên, số thứ  $i$  là  $\varphi(i)$ .

**Output:** Tệp văn bản CIRCLE.OUT

- Dòng 1: Ghi số học sinh cần phải thay đổi người truyền tin  $k$ .
- $k$  dòng tiếp theo, mỗi dòng ghi chỉ số của một học sinh cần thay đổi người truyền tin và chỉ số người truyền tin mới của người đó.

Các số trên một dòng phải ghi cách nhau ít nhất một dấu cách.

**Ví dụ:**

CIRCLE.INP	CIRCLE.OUT
6	
2 3 1 5 6 4	1 4 6 2

### 5.37. Hệ thống gần hoàn hảo

Một hệ thống  $S$  gồm  $m$  máy biến đổi số được đánh số từ 1 tới  $m$ . Hệ thống thực hiện phép biến đổi trên tập các số nguyên dương từ 1 tới  $n$ . Hoạt động của máy  $i$  được xác định bởi cặp số nguyên dương  $(a_i, b_i)$  ( $1 \leq a_i, b_i < n$ ). Máy nhận đầu vào là số nguyên dương  $a_i$  và trả ở đầu ra số nguyên dương  $b_i$ . Như vậy hệ thống  $S$  được mô tả bởi hai dãy số  $A = (a_1, a_2, \dots, a_n)$  và  $B = (b_1, b_2, \dots, b_m)$ .

Ta nói một số nguyên dương  $x$  có thể biến đổi thành số nguyên dương  $y$  nếu  $x = y$  hoặc tồn tại một dãy hữu hạn các số nguyên dương  $x = p_1, p_2, \dots, p_k = y$  sao cho đổi với hai phần tử liên tiếp  $p_i, p_{i+1}$  bất kì trong dãy, luôn tìm được một trong số các máy đã cho để biến đổi  $p_i$  thành  $p_{i+1}$ .

Hệ thống  $S$  được gọi là gần hoàn hảo nếu với hai số  $a, b$  bất kì thuộc tập  $A \cup B$ , hoặc  $a$  có thể biến đổi về  $b$ , hoặc  $b$  có thể biến đổi về  $a$ . Ở đây  $A \cup B$  là kí hiệu tập các phần tử thuộc dãy  $A$  hoặc dãy  $B$ .

**Yêu cầu:** Hãy kiểm tra xem hệ thống  $S$  cho trước có phải là gần hoàn hảo hay không?

**Input:** Tệp văn bản SPERFECT.INP chứa một số bộ dữ liệu:

- Dòng đầu tiên chứa số nguyên dương  $q$  là số bộ dữ liệu.
- Tiếp theo là  $q$  nhóm dòng mô tả các bộ dữ liệu:
  - Dòng đầu tiên trong nhóm chứa hai số nguyên dương  $n, m$  ( $1 \leq n, m \leq 10^5$ ).
  - $m$  dòng tiếp theo trong nhóm, mỗi dòng chứa một cặp số tương ứng với một máy biến đổi số.

**Output:** Ghi ra  $q$  dòng của tệp văn bản SPERFECT.OUT: dòng thứ  $i$  (tương ứng với bộ dữ liệu thứ  $i$  trong tệp dữ liệu vào) chứa thông báo “YES”, nếu hệ thống  $S$  trong bộ dữ liệu tương ứng là gần hoàn hảo và thông báo “NO” nếu trái lại.

**Ví dụ:**

SPERFECT.INP	SPERFECT.OUT
2	YES
6 3	NO
1 3	
2 3	
3 1	
6 2	
1 3	
2 3	

## Phần III

# Hướng dẫn giải bài tập

### Chuyên đề 1

1.1.  $O(n)$ .

1.2.  $O(n)$ .

1.3.  $O(n^2)$ .

1.4.  $O(n)$ .

1.5.  $O(n)$ .

1.6.  $O(n)$ .

1.7.  $O(n^2)$ .

1.8.  $O(n^3)$ .

1.9.  $O(\log n)$ .

1.10. a) Thủ tất cả các dãy con liên tiếp bắt đầu tại vị trí thứ  $i$  và kết thúc ở vị trí thứ  $j$  ( $1 \leq i \leq j \leq n$ , có  $\frac{n(n+1)}{2}$  dãy con). Với mỗi dãy con tiến hành tính tổng các phần tử của dãy để kiểm tra tổng có bằng  $k$  hay không? Độ phức tạp phần tính tổng là  $O(n)$ . Như vậy, độ phức tạp của thuật toán là  $O(n^3)$ .

b) Tối ưu việc tính tổng của một dãy con, từ độ phức tạp  $O(n)$  về còn  $O(1)$ . Cụ thể, để tính nhanh tổng các phần tử từ vị trí  $i$  đến vị trí  $j$  của mảng  $a_1, a_2, \dots, a_n$  ta làm như sau:

- Chuẩn bị trước mảng  $\text{Sum}[0..n]$  với độ phức tạp  $O(n)$ , trong đó  $\text{Sum}[i]$  là tổng các phần tử từ 1 đến  $i$ .

```
Sum[0] := 0;  
for i:=1 to n do Sum[i]:= Sum[i - 1] + a[i];
```

- Tổng một dãy con từ vị trí  $i$  đến vị trí  $j$  được tính bằng  $\text{Sum}[j] - \text{Sum}[i - 1]$ . Như vậy, độ phức tạp của thuật toán là  $O(n^2)$ .  
c) Với một vị trí  $i$ , xác định vị trí  $j$  mà tổng các phần tử từ  $i$  đến  $j$  lớn hơn hoặc bằng  $k$  ( $\text{Sum}[j] - \text{Sum}[i - 1] \geq k$ ).

```

jp:=1;
for i:=1 to n do
for j:=jp to n do begin
if Sum[j]-Sum[i-1]=k then begin writeln(i,' ',j); exit; end;
if Sum[j]-Sum[i-1]>k then begin
jp:=j;
break;
end;
end;

```

Thuật toán có độ phức tạp là  $O(n)$ .

## Chuyên đề 2

**2.1.** Thêm các kí tự 0 vào đầu xâu s để số lượng kí tự của xâu s là bội của 4. Đi từ đầu xâu s về cuối, nhóm 4 kí tự làm một nhóm, chuyển 4 kí tự đó sang sang kí tự ở hệ cơ số 16, cụ thể:

0000 → 0	1000 → 8
0001 → 1	1001 → 9
0010 → 2	1010 → A
0011 → 3	1011 → B
0100 → 4	1100 → C
0101 → 5	1101 → D
0110 → 6	1110 → E
0111 → 7	1111 → F

**2.2. a)** Phân tích  $N$  thành thừa số nguyên tố

```

i:=2;
while i*i <=N do
begin
  while N mod i=0 do
    begin
      write(i, ' ');
      N:=N div i;
    end;
    inc(i);
  end;
if N>1 then write(N);

```

**b)** Sau khi phân tích  $N$  thành thừa số nguyên tố, sử dụng công thức tính số ước (mục III.1- Tài liệu chuyên Tin học, Quyển 1).

c) Sử dụng công thức tính tổng các ước (mục III.2- Tài liệu chuyên Tin học, Quyển 1).

**2.3. Bước 1:** Xây dựng hàm kiểm tra số nguyên tố theo Fermat, sử dụng công thức:  $(A \times B) \bmod C = ((A \bmod C) \times (B \bmod C)) \bmod C$ .

**Bước 2:** Sử dụng sàng số nguyên tố để liệt kê được các số nguyên tố từ 1 đến  $10^6$ .

**Bước 3:** So khớp, kiểm tra lần lượt từng số từ 1 đến  $10^6$ .

Xem thêm bài 2.8.

**2.5.** Một số là số đẹp nếu số đó chia hết cho 9 (tổng các chữ số chia hết cho 9).

```
num:string; sum:longint;  
  
sum:=0;  
for i:=1 to length(num) do sum:=sum + ord(num[i])-48;  
if sum mod 9 = 0 then writeln('So dep') else writeln('khong  
phai so dep');
```

**2.6** Xử lí dấu trước rồi gọi đến các hàm xử lí số lớn không dấu, ví dụ như hàm cộng hai số lớn có dấu có thể thực hiện như dưới đây:

```
type bigNum = record  
    sign : longint; //sign = 1, nếu là số dương; sign = -1, nếu là số âm;  
    num : string;  
end;  
  
function add(a,b:bigNum):bigNum;  
var c:bigNum;  
begin  
    if a.sign = a.sign then begin  
        c.sign = a.sign;  
        c.num:= cong_hai_so_lon_khong_dau(a.num, b.num);  
    end  
    else begin  
        if so_sanh_lon_hon(a.num, b.num) then begin  
            c.sign:=a.sign;  
            c.num:=tru_hai_so_lon_khong_dau(a.num, b.num);  
        end  
        else begin  
            c.sign:=b.sign;  
            c.num:=tru_hai_so_lon_khong_dau(b.num, a.num);  
        end;  
    end;  
end;
```

**2.7.** Xem các phép toán trong bài 2.23.

Tin học,

sử dụng

đến  $10^6$

hỗn 9).

không

hư hàm

## 2.8.

```
M, N, K :longint;
du, cs : int64;
kq : string;
cs:=1;
for i:=1 to K do cs:=cs * 10;
du:=1;
for i:=1 to N do du:=(du * M) mod cs;
str(du,kq);
while length(kq)<K do kq:='0'+kq;
writeln(kq);
```

## 2.9. Tính ước số chung lớn nhất (USCLN) theo công thức sau:

$$\text{USCLN}(a_1, a_2, \dots, a_n) = \text{USCLN}(\text{USCLN}(a_1, a_2, \dots, a_{n-1}), a_n)$$

```
uc:=a[1];
for i:=2 to n do uc:=USCLN(uc,a[i]);
writeln(uc);
```

## Tính bội số chung nhỏ nhất (BSCNN) theo công thức sau:

$$\text{BSCNN}(a_1, a_2, \dots, a_n) = \text{BSCNN}(\text{BSCNN}(a_1, a_2, \dots, a_{n-1}), a_n)$$

```
bc:=1;
for i:=1 to n do bc:=bc * (a[i] div USCLN(bc,a[i]));
writeln(bc);
```

Chú ý: Do bội số có thể rất lớn nên phải sử dụng phép nhân số lớn với số nhỏ và phần tính  $\text{USCLN}(bc, a[i])$  là giữa số lớn ( $bc$ ) và số nhỏ ( $a[i]$ ). Tuy nhiên, việc tính  $\text{USCLN}(bc, a[i])$  cũng không phức tạp, chỉ sau một lần chia  $bc$  cho  $a[i]$  thì đã quy được về tính ước số chung của hai số nhỏ.

## Chương trình hoàn chỉnh

```
const fi = 'usbs.inp';
fo = 'usbs.out';
maxn = 20;
type bigNum =string;
var a :array[1..maxn]of longint;
n :longint;
uc :longint;
bc :bigNum;
procedure docf;
Var i:longint;
Begin
read(n);
```

```

for i:=1 to n do read(a[i]);
End;
function USCLN(a,b:longint):longint;
var tmp :longint;
begin
while b>0 do begin
  a:=a mod b;
  tmp:=a; a:=b; b:=tmp;
end;
  exit(a);
end;
procedure tim_uoc;
Var i :longint;
Begin
  uc:=a[1];
for i:=2 to n do uc:=USCLN(uc,a[i]);
writeln(uc);
End;
function multiply1(a:bigNum;b:int64):bigNum;
var i :integer;
  carry,s :int64;
  c,tmp :bigNum;
begin
  c:='';
  carry:=0;
for i:=length(a) downto 1 do
  begin
    s:=(ord(a[i])-48) * b + carry;
    carry:= s div 10;
    c:=chr(s mod 10 + 48)+c;
  end;
if carry>0 then str(carry,tmp) else tmp:='';
  multiply1:=tmp+c;
end;
function bigMod1(a:bigNum;b:int64):int64;
var i :longint;
  hold :int64;
begin
  hold:=0;
for i:=1 to length(a) do
  hold:=(ord(a[i])-48+hold*10) mod b;
  bigMod1:=hold;

```

```

end;
procedure tim_boi;
var i :longint;
    tmp:int64;
Begin
    bc:='1';
    for i:=1 to n do begin
        tmp:=bigMod1(bc,a[i]);
        tmp:=USCLN(tmp,a[i]);
        bc:=multiply1(bc,a[i] div tmp);
    end;
    writeln(bc);
End;
BEGIN
    assign(input,fi);assign(output,fo);
    reset(input);rewrite(output);
    docf;
    tim_uoc;
    tim_boi;
    close(input);close(output);
END.

```

**2.10.** Xây dựng hàm cộng hai số lớn, so sánh hai số lớn rồi tạo ra tất cả các số Fibonacci không vượt quá  $B$ , đếm số lượng số nhỏ hơn  $A$  để tính số lượng số Fibonacci trong đoạn  $[A, B]$ .

**2.11.** *Bước 1:* Tạo ra dãy số FIB gồm các số không vượt quá  $N$ .

*Bước 2:* Lặp lại bước 2.1 và 2.2 cho đến khi  $N = 0$ .

*Bước 2.1:* Tìm số  $FIB[i]$  lớn nhất nhỏ hơn  $N$ .

*Bước 2.2:* Trừ  $N$  đi  $FIB[i]$ .

**2.12.** Phân tích  $N!$  thành thừa số nguyên tố, cụ thể:

$N! = 2^x \times 5^y \times p_1^{k_1} \times p_2^{k_2} \times \dots \times p_m^{k_m}$  ( $p_1, p_2, \dots, p_m$  là các thừa số nguyên tố khác 2 và 5). Các thừa số  $p_1^{k_1} \times p_2^{k_2} \times \dots \times p_m^{k_m}$  không cần quan tâm, số lượng chữ số 0 tận cùng của  $N!$  chỉ phụ thuộc vào  $y$  (vì  $x \geq y$ ).

$$\text{Tính } y = \left[ \frac{N}{5^1} \right] + \left[ \frac{N}{5^2} \right] + \left[ \frac{N}{5^3} \right] + \dots$$

```

dem:=0;
lt5:=1;
while lt5*5<=n do begin
    lt5:=lt5*5;
    dem:=dem + n div lt5;
end;
writeln(dem);

```

**2.13.** Các kí tự 0, 1, 2, 3; 4, 5, 6, 7, 8, 9, A, B, C, D, E, F tương ứng có giá trị 0, 1, 2, .., 16.

Xây dựng hàm nhân, chia và hàm cộng số lớn với số nhỏ để chuyển số từ hệ cơ số  $a$  về hệ cơ số 10, rồi chuyển số từ hệ cơ số 10 về hệ cơ số  $b$ .

**2.14. Cách 1:** Tính căn bậc hai để kiểm tra số  $N$  có phải số chính phương không.

Ví dụ, có thể tính  $x(k)$  như sau:  $x(0) = 1$ ;  $x(k) = \frac{\frac{N}{x(k-1)} + x(k-1)}{2}$ .

Dãy  $x(k)$  hội tụ về căn bậc hai của  $N$ . Thực hiện theo cách này, ta phải xây dựng hàm xử lí số lớn như: chia số lớn cho số lớn, cộng số lớn với số lớn, so sánh hai số lớn.

*Cách 2:* Có thể tránh việc xây dựng hàm chia số lớn cho số lớn bằng cách tìm kiếm nhị phân như sau:

```

laSoChinhPhuong:=false;
min:=0; max:=N;
while (min<=max) do begin
    mid:=(max + min) div 2;
    if mid * mid = N then begin
        laSoChinhPhuong:=true;
        break;
    end;
    if mid * mid > N then max:= mid - 1
    else min:= mid + 1;
end;

```

Thực hiện theo cách này, ta phải xây dựng hàm xử lí số lớn như: nhân số lớn cho số lớn, cộng số lớn với số lớn, chia số lớn cho 2, cộng số lớn với 1, trừ số lớn đi 1, so sánh hai số lớn.

tri  
hệ  
ng  
y  
o  
n

2.15. *Cách 1:* tính  $C_n^k$  theo công thức  $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ . Thực hiện theo cách này, ta phải xây dựng hàm cộng hai số lớn. Thời gian tính toán mất:  $n^2$  nhân với thời gian xử lí số lớn.

*Cách 2:* tính  $C_n^k$  theo công thức  $C_n^k = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k+1)}{k!}$  bằng

cách phân tích tử số và mẫu số thành thừa số nguyên tố, rồi rút gọn. Cuối cùng nhận các thừa số còn lại với nhau. Thực hiện theo cách này, ta phải xây dựng hàm xử lí nhân số lớn với một số nhỏ. Thời gian tính toán bằng thời gian phân tích tử số, mẫu số thành thừa số nguyên tố ( $n\sqrt{n}$ ) cộng với thời gian nhân các thừa số còn lại ( $n$  nhân với thời gian xử lí số lớn).

2.16. Sử dụng công thức dưới đây để tính bằng cách phân tích tử số và mẫu số thành thừa số nguyên tố, rút gọn rồi nhân các thừa số của tử số.

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

2.17. Số cách đặt  $k$  quân xe lên bàn cờ  $n \times n$  sao cho không có quân nào ăn được nhau bằng:  $C_n^k \times C_n^k \times k!$

2.18. Theo định nghĩa, ta có  $M = N + S(N)$ , trong đó  $S(N)$  là tổng các chữ số của  $N$ . Nhận thấy  $S(N) < 9 \times$  số chữ số của  $M$ , do đó thử  $S(N)$  để tìm  $N$ .

```
const MAX = 100;
fi ='gen.in';
fo ='gen.out';

var s,kq,c :string;
f,g :text;
t,n :integer;

function tru(x,y:string):string;
var p :string;
nho,hieu,i :integer;
begin
p:='';
while length(x)<length(y) do x:='0'+x;
while length(y)<length(x). do y:='0'+y;

if (x>=y) then begin
nho:=0;
for i:=length(x) downto 1 do
begin
```

```

hieu:=ord(x[i])-ord(y[i])-nho;
if hieu<0 then begin hieu:=hieu+10; nho:=1;end
else nho:=0;
p:=chr(hieu+48)+p;
end;
while (length(p)>1)and(p[1]='0') do delete(p,1,1);
end;
tru:=p;
end;
procedure tim;
var i,j,w :integer;
begin
for i:=9*length(s) downto 0 do
begin
str(i,c);
kq:=tru(s,c);
if kq<>'' then begin
w:=0;
for j:=1 to length(kq) do w:=w+ord(kq[j])-48;
if w=i then exit;
end;
end;
kq:='0';
end;
BEGIN
assign(f,fi); reset(f);
assign(g,fo); rewrite(g);
readln(f,n);
for t:=1 to n do begin
readln(f,s);
tim;
writeln(g,kq);
end;
close(f); close(g);
END.

```

**2.19.** Phân tích  $N!$  thành thừa số nguyên tố rồi áp dụng công thức tính số ước và tổng ước.

**2.20.** Biểu diễn  $N$  ở hệ cơ số 3. Do mỗi loại chỉ có 1 quả cân, nếu trong biểu diễn của  $N$  ở hệ cơ số 3 có số 2 thì chỉ cần cộng 1 vào quả kê tiếp rồi đặt quả cân này sang đĩa bên kia.

**2.21.** Gọi  $F(n)$  là số xâu có độ dài  $n$ , ta có:

$$F(1) = 2, F(2) = 3, F(3) = 5, F(4) = 8, F(5) = 13, \dots$$

$$F(n) = F(n - 1) + F(n - 2)$$

2.22. Số hoán vị khác nhau của xâu s là

$$\frac{\text{length}(s)!}{(\text{số lượng kí tự } a)! \dots (\text{số lượng kí tự } z)!}$$

```
procedure tinh;
var i,j :longint;
begin
kq:='1';
for i:=1 to n do kq:= multiply1(kq,i);
for i:=1 to 26 do
for j:=1 to d[i] do
kq:=bigDiv1(kq,j);
writeln(kq);
end;
```

2.23.

```
uses math;
const nmax = 101;
      base = trunc(1e7);
type
  bigNum = record
    sl:longint;
    h:Array[0..20] of longint;
  end;
var
  n:longint;s:string;
  res:array[0..9] of bigNum;
  F:Array[0..nmax,0..nmax] of bigNum;
function add(a,b:bigNum):bigNum;
var i,s,hold:longint;
  c:bigNum;
begin
  c.sl:=max(a.sl,b.sl);hold:=0;
  for i:=a.sl+1 to b.sl do a.h[i]:=0;
  for i:=b.sl+1 to a.sl do b.h[i]:=0;
  for i:=1 to c.sl do
    begin
      s:=a.h[i]+b.h[i]+hold;
      c.h[i]:=s mod base;
      hold:=s div base;
    end;
  if hold>0 then begin inc(c.sl);c.h[c.sl]:=hold; end;
  exit(c);
end;
function mul(a:bigNum;b:longint):bigNum;
var i,s,hold:longint;
  c:bigNum;
begin
```

```

c.sl:=a.sl;hold:=0;
for i:=1 to c.sl do
begin
  s:=a.h[i]*b+hold;
  c.h[i]:=s mod base;
  hold:=s div base;
end;
while hold>0 do
begin
  inc(c.sl);c.h[c.sl]:=hold mod base;hold:=hold
div base;
end;
exit(c);
end;
Procedure init;
var i,j:longint;
begin
  F[0][0].sl:=1;F[0][0].h[1]:=1;
  for i:=1 to n do for j:=0 to i do
  begin
    F[i][j]:=mul(F[i-1][j],9);
    if j>0 then F[i][j]:=add(F[i][j],F[i-1][j-1]);
  end;
end;
function Cal(digit,sum:longint):bigNum;
var i,j,cur,ch:longint;
  sol:bigNum;
begin
  sol.sl:=0;
  for i:=1 to n do
  begin
    cur:=ord(s[i])-48;
    for ch:=0 to cur-1 do
      if (ch=digit)and(sum>0) then sol:=add(sol,f[n-i][sum-1])
      else if ch<>digit then sol:=add(sol,f[n-i][sum]);
    sum:=sum-ord(cur=digit);
    if sum<0 then break;
  end;
  exit(sol);
end;
function cal0(digit,sum:longint):bigNum;
var i,j,cur,ch,ss:longint;
  sol:bigNum;
begin
  sol.sl:=0;ss:=sum;
  for i:=1 to n do
  begin
    cur:=ord(s[i])-48;
    for ch:=0 to cur-1 do if (ch>0)or(i>1) then
  
```

```

begin
  if (ch=digit)and(sum>0) then sol:=add(sol,f[n-i][sum-1])
  else if ch>digit then sol:=add(sol,f[n-i][sum]);
end;
sum:=sum-ord(cur=digit);
if sum<0 then break;
end;
sum:=ss;
for i:=2 to n do if n-i>=sum then Sol:=add(Sol,mul(f[n-i][sum],9));
exit(sol);
end;
Procedure printResult(s:bigNum);
var i,j,d:longint;
begin
  if s.s1=0 then
    begin
      writeln(0);
      exit;
    end;
  write(s.h[s.s1]);for i:=s.s1-1 downto 1 do
    if s.h[i]=0 then write('0000000')
    else begin
      j:=s.h[i];d:=0;while j>0 do
      begin
        inc(d);j:=j div 10;
      end;
      for j:=1 to 7-d do write('0');
      write(s.h[i]);
    end;
    writeln;
  end;
Procedure process;
var i,j,u,v:longint;
  sol:bigNum;
begin
  readln(s);n:=length(s);init;
  for i:=1 to 9 do for j:=1 to n do
    begin
      sol:=cal(i,j);
      res[i]:=add(res[i],mul(sol,j));
    end;
  for i:=1 to n do
    begin
      scl:=cal0(0,i);
      res[0]:=add(res[0],mul(sol,i));
    end;
  for i:=1 to n do
    begin
      j:=ord(s[i])-48;

```

```

        res[j]:=add(res[j], f[0][0]);
    end;
    for i:=0 to 9 do printResult(res[i]);
end;
begin
    assign(input, 'digits.inp'); reset(input);
    assign(output, 'digits.out'); rewrite(output);
    process;
end.

```

**2.24.** Đánh số các hàng từ 1 đến  $N$  theo chiều từ trên xuống, các số trên hàng  $i$ , đánh số từ 1 đến  $i$  theo chiều từ trái sang phải. Vị trí thứ  $j$  trên hàng  $i$  được gọi là vị trí  $(i, j)$ . Từ  $(i, j)$  có thể đi đến  $(i + 1, j)$  và  $(i + 1, j + 1)$ . Gọi  $lmax[i, j]$ ,  $lmin[i, j]$  tương ứng là giá trị tích lớn nhất, nhỏ nhất đạt được khi đi từ đỉnh  $(1, 1)$  đến vị trí  $(i, j)$ , hai giá trị này có thể chuyển nhãn cho  $(i + 1, j)$  và  $(i + 1, j + 1)$ . Thực hiện theo cách này, ta phải xây dựng hàm xử lý nhân số lớn với một số nhỏ và phải xử lý cả dấu.

```

const MAX =100;
fi ='TGS.INP';
fo ='TGS.OUT';
type bigNum =record
sign :longint;
num :string;
end;
var a :array[1..MAX,1..MAX]of longint;
lmax,lmin :array[0..1,1..MAX]of bigNum;
n,tmp :longint;
i,j :longint;
p1,p2,p3,p4 :bigNum;
function cmp(a,b : bigNum): integer;
begin
if a.sign<b.sign then exit(-1);
if a.sign>b.sign then exit(1);
while length(a.num)<length(b.num) do a.num:='0'+a.num;
while length(b.num)<length(a.num) do b.num:='0'+b.num;
if a.num = b.num then exit(0);
if a.num > b.num then exit(1*a.sign);
exit(-1*a.sign);
end;
procedure getMaxMin(a,b:bigNum; var ma, mi :bigNum);
begin
if cmp(a,b)=1 then begin ma:=a; mi:=b; end
else begin ma:=b; mi:=a; end;
end;
function multiply1(a:bigNum;b:longint):bigNum;
var i :integer;
carry,s :longint;

```

```

c,tmp :bigNum;
begin
if (b=0) or (a.num='0') then begin
    c.sign:=1; c.num:='0'; exit(c);
end;
c.num:=' ';
if b>0 then c.sign:=a.sign
else c.sign:=-a.sign;
b:=abs(b);
carry:=0;
for i:=length(a.num) downto 1 do
begin
    s:=(ord(a.num[i])-48) * b + carry;
    carry:= s div 10;
    c.num:=chr(s mod 10 + 48)+c.num;
end;
if carry>0 then str(carry,tmp.num) else tmp.num:=' ';
c.num:=tmp.num + c.num;
multiplyl:=c;
end;
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]);
    tmp:=a[1,1];
    str(abs(tmp),lmax[1,1].num);
    if tmp>=0 then lmax[1,1].sign:=1 else lmax[1,1].sign:=-1;
    lmin[1,1]:=lmax[1,1];
    for i:=2 to n do begin
        getMaxMin(multiplyl(lmax[(i-1) and 1,1],a[i,1]),
        multiplyl(lmin[(i-1) and 1,1],a[i,1]),
        lmax[i and 1,1],lmin[i and 1,1]);
        getMaxMin(multiplyl(lmax[(i-1) and 1,i-1],a[i,i]),
        multiplyl(lmin[(i-1) and 1,i-1],a[i,i]),
        lmax[i and 1,i],lmin[i and 1,i]);
        for j:=2 to i-1 do begin
            getMaxMin(multiplyl(lmax[(i-1) and 1,j-1],a[i,j]),
            multiplyl(lmin[(i-1) and 1,j-1],a[i,j]),
            p1, p2);
            getMaxMin(multiplyl(lmax[(i-1) and 1,j],a[i,j]),
            multiplyl(lmin[(i-1) and 1,j],a[i,j]),
            p3, p4);
            if cmp(p1,p3)=1 then lmax[i and 1,j]:=p1
            else lmax[i and 1,j]:=p3;
            if cmp(p2,p4)=-1 then lmin[i and 1,j]:=p2
            else lmin[i and 1,j]:=p4;
        end;
    end;

```

```

end;
p1:=lmax[n and 1,1];
for i:=2 to n do
if cmp(lmax[n and 1,i],p1)=1 then p1:=lmax[n and 1,i];
if p1.sign=-1 then write(' ');
writeln(p1.num);
close(input);
close(output);
END.

```

**2.25.** Gọi  $W$  là số đường đi khác nhau từ ô  $(1,1)$  đến ô  $(M,N)$ .

Gọi  $A$  là số đường đi khác nhau từ ô  $(1,1)$  đến ô  $(M,N)$  chỉ đi qua ô nấm loại 1.

Gọi  $B$  là số đường đi khác nhau từ ô  $(1,1)$  đến ô  $(M,N)$  chỉ đi qua ô nấm loại 2.

Gọi  $C$  là số đường đi khác nhau từ ô  $(1,1)$  đến ô  $(M,N)$  chỉ đi qua ô nấm loại 3.

Số đường đi khác nhau từ ô  $(1,1)$  đến ô  $(M,N)$  qua ít nhất hai loại nấm bằng

$$(W - A - B - C).$$

Đọc dữ liệu vào chuyển vào mảng kí tự  $A[1..M,1..N]$ , trong đó  $A[i,j] = '0'$  với hai ô  $(1,1)$  và  $(M,N)$ ;  $A[i,j] = '#'$  nếu là ô không được đi vào;  $A[i,j] = '1'$  hoặc '2' hoặc '3' tương ứng là ô có nấm loại 1 hoặc 2 hoặc 3.

```

function demSL(mask:string):bigNum; {đếm số lượng đường đi từ ô(1,1) đến ô(M,N) chỉ đi qua các ô xuất hiện trong mask}
begin
{Dùng mảng F[0..M,0..N] để tính số lượng đường đi, cụ thể F[i,j] là số lượng đường đi từ ô (1,1) đến ô (i,j) chỉ qua các ô xuất hiện trong mask; F[i,j] được khởi tạo bằng '0'}
F[0,1]:='1';
for i:=1 to M do
  for j:=1 to N do
    if pos(A[i,j],mask)>0 then
      F[i,j]:=cong(F[i-1,j],F[i,j-1]);
  exit(F[M,N]);
end;
writeln(demSL('0123')-demSL('01')-demSL('02')-demSL('03'));

```

**2.26.**

```

const max =1000000;
var us :array[1..max]of byte;
i,j :longint;
m,n,count :longint;
BEGIN
  readln(n,m);
  for i:=1 to m+1 do begin
    j:=0;
    while j+i<=n do begin
      j:=j+i;

```

```

us[j]:=us[j] xor 1;
// inc(us[j]);
end;
end;
for i:=1 to n do
if not odd(us[i]) then inc(count);
writeln(count);
END.

