HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

BÀI TẬP LẬP TRÌNH

Biên soạn : TS. NGUYỄN DUY PHƯƠNG

Hà Nội, tháng 12/2016

MỤC LỤC

I. Lập trình với các cấu trúc dữ liệu cơ bản

1.1. Các bài tập với dữ liệu kiểu số nguyên

BÀI 1.1.1: *TỔNG CHỮ SỐ*

Viết chương trình tính tổng chữ số của một số không quá 9 chữ số.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng số nguyên tương ứng

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra trên một dòng giá trị tổng chữ số tương ứng

Ví dụ:

Input	**Output**
2 1234 1000001	10 2

BÀI 1.1.2: *BẮT ĐẦU VÀ KẾT THÚC*

Viết chương trình kiểm tra một số nguyên dương bất kỳ (2 chữ số trở lên, không quá 9 chữ số) có chữ số bắt đầu và kết thúc bằng nhau hay không.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng số nguyên dương tương ứng cần kiểm tra

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra YES hoặc NO, tương ứng với bộ dữ liệu vào

Ví dụ:

Input	**Output**
2 12451 1000012	YES NO

BÀI 1.1.3: *BỘI SỐ CHUNG NHỎ NHẤT*

Viết chương trình tính bội số chung nhỏ nhất của hai số nguyên dương không quá 7 chữ số.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra trên một dòng giá trị bội số chung nhỏ nhất của hai số đó

Ví dụ

Input	**Output**
2 30 20 222222 8888888	60 98754568

BÀI 1.1.4: *SỐ CÓ TỔNG CHỮ SỐ CHIA HẾT CHO 10*

Viết chương trình kiểm tra một số có thỏa mãn tính chất tổng chữ số của nó chia hết cho 10 hay không.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Mỗi bộ test viết trên một dòng một số nguyên dương, ít nhất 2 chữ số nhưng không quá 9 chữ số.

Kết quả:

Mỗi bộ test viết ra YES hoặc NO tùy thuộc kết quả kiểm tra.

Ví dụ

Input	**Output**
3 3333 555555 123455	NO YES YES

BÀI 1.1.5: *SỐ ĐẸP 1*

Một số được coi là đẹp nếu nó là số thuận nghịch, tổng chữ số là số nguyên tố và tất cả các chữ số đều lẻ. Bài toán đặt ra là đếm xem trong một đoạn giữa hai số nguyên cho trước có bao nhiêu số đẹp như vậy.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không vượt quá 9 chữ số.

Kết quả:

Với mỗi bộ test viết ra số lượng các số thuần nguyên tố tương ứng.

Ví dụ

Input	**Output**
3 23 199 2345 6789 222222 99999999	4 0 311

BÀI 1.1.6: *SỐ ĐẸP 2*

Một số được coi là đẹp nếu nếu nó có tính chất thuận nghịch và tổng chữ số chia hết cho 10. Bài toán đặt ra là cho trước số chữ số. Hãy đếm xem có bao nhiêu số đẹp với số chữ số như vậy.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng số chữ số tương ứng cần kiểm tra (lớn hơn 1 và nhỏ hơn 10)

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra số lượng số đẹp tương ứng

Ví dụ

Input	**Output**
2 2 5	1 90

BÀI 1.1.7: *TRÒ CHƠI ĐOÁN SỐ*

Trong lúc rảnh rỗi, hai bạn sinh viên quyết định chơi trò đoán số giống học sinh cấp 1. Mỗi bạn nghĩ ra hai con số nguyên không âm sau đó viết ra tổng và hiệu của chúng (cũng là các số nguyên không âm). Công việc của bạn kia là xác định hai con số ban đầu. Ở một số lượt chơi, một bạn có thể cố tình đưa ra một cặp giá trị không thể là tổng và hiệu của hai số nguyên nào cả.

Viết chương trình giúp tính toán nhanh ra kết quả cho bài toán trên.

Dữ liệu vào:

Dòng đầu là số bộ test, không quá 200.

Mỗi dòng sau chứa hai số nguyên không âm s và d lần lượt là giá trị tổng và hiệu hai số. Cả hai số s và d đều không quá 10^4.

Kết quả: Ghi ra màn hình

Với mỗi bộ dữ liệu, đưa ra hai số ban đầu, số lớn viết trước, cách nhau một khoảng trống. Nếu không thể có cặp số như vậy thì in ra "impossible"

Ví dụ

Input	**Output**

2 40 20 20 40	30 10 impossible

BÀI 1.1.8: *MÁY BÁN HÀNG TỰ ĐỘNG*

Khi mua hàng bằng máy bán hàng tự động, người mua sẽ trả bằng một số tiền chẵn lớn hơn hoặc bằng giá của sản phẩm. Máy sẽ tính toán để trả lại số tiền thừa cho người mua. Giả sử trong máy chỉ có ba mệnh giá tiền là 1 dollar, 5 dollar và 10 dollar với quy ước mỗi lần trả chỉ được phép dùng ít hơn 5 tờ 1 dollar và ít hơn 2 tờ 5 dollar.

Hãy viết chương trình tính số tiền mỗi loại mà máy bán hàng tự động phải trả lại cho người mua.

Dữ liệu vào:

Dòng đầu tiên là số bộ test, mỗi bộ test ghi trên một dòng hai số nguyên không âm là giá của sản phẩm và tổng số tiền người mua đưa vào. Cả hai giá trị này đều không vượt quá 10^5.

Kết quả:

Với mỗi bộ test, viết ra biểu diễn số tiền cần trả của máy bán hàng tự động theo mẫu trong bộ test ví dụ dưới đây. (Chú ý: giữa các số và các dấu luôn có đúng một khoảng trống, cả với dấu =, dấu * hoặc dấu +)

Ví dụ cho Input và Output:

Input	**Output**
3 72 100 37 200 5 50	28 = 2 * 10 + 1 * 5 + 3 * 1 163 = 16 * 10 + 0 * 5 + 3 * 1 45 = 4 * 10 + 1 * 5 + 0 * 1

BÀI 1.1.9: *(*) BIỂU DIỄN SỐ BẰNG QUE DIÊM*

Một trong những cách biểu diễn số khá phổ biến trong các đồng hồ điện tử là sử dụng que diêm. Các ký tự số sẽ được biểu diễn như sau:

Với một số lượng que diêm cho trước, hãy các định số nhỏ nhất và số lớn nhất mà bạn có thể biểu diễn được.

Chú ý:

- Bạn không được phép để thừa que diêm nào khi xếp.
- Các số biểu diễn không được bắt đầu bằng số 0

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test viết trên một dòng một số nguyên duy nhất không lớn hơn 100 là số que diêm bạn có.

Kết quả:

Với mỗi bộ test, output đưa ra hai số nguyên theo thứ tự là số nhỏ nhất và số lớn nhất có thể biểu diễn bởi số que diêm cho bởi input (mỗi số cách nhau một khoảng trống).

Ví dụ cho Input và Output:

Input	**Output**
4 3 6 7 15	7 7 6 111 8 711 108 7111111

BÀI 1.1.10: ĐẾM SỐ CHÍNH PHƯƠNG TRONG ĐOẠN

Viết chương trình đếm trong một đoạn giữa hai số nguyên có bao nhiêu số chính phương.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không quá 9 chữ số.

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra trên một dòng giá trị số các số chính phương đếm được.

Ví dụ:

Input	**Output**
2 23 199 2345 6789	10 34

BÀI 1.1.11: SỐ THUẦN NGUYÊN TỐ

Một số được coi là thuần nguyên tố nếu nó là số nguyên tố, tất cả các chữ số là nguyên tố và tổng chữ số của nó cũng là một số nguyên tố. Bài toán đặt ra là đếm xem trong một đoạn giữa hai số nguyên cho trước có bao nhiêu số thuần nguyên tố.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không vượt quá 9 chữ số.

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra số lượng các số thuần nguyên tố tương ứng.

Ví dụ

Input	**Ouput**

2 23 199 2345 6789	1 15

BÀI 1.1.12: SỐ CÓ TỔNG CHỮ SỐ CHIA HẾT CHO 10

Viết chương trình kiểm tra một số có thỏa mãn tính chất tổng chữ số của nó chia hết cho 10 hay không.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Mỗi bộ test viết trên một dòng một số nguyên dương, ít nhất 2 chữ số nhưng không quá 9 chữ số.

Kết quả:

Mỗi bộ test viết ra YES hoặc NO tùy thuộc kết quả kiểm tra.

Ví dụ:

Input	**Output**
3 3333 555555 123455	NO YES YES

BÀI 1.1.13: SỐ ĐẸP 7

Một số được coi là đẹp nếu nó là số thuận nghịch, tổng chữ số là số nguyên tố và tất cả các chữ số đều lẻ. Bài toán đặt ra là đếm xem trong một đoạn giữa hai số nguyên cho trước có bao nhiêu số đẹp như vậy.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Mỗi bộ test viết trên một dòng hai số nguyên dương tương ứng, cách nhau một khoảng trống. Các số đều không vượt quá 9 chữ số.

Kết quả:

Với mỗi bộ test viết ra số lượng các số thuần nguyên tố tương ứng.

Ví dụ

Input	**Output**
3 23 199 2345 6789 222222 99999999	4 0 311

BÀI 1.1.14:

Hãy viết chương trình tìm số các số tự nhiên N thỏa mãn đồng thời những điều kiện dưới đây ($N \le 2^{31}$):

- N là số có K chữ số (K≤15).
- N là số nguyên tố.
- Đảo ngược các chữ số của N cũng là một số nguyên tố.
- Tổng các chữ số của N cũng là một số nguyên tố.
- Mỗi chữ số của N cũng là các số nguyên tố ;
- Thời gian thực hiện chương trình không quá 1sec.

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số lượng các test (M≤100).
- M dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một số *K*. Hai số được viết cách nhau một vài khoảng trống.

Kết quả ra (Output): ghi lại M dòng trong file ketqua.out, mỗi dòng ghi lại bộ hai số số *N, X*. Trong đó *X* là số các số có N chữ số thỏa mãn yêu cầu của bài toán. Ví dự dưới đây minh họa cho file input và output của bài toán.

Input.in	**Output.out**	
5	2	0
2	3	8
3	4	15
4	5	46
5	7	359
7		

BÀI 1.1.15:

Trong ngày đầu tiên phát hành các số điện thoại di động, công ty viễn thông dự định khuyến mại cho N khách hàng đăng ký trước nhất các số điện thoại loại 1, M khách hàng kế tiếp số điện thoại loại 2 và K khách hàng cuối cùng các số điện thoại loại 3. Các số điện thoại loại 1, loại 2 và loại 3 có tính chất sau:

- Số loại 3 (Loại 3): là các số điện thoại mà sáu số cuối cùng của nó tạo thành một số thuận nghịch có sáu chữ số. Ví dụ số : 0913.104401.
- Số loại 2 (Loại 2): là các số điện thoại Loại 3 có tổng sáu số cuối cùng của nó là một số chia hết cho 10. Ví dụ số : 0913.104401.
- Số loại 1 (Loại 1): là các số điện thoại Loại 2 có tổng sáu số cuối cùng của nó không chứa bất kỳ số 0 nào. Ví dụ số : 0913.686686.

Bài toán được đặt ra là cho trước một phương án N, M, K, hãy trả lời "YES" nếu công ty thực hiện được, trả lời "NO" nếu công ty không thực hiện được.

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test là một bộ 3 số N, M, K được ghi trên một dòng. Các số được ghi cách nhau một vài khoảng trống.

Output: Với mỗi bộ test, viết ra trên một dòng giá trị "YES" hoặc "NO" tương ứng với phương án thực hiện được, hoặc phương án không thực hiện được.

Ví dụ cho Input và Output:

INPUT	OUTPUT
5 100 100 200 50 150 200 100 50 300 120 50 500 140 50 700	**NO** **NO** **YES** **YES** **NO**

BÀI 1.1.16:

Số N nguyên hệ cơ số ACM là những số nguyên thông thường sử dụng các ký hiệu từ *0, 1, ..,9* làm ký hiệu của hệ đếm (Ví dụ số 719_{ACM}). Nguyên tắc chung để đổi một số $A = (a_1, a_2, .., a_N)$ ở hệ cơ số ACM sang số ở hệ cơ số 10 được thực hiện như sau:

$$K_{10} = \sum_{i=1}^{N} a_i * (i!)$$, trong đó a_i chữ số tại vị trí thứ *i* của ở hệ cơ số ACM .

Ví dụ:

$A = 719_{ACM} = 9.(1!) + 1.(2!) + 7.(3!) = 53_{10}$.

Nhiệm vụ của bạn là viết một chương trình đọc một số nguyên ở hệ cơ số ACM rồi đổi số đó thành số hệ cơ số 10 .

Dữ liệu vào:

Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu tiên chứa một số nguyên không lớn hơn 100 là số lượng các bộ dữ liệu. Những dòng tiếp theo chứa các bộ dữ liệu. Mỗi bộ dữ liệu được viết trên một dòng. Mỗi dòng viết một số nhỏ hơn 2^{32} là số ở hệ cơ số ACM .

Dữ liệu ra:

Với mỗi bộ dữ liệu, ghi ra trên một dòng một số được chuyển đổi.

Ví dụ dữ liệu vào	**Ví dụ dữ liệu ra**
6 719 1 15 110 102 8	53 1 7 8 8 0

BÀI 1.1.17: SỐ MAY MẮN

Hoàng yêu thích các số may mắn. Ta biết rằng một số là số may mắn nếu biểu diễn thập phân của nó chỉ chứa các chữ số may mắn là 4 và 7. Ví dụ, các số 47, 744, 4 là số may mắn và 5, 17, 467 không phải. Hoàng muốn tìm số may mắn bé nhất có tổng các chữ số bằng n. Hãy giúp anh ấy

Dữ liệu vào: Dòng đầu ghi số bộ test, mỗi bộ test có một dòng chứa số nguyên n ($1 \le n \le 106$) — tổng các chữ số của số may mắn cần tìm.

Kết quả: In ra trên 1 dòng số may mắn bé nhất, mà tổng các chữ số bằng n. Nếu không tồn tại số thỏa mãn, in ra -1.

Ví dụ

Input	**Output**
2 11 10	47 -1

1.2. Các bài tập về mảng và ma trận

BÀI 1.2.1: SỐ CẶP BẰNG NHAU TRONG DÃY

Viết chương trình đếm các cặp số bằng nhau liên tiếp trong dãy số nguyên.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test có hai dòng:

- Dòng đầu ghi số phần tử của dãy, không quá 30
- Dòng tiếp theo ghi các phần tử của dãy, mỗi phần tử cách nhau một khoảng trống. Các phần tử không quá 100.

Kết quả: Ghi ra màn hình

Mỗi bộ test viết ra trên một dòng giá trị tổng chữ số tương ứng

Ví dụ:

Input	**Output**
2 4 1 3 3 4 12 1 2 3 3 3 3 4 4 5 5 5 1	1 6

BÀI 1.2.2: ĐẾM CÁC SỐ LỚN HƠN SỐ ĐỨNG TRƯỚC TRONG DÃY

Cho một dãy số nguyên dương có n phần tử (2<=n<=50). Hãy liệt kê số các phần tử trong dãy không nhỏ hơn các số đứng trước nó (tính cả phần tử đầu tiên).

Dữ liệu vào:

- Dòng 1 ghi số bộ test

- Với mỗi bộ test: dòng đầu tiên ghi số n. Dòng tiếp theo ghi n số nguyên dương của dãy (các số không vượt quá 1000).

Kết quả:

Ghi ra màn hình trên một dòng số các phần tử thỏa mãn.

Ví dụ:

Input	**Output**
2 7 3 5 6 8 4 2 9 15 9 8 123 7 11 14 18 21 399 10 5 4 1 2 3	5 3

BÀI 1.2.3: *ĐOÁN SỐ TIẾP THEO*

An và Bình chơi trò chơi số học đơn giản. Dãy số mà An đưa ra là A = {1,1,3,4,5,9,7,16,9,…}và đố Bình tìm ra số tiếp theo trong dãy đó. Rất nhanh chóng, Bình tìm được số tiếp theo là số 25. Bình đố lại An, nếu cho trước một số k không quá 100, hãy tính số đứng vị trí đó trong dãy đã cho (thứ tự trên dãy tính từ 1). Bạn hãy giúp An tính ra kết quả trên.

Dữ liệu vào:

Dòng đầu là số bộ test, không quá 20. Mỗi bộ test ghi trên một dòng số nguyên dương k.

Kết quả:

Với mỗi bộ test, đưa ra trên một dòng giá trị ở vị trí k của dãy.

Ví dụ:

C.IN	**Kết quả**
3 1 4 10	1 4 25

BÀI 1.2.4: TỔNG HAI ĐA THỨC

Cho hai đa thức *P(x) -bậc n* và *Q(x) -bậc m* có các hệ số nguyên, n và m không quá 100. Hãy viết chương trình tính tổng hai đa thức trên.

Dữ liệu vào:

Dòng đầu tiên chứa số nguyên N là số bộ dữ liệu ($1 \le N \le 10$).

Mỗi bộ dữ liệu gồm 4 dòng:

- Dòng 1 ghi số n là bậc của P. Dòng 2 ghi n+1 số nguyên tương ứng là hệ số của P từ P_0 đến P_n
- Dòng 3 ghi số m là bậc của Q. Dòng 4 ghi m+ 1 số nguyên tưng ứng là hệ số của Q, từ Q_0 đến Q_m

Kết quả: Ghi ra màn hình

Với mỗi bộ dữ liệu vào, in ra kết quả trên hai dòng: Dòng 1 ghi bậc của đa thức tổng. Dòng 2 ghi lần lượt các hệ số của đa thức tổng, tính từ 0.

Ví dụ:

D.IN	**Kết quả**
1 3 1 2 3 4 5 1 1 2 -2 3 3	5 2 3 5 2 3 3

BÀI 1.2.5: TÌM CÁC VỊ TRÍ BẰNG NHAU CỦA HAI MA TRẬN

Cho hai ma trận vuông A và B chỉ gồm số nguyên dương cấp n . Hãy viết chương trình tìm ra các vị trí bằng nhau trong hai ma trận (vị trí [i,j] được coi là bằng nhau nếu A[i,j]=B[i,j]).

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Với mỗi bộ test: Dòng đầu tiên ghi số n; n dòng tiếp theo ghi ma trận A; n dòng tiếp theo ghi ma trận B

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra một số nguyên dương m là số vị trí bằng nhau. m dòng tiếp theo ghi hai giá trị chỉ số của từng cặp (cách nhau một khoảng trống). (Chú ý: các chỉ số đều tính từ 0 đến n-1).

Ví dụ:

Input	**Output**
1 2 1 1 1 2 9 1 4 2	2 0 1 1 1

BÀI 1.2.6: *SỐ FIBONACI THỨ N*

Dãy Fibonaci được định nghĩa như sau: F(0) = 0, F(1)=1, …, F(n)=F(n-1)+F(n-2). Cho trước số nguyên dương n (không quá 45), hãy tính số Fibonaci thứ n.

Dữ liệu vào:

Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng số nguyên dương n.

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra giá trị số Fibonaci thứ n.

Ví dụ:

Input	**Output**
3 1 7 10	1 13 55

BÀI 1.2.7: TÍCH MA TRẬN VỚI CHUYỂN VỊ CỦA NÓ

Cho ma trận A chỉ gồm các số nguyên dương cấp n*m . Hãy viết chương trình tính tích của A với ma trận chuyển vị của A.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test. Với mỗi bộ test: Dòng đầu tiên ghi hai số n và m là bậc của ma trân a; n dòng tiếp theo, mỗi dòng ghi m số của một dòng trong ma trận A.

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra ma trận tích tương ứng, mỗi số cách nhau đúng một khoảng trống.

Ví dụ:

Input	**Output**
1 2 2 1 2 3 4	5 11 11 25

BÀI 1.2.8: SỐ XUẤT HIỆN NHIỀU LẦN NHẤT TRONG DÃY

Cho một dãy số nguyên dương không quá 100 phần tử, các giá trị trong dãy không quá 30000. Hãy xác định xem số nào là số xuất hiện nhiều lần nhất trong dãy. Chú ý: trong trường hợp nhiều số khác nhau cùng xuất hiện số lần bằng nhau và là lớn nhất thì in ra tất cả các số đó theo thứ tự xuất hiện trong dãy ban đầu.

Dữ liệu vào:

Dòng đầu là số bộ test, không quá 20.

Mỗi bộ test gồm hai dòng. Dòng đầu ghi số phần tử của dãy, dòng tiếp theo ghi các phần tử của dãy.

Kết quả: Ghi ra màn hình

Với mỗi bộ test, đưa ra số xuất hiện nhiều lần nhất trong dãy đã cho.

Ví dụ:

Input	**Output**
2 10 1 2 3 1 2 3 1 2 3 1 10 1 2 3 4 5 6 7 8 9 0	1 1 2 3 4 5 6 7 8 9 0

BÀI 1.2.9: *MA TRẬN ĐƠN VỊ*

Một ma trận vuông A cấp n chỉ gồm các số nguyên dương. A được gọi là ma trận đơn vị nếu tất cả các phần tử trong A đều là 0 hoặc 1. Viết chương trình kiểm tra xem một ma trận có đối xứng hay không.

Dữ liệu vào:

Dòng đầu tiên ghi số bộ test (không quá 10). Với mỗi bộ test: Dòng đầu tiên ghi số n là bậc của ma trân A; n dòng tiếp theo, mỗi dòng ghi n số của một dòng trong ma trận A.

Các giá trị đều không quá 100.

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra màn hình chữ YES nếu đó đúng là ma trận đơn vị, NO nếu ngược lại.

Ví dụ:

Input	**Output**
2 2 1 1 1 2 4 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0	NO YES

BÀI 1.2.10: *TỔNG TAM GIÁC*

Số tam giác thứ n là tổng của n số tự nhiên đầu tiên, T(n) = 1 + … + n. Nó cũng chính là số lượng điểm trong một mảng tam giác có cạnh là n, ví dụ như với T(4) thì:

X
X X
X X X
X X X X

Hãy viết chương trình tính tổng có trọng số của số tam giác:

$$W(n) = \sum_{k=1}^{n} k.T(k+1)$$

Dữ liệu vào:

Dòng đầu tiên chứa số nguyên N là số bộ dữ liệu ($1 \le N \le 1000$).

Mỗi bộ dữ liệu gồm 1 dòng chứa duy nhất 1 số nguyên n ($1 \le n \le 300$) là số điểm trên cạnh của tam giác.

Kết quả: Ghi ra màn hình

Với mỗi bộ dữ liệu vào, in ra trên một dòng số thứ tự của bộ dữ liệu (1 đến N), một dấu cách, giá trị của n trong bộ dữ liệu, một dấu cách và tổng có trọng số W(n) của những số tam giác tương ứng với n.