```

## 2.27. Số đẹp

Thuật toán:

- Duyệt từng số nguyên bắt đầu từ 1, tính tổng bình phương các chữ số và kiểm tra nếu tổng đó là số nguyên tố thì tăng biến đếm lên 1. Thực hiện cho đến khi biến đếm bằng  $n$ .
- Để tăng tốc chương trình có thể cải tiến khâu kiểm tra tổng bình phương các chữ số có nguyên tố hay không bằng cách chuẩn bị trước một mảng  $isPrime[0..9*9*10]$  of boolean, trong đó  $isPrime[x] = true$  nếu  $x$  là số nguyên tố. Như vậy việc kiểm tra tổng bình phương các chữ số có nguyên tố hay không chỉ mất  $O(1)$ .
- Có thể áp dụng thuật toán quy hoạch động để giải bài toán với kích thước lớn hơn ( $n \leq 10^{15}$ ).

## 2.28. Đầu giá

Cách 1: Duyệt từng số nguyên từ  $A$  đến  $B$ , với mỗi số kiểm tra tính chất nguyên tố và đối xứng, nếu thỏa mãn thì tăng biến đếm lên 1.

Chú ý: nên kiểm tra tính chất đối xứng trước, nếu thỏa mãn mới kiểm tra tính chất nguyên tố vì chi phí kiểm tra tính đối xứng nhỏ hơn chi phí kiểm tra nguyên tố, hơn nữa số lượng số đối xứng nhỏ hơn nhiều so với số lượng số nguyên tố. Tuy nhiên, cách giải này vẫn chưa chạy được giới hạn của đề bài.

Cách 2: Xây dựng các số đối xứng nằm trong khoảng  $[A; B]$ , bằng cách xây dựng một nửa số rồi dựng đối xứng. Với mỗi số dựng được trong khoảng  $[A; B]$  kiểm tra tính nguyên tố, nếu thỏa mãn thì tăng biến đếm lên 1.

## 2.29. PAIRS

Gọi  $H$  là hiệu số lớn với số nhỏ trong một nhóm, như vậy  $H$  đặc trưng cho một cách chia nhóm, ví dụ với  $n = 2$ , ta có hai cách chia nhóm:

Cách chia thứ nhất:  $\{1, 2\}, \{3, 4\}$  với  $H = 1$ .

Cách chia thứ hai:  $\{1, 3\}, \{2, 4\}$  với  $H = 2$ .

*Cách 1:* Duyệt  $H$  từ 1 đến  $2^n - 1$ , với mỗi  $H$  tiến hành chia nhóm, nếu thỏa mãn thì tăng biến đếm lên 1. Cách chia nhóm thực hiện như sau:

- Dùng mảng  $mark[1..2^n]$  of boolean để đánh dấu các số đã được xếp nhóm, cụ thể  $mark[x] = true$  nếu số  $x$  đã được xếp vào một nhóm nào đó. Mỗi lần thực hiện chia nhóm mảng  $mark$  được khởi tạo bằng  $false$ .
- Cho  $x$  chạy từ 1 đến  $2^n$ , nếu  $mark[x] = false$  (chưa được xếp vào nhóm) thì xếp  $x$  và  $x + H$  vào làm một nhóm, đánh dấu  $x$  và  $x + H$ . Chú ý rằng  $x + H$  phải nhỏ hơn  $2^n$ , nếu  $x + H > 2^n$  thì cách chia nhóm này không thỏa mãn.

*Cách 2:* Nhận xét  $H$  là ước của  $n$ , như vậy bản chất là tính số ước của  $n$ . Để chạy được  $10^6$  bộ kiểm thử với  $n \leq 10^6$  cần một thuật toán đếm số ước hiệu quả cho các số  $n \leq 10^6$ .

*Ý tưởng:* Phân tích  $n$  thành thừa số nguyên tố và áp dụng công thức tính số ước. Giả sử  $n = a^i \times b^j \times \dots \times c^k$  thì số ước bằng  $(i+1) \times (j+1) \times \dots \times (k+1)$ . Tuy nhiên, chi phí phân tích một số thành thừa số nguyên tố mất  $O(\sqrt{n})$ , cũng bằng chi phí tính số ước. Sử dụng sàng số nguyên tố để phân tích một số thành thừa số nguyên tố chỉ mất  $O(\log n)$ .

```
LIMIT := 1000000;
fillchar(Prime,sizeof(Prime),0);
for i:=2 to trunc(sqrt(LIMIT)) do
  if Prime[i]=0 then
    begin
      j:=i*i;
      while j<=LIMIT do
        begin
          Prime[j]:=i;
          j:=j+i;
        end;
    end;
```

*Chú ý:* Câu lệnh  $Prime[j]:=i$  chứ không phải là  $Prime[j]:=1$  như sàng nguyên tố. Việc gán  $Prime[j]:=i$  cho nhiều thông tin hơn, nếu  $Prime[x]:=0$  tức là số  $x$  là số nguyên tố, nếu không  $x$  là hợp số và có một thừa số nguyên tố là  $Prime[x]$ . Sử dụng mảng  $Prime$  để phân tích thừa số. Cụ thể, nếu  $x$  là hợp số thì  $x = Prime[x] \times x_1$ , tiếp tục phân tích  $x_1$ . Thuật toán này có độ phức tạp  $O(T \log n)$ , trong đó  $T$  là số bộ dữ liệu vào.

Có thể tiếp tục cải tiến thuật toán dựa trên nhận xét sau: Gọi  $n_1 = b^j \times \dots \times c^k$  thì  $n = a^i \times n_1$ , khi đó số ước của  $n$  bằng  $(i+1)$  nhân với số ước của  $n_1$ .

### 2.30. FDP

Thuật toán:

- \* *Bước 1:* Phân tích  $k$  thành thừa số nguyên tố  $k = p_1^{k_1} \times p_2^{k_2} \times \dots \times p_l^{k_l}$ .
- \* *Bước 2:* Phân tích  $n!$  thành các thừa số (chỉ quan tâm đến các thừa số  $p_1, p_2, \dots, p_l$ )  
$$n! = p_1^{n_1} \times p_2^{n_2} \times \dots \times p_l^{n_l}.$$
- \* *Bước 3:* Tìm  $m = \min \left\{ \left| \frac{n_1}{k_1} \right|, \dots, \left| \frac{n_l}{k_l} \right| \right\}$ .

Độ phức tạp:

*Bước 1:* mất chi phí  $O(\sqrt{k})$ .

*Bước 2:* mất chi phí  $O(n \log k)$ .

*Bước 3:* mất  $O(l)$  với  $l \leq \log k$ .

Cải tiến bước 2 theo cách dưới đây với độ phức tạp là  $O(\log n - \log k)$  thay cho  $O(n \log k)$ .

$$n_1 = \left\lfloor \frac{n}{p_1} \right\rfloor + \left\lfloor \frac{n}{p_1^2} \right\rfloor + \left\lfloor \frac{n}{p_1^3} \right\rfloor + \dots$$

### 2.31. Tổng hiệu

Cần định nghĩa số lớn có dấu, ví dụ một định nghĩa trong Pascal như sau:

```
bigNum = record
  sign: longint; {đầu nhận giá trị -1 hoặc 1 tương ứng là số âm hay số dương}
  num: string;
end;
```

*Bước 1:* Xây dựng hàm tính tổng hai số để tính  $2A = (A + B) + (A - B)$ ,

*Bước 2:* Xây dựng hàm chia số lớn cho số 2 để tính  $A = \frac{2A}{2}$ .

*Bước 3:* Tính  $B = A + B + (-A)$ .

### 2.32. Bội số chung nhỏ nhất

Gọi  $BCNN_i$  là bội chung nhỏ nhất của các số từ 1 đến  $i$ . Như vậy,

$$BCNN_i = BCNN_{i-1} \times \frac{i}{UCLN(BCNN_{i-1}, i)}$$

- Xây dựng hàm tính chia số lớn (số lớn  $BCNN_{i-1}$ ) cho số nhỏ ( $số i$ ) để lấy dư, sau lần chia đầu tiên này ta nhận được kết quả là số nhỏ. Việc tìm ước số chung lớn nhất tiếp theo thực hiện trên hai số nhỏ.
- Xây dựng hàm nhân số lớn  $BCNN_{i-1}$  với số nhỏ  $\frac{i}{UCLN(BCNN_{i-1}, i)}$ .

### 2.33. Thủ thuật minh

Thuật toán đầu tiên khi đọc đề là phân tích các số  $k_i$  và  $x_{ij}$  thành thừa số nguyên tố, tuy nhiên thuật toán này sẽ không thực hiện được vì các số quá lớn không quá  $10^{15}$ .

*Thuật toán 1:*

- *Bước 1:* Xây dựng hàm tính số lớn nhân số nhỏ để tính tích  $k = k_1 \times k_2 \times \dots \times k_m$ .

*Bước 2:* Xây dựng hàm chia số lớn cho số nhỏ để tính  $k$  chia cho  $x_{i1} \times x_{i2} \times \dots \times x_{im}$  bằng cách chia lần lượt cho từng thừa số  $x_{i1}, x_{i2}, \dots, x_{im}$ .

*Thuật toán 2:*

Để kiểm tra  $k_1 \times k_2 \times \dots \times k_m$  có chia hết cho  $x_{i1} \times x_{i2} \times \dots \times x_{im}$  hay không có thể thực hiện bằng cách đi rút gọn phân số sau:  $\frac{k_1 \times k_2 \times \dots \times k_m}{x_{i1} \times x_{i2} \times \dots \times x_{im}}$ .

Để rút gọn phân số ta đi tính ước số chung lớn nhất giữa mọi cặp  $k_i$  và  $x_{ij}$ .

### 2.34. Máy tính sinh học

Thuật toán tương tự như bài FDP, tuy nhiên có thể giải hiệu quả, đơn giản hơn bằng cách chỉ quan tâm đến luỹ thừa của thừa số lớn nhất.

Ví dụ,  $K = 10 = 2 \times 5$ , ta chỉ cần quan tâm đến luỹ thừa của 5.

### 2.35. Xâu nhị phân gần đối xứng

*Nhận xét:*

- Nếu  $n$  lẻ, mọi xâu nhị phân đều là xâu nhị phân gần đối xứng, như vậy bài toán từ số thứ tự từ điển tìm cấu hình là chuyển từ số thập phân sang số nhị phân; bài toán từ cấu hình tìm số thứ tự là chuyển từ số nhị phân sang số thập phân.

- Nếu  $n$  chẵn, bỏ bit cuối cùng, chuyển về trường hợp lẻ.

```

const
  fi ='NBS.INP';
  fo ='NBS.OUT';

type bigNum =string;

var n :longint;
  t :bigNum;
  s :ansistring;
  f,g :text;

procedure xuly1;
var i,j, sl0, tmp, nho :longint;
begin
//trừ t đi I
  nho:=1;
  for i:=length(t) downto 1 do begin
    tmp:=ord(t[i])-48-nho;
    if tmp<0 then begin
      tmp:=tmp + 10;
      nho:=1;
    end else nho:=0;
    t[i]:=char(tmp+48);
  end;
  s:='';
  for i:=1 to n do s:=s+'0';
  j:=n-1 + n mod 2;
  while t>'0' do begin
    s[j]:=char(ord(t[length(t)]) mod 2 + 48);
    dec(j);
    tmp:=0;
    for i:=1 to length(t) do begin
      tmp:=tmp * 10 + ord(t[i]) - 48;
      t[i]:=char(tmp div 2 + 48);
      tmp:=tmp mod 2;
    end;
    while (length(t)>1)and(t[1]='0') do delete(t,1,1);
  end;
  if n mod 2 = 0 then begin
    sl0:=0;
    for i:=1 to length(s)-1 do sl0:=sl0 + ord(s[i]='0');
    if sl0 mod 2 = 1 then s[n]:='0' else s[n]:='1';
  end;
  writeln(g,s);
  end;

procedure xuly2;
var res, i :longint;

```

```

begin
  if n mod 2 = 0 then n:=n-1;
  res:=0;
  for i:=1 to n do res:=(res * 2 + ord(s[i]='1')) mod 111539786;
  writeln(g, (res+1) mod 111539786);
end;

BEGIN
  assign(f,fi); reset(f);
  assign(g,fo); rewrite(g);
  readln(f,n);
  readln(f,t);
  xuly1;
  readln(f,s);
  xuly2;
  close(f);
  close(g);
END.

```

**2.36.** Phần tính số lượng cách xếp có thể phát biểu lại như sau: Có bao nhiêu hoán vị  $\partial$  của  $1, 2, \dots, n$  mà không tồn tại ba chỉ số  $1 \leq i < j < k \leq n$  để  $\partial[i] < \partial[j] < \partial[k]$ . Số lượng hoán vị như vậy tương ứng với số *Catalan*.

### Chuyên đề 3

**3.1. a)** Với một xâu họ và tên cần lấy ra được tên. Tên được lấy bằng cách đi từ cuối xâu về đầu gấp dấu cách đầu tiên thì dừng.

```

function layTen(hovaten:string):string;
var ten:string; i:longint;
begin
  ten:='';
  for i:=length(hovaten) downto 1 do
    if hovaten[i]<>' ' then ten:=hovaten[i]+ten
    else break;
  exit(ten);
end;

```

Để sắp xếp danh sách họ và tên theo thứ tự abc (uru tiên tên, họ, đệm) thì chỉ cần sắp xếp theo thứ tự: tên + họ và tên.

- b) Sử dụng hàm lấy tên ở trên để lấy ra hết tất cả các tên sau đó loại bỏ trùng lặp.
- c) Sắp xếp danh sách theo điểm giảm dần, rồi duyệt từ đầu danh sách về cuối danh sách để đánh thứ hạng.

Giả sử danh sách  $ds[1..n]$  of record  $hovaten:string$ ;  $diem:real$ ;  $hang:longint$ ; end; đã sắp xếp theo điểm giảm dần. Để đánh thứ hạng theo điểm (từ cao về thấp) thực hiện như sau:

```
thuHang:=0; diemChot:=11;
for i:=1 to n do
  if diemChot>ds[i].diem then begin
    thuHang:=thuHang + 1;
    ds[i].hang:=thuHang;
    diemChot:=ds[i].diem;
  end
  else ds[i].hang:=thuHang;
```

Lấy ra những thí sinh có thứ hạng 1, 2, 3.

- 3.2. a) Với mỗi chỉ số  $i$ , dùng thuật toán tìm kiếm nhị phân để tìm phần tử có giá trị  $-a_i$  trong  $a_{i+1}, \dots, a_n$ .
- b) Với mỗi cặp chỉ số  $i, j$  ( $i < j$ ), dùng thuật toán tìm kiếm nhị phân để tìm phần tử có giá trị  $-(a_i + a_j)$  trong  $a_{j+1}, \dots, a_n$ .
- c) Với mỗi chỉ số  $i$ , dùng thuật toán tìm kiếm tuyến tính thay cho thuật toán tìm kiếm nhị phân (ở phần a) như sau: Với một vị trí  $i$ , xác định vị trí  $j$  gần  $n$  nhất mà  $a_j \leq -a_i$ .

```
jp:=n;
for i:=1 to n do
  for j:=jp downto i+1 do begin
    if a[j]==-a[i] then
      begin writeln(i, ', ', j); exit; end;
    if a[j]<-a[i] then begin
      jp:=j;
      break;
    end;
  end;
```

- d) với mỗi cặp chỉ số  $i, j$  ( $i < j$ ), dùng thuật toán tìm kiếm tuyến tính thay cho thuật toán tìm kiếm nhị phân (ở phần b) như sau: Với một cặp vị trí  $i, j$ , xác định vị trí  $k$  gần  $n$  nhất mà  $a_k \leq -(a_i + a_j)$ .

3.3. Vì độ dài xâu  $s$  nhỏ (không quá 200) nên có thể làm trực tiếp như sau: xét từng loại độ dài, với một độ dài  $l$  liệt kê tất cả các xâu con liên tiếp có độ dài  $l$  rồi đếm số xâu khác nhau.

3.4. Từ  $N$  tạo ra xâu số  $M$ . Thực hiện  $K$  lần xoá như sau: Duyệt từ đầu xâu về cuối sẽ xoá kí tự thứ  $i$  ( $i < \text{length}(M)$  đầu tiên thoả mãn:  $M[i] < M[i + 1]$ ). Nếu

không tồn tại vị trí  $i$  như vậy thì xoá kí tự cuối cùng của xâu. Cách làm này có độ phức tạp là  $O(K.\text{length}(M))$ , có thể làm tốt hơn bằng cách sử dụng *stack* có độ phức tạp  $O(\text{length}(M))$ .

**3.5.** Vì  $N$  nhỏ (không quá 100) nên có thể làm trực tiếp như sau: sử dụng hai vòng lặp lồng nhau để liệt kê hết tất cả các phân số, rồi tối giản để cho vào tập  $F(N)$ .

```

F(N) := {0/1; 1/1}
for p:=1 to N-1 do
    for q:=p+1 to N do begin
        tối giản phân số p/q;
        thêm phân số p/q vào F(N);
    end;

```

Tiến hành sắp xếp các phân số, loại bỏ trùng lặp, đưa ra phân số thứ  $K$ .

**3.6.** Vì xâu  $s$  chỉ gồm các kí tự 'a' đến 'z' nên có thể sử dụng thuật toán đếm phân phôi bằng cách sử dụng mảng  $f['a'..'z']$ , trong đó  $f[c]$  là số lần xuất hiện của kí tự  $c$  trong xâu  $s$ .

**3.7.** Dùng chỉ số  $i$  (ban đầu được khởi tạo bằng 1) để quản lí dãy  $a_1, a_2, \dots, a_n$ ; chỉ số  $j$  (ban đầu được khởi tạo bằng 1) để quản lí dãy  $b_1, b_2, \dots, b_m$ . Thêm vào dãy  $a$  phần tử  $a_{n+1} = \infty$ . Thực hiện  $n+m$  lần như sau: Mỗi lần so sánh  $a_i$  với  $b_j$ , để chọn phần tử cho vào dãy  $c$ , nếu chọn phần tử ở dãy nào thì tăng chỉ số của dãy đó lên.

**3.8.** Sắp xếp dãy số tăng dần, xét hai trường hợp:

- $n$  lẻ:  $X = a[n \text{ div } 2 + 1];$
- $n$  chẵn:  $a[n \text{ div } 2] \leq x \leq a[n \text{ div } 2 + 1];$

**3.9.** Áp dụng cách làm của bài 3.8. Từ  $x_1, x_2, \dots, x_N$ , tìm toạ độ  $X_A$ ; từ  $y_1, y_2, \dots, y_N$ , tìm toạ độ  $Y_A$  vì  $X_A$  độc lập với  $Y_A$ .

**3.10.** Sắp xếp các đoạn số theo điểm đầu  $x_i$ , sau đó duyệt lần lượt từng đoạn một để tính tổng độ dài bị phủ.

Giả sử các đoạn  $(x_1, d_1), (x_2, d_2), \dots, (x_n, d_n)$  đã sắp xếp tăng dần theo  $x_i$ .

```

l:=d[1];
p:=x[1]+d[1];
for i:=2 to n do
    if p<x[i] then begin l:=l+d[i]; p:=x[i]+d[i]; end
    else if p<x[i]+d[i] then
        begin l:=l+x[i]+d[i]-p; p:=x[i]+d[i];
    end;
writeln(l);

```

1 này có  
stack có  
hai vòng  
N).

án đêm  
át hiện  
...,  $a_n$ ;  
Thêm  
sánh  $a_i$   
ing chỉ

$y_1, y_2,$   
g đoạn

3.11. Sắp xếp  $N$  điểm tăng dần theo toạ độ  $x$ , nếu toạ độ  $x$  bằng nhau thì ưu tiên theo toạ độ  $y$  tăng dần. Thủ tất cả các bộ 3 cặp điểm  $i < j < k$  tương ứng làm ba đỉnh A, B, D của một hình chữ nhật ABCD. Từ ba đỉnh A, B, D suy ra toạ độ của đỉnh C (nếu ABD là một tam giác vuông). Kiểm tra xem đỉnh C có tồn trong  $N$  điểm hay không? Nếu sử dụng thuật toán tìm kiếm nhị phân để kiểm tra sự tồn tại của điểm C thì thuật toán có độ phức tạp  $O(N^2 \log N)$ . Nếu tận dụng hạn chế  $|x_i|, |y_i| \leq 1000$  (dùng mảng đánh dấu  $dd[-1000..1000, -1000..1000]$ ,  $dd[x,y] = 1$  nếu tồn tại điểm  $x, y$ ) thì thuật toán có độ phức tạp  $O(N^3)$ .

3.12. *Bước 1:* Sắp xếp tăng dần dãy gồm  $2n$  phần tử sau:  $a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n$ . Chú ý trường hợp nếu  $a_i = b_j$  thì  $a_i$  được xếp trước  $b_j$ .

*Bước 2:* Khởi trị  $count := 0$ ; Lần lượt duyệt từ 1 đến  $2n$ , nếu là phần tử loại  $a$  thì tăng  $count$  lên 1, nếu là phần tử loại  $b$  thì giảm  $count$  đi 1. Giá trị  $count$  lớn nhất trong quá trình duyệt chính là đáp số.

3.13. *Bước 1:* Xây dựng mảng  $S$ :

$$S[0] = 0; S[1] = a[1]; \dots; S[i] = \sum_{j=1}^i a_j; \dots; S[n] = \sum_{j=1}^n a_j.$$

Nếu đoạn con liên tiếp từ  $i$  đến  $j$  có tổng bằng 0 thì  $S[i-1] = S[j]$ .

*Bước 2:* Dễ dàng tìm dãy con liên tiếp có tổng bằng 0 nếu dãy  $S$  đã được sắp xếp.

3.14.

```
const inp ='ESEQ.INP';
out ='ESEQ.OUT';
maxn = 100001;
var fi,fo :text;
n :longint;
a :array[0..maxn+1] of longint;
l,r,c :array[0..maxn+1] of int64;
procedure openfile;
begin
  assign(fi,inp); reset(fi);
  assign(fo,out); rewrite(fo);
end;
procedure enter;
var i:longint;
begin
  readln(fi,n);
  for i:=1 to n do
```

```

read(fi,a[i]);
end;

procedure sort(left,right:longint);
var i,j:longint;
key,tg:int64;
begin
i:=left; j:=right;
key:=r[i+random(j-i+1)];
repeat
while r[i] < key do inc(i);
while r[j] > key do dec(j);
if i<= j then
begin
tg:=r[i]; r[i]:=r[j]; r[j]:=tg;
inc(i); dec(j);
end;
until i> j;
if i < right then sort(i,right);
if j > left then sort(left,j);
end;

procedure init; .
var i:longint;
begin
l[0]:=0;
for i:=1 to n do l[i]:=l[i-1] + a[i];
r[n+1]:=0;
for i:=n downto 1 do r[i]:=r[i+1] + a[i];
sort(l,n);
end;

function find(x:int64):longint;
var first,last,mid, res:longint;
begin
res:=-1;
first:=1; last:=n;
while first<=last do begin
mid:=(first+last) shr 1;
if r[mid] = x then begin res:=mid; last:=mid-1; end
else if r[mid] < x then first:=mid+1
else last:=mid-1;
end;
exit(res);
end;

procedure solve;
var i,j:longint;
res, x :int64;

```

```

begin
  fillchar(c,sizeof(c),0);
  for i:=1 to n do begin
    j:=find(l[i]);
    if j>-1 then inc(c[j]);
  end;
  res:=0;
  x:=0;
  for i:=n downto 1 do
  begin
    j:=find(l[i]);
    if j>-1 then dec(c[j]);
    x:= x + a[i];
    j:=find(x);
    res := res + c[j];
  end;
  writeln(fo,res);
end;

procedure closefile;
begin
  close(fi);
  close(fo);
end;

begin
  openfile;
  enter;
  init;
  solve;
  closefile;
end.

```

**3.15.** Sắp xếp dãy  $a_1, a_2, \dots, a_n$  theo trật tự:  $a_i$  được xếp trước  $a_j$  nếu  $a_i$  ghép với  $a_j$  lớn hơn  $a_j$  ghép với  $a_i$ .

**3.16. Bước 1:** Sắp xếp từng hàng của bảng  $A$  theo thứ tự tăng dần;

**Bước 2:** Duyệt tất cả các cặp hàng, với hai hàng đã được sắp tăng  $A_{i1}, A_{i2}, \dots, A_{in}; A_{j1}, A_{j2}, \dots, A_{jn}$ , trộn hai hàng thành một dãy tăng (bài 3.7) và tìm  $X_{ij}$  (bài 3.8). Tuy nhiên, có thể áp dụng thuật toán tìm kiếm nhị phân để tìm  $X_{ij}$  chỉ trong  $O(\log n)$  như chương trình dưới đây.

```

const max =1000;
fi ='wmt.inp';
fo ='wmt.out';
var a :array[1..max,0..max+1]of longint;
sum1,sum2 :array[1..max,0..max+1]of int64;

```

```

m,n :longint;
i,j :longint;
f :text;
res :int64;

function tim(i,j:longint):longint;
var di,ci,mi,mj :longint;
begin
di:=1; ci:=n;
while di<=ci do begin
mi:=(di+ci) shr 1;
mj:=n-mi+1;
if (a[i,mi]>=a[j,mj-1]) and (a[i,mi]<=a[j,mj]) then exit (a[i,mi])
else if (a[j,mj]>=a[i,mi-1]) and (a[j,mj]<=a[i,mi]) then exit (a[j,mj])
else if a[i,mi]>a[j,mj] then ci:=mi-1
else di:=mi+1;
end;
end;

function chiadoi(k,x:longint):longint;
var dau, cuoi, mid :longint;
begin
dau:=0; cuoi:=n+1;
while dau<cuoi do
begin
mid:=(dau+cuoi+1) shr 1;
if a[k,mid]>x then cuoi:=mid-1
else dau:=mid;
end;
chiadoi:=dau;
end;

function tinh(i,j:longint):int64;
var p :longint;
temp,x :int64;
begin
temp:=0;
x:=tim(i,j);
p:=chiadoi(i,x);
temp:=x*p-sum1[i,p] - x*(n-p)+sum2[i,p+1];
p:=chiadoi(j,x);
temp:=temp + x*p-sum1[j,p] - x*(n-p)+sum2[j,p+1];
tinh:=temp;
end;

procedure QS(l,r:longint);
var ii,jj,k,tg:longint;
begin
if l>=r then exit;

```

```

k:=a[i, (l+r) shr 1];
ii:=l;
jj:=r;
repeat
  while (a[i,ii]<k) do inc(ii);
  while (a[i,jj]>k) do dec(jj);
  if ii<=jj then
begin
  tg:=a[i,ii]; a[i,ii]:=a[i,jj]; a[i,jj]:=tg;
  inc(ii);
  dec(jj);
end;
until ii>jj;
QS(l,jj);
QS(ii,r);
end;

BEGIN
  assign(f,fi); reset(f);
  readln(f,m,n);
  for i:=1 to m do begin
    for j:=1 to n do read(f,a[i,j]);
    a[i,0]:=-1; a[i,n+1]:=maxlongint;
  end;
  close(f);
  for i:=1 to m do qs(i,n);
  for i:=1 to m do begin
    sum1[i,0]:=0; sum2[i,n+1]:=0;
    for j:=1 to n do sum1[i,j]:=sum1[i,j-1]+a[i,j];
    for j:=n downto 1 do sum2[i,j]:=sum2[i,j+1]+a[i,j];
  end;
  for i:=1 to m-1 do
    for j:=i+1 to m do
      res:=res + tinh(i,j);
  assign(f,fo); rewrite(f);
  writeln(f,res);
  close(f);
END.

```

**3.17.** Dùng thuật toán đếm phân phối.

**3.18. Bước 1:** Sắp xếp dãy  $b$  và dãy  $c$  theo thứ tự tăng dần.

*Bước 2:* Điều khiển hai biến  $i, j$  (chạy  $i$  trên dãy  $b$  và  $j$  trên dãy  $c$ ) để tìm  $|b_i + c_j|$  đạt giá trị nhỏ nhất.

**3.19. Bước 1:** Tạo mảng  $S$  là tất cả các loại tổng  $a_i + a_j$  với  $i < j$ .

*Bước 2:* Sắp xếp mảng  $S$  tăng dần (có kèm chỉ số  $i, j$ ).

Bước 3: Đếm số lượng, với một số  $a_i$  bất kì ta kiểm tra xem nó có phải là số trung bình không.

```
const max =1000;
fi ='tbc.inp';
fo ='tbc.out';
var a :array[1..max]of longint;
w :array[1..max*max div 2]of longint;
id :array[1..max*max div 2,1..2]of longint;
n,i,j,m,p,dem,k :longint;
s :longint;

procedure trao(var x,y:longint);
var tg : longint;
begin
tg:=x; x:=y; y:=tg;
end;

procedure qs(dau,cuoi:longint);
var i,j,k :longint;
begin
i:=dau; j:=cuoi; k:=w[(i+j) div 2];
repeat
while w[i]<k do inc(i);
while w[j]>k do dec(j);
if i<=j then begin
trao(w[i],w[j]);
trao(id[i,1],id[j,1]);
trao(id[i,2],id[j,2]);
inc(i); dec(j);
end;
until i>j;
if dau<j then qs(dau,j);
if i<cuoi then qs(i,cuoi);
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-1 do
for j:=i+1 to n do
if a[i]>a[j] then trao(a[i],a[j]);
m:=0;
for i:=1 to n-1 do
for j:=i+1 to n do begin
inc(m);
w[m]:=a[i]+a[j];
```

```

id[m,1]:=i; id[m,2]:=j;
end;
qs(1,m);
dem:=0;
for p:=1 to n do begin
s:=a[p]; s:=s*3;
i:=m;
for j:=1 to n do
begin
for k:=i downto 1 do
if (w[k]<=s-a[j]) then break;
i:=k;
if w[k]=s-a[j] then begin
if (i>0) and (id[i,1]<>j) and (id[i,2]<>j) then begin
inc(dem);
//writeln(p);
break;
end;
if (i>1) and (w[i-1]=s-a[j]) and (id[i-1,1]<>j) and (id[i-1,2]<>j) then begin
inc(dem);
//writeln(p);
break;
end;
end;
end;
end;
write(dem);
close(input); close(output);
END.

```

**3.20. Nhận xét 1:** Dãy  $d$  sau từ sau phần tử thứ  $m + 1$  sẽ đi vào chu kì, chu kì có độ dài nhỏ hơn hoặc bằng  $m$ .

Mục tiêu là liệt kê các giá trị  $d[.]$  khác nhau và số lần xuất hiện của mỗi giá trị. Ví dụ, hai đoạn độ dài 1; bốn đoạn độ dài 2; ba đoạn độ dài 4;....

**Nhận xét 2:** Chỉ cần giữ lại ba đoạn độ dài  $L$  mà không ảnh hưởng tới số tam giác, như vậy nếu  $n > 4m$  thì  $n := 4m + 1$  ( $m + 1$  đoạn đầu tiên được giữ,  $3m$  đoạn tiếp theo cũng giữ, còn lại bỏ tất, chu kì chỉ cho lặp 3 lần là đủ).

Độ phức tạp:  $O(n^2)$ .