Ví dụ:

D.IN	**Kết quả**
4 3 4 5 10	1 3 45 2 4 105 3 5 210 4 10 2145

BÀI 1.2.11: *TRỘN HAI DÃY VÀ SẮP XẾP*

Cho hai dãy số nguyên dương A và B không quá 100 phần tử, các giá trị trong dãy không quá 30000 và số phần tử của hai dãy bằng nhau. Hãy trộn hai dãy với nhau sao cho dãy A

được đưa vào các vị trí có chỉ số chẵn, dãy B được đưa vào các vị trí có chỉ số lẻ. Đồng thời, dãy A được sắp xếp tăng dần, còn dãy B được sắp xếp giảm dần. *(Chú ý: chỉ số tính từ 0)*

Dữ liệu vào

- Dòng 1 ghi số bộ test
- Với mỗi bộ test: dòng đầu tiên ghi số n. Dòng tiếp theo ghi n số nguyên dương của dãy A. Dòng tiếp theo ghi n số nguyên dương của dãy B

Kết quả

Với mỗi bộ test, đưa ra thứ tự bộ test và dãy kết quả.

Ví dụ:

Input	**Output**
2 5 1 2 3 1 2 3 1 2 3 1 4 4 2 7 1 5 6 2 8	Test 1: 1 3 1 3 2 2 2 1 3 1 Test 2: 1 8 2 6 4 5 7 2

BÀI 1.2.12: MA TRẬN XOÁY ỐC NGƯỢC

Ma trận xoáy ốc ngược cấp N là ma trận vuông có N*N phần tử. Các số được điền vào ma trận trận theo chiều kim đồng hồ theo thứ tự giảm dần. Hãy xây dựng ma trận xoáy ốc ngược sao cho các số trong ma trận đều là số nguyên dương chia hết cho 3 (tính cả số 0).

Dữ liệu vào

- Dòng 1 ghi số bộ test
- Mỗi bộ test ghi số N (1<N<20).

Kết quả

Ghi ra thứ tự bộ test và ma trận xoáy ốc ngược tương ứng

Ví dụ:

Input	**Output**
1 3	Test 1: 24 21 18 3 0 15 6 9 12

BÀI 1.2.13: *ĐẾM CÁC SỐ NGUYÊN TỐ TRONG DÃY*

Cho dãy số A có n phần tử chỉ bao gồm các số nguyên dương (không quá 9 chữ số). Hãy xác định các số nguyên tố trong dãy và đếm xem mỗi số xuất hiện bao nhiêu lần.

Dữ liệu vào

Dòng đầu tiên ghi số bộ test. Với mỗi bộ test: dòng đầu ghi số n; dòng tiếp theo ghi n số của dãy,

Kết quả

Với mỗi bộ test ghi ra thứ tự bộ test, sau đó lần lượt là các số nguyên tố trong dãy theo thứ tự từ nhỏ đến lớn và số lần xuất hiện của nó.

Ví dụ

Input	**Output**
2 10 1 2 3 3 2 1 3 2 7 8 8 10 7 6 7 3 2 5 7	Test 1: 2 xuat hien 3 lan 3 xuat hien 3 lan 7 xuat hien 1 lan Test 2: 2 xuat hien 1 lan 3 xuat hien 1 lan 5 xuat hien 1 lan 7 xuat hien 2 lan

BÀI 1.2.14: *TÍCH MA TRẬN VỚI CHUYỂN VỊ CỦA NÓ*

Cho một số nguyên dương N không quá 20. Ma trận vuông A cấp N*N được tạo theo mẫu trong bảng dưới. Viết chương trình tính tích của A với chuyển vị của A.

Với N = 4 1 0 0 0 1 2 0 0 1 2 3 0 1 2 3 4	Với N = 5 1 0 0 0 0 1 2 0 0 0 1 2 3 0 0 1 2 3 4 0 1 2 3 4 5

Dữ liệu vào:

Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng số N (1<N<20).

Kết quả:

Ghi thứ tự bộ test, sau đó là N hàng ghi ma trận kết quả. Tiếp theo là một dòng trống.

Ví dụ

Input	Output
1 4	Test 1: 1 1 1 1 1 5 5 5 1 5 14 14 1 5 14 30

BÀI 1.2.15: *ĐỖ XE TỐI ƯU*

Khi mua sắm trên khu Long Street, Michael thường đỗ xe của mình ở một số vị trí ngẫu nhiên và đi bộ vào cửa hàng. Bạn hãy giúp Michael chọn một chỗ đỗ xe để khoảng cách phải đi bộ khi mua hàng là nhỏ nhất.

Long Street có thể coi như là một đường thẳng mà tất cả những điểm mua hàng là các điểm có tọa độ nguyên.

Dữ liệu vào:

Dòng đầu tiên chứa một số nguyên $1 \le t \le 100$ là số lượng bộ test. Mỗi bộ test gồm 2 dòng, dòng đầu tiên ghi số cửa hàng n mà Michael muốn qua mua hàng, $1 \le n \le 20$ và dòng thứ hai ghi n số nguyên là tọa độ các điểm này trên phố Long Street, $0 \le x_i \le 99$.

Kết quả:

Với mỗi bộ test, in ra màn hình trên một dòng khoảng cách nhỏ nhất phải đi bộ với chỗ đỗ xe tối ưu.

Ví dụ cho Input và Output:

Input	**Ouput**
2 4 24 13 89 37 6 7 30 41 14 39 42	152 70

BÀI 1.2.16:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ W trong tập văn bản D_1 *và* D_2, ký hiệu là $P(W)$ được tính theo công thức:

$P(W) = \dfrac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ W trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản data1.in và data2.in. *Sử dụng CTDL Mảng*, hãy tìm tập các từ và tần xuất xuất hiện của mỗi từ hoặc xuất hiện trong data1.in hoặc xuất hiện trong data2.in. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ W tìm được theo yêu cầu của bài toán.
- K dòng kế tiếp, mỗi dòng ghi lại một từ W và tần xuất xuất hiện $P(W)$ thỏa mãn yêu cầu của bài toán. W và $P(W)$ được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	3 AB 0.2 AC 0.2 AD 0.2

BÀI 1.2.17:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ *W* trong tập văn bản D_1 *và* D_2, ký hiệu là *P*(*W*) được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ *W* trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản *data1.in* và *data2.in*. *Sử dụng CTDL Mảng*, hãy tìm tập các từ và tần xuất xuất hiện của các từ xuất hiện trong file *data1.in* nhưng không xuất hiện trong file *data2.in*. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên *K* là số từ *W* tìm được theo yêu cầu của bài toán.
- *K* dòng kế tiếp, mỗi dòng ghi lại một từ *W* và tần xuất xuất hiện *P*(*W*) thỏa mãn yêu cầu của bài toán. *W* và *P*(*W*) được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	7 AB 0.2 AC 0.2 AD 0.2 AE 0.1 AF 0.1 AH 0.1 AK 0.1

BÀI 1.2.18:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ *W* trong tập văn bản D_1 *và* D_2, ký hiệu là *P*(*W*) được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ W trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản data1.in và data2.in. *Sử dụng CTDL Mảng*, hãy tìm tập các từ và tần xuất xuất hiện của mỗi từ đồng thời trong cả hai tập data1.in và data2.in. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ W tìm được theo yêu cầu của bài toán.
- K dòng kế tiếp, mỗi dòng ghi lại một từ W và tần xuất xuất hiện $P(W)$ thỏa mãn yêu cầu của bài toán. W và $P(W)$ được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	2 AE 0.1 AF 0.1

BÀI 1.2.19:

Cho tập các số tự nhiên ghi lại trong file data.in theo từng dòng; mỗi dòng ghi lại nhiều nhất 10 số; hai số được viết cách nhau một vài khoảng trống; mỗi số tự nhiên có thể xuất hiện nhiều lần ở những vị trí khác nhau trong file. Ta gọi tần xuất xuất hiện số tự nhiên X trong file data.in là $P(X) = \frac{N(X)}{K}$; trong đó N(X) là số lần xuất hiện số tự nhiên X trong file data.in, K là số các số tự nhiên trong file data.in. *Sử dụng CTDL mảng*, hãy viết chương trình tìm số vừa nguyên tố vừa thuận nghịch X có P(X) lớn nhất đầu tiên tìm được trong file data.in. Ví dụ với file data.in được cho dưới đây, ta tìm được số X = 131 vừa là số nguyên tố, vừa là số thuận nghịch với tần xuất xuất hiện lớn nhất là P(X) = 3/30 = 0.1.

data.in

10 131 171 13 37 27 30 23 77 444

10 131 171 20 23 77 23 27 77 444

10 131 171 20 23 77 23 27 77 444

BÀI 1.2.20:

Cho tập các số tự nhiên trong file data.in được ghi theo từng dòng, mỗi dòng ghi nhiều nhất 5 số, hai số được viết cách nhau một vài khoảng trống. Biết rằng, mỗi số tự nhiên trong file data.in hoặc là số nguyên tố, hoặc là số thuận nghịch và có thể xuất hiện nhiều lần ở những vị trí khác nhau trong file. *Sử dụng cấu trúc dữ liệu mảng*, hãy viết chương trình tách tập các số và đếm số lần xuất hiện của mỗi số trong file data.in thành 3 file ketqua1.out, ketqua2.out, ketqua3.out thỏa mãn những yêu cầu dưới đây.

a) File ketqua1.out ghi lại các số nguyên tố nhưng không là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

b) File ketqua2.out ghi lại các số thuận nghịch nhưng không là nguyên tố cùng với số lần xuất hiện của nó trong file data.in;

c) File ketqua3.out ghi lại các số vừa là số nguyên tố vừa là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

d) Khuôn dạng của các file kết quả được qui định như sau:

- Dòng đầu tiên của mỗi file ghi lại số các số của mỗi file kết quả;
- Những dòng kế tiếp mỗi dòng ghi lại một số cùng với số lần xuất hiện của nó trong file data.in. Hai số được viết cách nhau một vài khoảng trống.

Ví dụ dưới đây minh họa cho các file data.in, ketqua1.out, ketqua2.out và ketqua3.out.

Data.in
```
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
```

Ketqua1.out
```
2
10007 4
10009 4
```

Ketqua2.out
```
2
10801 4
10901 4
```

Ketqua3.out
```
2
13831 2
34543 2
```

BÀI 1.2.21:

Cho một hình vuông gồm 5 x 5 hình vuông đơn vị như hình vẽ sau:

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

Hãy điền các chữ số từ 0 đến 9 vào mỗi hình vuông đơn vị sao cho những điều kiện sau được thỏa mãn:

(i) Đọc từ trái sang phải theo hàng ta nhận được năm số nguyên tố có năm chữ số;

(ii) Đọc từ trên xuống dưới theo cột ta nhận được năm số nguyên tố có năm chữ số;

(iii) Đọc theo hai đường chéo chính từ trái sang phải ta nhận được hai số nguyên tố có năm chữ số;

(iv) Tổng các chữ số trong mỗi số nguyên tố đều bằng S cho trước (trong ví dụ trên S=11).

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test viết trên một dòng một bộ đôi hai số S, T tương ứng với tổng các chữ số trong mỗi số nguyên tố và giá trị ô vuông trên cùng góc trái.

Output: Với mỗi bộ test, output đưa ra một số duy nhất là số các lời giải của bài toán. Mỗi lời giải của bài toán là một hình vuông gồm 5×5 hình vuông đơn vị. Hai lời giải được xem là khác nhau nếu tồn tại một ô vuông đơn vị ở cùng một vị trí có giá trị khác nhau.

Ví dụ cho Input và Output của bài toán:

INPUT	OUTPUT
3 11 3 11 9 **11 8**	5 0 0

BÀI 1.2.22:

Trò chơi Rubik trên mặt phẳng có thể được mô tả bằng dãy các số từ 1 đến 8 và được chia thành hai hàng. Ta qui ước trạng thái khởi đầu S của Rubik là trạng thái trong hình 1. Người ta đã chứng minh được rằng, từ trạng thái khởi đầu S ta có thể dịch chuyển thành trạng thái T bất kỳ trên các thao tác A, B, C dưới đây:

A : Đổi hàng trên cho hàng dưới (Hình 2).

B : Tạo nên một hoán vị vòng trên mỗi hàng (Hình 3).

C : Quay 90 độ 4 ô ở giữa (Hình 4)

Hình 1. Trạng thái khởi đầu S

1	2	3	4
8	7	6	5

Hình 2. Thao tác A				**Hình 3.** Thao tác B				**Hình 4.** Thao tác C			
8	7	6	5	4	1	2	3	1	7	2	4
1	2	3	4	5	8	7	6	8	6	3	5

Yêu cầu: Cho trước một trạng thái kết thúc T. Hãy chuyển S thành T sao cho số các phép A, B, C thực hiện là ít nhất.

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức trên một dòng là giá trị trạng thái kết thúc T.

Output: Với mỗi bộ test, viết ra trên một dòng số các bước A, B, C ít nhất.

INPUT	OUTPUT
2 **2 6 8 4 5 7 3 1** **1 2 3 4 5 6 7 8**	**7** **0**

BÀI 1.2.23: *SẮP XẾP CHÈN*

Hãy thực hiện thuật toán sắp xếp chèn trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán.

Dữ liệu vàoDòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100).

Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trống.

Ví dụ:

Input	**Output**
4 5 7 3 2	Buoc 0: 5 Buoc 1: 5 7

	Buoc 2: 3 5 7 Buoc 3: 2 3 5 7

BÀI 1.2.24: SẮP XẾP NỔI BỌT

Hãy thực hiện thuật toán sắp xếp nổi bọt trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán.

Dữ liệu vào Dòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100).

Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trống.

Ví dụ:

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

BÀI 1.2.25: CHỌN MÁY BAY

Một hãng hàng không có quy tắc phục vụ khách hàng rất đặc biệt. *Thứ nhất*, khách hàng được chọn máy bay mà mình sẽ đi. *Thứ hai*, nếu máy bay mà khách hàng chọn còn x ghế trống thì chi phí phải trả sẽ là x USD. Các khách hàng sẽ lần lượt chọn máy bay cho riêng mình, người thứ nhất, người thứ hai, ... cho đến người cuối cùng. Hãy tính xem hãng máy bay này thu được nhiều nhất và ít nhất bao nhiêu tiền.

Dữ liệu vào Dòng đầu ghi số bộ test. Với mỗi bộ test, dòng đầu tiên gồm hai số N, M($1 \le n, m \le 1000$) lần lượt là số khách hàng và số máy bay. Dòng tiếp theo chứa các số nguyên a[i] ($1 \le a[i] \le 1000$) là số ghế trống của máy bay i. *Dữ liệu luôn đảm bảo có ít nhất N ghế trống.*

Kết quả: Mỗi bộ test in ra 2 số nguyên, là số tiền nhiều nhất và ít nhất mà hãng máy bay thu được.

Ví dụ

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

BÀI 1.2.26: TÍCH LỚN NHẤT

Cho bảng N hàng, M cột, gồm các số nguyên 32-bit có dấu. Các cột đánh số từ 1đến M, bắt đầu từ bên trái. Gọi A[i] (1≤i≤M) là tích tất cả các số của cột i. Tìm cột i mà A[i] lớn nhất.

Dữ liệu vào: Gồm nhiều bộ test, mỗi bộ test có dạng:

- Dòng 1: 2 số nguyên m và n (1≤m≤20,1≤n≤1000)
- N dòng sau, mỗi dòng chứa m số nguyên có dấu.

Input kết thúc bởi dòng chứa 2 số 0.

Kết quả: Với mỗi bộ test, in trên 1 dòng chứa cột i mà A[i] lớn nhất, nếu có nhiều cột thỏa mãn, in ra cột có chỉ số lớn nhất.

Ví dụ:

Input	Output
3 3 20 10 30 15 20 20 30 30 20 3 2 2 -2 2 2 -2 2 0 0	3 3

BÀI 1.2.27: LEO NÚI

Có N (1≤N≤25000) người leo lên và leo xuống trên 1 ngọn núi. Người i mất U(i) thời gian leo lên và D(i) thời gian để leo xuống. Trong một thời điểm chỉ có tối đa người 1 người có thể lên và tối đa 1 người có thể xuống (có thể 1 người lên, 1 người xuống). Những người khác có thể đứng chờ ở đỉnh ngọn núi. Thứ tự đi xuống có thể khác thứ tự đi lên. Bạn hãy xác định xem thời gian tối thiểu để cho N người lên và xuống ngọn núi là bao nhiêu.

Dữ liệu vào: Dòng 1 ghi số N. N dòng tiếp theo chứa 2 số U(i) và D(i) (1 ≤ U(i) , D(i) ≤ 50000)

Kết quả: Ghi ra thời gian tối thiểu có thể.

Ví dụ: *(Giải thích: đi lên và xuống theo thứ tự người 3->1->2)*

Input	Output
3 6 4 8 1 2 3	17

1.3. Các bài tập về xâu ký tự

BÀI 1.3.1: *PHÉP CỘNG SỐ NGUYÊN*

Viết chương trình cộng hai số nguyên dương bất kỳ (không quá 256 chữ số).

Dữ liệu vào:

Dòng 1 ghi số bộ test. Mỗi bộ test gồm 2 dòng, mỗi dòng ghi một số nguyên dương

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra một số nguyên dương là tổng hai số đã cho (số này cũng không quá 256 chữ số).

Ví dụ:

Input	**Output**
3 12 100 1212 8888 1212121212121212 45678978	112 10100 1212121212257800190

BÀI 1.3.2: *HIỆU HAI SỐ NGUYÊN LỚN*

Viết chương trình tính hiệu của hai số nguyên lớn. Chú ý: luôn luôn lấy số lớn hơn trừ đi số nhỏ hơn.

Dữ liệu vào:

- Dòng 1 ghi số bộ test
- Mỗi bộ test gồm 2 dòng, mỗi dòng ghi một số không quá 1000 chữ số..

Kết quả

Với mỗi bộ test ghi ra kết quả tính toán tương ứng

Ví dụ:

Input	**Ouput**
3 456 789 10000000000000000000000000000 01 99 12345678901234567890123456 7890 1234567890	333 99999999999999999999999999 902 12345678901234567890000000 000

BÀI 1.3.3: *SỐ ĐẸP*

Một số được coi là đẹp nếu đó là số thuận nghịch và chỉ toàn các chữ số nguyên tố. Viết chương trình đọc vào các số nguyên dương có không quá 500 chữ số và kiểm tra xem số đó có đẹp hay không.

Dữ liệu vào

Dòng đầu tiên ghi số bộ test.

Mỗi bộ test viết trên một dòng số nguyên dương n không quá 500 chữ số.

Kết quả

Mỗi bộ test viết ra trên một dòng chữ YES nếu đó là số đẹp, chữ NO nếu ngược lại

Ví dụ

Input	**Output**
4 123456787654321 235755557352 22223333555577772357753277775555333322222 56789087654678923543245435436542343 5	NO YES YES NO

BÀI 1.3.4: *DANH SÁCH ĐIỆN THOẠI NHẤT QUÁN*

Cho một danh sách các số điện thoại, hãy xác định danh sách này có số điện thoại nào là phần trước của số khác hay không? Nếu không thì danh sách này được gọi là nhất quán. Giả sử một danh sách có chứa các số điện thoại sau:

- Số khẩn cấp: 911
- Số của alice: 97 625 999

- Số của Bob: 91 12 54 26

Trong trường hợp này, ta không thể gọi cho Bob vì tổng đài sẽ kết nối bạn với đường dây khẩn cấp ngay khi bạn quay 3 số đầu trong số của Bob, vì vậy danh sách này là không nhất quán.

Dữ liệu vào:

Dòng đầu tiên chứa một số nguyên $1 \leq t \leq 40$ là số lượng bộ test. Mỗi bộ test sẽ bắt đầu với số lượng số điện thoại n được ghi trên một dòng, $1 \leq n \leq 10000$. Sau đó là n dòng, mỗi dòng ghi duy nhất 1 số điện thoại. Một số điện thoại là một dãy không quá 10 chữ số.

Kết quả:

Với mỗi bộ dữ liệu vào, in ra "YES" nếu danh sách nhất quán và "NO" trong trường hợp ngược lại.

Ví dụ cho Input và Output:

Input	Output
2 3 911 97625999 91125426 5 113 12340 123440 12345 98346	NO YES

BÀI 1.3.5: TẬP TỪ RIÊNG CỦA HAI XÂU

Cho hai xâu ký tự S1 và S2. Hãy viết chương trình tìm các từ chỉ xuất hiện trong S1 mà không xuất hiện trong S2. Chú ý: mỗi từ chỉ liệt kê 1 lần.

Dữ liệu vào

Dòng 1 ghi số bộ test. Mỗi bộ test gồm 2 dòng, mỗi dòng ghi một xâu ký tự độ dài không quá 200, chỉ bao gồm các ký tự viết thường và các khoảng trống.

Kết quả

Với mỗi bộ test ghi ra các từ có trong S1 mà không có trong S2. Các từ được ghi theo thứ tự từ điển.

Ví dụ:

Input	**Output**
2 abc ab ab ab abcd ab abc aaa xyz ab zzz abc dd dd abc xyz dd ttt sas cdc	abcd aaa ab abc zzz

BÀI 1.3.6: CHUẨN HÓA XÂU HỌ TÊN 1

Một xâu họ tên được coi là viết chuẩn nếu chữ cái đầu tiên mỗi từ được viết hoa, các chữ cái khác viết thường. Các từ cách nhau đúng một dấu cách và không có khoảng trống thừa ở đầu và cuối xâu. Hãy viết chương trình đưa các xâu họ tên về dạng chuẩn.

Dữ liệu vào:

Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng xâu ký tự họ tên, không quá 80 ký tự.

Kết quả (ghi ra màn hình):

Với mỗi bộ test ghi ra xâu ký tự họ tên đã chuẩn hóa.

Ví dụ:

Input	**Output**
3 nGuYEN vAN naM tRan TRUNG hiEU vO le hOA bINh	Nguyen Van Nam Tran Trung Hieu Vo Le Hoa Binh

BÀI 1.3.7: BÀI G: CHUẨN HÓA XÂU HỌ TÊN 2

Các cán bộ, giảng viên PTIT khi tham gia hội nghị quốc tế sẽ được viết lại xâu họ tên theo dạng chuẩn trong đó họ được viết sau cùng, phân tách với phần tên đệm và tên bởi dấu

phẩy. Các chữ cái của phần họ đều viết hoa. Hãy viết chương trình chuẩn hóa các xâu họ tên theo yêu cầu nêu trên.

Dữ liệu vào:

- Dòng 1 ghi số N là xâu họ tên trong danh sách
- N dòng tiếp theo ghi lần lượt các xâu họ tên (không quá 50 ký tự)

Kết quả: ghi lại các xâu chuẩn, mỗi xâu một dòng.

Ví dụ:

Input	Output
4 nGUYEn quaNG vInH tRan thi THU huOnG nGO quoC VINH lE tuAn aNH	Quang Vinh, NGUYEN Thi Thu Huong, TRAN Quoc Vinh, NGO Tuan Anh, LE

BÀI 1.3.8: ĐỊA CHỈ EMAIL PTIT

Địa chỉ email của các cán bộ, giảng viên PTIT được tạo ra bằng cách viết đầy đủ tên và ghép với các chữ cái đầu của họ và tên đệm. Nếu có nhiều người cùng email thì từ người thứ 2 sẽ thêm số thứ tự vào email đó.

Cho trước các xâu họ tên (có thể không chuẩn). Hãy tạo ra các địa email tương ứng.

Dữ liệu vào:

- Dòng 1 ghi số N là xâu họ tên trong danh sách
- N dòng tiếp theo ghi lần lượt các xâu họ tên (không quá 50 ký tự)

Kết quả: ghi lại các email được tạo ra.