**3.21.** Khảo sát tìm ra chu kì của toàn bộ dãy, từ đó tính được chu kì cho đoạn  $u, v$ .

### 3.22. Tập số

Vì số  $n$  có tới 20 chữ số, do đó phải lưu trữ bằng xâu kí tự. Liệt kê tất cả các số nhận được bằng cách thử tất cả các cách xoá một hoặc một vài chữ số liên tiếp của  $n$  (nhưng không xoá hết tất cả các chữ số của  $n$ ). Lọc bỏ các số không chia hết cho 3, việc kiểm tra có chia hết cho 3 không thực hiện đơn giản bằng cách kiểm tra tổng các chữ số có chia hết cho 3 hay không. Sắp xếp các số tăng dần, dễ dàng đếm được số lượng số khác nhau.

Mở rộng: Thay đổi điều kiện “xoá một hoặc một vài chữ số *liền tiếp* của  $n$  (nhưng không xoá hết tất cả các chữ số của  $n$ )” thành “xoá một hoặc một vài chữ số của  $n$  (nhưng không xoá hết tất cả các chữ số của  $n$ )”. Bài toán mở rộng này khó hơn bài toán gốc trong việc liệt kê tất cả các số nhận được, còn việc kiểm tra số chia hết cho 3 và đếm số lượng số khác nhau không thay đổi. Để liệt kê tất cả các số nhận được, thực hiện bằng cách liệt kê tất cả các dãy nhị phân độ dài  $l$  (trong đó  $l$  là số lượng chữ số của  $n$ ). Mỗi dãy nhị phân tương ứng tạo được một số, kí tự thứ  $i$  của dãy nhị phân bằng 1 thì chữ số thứ  $i$  của  $n$  được giữ lại, bằng 0 nếu bị xoá đi.

### 3.23. Đường thẳng

Chia  $n$  đường thẳng thành hai loại:

- Loại 1: Các đường thẳng có dạng  $Ax + C = 0$  ( $B = 0$ ), chọn được một đường thẳng trong tập này.
- Loại 2: Các đường thẳng có dạng  $Ax + By + C = 0$  ( $B \neq 0$ ), sắp xếp các đường thẳng theo hệ số góc để đếm số lượng giá trị khác nhau, số lượng giá trị khác nhau chính là số lượng đường thẳng được chọn trong tập này.

Độ phức tạp thuật toán:  $O(n \log n)$ .

### 3.24. Hình chữ nhật bốn màu

```
program ColorRects;
const InputFile = '';
      OutputFile = '';
      maxxy = 200;
type TList = record
  p, c: array[1..maxxy * 2 + 1] of Integer;
  num: Integer;
end;
var l: array[-maxxy..maxxy] of TList;
  n: Integer;
  res: Cardinal;
```

```

procedure Enter;
var f: TextFile;
x: Integer;
i: Integer;
begin
AssignFile(f, InputFile); Reset(f);
try
for x := -maxxy to maxx do l[x].num := 0;
ReadLn(f, n);
for i := 1 to n do
begin
Read(f, x);
with l[x] do
begin
Inc(num);
ReadLn(f, p[num], c[num]);
c[num] := 1 shl (c[num] - 1);
end;
end;
finally
CloseFile(f);
end;
end;

procedure Solve;
var count: array[%0000..%1111] of Integer;
mark: array[-maxxy .. maxx] of Byte;
i, k, j: Integer;
value: Integer;
begin
res := 0;
for i := -maxxy to maxx do
begin
FillChar(mark, SizeOf(mark), 0);
with l[i] do
for j := 1 to num do mark[p[j]] := c[j];
for k := i + 1 to maxx do
begin
FillChar(count, SizeOf(count), 0);
with l[k] do
for j := 1 to num do
begin
value := c[j] or mark[p[j]];
Inc(count[value]);
Inc(res, count[%1111 xor value]);
end;
end;
end;

```

```

end;

procedure PrintResult;
var f: TextFile;
begin
  AssignFile(f, OutputFile); Rewrite(f);
  try
    Write(f, res);
  finally
    CloseFile(f);
  end;
end;

begin
  Enter;
  Solve;
  PrintResult;
end.

```

### 3.25. Phần tử chung

Gọi  $n$  là tổng số phần tử. Từ mỗi phần tử trong mỗi dãy tạo ra bộ gồm hai thành phần: giá trị phần tử và chỉ số dãy, nhận được  $n$  bộ. Sắp xếp các bộ tăng dần theo giá trị phần tử, nếu giá trị phần tử bằng nhau thì sắp xếp theo chỉ số dãy (chi phí  $O(n\log n)$ ). Duyệt dãy đã sắp xếp, trong mỗi vùng giá trị bằng nhau kiểm soát xem có đủ  $k$  chỉ số khác nhau không, nếu có thì đó là giá trị cần tìm (chi phí  $O(n)$ ). Độ phức tạp thuật toán:  $O(n\log n)$ .

Trong ví dụ 1:

- Dãy 1 gồm 3 số 1 2 3, tạo ra 3 bộ (1,1), (2,1), (3,1).
- Dãy 2 gồm 4 số 4 3 2 -1, tạo ra 4 bộ (4,2), (3,2), (2,2), (-1,2).
- Sắp xếp các bộ: (-1,2), (1,1), (2,1), (2,2), (3,1), (3,2), (4,2).

Duyệt dãy đã sắp xếp, vùng giá trị 2 có đủ 2 chỉ số khác nhau, giá trị cần tìm là 2.

### 3.26. Xây dựng hàm tính thời gian từ thời điểm $t1$ đến $t2$ :

```

function tinhThoiGian(t1, t2:string):longint;
var p1, p2 :longint;
begin
  p1:=  ( (ord(t1[1])-48)*10 + (ord(t1[2])-48) ) * 60 * 60
        +((ord(t1[3])-48)*10 + (ord(t1[4])-48)) * 60
        +((ord(t1[5])-48)*10 + (ord(t1[6])-48));
  p2:=  ( (ord(t2[1])-48)*10 + (ord(t2[2])-48) ) * 60 * 60
        +((ord(t2[3])-48)*10 + (ord(t2[4])-48)) * 60
        +((ord(t2[5])-48)*10 + (ord(t2[6])-48));
end.

```

```

        +((ord(t2[3])-48)*10 + (ord(t2[4])-48)) * 60
        +((ord(t2[5])-48)*10 + (ord(t2[6])-48));
exit(p2-p1);
end;

```

Đánh số thứ tự các số điện thoại tương ứng từ 1 đến  $n$  để dễ quản lý;  
Xử lí lần lượt từng cuộc gọi.

## Chuyên đề 4

**4.1. Đọc danh sách học sinh vào mảng  $dsHS[1..n]$  of string.** Dùng mô hình đệ quy để sinh tất cả các tổ hợp chập  $k$  của  $n$ , với mỗi tổ hợp khi ghi ra thì ghi theo danh sách học sinh (xem thủ tục ghi nghiệm ở chương trình dưới đây).

```

const MAX =20;
type vector =array[0..MAX]of longint;
var x :vector;
n,k :longint;
dsHS :array[1..MAX]of string;
procedure GhiNghiem(x:vector);
var i :longint;
begin
for i:=1 to k do write(dsHS[x[i]],' ');
writeln;
end;
procedure ToHop(i:longint);
var j:longint;
begin
for j := x[i-1]+1 to n-k+i do begin
x[i] := j;
if i=k then GhiNghiem(x)
else ToHop(i+1);
end;
end;
procedure nhapDL;
var i :longint;
begin
write('Nhập n, k:'); readln(n,k);
for i:=1 to n do readln(dsHS[i]);
end;

BEGIN
nhapDL;
x[0]:=0;
ToHop(1);
END.

```

4.2. Dùng mô hình đệ quy để sinh tất cả chỉnh hợp lặp chap 2 của  $n$  phần tử chính là các dãy nhị phân độ dài  $n$  cần liệt kê.

4.3. Gọi  $n$  là độ dài của xâu  $S$ , dùng mô hình đệ quy để sinh tất cả chỉnh hợp không lặp chap  $k$  (với  $k = n$ ) của  $n$  phần tử chính là các hoán vị độ dài  $n$ . Tương tự như bài 4.1, khi ghi từng hoán vị thì ghi ra kí tự tương ứng.

4.4. Khai biến toàn cục  $s$  (kiểu xâu) được khởi tạo gồm  $n$  dấu cách; sửa đổi một chút trong mô hình đệ quy sinh tất cả chỉnh hợp lặp chap 2 của  $n$  phần như đoạn chương trình dưới đây.

```
procedure lietKe(i:longint);
var c:char;
begin
  if i>n then begin writeln(s); exit;end;
  for c:='A' to 'B' do
    if s[i-1]+s[i]<>'BB' then begin
      s[i]:=c;
      lietKe(i+1);
    end;
end;
```

4.5. Dùng mô hình đệ quy để sinh tất cả chỉnh hợp lặp chap  $k$  của  $n$  phần tử, với mỗi chỉnh hợp lặp chap  $k$  của  $n$  phần tử (tương ứng là một cách chia  $n$  phần tử thành  $k$  nhóm). Tiến hành kiểm tra có là cách chia thành  $k$  nhóm mà các nhóm có tổng bằng nhau hay không?

Hàm kiểm tra dưới đây kiểm tra một mỗi chỉnh hợp lặp chap  $k$  của  $n$  phần tử (lưu trữ trong mảng  $x$ ) có là cách chia thành  $k$  nhóm mà các nhóm có tổng bằng nhau hay không?

```
function laKnhom(x:mang):boolean;
var tong[1..MAXX] of longint; i:longint;
begin
  fillchar(tong,sizeof(tong),0);
  for i:=1 to n do tong[x[i]]:=tong[x[i]]+a[i];
  for i:=2 to k do
    if tong[i]<>tong[1] then exit(false);
  exit(true);
end;
```

4.6. Gọi  $n$  là độ dài của xâu  $S$ , dùng mô hình đệ quy để sinh tất cả các xâu nhị phân độ dài  $n$  (chỉnh hợp lặp chap 2 của  $n$ ). Với một xâu nhị phân độ dài  $n$ , kí tự thứ  $i$  bằng '1' nếu kí tự thứ  $i$  của xâu  $S$  được giữ lại, ngược lại kí tự thứ  $i$  bằng '0' nếu kí tự thứ  $i$  của xâu  $S$  bị xoá đi. Như vậy, một xâu nhị phân tương ứng với một xâu con của xâu  $S$ . Tuy nhiên, các xâu con này có thể xuất hiện

nhiều lần. Do đó lưu tất cả các xâu con (có  $2^n$  xâu con), sắp xếp các xâu theo thứ tự từ điển, loại bỏ trùng lặp, loại bỏ xâu rỗng để nhận được các xâu con khác nhau của  $S$ .

**4.7.** Trước tiên xây dựng một hàm kiểm tra một xâu có phải là một biểu thức ngoặc đúng hay không? Cách kiểm tra như sau: Đi từ đầu xâu về cuối xâu, duy trì một biến đếm số lượng dấu mở ngoặc trừ đi số lượng dấu đóng ngoặc tính từ trái sang phải đến vị trí hiện tại (ban đầu được khởi tạo bằng 0), nếu gặp kí tự '(' thì tăng biến đếm, còn nếu gặp kí tự ')' thì giảm biến đếm. Một biểu thức ngoặc đúng nếu biến đếm tại mọi vị trí đều không âm và cuối cùng biến đếm bằng 0.

```
function bieuThucNgoacDung(s:string):boolean;
var i, dem :longint;
begin
dem:=0;
for i:=1 to length(s) do
    if s[i]='(' then inc(dem)
    else begin
        dec(dem);
        if dem<0 then exit(false);
    end;
exit(dem=0);
end;
```

Dùng mô hình đệ quy để sinh tất cả các xâu nhị phân độ dài  $2n$  (chỉnh hợp lặp chap 2 của  $2n$ ). Với một xâu nhị phân độ dài  $2n$ , thay kí tự '0' bằng '(' và kí tự '1' thay bằng ')'. Như vậy, một xâu nhị phân tương ứng với một biểu thức ngoặc, tiến hành kiểm tra xem xâu có phải là biểu thức ngoặc đúng không? Nếu đúng đưa ra biểu thức ngoặc.

**4.8.** Một số nguyên dương  $m$  có thể phân tích được dưới dạng tổng của một số số trong  $a_1, a_2, \dots, a_n$  (mỗi số không quá một lần) tức là tồn tại một dãy nhị phân  $(x_1, x_2, \dots, x_n)$  thoả mãn:  $m = x_1 \times a_1 + x_2 \times a_2, \dots, x_n \times a_n$ .

Dùng mô hình đệ quy để sinh tất cả các dãy nhị phân độ dài  $n$ . Với một dãy nhị phân  $(x_1, x_2, \dots, x_n)$ , tính giá trị  $m = x_1 \times a_1 + x_2 \times a_2, \dots, x_n \times a_n$ , chỉ cần quan tâm đến các giá trị  $m < 2^n$ , đánh dấu giá trị  $m$  xuất hiện vào mảng đánh dấu  $dd[0..2^n]$ .

Sau khi duyệt xong, tìm số nguyên dương nhỏ nhất không được đánh dấu trong mảng đánh dấu  $dd$  chính là kết quả cần tìm.

4.9. Dùng mảng  $d['A'..'Z']$  để lưu số lượng kí tự của từng loại kí tự. Ở mỗi vị trí  $i$  lần lượt xét các khả năng nhận một trong các kí tự 'A' đến 'Z', kí tự  $c$  có thể được sử dụng nếu  $d[c] > 0$ , giảm đi 1 nếu sử dụng.

```
const fi = 'HV.inp';
fo = 'HV.out';
max = 20;
var d :array['A'..'Z']of longint;
s,p :string;
n,k :longint;
procedure duyet(i:longint);
var c: char;
begin
if i>n then
begin
writeln(p);
exit;
end;
for c:='A' to 'Z' do
if d[c]>0 then
begin
d[c]:=d[c]-1;
p:=p+c;
duyet(i+1);
delete(p,i,1);
d[j]:=d[c]+1;
end;
end;
BEGIN
assign(input,fi); reset(input);
assign(output,fo); rewrite(output);
read(s);
n:=length(s);
for k:=1 to n do inc(d[s[k]]);
p:='';
duyet(1);
close(input);
close(output);
END.
```

4.11. Xây dựng một hàm kiểm tra một số nguyên có phải là số nguyên tố hay không. Xây dựng một hàm đệ quy liệt kê các số siêu nguyên tố, bắt đầu từ vị trí 1, thử đặt các chữ số từ 0 đến 9, nếu tạo được số nguyên tố thì gọi đệ quy xây dựng vị trí tiếp theo cho đến vị trí thứ  $n$ .

```

procedure duyet(i:longint; so:longint);
var j:longint;
begin
  if i>n then writeln(so)
  else for j:=0 to 9 do
    if laSoNguyenTo(so*10+j) then duyet(i+1,so*10+j);
end;

```

Từ chương trình chính gọi: duyet(1,0).

4.12. Một xâu  $S$  độ dài  $n$ , ta có  $n$  khe để chèn dấu '+' hoặc dấu '-'. Dùng mô hình đệ quy để sinh tất cả các dãy chỉnh hợp lặp chap 3 của  $n$ . Với một dãy tương ứng với một cách chèn, vị trí thứ  $i$  bằng 0 tức là khe thứ  $i$  không chèn, bằng 1 tức là chèn dấu '+', bằng 2 tức là chèn dấu '-', tính tổng biểu thức nhận được và kiểm tra xem biểu thức có bằng  $M$  hay không.

```

type mang = array[1..10]of longint;
var s :string;
m, n :longint;
x :mang;

procedure ghi(x:mang);
var i :longint;
begin
  for i:=1 to n do begin
    if x[i]=1 then write('+')
    else if x[i]=2 then write('-');
    write(s[i]);
  end;
  writeln;
end;

function kiemTra(x:mang):boolean;
var i, tong, so, dau :longint;
begin
  tong:=0;
  so:=0;
  dau:=1;
  for i:=1 to n do
    if x[i]=0 then so:=so*10+ord(s[i])-48
    else if x[i]=1 then
      begin
        tong:=tong + dau * so;
        so:=ord(s[i])-48;
        dau:=1;
      end
end;

```

```

else begin
tong:=tong + dau * so;
so:=ord(s[i])-48;
dau:=-1;
end;
tong:=tong + dau * so;
kiemTra:=(tong = M);
end;

procedure duyet(i:longint);
var j :longint;
begin
if i>n then
begin
if kiemTra(x) then ghi(x);
exit;
end;
for j:=0 to 2 do
begin
x[i]:=j;
duyet(i+1);
end;
end;

BEGIN
M:=8;
S:='1234567';
n:=length(S);
duyet(1);
END.

```

4.13. Dùng mô hình đệ quy để sinh tất cả các bảng nhị phân kích thước  $n \times n$ .

Sử dụng mảng  $c1[2..MAX+MAX] of integer$  và  $c2[1-MAX..MAX-1] of integer$  để kiểm soát các quân cờ nằm trên đường chéo.

Có thể tăng tốc thuật toán trên nhận xét: Tô màu bàn cờ bằng hai màu đen và trắng. Ô  $(x, y)$  được tô màu đen nếu  $x + y$  lẻ, các ô còn lại tô màu trắng. Khi đó hai quân tượng đặt ở hai ô khác màu nhau sẽ không chiếu nhau. Duyệt để đếm số cách đặt  $l$  ( $l \in [0, k]$ ) quân tượng vào các ô đen, tương tự duyệt để đếm số cách đặt  $l$  ( $l \in [0, k]$ ) quân tượng vào các ô trắng. Kết quả sẽ là tổng của các tích: số cách đặt  $l$  quân tượng vào các ô đen nhân với số cách đặt  $k - l$  quân tượng vào các ô trắng.

```

const max = 10;
fi = 'bishops.inp';
fo = 'bishops.out';
var c1 : array[2..MAX+MAX]of integer;
c2 : array[1-MAX..MAX-1]of integer;
d : array[0..1,0..MAX*MAX]of int64;
NTest,k : integer;
n,min : integer;
f,g : text;

procedure duyet(i,x,y:integer);
var j : integer;
begin
if i>min then exit;
if (c1[x+y]=0)and(c2[x-y]=0) then
begin
c1[x+y]:=1;
c2[x-y]:=1;
inc(d[(x+y)mod 2,i]);
if y+2<=n then
duyet(i+1,x,y+2)
else if x<n then
duyet(i+1,x+1,2-(y+1) mod 2);
c1[x+y]:=0;
c2[x-y]:=0;
end;
if y+2<=n then duyet(i,x,y+2)
else if x<n then duyet(i,x+1,2-(y+1) mod 2);
end;

function count():int64;
var c : int64;
i : integer;
begin
min:=k;
if min>n then min:=n;
fillchar(d,sizeof(d),0);
d[0,0]:=1;
d[1,0]:=1;
fillchar(c1,sizeof(c1),0);
fillchar(c2,sizeof(c2),0);
duyet(1,1,1);
duyet(1,1,2);
c:=0;
for i:=0 to k do

```

```

c := c + d[1,i]*d[0,k-i];
count:=c;
end;

BEGIN
assign(f,fi); assign(g,fo);
reset(f); rewrite(g);
readln(f,NTest);
while NTest>0 do begin
dec(Ntest);
readln(f,n,k);
writeln(g,count());
end;
close(f); close(g);
END.

```

4.14. Chương trình dưới đây sẽ liệt kê tất cả các hình N-mino chưa xét đến vấn đề xoay và đối xứng hình. Dùng mô hình duyệt đệ quy để chọn  $k$  ô trong bảng  $n \times n$  với điều kiện: có ít nhất một ô ở hàng 1, có ít nhất một ô ở cột 1 và  $k$  ô liên thông với nhau.

```

const tx :array[1..4]of longint=(0,-1,0,1);
ty :array[1..4]of longint=(-1,0,1,0);
max =7;
var x,d :array[1..max,1..max]of longint;
n :longint;
dem :longint;
sum :longint;
procedure di(i,j:longint);
var u,v,k:longint;
begin
d[i,j]:=1; inc(sum);
for k:=1 to 4 do
begin
u:=i+tx[k]; v:=j+ty[k];
if (u>0)and(v>0)and(u<=n)and(v<=n)
and(x[u,v]=1)and(d[u,v]=0) then di(u,v);
end;
end;
function KtraHinh:boolean;
var i,j :longint;
begin
sum:=0;
for i:=1 to n do
if x[1,i]=1 then inc(sum);
if sum=0 then exit(false);
sum:=0;
for i:=1 to n do

```

```

if x[i,1]=1 then inc(sum);
if sum=0 then exit(false);
fillchar(d,sizeof(d),0);
sum:=0;
for i:=1 to n do
for j:=1 to n do
if x[i,j]=1 then begin
di(i,j);
if sum<>n then exit(false)
else exit(true);
end;
end;
procedure hien;
var i,j:longint;
begin
for i:=1 to n do begin
for j:=1 to n do write(x[i,j]);
writeln;
end;
writeln;
end;
procedure duyet(i,j,c:longint);
var k:longint;
begin
if c>n then exit;
if i>n then begin
//hien;
if ktraHinh then begin
inc(dem);
hien;
end;
exit;
end;
for k:=0 to 1 do begin
x[i,j]:=k;
if j<n then duyet(i,j+1,c+x[i,j])
else duyet(i+1,1,c+x[i,j]);
end;
end;
BEGIN
n:=5;
duyet(1,1,0);
writeln(dem);
END.

```

Để loại bỏ các hình có thẻ nhận được từ phép quay hoặc đối xứng cần xây dựng thêm thủ tục quay  $90^\circ$  và đối xứng hình.

**4.15.** Một cách đánh giá tốt hơn được thực hiện đơn giản như sau: gọi  $c_{min}$  là trọng số của cạnh bé nhất, khi đó có thể ước lượng chi phí ít nhất nếu tiếp tục đi theo nhánh hiện tại là: chi phí hiện tại cộng với  $c_{min}$  nhân với số cạnh còn lại phải đi.

**4.16.** Dùng mô hình duyệt đệ quy để liệt kê tất cả các cách xếp 8 quân hậu lên bàn cờ  $8 \times 8$ , có tất cả 92 cách. Với mỗi cách tính tổng các số trên các ô mà có quân hậu đứng để chọn ra cách xếp có tổng lớn nhất.

**4.17.** Dùng mô hình duyệt đệ quy để liệt kê tất cả dãy nhị phân độ dài  $n$ , mỗi một dãy nhị phân tương ứng với một cách chọn các đồ vật, số thứ  $i$  bằng 1 tức là vật thứ  $i$  được chọn, số thứ  $i$  bằng 0 tức là vật thứ  $i$  không được lựa chọn. Với mỗi dãy nhị phân, kiểm tra tổng khối lượng các vật được chọn thỏa mãn điều kiện không vượt quá  $w$  và chọn phương án có tổng giá trị là lớn nhất.

**4.18.** Dùng mô hình duyệt đệ quy để liệt kê tất cả dãy nhị phân độ dài  $n$ , mỗi một dãy nhị phân tương ứng với một cách thay đổi các quân domino, số thứ  $i$  bằng 1 tức là quân thứ  $i$  phải quay  $180^\circ$ , số thứ  $i$  bằng 0 tức là quân thứ  $i$  giữ nguyên. Với mỗi dãy nhị phân, tính độ chênh lệnh và số quân domino quay (bằng số lượng số 1 trong dãy nhị phân) để chọn phương án tối ưu.

**4.19. Nhận xét 1:** Thứ tự thực hiện các công tắc là không quan trọng.

**Nhận xét 2:** Mỗi công tắc tác động không quá 1 lần.

Dùng mô hình duyệt đệ quy để liệt kê tất cả dãy nhị phân độ dài  $M + N$ , mỗi một dãy nhị phân tương ứng với một cách tác động vào các công tắc dòng và công tắc cột. Các bit từ thứ 1 đến bit thứ  $M$  mô tả cho các công tắc dòng, các bit thứ  $M + 1$  đến bit thứ  $M + N$  mô tả cho các công tắc cột, bit 1 thể hiện công tắc tương ứng sẽ được tác động, bằng 0 nếu không tác động. Số trường hợp phải thử là  $2^{M+N}$ .

Ta có thể cải tiến thuật toán trên như sau: Duyệt nhị phân (các công tắc dòng có tác động hay không tác động). Khi đã biết các công tắc dòng có hoặc không tác động, ta sẽ quyết định được các công tắc cột có tác động hay không. Số trường hợp phải thử là  $2^M$ .

**4.20. Nhận xét 1:** Có 16 phép biến đổi.

**Nhận xét 2:** Thứ tự thực hiện các phép biến đổi là không quan trọng.

**Nhận xét 3:** Mỗi phép biến đổi thực hiện không quá 1 lần.

Dùng mô hình duyệt đệ quy để liệt kê tất cả bảng nhị phân kích thước  $4 \times 4$ . Mỗi một bảng nhị phân tương ứng với một phép biến đổi. Ô chứa bit 1 thể hiện đồng xu ở ô đó được chọn, bằng 0 nếu không chọn. Số trường hợp phải thử là  $2^{16}$ .

**4.21.** a) Dùng mô hình duyệt đệ quy để liệt kê tất cả các hoán vị của  $1, 2, \dots, n$ . Với mỗi hoán vị tương ứng với một cách sao chép, lần lượt sao chép từng tệp theo thứ tự hoán vị, chỉ chuyển sang đĩa mới khi đĩa hiện tại không thể chứa thêm.

b) Sắp xếp các tệp theo dung lượng giảm dần, tiến hành sao chép các tệp vào từng đĩa CD. Với một đĩa CD, duyệt từ lần lượt các tệp (theo thứ tự đã sắp xếp), nếu tệp nào chưa sao chép mà có thể chứa vào đĩa thì sao chép luôn.

**4.22.** Trong hàm *check*, phần sắp xếp lại các công việc theo thời hạn thực hiện mất  $O(n^2)$ , còn phần thử thực hiện các công việc mất  $O(n)$ . Như vậy, có thể cải tiến phần sắp xếp còn  $O(n)$  để hàm *check* có độ phức tạp  $O(n)$ . Nhận thấy, trong hàm *check*, các công việc trong *Js* chỉ có công việc nằm ở cuối cùng (công việc mới thêm) là không đúng thứ tự, các công việc còn lại đều đang đúng thứ tự, sử dụng ý tưởng thuật toán sắp xếp chèn để sắp xếp.

```
function check(var Js:TArrJobs; nJob:longint):boolean;
var i,j :longint;
t :longint;
begin
for i:=nJob-1 downto 1 do
if Js[i].d>Js[i+1].d then swap(Js[i],Js[i+1]);
t:=0;
for i:=1 to nJob do begin
if t+Js[i].p>Js[i].d then exit(false);
t:=t+Js[i].p;
end;
exit(true);
end;
```

**4.23.** Giả sử  $x$  là số lượng kí tự 'A',  $y$  là số lượng kí tự 'B',  $z$  là số lượng kí tự 'C' của xâu  $S$ . Như vậy, khi biến đổi các kí tự 'A' sẽ nằm ở vùng từ vị trí 1 đến  $x$ , các kí tự 'B' sẽ nằm ở vùng  $(x + 1)$  đến  $(x + y)$ , các kí tự 'C' sẽ nằm ở vùng  $(x + y + 1)$  đến  $(x + y + z)$ .

*Bước 1:* Thực hiện đổi chỗ tất cả các cặp kí tự không đúng vùng để sau khi đổi cả hai đều nằm đúng vùng, mỗi cặp mất một bước biến đổi.

*Bước 2:* Thực hiện đổi chỗ tất cả các bộ ba kí tự không đúng vùng để sau khi đổi cả ba đều nằm đúng vùng, mỗi bộ ba kí tự mất hai bước biến đổi.

**4.24. Bước 1:** Sắp xếp các đoạn tăng dần theo điểm cuối, nếu điểm cuối bằng nhau thì ưu tiên đoạn nào có điểm đầu nhỏ hơn xếp trước.

*Bước 2:* Chọn các số theo thuật toán tham như sau, duyệt từng đoạn theo thứ tự đã sắp xếp, với một đoạn  $[a_i, b_i]$  có ba khả năng xảy ra:

- + đoạn chưa có số nào được chọn sẽ chọn hai số:  $b_i - 1$  và  $b_i$ ;
- + đoạn có một số được chọn rồi sẽ chọn thêm số:  $b_i$ ;
- + đoạn có hai số được chọn rồi không cần chọn.

```
const fi = 'b'; fo = '';
max = 1000;
var a :array[1..max,1..2] of longint;
b :array[1..max] of longint;
n,kq :longint;
f,g :Text;

procedure docdl;
var i :longint;
begin
read(f,n);
for i:=1 to n do read(f,a[i,1],a[i,2]);
end;

procedure qs(x,y:longint);
var dau,cuoi,mid1,mid2,tg :longint;
begin
dau:=x;
cuoi:=y;
mid1:=a[(x+y) shr 1,1];
mid2:=a[(x+y) shr 1,2];
repeat
while (a[dau,2]<mid2) or ((a[dau,2]=mid2) and (a[dau,1]<mid1)) do inc(dau);
while (a[cuoi,2]>mid2) or ((a[cuoi,2]=mid2) and (a[cuoi,1]>mid1)) do dec(cuoi);
if dau<=cuoi then
begin
tg:=a[dau,1];
a[dau,1]:=a[cuoi,1];
a[cuoi,1]:=tg;
tg:=a[dau,2];
a[dau,2]:=a[cuoi,2];
a[cuoi,2]:=tg;
inc(dau);
dec(cuoi);
end;
end;
```

```

end;
until dau>cuoi;
if x<cuoi then qs(x,cuoi);
if dau<y then qs(dau,y);
end;

procedure xuli;
var sl,i,j:longint;
ok:boolean;
begin
qs(1,n);
for i:=1 to n do b[i]:=2;
sl:=0;
for i:=1 to n do
if b[i]>0 then begin
if b[i]=1 then begin
inc(sl);
for j:=i+1 to n do begin
if (a[j,1]<=a[i,2])and(b[j]>0) then dec(b[j]);
end;
end;
if b[i]=2 then begin
inc(sl,2);
for j:=i+1 to n do begin
if (a[j,1]<=a[i,2])and(b[j]>0) then dec(b[j]);
if (a[j,1]<=a[i,2]-1)and(b[j]>0) then dec(b[j]);
end;
end;
writeln(g,sl);
end;

BEGIN
assign(f,fi); reset(f);
assign(g,fo); rewrite(g);
docdl;
xuli;
close(f);
close(g);
END.

```

4.25. Với phân số  $\frac{M}{N}$ , tìm  $K$  lớn nhất không vượt quá  $M$  mà  $K$  là ước của  $N$ ,

phân tích  $\frac{M}{N} = \frac{K}{N} + \frac{M-K}{N}$ , tiếp tục phân tích phân số  $\frac{M-K}{N}$ .

Ví dụ,  $\frac{5}{6} = \frac{3}{6} + \frac{2}{6} = \frac{1}{2} + \frac{1}{3}$ .