Ví dụ:

Input	Output
4 nGUYEn quaNG vInH tRan thi THU huOnG nGO quoC VINH lE tuAn aNH	vinhnq@ptit.edu.vn huongttt@ptit.edu.vn vinhnq2@ptit.edu.vn anhlt@ptit.edu.vn

BÀI 1.3.9: SỐ NGUYÊN TỐ LỚN NHẤT

Cho một số nguyên không quá 255 chữ số. Hãy tìm số nguyên tố lớn nhất được tạo thành bởi một dãy liên tiếp các chữ số trong số nguyên ban đầu. Chỉ xét các số nguyên tố trong đoạn [2, 100000].

Dữ liệu vào: Mỗi bộ test có một dòng chứa 1 số nguyên dương (tối đa 255 số). Bộ test cuối cùng chứa số 0 (không cần xử lý)
Kết quả: Với mỗi bộ test in ra kết quả tìm được trên một dòng.
Ví dụ:

Input	Output
11245 91321150448 1226406 0	11 1321 2

1.4. Các bài tập về file và cấu trúc

BÀI 1.4.1: *ĐẾM TỪ KHÁC NHAU*

Viết chương trình đếm số từ khác nhau của một file văn bản.

Dữ liệu vào: File: A.IN

Gồm nhiều dòng, chỉ bao gồm các chữ cái.

Kết quả: Ghi ra màn hình

Viết ra số từ khác nhau trong file.

Ví dụ:

A.IN	Kết quả
Xin chao cac ban Xin moi cac ban tap trung hoc tap	8

BÀI 1.4.2: *ĐẾM SỐ NGUYÊN TỐ KHÁC NHAU*

Cho một file văn bản chỉ bao gồm các số nguyên không quá 9 chữ số. Viết chương trình đếm xem trong file đó có bao nhiêu số nguyên tố khác nhau.

Dữ liệu vào: File: B.IN

Gồm nhiều dòng, chỉ bao gồm các số nguyên, cách nhau một hoặc một vài khoảng trống.

Kết quả: Ghi ra màn hình

Viết ra số các số nguyên tố khác nhau.

Ví dụ:

B.IN	**Kết quả**
11 13 14 16 18 21 13 23 99 88 77 66 13 13 13 13 23 23 23 13 24 25 26 27 28 97	4

BÀI 1.4.3: TÌM TỪ THUẬN NGHỊCH DÀI NHẤT TRONG FILE VĂN BẢN

Cho một file văn bản bất kỳ. Hãy tìm ra từ thỏa mãn tính chất ***thuận nghịch có độ dài lớn nhất*** trong file đó và cho biết từ đó ***xuất hiện bao nhiêu lần***. Nếu có nhiều từ cùng có độ dài lớn nhất thì in ra tất cả các từ đó theo thứ tự xuất hiện trong file ban đầu.

Dữ liệu vào: File C.IN

Gồm một đoạn văn bản bất kỳ. Không quá 1000 từ.

Kết quả (ghi ra màn hình):

- Ghi ra trên một dòng từ thuận nghịch có độ dài lớn nhất và số lần xuất hiện của nó.
- Nếu có nhiều từ cùng có độ dài lớn nhất thì các từ được liệt kê theo thứ tự xuất hiện ban đầu.

Ví dụ:

C.IN	**KẾT QUẢ**
AAA BAABA HDHDH ACBSD SRGTDH DDDDS DUAHD AAA AD DA HDHDH AAA AAA AAA AAA DDDAS HDHDH HDH AAA AAA AAA AAA AAA AAA AAA AAA DHKFKH DHDHDD HDHDHD DDDHHH HHHDDD TDTD	HDHDH 3

BÀI 1.4.4: TÌM TỪ DÀI NHẤT TRONG FILE

Cho một file văn bản bất kỳ. Hãy tìm ra từ có độ dài lớn nhất trong file. Nếu có nhiều từ khác nhau có độ dài bằng nhau và bằng giá trị lớn nhất thì in ra tất cả các từ đó theo thứ tự

xuất hiện trong file dữ liệu vào (nhưng một từ dù xuất hiện nhiều lần cũng chỉ được liệt kê một lần).

Dữ liệu vào: File D.IN

Gồm một đoạn văn bản bất kỳ. Không quá 1000 từ.

Kết quả:

Ghi ra màn hình từ dài nhất và độ dài của nó, cách nhau một khoảng trống. Nếu có nhiều từ như vậy thì liệt kê lần lượt các từ theo thứ tự xuất hiện trong file ban đầu.

Ví dụ:

D.IN	**KẾT QUẢ**
Tiet hoc cuoi cung da ket thuc. Mon hoc Tin hoc co so 2 da ket thuc. Cac ban co gang on tap tot de thi dat ket qua cao. Chuc cac ban ngay cang gat hai duoc nhieu thanh cong tren con duong da chon	thuc. 5 nhieu 5 thanh 5 duong 5

BÀI 1.4.5: TÌM TỪ THUẬN NGHỊCH DÀI NHẤT TRONG FILE VĂN BẢN

Cho một file văn bản bất kỳ. Hãy tìm ra từ thỏa mãn tính chất ***thuận nghịch có độ dài lớn nhất*** trong file đó và cho biết từ đó ***xuất hiện bao nhiêu lần***. Nếu có nhiều từ cùng có độ dài lớn nhất thì in ra tất cả các từ đó theo thứ tự xuất hiện trong file ban đầu.

Dữ liệu vào: File E.IN

Gồm một đoạn văn bản bất kỳ. Không quá 1000 từ.

Kết quả (ghi ra màn hình):

- Ghi ra trên một dòng từ thuận nghịch có độ dài lớn nhất và số lần xuất hiện của nó.
- Nếu có nhiều từ cùng có độ dài lớn nhất thì các từ được liệt kê theo thứ tự xuất hiện ban đầu.

Ví dụ:

E.IN	**KẾT QUẢ**

AAA BAABA HDHDH ACBSD SRGTDH DDDDS DUAHD AAA AD DA HDHDH AAA AAA AAA AAA DDDAS HDHDH HDH AAA AAA AAA AAA AAA AAA AAA AAA DHKFKH DHDHDD HDHDHD DDDHHH HHHDDD TDTD	HDHDH 3

BÀI 1.4.6: *SẮP XẾP THÍ SINH*

Hãy sắp xếp danh sách thí sinh đã có trong file theo tổng điểm giảm dần.

Mỗi thí sinh gồm các thông tin:

- Mã thí sinh: là một số nguyên, tự động tăng. Tính từ 1.
- Tên thí sinh, ngày sinh
- Điểm môn 1, điểm môn 2, điểm môn 3

Dữ liệu vào: File: F.IN

Dòng đầu chứa số thí sinh. Mỗi thí sinh viết trên 3 dòng:

- Dòng 1: Tên thí sinh
- Dòng 2: Ngày sinh
- Dòng 3,4,5: 3 điểm thi tương ứng. Các điểm thi đều đảm bảo hợp lệ (từ 0 đến 10).

Kết quả: Ghi ra màn hình

In ra danh sách thí sinh đã sắp xếp theo tổng điểm giảm dần. Nếu 2 thí sinh bằng điểm nhau thì thí sinh nào xuất hiện trước trong file sẽ viết trước. Mỗi thí sinh viết trên một dòng gồm: mã, tên, ngày sinh và tổng điểm. Các thông tin cách nhau đúng 1 khoảng trống. Điểm tổng được làm tròn đến 1 số sau dấu phẩy.

Ví dụ

F.IN	**Kết quả**
3 Nguyen Van A 12/12/1994 3.5 7.0 5.5	2 Nguyen Van B 1/9/1994 26.5 1 Nguyen Van A 12/12/1994 16.0 3 Nguyen Van C 6/7/1994 14.0

Nguyen Van B 1/9/1994 7.5 9.5 9.5 Nguyen Van C 6/7/1994 4.5 4.5 5.0	

BÀI 1.4.7: SẮP XẾP MẶT HÀNG

Hãy sắp xếp danh sách các mặt hàng đã có trong file theo lợi nhuận giảm dần.

Mỗi mặt hàng gồm các thông tin:

- Mã mặt hàng: là một số nguyên, tự động tăng. Tính từ 1.
- Tên mặt hàng, nhóm hàng: là các xâu ký tự
- Giá mua, giá bán: là các số thực (không quá 9 chữ số)

Dữ liệu vào: File: H.IN

Dòng đầu chứa số mặt hàng. Mỗi mặt hàng viết trên 4 dòng:

- Dòng 1: Tên mặt hàng
- Dòng 2: Nhóm hàng
- Dòng 3: Giá mua.
- Dòng 4: Giá bán

Kết quả: Ghi ra màn hình

In ra danh sách mặt hàng đã sắp xếp theo lợi nhuận giảm dần (lợi nhuận tính bằng giá bán trừ đi giá mua). Nếu 2 mặt hàng lợi nhuận bằng nhau thì mặt hàng nào xuất hiện trước trong file sẽ viết trước. Mỗi mặt hàng viết trên một dòng gồm: mã, tên, nhóm hàng và lợi nhuận. Các thông tin cách nhau đúng 1 khoảng trống.

Ví dụ

D.IN	**Kết quả**
`3` `May tinh SONY VAIO`	`2 Tu lanh Side by Side Dien lanh 7699` `1 May tinh SONY VAIO Dien tu 1299`

Dien tu 16400 17699 Tu lanh Side by Side Dien lanh 18300 25999 Banh Chocopie Tieu dung 27.5 37	3 Banh Chocopie Tieu dung 10.5

II. Lập trình dựa vào kỹ thuật duyệt và đệ qui

1.5. Kỹ thuật vét cạn

BÀI 2.1.1: *BÀI TOÁN CỔ: VỪA GÀ VỪA CHÓ*

Vừa gà vừa chó, bó lại cho tròn 36 con, một trăm chân chẵn. Hỏi số con mỗi loại. Hãy sử dụng phương pháp vét cạn để giải bài toán.

Dữ liệu vào:

Số con 36.

Số chân 100.

Kết quả:

Ghi ra màn hình số con gà, số con chó. Mỗi loại một dòng.

BÀI 2.1.2: *TÌM CỰC ĐẠI CHO HÀM SỐ*

Viết chương trình tìm $X = (x_1, x_2,..,x_n)$ và giá trị $f(X)$ của hàm

$f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} c_i x_i$ đạt giá trị lớn nhất. Trong đó,

$X = (x_1, x_2, \dots, x_n) \in D = \{\sum_{i=1}^{n} a_i x_i \leq b; x_i \in \{0,1\}\}$, c_i, a_i, b là các số nguyên dương, $n \leq 100$.

Dữ liệu vào:

Dữ liệu vào n, c_j, a_j, b được cho trong file *data.in* theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên n và b. Hai số được ghi cách nhau bởi một vài ký tự trống;
- Dòng kế tiếp ghi lại n số c_i ($i=1, 2, .., n$). Hai số được ghi cách nhau bởi một vài ký tự trống;
- Dòng cuối cùng ghi lại n số a_i ($i = 1, 2, ..,n$). Hai số được ghi cách nhau bởi một vài ký tự trống.

Kết quả:

Giá trị tối ưu $f(x_1,x_2,..,x_n)$ và phương án tối ưu $X = (x_1 , x_2,..,x_n)$ tìm được ghi lại trong file *ketqua.out* theo khuôn dạng sau:

- Dòng đầu tiên ghi lại giá trị tối ưu f(x1,x2,..,xn);
- Dòng kế tiếp ghi lại phương án tối ưu X = (x1, x2,..,xn). Hai phần tử khác nhau của X được ghi cách nhau bởi một vài khoảng trống.

Ví dụ:

Ví dụ dưới đây sẽ minh họa cho file data.in và ketqua.out của bài toán:

Data.in	**Ketqua.out**
4 10 5 1 9 3 5 3 6 4	12 0 0 1 1

BÀI 2.1.3: *LIỆT KÊ DÃY CON*

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy viế t chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K.

Dữ liệu vào:

Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các số của dãy số A[] và số tự nhiên K, hai số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Kết quả:

Các dãy con thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ:

Ví dụ dưới đây sẽ minh hoạ cho file dayso.in và ketqua.out của bài toán.

Dayso.in	**ketqua.out**
7 50 5 10 15 20 25 30 35	7 20 30 15 35 5 20 25 5 15 30 5 10 35 5 10 15 20 5 10 15 20

BÀI 2.1.4: *TỔ HỢP*

Một tổ hợp chập k của n là một tập con k phần tử của tập n phần tử.

Chẳng hạn tập {1,2,3,4} có các tổ hợp chập 2 là: {1,2}, {1,3, {1,4, {2,3}, {2,4}, {3,4}. Vì trong tập hợp các phần tử không phân biệt thứ tự nên tập {1,2} cũng là tập {2,1} và do đó, ta coi chúng chỉ là một tổ hợp.

Bài toán đặt ra cho chúng ta là hãy xác định tất cả các tổ hợp chập k của tập n phần tử. Để đơn giản ta chỉ xét bài toán tìm các tổ hợp của tập các số nguyên từ 1 đến n. Đối với một tập hữu hạn bất kì, bằng cách đánh số thứ tự của các phần tử, ta cũng đưa được về bài toán đối với tập các số nguyên từ 1 đến n.

Dữ liệu vào:

File input gồm: dòng đầu ghi 2 số nguyên dương n, k cách nhau bởi dấu cách.

Kết quả:

File output ghi mỗi dòng một tổ hợp k của n phần tử.

BÀI 2.1.5: CHỈNH HỢP LẶP

Chỉnh hợp lặp chập k của n là một dãy k thành phần, mỗi thành phần là một phần tử của tập n phần tử, có xét đến thứ tự và không yêu cầu các thành phần khác nhau.

Một ví dụ dễ thấy nhất của chỉnh hợp lặp là các dãy nhị phân. Một dãy nhị phân độ dài m là một chỉnh hợp lặp chập m của tập 2 phần tử {0,1}. Chẳng hạn 101 là một dãy nhị phân độ dài 3. Ngoài ra ta còn có 7 dãy nhị phân độ dài 3 nữa là 000, 001, 010, 011, 100, 110, 111. Vì có xét thứ tự nên dãy 101 và dãy 011 là 2 dãy khác nhau.

Dữ liệu vào:

File input gồm: dòng đầu ghi 2 số nguyên dương n, k cách nhau bởi dấu cách.

Kết quả:

File output ghi mỗi dòng một chỉnh hợp lặp chập k của n phần tử.

BÀI 2.1.6: CHỈNH HỢP KHÔNG LẶP

Khác với chỉnh hợp lặp là các thành phần được phép lặp lại, tức là có thể giống nhau, chỉnh hợp không lặp chập k của tập n phần tử cũng là một dãy k thành phần lấy từ tập n phần tử có xét thứ tự nhưng các thành phần không được phép giống nhau.

Chẳng hạn có n người, một cách chọn ra k người để xếp thành một hàng là một chỉnh hợp không lặp chập k của n.

Một trường hợp đặc biệt của chỉnh hợp không lặp là hoán vị. Hoán vị của một tập n phần tử là một chỉnh hợp không lặp chập n. Nói một cách trực quan thì hoán vị của tập n phần tử là phép thay đổi vị trí của các phần tử (do đó mới gọi là hoán vị).

Dữ liệu vào:

File input gồm: dòng đầu ghi 2 số nguyên dương n, k cách nhau bởi dấu cách.

Kết quả:

File output ghi mỗi dòng một chỉnh hợp không lặp chập k của n phần tử.

BÀI 2.1.7: TÌM 2 SỐ NGUYÊN TỐ

Tìm 2 số nguyên tố thỏa mãn khi biết n: n = p*q.

Dữ liệu vào (nhập từ bàn phím):

n là số nguyên dương.

Kết quả:

In ra màn hình 2 số p, q cách nhau bởi dấu cách.

BÀI 2.1.8: TÌM NGHIỆM PHƯƠNG TRÌNH

Tìm nghiệm nguyên của phương trình: ax+by = c, với a,b,c là các số nguyên.

Dữ liệu vào (nhập từ bàn phím):

Các số nguyên a,b,c.

Kết quả:

In ra màn hình các nghiệm thỏa mãn, mỗi dòng 2 số x,y cách nhau bởi dấu cách.

BÀI 2.1.9: TÌM SỐ LỚN NHẤT TRONG DÃY SỐ CHO TRƯỚC

Cho một dãy số nguyên bất kỳ, hãy tìm ra số lớn nhất.

Dữ liệu vào (nhập từ bàn phím):

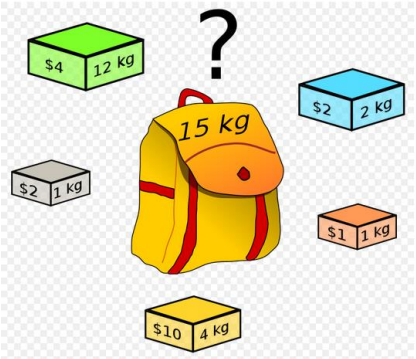
Dãy số nguyên, các số cách nhau bởi dấu cách, kết thúc nhập bởi Enter.

Kết quả:

In ra màn hình số lớn nhất.

BÀI 2.1.10: BÀI TOÁN CÁI TÚI

Một kẻ trộm đột nhập vào một cửa hiệu tìm thấy có n mặt hàng có trọng lượng và giá trị khác nhau, nhưng hắn chỉ mang theo một cái túi có sức chứa về trọng lượng tối đa là M. Vậy kẻ trộm nên bỏ vào ba lô những món nào và số lượng bao nhiêu để đạt giá trị cao nhất trong khả năng mà hắn có thể mang đi được.



Dữ liệu vào:

File input: dòng đầu ghi n, M là số nguyên dương, dòng thứ hai ghi trọng lượng của n mặt hàng w_i, cách nhau bởi dấu cách, dòng thứ 3 ghi giá trị của n mặt hàng c_i.

Kết quả:

File output ghi phương án tối ưu nhất thứ tự các mặt hàng mang đi được.

BÀI 2.1.11: ĐẶT DẤU NGOẶC

Cho số nguyên dương n (n<=10), liệt kê tất cả các cách khác nhau đặt n dấu ngoặc mở và n dấu ngoặc đóng đúng đắn?

VD:

n=3 -> có 5 cách:

1. ((( ))) 2. (( ) ( )) 3. (( ))() 4. ()(( )) 5. ()()()

BÀI 2.1.12: TÌM SỐ NGUYÊN DƯƠNG NHỎ NHẤT

Cho n (n<=10) số nguyên dương a1,a2,...,an (ai<=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 1 lần) thuộc n số trên.

VD: n=4, A= (1,2,3,4) kết quả ra sẽ là 13.

BÀI 2.1.13: BÀI TOÁN 36 SĨ QUAN

Bài toán 36 sĩ quan là bài toán tổ hợp nổi tiếng do nhà toán học Thụy Sĩ Ole đưa ra vào thế kỷ XVIII. Nội dung bài toán như sau: Có 36 sĩ quan thuộc 6 loại quân hàm, từ 6 đơn vị khác nhau. Hãy tìm cách xếp thành đội hình 6 hàng ngang, 6 hàng dọc thế nào cho trên mỗi hàng ngang, hàng dọc chỉ có các sĩ quan không cùng quân hàm và cùng đơn vị.

BÀI 2.1.14: ĐẢO CHỮ CÁI

Bạn phải viết chương trình đưa ra tất cả các từ có thể có phát sinh từ một tập các chữ cái. Ví dụ: Cho từ “abc”, chương tr ình của bạn phải đưa ra được các từ "abc", "acb", "bac",

"bca", "cab" và "cba" (bằng cách khảo sát tất cả các trường hợp khác nhau của tổ hợp ba chữ cái đã cho).

Dữ liệu vào:

Dữ liệu vào được cho trong tệp input.txt chứa một số từ. Dòng đầu tiên là một số tự nhiên cho biết số từ được cho ở dưới. Mỗi dòng tiếp theo chứa một từ. Trong đó, một từ có thể chứa cả chữ cái thường hoặc hoa từ A đ ến Z. Các chữ thường và hoa được coi như là khác nhau. Một chữ cái nào đó có thể xuất hiện nhiều hơn một lần.

Kết quả:

Với mỗi từ đã cho trong file Input.txt, kết quả nhận được ra file Output.txt phải chứa tất cả các từ khác nhau được sinh từ các chữ cái của từ đó. Các từ được sinh ra từ một từ đã cho phải được đưa ra theo thứ tự tăng dần của bảng chữ cái.

Ví dụ:

Input.txt	**Output.txt**
2 abc acba	abc acb bac bca cab cba aabc aacb abac abca acab acba baac baca bcaa caab caba cbaa

BÀI 2.1.15: *TẠO MA TRẬN SỐ*

Cho trước số nguyên dương N bất kỳ. H ãy viết thuật toán và chương trình để tạo lập bảng NxN phần tử nguyên dương theo quy luật được cho trong ví dụ sau:

1 2 3 4 5 6

2 4 6 8 10 12

3 6 9 12 2 4

4 8 12 2 4 6

5 10 2 4 6 8

6 12 4 6 8 10

Thực hiện chương trình đó trên máy với N=12, đưa ra màn hình ma trận kết quả (có dạng như trong ví dụ).

BÀI 2.1.16: *DÃY SỐ NGUYÊN*

Dãy các số tự nhiên được viết ra thành một dãy vô hạn trên đ ường thẳng:

12345678910111213 14..... (1)

Hỏi số ở vị trí thứ 1000 trong dãy trên là số nào?

Bạn hãy viết chương trình để tính toán.

Tổng quát bài toán trên: Chương trình yêu cầu nhập số K từ bàn phím và in ra trên màn hình kết quả là số nằm ở vị trì thứ K trong dãy (1) trên. Yêu cầu chương trình chạy càng nhanh càng tốt.

BÀI 2.1.17: *MÃ NHỊ PHÂN GRAY*

Trong giờ học môn Điện tử số về mã Gray, MĐ chợt nảy sinh ra một bài toán để code. Bài toán rất đơn giản như sau: In ra lần lượt bảng mã gray n-bit.

Mã Gray là mã nhị phân mà hai mã liền kề trong bảng mã chỉ khác nhau một bit. Các giá trị ở nửa sau của bảng mã có sự đối xứng với nửa đầu của bảng mã theo thứ tự ngược lại, ngoại trừ bit cao nhất bị đảo giá trị (bit cao nhất là bit ngoài cùng bên trái). Tính chất đối xứng này vẫn đúng cho các bit thấp hơn trong mỗi nửa, mỗi phần tư,... của bảng mã.

Dữ liệu vào:

Một số nguyên duy nhất n (1<=n<=16).

Kết quả:

Bảng mã gray n-bit theo thứ tự, mỗi mã trên một dòng.

Ví dụ:

INPUT	OUTPUT
2	00 01 11 10

INPUT	OUTPUT
3	000 001 011 010 110 111 101 100

INPUT	OUTPUT
4	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000

BÀI 2.1.18: *LIỆT KÊ HOÁN VỊ*

Liệt kê hoán vị của n phần tử của một tập gồm các số từ 1->n.

Dữ liệu vào:

Dòng duy nhất chứa số n (1<=n<=8)

Kết quả:

Các hoán vị sắp xếp theo thứ tự từ điển tăng dần.