Tuy nhiên, thuật toán trên sẽ không đúng với trường hợp  $\frac{4}{5}$ , nếu làm theo

thuật toán trên thì  $\frac{4}{5} = \frac{1}{5} + \frac{1}{5} + \frac{1}{5} + \frac{1}{5}$ . Có thể cải tiến thuật toán bằng cách, trước khi tách nhân cả tử và mẫu với một số.

Ví dụ,  $\frac{4}{5} = \frac{8}{10} = \frac{5}{10} + \frac{3}{10} = \frac{1}{2} + \frac{1}{5} + \frac{1}{10}$ .

#### 4.26.

```
const max =1000;
fi ='PT.INP';
fo ='PT.OUT';
var n,start,cur,k : longint;
v :array[1..max] of boolean;
s :string;
f,g :text;

function nhan(s:string;i:longint):string;
var j,nho,tich:longint;
t :string;
begin
nho:=0;t:='';
for j:=length(s) downto 1 do begin
tich:=(ord(s[j])-48)*i+nho;
nho:=tich div 10;
t:=char(tich mod 10+48)+t;
end;
while nho>0 do begin
t:=char((nho mod 10)+48)+t;
nho:=nho div 10;
end;
nhan:=t;
end;

begin
assign(f,fi); reset(f);
assign(g,fo); rewrite(g);
while not seeeof(f) do begin
read(f,N);
for start:= 2 to 3 do begin
fillchar(v, sizeof(v), false );
k:=start;
cur:=0;
while cur < N do begin
inc(cur,k);
v[k]:=true;
```

```

inc(k);
end;
if (cur > N) and v[cur-N] then begin
v[cur-N]:=false;
cur:=N;
end;
if cur = N then break;
end;
s:='1';
for k:=2 to max do
if v[k] then s:=nhan(s,k);
writeln(g,s);
end;
close(f);close(g);
end.

```

**4.27.** Từ  $N$ , tìm cách biến đổi về 1 dựa trên hai thao tác:

- + Chia 2, nếu  $N$  chẵn;
- + Nhân 3 rồi cộng 1, nếu  $N$  lẻ.

Lật ngược cách biến đổi nhận được biểu thức cần tìm.

Ví dụ:  $6 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ , biểu thức cần tìm là  $1*2*2*2*2/3*2/3*2$ .

**4.28. Bước 1:** Tạo ra dãy số FIB gồm các số không vượt quá  $N$ .

*Bước 2:* Lặp lại bước 2.1 và 2.2 sau cho đến khi  $N = 0$ .

*Bước 2.1:* Tìm số  $\text{FIB}[i]$  lớn nhất nhỏ hơn  $N$ .

*Bước 2.2:* Trừ  $N$  đi  $\text{FIB}[i]$ .

Thuật toán trên cũng là cách phân tích  $N$  thành các số FIB đôi một khác nhau (bài 2.11).

**4.29.** Tương tự bài toán lập lịch giảm thiểu trễ hạn ta có hai nhận xét:

- + Chỉ quan tâm đến việc xếp lịch cho các công việc hoàn thành đúng hạn, còn các công việc bị trễ hạn có thể thực hiện theo trình tự bất kì.
- + Giả sử  $js$  là tập gồm  $k$  công việc (mà cả  $k$  công việc này đều có thể thực hiện đúng hạn) và  $\sigma = (i_1, i_2, \dots, i_k)$  là một hoán vị của các công việc trong  $js$  sao cho  $d_{i_1} \leq d_{i_2} \leq \dots \leq d_{i_k}$ , thì thứ tự  $\sigma$  là thứ tự để hoàn thành đúng hạn được cả  $k$  công việc.

Sử dụng chiến lược tham lam, ta xây dựng tập công việc  $js$  theo từng bước, ban đầu  $js = \emptyset$ . Hàm chọn được xây dựng như sau: tại mỗi bước ta sẽ chọn

công việc  $job_i$  mà có tiền công lớn nhất trong số các công việc còn lại cho vào tập  $js$ . Nếu sau khi kết nạp  $job_i$ , các công việc trong tập  $js$  đều có thể thực hiện đúng hạn thì cố định việc kết nạp  $job_i$  vào tập  $js$ , nếu không thì không kết nạp  $job_i$ .

**4.30.** Bài toán này giống bài 2.8, tuy nhiên  $N$  rất lớn, thuật toán trong bài 2.8 không đáp ứng được, có thể sử dụng kĩ thuật chia để trị có độ phức tạp  $O(\log N)$  như sau:

$$M^N \bmod A = \begin{cases} ((M^{N \text{ div } 2} \bmod A) \times (M^{N \text{ div } 2} \bmod A)) \bmod A, & \text{nếu } N \text{ chẵn} \\ ((M^{N \text{ div } 2} \bmod A) \times (M^{N \text{ div } 2} \bmod A) \times M) \bmod A, & \text{nếu } N \text{ lẻ.} \end{cases}$$

```
function luythua(M,N:longint):int64;
var w : int64;
begin
  w:=luythua(M,N div 2);
  w:=(w*w) mod 10K;
  if N mod 2=1 then w:=(w*M) mod 10K;
  exit(w);
end;
```

**4.31.** Sử dụng hàm *luythua* của bài 4.30.

**4.32.** Áp dụng kĩ thuật chia để trị, với hình kích thước  $2^k$  ( $k > 1$ ) bị khuyết một ô, tiến hành chia làm 4 hình vuông kích thước  $2^{k-1}$  bằng cắt ngang và cắt dọc. Kiểm tra xem ô bị khuyết nằm ở hình vuông nào thì tiếp tục giải quyết bài toán với hình vuông kích thước  $2^{k-1}$  có khuyết một ô, lát 3 hình vuông còn lại như bài lát nền đã trình bày trong phần lí thuyết.

**4.34.** Sử dụng phương pháp quy hoạch động tìm dãy con đơn điệu giảm của dãy  $a_1, a_2, \dots, a_n$ . Gọi  $L[i]$  là độ dài dãy con đơn điệu giảm lớn nhất của dãy  $a_1, a_2, \dots, a_i$ . Tương tự, sử dụng phương pháp quy hoạch động tìm dãy con đơn điệu giảm của dãy  $a_n, a_{n-1}, \dots, a_1$ . Gọi  $R[i]$  là độ dài dãy con đơn điệu giảm lớn nhất của dãy  $a_n, a_{n-1}, \dots, a_i$ . Độ dài dãy con lồi dài nhất là giá trị lớn nhất của  $L[i] + R[i] - 1$  ( $i = 2, 3, \dots, n - 1$ ).

**4.35.** Sử dụng phương pháp quy hoạch động, gọi  $F[i, j]$  là số kí tự ít nhất cần thêm để xâu con liên tiếp từ  $i$  đến  $j$  tạo thành một xâu đối xứng.

+ Trường hợp 1: nếu  $S[i] = S[j]$  thì  $F[i, j] = F[i+1, j-1]$

+ Trường hợp 2: nếu  $S[i] \neq S[j]$  thì  $F[i, j] = \min\{F[i+1, j]+1, F[i, j-1]+1\}$ .

```

uses math;
const MAX = 200;

var f :array[1..MAX,1..MAX]of longint;
s :string;
n :longint;
function qhd(i,j:longint):longint;
begin
  if f[i,j]>-1 then exit(f[i,j]);
  if i>=j then
  begin
    f[i,j]:=0;
    exit(0);
  end;
  if s[i]=s[j] then f[i,j]:=qhd(i+1,j-1)
  else f[i,j]:=math.min(qhd(i+1,j)+1,qhd(i,j-1)+1);
  exit(f[i,j]);
end;

BEGIN
  fillchar(f,sizeof(f),255);
  s:='acbcd';
  n:=length(s);
  writeln(qhd(1,n));
END.

```

4.36. Gọi  $L[i,j]$  là chi phí nhỏ nhất để ghép các đồng sói từ  $i$  đến  $j$  thành một đồng.

Tính  $L[i,j]$ : thử chọn tất cả  $k$  trong khoảng  $[i..j]$ , khi đó:

$$L[i,j] = \min(L[i,k] + L[k+1,j] + T[i,j])$$

( $i \leq k < j$ ;  $T[i,j]$  là tổng số sói từ đồng thứ  $i$  đến đồng thứ  $j$ ).

```

const MAX = 200;
var f :array[1..MAX,1..MAX]of longint;
a,s :array[0..MAX]of longint;
n :longint;

function qhd(i,j:longint):longint;
var k,w :longint;
begin
  if f[i,j]>-1 then exit(f[i,j]);
  if i=j then
  begin
    f[i,j]:=0;
    exit(0);
  end;
  f[i,j]:=qhd(i+1,j)+s[j]-s[i-1];
  for k:=i+1 to j-1 do begin

```

```

w:=qhd(i,k)+qhd(k+1,j)+s[j]-s[i-1];
if w<f[i,j] then f[i,j]:=w;
end;
exit(f[i,j]);
end;

procedure docDL;
var i :longint;
begin
read(n);
for i:=1 to n do read(a[i]);
s[0]:=0;
for i:=1 to n do s[i]:=s[i-1]+a[i];
end;

BEGIN
docDL;
fillchar(f,sizeof(f),255);
writeln(qhd(1,n));
END.

```

4.37. Gọi  $f(a, b)$  là số hình vuông ít nhất được cắt từ hình chữ nhật kích thước  $a \times b$  ( $a \in [1..M]$ ,  $b \in [1..N]$ ). Khi đó, dùng bảng  $F[1..100,1..100]$  để lưu lại các giá trị  $f(a, b)$ .

- Các bài toán con nhỏ nhất (cơ bản) có  $a = b$ , khi đó  $f(a, b) = 1$ .
- Công thức tính:  $f(a, b) = \min \begin{cases} f(a_1, b) + f(a_2, b) & \text{với } a_1 + a_2 = a \\ f(a, b_1) + f(a, b_2) & \text{với } b_1 + b_2 = b \end{cases}$

Như vậy, có  $M \times N$  bài toán con và chi phí chuyển mất  $O(M + N)$ , do đó độ phức tạp của thuật toán là:  $O(M \times N \times (M + N))$ .

*Cách cài đặt bằng vòng lặp*

```

const MAX =500;
fi ='hcn.inp';
fo ='hcn.out';
var f :array[1..MAX,1..MAX]of longint;
m,n :longint;

procedure tinh(m,n:longint);
var a,b,c,res:longint;
begin
for a:=1 to m do
for b:=1 to n do
begin
if a=b then begin
f[a,b]:=1;

```

```

        continue;
    end;
    res:=a*b;
    for c:=1 to a-1 do
        if res>(f[c,b]+f[a-c,b]) then res:=(f[c,b]+f[a-c,b]);
    for c:=1 to b-1 do
        if res>(f[a,c]+f[a,b-c]) then res:=(f[a,c]+f[a,b-c]);
    f[a,b]:=res;
    end;
    end;

BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(m,n);
    tinh(m,n);
    writeln(f[m,n]);
    close(input);
    close(output);
END.

```

Cách cài đặt bằng đệ quy có nhớ

```

const MAX =500;
fi ='hcn.inp';
fo ='hcn.out';
var f :array[1..MAX,1..MAX]of longint;
m,n :longint;

function tinh(a,b:longint):longint;
var c,res:longint;
begin
    if f[a,b]<>-1 then exit(f[a,b]);
    if a=b then begin
        f[a,b]:=1;
        exit(f[a,b]);
    end;
    res:=a*b;
    for c:=1 to a-1 do
        if res>(tinh(c,b)+tinh(a-c,b)) then res:=(f[c,b]+f[a-c,b]);
    for c:=1 to b-1 do
        if res>(tinh(a,c)+tinh(a,b-c)) then res:=(f[a,c]+f[a,b-c]);
    f[a,b]:=res;
    exit(f[a,b]);
end;

BEGIN
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(m,n);

```

```

fillchar(f,sizeof(f),255);
writeln(tinh(m,n));
close(input);
close(output);
END.

```

**4.38.** Gọi  $L[x, y, u, v]$  là số hình chữ nhật ít nhất khi cắt hình chữ nhật có điểm trái trên là  $(x,y)$ , điểm phải dưới là  $(u,v)$ . Nếu hình chữ nhật  $(x, y, u, v)$  toàn là 0 hoặc 1 thì  $L[x, y, u, v] = 1$ .

Tính  $L[x, y, u, v]$ : thử tất cả các nhát cắt ngang theo hàng từ  $x$  đến  $u$  và các nhát cắt dọc theo cột từ  $y$  đến  $v$ , với mỗi nhát cắt ta sẽ chia hình chữ nhật đang xét thành 2 hình nhỏ hơn, tổng kết quả của 2 hình con sẽ là kết quả của hình lớn.

$$L[x, y, u, v] = \min \begin{cases} L[x, y, x_c, v] + L[x_c + 1, y, u, v] \\ L[x, y, u, y_c] + L[x, y_c + 1, u, v] \end{cases}$$

**4.39.** Yêu cầu bài toán là chọn ra  $3K$  phần tử trong  $N$  phần tử và chia thành  $k$  nhóm. Gọi  $L(i, k)$  là tổng theo cách chọn tối ưu  $k$  nhóm trong các phần tử từ 1 đến  $i$ , khi đó  $L(i, k) = \max\{L(j, k - 1)\} + \text{tổng tối ưu cách chọn ba chỉ số trong đoạn từ phần tử } j + 1 \text{ đến } i\}$ .

Số bài toán con là  $N \times k$ . Chi phí chuyển trạng thái mất  $N$  nhân chi phí tính tổng tối ưu cách chọn 3 chỉ số trong đoạn từ phần tử  $j + 1$  đến  $i$ . Việc tính tổng tối ưu cách chọn 3 chỉ số trong đoạn từ phần tử  $j + 1$  đến  $i$  có thể chuẩn bị trước và lưu trữ vào mảng hai chiều, khi đó việc tính tổng chỉ còn  $O(1)$ . Độ phức tạp thuật toán là:  $O(N^2 K)$ .

Tuy nhiên, ta có thể thay đổi một chút cách nhìn và có được lời giải đơn giản ngắn gọn hơn. Gọi  $L(i, k)$  là tổng chọn ra  $k$  phần tử trong  $i$  phần tử đầu tiên của dãy,  $L(i, k) = \max\{L(j, k - 1) + \text{sign}(k) * a[i]\}$ , trong đó

$$\text{sign}(k) = \begin{cases} -1, & \text{nếu } k \bmod 3 = 2 \\ 1, & \text{ngược lại.} \end{cases}$$

```

const max = 500;
fi ='tkseq.inp';
fo ='tkseq.out';
limit =trunc(1e13);
cs :array[0..2]of int64=(1,1,-1);
var a :array[1..max]of int64;
l :array[0..max,0..max]of int64;
n,m :longint;

```

```

f :text;
procedure docf;
var i :longint;
begin
assign(f,fi); reset(f);
readln(f,n,m);
for i:=1 to n do read(f,a[i]);
close(f);
end;

procedure tinh;
var i,j,k,mm :longint;
p :int64;
begin
m:=m*3;
for i:=0 to n do
for j:=0 to m do l[i,j]:=-limit;

for i:=0 to n do l[i,0]:=0;

for i:=1 to n do begin
if m>i then mm:=i else mm:=m;

for j:=1 to mm do
for k:=i downto j do
if l[i,j]< l[k-1,j-1]+a[k]*cs[j mod 3] then
l[i,j]:=l[k-1,j-1]+a[k]*cs[j mod 3];
end;
assign(f,fo); rewrite(f);
writeln(f,l[n,m]);
close(f);
end;

BEGIN
docf;
tinh;
END.

```

4.40. Gọi  $f(i, k)$  là tổng trọng số của cách chia tối ưu các phần tử từ 1 đến  $i$  thành  $k$  đoạn, khi đó  $f(i, k) = \min\{f(i, k-1)\} +$  trọng số đoạn từ phần tử  $j+1$  đến  $i\}$ .

Số bài toán con là  $N \times G$ . Chi phí chuyển trạng thái là  $O(N^2)$ , trong đó bao gồm việc tính trọng số đoạn từ phần tử  $j+1$  đến  $i$ . Như vậy, độ phức tạp theo thuật toán trên là  $O(N^3G) = O(N^{3.5})$  và không đáp ứng được với  $n = 1000$ . Có thể giảm độ phức tạp thuật toán bằng cách tối ưu công thức tính trọng số đoạn từ phần

từ  $j + 1$  đến  $i$ . Thay vì phải tính trọng số đoạn từ phần tử  $j + 1$  đến  $i$  mất  $O(N)$  thì việc tính này chỉ mất  $O(1)$ , khi đó độ phức tạp thuật toán chỉ còn  $O(N^{25})$ .

```
const max =1000;
fi ='lss.inp';
fo ='lss.out';
var a :array[1..max]of extended;
f :array[1..max,1..35]of extended;
sum,sum2 :array[0..max]of extended;
n,g :longint;

procedure docf;
var i :longint;
begin
readln(n,g);
for i:=1 to n do begin
readln(a[i]);
sum[i]:=sum[i-1]+a[i];
sum2[i]:=sum2[i-1]+ a[i] * a[i];
end;
end;

procedure qhd;
var i,j,k,soNhom :longint;
p :extended;
begin
for i:=1 to n do
for j:=1 to g do f[i,j]:=1e100;

for i:=1 to n do f[i,1]:=sum2[i]-sqr(sum[i])/i;
for i:=2 to n do begin
soNhom:=i;
if i>g then soNhom:=g;
for j:=2 to soNhom do
for k:=i downto j do begin
p:=(sum2[i]-sum2[k-1])-sqr(sum[i]-sum[k-1])/(i-k+1);
if f[i,j]>p+f[k-1,j-1] then f[i,j]:=p+f[k-1,j-1];
end;
end;
end;
BEGIN
assign(input,fi); reset(input);
assign(output,fo); rewrite(output);
docf;
qhd;
writeln(f[n,g]:0:2);
close(input); close(output);
END.
```

N) thì

**4.41.** Gọi  $F[i]$  là hệ số phật nhỏ nhất để phân các từ 1 đến  $i$  vào một số dòng.

Khi đó  $F(i) = \min \{ \max \{ F(j-1, L - \sum_{k=j}^i w_k) \} \}$  với  $1 \leq j \leq i$  và  $\sum_{k=j}^i w_k \leq L$ .

**4.42.** Một số nguyên  $t$  ( $0 \leq t \leq 2^n$ ) biểu diễn ở dạng nhị phân gồm đúng  $n$  bit sẽ biểu diễn cho một tập con của tập  $\{1, 2, \dots, n\}$ . Bit thứ  $i$  bằng 1 nếu số thứ  $i$  thuộc tập con, bằng 0 nếu ngược lại. Gọi  $F[i, t]$  là tổng lớn nhất đạt được khi chọn các số thuộc từ dòng 1 đến dòng  $i$ ,  $t$  là trạng thái mô tả các cột đã được chọn số, số lượng bit 1 trong  $t$  chính là số lượng số đã chọn.

$F[i, t] = \max \{ F[i-1, t], \max \{ F[i-1, t'] + a[i, j] \} \}$ , trong đó  $j = 1, 2, \dots, n$  và  $t'$  nhận được từ  $t$  nếu tắt bit thứ  $j$  của  $t$  (bit thứ  $j$  của  $t$  phải bằng 1).

```
const MAX =15;
fi ='SELECT.INP';
fo ='SELECT.OUT';
var a :array[1..max,1..max]of integer;
l1,l2 :array[0..(1 shl max)-1] of longint;
c1,c2 :array[0..(1 shl max)-1] of int64;
n,KK :integer;
tong,socach :int64;

procedure docf;
var i,j :integer;
f :text;
begin
assign(f,fi); reset(f);
readln(f,n,KK);
for i:=1 to n do
for j:=1 to n do read(f,a[i,j]);
close(f);
end;

procedure lam;
var i,j,k,dem : longint;
d : array[1..max]of integer;
begin
c2[0]:=1;
for i:=1 to n do begin
l1:=l2; c1:=c2;
for j:=0 to (1 shl n)-1 do
begin
dem:=0;
fillchar(d,sizeof(d),0);
for k:=0 to n-1 do
if j and (1 shl k) > 0 then begin d[k+1]:=1; inc(dem); end;
if dem<KK then
```

```

for k:=1 to n do
if d[k]=0 then
begin
if l1[j]+a[i,k]>l2[j or (1 shl (k-1))] then
begin
l2[j or (1 shl (k-1))]:=l1[j]+a[i,k];
c2[j or (1 shl (k-1))]:=c1[j];
end
else if (l1[j]+a[i,k]=l2[j or (1 shl (k-1))]) then
c2[j or (1 shl (k-1))]:=c2[j or (1 shl (k-1))]+c1[j];
end;
end;
end;

procedure ghikq;
var f :text;
j,k,dem :longint;
begin
tong := 0;
for j:=0 to (1 shl n) - 1 do
begin
dem:=0;
for k:=0 to n-1 do
if j and (1 shl k) > 0 then inc(dem);
if dem = KK then
if tong<l2[j] then begin tong:=l2[j]; socach:=c2[j]; end
else if tong=l2[j] then socach:=socach + c2[j];
end;
assign(f,fo); rewrite(f);
writeln(f,tong,' ',socach);
close(f);
end;

BEGIN
docf;
lam;
ghikq;
END.

```

4.43. Thêm tạm các số 0 vào đầu các xâu để hai số hạng có độ dài bằng độ dài của tổng (gọi  $n$  là độ dài xâu tổng). Sử dụng phương pháp quy hoạch động để lập bảng phương án như sau:  $f[p,nho]=\text{true/false}$  có ý nghĩa là có tồn tại/không tồn tại phương án điền các số từ cuối về vị trí ở cột  $p$  ( $1 \leq p \leq n$ ) và có nhó lên cột  $p - 1$  là  $nho$  ( $0 \leq nho \leq 1$ ). Sau khi lập được bảng phương án xong, sử dụng bảng phương án để tìm một nghiệm. Tuy nhiên, cách làm này

cài đặt khá phức tạp. Chương trình dưới đây sử dụng kĩ thuật duyệt quay lui nhưng có đánh dấu các trạng thái đã duyệt.

```
{$mode objfpc}
const fi='rebuss.inp';
      fo='rebuss.out';
type TNum= Record
  x, y, z: char;
end;
var f: text;
n: integer;
a, kq: array[1..3] of string;
b: array[1..3] of integer;
t: array[0..1, 0..100] of TNum;
free: array[0..1, 0..100] of boolean;

procedure enter;
var i: integer;
begin
  assign(f, fi); reset(f);
  for i:= 1 to 3 do
    begin
      readln(f, a[i]);
      b[i]:= length(a[i]);
    end;
  close(f);
end;

procedure printresult;
var i, p: integer;
begin
  kq[1]:=''; kq[2]:=''; kq[3]:='';
  p:= 0;
  for i:= n downto 1 do begin
    kq[1]:= t[p, i].x+ kq[1];
    kq[2]:= t[p, i].y+ kq[2];
    kq[3]:= t[p, i].z+ kq[3];
    if ord(t[p, i].x)+ ord(t[p, i].y) - 48 + p= ord(t[p, i].z) then p:= 0
    else p:= 1;
  end;
  for i:= 1 to 3 do
    while kq[i][1]= '0' do delete(kq[i], 1, 1);
  for i:= 1 to 3 do writeln(f, kq[i]);
end;

procedure thu(i, p: integer);
var
  x: array[1..2, 1..3] of char;
  j: integer;
```

dài  
g đê  
tòn  
) và  
g án  
này

```

pp, q, k: char;
begin
  if not free[p, i] then exit;
  if (i= 0) and (p=0) then begin
    printresult;
    close(f); halt;
  end;
  for j:= 1 to 3 do
    if a[j][i] <> '*' then begin
      x[1, j]:= a[j][i]; x[2, j]:= a[j][i];
    end else begin
      if i<> n- b[j]+ 1 then x[1, j]:= '0' else x[1, j]:= '1';
      x[2, j]:= '9';
    end;
    for pp:= x[1, 1] to x[2, 1] do
      for q:= x[1, 2] to x[2, 2] do
        for k:= x[1, 3] to x[2, 3] do
          if (ord(pp)+ ord(q)- 48 + p= ord(k)) and free[0, i-1] then
            begin
              t[p, i].x:= pp; t[p, i].y:= q; t[p, i].z:= k;
              free[p, i]:= false; thu(i-1, 0);
            end else
              if (ord(pp)+ ord(q)- 48+ p= ord(k)+ 10) and free[1, i-1] then begin
                t[p, i].x:= pp; t[p, i].y:= q; t[p, i].z:= k;
                free[p, i]:= false;
                thu(i-1, 1);
              end;
            end;
  end;

procedure solve;
var i, j: integer;
begin
  fillchar(free, sizeof(free), true);
  n:= b[3];
  for i:= 1 to 3 do
    begin
      j:= b[i];
      while j< n do begin a[i]:= '0'+a[i]; inc(j); end;
    end;
  thu(n, 0);
  writeln(f, 'No Solution');
end;

begin
  enter;
  assign(f, fo); rewrite(f);
  solve;
  close(f);
end.

```

#### 4.44.

```
uses crt;
const max = 1000;
fi = 'SCHEDULING.INP';
fo = 'SCHEDULING.OUT';
var a,s : array[1..max+1]of integer;
w : array[1..max+1]of longint;
d : array[1..max+1]of byte;
l,c,n : integer;

procedure input;
var f : text;
k : integer;
begin
assign(f,fi);
reset(f);
readln(f,n);
readln(f,l,c);
for k:=1 to n do read(f,a[k]);
close(f);
end;

function solution(k:integer):integer;
var sum,sb : integer;
begin
sb:=0;
while k<=n do
begin
sum:=1;
while (k<=n) and (sum>=a[k]) do
begin
sum:=sum-a[k];
inc(k);
end;
inc(sb);
end;
solution:=sb;
end;

procedure init;
var k : integer;
begin
fillchar(d,sizeof(d),0);
fillchar(w,sizeof(w),0);
d[n+1]:=1;w[n+1]:=0;
fillchar(s,sizeof(s),0);
for k:=1 to n do s[k]:=solution(k);
end;
```

```

function qhd(i:integer):longint;
var j,sum : integer;
s1,s2 : longint;
begin
if d[i]=0 then
begin
j:=i;
sum:=1;
s1:=maxlongint;
while (j <= n) and (sum >= a[j]) do
begin
sum:=sum-a[j];
if s[i] = s[j+1] + 1 then
begin
if (sum<=10)and(sum>0) then s2:=-c+qhd(j+1)
else if sum>10 then
s2:=(sum-10)*(sum-10)+qhd(j+1)
else if sum=0 then s2:=qhd(j+1);
if s2<s1 then s1:=s2;
end;
inc(j);
end;
d[i]:=1;
w[i]:=s1;
end;
qhd:=w[i];
end;

procedure output;
var f : text;
begin
assign(f,fo);
rewrite(f);
writeln(f,solution(1));
writeln(f,qhd(1));
close(f);
end;

BEGIN
input;
init;
output;
END.

```

#### 4.45.

```

{$M 200000000}
Const FI='GARDEN.INP';
FO='GARDEN.OUT';
Maxn=1000001;

```

```

Var L: Array[0..maxn, 0..2, 0..2, 0..1] of longint;
S: Array[0..maxn] of longint;
N, M, Re :Longint;
F: Text;

Procedure Enter;
Var i: longint;
c: char;
Begin
Assign(f, fi); reset(f);
Readln(f, n);
Readln(f, m);
For i:=1 to n do
Begin
read(f, c);
If c='L' then S[i]:=0
Else S[i]:=1;
End;
Close(f);
End;

Function Max(x, y: longint): longint;
Begin
If x>=y then Exit(x) else exit(y);
End;

Function Min(x, y: longint): longint;
Begin
If x<=y then Exit(x) else Exit(y);
End;

Function Tinh(i, hl, hp, ok: longint): longint;
Var
j: longint;
Begin
If L[i, hl, hp, ok] <> -1 then Exit(L[i, hl, hp, ok]);
If i>n then
Begin
L[i, hl, hp, ok]:=1;
Exit(1);
End;
L[i, hl, hp, ok]:=0;
If (hl<2) then
L[i, hl, hp, ok]:=Tinh(i+1, hl+1, max(0, hp-1), ok or
ord(0<s[i]));
If ((ok=0) and (s[i]=1)) or (ok=1) ) and (hp<2) then
L[i, hl, hp, ok]:=(L[i, hl, hp, ok] + Tinh(i+1, max(0, hl-1),
hp+1, ok)) mod m;
Exit(l[i, hl, hp, ok]);

```

```

End;

Procedure Solve;
Var tg: longint;
Begin
  Fillchar(l, sizeof(l), 255);
  Re:=Tinh(l, 0, 0, 0);
End;

Procedure Print;
Begin
  Assign(f, fo); rewrite(f);
  Writeln(f, (re+m) mod m);
  Close(f);
End;

BEGIN
  Enter;
  Solve;
  Print;
END.

```

#### 4.46.

• Một số là số rõ ràng nếu tổng bình phương các chữ số của nó là một số rõ ràng. Ta sẽ chuẩn bị trước mảng *isClear[0..MaxLeng\*9\*9] of boolean* để dễ dàng kiểm tra tính rõ ràng của một số có không quá *MaxLeng* chữ số.

• Xét tập các số có dạng  $x_1x_2\dots x_{16}$  thỏa mãn  $x_i \in [0..9]$  và  $\sum x_i^2$  là số rõ ràng. Với một số  $n$  ta có thể xác định được  $t$ , là thứ tự từ điển của số rõ ràng nhỏ nhất lớn hơn  $n$ . Sau đó xác định số rõ ràng có thứ tự từ điển thứ  $t + m$ . Để tìm từ thứ tự sang số và từ số sang thứ tự có thể sử dụng phương pháp quy hoạch động như sau:  $C[l, s]$  là số lượng cấu hình có  $l$  chữ số và tổng bình phương các chữ số là  $s$ .