Ví dụ:

Input.txt	**Output.txt**

3	123 132 213 231 312 321

BÀI 2.1.19:

Cho một chuỗi gồm n bít X = (x_1, x_2, .., x_n). Ta gọi K là tổng các bít liền kề của chuỗi X (*ký hiệu là AdjBC*(*X*)) được tính toán như sau:

$$K = \sum_{i=1}^{n-1} x_i x_{i+1};$$

Ví dụ :

AdjBC(011101101) = 3
AdjBC(111101101) = 4
AdjBC(010101010) = 0

Viết chương trình tìm số các chuỗi bít có độ dài N có tổng các bít liền kề đúng bằng K. Ví dụ với N = 5, ta có 6 chuỗi có tổng các bít liền kề là 2 như sau:

11100, 01110, 00111, 10111, 11101, 11011.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số test của bài toán;
- T dòng kế tiếp, mỗi dòng ghi lại một test. Mỗi test ghi lại cặp ba số i, N, K. Trong đó, i là số thứ tự của test, N là độ dài chuỗi bít, K là tổng các bít liền kề. Các số được ghi cách nhau một vài khoảng trống.

Output:

- Ghi lại số hiệu test và số các bít liền kề thỏa mãn yêu cầu bài toán trên một dòng. Hai số được viết cách nhau một vào khoảng trống.

InputB:

```
10
1 5 2
2 20 8
3 30 17
4 40 24
5 50 37
```

OutputB:

```
1 6
2 63426
3 1861225
4 1682122501
5 44874764
6 160916
```

6	60	52
7	70	59
8	80	73
9	90	84
10	100	90

7	22937308
8	99167
9	15476
10	23076518

1.6. Kỹ thuật sinh kế tiếp

BÀI 2.2.1: LIỆT KÊ DÃY NHỊ PHÂN CÓ ĐỘ DÀI CHO TRƯỚC

Sinh các dãy nhị phân có độ dài n.

Dữ liệu vào:

Số nguyên duy nhất n (1<=n<=9)

Kết quả:

Mỗi dòng một dãy nhị phân. Các dãy nhị phân phải được liệt kê theo thứ tự từ điển.

Ví dụ:

Input.txt	**Output.txt**
2	00 01 10 11

Gợi ý:

Ta dễ dàng xác định được cấu hình đầu tiên là 0000….0 (độ dài n) và dãy cuối cùng là 1111…1. Để xác định được dãy nhị phân đứng sau ta lấy dãy nhị phân hiện tại + 1 (bit 1). Ví dụ: dãy nhị phân hiện tại 000 —> dãy kế tiếp 000+1=001.

Xây dựng thuật toán sinh kế tiếp:

- Khởi tạo dãy nhị phân ban đầu là 000…0

- Duyệt dãy nhị phân từ cuối, nếu phần tử cuối là bit 0 thì thay nó bằng bit 1.Còn các bit khác được duyệt qua được gán =1. (sinh kế tiếp)

```
while (chưa về đầu dãy nhị phân) do
i=n;
while(s[i]==1) do
begin
s[i]=0;
i-
end
s[i]=1;
end;
```

BÀI 2.2.2: LIỆT KÊ CHỈNH HỢP LẶP CHẬP K CỦA N PHẦN TỬ

Cho n phần tử từ 1,..,n. Hãy liệt kê các chỉnh hợp lặp chập k của n phần tử trên.

Dữ liệu vào:

Số nguyên k, n (1<=k, n<=9)

Kết quả:

Mỗi dòng một dãy các chỉnh hợp lặp chập k của n phần tử 1,…,n.

BÀI 2.2.3: HOÁN VỊ CỦA N PHẦN TỬ

Xem bài 2.1.18.

BÀI 2.2.4: TỔ HỢP CHẬP K CỦA N PHẦN TỬ

Xem bài 2.1.4.

BÀI 2.2.5: LIỆT KÊ DÃY NHỊ PHÂN KHÔNG CÓ HAI SỐ 0 LIÊN TIẾP

Hãy liệt kê các dãy nhị phân có độ dài n sao cho dãy tạo ra không có 2 số 0 liên tiếp nhau.

Dữ liệu vào:

Số nguyên duy nhất n (1<=n<=9)

Kết quả:

Mỗi dòng một dãy nhị phân. Các dãy nhị phân phải được liệt kê theo thứ tự từ điển.

Ví dụ:

Input.txt	**Output.txt**
2	01 10 11

BÀI 2.2.6: *SINH TỔNG CON KẾ TIẾP*

Cho n là số nguyên dương. Một cách phân chia số n là biểu diễn n thành tổng các số tự nhiên không lớn hơn n. Liệt kê tất cả các tổng con có thể có của n, thứ tự đơn giản nhất là thứ tự từ điển (sử dụng phương pháp sinh tổ hợp để sinh ra cấu hình tiếp theo không dùng quay lui).

Dữ liệu vào:

Số nguyên duy nhất n (1<=n<=9)

Kết quả:

Mỗi dòng một dãy thỏa mãn yêu cầu. Các dãy nhị phân phải được liệt kê theo thứ tự từ điển.

Ví dụ:

Input.txt	**Output.txt**
7	7 6 1 5 2 5 1 1 4 3 4 2 1 4 1 1 1 3 3 1

	3 2 2 3 2 1 1 3 1 1 1 1 2 2 2 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1

BÀI 2.2.7:

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy sử dụng *thuật toán sinh* viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các số của dãy số A[] và số tự nhiên K, hai số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Các dãy con thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ dưới đây sẽ minh hoạ cho file dayso.in và ketqua.out của bài toán.

Dayso.in	**Ketqua.out**
5 50 5 10 15 20 25	3 10 15 25 5 20 25 5 10 15 20

BÀI 2.2.8:

Cho dãy gồm *N* số nguyên phân biệt A[] = {a_1, a_2, .., a_N } và số tự nhiên *K* ($K \le N \le 100$). Hãy sử dụng *thuật toán sinh* viết chương trình liệt kê tất cả các dãy con *K* phần tử tăng dần tự nhiên của dãy số A[]. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, K. Hai số được viết cách nhau một vài khoảng trống;
- Những dòng kế tiếp ghi lại N số nguyên của dãy số A[], hai số khác nhau được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử tăng dần của dãy số A[] tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số các dãy con K phần tử tăng dần của dãy số A[] tìm được;
- M dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in	**ketqua.out**
5 3 2 5 15 10 20	7 2 5 15 2 5 10 2 5 20 2 15 20 2 10 20 5 15 20 5 10 20

BÀI 2.2.9:

Cho dãy gồm *N* số nguyên phân biệt A[] = {a_1, a_2, .., a_N } và số tự nhiên *K* ($K \le N \le 100$). Hãy sử dụng *thuật toán sinh* viết chương trình liệt kê tất cả các dãy con *K* phần tử tăng dần tự nhiên của dãy số A[]. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, K. Hai số được viết cách nhau một vài khoảng trống;
- Những dòng kế tiếp ghi lại N số nguyên của dãy số A[], hai số khác nhau được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử tăng dần của dãy số A[] tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số các dãy con K phần tử tăng dần của dãy số A[] tìm được;
- M dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in					**ketqua.out**		
5	3				7		
2	5	15	10	20	2	5	15
					2	5	10
					2	5	20
					2	15	20
					2	10	20
					5	15	20
					5	10	20

BÀI 2.2.10:

Cho ma trận vuông C = (c_{ij}) cấp N (*$1 \le i, j \le N \le 100$*) gồm N^2 số tự nhiên (*các số không nhất thiết phải khác nhau*) ghi lại trong file matran.in theo khuôn dạng sau :

- Dòng đầu tiên ghi lại số tự nhiên N là cấp của ma trận vuông C;
- N dòng kế tiếp ghi lại ma trận vuông C = (c_{ij}). Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy sử dụng *thuật toán sinh* thích hợp viết chương trình lấy trên mỗi hàng, mỗi cột duy nhất một phần tử của ma trận C sao cho tổng các phần tử này là nhỏ nhất. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại tổng giá trị nhỏ nhất của N phần tử tìm được;
- N dòng kế tiếp, mỗi dòng ghi lại ba số i, j, c_{ij} tương ứng với chỉ số hàng, chỉ số cột và giá trị phần tử tương ứng của ma trận. Ba số được viết cách nhau một vài khoảng trống.

Ví dụ về file **matran.in** và **ketqua.out**:

matran.in						**ketqua.out**		
6						82		
10	64	57	29	18	15	1	1	10
34	20	19	30	16	**12**	2	6	12
57	49	40	**16**	11	19	3	4	16
29	21	46	26	**21**	18	4	5	21
28	16	**11**	21	21	37	5	3	11
15	**12**	15	48	37	30	6	2	12

BÀI 2.2.11:

Cho dãy gồm *N* số nguyên phân biệt A[] = {a_1, a_2, .., a_N } và số tự nhiên *K* ($K \le N \le 100$). Hãy sử dụng *thuật toán sinh* viết chương trình liệt kê tất cả các dãy con *K* phần tử giảm dần tự nhiên của dãy số A[]. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, K. Hai số được viết cách nhau một vài khoảng trống;
- Những dòng kế tiếp ghi lại N số nguyên của dãy số A[], hai số khác nhau được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử tăng dần của dãy số A[] tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số các dãy con K phần tử tăng dần của dãy số A[] tìm được;
- M dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in	**ketqua.out**
5 3 2 5 15 10 20	7 2 5 15 2 5 10 2 5 20 2 15 20 2 10 20 5 15 20 5 10 20

BÀI 2.2.12:

Hãy sử dụng thuật toán *sinh* (*quay lui, nhánh cận, qui hoạch động*) viết chương trình Viết chương trình tìm $X = (x_1, x_2,..,x_n)$ và giá trị $f(x_1, x_2,.., x_n) = \sum_{i=1}^{n} c_i x_i$ đạt giá trị lớn nhất. Trong đó, $X = (x_1, x_2,.., x_n) \in D = \left\{ \sum_{i=1}^{n} a_i x_i \le b; x_i \in \{0,1\} \right\}$, c_i, a_i, b là các số nguyên dương, $n \le 100$

Dữ liệu vào n, c_j, a_j, b được cho trong file *data.in* theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên n và b. Hai số được ghi cách nhau bởi một vài ký tự trống;
- n dòng kế tiếp ghi lại trên mỗi dòng một cặp số c_i , a_i ($i=1, 2, .., n$). Hai số được ghi cách nhau bởi một vài ký tự trống;

Giá trị tối ưu $f(x_1,x_2,..,x_n)$ và phương án tối ưu $X = (x_1, x_2,..,x_n)$ tìm được ghi lại trong file *ketqua.out* theo khuôn dạng sau:

- Dòng đầu tiên ghi lại giá trị tối ưu $f(x_1,x_2,..,x_n)$;
- Dòng kế tiếp ghi lại phương án tối ưu $X = (x_1, x_2,..,x_n)$. Hai phần tử khác nhau của X được ghi cách nhau bởi một vài khoảng trống.

Ví dụ dưới đây sẽ minh họa cho file data.in và ketqua.out của bài toán:

Data.in	***Ketqua.out***
4 10 5 5 1 3 9 6 3 4	12 0 0 1 1

BÀI 2.2.13:

Cho dãy gồm n số tự nhiên phân biệt $a_1, a_2, .., a_n$ và số tự nhiên B. Hãy sử dụng thuật toán *sinh* (*quay lui, nhánh cận, qui hoạch động*) liệt kê tất cả các phần tử của tập $D = \left\{(x_1, x_2, \cdots, x_n): \sum_{i=1}^{n} a_i x_i = B, \quad x_i \in \{0,1\}, i = 1,2,..., n\right\}$;

Dữ liệu vào cho bởi file data.in theo khuôn dạng như sau:

- Dòng đầu tiên ghi lại hai số tự nhiên n và B. Hai số được viết cách nhau bởi một vài khoảng trống.
- Dòng kế tiếp ghi lại n số nguyên dương $a_1, a_2,..,a_n$. Hai số khác nhau được viết cách nhau bởi một vài kí tự trống.