Khi cài đặt chương trình giải bằng quy hoạch động  $C[l, s, ok]$ , thêm *ok* để kiểm soát tập các số lớn hơn  $n$ , như vậy chỉ cần giải bài toán tìm số có thứ tự thứ  $m$ .

```

const fi = 'clear.inp';
fo = 'clear.out';
MaxN = 15 + 1;
var a, m, count :int64;
l :array[1..MaxN+1, 0..MaxN*9*9, 0..1]of int64;
clear :array[0..MaxN*9*9]of longint;
f, g :text;
tmps, res :string;
t :longint;

```

```

procedure init;
var i,j,k,p1,p2 :longint;
sp :string;
begin
fillchar(clear,sizeof(clear),0);
clear[1]:=1;
clear[0]:=2;
for i:=2 to MaxN*9*9 do begin
p1:=i;
for j:=1 to MaxN*9*9 do begin
str(p1,sp);
p2:=0;
for k:=1 to length(sp) do p2:=p2 + sqr(ord(sp[k])-48);
if clear[p2]>0 then begin clear[i]:=clear[p2]; break; end;
p1:=p2;
end;
if clear[i]=0 then clear[i]:=2;
end;
end;

function dp(i, s, ok :longint):int64;
var j :longint;
begin
if l[i,s,ok]=-1 then begin
l[i,s,ok]:=0;
if i=MaxN+1 then begin
if clear[s]=1 then l[i,s,ok]:=1;
end
else for j:=(1-ok)*(ord(tmps[i])-48) to 9 do
l[i,s,ok]:=l[i,s,ok]+dp(i+1,s+sqr(j),ok or ord(j>ord(tmps[i])-48));
end;
dp:=l[i,s,ok];
end;

procedure find(i, s, ok :longint);
var j :longint;
begin
for j:=(1-ok)*(ord(tmps[i])-48) to 9 do begin
if count+l[i+1,s+sqr(j),ok or ord(j>ord(tmps[i])-48)]<m then
count:=count+l[i+1,s+sqr(j),ok or ord(j>ord(tmps[i])-48)]
else begin
res:=res + chr(j+48);
if i<MaxN then find(i+1,s+sqr(j),ok or ord(j>ord(tmps[i])-48));
break;
end;
end;
end;

```

```

BEGIN
  assign(f,fi); reset(f);
  assign(g,fo); rewrite(g);
  init;
  readln(f,t);
  while t>0 do begin
    readln(f,a,m);
    dec(t);
    str(a+1,tmps);
    while length(tmps)<MaxN do tmps:='0'+tmps;
    fillchar(l,sizeof(l),255);
    dp(1,0,0);
    count:=0;
    res:='';
    find(1,0,0);
    while (length(res)>0) and (res[1]='0') do delete(res,1,1);
    writeln(g,res);
  end;
  close(f); close(g);
END.

```

4.47. Giả sử  $x$  là tập các khu rừng có nấm mà Bông sẽ đến, gọi  $t(x)$  là thời gian nhỏ nhất để xuất phát từ khu rừng 1 và đi được đến hết tất cả tập  $x$ . Như vậy còn  $P - t(x)$  thời gian dành cho hái nấm.

+ Việc tính  $t(x)$  có thể sử dụng phương pháp quy hoạch động với độ phức tạp  $O(n^2 2^n)$  như sau: gọi  $F[i,y]$  là thời gian nhỏ nhất để xuất phát từ khu rừng 1 và đi được đến hết tất cả tập  $y$  và đang đứng ở khu rừng  $i$ .

+ Việc tính số nấm nhiều nhất có thể hái được với  $P - t(x)$  thời gian có thể thực hiện bằng cách sử dụng thuật toán tham như sau: với mỗi đơn vị thời gian, tìm khu rừng còn nhiều nấm nhất trong tập  $x$  để hái, sau đó cập nhật lại lượng nấm ở khu rừng đó.

Thử tất cả các tập  $x$  để chọn phương án tối ưu.

```

uses math;
const
  inputfile = 'mushroom.inp';
  outputfile = 'mushroom.out';
  maxn = 101;
  maxm = 11;
  stt = 1 shl maxm;
  oo = maxlongint div 10;

var
  fi, fo: text;
  d: array[0..maxn, 0..maxn] of longint;
  dr: array[0..maxm, 0..maxm] of longint;
  f: array[0..stt, 0..maxm] of longint;

```

```

fr: array[0..stt] of longint;
r, Sr, Sr2: array[0..maxm] of longint;
n, m, time, tt: longint;
ans: int64;

procedure nhap;
var i, j: longint;
begin
  readln(fi, m, n, time);
  for i := 1 to m do
    readln(fi, r[i], sr[i]);
  for i := 1 to n do
    begin
      for j := 1 to n do
        read(fi, d[i, j]);
      readln(fi);
    end;
  end;

procedure floyd;
var i, j, k: longint;
begin
  for k := 1 to n do
    for i := 1 to n do
      for j := 1 to n do
        d[i, j] := min(d[i, j], d[i, k] + d[k, j]);
  end;

procedure taodothi;
var i, j: longint;
begin
  r[0] := 1;
  for i := 0 to m do
    for j := 0 to m do
      dr[i, j] := d[r[i], r[j]];
  end;

function orbit(x, j: longint): longint;
begin
  exit(x or (1 shl (j-1)));
end;

procedure khoitao;
var i, j: longint;
begin
  tt := 1 shl m - 1;
  for i := 1 to tt do
    for j := 1 to m do
      f[i, j] := oo;

```

```

for j := 1 to m do
  f[orbit(0, j), j] := dr[0, j];
end;
procedure qhd;
var
  i, j, k, x: longint;
begin
  for i := 1 to tt do
    for j := 1 to m do
      for k := 1 to m do
        begin
          x := orbit(i, k);
          if f[x, k] > f[i, j] + dr[j, k] then
            f[x, k] := f[i, j] + dr[j, k];
          end;
        end;
      end;
    end;

function andbit(x, j: longint): longint;
begin
  exit(x and (1 shl (j-1)));
end;

procedure gopdd;
var i, j: longint;
begin
  for i := 1 to stt do
    fr[i] := oo;
  for i := 1 to stt do
    for j := 1 to m do
      fr[i] := min(fr[i], f[i, j]);
  end;

function dem(p: longint): int64;
var i, j, k: longint;
  sum: int64;
begin
  sum := 0;
  for i := 1 to p do
    begin
      k := 1;
      for j := 1 to m do
        if sr2[j] > sr2[k] then k := j;
        if sr2[k] < 2 then exit(sum);
        sr2[k] := sr2[k] div 2;
      sum := sum + sr2[k];
    end;
    exit(sum);
  end;
procedure truyvet;

```

```

var i, j: longint;
begin
  ans := 0;
  for i := 1 to tt do
    if fr[i] < time then
      begin
        for j := 1 to m do
          if andbit(i, j) > 0 then
            sr2[j] := sr[j]
          else
            sr2[j] := 0;
        ans := max(ans, dem(time - fr[i]));
      end;
    end;

procedure gnkq;
begin
  writeln(fo, ans);
end;

Begin
  assign(fi, inputfile); reset(fi);
  assign(fo, outputfile); rewrite(fo);
  nhap;
  floyd;
  taodothi;
  khoitao;
  qhd;
  gopdd;
  truyvet;
  gnkq;
  close(fi);
  close(fo);
End.

```

#### 4.48. Mật khẩu

*Thuật toán 1:* Độ phức tạp  $O(n^3)$ .

Xác định vector nghiệm:  $(x_1, x_2)$  trong đó  $1 \leq x_1 \leq x_2 \leq \text{length}(S)$  tương ứng là cặp chỉ số  $(i, j)$ .

Xác định ràng buộc: Trong xâu  $S$ , từ vị trí  $x_1$  đến vị trí  $x_2$  phải xuất hiện chữ in hoa ('A'..'Z'), một chữ cái thường ('a'..'z'), một chữ số ('0'..'9') và  $x_2 - x_1 \geq 5$ .

Để liệt kê tất cả các khả năng ta có thể sử dụng hai vòng lặp lồng nhau, cụ thể thủ tục liệt kê như sau:

```

procedure try;
var x1, x2, n :longint;
begin
n:=length(S);
count:=0;
for x1:=1 to n do
  for x2:=1 to n do
    if isOK(x1,x2) then count:=count+1;
end;

```

Trong thủ tục trên, biến  $S$  là biến toàn cục, lưu xâu  $S$  dữ liệu vào; biến  $count$  là biến toàn cục dùng để đếm số lượng cặp thỏa mãn; hàm  $isOK(x1,x2)$  nhận hai tham số đầu vào là hai chỉ số của xâu  $S$  có nhiệm vụ kiểm tra ràng buộc, trả về TRUE nếu thỏa mãn ràng buộc, FALSE trong trường hợp ngược lại. Trong hàm này ta sẽ phải kiểm tra các điều kiện sau:

- 1)  $x_2 - x_1 \geq 5$ ;
- 2) Từ vị trí  $x_1$  đến vị trí  $x_2$  của xâu  $S$  có kí tự thuộc  $['A'..'Z']$
- 3) Từ vị trí  $x_1$  đến vị trí  $x_2$  của xâu  $S$  có kí tự thuộc  $['a'..'z']$
- 4) Từ vị trí  $x_1$  đến vị trí  $x_2$  của xâu  $S$  có kí tự thuộc  $['0'..'9']$

Hàm  $isOK(x1,x2)$  cụ thể như sau:

```

function isOK(x1,x2:longint):boolean;
begin
  isOK:=(x2-x1>=5) and (checkHAZ(x1,x2)) and (checkLaz(x1,x2))
        and (check09(x1,x2));
end;

```

Trong đó hàm  $checkHAZ(x1,x2)$  để kiểm tra điều kiện thứ hai (từ vị trí  $x_1$  đến vị trí  $x_2$  của xâu  $S$  có kí tự thuộc  $['A'..'Z']$ ). Hàm này trả về giá trị TRUE nếu thỏa, FALSE trong trường hợp ngược lại. Tương tự  $checkLaz(x1,x2)$  để kiểm tra điều kiện thứ ba,  $check09(x1,x2)$  để kiểm tra điều kiện thứ tư.

```

function checkHAZ(x1,x2:longint):boolean;
var i : longint;
begin
  for i:=x1 to x2 do
    if (S[i]>='A') and (S[i]<='Z') then exit(TRUE);
    exit(FALSE);
end;
function checkLaz(x1,x2:longint):boolean;
var i : longint;
begin

```

```

for i:=x1 to x2 do
  if (S[i]>='a')and(S[i]<='z') then exit(TRUE);
  exit(FALSE);
end;
function check09(x1,x2:longint):boolean;
var i : longint;
begin
  for i:=x1 to x2 do
    if (S[i]>='0')and(S[i]<='9') then exit(TRUE);
    exit(FALSE);
end;

```

Tuy nhiên, ta nhận thấy ba hàm kiểm tra trên có thể thu gọn thành hàm *check* dưới đây.

```

function check(x1,x2:longint; c1,c2:char):boolean;
var i : longint;
begin
  for i:=x1 to x2 do
    if (S[i]>=c1)and(S[i]<=c2) then exit(TRUE);
    exit(FALSE);
end;

```

### Chương trình hoàn chỉnh

```

const fi ='MATKHAU.INP';
fo ='MATKHAU.OUT';
var S :ansistring;
count :int64;

function check(x1,x2:longint; c1,c2:char):boolean;
var i :longint;
begin
  for i:=x1 to x2 do
    if (S[i]>=c1)and(S[i]<=c2) then exit(TRUE);
    exit(FALSE);
end;

function isOK(x1,x2:longint):boolean;
begin
  isOK:=(check(x1,x2,'A','Z'))and(check(x1,x2,'a','z'))and(check(x1,x2,'0','9'));
end;

procedure try;
var x1, x2, n :longint;
begin
  n:=length(S);
  count:=0;
  for x1:=1 to n-5 do

```

```

for x2:=x1+5 to n do
  if isOK(x1,x2) then count:=count+1;
end;

procedure readFile;
var f :text;
begin
  assign(f,fi); reset(f);
  readln(f,S);
  close(f);
end;

procedure writeResult;
var f :text;
begin
  assign(f,fo); rewrite(f);
  writeln(f,count);
  close(f);
end;

BEGIN
  readFile;
  try;
    writeResult;
END.

```

Trong chương trình trên, thủ tục *try* đã có thay đổi nhằm mục đích chỉ thử các trường hợp mà  $x_2 - x_1 \geq 5$ , do đó giảm số trường hợp phải thử giảm và trong thủ tục check cũng không cần phải kiểm tra điều kiện  $x_2 - x_1 \geq 5$ . Độ phức tạp của thuật toán: Số trường hợp phải thử  $O(n^2)$  nhân với chi phí kiểm tra  $O(n)$ , do đó độ phức tạp thuật toán trên là:  $O(n^3)$ .

Chương trình trên có biến S thuộc kiểu dữ liệu *ansistring* (kiểu xâu kí tự cho phép độ dài lớn hơn 255) vì độ dài xâu vào lên tới  $10^6$ , biến *count* thuộc kiểu *int64* (miền giá trị  $-2^{63}$  đến  $2^{63}-1$ ) vì số lượng có thể vượt quá kiểu *longint* khi độ dài xâu lên tới  $10^6$ . Kiểu *ansistring* và *int64* là hai kiểu dữ liệu có trong FreePascal.

*Thuật toán 2: Độ phức tạp  $O(n^2)$ .*

Độ phức tạp của lời giải trên là  $O(n^3)$  với  $n$  là độ dài xâu S. Để giảm độ phức tạp thì phải giảm số trường hợp phải thử hoặc giảm chi phí phần kiểm tra. Để nhận thấy có thể giảm chi phí kiểm tra theo nhận xét sau: việc kiểm tra

$(x_1, x_2)$  có thể kiểm tra nhanh trong  $O(1)$  khi đã kiểm tra  $(x_1, x_2 - 1)$ . Như vậy độ phức tạp thuật toán chỉ còn  $O(n^2)$ .

### Thuật toán 3: Độ phức tạp $O(n)$ .

Ta có thể giảm số trường hợp phải thử từ  $O(n^2)$  về  $O(n)$  bằng nhận xét sau: nếu  $(x_1, x_2)$  thoả mãn thì  $(x_1, x_2 + 1)$  cũng thoả mãn, như vậy với mỗi  $x_1$  ta xác định  $x_{\min}$  nhỏ nhất để  $(x_1, x_{\min})$  thoả mãn thì  $x_2 \in [x_{\min}, n]$  đều thoả mãn, có  $n - x_{\min} + 1$  vị trí thoả mãn cho  $x_2$ . Như vậy chỉ cần thử  $x_1$ , số lượng  $x_2$  thoả mãn (tương ứng với  $x_1$ ) dễ dàng tính được.

Với  $x_1$  cách xác định  $x_{\min}$  như sau:  $x_{\min} = \max\{x_1 + 5, x_{HAZ}, x_{LaZ}, x_{09}\}$  trong đó  $x_{HAZ}$  là vị trí nhỏ nhất lớn hơn  $x_1$  mà từ  $x_1$  đến  $x_{HAZ}$  có xuất hiện kí tự thuộc ['A'..'Z'], tương tự  $x_{LaZ}$  là vị trí nhỏ nhất lớn hơn  $x_1$  mà từ  $x_1$  đến  $x_{LaZ}$  có xuất hiện kí tự thuộc ['a'..'z'],  $x_{09}$  là vị trí nhỏ nhất lớn hơn  $x_1$  mà từ  $x_1$  đến  $x_{09}$  có xuất hiện kí tự thuộc ['0'..'9']. Việc xác định  $x_{HAZ}$ ,  $x_{LaZ}$ ,  $x_{09}$  có thể xác định bằng cách chuẩn bị trước.

### Chương trình hoàn chỉnh

```
const MAX = 1000000;
fi ='MATKHAU.INP';
fo ='MATKHAU.OUT';
var next :array[1..MAX,'1'..'3']of longint;
s :ansistring;
n :longint;
count :int64;

procedure readFile;
var f :text;
i :longint;
c :char;
begin
assign(f,fi); reset(f);
readln(f,s);
n:=length(s);
close(f);
for i:=1 to length(s) do
if (s[i]>='0') and (s[i]<='9') then s[i]:='1'
else if (s[i]>='a') and (s[i]<='z') then s[i]:='2'
else s[i]:='3';
for c:='1' to '3' do next[n,c]:=n+1;
next[n,s[n]]:=n;
```

```

for i:=n-1 downto 1 do
begin
  for c:='1' to '3' do next[i,c]:=next[i+1,c];
  next[i,s[i]]:=i;
end;
end;

procedure try;
var i,j :longint;
c :char;
begin
count:=0;
for i:=1 to n-5 do begin
j:=i+5;
for c:='1' to '3' do
if next[i,c]>j then j:=next[i,c];
if (j<=n) then
count:=count + (n-j+1)
end;
end;
procedure writeResult;
var f :text;
begin
assign(f,fo); rewrite(f);
writeln(f,count);
close(f);
end;

BEGIN
readFile;
try;
writeResult;
END.

```

#### 4.49. Dây dẫn

*Thuật toán 1: Độ phức tạp O(nL).*

Ta thử lần lượt độ dài cần tìm ( $x_1$ ) từ nhỏ đến lớn (hoặc lớn về nhỏ), sau đó tiến hành kiểm tra xem có cắt được  $k$  đoạn có độ dài  $x_1$  không?

```

const MAX =100000;
fi ='wires.inp';
fo ='wires.out';
var n,k,res :longint;
l :array[1..MAX]of longint;

procedure readFile;
var f :text;

```

```

i :longint;
begin
assign(f,fi); reset(f);
readln(f,n,k);
for i:=1 to n do readln(f,l[i]);
close(f);
end;

function isOK(x1:longint):boolean;
var i, count : longint;
begin
count:=0;
for i:=1 to n do
count:=count + l[i] div x1;
isOk:=count>=k;
end;

procedure try;
var i,x1,lmax : longint;
sum : int64;
begin
sum:=0;
for i:=1 to n do sum:=sum+l[i];
lmax:=sum div k;
res:=0;
for x1:=1 to lmax do
if isOK(x1) then res:=x1;
end;

procedure writeResult;
var f :text;
begin
assign(f,fo); rewrite(f);
writeln(f,res);
close(f);
end;

BEGIN
readFile;
try;
writeResult;
END.

```

Trong chương trình trên, hàm *isOK* sẽ kiểm tra xem với độ dài  $x_1$  có thể có cách cắt được thành  $k$  đoạn hay không? Chi phí kiểm tra là  $O(n)$ .

Số lượng thử  $x_1$  phụ thuộc vào độ dài của các đoạn dây mà độ dài các đoạn dây có thể lên tới  $10^9$ .

*Thuật toán 2: Độ phức tạp O(nlogL).*

Ta có nhận xét sau: nếu độ dài  $x_1$  có thể cắt được thành  $k$  đoạn thì đương nhiên ta có thể cắt được thành  $k$  đoạn có độ dài  $x_1 - 1$ . Từ nhận xét này ta sẽ giảm thiểu số trường hợp phải thử bằng thuật toán chia nhị phân, cụ thể thay thủ tục *try* bằng thủ tục *bs\_try* như sau:

```
procedure bs_try;
var i,x1,lmin,lmax :longint;
    sum :longint;
begin
    sum:=0;
    for i:=1 to n do sum:=sum+l[i];
    lmax:=sum div k;
    lmin:=1;
    while lmax>lmin do begin
        x1:=(lmax+lmin) div 2;
        if isOK(x1) then begin
            res:=x1;
            lmin:=x1 + 1;
        end
        else lmax:=x1 - 1;
    end;
end;
```

## 4.50. Lucky Numbers

Liệt kê tất cả các số có dạng 8...86...6 tăng dần theo giá trị bằng hai vòng lặp nhau: vòng số lượng chữ số tăng dần ( $l = 1..200$ ) và số lượng chữ số 8 tăng dần ( $t = 0..l$ ). Xây dựng hàm chia số lớn cho số nhỏ để kiểm tra tính chia hết.

## 4.51. ACM

Bản chất bài toán là cần chia tập 11 chủ đề thành ba tập để mỗi học sinh được chọn phụ trách một tập.

*Bước 1:* Chuẩn bị dữ liệu, cụ thể với một tập con (có  $2^{11}$  tập con, kể cả tập rỗng) do một học sinh phụ trách thì đạt được điểm lớn nhất là bao nhiêu.

*Bước 2:* Duyệt các chọn ba học sinh, cụ thể duyệt tập  $x$  (do học sinh thứ nhất phụ trách), duyệt tập  $y$  (do học sinh thứ hai phụ trách), tập  $z$  là tập còn lại (tập 11 chủ đề loại trừ tập  $x$ , tập  $y$ ).

## 4.52. Phần thưởng

```
const MAX =1000;
LIMIT =1000 * 1000 * 1000;
fi ='bonus2.inp';
fo ='bonus2.out';

var a,s :array[0..MAX,0..MAX]of longint;
u,d,l,r :array[1..MAX]of longint;
n,k :longint;
best :longint;

procedure docf;
var f :text;
i,j :longint;
begin
fillchar(s,sizeof(s),0);
assign(f,fi); reset(f);
readln(f,n,k);
for i:=1 to n do
for j:=1 to n do
begin
read(f,a[i,j]);
s[i,j]:=s[i-1,j]+s[i,j-1]-s[i-1,j-1] + a[i,j];
end;
close(f);
end;

procedure tim;
var i,j :longint;
sum :longint;
begin
for i:=1 to n do
begin
u[i]:=-LIMIT;
d[i]:=-LIMIT;
l[i]:=-LIMIT;
r[i]:=-LIMIT;
end;
for i:=1 to n-k+1 do
for j:=1 to n-k+1 do begin
sum:=s[i+k-1,j+k-1]-s[i+k-1,j-1]-s[i-1,j+k-1]+s[i-1,j-1];
if sum>u[i+k-1] then u[i+k-1]:=sum;
if sum>d[i] then d[i]:=sum;
if sum>l[j+k-1] then l[j+k-1]:=sum;
if sum>r[j] then r[j]:=sum;
end;
for i:=2 to n do begin
if u[i]<u[i-1] then u[i]:=u[i-1];
if l[i]<l[i-1] then l[i]:=l[i-1];
end;

```

```

end;
for i:=n-1 downto 1 do begin
if d[i]<d[i+1] then d[i]:=d[i+1];
if r[i]<r[i+1] then r[i]:=r[i+1];
end;
best:=-1;
for i:=1 to n-1 do begin
if u[i]+d[i+1]>best then best:=u[i]+d[i+1];
if l[i]+r[i+1]>best then best:=l[i]+r[i+1];
end;
end;

procedure ghikq;
var f :text;
begin
assign(f,fo); rewrite(f);
writeln(f,best);
close(f);
end;

BEGIN
docf;
tim;
ghikq;
END.

```

### 4.53. Bốc sỏi

#### *Phân tích*

Với Subtask 1, ta có thể sử dụng ba tham số  $a, b, c$  để mô tả bài toán và  $f(a, b, c)$  là số cách khác nhau để bốc được hết tất cả các đồng sỏi. Dùng mảng  $F[0..30, 0..30, 0..30]$  để lưu kết quả của các bài toán con  $f(a, b, c)$ .

Để thuận tiện khi viết chương trình và giải được Subtask , ta dùng  $x: array[1..5] of longint$  để mô tả bài toán (trường hợp  $n < 5$  thì  $x[n + 1], \dots, x[5]$  bằng 0) và dùng mảng  $F$  có 5 chiều (mỗi chiều từ 0 đến 30) để lưu trữ. Đoạn lệnh để tính cụ thể như sau:

```

function qhd(var x:prob):longint;
var i,j :longint;
z :prob;
tmp :longint;
begin
if f[x[0],x[1],x[2],x[3],x[4]]<>-1 then
exit(f[x[0],x[1],x[2],x[3],x[4]]);
tmp:=0;
for i:=0 to n-1 do begin

```

```

j:=(i+1) mod n;
z:=x;
if (z[i]>0) and (z[j]>0) then begin
if odd(z[i]) and odd(z[j]) then begin
dec(z[i]); dec(z[j]);
tmp:=(tmp + qhd(z)) mod NMOD;
end;
z[i]:=x[i] div 2; z[j]:=x[j] div 2;
tmp:=(tmp + qhd(z)) mod NMOD;
end;
end;
f[x[0],x[1],x[2],x[3],x[4]]:=tmp;
exit(tmp);
end;

```

Số bài toán con theo thuật toán trên là  $a_1 \times a_2 \times \dots \times a_n$ , chi phí chuyển trạng thái  $O(n)$ . Do đó, độ phức tạp theo thuật toán trên là  $O(a_1 \times a_2 \times \dots \times a_n)$ . Như vậy, chương trình trên có thể đáp ứng cho Subtask 2. Với Subtask 3, ta có nhận xét sau: xét một đồng cỗ số sỏi bằng  $a$ , có hai trường hợp xảy ra:

- nếu  $a = 2k + 1$ , ta có thể bóc 1 viên để đưa đồng sỏi về còn  $2k$ , sau đó sẽ về còn  $k$  viên;
- nếu  $a = 2k$ , chỉ có cách bóc để đưa đồng sỏi về còn  $k$  viên.

Nhận thấy các khả năng  $k + 1, k + 2, \dots, 2k - 1$  không xảy ra. Như vậy, bản chất độ phức tạp thuật toán trên là  $O(n \times \log a_1 \times \log a_2 \times \dots \times \log a_n)$  và có thể giải với  $a_i \leq 30000$ . Nhưng vấn đề khó khăn là không thể khai báo được mảng 5 chiều, mỗi chiều kích thước 30000 để lưu trữ được. Để giải quyết vấn đề này, với từng đồng sỏi ta xét các giá trị có thể xảy ra rồi đánh thứ tự các giá trị này từ nhỏ đến lớn, việc lưu trữ sẽ theo số thứ tự giá trị. Chương trình hoàn chỉnh dưới đây.

```

const MAX =5;
LIMIT =28;
NMOD =111539786;
fi ='bocsroi.inp.';
fo ='bocsroi.out';
type prob =array[0..MAX]of longint;
var s :prob;
idx :array[0..30000]of longint;
f :array[0..LIMIT,0..LIMIT,0..LIMIT,0..LIMIT] of
longint;
n :longint;
ft,gt :text;

```

```

procedure chuanbi;
var i, maxIdx :longint;

function getIndex(i:longint):longint;
begin
  if idx[i]<>-1 then exit(idx[i]);
  if odd(i) then idx[i]:=getIndex(i-1)+1
  else idx[i]:=getIndex(i div 2)+1;
  exit(idx[i]);
end;
begin
  fillchar(idx,sizeof(idx),255);
  idx[0]:=0;
  maxIdx:=0;
  for i:=30000 downto 0 do begin
    idx[i]:=getIndex(i);
    if idx[i]>maxIdx then maxIdx:=idx[i];
  end;
  // writeln(maxIdx);
end;

function qhd(var x:prob):longint;
var i, j :longint;
y, z :prob;
tmp :longint;
begin
  for i:=0 to 4 do y[i]:=idx[x[i]];
  if f[y[0],y[1],y[2],y[3],y[4]]<>-1 then
    exit(f[y[0],y[1],y[2],y[3],y[4]]);
  tmp:=0;
  for i:=0 to n-1 do begin
    j:=(i+1) mod n;
    z:=x;
    if (z[i]>0) and (z[j]>0) then begin
      if odd(z[i]) and odd(z[j]) then begin
        dec(z[i]); dec(z[j]);
        tmp:=(tmp + qhd(z)) mod NMOD;
      end;
      z[i]:=x[i] div 2; z[j]:=x[j] div 2;
      tmp:=(tmp + qhd(z)) mod NMOD;
    end;
  end;
  f[y[0],y[1],y[2],y[3],y[4]]:=tmp;
  exit(tmp);
end;
procedure xuly;
var i :longint;
begin
  fillchar(f,sizeof(f),255);

```

```

f[0,0,0,0,0]:=1;
fillchar(s,sizeof(s),0);
readln(ft,n);
for i:=0 to n-1 do
read(ft,s[i]);
writeln(gt,qhd(s));
end;
BEGIN
chuambi;
assign(ft,fi); reset(ft);
assign(gt,fo); rewrite(gt);
xuly;
close(ft);
close(gt);
END.

```

#### 4.54. Kỉ lục đồ đốm minô

*Cách giải cho công đoạn 1:* Quân đốm minô với trạng thái  $(i, j, s)$  có nghĩa là xếp đến ô  $(i, j)$  (xếp lần lượt từ trên xuống dưới, từ trái qua phải) với  $s$  là dãy gồm  $n$  bit 0, 1 mô tả trạng thái  $j - 1$  ô đầu ở dòng  $i$  và  $(n - j + 1)$  ô cuối ở dòng  $i - 1$ . Để thuận tiện lưu trữ,  $s$  sẽ được mã hoá là một số nguyên  $t$  ( $0 \leq t \leq 2^N - 1$ ) mà trong biểu diễn nhị phân của  $t$  là dãy  $s$ .

*Cách giải cho công đoạn 2:* Quân đốm minô với trạng thái  $(i, t)$  có nghĩa là xếp đến hàng  $i$  có trạng thái hàng trước là  $t$  với  $t$  là dãy  $n$  bit 0, 1 thoả mãn tích chất màu trên một dòng (với  $R = 8$  thì  $t \leq 55$ ). Dùng nhân ma trận để tính nhanh.

```

const MAXN =8;
MAXN2 =16;
MAXSL =55;
fi ='DOMINO.INP';
fo ='DOMINO.OUT';
type dsType =array[1..MAXN]of longint;
matrixType =array[1..MAXSL,1..MAXSL]of longint;
var m,n,nMod,SL:longint;
ds :array[1..2,1..1 shl MAXN]of dsType;
count :array[1..2]of longint;
A, B, AB :matrixType;
x :dsType;
sum :longint;
f,g :text;
dp :array[1..MAXN2+1,1..MAXN2,0..1 shl MAXN2]of longint;
w :array[1..MAXN2,1..MAXN2]of longint;
mu2 :array[1..MAXN2]of longint;
function check(t:longint):boolean;

```

```

var i :longint;
begin
for i:=2 to n do begin
if ((i+t) mod 2 = 1)and(x[i]<x[i-1]) then exit(false);
if ((i+t) mod 2 = 0)and(x[i]>x[i-1]) then exit(false);
end;
exit(true);
end;
procedure try(i :longint);
var c : longint;
begin
if i>n then begin
for c:=1 to 2 do
if check(c) then begin
inc(count[c]); ds[c][count[c]]:=x;
end;
end
else begin
for c:=0 to 1 do
begin
x[i]:=c;
try(i+1);
end;
end;
end;
function ghep(j1,j2,i:longint):boolean;
var j :longint;
begin
for j:=1 to n do begin
if ((i+1+j) mod 2=1)and(ds[(i+1) mod 2+1][j1][j]>ds[i mod
2+1][j2][j]) then exit(false);
if ((i+1+j) mod 2=0)and(ds[(i+1) mod 2+1][j1][j]<ds[i mod
2+1][j2][j]) then exit(false);
end;
exit(true);
end;
function mulMatrix(X, Y: matrixType):matrixType;
var i,j,k :longint;
Z :matrixType;
begin
fillchar(Z,sizeof(Z),0);
for i:=1 to SL do
for j:=1 to SL do
for k:=1 to SL do Z[i,j]:=(Z[i,j]+int64(X[i,k])*Y[k,j]) mod
nMod;
exit(Z);
end;
function expMatrix(X: matrixType; e:longint):matrixType;
var Y :matrixType;

```

```

begin
if e=1 then exit(X);
Y:=expMatrix(X,e div 2);
Y:=mulMatrix(Y,Y);
if e mod 2 = 1 then Y:=mulMatrix(Y,X);
exit(Y);
end;

procedure chuanbi;
var i,j :longint;
begin
count[1]:=0; count[2]:=0;
try(1);
SL:=count[1];
fillchar(A,sizeof(A),0);
for i:=1 to SL do
for j:=1 to SL do
A[i,j]:=ord(ghep(i,j,1));
fillchar(B,sizeof(B),0);
for i:=1 to SL do
for j:=1 to SL do
B[i,j]:=ord(ghep(i,j,2));
AB:=mulMatrix(A,B);
end;

procedure qhd2;
var i,j :longint;
begin
m:=m-1;
if m>1 then begin
AB:=expMatrix(AB,m div 2);
if m mod 2 = 1 then AB:=mulMatrix(AB,A);
sum:=0;
for i:=1 to SL do
for j:=1 to SL do sum:=(sum + AB[i,j]) mod nMod;
end
else sum:=SL;
end;

procedure xuly2;
begin
readln(f,n,m);
chuanbi;
qhd2;
writeln(g,sum);
end;

function check1(i,j,t,c:longint):boolean;
begin

```

mod

mod

mod

```

if (i+j) mod 2 = 1 then begin
  if (j>1)and(c=0)and(t and mu2[j-1]>0)and(w[i,j-1]=0) then
    exit(false);
  if (i>1)and(c=0)and(t and mu2[j]>0)and(w[i-1,j]=0) then
    exit(false);
end else begin
  if (j>1)and(c=1)and(t and mu2[j-1]=0)and(w[i,j-1]=0) then exit(false);
  if (i>1)and(c=1)and(t and mu2[j]=0)and(w[i-1,j]=0) then exit(false);
end;
exit(true);
end;

function tinh(i,j,t :longint):longint;
var c : longint;
sum : longint;
begin
if dp[i,j,t]>=0 then exit(dp[i,j,t]);
if i>m then begin
  dp[i,j,t]:=1;
  exit(1);
end;
if w[i,j]=1 then begin
  if j<n then dp[i,j,t]:=tinh(i,j+1,t)
  else dp[i,j,t]:=tinh(i+1,1,t);
  exit(dp[i,j,t]);
end;
sum:=0;
for c:=0 to 1 do
  if check1(i,j,t,c) then
begin
  if j<n then sum:=sum + tinh(i,j+1,(t or mu2[j])-(1-c)*mu2[j])
  else sum:=sum + tinh(i+1,1,(t or mu2[j])-(1-c)*mu2[j]);
end;
dp[i,j,t]:=sum mod nMod;
exit(dp[i,j,t]);
end;

procedure xuly1;
var i,j,k,T :longint;
begin
fillchar(w,sizeof(w),0);
read(f,m,n,T);
for k:=1 to T do begin
  read(f,i,j);
  w[i,j]:=1;
end;
mu2[1]:=1;
for k:=2 to MAXN2 do mu2[k]:=mu2[k-1]*2;
fillchar(dp,sizeof(dp),255);

```

```

then
then
e);
;
BEGIN
  assign(f, fi); reset(f);
  assign(g, fo); rewrite(g);
  readln(f, nMod);
  xuly1;
  xuly2;
  close(f);
  close(g);
END.

```

#### 4.55. VACCINE

Công việc chính của bài toán đó là tìm xâu con chung của một xâu  $s$  có độ dài lớn (lên tới 10000 kí tự) với một xâu  $w$  có độ dài nhỏ (không vượt quá 500 kí tự).

Cách giải thông thường gọi  $f(i, j) = k$  là độ dài xâu con chung dài nhất của xâu  $s_1..s_i$  và  $w_1..w_j$ . Thuật toán có độ phức tạp  $O(\text{length}(s) * \text{length}(w))$ .

Cách giải tốt hơn: gọi  $f(j, k)$  là vị trí  $i$  nhỏ nhất trên xâu  $s$  để độ dài xâu con chung dài nhất của xâu  $s_1s_2..s_i$  và  $w_1w_2..w_j$  bằng  $k$ .

$f(j, k) = i$  tính nhän cho  $f(j + 1, k) = i$  và  $f(j + 1, k + 1) = i'$  trong đó  $i'$  là vị trí đầu tiên sau  $i$  mà  $s[i] = w[j + 1]$ .

Thuật toán có độ phức tạp  $O(\text{length}(w) * \text{length}(w))$ .

```

{$H+}
const MAXM = 500;
MAXN = 10000;
fi ='VIRUS.INP';
fo ='VIRUS.OUT';
var w :array[0..MAXM+1,0..MAXM+1]of longint;
next :array[0..MAXN+1,'A'..'Z']of longint;
a,b :string;
f,g :text;
na,nb,T,alpha :longint;
mk,mkMin :longint;
count :longint;
c :char;
nTest :longint;
tong,kq :longint;

procedure makeNext(c1,c2:char);
var c :char; j:longint;
begin
  fillchar(next,sizeof(next),0);

```

```

for c:=c1 to c2 do begin
  next[nb,c]:=nb+1; next[nb+1,c]:=nb+1;
  for j:=nb-1 downto 0 do
    if b[j+1]=c then next[j,c]:=j+1
    else next[j,c]:=next[j+1,c];
  end;
end;

function tinh:longint;
var i,j :longint;
begin
  for i:=0 to na do begin
    for j:=0 to na do w[i,j]:=nb+1;
    w[i,0]:=0;
  end;
  mk:=0;
  for i:=0 to na-1 do begin
    if alpha+1-na+i>0 then mkMin:=alpha+1-na+i
    else mkMin:=0;
    for j:=mk downto mkMin do begin
      if w[i+1,j]>w[i,j] then w[i+1,j]:=w[i,j];
      if w[i+1,j+1]>next[w[i,j],a[i+1]] then begin
        if j+1>mk then mk:=j+1;
        w[i+1,j+1]:=next[w[i,j],a[i+1]];
      end;
    end;
  end;
  exit(mk);
end;

BEGIN
  assign(f,fi); reset(f);
  assign(g,fo); rewrite(g);
  readln(f,nTest);
  while nTest>0 do
  begin
    dec(nTest); tong:=0;
    count:=0;
    alpha:=0;
    readln(f,T);
    readln(f,b);
    nb:=length(b);
    makeNext('A','Z');
    while T>0 do begin
      readln(f,a); na:=length(a);
      dec(T);
      kq:=tinh;
      if kq>alpha then alpha:=kq;
    end;
  end;

```

```

writeln(g,alpha);
end;
close(f);
close(g);
END.

```

#### 4.56. Cổ phiếu

Gọi  $L[d, i_1, i_2, i_3]$  là số tiền nhiều nhất đến hết ngày  $d$  và đang có  $i_1$  cổ phiếu loại 1,  $i_2$  cổ phiếu loại 2,  $i_3$  cổ phiếu loại 3 (nếu chỉ có một loại cổ phiếu thì  $Q_2 = Q_3 = 0$ ; nếu chỉ có hai loại cổ phiếu thì  $Q_3 = 0$ ).

Hành động bán cổ phiếu: Từ  $L[d, i_1, i_2, i_3]$  chuyển nhãn cho  $L[d, i_1 - 1, i_2, i_3]$ ,  $L[d, i_1, i_2 - 1, i_3]$  hoặc  $L[d, i_1, i_2, i_3 - 1]$ .

Hành động mua cổ phiếu: Từ  $L[d, i_1, i_2, i_3]$  chuyển nhãn cho  $L[d, i_1 + 1, i_2, i_3]$ ,  $L[d, i_1, i_2 + 1, i_3]$  hoặc  $L[d, i_1, i_2, i_3 + 1]$ .

Chuyển nhãn sang ngày mới: Từ  $L[d, i_1, i_2, i_3]$  chuyển nhãn cho  $L[d + 1, i_1, i_2, i_3]$ .

Tuy nhiên kích thước mảng  $L$  quá lớn, thông thường sử dụng mảng  $L[0..1, *, *, *]$  tính cho nhau. Trong bài toán này chỉ cần dùng một mảng trực tiếp mảng  $L[*, *, *, *]$ .

Độ phức tạp thuật toán:  $O(D \cdot Q_1 \cdot Q_2 \cdot Q_3)$ .

```

const max =100;
fi ='stock.inp';
fo ='stock.out';

var l :array[0..max,0..max,0..max]of int64;
q :array[1..3]of longint;
cost :array[1..10000,1..3]of int64;
n,m,t :int64;

procedure ReadIn;
var f :text;
i,j :longint;
begin
assign(f,fi); reset(f);
readln(f,n,m,t);
for i:=1 to n do read(f,q[i]);
for i:=1 to t do
for j:=1 to n do read(f,cost[i,j]);
close(f);
for i:=n+1 to 3 do q[i]:=0;

```

```

n:=3;
end;

procedure Process;
var i1,i2,i3,k :longint;
money :int64;
begin
for i1:=0 to q[1] do
for i2:=0 to q[2] do
for i3:=0 to q[3] do l[i1,i2,i3]:=-1;
l[0,0,0]:=m;
for k:=1 to t do begin
// ban co phieu
for i1:=q[1] downto 0 do
for i2:=q[2] downto 0 do
for i3:=q[3] downto 0 do begin
money:=l[i1,i2,i3];
if money > -1 then
begin
if (i1>0) and (l[i1-1,i2,i3] < money+cost[k,1]-1) then
l[i1-1,i2,i3] := money+cost[k,1]-1;
if (i2>0) and (l[i1,i2-1,i3] < money+cost[k,2]-1) then
l[i1,i2-1,i3] := money+cost[k,2]-1;
if (i3>0) and (l[i1,i2,i3-1] < money+cost[k,3]-1) then
l[i1,i2,i3-1] := money+cost[k,3]-1;
end;
end;
// mua co phieu
for i1:=0 to q[1] do
for i2:=0 to q[2] do
for i3:=0 to q[3] do begin
money:=l[i1,i2,i3];
if money > -1 then
begin
if (i1<q[1]) and (money > cost[k,1]) and (l[i1+1,i2,i3] <
money-cost[k,1]-1) then
l[i1+1,i2,i3] := money-cost[k,1]-1;
if (i2<q[2]) and (money > cost[k,2]) and (l[i1,i2+1,i3] <
money-cost[k,2]-1) then
l[i1,i2+1,i3] := money-cost[k,2]-1;
if (i3<q[3]) and (money > cost[k,3]) and (l[i1,i2,i3+1] <
money-cost[k,3]-1) then
l[i1,i2,i3+1] := money-cost[k,3]-1;
end;
end;
end;
end;
procedure WriteOut;
var f :text;

```

```

begin
  assign(f,fo); rewrite(f);
  write(f,l[0,0,0]);
  close(f);
end;
BEGIN
  ReadIn;
  Process;
  WriteOut;
END.

```

#### 4.57. Đồ chơi XYZ

```

const max =30;
fi ='xyz.inp';
fo ='xyz.out';
c1 =char(65);
c2 =char(65+31);
Nmod =1000000;
type arr_vt =array[0..max,c1..c2]of longint;
var W3:array[0..max,0..max,0..max]of int64;
W2:array[0..max,0..max]of int64;
mu :array[0..max]of int64;
vt1,vt2 :arr_vt;
A, B :string;
n,l :longint;
kq :int64;

procedure tao_vt(var s:string;var vt:arr_vt);
var j :longint;
c :char;
p :string;
begin
for j:=0 to l-1 do
for c:=c1 to c2 do
begin
p:=copy(s,l,j)+c;
while (length(p)>0) and (copy(s,l,length(p))<>p) do delete(p,1,1);
vt[j,c]:=length(p);
end;
for c:=c1 to c2 do vt[l,c]:=l;
end;

procedure docf;
var f :text;
s :array[1..5]of string;
i,j,p :longint;
begin
assign(f,fi); reset(f);

```

```

readln(f,n,l);
a:=''; b:='';
for i:=1 to 5 do readln(f,s[i]);
for i:=1 to l do begin
p:=0;
for j:=1 to 5 do p:=p*2+ord(s[j,i]='#');
a:=a+chr(p+65);
end;
for i:=1 to 5 do readln(f,s[i]);
for i:=1 to l do begin
p:=0;
for j:=1 to 5 do p:=p*2+ord(s[j,i]='#');
b:=b+chr(p+65);
end;
close(f);
end;

function chual(var s:string; var vt:arr_vt):int64;
var i,j :longint;
c :char;
count :int64;
begin
fillchar(W2,sizeof(W2),0);
W2[0,0]:=1;
for i:=0 to n-1 do
for j:=0 to l-1 do
for c:=c1 to c2 do
W2[i+1,vt[j,c]]:=(W2[i+1,vt[j,c]]+W2[i,j]) mod Nmod;
count:=0;
for i:=1 to n do
if W2[i,1]>0 then count:=(count + W2[i,1] * mu[n-i]) mod
Nmod;
exit(count);
end;

function chua2_ab:int64;
var i,j,k :longint;
c :char;
count :int64;
begin
fillchar(W3,sizeof(W3),0);
W3[0,0,0]:=1;
for i:=0 to n-1 do
for j:=0 to l do
for k:=0 to l do
if j+k<2*i then
for c:=c1 to c2 do
W3[i+1,vt1[j,c],vt2[k,c]]:=(W3[i+1,vt1[j,c],vt2[k,c]]+W3[i,j,
k]) mod Nmod;

```

```

count:=0;
for i:=1 to n do
if W3[i,1,1]>0 then count:=(count + W3[i,1,1] * mu[n-i]) mod Nmod;
exit(count);
end;
procedure tinh;
var ca,cb,cab :int64;
i :longint;
f :text;
begin
tao_vt(a,vt1);
tao_vt(b,vt2);
mu[0]:=1;
for i:=1 to n do mu[i]:=(mu[i-1]*32) mod Nmod;
ca:=chual(a,vt1);
cb:=chual(b,vt2);
cab:=chua2_ab;
assign(f,fo); rewrite(f);
writeln(f,(2*Nmod+ca+cb-2*cab) mod Nmod);
close(f);
end;
BEGIN
docf;
tinh;
END.

```

**4.58.** Gọi  $f[i]$  là số cách xếp các loại hình của cây có độ sâu là  $i$ . Khi tính  $f[i]$  tiến hành thử đặt các loại hình được sử dụng để tính, cụ thể:

- Nếu sử dụng loại 1 thì có  $f[i - 1] * f[i - 1]$  cách;
- Nếu sử dụng loại 2 thì có  $f[i - 2] * f[i-2] * f[i - 2] * f[i - 2]$  cách;
- Nếu sử dụng loại 3 (hoặc loại 4) thì có  $f[i - 1] * f[i - 2] * f[i - 3] * f[i - 4] * f[i - 4]$  cách;
- Nếu sử dụng loại 5 thì có  $f[i - 2] * f[i - 3] * f[i - 3] * f[i - 2] * f[i - 3] * f[i - 3]$  cách

**4.59.**

```

const      fi    ='nembong.inp';
            fo    ='nembong.out';
            max   =101;

var       l    :array[-1..max,-1..max] of longint;
a,b    :array[1..max] of longint;
n,kq,luu,w,c,dem :longint;
f,g    :text;

```

```

Procedure docdl;
var i :longint;
begin
  read(f,w,c,n);
  for i:=1 to n do read(f,a[i]);
end;

function maxkq(x,y:longint):longint;
begin
  if x>y then exit(x) else exit(y);
end;

Procedure taolop(x:longint);
begin
  dem:=1;
  b[dem]:=x;
  while b[dem]+w <=n do
    begin
      inc(dem);
      b[dem]:=b[dem-1]+w;
    end;
end;

function maxtrai(x:longint):longint;
var i,tg,tong :longint;
begin
  tg:=-maxlongint;
  tong:=0;
  for i:=1 to dem do
    begin
      tong:=tong+a[b[i]];
      if tong>tg then tg:=tong;
    end;
  maxtrai:=tg;
end;

function maxphai(x:longint):longint;
var i,tg,tong :longint;
begin
  tg:=-maxlongint;
  tong:=0;
  for i:=dem downto 1 do
    begin
      tong:=tong+a[b[i]];
      if tong>tg then tg:=tong;
    end;
  maxphai:=tg;
end;

```

```

function nem2lan(x:longint):longint;
var i,j,tong,tg,k :longint;
begin
    tg:=-maxlongint;
    for i:=1 to dem-1 do
        for j:=dem downto i+1 do
            begin
                tong:=0;
                for k:=1 to i do tong:=tong+a[b[k]];
                for k:=dem downto j do
                    tong:=tong+a[b[k]];
                    if tong>tg then tg:=tong;
            end;
    nem2lan:=tg;
end;

Procedure xuli;
var i,j,k,tg :longint;
begin
    fillchar(l,sizeof(l),0);
    for i:=1 to w do
        for j:=1 to c do
            begin
                taolop(i);
                for k:=0 to 2 do
                    case k of
                        0: l[i,j]:=l[i-1,j];
                        1: if j>=1 then
                            begin
                                tg:=maxtrai(i);
                                tg:=maxkq(tg,maxphai(i));
                                l[i,j]:=maxkq(l[i,j],l[i-1,j-1]+tg);
                            end;
                        2: if j>=2 then
                            begin
                                tg:=nem2lan(i);
                                l[i,j]:=maxkq(l[i,j],l[i-1,j-2]+tg);
                            end;
            end;
    end;
Procedure ghikq;
var i :longint;
begin
    kq:=0;luu:=0;
    for i:=1 to c do
        if l[w,i]>kq then

```

```

begin
    kq:=l[w,i];
    luu:=i;
end;
writeln(g,kq,' ',luu);
end;

BEGIN
    assign(f,fi);
    reset(f);
    assign(g,fo);
    rewrite(g);

    repeat
        docdl;
        if (w=0) and (c=0) and (n=0) then break;
        xuli;
        ghikq;
    until false;

    close(f);
    close(g);
END.

```

#### 4.60. Nối điểm

Cách 1: Quy về bài toán tìm dãy con chung dài nhất nhưng với quan niệm số thứ i của dãy A bằng số thứ j của dãy B là  $\text{ABS}(A[i]-B[j]) \leq 1$ .

Độ phức tạp thuật toán  $O(N^2)$ .

Cách 2: Quy về bài toán tìm dãy con tăng dài nhất. Tạo dãy E gồm  $3n$  phần tử từ dãy D như sau: Với mỗi số  $D[i]$  được thay bằng ba số  $D[i]+1, D[i], D[i]-1$ . Tìm dãy con tăng dài nhất trên dãy E.

Độ phức tạp thuật toán  $O(N \log N)$ .

#### 4.61. Xâu gần nhất

$f(p, i, j, k) = \frac{\text{true}}{\text{false}}$  có ý nghĩa là có/không thể xây dựng được xâu đến vị trí  $p$  mà có khoảng cách hiện tại với xâu thứ nhất là  $i$ , xâu thứ hai là  $j$ , xâu thứ ba là  $k$ . Dựa vào bảng quy hoạch động để lần ra kết quả.

Độ phức tạp thuật toán  $O(N^4)$ .

Chú ý: có thể giảm thời gian tính toán xuống bằng cách giới hạn  $i, j, k \leq \left\lceil \frac{2n}{3} \right\rceil$ .

Một cách cài đặt bằng mô hình duyệt đệ quy có ngắt nhánh, sử dụng mảng đánh dấu những trạng thái đã gặp cũng cho độ phức tạp  $O(N^4)$ .

```
const MAX = 100;
fi ='cstr.inp';
fo ='cstr.out';
var s :array[1..3]of string;
cs :array[1..MAX]of string;
n :longint;
best :longint;
bestkq,kq :string;
dd :array[1..MAX,0..MAX,0..MAX,0..MAX]of byte;

procedure docf;
var f :text;
i :longint;
begin
assign(f,fi); reset(f);
for i:=1 to 3 do readln(f,s[i]);
close(f);

n:=length(s[1]);
for i:=1 to n do
cs[i]:=s[1,i] + s[2,i] + s[3,i] + 'A';
end;

procedure capnhat(kq:string; c1, c2, c3:longint);
var min :longint;
begin
min:=c1;
if min<c2 then min:=c2;
if min<c3 then min:=c3;
if min<best then begin
best:=min;
bestkq:=kq;
end;
end;

procedure duyet(i:longint; c1, c2, c3:longint);
var c :char; e1, e2, e3 :longint;
begin
if (c1>best) or (c2>best) or (c3>best) or (dd[i,c1,c2,c3]>0)
then exit;
if (dd[i,c1,c2,c3]>0) then exit;
if i>n then begin
capnhat(kq,c1,c2,c3);

```

```

exit;
end;
for c:='A' to 'Z' do
if pos(c,cs[i])>0 then begin
kq:=kq + c;
if c=s[1][i] then e1:=c1 else e1:=c1+1;
if c=s[2][i] then e2:=c2 else e2:=c2+1;
if c=s[3][i] then e3:=c3 else e3:=c3+1;
duyet(i+1,e1,e2,e3);
delete(kq,i,1);
end;
dd[i,c1,c2,c3]:=1;
end;

procedure ghikq;
var f :text;
begin
assign(f,fo); rewrite(f);
writeln(f,bestkq);
close(f);
end;

BEGIN
docf;
best:=(2*n) div 3 + 2;
kq:='';
duyet(1,0,0,0);
writeln(bestkq, ' ',best);
ghikq;
END.

```

#### 4.62. Password

```

const MAX =100+10;
fi ='password.inp';
fo ='password.out';
type mystr =ansistring;
var num :array[1..MAX]of longint;
s :array[1..MAX]of mystr;
l :array[0..MAX,0..MAX,0..8]of mystr;
d :array[0..MAX,0..MAX,0..8]of longint;
n,m :longint;
T :longint;

procedure docfile;
var i,j,p :longint;
tmp :string;
begin
for i:=1 to n do begin

```

```

readln(num[i]);
str(num[i],s[i]);
end;
for i:=1 to n-1 do
for j:=i+1 to n do
if s[i]+s[j]<s[j]+s[i] then begin
tmp:=s[i];
s[i]:=s[j];
s[j]:=tmp;
p:=num[i];
num[i]:=num[j];
num[j]:=p;
end;
end;

function sosanh(p,q:mystr):boolean;
begin
if length(p)>length(q) then exit(true);
if (length(p)=length(q)) and (p>q) then exit(true);
exit(false);
end;

procedure tim;
var i,j,k :longint;
p,q :mystr;
begin
fillchar(d,sizeof(d),0);
fillchar(l,sizeof(l),0);
d[0,0,0]:=1;
for i:=0 to n-1 do
for j:=0 to i do
for k:=0 to 8 do
if d[i,j,k]=1 then
begin
p:=l[i,j,k]+s[i+1];
q:=l[i+1,j+1,(k+num[i+1]) mod 9];
if sosanh(p,q) then q:=p;
l[i+1,j+1,(k+num[i+1]) mod 9]:=q;
d[i+1,j+1,(k+num[i+1]) mod 9]:=1;
p:=l[i,j,k];
q:=l[i+1,j,k];
if sosanh(p,q) then q:=p;
l[i+1,j,k]:=q;
d[i+1,j,k]:=1;
end;
end;

BEGIN
assign(input,fi); reset(input);

```

```

assign(output,fo); rewrite(output);
readln(T);
while T>0 do begin
dec(T);
readln(n,m);
docfile;
tim;
if d[n,m,0]=0 then writeln(-1)
else writeln(l[n,m,0]);
end;
close(input);
close(output);
END.

```

#### 4.63.

```

const nfin = 'QLKHO.INP';
nfout = 'QLKHO.OUT';
maxn = 1010;
oo = 1000000010;
var f:array[0..maxn, 0..maxn] of longint;
p:array[0..maxn] of longint;
r1, r2:longint;
m, n:longint;
fin, fout:text;

function min(x, y: longint): longint;
begin
if x<=y then exit(x) else exit(y);
end;

function check(v: longint): boolean;
var i, sl: longint;
begin
sl:=n;
for i:=1 to m do begin
    sl:=sl - p[i] div v;
    if sl <= 0 then exit(true);
end;
exit(false);
end;

Procedure solve;
var i, j, k, l, r, mid: longint;
begin
for i:=1 to m do read(fin, p[i]);
readln(fin);
r1:=0; l:=1; r:=1000000;
while l<=r do begin

```

```

        mid:=(l+r) shr 1;
        if check(mid) then begin
            l:=mid+1;
            if r1<mid then r1:=mid;
        end else r:=mid-1;
    end;
    if r1=0 then begin
        r2:=0; exit;
    end;
    for i:=0 to n do
        for j:=0 to m do f[i, j]:=oo;
    for i:=0 to m do f[0, i]:=0;
    for i:=1 to n do
        for j:=1 to m do begin
            f[i, j]:=f[i, j-1];
            k:=min(i, p[j] div r1);
            f[i, j]:=min(f[i, j], p[j] + f[i-k, j-1]);
        end;
    r2:=f[n, m];
end;

BEGIN
    assign(fin, nfin); reset(fin);
    assign(fout, nfout); rewrite(fout);
    readln(fin, n, m);
    while (n <> 0) or (m <> 0) do begin
        solve;
        writeln(fout, r1, ', ', r2);
        readln(fin, n, m);
    end;
    close(fout);
    close(fin);
END.

```

#### 4.64.

```

const MAX =1000+1;
cs =30;
LIMIT =(1 shl cs)-1;
fi ='computer.inp';
fo ='computer.out';

var a,b,x,y,m,rlist :longint;
w :array[0..MAX,0..MAX]of int64;
f1,f2 :text;
lmin,lmax,lmid :int64;

function qhd(x,y:longint):int64;
var p1,p2 :int64;

```

```

begin
  if w[x,y]<>-1 then exit(w[x,y]);

  if x>0 then begin
    p1:=qhd(x-1,y) shr cs;
    p2:=w[x-1,y] and LIMIT;

    if p2+a>=lmid then begin
      p1:=p1+1; p2:=0;
    end
    else p2:=p2+a;
    if (p1 shl cs) + p2 > w[x,y] then w[x,y]:=(p1 shl cs)+p2;
  end;

  if y>0 then begin
    p1:=qhd(x,y-1) shr cs;
    p2:=w[x,y-1] and LIMIT;

    if p2+b>=lmid then begin p1:=p1+1; p2:=0; end
    else p2:=p2+b;

    if (p1 shl cs)+p2 > w[x,y] then w[x,y]:=(p1 shl cs)+p2;
  end;

  exit(w[x,y]);
end;

Procedure tim;
begin
  lmin:=0;
  lmax:=(x*a + y*b) div m;
  rlst:=0;

  while lmin<=lmax do begin
    lmid:=(lmin+lmax) div 2;
    fillchar(w,sizeof(w),255);
    w[0,0]:=0;
    if qhd(x,y) shr cs >=m then begin
      rlst:=lmid;
      lmin:=lmid + 1;
    end else lmax:=lmid -1;
  end;
  writeln(f2,rlst);
end;

BEGIN
  assign(f1,fi); reset(f1);
  assign(f2,fo); rewrite(f2);
  readln(f1,x,a,y,b,m);

```

```

    tim;
    readln(f1,x,a,y,b,m);
    tim;
    close(f1);
    close(f2);
END.

```

#### 4.65.

```

{$M 10000000}
uses math;

const fi = 'palpath.inp';
      fo = 'palpath.out';
      maxn = 55;
      tx :array[0..2]of longint=(0,1,0);
      ty :array[0..2]of longint=(0,0,1);

Type dinh = record
      x,y:longint;
end;

VAR a :array[1..maxn]of string;
    f :array[0..maxn,0..maxn,0..maxn,0..maxn]of longint;
    tr :array[1..maxn,1..maxn,1..maxn,1..maxn]of dinh;
    kq:array[1..maxn*4]of dinh;
    m,n,nt :longint;

Procedure enter;
Var i:longint;
Begin
  readln(m,n);
  for i:=1 to m do readln(a[i]);
end;

Function find(i,j,k,h:longint):longint;
Var u,v,w:longint;
Begin
  if f[i,j,k,h]=-1 then
    begin
      if(k<i)or(h<j)or(i>m)or(j>n)then
        begin f[i,j,k,h]:=0;exit(0);end;
      w:=ord(a[i,j]=a[k,h]);
      if(i<>k)or(j<>h)then w:=w*2;
      f[i,j,k,h]:=w;
      if w>0 then
        for u:=1 to 2 do
          for v:=1 to 2 do
            if w+find(i+tx[u],j+ty[u],k-tx[v],h-ty[v])>f[i,j,k,h] then
              begin

```

```

f[i,j,k,h]:=w+f[i+tx[u],j+ty[u],k-tx[v],h-ty[v]];
with tr[i,j,k,h] do
begin x:=u;y:=v;end;
end;
(*****)
v:=0;
for u:=1 to 2 do
if find(i+tx[u],j+ty[u],k,h)>f[i,j,k,h] then
begin
f[i,j,k,h]:=f[i+tx[u],j+ty[u],k-tx[v],h-ty[v]];
with tr[i,j,k,h] do
begin x:=u;y:=v;end;
end;
(*****)
u:=0;
for v:=1 to 2 do
if find(i,j,k-tx[v],h-ty[v])>f[i,j,k,h] then
begin
f[i,j,k,h]:=f[i+tx[u],j+ty[u],k-tx[v],h-ty[v]];
with tr[i,j,k,h] do
begin x:=u;y:=v;end;
end;
end;

find:=f[i,j,k,h];
end;

Procedure print;
var i,j,k,h,u,v,p:longint;
nho:dinh;
Begin
i:=1;j:=1;k:=m;h:=n;
nt:=f[1,1,m,n];
p:=1;
while true do
begin
if a[i,j]=a[k,h] then
begin
with kq[p] do begin x:=i;y:=j;end;
with kq[nt-p+1] do begin x:=k;y:=h;end;
inc(p);
end;

nho:=tr[i,j,k,h];
with nho do
begin
if (x=0) and (y=0) then break;
i:=i+tx[x];
j:=j+ty[x];

```

```

        k:=k-tx[y];
        h:=h-ty[y];
    end;
end;

writeln(nt);
for i:=1 to nt do
    with kq[i] do writeln(x, ' ',y);
end;

BEGIN
assign(input,fi);assign(output,fo);
reset(input);rewrite(output);
enter;
fillchar(f,sizeof(f),255);
find(l,1,m,n);
print;
close(input);close(output);
END.

```

#### 4.66.

```

{$r+,q+}
{$m 10000000}
const MAX = 100;
    fi ='option.inp';
    fo ='option.out';
var key :array[1..MAX,1..MAX]of longint;
    l :array[1..MAX,0..MAX]of longint;
    v,m :array[1..MAX]of longint;
    n,k,start:longint;
    best:longint;

Procedure docf;
var f :text;
    i,j :longint;
begin
    assign(f,fi); reset(f);
    readln(f,n,k);
    for i:=1 to n do begin
        read(f,v[i],m[i]);
        for j:=1 to m[i] do read(f,key[i,j]);
    end;
    close(f);
end;

function qhd(node, s:longint):longint;
var w :array[0..MAX,0..MAX]of longint;
    i,j,t :longint;
begin

```

```

if l[node,s]=-1 then begin
  if s=0 then begin
    l[node,s]:=0;
    exit(0);
  end;
  if m[node]=0 then begin
    l[node,s]:=v[node];
    exit(v[node]);
  end;

  for i:=1 to m[node] do
    if key[node,i]<>start then
      for j:=0 to s-1 do qhd(key[node,i],j);
  fillchar(w,sizeof(w),0);
  for i:=1 to m[node] do
    begin
      if key[node,i]=start then begin
        w[i]:=w[i-1];
        continue;
      end;

      for j:=0 to s-1 do
        for t:=0 to j do
          if w[i,j] < w[i-1,t] + l[key[node,i],j-t] then
            w[i,j] := w[i-1,t] + l[key[node,i],j-t];
    end;
    l[node,s]:= v[node] + w[m[node],s-1];
  end;
  qhd:=l[node,s];
end;
Procedure lam;
begin
  best:=0;
  for start:=1 to n do begin
    fillchar(l,sizeof(l),255);
    qhd(start,k);
    if best<l[start,k] then best:=l[start,k];
//    writeln(start,' ',l[start,k]);
  end;
end;

Procedure ghikq;
var f :text;
begin
  assign(f,fo); rewrite(f);
  writeln(f,best);
  close(f);
end;

```

```

BEGIN
    docf;
    lam;
    ghiq;
END .

```

4.67. Tìm bao lồi của tập hợp điểm, giả sử bao lồi có  $m$  đỉnh đánh số từ 1 tới  $m$ .

Nếu  $k \geq m$ , ta sẽ lấy hết  $m$  điểm.

Trường hợp  $k < m$ , ta quy hoạch động theo các tham số:

$f(i, j, p)$  là diện tích lớn nhất của hình đa giác gồm  $p + 1$  đỉnh trong số các đỉnh từ  $i$  đến  $j$ , trong đó chắc chắn có chọn điểm  $i$  và điểm  $j$  ( $1 \leq i < j \leq m$ ;  $3 \leq p < k$ ).

Nếu  $p$  chẵn,  $f(i, j, p) = \max_{i < t < j} \{f(i, t, p-1) + S_\Delta(i, t, j)\}$ . Giải thích là diện tích đa giác  $p$  đỉnh = diện tích một đa giác  $p-1$  đỉnh cộng thêm diện tích một tam giác nữa.

Nếu  $p$  lẻ,  $f(i, j, p) = \max_{i < t < j} \left\{ f(i, t, \frac{p+1}{2}) + f(t, j, \frac{p+1}{2}) + S_\Delta(i, t, j) \right\}$ . Giải thích là nếu có một đa giác  $p$  đỉnh thì sẽ tồn tại một đỉnh  $t \in (i, j)$  sao cho diện tích đa giác bạn đầu được tách làm ba phần:

- Diện tích một đa giác gồm  $\frac{p+1}{2}$  đỉnh chọn trong các điểm từ điểm  $i$  tới điểm  $t$  và có chọn điểm  $i$  cũng như điểm  $t$ .
- Diện tích một đa giác gồm  $\frac{p+1}{2}$  đỉnh chọn trong các điểm từ điểm  $t$  tới điểm  $j$  và có chọn điểm  $t$  cũng như điểm  $j$ .
- Diện tích tam giác  $(i, t, j)$ .
- Cuối cùng ta quan tâm tới  $f(1, m, k)$ . Chú ý là để tính một giá trị  $f(i, j, p)$  ta mất thời gian  $O(n)$  tuy nhiên không phải giá trị  $f(i, j, p)$  nào cũng phải tính; để tính  $f(*, *, p)$  ta quy về tính  $f(*, *, \left[ \frac{p}{2} \right])$ . Sử dụng đệ quy kết hợp với bảng lưu trữ kết quả, công thức truy hồi tính  $f(1, m, k)$  có thể giải trong thời gian  $O(n^2 \log n)$ .

4.68.

- Bài toán cơ bản: Có  $n$  công việc, mỗi công việc có thời gian thực hiện  $t_i$  và mức độ quan trọng là  $w_i$ . Gọi  $f_i$  là thời gian hoàn thành, tìm trình tự thực hiện các công việc để tổng  $w_i * f_i$  là nhỏ nhất. Bài toán cơ bản này được thực hiện theo thứ tự sắp xếp giảm dần của  $w_i/t_i$ .
- Giải quyết bài toán gốc bằng cách dùng thuật toán quy hoạch động để tìm cách chia  $n$  món ăn thành hai tập, mỗi tập được thực hiện theo bài toán cơ bản để tổng  $|f_i - D|$  nhỏ nhất. Gọi  $F[i, t]$  chi phí nhỏ nhất để xếp các món từ 1 đến  $i$  vào 2 nhóm mà nhóm 1 thực hiện hết  $t$  đơn vị thời gian.

Ta có  $F[i, t] = \min \{F[i-1, t-t_i] + w_i * t, F[i-1, t] + w_i * (t_1 + t_2 + \dots + t_{i-1} - t)\}$ .

**4.69.** Quy hoạch động để tính  $f(b_i)$ :  $i = 1, 2, \dots, m * n$ .

Thời gian:  $O(mn \log b)$  ( $b < 10^9$ ).

- Chuẩn bị hai bảng phụ để trả lời truy vấn với thời gian  $O(1)$ :  $O(mn)$ .
- Duyệt truy vấn để trả lời  $O(k)$ .

## Chuyên đề 5

**5.1.** Có thể tính bán bậc ra của  $v$  trong thời gian  $O(\deg^+(v))$  bằng cách duyệt danh sách kè của  $v$ .

Để tính bán bậc vào của  $v$ , ta phải xác định  $v$  nằm trong bao nhiêu danh sách kè của đỉnh khác, tức là nếu gọi danh sách các đỉnh nối từ  $u$  là  $\text{list}[u]$  thì ta cần xác định số lượng đỉnh  $u$  mà  $v \in \text{list}[u]$ .

Thuật toán duyệt tất cả các đỉnh  $u$  và với mỗi đỉnh  $u$  duyệt toàn bộ  $\text{list}[u]$  có thời gian thực hiện giải thuật là  $O(m)$ .

Nếu  $\text{list}[u]$  được tổ chức dưới dạng các cấu trúc dữ liệu hỗ trợ tìm kiếm nhanh như bảng băm, ta có thể có giải thuật  $O(n)$  do việc kiểm tra  $v \in \text{list}[u]$  có thể thực hiện trong thời gian  $O(1)$ .

**5.2.** Nếu đồ thị được biểu diễn bằng ma trận kè, việc xây dựng đồ thị chuyển vị mất thời gian  $O(n^2)$ , nếu đồ thị được biểu diễn bằng danh sách kè, việc xây dựng đồ thị chuyển vị mất thời gian  $O(m)$ . Với  $n, m$  lần lượt là số đỉnh và số cạnh của đồ thị

Thuật toán:

$t_i$  và  
thực  
được

tìm  
n cờ  
từ 1  
])].

yết  
ách  
cần  
] có  
nh  
thể  
en  
ây  
số

Trong trường hợp đồ thị biểu diễn bằng ma trận kè  $A = \{a_{ij}\}_{n \times m}$ , ta sẽ xây dựng ma trận kè  $A^T = \{a_{ij}\}_{m \times n}$  của đồ thị chuyển vị như phương pháp chuyển vị ma trận:

```
for i := 1 to n do
    for j := 1 to n do
        aT[j, i] := a[i, j];
```

Trong trường hợp đồ thị biểu diễn bằng danh sách kè với  $list[u]$  là danh sách các đỉnh nối từ  $u$ , ta sẽ xây dựng các danh sách kè của đồ thị chuyển vị với  $list^T[v]$ . là danh sách các đỉnh nối tới  $v$ .

```
for v := 1 to n do
    listT[v] := ∅;
    for u := 1 to n do
        for v ∈ list[u] do
            listT[v].Insert(u);
```

Ở đây phương thức  $list^T[v].Insert(u)$  là lệnh chèn  $u$  vào danh sách  $list^T[v]$ .

**5.3.** Giả sử đồ thị có  $n$  đỉnh và  $m$  cạnh, biểu diễn bằng danh sách kè với  $list[u]$  là danh sách các đỉnh kè với  $u$ . Ta sẽ xây dựng các danh sách kè  $list'[u]$  của đơn đồ thị tương ứng.

```
for u := 1 to n do
    list'[u] := ∅;
    for u := 1 to n do
        mark[u] := False;
    for u := 1 to n do
        begin
            for ∀v ∈ list[u] do
                if not mark[v] then //mark[v] = True: v đã được đưa vào list[u]
                begin
                    list'[u].Insert(v);
                    mark[v] := True; //Đánh dấu
                end;
                for ∀v ∈ list[u] do //Đặt lại mảng mark thành false trong thời gian O(deg[u])
                    mark[v] := False;
            end;
```

Dễ thấy rằng thời gian thực hiện giải thuật trên là  $O(n + m)$ .

#### 5.4. Chứng minh

Kí hiệu  $A^k = \{a_{ij}^{(k)}\}_{n \times m}$  là lũy thừa bậc  $k$  của ma trận  $A$ , ta sẽ chứng minh rằng  $a_{ij}^{(k)}$  là số đường đi từ  $i$  tới  $j$  qua đúng  $k$  cạnh.

Khi  $k = 1$ , ta có ngay điều phải chứng minh từ định nghĩa của ma trận  $A$ . Với  $k > 1$  giả sử điều này đúng với giá trị  $k - 1$ , tức là  $a_{ij}^{(k-1)}$  là số đường đi từ  $i$  tới  $j$  qua đúng  $k - 1$  cạnh.

Xét một đường đi từ  $i$  tới  $j$  qua đúng  $k$  cạnh, đường đi này trước khi tới  $j$  sẽ phải tới một đỉnh  $q$  nào đó kè với  $j$ . Với đỉnh  $q$  kè  $j$  thì số đường đi  $i \rightarrow q \rightarrow j$  qua đúng  $k$  cạnh sẽ bằng số đường đi từ  $i$  tới  $q$  qua đúng  $k - 1$  cạnh nhân với số cạnh nối  $(q, j)$ . Từ đó ta có công thức truy hồi:

$$a_{ij}^{(k)} = \sum_{q \in V} a_{iq}^{(k-1)} * a_{qj}$$

Theo công thức nhân ma trận, ta suy ra  $a_{ij}^{(k)}$  là phần tử trên hàng  $i$ , cột  $j$  của ma trận tích  $A^{(k-1)} \times A$ , tức là ma trận  $A^k$  (đpcm).

#### 5.5. Giả sử đồ thị $n$ đỉnh và $m$ cung.

Nếu  $G$  được biểu diễn bằng ma trận kè  $A$ , việc tính  $A^2$  có thể thực hiện trong thời gian  $O(n^3)$  bằng phương pháp nhân ma trận. (Có thể dùng thuật toán Strassen để có độ phức tạp  $O(n^{\lg 7})$ ).

Nếu  $G$  được biểu diễn bằng danh sách kè trong đó  $list[u]$  là danh sách các đỉnh kè với  $u$ . Thuật toán có thể tiến hành như sau:

```
for ∀u ∈ V do
    l2[u] := ∅;
for ∀u ∈ V do
    for ∀v ∈ list[u] do
        for ∀w ∈ list[v] do
            l2[u].Include(w);
Output ← l2;
```

(Lệnh  $l2[u].Include(w)$  để bổ sung  $w$  vào danh sách  $l2[u]$ );

Với mỗi giá trị  $u$  của vòng lặp *for* ngoài cùng, lệnh *Include* được thực hiện không quá  $m$  lần. Suy ra thuật toán có độ phức tạp  $O(mn)$ . Kết thúc thuật toán, nếu cần có thao tác loại bỏ các đỉnh lặp trong các danh sách  $l2[.]$ , ta có thể xử

lý bằng kĩ thuật đánh dấu (mất thời gian  $O(n + |l2[u]|)$  trên mỗi danh sách  $l2[u]$ ). Thao tác chuẩn hoá này có thời gian là  $O(\sum_{u \in V} (n + |l2[u]|)) = O(n^2 + m)$ .

**5.6.** Xây dựng cấu trúc dữ liệu để biểu diễn đơn đồ thị vô hướng và các thao tác:

- Liệt kê các đỉnh kề với một đỉnh  $u$  cho trước trong thời gian  $O(\deg(u))$ .
- Kiểm tra hai đỉnh có kề nhau hay không trong thời gian  $O(1)$ .
- Loại bỏ một cạnh trong thời gian  $O(1)$ .
- Bổ sung một cạnh  $(u, v)$  nếu nó chưa có trong thời gian  $O(1)$ .

*Cách xây dựng*

Sử dụng các danh sách kề biểu diễn theo phương pháp mốc nối kép:

List[ $u$ ]: Danh sách các đỉnh kề  $u$ .

Bảng con trỏ  $A = \{a_{ij}\}$  trong đó  $a_{ij}$  là con trỏ tới nút chứa đỉnh  $j$  trong list[ $i$ ], trường hợp  $(i, j)$  không là cạnh thì  $a_{ij} = \text{nil}$ .

Để liệt kê các đỉnh kề với  $u$ , đơn thuần duyệt list[ $u$ ] mất thời gian  $O(\deg(u))$ .

Để kiểm tra hai đỉnh  $u, v$  kề nhau hay không, ta chỉ cần kiểm tra  $a_{uv} \neq \text{nil}$  trong thời gian  $O(1)$ .

Để loại bỏ một cạnh  $(u, v)$ , ta xoá nút  $a_{uv}$  khỏi list[ $u$ ] và xoá nút  $a_{vu}$  khỏi list[ $v$ ], sau đó đặt lại  $a_{uv}$  cũng như  $a_{vu}$  thành nil. Thao tác xoá trong danh sách mốc nối kép mất thời gian  $O(1)$ .

Để bổ sung một cạnh  $(u, v)$ , trước hết ta kiểm tra  $a_{uv}$  phải bằng nil trước. Sau đó chèn  $v$  vào cuối list[ $u$ ] cũng như chèn  $u$  vào cuối list[ $v$ ], sau đó cập nhật lại  $a_{uv}$  cũng như  $a_{vu}$ . Thao tác chèn trên danh sách mốc nối kép mất thời gian  $O(1)$ .

**5.7.** Có thể tìm “bồn chúa” bằng thuật toán sau (giả sử đồ thị có  $n$  đỉnh biểu diễn bằng ma trận kề  $A$ ).

```
i := 1;
repeat
    j := i + 1;
    while (j <= n) and a[j, i] do j := j + 1;
    if j > n then
        begin
            if <i là bồn chúa> then Exit(i)>
```

```

Break;
end
else
i := j;
until i > n;

```

Thuật toán được thực hiện trong thời gian  $O(n)$ .

### 5.8. Phân tích:

$$BB^T(i, j) = \sum_{e \in E} B(i, e)B^T(e, j) = \sum_{e \in E} B(i, e)B(j, e)$$

Nếu  $i = j$ , ta có  $B(i, e).B(j, e) = (B(i, e))^2 = \begin{cases} 1, & \text{nếu } e \text{ đi vào hoặc đi ra đỉnh } i \\ 0, & \text{ngược lại.} \end{cases}$

Nếu  $i \neq j$ , ta có  $B(i, e).B(j, e) = \begin{cases} 1, & \text{nếu } e = (i, j) \text{ hoặc } e = (j, i) \\ 0, & \text{nếu } e \text{ không liên thuộc với } i \text{ hoặc } j. \end{cases}$

Vậy  $BB^T(i, j) = \begin{cases} \text{Số cung liên thuộc với } i, \text{ nếu } i = j \\ \text{Số cung kết nối } i \text{ với } j, \text{ nếu } i \neq j. \end{cases}$

### 5.9. Mô hình cài đặt

Biểu diễn đồ thị dưới dạng danh sách kề, mỗi đỉnh  $u$  ứng với một danh sách  $list[u]$  là danh sách các đỉnh nối từ  $u$ .

Xây dựng hai hàm:

- $First(u)$ : Trả về đỉnh đứng đầu danh sách  $list[u]$ , trả về 0 nếu  $list[u]$  rỗng;
- $Removefirst(u)$ : Huỷ bỏ đỉnh đứng đầu danh sách  $list[u]$ .

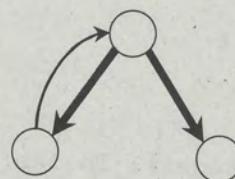
Khi đó thuật toán DFS không đệ quy bắt đầu từ đỉnh  $s$  có thể viết như sau:

```

u := s;
trace[s] := 0;
repeat
    Output ← u;
    while (u ≠ 0) and (first(u) = 0) do
        u := trace[u];
        if u ≠ 0 then
            begin
                v := first(u);
                removefirst(u);
                u := v;
            end;
    until u = 0;

```

### 5.10. Phản ví dụ:



5.11. Thực hiện thuật toán DFS. Khi từ đỉnh  $u$  xét một đỉnh  $v$  kề  $u$ , nếu  $v$  chưa thăm thì đi thăm  $v$ , nếu  $v$  đã thăm,  $v \neq u$  và  $v$  không phải nút cha của  $u$  thì truy vết đường đi từ  $u$  về  $v$  nối thêm cạnh  $(v, u)$  nữa để được chu trình đơn.

5.12. *Đáp án:* Các đỉnh sẽ được liệt kê theo thứ tự tăng dần của nhãn  $a[.]$ .

5.13. Áp dụng thuật toán BFS với nơi xuất phát là các ô biên để tìm đường đi tới vị trí nhà thám hiểm.

Truy vết ngược lại để tìm đường đi.

```
{$MODE OBJFPC}
{$INLINE ON}
program Escaping;
const InputFile = 'LABYR.INP';
OutputFile = 'LABYR.OUT';
maxMN = 1000;
dx: array[1..4] of Integer = (1, -1, 0, 0);
dy: array[1..4] of Integer = (0, 0, 1, -1);
var
a: array[0..maxMN + 1, 0..maxMN + 1] of Char;
trace: array[1..maxMN, 1..maxMN] of Integer;
qx, qy: array[1..maxMN * maxMN] of Integer;
front, rear: Integer;
m, n: Integer;
fi, fo: TextFile;
sx, sy: Integer;

procedure Enter;
var i, j: Integer;
begin
ReadLn(fi, m, n);
FillChar(a, SizeOf(a), 'X');
for i := 1 to m do
begin
for j := 1 to n do
begin
Read(fi, a[i, j]);
if a[i, j] = 'E' then
```

```

begin sx := i; sy := j; end;
end;
ReadLn(fi);
end;
end;

procedure Push(x, y: Integer); inline;
begin Inc(rear); qx[rear] := x; qy[rear] := y; end;

procedure Pop(var x, y: Integer); inline;
begin x := qx[front]; y := qy[front]; Inc(front); end;

procedure Init;
var i, j: Integer;
begin
FillChar(trace, SizeOf(trace), $FF); // -1;
front := 1; rear := 0;
for i := 1 to m do
begin
if a[i, 1] = 'O' then
begin trace[i, 1] := 0; Push(i, 1); end;
if (n > 1) and (a[i, n] = 'O') then
begin trace[i, n] := 0; Push(i, n); end;
end;
for j := 2 to n - 1 do
begin
if a[1, j] = 'O' then
begin trace[1, j] := 0; Push(1, j); end;
if (m > 0) and (a[m, j] = 'O') then
begin trace[m, j] := 0; Push(m, j); end;
end;
end;

procedure Solve;
var ux, uy, vx, vy, d: Integer;
begin
if trace[sx, sy] <> -1 then Exit;
repeat
Pop(ux, uy);
for d := 1 to 4 do
begin
vx := ux + dx[d];
vy := uy + dy[d];
if (a[vx, vy] <> 'X') and (trace[vx, vy] = -1) then
begin
trace[vx, vy] := d;
if (vx = sx) and (vy = sy) then
Exit;
Push(vx, vy);
end;
end;
end;

```

```

end;
end;
until front > rear;
end;

procedure PrintResult;
var
  ux, uy, d: Integer;
  len: Integer;
begin
  ux := sx; uy := sy;
  len := 1;
  repeat
    d := trace[ux, uy];
    if d = 0 then Break;
    Inc(len);
    Dec(ux, dx[d]);
    Dec(uy, dy[d]);
  until False;
  WriteLn(fo, len);
  ux := sx; uy := sy;
  repeat
    WriteLn(fo, ux, ' ', uy);
    d := trace[ux, uy];
    if d = 0 then Break;
    Inc(len);
    Dec(ux, dx[d]);
    Dec(uy, dy[d]);
  until False;
end;

begin
  AssignFile(fi, InputFile); Reset(fi);
  AssignFile(fo, OutputFile); Rewrite(fo);
  try
    Enter;
    Init;
    Solve;
    PrintResult;
  finally
    CloseFile(fi); CloseFile(fo);
  end;
end.

```

**5.14.** Duyệt DFS trên  $G$ , gọi  $f[u]$  là thời điểm duyệt xong đỉnh  $u$ .

“=>” Nếu  $G$  có cung ngược thì  $G$  có chu trình:

Vì mỗi cung ngược  $(u, v)$  nối từ  $u$  lên một đỉnh  $v$  là tiền bối của  $u$  trên cây DFS, đường đi từ  $v$  xuống  $u$  theo cây DFS rồi nối thêm cung  $(u, v)$  sẽ tạo ra chu trình.

“ $\leq$ ” Nếu  $G$  không có cung ngược thì  $G$  không có chu trình.

Ta thấy rằng nếu  $(u, v)$  là cung DFS, cung xuôi hay cung chéo thì  $f[u] > f[v]$ , tức là đỉnh  $v$  luôn được duyệt xong trước đỉnh  $u$ . Nếu  $G$  có chu trình  $(x_1, x_2, \dots, x_k = x_1)$  thì ta có  $f[x_1] > f[x_2] > \dots > f[x_k] = f[x_1]$ .

Sự mâu thuẫn trên cho ta đpcm.

**5.15.** Sắp xếp topo các đỉnh, thực hiện thuật toán quy hoạch động tính  $f(v)$  là số đường đi từ  $s$  tới  $v$ . Rõ ràng  $f(s) = 1$ , với  $\forall v \in V: f(v) = 0$  nếu  $v < s$  và  $f(v) = \sum_{u:(u,v) \in E} f(u)$  nếu  $v > s$ .

**5.16.** Coi mỗi đường tròn là một đỉnh đồ thị, hai đỉnh ứng với hai đường tròn  $(x_1, y_1, r_1)$  và  $(x_2, y_2, r_2)$  kề nhau nếu hai hình tròn có điểm chung, tức là:

$$|r_1 - r_2| \leq \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq r_1 + r_2$$

Vấn đề còn lại là liệt kê các miền liên thông của đồ thị vừa xây dựng

**5.17.** Liệt kê các miền liên thông gồm các ô số 1, với mỗi miền cho ứng với danh sách các ô thuộc miền liên thông đó.

Với mỗi miền, xét 4 mẫu khi quay miền các góc 0, 90, 180 và 270 độ, với mỗi mẫu tịnh tiến sao cho mẫu đó nằm trong bảng và tiếp xúc với cạnh trên và cạnh trái của bảng. Các ô của mẫu lấy theo thứ tự từ trên xuống và nếu hai ô cùng dòng thì lấy ô bên trái trước. Trong 4 mẫu, chọn mẫu có thứ tự từ điển nhỏ nhất (theo toạ độ các ô trong danh sách) làm đại diện cho miền. Hai miền chỉ sai khác nhau một phép dời hình khi đó sẽ có mẫu đại diện giống hệt nhau. Có thể cải tiến thêm nữa bằng cách tính hash code cho mẫu đại diện.

Xếp các mẫu đại diện theo thứ tự tăng dần để các mẫu đại diện giống nhau dồn lại thành một đoạn liên tiếp trong danh sách sắp xếp, từ các mẫu đại diện đánh số hiệu cho miền.

**5.18.** Xem lại định lí chứng minh tính đúng đắn thuật toán Kosaraju-Sharir. Thuật toán dưới đây chỉ khác thuật toán Kosaraju-Sharir ở chỗ không cần đảo chiều đồ thị.

*Thuật toán:*

*Bước 1:* Duyệt DFS, đánh số lại các đỉnh theo thứ tự duyệt xong.

*Bước 2:* Bắt đầu từ đỉnh  $u$  có chỉ số lớn nhất, nếu  $u$  chưa bị loại khỏi đồ thị thì kết nạp  $u$  và  $S$ , thực hiện DFS từ  $u$  loại  $u$  và tất cả những đỉnh đến được từ  $u$  khỏi đồ thị. Lặp lại cho tới khi đồ thị không còn đỉnh nào.

**5.19.** a) Trước hết dễ thấy rằng  $G$  nửa liên thông nếu và chỉ nếu  $G^{SCC}$  nửa liên thông.  $G^{SCC}$  là một đồ thị có hướng không có chu trình, mỗi đỉnh của  $G^{SCC}$  ứng với một thành phần liên thông trên  $G$ , mỗi cạnh của  $G$  nối hai đỉnh thuộc hai thành phần liên thông khác nhau sẽ ứng với một cạnh trên  $G^{SCC}$  nối hai thành phần liên thông đó. Ta chỉ cần chứng minh mệnh đề trên đồ thị  $G^{SCC}$ .

Nếu  $G^{SCC}$  nửa liên thông, sắp xếp topo các đỉnh của  $G^{SCC}$ , đỉnh 1 không có cung đi vào nên 1 phải tới được mọi đỉnh khác, trong đó có đỉnh 2. Nếu loại đỉnh 1 đi thì đỉnh 2 không có cung đi vào nên 2 phải tới được mọi đỉnh khác, trong đó có đỉnh 3. Lại loại tiếp đỉnh 2 và làm tương tự, ta có:

Đỉnh 1 tới được 2, đỉnh 2 tới được 3, ..., tức là trên  $G^{SCC}$  tồn tại đường đi qua tất cả các đỉnh.

Điều ngược lại hiển nhiên, nếu trên  $G^{SCC}$  tồn tại đường đi  $P$  qua tất cả các đỉnh thì với hai đỉnh  $u, v$  bất kì, nếu  $u$  xuất hiện trước  $v$  trên  $P$ , ta có  $u \rightarrow v$ . Ngược lại nếu  $v$  xuất hiện trước  $u$  trên  $P$  ta có  $v \rightarrow u$ , tức là  $G^{SCC}$  nửa liên thông.

b) Thuật toán kiểm tra tính nửa liên thông

Xây dựng đồ thị  $G^{SCC}$  và sắp xếp topo các đỉnh, đánh số theo thứ tự từ 1 tới  $k$ .

Kiểm chứng sự tồn tại của các cạnh  $(1, 2); (2, 3); \dots; (k-1, k)$ .

**5.20.** Các thuật toán trên có thể dễ dàng cài đặt trong trường hợp đồ thị biểu diễn bằng danh sách kè hay danh sách liên thuộc, chỉ chú ý là nếu  $(u, v)$  là cung DFS thì thủ tục  $\text{Visit}(v)$  không được xét  $u$  nữa.

**5.21.** Gọi số cây khung khác nhau của đồ thị  $G$  là  $\tau(G)$ .

Khi đó với một cạnh  $e$  bất kì, ta có công thức truy hồi:

$$\tau(G) = \tau(G - e) + \tau\left(\frac{G}{e}\right).$$

Ở đây  $G - e$  là đồ thị  $G$  bỏ đi cạnh  $e$  (số cạnh giảm 1),  $\frac{G}{e}$  là đồ thị  $G$  sau khi ta chập hai đầu cạnh  $e$  thành một đỉnh duy nhất (số đỉnh giảm 1).

Skiena (1990) còn chứng minh được định lí (matrix-tree theorem) để tính trực tiếp số cây khung của đồ thị dựa vào ma trận liền kề.

**5.22.** Vạch hành trình ban đầu đi quanh một hình chữ nhật. Sau đó bắt đầu đi dọc hành trình, khi đi qua một giao điểm với một hình chữ nhật *chưa nằm trên hành trình*, thay thế điểm đó bằng đường đi vòng quanh hình chữ nhật mới và tiếp tục đi trên hành trình mới.

**5.23.** Xếp  $2n$  người quanh vòng tròn theo thứ tự tùy ý và đánh số từ 1 tới  $2n$  theo vòng tròn.

Nếu tồn tại hai người  $i$  và  $i + 1$  là kẻ thù, trong  $2n - 2$  người còn lại lớn hơn hoặc bằng  $n + 1$  người không phải kẻ thù của  $i$  và cũng có lớn hơn hoặc bằng  $n + 1$  người không phải kẻ thù của  $i + 1$ . Suy ra chắc chắn sẽ có một cặp người  $j$  và  $j + 1$  ngồi cạnh nhau mà người  $i$  không phải kẻ thù của người  $j$  và người  $j + 1$  không phải kẻ thù của người  $i + 1$ . Lật ngược thứ tự những người từ người  $i + 1$  tới người  $j$  và lặp lại cho tới khi không còn hai người ngồi cạnh nhau là kẻ thù.

**5.24. Thuật toán:** Để điền hình tròn với mã Graycode  $n$  bit, ta chia nửa hình tròn bằng một đường kính qua tâm. Một nửa hình điền những số có bit đầu là 0, nửa kia điền những số có bit đầu là 1. Hai số trên hai vị trí đối xứng trực qua đường kính chỉ có duy nhất bit đầu khác nhau. Việc điền trên hai nửa được thực hiện tương tự bằng đệ quy (như trên hình tròn ứng với mã Graycode  $n - 1$  bit).

```

{$MODE OBJFPC}
program Test;
var i, n: Integer;

function Value(x: Integer; n: Integer): Integer;
var range: Integer;
begin
  if n = 1 then Exit(x);
  range := 1 shl (n - 1);
  if x < range then
    Result := Value(x, n - 1)
  else Result := Value(2 * range - 1 - x, n - 1) + range;
end;

begin

```

```

ReadLn(n);
for i := 0 to 1 shl n - 1 do
  Write(Value(i, n), ' ');
end.

```

**5.26.** Dùng mảng  $d[1..M, 1..N]$  of int64; ban đầu được khởi tạo bằng 0. Nếu ô  $(i,j)$  thuộc vào hình chữ nhật con thứ  $k$  thì bật bit thứ  $k$  của  $d[i,j]$  bằng cách:  $d[i,j] \text{ or } (1 \text{ shl } k)$ . Như vậy, hai ô cùng thuộc một vùng nếu hai ô liên thông với nhau và có  $d[i,j]$  bằng nhau. Dùng thuật toán loang để xác định số vùng.

Để giải được  $M, N$  lớn cần đánh chỉ số các toạ độ, rồi quản lí theo chỉ số. Như vậy chỉ cần khai báo mảng  $d[1..2*k, 1..2*k]$ .

**5.27.** Chú ý các vật cản có toạ độ  $|x_i|, |y_i| < 1001$ . Do đó, sử dụng thuật toán loang từ ô  $(0,0)$  trên bảng  $d[-1000..1000, -1000..1000]$ . Giá trị  $d[x,y]$  là số bước ít nhất để đi từ ô  $(0,0)$  đến ô  $(x,y)$ . Từ các giá trị của bảng  $d$  xác định số ô cần tìm.

### 5.29. Mã và tốt

- Đặc tính của quân mã: Nếu quân mã đứng ở ô  $(x, y)$  mà di chuyển đến ô  $(u, v)$  thì mọi đường đi từ  $(x, y)$  đến  $(u, v)$  đều mất chẵn bước hoặc đều mất lẻ bước. Dùng thuật toán loang để xây dựng mảng  $Ma[x, y]$  là số bước ít nhất để quân mã di chuyển từ vị trí ban đầu đến ô  $(x, y)$ .
- Đặc tính của quân tốt: Nếu quân tốt đứng ở ô  $(x, y)$  mà di chuyển đến ô  $(u, v)$  thì có thể mất chẵn bước hoặc có thể mất lẻ bước. Dùng thuật toán loang để xây dựng mảng  $Tot[x, y, 0]$  là số bước ít nhất để quân tốt di chuyển từ vị trí ban đầu đến ô  $(x, y)$  với chẵn bước,  $Tot[x, y, 1]$  là số bước ít nhất để quân tốt di chuyển từ vị trí ban đầu đến ô  $(x, y)$  với lẻ bước.
- Duyệt tất cả các ô có khả năng gặp nhau của mã và tốt.
- Bài toán có thể mở rộng với nhiều quân mã và nhiều quân tốt.

```

{$M 10000000}
uses math;
const finp ='matot.inp';
fout ='matot.out';
maxn =100+5;
m1:array[1..8] of longint=(-2,-1,1,2,2,1,-1,-2);
m2:array[1..8] of longint=(1,2,2,1,-1,-2,-2,-1);
t1:array[1..8] of longint=(-1,-1,-1,0,1,1,1,0);
t2 :array[1..8] of longint=(-1,0,1,1,1,0,-1,-1);
var fi,fo:text;
res,n:longint;
mi,mj,ti,tj:longint;

```

```

a, ma:array[1..maxn,1..maxn] of longint;
tot:array[1..maxn,1..maxn,0..1] of longint;

procedure docFile;
var i,j:longint;
ch:char;
begin
readln(fi,n);
for i:=1 to n do
begin
for j:=1 to n do
begin
read(fi,ch);
case ch of
'M':begin
mi:=i;
mj:=j;
end;
'T':begin
ti:=i;
tj:=j;
end;
'#':a[i,j]:=-1;
end;
end;
readln(fi);
end;
end;

procedure LoangMa;
var i,j,dau,cuoi,u,v,t :longint;
dd :array[1..maxn,1..maxn] of boolean;
qx,qy:array[1..maxn*maxn] of longint;
begin
fillchar(dd,sizeof(dd),0);
qx[1]:=mi;
qy[1]:=mj;
dau:=1;
cuoi:=1;
dd[mi,mj]:=true;
while dau<=cuoi do
begin
i:=qx[dau];
j:=qy[dau];
inc(dau);
for t:=1 to 8 do
begin
u:=i+m1[t];
v:=j+m2[t];

```

```

if (u>0) and (u<=n) and (v>0) and (v<=n) then
if not dd[u,v] and (a[u,v]<>-1) then
begin
inc(cuoi);
qx[cuoi]:=u;
qy[cuoi]:=v;
dd[u,v]:=true;
ma[u,v]:=ma[i,j]+1;
end;
end;
end;
end;

procedure LoangTot;
var i,j,dau,cuoi,u,v,t,tt:longint;
dd:array[1..maxn,1..maxn,0..1] of boolean;
qx,qy,qt:array[1..2*maxn*maxn] of longint;
begin
fillchar(dd,sizeof(dd),false);
dau:=1;
cuoi:=1;
qx[1]:=ti;
qy[1]:=tj;
qt[1]:=0;
dd[ti,tj,0]:=true;
while dau<=cuoi do
begin
i:=qx[dau];
j:=qy[dau];
tt:=qt[dau];
inc(dau);
for t:=1 to 8 do
begin
u:=i+t1[t];
v:=j+t2[t];
if (u>0) and (u<=n) and (v>0) and (v<=n) then
if not dd[u,v,1-tt] and (a[u,v]<>-1) then
begin
dd[u,v,1-tt]:=true;
tot[u,v,1-tt]:=tot[i,j,tt]+1;
inc(cuoi);
qx[cuoi]:=u;
qy[cuoi]:=v;
qt[cuoi]:=1-tt;
end;
end;
end;
end;

```

```

procedure chuanBi;
begin
  LoangMa;
  LoangTot;
end;

procedure tim;
var i,j,temp :longint;
begin
  res:=100000;
  for i:=1 to n do
  for j:=1 to n do
  begin
    if a[i,j]=-1 then
    continue;
    temp:=max(ma[i,j],tot[i,j],(ma[i,j] mod 2));
    if temp<res then
    res:=temp;
  end;
end;

BEGIN
  assign(fi,finp);
  reset(fi);
  assign(fo,fout);
  rewrite(fo);
  docFile;
  chuanBi;
  tim;
  writeln(fo,res);
  close(fi);
  close(fo);
END.

```

**5.30.** Xét bảng từ dòng 2 đến dòng  $M-1$ , tìm đường đi xuất phát từ cột 1 qua ít ô chưa được tô màu nhất đến cột  $N$ .

### 5.31. Độ nước

*Thuật toán 1:* Loang trên đồ thị có đỉnh là  $(v_1, v_2)$  với  $v_1 \leq v_2$  có ý nghĩa là: có một bình chứa  $v_1$  lít, một bình chứa  $v_2$  lít, bình còn lại chứa  $v - v_1 - v_2$ .

Độ phức tạp:  $O(v^2 n)$ .

*Thuật toán 2:* Loang trên đồ thị có đỉnh là  $(l_1, v_2)$  với  $l_1 \leq n, v_2 \leq v$  có ý nghĩa là: có một bình chứa nước đến vạch  $l_1$ , một bình chứa  $v_2$  lít.

Độ phức tạp:  $O(vn^2)$ .

```
{$R+, Q+}
const maxV =30000;
maxN =20;
fi ='POUR.INP';
fo ='POUR.OUT';
type item =record
  l, v :longint;
end;
var d :array[0..maxN, 0..maxV]of longint;
q :array[1..(maxN+1)* (maxV+1)]of item;
vach :array[0..maxN]of longint;
dau, cuoi :longint;
v,n,m :longint;

procedure docf;
var f :text; i :longint;
begin
  assign(f, fi); reset(f);
  readln(f, v, n, m);
  for i:=1 to n do read(f, vach[i]);
  vach[0]:=0;
  close(f);
end;

procedure ghikq(res :longint);
var f :text;
begin
  assign(f, fo); rewrite(f);
  writeln(f, res);
  close(f);
  halt;
end;

procedure nap(l1, v1, nstep:longint);
begin
  if d[l1, v1]=0 then begin
    inc(cuoi);
    q[cuoi].l:=l1;
    q[cuoi].v:=v1;
    d[l1, v1]:=nstep+1;
    if (v1=m)or(vach[l1]=m)or(v-v1-vach[l1]=m) then ghikq(nstep);
  end;
end;

procedure loang;
var w :array[1..3]of longint;
k, ns, i, j :longint;
```

```

begin
fillchar(d,sizeof(d),0);
dau:=1; cuoi:=0;
nap(0,v,0);
while dau<=cuoi do begin
w[1]:=vach[q[dau].l];
w[2]:=q[dau].v;
w[3]:=v-w[1]-w[2];
ns:=d[q[dau].l,q[dau].v];
inc(dau);
for i:=1 to 3 do
for j:=1 to 3 do
if i<>j then
for k:=0 to n do begin
if w[i]<=vach[k] then break;
nap(k,w[j]+w[i]-vach[k],ns);
nap(k,v-w[j]-w[i],ns);
end;
for i:=1 to 3 do
for j:=1 to 3 do
if i<>j then
for k:=1 to n do
if (w[j]<vach[k])and(w[i]+w[j]>=vach[k]) then begin
nap(k,w[j]+w[i]-vach[k],ns);
nap(k,v-w[j]-w[i],ns);
end;
end;
end;

BEGIN
docf;
loang;
ghikq(-1);
END.

```

**5.32.** Dùng thuật toán Dijkstra để tìm đường đi ngắn nhất từ nhà đến tất cả các nút giao thông bằng ô tô, tìm đường đi ngắn nhất từ các nút giao thông đến trường bằng đi bộ và tìm đường đi ngắn nhất từ các nút giao thông đến cơ quan của mẹ Tuấn bằng ô tô. Thủ các nút để chọn nút giao thông để Tuấn đến trường không bị muộn giờ còn mẹ đến cơ quan làm việc sớm nhất. Chương trình dưới đây thể hiện thuật toán đạt 50% số bộ kiểm thử. Để chạy được với kích thước tối đa cần cài tiến chương trình dùng danh sách cạnh và thuật toán Dijkstra Heap.

```

{ $R+, $Q+ }
const MAX =100;
limit =MAX * maxint;

```

```

        fi          ='SCHOOL.INP';
        fo          ='SCHOOL.OUT';

var   w:array[1..MAX,1..MAX,1..2]of integer;
        ab,d,L1,Ln,Lk:array[1..MAX]of longint;
        tr:array[1..MAX,1..3]of longint;
        n,m,shool :longint;
        best   :longint;

Procedure docfile;
var   i,j,k,c :longint;
        f       :text;
begin
        fillchar(ab,sizeof(ab),0);
        assign(f,fi); reset(f);
        readln(f,n,m,shool);
        for k:=1 to m do
            readln(f,i,j,w[i,j,1],w[i,j,2]);
        close(f);
end;

Procedure tuNha;
var   i,j,imin,lmin :longint;
begin
        fillchar(d,sizeof(d),0);
        for i:=1 to n do
            l1[i]:=limit;
            l1[1]:=0;
        repeat
            lmin:=limit;
            for i:=1 to n do
                if (l1[i]<lmin)and(d[i]=0) then
                begin
                    lmin:=l1[i];
                    imin:=i;
                end;
                if lmin=limit then break;
                i:=imin; d[i]:=1;
            for j:=1 to n do
                if (d[j]=0)and(w[i,j,2]>0)and(l1[j]>l1[i]+w[i,j,2]) then
                begin
                    l1[j]:=l1[i]+w[i,j,2];
                    tr[j,1]:=i;
                end;
            until false;
end;

Procedure denTruong;
var   i,j,imin,lmin :longint;

```

```

begin
  fillchar(d,sizeof(d),0);
  for i:=1 to n do
    lk[i]:=limit;
  lk[shool]:=0;
repeat
  lmin:=limit;
  for i:=1 to n do
    if (lk[i]<lmin)and(d[i]=0) then
      begin
        lmin:=lk[i];
        imin:=i;
      end;
    if lmin=limit then break;
    i:=imin; d[i]:=1;
  for j:=1 to n do
    if (d[j]=0)and(w[j,i,1]>0)and(lk[j]>lk[i]+w[j,i,1]) then
      begin
        lk[j]:=lk[i]+w[j,i,1];
        tr[j,2]:=i;
      end;
  until false;
end;

Procedure denCoQuan;
var i,j,imin,lmin :longint;
begin
  fillchar(d,sizeof(d),0);
  for i:=1 to n do
    ln[i]:=limit;
  ln[n]:=0;
repeat
  lmin:=limit;
  for i:=1 to n do
    if (ln[i]<lmin)and(d[i]=0) then begin
      lmin:=ln[i];
      imin:=i;
    end;
  if lmin=limit then break;
  i:=imin; d[i]:=1;
  for j:=1 to n do
    if (d[j]=0)and(w[j,i,2]>0)and(ln[j]>ln[i]+w[j,i,2]) then
      begin
        ln[j]:=ln[i]+w[j,i,2];
        tr[j,3]:=i;
      end;
  until false;
end;

```

```

Procedure ghikq;
var f :text;
    i,ibest :longint;
begin
    assign(f,fo); rewrite(f);
    best:=maxlongint;
    for i:=1 to n do
        if (l1[i]+lk[i]<60) and (best>l1[i]+ln[i]) then
        begin
            best:=l1[i]+ln[i];
            ibest:=i;
        end;
    writeln(f,best);
    close(f);
end;

BEGIN
    docfile;
    tuNha;
    denCoQuan;
    denTruong;
    ghikq;
END.

```

### 5.33. Vé xe miễn phí

*Thuật toán 1:*

- Xây dựng mảng  $L[i]$  là độ dài đường đi ngắn nhất từ 1 đến  $i$  ( $i = 1, 2, \dots, N$ ), mảng  $R[i]$  là độ dài đường đi ngắn nhất từ  $N$  đến  $i$  ( $i = 1, 2, \dots, N$ ). Thời gian chuẩn bị mảng  $L$  và  $R$  bằng thuật toán Dijkstra Heap trên đồ thị gốc mất  $O(N \log N + M)$ .
- Thủ lần lượt từng cạnh để chọn cạnh  $(i, j)$  mà  $L[i] + R[j]$  là nhỏ nhất.

*Thuật toán 2:*

- Xây dựng đồ thị mới từ đồ thị gốc như sau: với mỗi đỉnh  $i$  của đồ thị gốc tạo hai đỉnh của đồ thị mới  $(i, 0)$  và  $(i, 1)$ . Từ đỉnh  $(i, t)$  sang đỉnh  $(j, t)$  có trọng số bằng trọng số cạnh  $(i, j)$ , từ đỉnh  $(i, 0)$  sang  $(j, 1)$  có trọng số bằng 0.
  - Xây dựng mảng  $L[i, t]$  là độ dài đường đi ngắn nhất từ 1 đến  $i$  ( $i = 1, 2, \dots, N$ ) đã sử dụng ( $t = 1$ ) hay chưa sử dụng ( $t = 0$ ) vé khuyến mại bằng thuật toán Dijkstra Heap trên đồ thị mới với độ phức tạp  $O(N \log N + M)$ .
- Chú ý, thuật toán 1 chỉ có thể giải với số vé khuyến mại là 1, nhưng với thuật toán 2 có thể giải cho bài toán mở rộng với số vé khuyến mại là  $k$ , khi đó đồ thị mới có  $(k+1)N$  đỉnh.

### 5.34. Hexgame

*Thuật toán 1:* Với một trạng thái của trò chơi, lấy các số từ trên xuống dưới, từ trái qua phải rồi thay số 0 đầu tiên thành số 9, ta được một hoán vị của 0, 1, ..., 9. Như vậy mỗi trạng thái của trò chơi tương ứng với một hoán vị của 0, 1, ..., 9 (ví dụ trạng thái ban đầu là hoán vị 1, 2, 3, 8, 9, 0, 4, 7, 6, 5). Xây dựng đồ thị, trong đó mỗi đỉnh tương ứng là một hoán vị, cung giữa hai đỉnh của đồ thị được xác định bằng cách kiểm tra có tồn tại phép biến đổi giữa hai hoán vị hay không. Sử dụng thuật toán BFS trên đồ thị vừa xây dựng để tìm cách biến đổi ít nhất.

*Thuật toán 2:* Nhận thấy, với một trạng thái chỉ có hai cách biến đổi mà giới hạn 50% bộ kiểm thử của bài toán là số phép biến đổi không vượt quá 15, duyệt nhị phân giới hạn độ sâu.

*Thuật toán 3:* Khảo sát bài toán và nhận thấy số phép biến đổi không vượt quá 30. Sử dụng một trong hai thuật toán trên nhưng tìm kiếm từ hai đầu.

### 5.35. Nhắn tin

Coi mỗi học viên là một đỉnh của đồ thị, cạnh  $(u, v)$  ứng với quan hệ: học viên  $u$  có thể nhắn tin cho học viên  $v$ .

Duyệt DFS, duyệt xong đỉnh nào đầy đỉnh đó vào một danh sách  $L$ .

Đánh dấu các đỉnh là “chưa nhận tin”. Xét các đỉnh trong danh sách  $L$  từ đỉnh cuối danh sách tới đỉnh đầu danh sách, mỗi khi xét tới một đỉnh  $u$  “chưa nhận tin”, dùng BFS hoặc DFS liệt kê tất cả các đỉnh “chưa nhận tin” đến được từ  $u$  thành “đã nhận tin”, đỉnh  $u$  ứng với một học viên mà thầy giáo phải nhắn tin trực tiếp.

Tính đúng đắn của thuật toán được suy ra từ thuật toán Kosaraju–Sharir: Đỉnh  $u$  “chưa nhận tin” đứng cuối danh sách  $L$  tại mỗi bước là đỉnh thuộc một thành phần liên thông mạnh không có cung đi vào.

Thời gian thực hiện thuật toán:  $O(m + n)$ .

*Chương trình*

```
{$MODE OBJFPC}
program MessageSending;
const InputFile = 'MESSAGE.INP';
OutputFile = 'MESSAGE.OUT';
maxN = 100000;
maxM = 100000;
var n, m, k: Integer;
```

```

adj: array[1..maxM] of Integer;
link: array[1..maxM] of Integer;
head: array[1..maxN] of Integer;
avail: array[1..maxN] of Boolean;
Stack: array[1..maxN] of Integer;
res: array[1..maxN] of Integer;

procedure Enter;
var f: TextFile;
u, i: Integer;
begin
AssignFile(f, InputFile); Reset(f);
try
ReadLn(f, n, m);
FillDWord(head[1], n, 0);
for i := 1 to m do
begin
Read(f, u, adj[i]);
link[i] := head[u];
head[u] := i;
end;
finally
CloseFile(f);
end;
end;

procedure Numbering;
var u: Integer;
Top: Integer;

procedure Visit(u: Integer);
var i: Integer;
begin
avail[u] := False;
i := head[u];
while i <> 0 do
begin
if avail[adj[i]] then Visit(adj[i]);
i := link[i];
end;
Inc(Top);
Stack[Top] := u;
end;
begin
FillChar(avail[1], n, True);
Top := 0;
for u := 1 to n do
if avail[u] then Visit(u);
end;

```

```

procedure Selecting;
var i: Integer;

procedure Visit(u: Integer);
var i: Integer;
begin
avail[u] := False;
i := head[u];
while i <> 0 do
begin
if avail[adj[i]] then Visit(adj[i]);
i := link[i];
end;
end;
begin
FillChar(avail[1], n, True);
k := 0;
for i := n downto 1 do
if avail[Stack[i]] then
begin
Inc(k);
res[k] := Stack[i];
Visit(Stack[i]);
end;
end;

procedure PrintResult;
var f: TextFile;
i: Integer;
begin
AssignFile(f, OutputFile);
Rewrite(f);
try
Writeln(f, k);
for i := 1 to k do Write(f, res[i], ' ');
finally
CloseFile(f);
end;
end;

begin
Enter;
Numbering;
Selecting;
PrintResult;
end.

```

### 5.36. Dây chuyền thông báo

Gọi  $\text{next}[i]$  là người truyền tin của người  $i$ . Mô hình hoá đồ thị với mỗi đỉnh là một học sinh và các cạnh  $(i, \text{next}[i])$ .

Duyệt DFS, đánh số các đỉnh ngược lại với thứ tự duyệt xong. Bắt đầu xét từ đỉnh 1 tới đỉnh  $n$  theo cách đánh số mới, mỗi khi xét tới một đỉnh  $u$ , đi liên tiếp theo liên kết  $\text{next}[\cdot]$ , đi qua đỉnh nào loại đỉnh đó khỏi đồ thị cho tới khi không đi tiếp được nữa các đỉnh đi qua được ghi nhận thành một dãy đỉnh.

Giả sử các dãy đỉnh thu được ở bước trước là:

$$L_1 = (x_1^1, x_2^1, \dots, x_{k_1}^1)$$

$$L_2 = (x_1^2, x_2^2, \dots, x_{k_2}^2)$$

...

$$L_q = (x_1^q, x_2^q, \dots, x_{k_q}^q)$$

Ta nối phần tử cuối của  $L_1$  với phần tử đầu của  $L_2$ , phần tử cuối của  $L_2$  với phần tử đầu của  $L_3, \dots$ . Cụ thể là đặt  $\text{next}[x_{k_i}^i] = x_1^{i+1}$ . Ta thu được một danh sách  $L$  gồm đúng  $n$  phần tử:  $(l_1, l_2, \dots, l_n)$ .

Cuối cùng nếu  $\text{next}[l_n] \neq l_1$ , ta đặt lại  $\text{next}[l_n] = l_1$  để tạo ra một vòng tròn.

Chương trình

```
{$MODE OBJFPC}
program CircularMessageSending;
const InputFile = 'CIRCLE.INP';
  OutputFile = 'CIRCLE.OUT';
  max = 1000000;
var next: array[1..max] of Integer;
  free: array[1..max] of Boolean;
  n: Integer;
  Count: Integer;
  Stack, List: array[1..max] of Integer;
  nStack, nList: Integer;
  
```

```
procedure Enter;
var f: TextFile;
  i: Integer;
begin
  AssignFile(f, InputFile); Reset(f);
  try
    ReadLn(f, n);
    for i := 1 to n do Read(f, next[i]);
    finally
      CloseFile(f); 
```

```

end;
end;

procedure DFS(from: Integer);
var i: Integer;
begin
nStack := 0;
repeat
Inc(nStack);
Stack[nStack] := from;
Free[from] := False;
from := next[from];
until not Free[from];
for i := nStack downto 1 do
begin Inc(nList); List[nList] := Stack[i]; end;
end;
function FindSegment(s: Integer): Integer;
begin
Result := s;
repeat
Free[Result] := False;
if Free[next[Result]] then Result := next[Result]
else Break;
until False;
end;

procedure Solve;
var i: Integer;
p, q, s, t: Integer;
begin
nList := 0;
FillChar(Free, n, True);
for i := 1 to n do
if Free[i] then DFS(i);
FillChar(Free, n, True);
Count := 0; p := 0; q := 0;
for i := n downto 1 do
begin
s := List[i];
if Free[s] then
begin
t := FindSegment(s);
if p = 0 then begin p := s; q := t; end
else begin next[q] := -s; Inc(Count); q := t; end;
end;
end;
if next[q] <> p then begin next[q] := -p; Inc(Count); end;
end;

```

```

procedure PrintResult;
var f: TextFile;
i: Integer;
begin
AssignFile(f, OutputFile); Rewrite(f);
try
WriteLn(f, Count);
for i := 1 to n do
if next[i] < 0 then
WriteLn(f, i, ' ', -next[i]);
finally
CloseFile(f);
end;
end;

begin
Enter;
Solve;
PrintResult;
end.

```

### 5.37. Hệ thống gần hoàn hảo

Đồ thị  $G = (V, E)$  được gọi là đồ thị nửa liên thông nếu nó thoả mãn điều kiện sau: với mọi cặp đỉnh  $u, v \in V$ . Nếu  $u$  không tới được  $v$  thì  $v$  phải tới được  $u$ .

Ta có một tính chất quan trọng: Điều kiện cần và đủ để đồ thị  $G$  là nửa liên thông là trên  $G$  tồn tại đường đi qua tất cả các đỉnh.

Liệt kê các thành phần liên thông mạnh, các đỉnh thuộc cùng một thành phần liên thông mạnh được hợp nhất lại, các cung nối hai đỉnh thuộc cùng một thành phần liên thông mạnh được bỏ đi, ta thu được một đồ thị có hướng không có chu trình (DAG) gọi là đồ thị  $G^{SCC}$ .

Sắp xếp topo các đỉnh thuộc  $G^{SCC}$ , sao cho các cung nối từ đỉnh chỉ số nhỏ tới đỉnh chỉ số lớn. Nếu  $G^{SCC}$  có  $k$  đỉnh, ta chỉ cần kiểm tra sự tồn tại của các cạnh  $(1, 2); (2, 3) \dots (k-1, k)$  là xác định được tính nửa liên thông.

*Chương trình*

```

{$MODE OBJFPC}
{$DEFINE RELEASE}
{$R-, S-, Q-, I-}
{$OPTIMIZATION LEVEL2}
{$INLINE ON}
{$M 20000000 0 0}
program SemiConnectedGraph;
const InputFile = 'SPERFECT.INP';

```

```

OutputFile = 'SPERFECT.OUT';
maxN = Round(1E5);
maxM = Round(1E5);
type Tedge = record
  x, y: Integer;
  link: Integer;
end;
Tstack = class
public
  items: array[1..maxN] of Integer;
  top: Integer;
  constructor Create;
  procedure Push(v: Integer); inline;
  function Pop: Integer; inline;
end;
var fi, fo: TextFile;
  itest, Ntest: Integer;
  n, m, nc: Integer;
  e: array[1..maxM] of Tedge;
  head: array[1..maxN] of Integer;
  number, low: array[1..maxN] of Integer;
  avail: array[1..maxN] of Boolean;
  lab: array[1..maxN] of Integer;
  InAB: array[1..maxN] of Boolean;
  IsSemiConnected: Boolean;
  mark: array[1..maxN] of Boolean;

procedure OpenFiles;
begin
  AssignFile(fi, InputFile); Reset(fi);
  AssignFile(fo, OutputFile); Rewrite(fo);
end;
procedure CloseFiles;
begin CloseFile(fi); CloseFile(fo);end;
procedure ReadData;
var i: Integer;
begin
  ReadLn(fi, n, m);
  FillChar(InAB, SizeOf(InAB), False);
  FillDWord(head[1], n, 0);
  for i := 1 to m do
    with e[i] do
      begin
        ReadLn(fi, x, y);
        InAB[x] := True; InAB[y] := True;
        link := head[x]; head[x] := i;
      end;
  end;
end;

```

```

constructor Tstack.Create;
begin top := 0;end;
procedure Tstack.Push(v: Integer);
begin
  Inc(top);
  Items[top] := v;
end;
function Tstack.Pop: Integer;
begin
  Result := Items[top];
  Dec(top);
end;
procedure Minimize(var target: Integer; value: Integer); inline;
begin
  if target > value then target := value;
end;
procedure GetConnectedComponents;
var Stack: Tstack;
  count: Integer;
  u: Integer;
  procedure Visit(u: Integer);
  var i, v: Integer;
  begin
    Inc(count);
    number[u] := Count;
    low[u] := maxN + 1;
    Stack.Push(u);
    i := head[u];
    while i <> 0 do
      with e[i] do
        begin
          v := y;
          if avail[v] then
            if number[v] = 0 then
              begin Visit(v); Minimize(low[u], low[v]); end
            else
              Minimize(low[u], number[v]);
          i := link;
        end;
        if low[u] >= number[u] then
          begin
            Inc(nc);
            repeat
              v := Stack.Pop;
              lab[v] := nc;
              avail[v] := False;
            until v = u;
          end;
        end;
  end;

```

```
begin
  FillChar(avail[1], n, True);
  FillDWord(number[1], n, 0);
  count := 0;
  nc := 0;
  Stack := Tstack.Create;
  for u := 1 to n do
    if inAB[u] and (number[u] = 0) then
      Visit(u);
    Stack.Free;
  end;

procedure Verify;
var i, lx, ly: Integer;
begin
  FillChar(mark, SizeOf(mark), False);
  for i := 1 to m do
    with e[i] do
      if lab[x] = lab[y] + 1 then
        mark[lab[x]] := True;
  IsSemiConnected := True;
  for i := 2 to nc do
    if not mark[i] then
      begin IsSemiConnected := False; Break; end;
  end;

procedure PrintResult;
begin
  if IsSemiConnected then WriteLn(fo, 'YES')
  else WriteLn(fo, 'NO');
end;

begin
  OpenFiles;
  try
    ReadLn(fi, Ntest);
    for itest := 1 to Ntest do
      begin
        ReadData;
        GetConnectedComponents;
        Verify;
        PrintResult;
      end;
    finally
      CloseFiles;
    end;
  end.
```

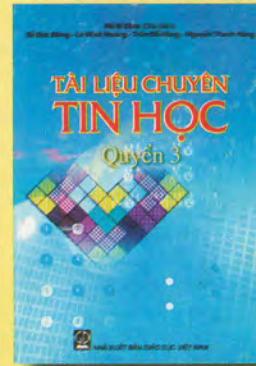
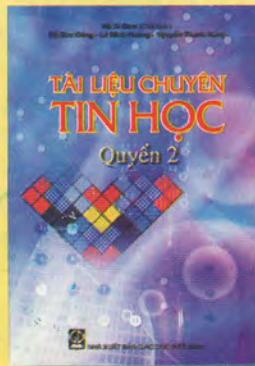
# MỤC LỤC

	Trang
LỜI NÓI ĐẦU	5
<b>PHẦN I. BÀI TẬP</b>	
CHUYÊN ĐỀ 1. THUẬT TOÁN VÀ PHÂN TÍCH THUẬT TOÁN	5
CHUYÊN ĐỀ 2. CÁC KIÊN THỨC CƠ BẢN	7
CHUYÊN ĐỀ 3. SẮP XẾP	18
CHUYÊN ĐỀ 4. THIẾT KẾ THUẬT TOÁN	31
CHUYÊN ĐỀ 5. CÁC THUẬT TOÁN TRÊN ĐỒ THỊ	76
<b>PHẦN II. HƯỚNG DẪN GIẢ BÀI TẬP</b>	
CHUYÊN ĐỀ 1. THUẬT TOÁN VÀ PHÂN TÍCH THUẬT TOÁN	94
CHUYÊN ĐỀ 2. CÁC KIÊN THỨC CƠ BẢN	95
CHUYÊN ĐỀ 3. SẮP XẾP	114
CHUYÊN ĐỀ 4. THIẾT KẾ THUẬT TOÁN	127
CHUYÊN ĐỀ 5. CÁC THUẬT TOÁN TRÊN ĐỒ THỊ	198



VƯƠNG MIỆN KIM CƯƠNG  
CHẤT LƯỢNG QUỐC TẾ

## GIỚI THIỆU BỘ SÁCH TÀI LIỆU CHUYÊN TIN HỌC DÀNH CHO HỌC SINH PHỔ THÔNG



Bạn đọc có thể mua sách tại các Công ty Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng sách của Nhà xuất bản Giáo dục Việt Nam :

- Tại TP. Hà Nội : 45 Phố Vọng ; 187, 187C Giảng Võ ; 232 Tây Sơn ; 25 Hàn Thuyên ;  
51 Lò Đúc ; 45 Hàng Chuối ; ngõ 385 Hoàng Quốc Việt ;  
17T2 - 17T3 Trung Hoà - Nhân Chính ; Toà nhà HESCO Văn Quán - Hà Đông.
- Tại TP. Đà Nẵng : 78 Pasteur ; 145 Lê Lợi ; 223 Lê Đình Lý.
- Tại TP. Hồ Chí Minh : 2A Đinh Tiên Hoàng, quận 1 ; 231 Nguyễn Văn Cừ, quận 5 ;  
116 Đinh Tiên Hoàng, phường 1, quận Bình Thạnh.
- Tại TP. Cần Thơ : 162D Đường 3/2, phường Xuân Khánh, quận Ninh Kiều.
- Tại Website bán hàng trực tuyến : [www.sach24.vn](http://www.sach24.vn)

Website : [www.nxbgd.vn](http://www.nxbgd.vn)

ISBN 978-604-0-08215-2



9 786040 082152



*Giá: 39.000đ*