Kết quả ra ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên k là số phần tử của tập D.
- k dòng tiếp theo mỗi dòng ghi lại một vector nhị phân $x = (x_1, x_2 ,.., x_n)$ là phần tử của D. Hai thành phần khác nhau của vector x được viết cách nhau bởi một vài khoảng trống.

Ví dụ với $n = 7, B = 25, \{ a_1, a_2, a_3, a_4, a_5, a_6, a_7\} = \{5, 10, 15, 20, 25, 30, 35\}$ trong file data.in sẽ cho ta 3 phần tử của tập D tương ứng với 3 vector nhị phân độ dài n trong file ketqua.out dưới đây:

Data.in

```
7    25
5    10    15    20    25
```

Ketqua.Out

```
3
0  0  0  0  1  0  0
1  0  0  1  0  0  0
0  1  1  0  0  0  0
```

BÀI 2.2.14:

Một người quản lý có n công việc cần thực hiện cùng một lúc. Biết rằng có n công nhân, mỗi công nhân đều có thể thực hiện được tất cả các công việc nhưng với thời gian khác nhau. Thời gian để công nhân thứ i thực hiện công việc j là $C_{i,j}$ (tính theo giờ). Hãy sử dụng thuật toán *sinh* (*quay lui, nhánh cận, qui hoạch động*) viết chương trình tìm cách bố trí n công việc cho n công nhân sao cho tổng thời gian thực hiện là nhỏ nhất.

Dữ liệu vào được cho bởi file: VIEC.INP trong đó:

- Dòng thứ nhất ghi số N ;

- N dòng tiếp theo ghi các giá trị của ma trận thời gian C. Hai phần tử khác nhau được viết cách nhau một vài khoảng trống.

Kết quả tìm được lưu vào file KETQUA.OUT trong đó:

- Dòng thứ nhất ghi giá trị tổng thơi gian nhỏ nhất có thể đạt được

- Dòng thứ hai ghi cách bố trí việc cho từng ông thợ.

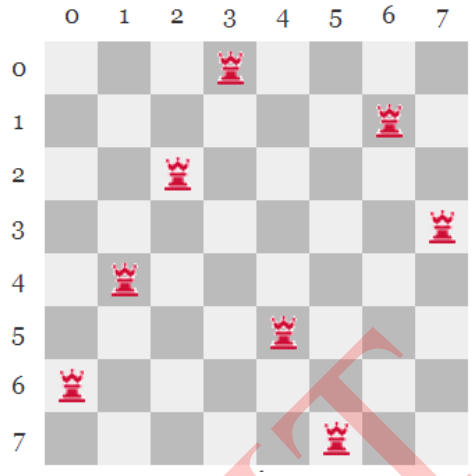
Ví dụ: *File VIEC.INP và KETQUA.OUT*

VIEC.INP	**KETQUA.OUT**
6 10 64 57 29 18 15 34 20 19 71 16 12 57 49 40 16 11 19 29 21 46 26 21 18 28 16 11 21 21 37 15 12 15 48 37 30	82 1 6 5 3 4 2

1.7. Kỹ thuật quay lui

BÀI 2.3.1: BÀI TOÁN XẾP HẬU (THAM KHẢO)

Có một bàn cờ kích thước n x n (n ≥ 4). Yêu cầu: tìm tất cả các cách để đặt n con hậu trên bàn cờ sao cho không có con nào khống chế lẫn nhau. Biết rằng, một con hậu có thể khống chế tất cả các quân cờ trên hàng ngay, cột dọc và hai đường chéo đi qua vị trí của nó. Chẳng hạn ta có một cách đặt sau đối với các con hậu:



Dữ liệu vào:

File input gồm: số nguyên dương n.

Kết quả:

File output: dòng đầu tiên ghi số phương án thỏa mãn m. Các dòng tiếp theo ghi m ma trận nxn gồm các số 0 hoặc 1 với 1 là vị trí đặt của quân hậu, các ma trận liền nhau cách nhau một dòng trống.

Gợi ý:

Ta có có nhận xét: mỗi con hậu phải ở trên một hàng và một cột. Do đó ta coi con hậu thứ i ở hàng i và nếu biết x[i] là cột đặt con hậu thứ i thì ta suy ra được lời giải. Vậy nghiệm của bài toán có thể coi là một vector x gồm n thành phần với ý nghĩa:

1. Con hậu thứ i được đặt ở hàng i và cột x[i].

2. x[i] lấy giá trị trong tập {1,2...n}

3. Ràng buộc: các giá trị x[i] khác nhau từng đôi một và không có 2 con hậu ở trên cùng một đường chéo.

BÀI 2.3.2: BÀI TOÁN MÃ ĐI TUẦN (THAM KHẢO)

Cho bàn cờ vua có n x n ô . Một con mã được phép đi theo luật cờ vua, đầu tiên được đặt vào ô có tọa độ (x_0,y_0). Yêu cầu : Hãy chỉ ra các hành trình nếu có của con mã sao cho con mã đi qua tất cả cá ô của bàn cờ,mỗi ô đi qua đúng 1 lần.

Dữ liệu vào:

File input gồm: số nguyên dương n.

Kết quả:

File output: các cặp tọa độ (x,y) cách nhau bởi dấu cách chỉ ra hành trình của con mã, bắt đầu với (x_0,y_0).

BÀI 2.3.3: TÌM ĐƯỜNG ĐI TRONG MÊ CUNG

Mê cung là một khu vực hình chữ nhật được chia là M*N phòng. Tại mỗi phòng có thể nhốt một quái vật. Từ một phòng ta có thể đi đến 4 phòng xung quang nếu chúng không có quái vật. Giả sử bạn đang lạc trong mê cung tại ô (x0,y0). Hãy tìm đường đi thoát khỏi mê cung ra một phòng trên cạnh của mê cung một cách nhanh nhất (đi qua ít đỉnh nhất).

Dữ liệu vào:

Dòng đầu là 2 số M,N.M dòng tiếp theo mô tả ma trận A trong đó A[x,y]=1 nếu tại phòng (x,y) có quái vật, ngược lại thì A[x,y]=0.

Hạn chế: 4<=M,N<=100.

Kết quả:

Dòng đầu tiên là số K mô tả số các phòng đã đi qua. Nếu không tìm được đường ra thì K=0. Các dòng tiếp theo là toạ độ các phòng đã đi qua.

BÀI 2.3.4: BÀI TOÁN NGƯỜI BÁN HÀNG

Có một người giao hàng cần đi giao hàng tại n thành phố. Anh ta xuất phát từ một thành phố nào đó, đi qua các thành phố khác để giao hàng và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, và khoảng cách từ một thành phố đến các thành phố khác đã được biết trước. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất.

Gợi ý:

Bài toán người bán hàng có thể được mô hình hoá như một đồ thị vô hướng có trọng số, trong đó mỗi thành phố là một đỉnh của đồ thị còn đường đi giữa các thành phố là mỗi cách. Khoảng cách giữa hai thành phố là độ dài cạnh. Đây là vấn đề cực tiểu hoá với điểm đầu và điểm cuối là cùng một đỉnh sau khi thăm hết các đỉnh còn lại đúng một lần. Mô hình này thường là một đồ thị đầy đủ (giữa mỗi cặp đỉnh đều có cạnh). Nếu không có đường giữa hai thành phố thì có thể thêm một cạnh với độ dài đủ lớn vào đồ thị mà không ảnh hưởng đến kết quả tối ưu sau cùng.

BÀI 2.3.5: LIỆT KÊ DÃY NHỊ PHÂN KHÔNG CÓ HAI SỐ 0 LIÊN TIẾP

Xem bài 2.2.4.

BÀI 2.3.6: TAM GIÁC SỐ

Hình sau mô tả một tam giác số có số hàng N=5:

				7				
			3		**8**			
		8		**1**		**0**		
	2		**7**		**4**		**4**	
4		**5**		**2**		**6**		**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 cộng các số trên đường đi lại ta được một tổng.

Ví dụ: đường đi 7 8 1 4 6 có tổng là S=26, đường đi 7 3 1 7 5 có tổng là S=23

Trong hình trên, tổng Smax=30 theo đường đi 7 3 8 7 5 là tổng lớn nhất trong tất cả các tổng.

Nhiệm vụ của bạn và viết chương trình nhận dữ liệu vào là một tam giác số chứa trong text file INPUT.TXT và đưa ra kết quả là giá trị của tổng Smax trên màn hình.

File INPUT.TXT có dạng như sau:

Dòng thứ 1: có duy nhất 1 số N là số hàng của tam giác số (0<N<100).

N dòng tiếp theo, từ dòng thứ 2 đến dòng thứ N+1: dòng thứ i có (i-1) số cách nhau bởi dấu trống (space).

Ví dụ: với nội dung của file INPUT.TXT là

5

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

thì kết quả chạy chương trình sẽ là: Smax=30.

BÀI 2.3.7: Ô CHỮ

Trò chơi ô chữ thông dụng 30 năm trước của trẻ em gồm một khung ô chữ kích thước 5x5 chứa 24 hình vương nhỏ kích thước như nhau. Trên mặt mỗi hình vuông nhỏ có in một chữ cái trong bảng chữ cái. Vì chỉ có 24 hình vuông trong ô chữ nên trong ô chữ còn thừa ra một ô trống, có kích thước đúng bằng kích thước các hình vuông. Một hình vuông có thể đẩy trượt vào ô trống đó nếu nó nằm ngay sát bên trái, bên phải, bên trên hay bên dưới ô trống. Mục tiêu của trò chơi là trượt các h ình vuông vào ô trống sao cho cuối cùng các chữ cái trong ô chữ được xếp theo đúng thứ tự của chúng trong bảng chữ cái. Hình sau đây minh hoạ một ô chữ với cấu hình ban đầu và cấu hình của nó sau 6 nước đi sau:

1.Trượt hình vuông phía trên ô trống.

2.Trượt hình vuông bên phải ô trống.

3.Trượt hình vuông bên phải ô trống.

4.Trượt hình vuông phía dưới ô trống.

5.Trượt hình vuông phía dưới ô trống.

6.Trượt hình vuông bên trái ô trống.

TT	RR	GG	SS	JJ
XX	OO	KK	LL	II
MM	DD	VV	BB	NN
WW	PP		AA	EE
UU	QQ	HH	CC	FF

Cấu hình của ô chữ sau 6 nước.

T	R	G	S	J
X	D	O	K	I
M		V	L	N
W	P	A	B	E
U	Q	H	C	F

Cấu hình ban đầu của ô chữ

Bạn hãy viết một chương trình của bạn chứa cấu hình ban đầu của ô chữ cùng các nước đi để vẽ ra ô chữ kết quả.

Input

Đầu vào của chương trình của bạn chứa cấu hình ban đầu của một ô chữ và một dẫy các nước đi trong ô chữ đó .

Năm dòng đầu tiên mô tả cấu hình ban đầu của ô chữ, mỗi dòng tương ứng với một hàng của ô chữ và chứa đú ng 5 ký tự tương ứng với 5 hình vuông của ô chữ trên hàng đó. Ô trống đ ược diễn tả bằng một dấu cách.

Các dòng tiếp theo sau là dẫy các nước đi. Dãy các nước đi được ghi bằng dãy các chữ A,B,R và L để thể hiện hình vuông nào được trượt vào ô trống. A thể hiện hình vuông phía trên ô trống được trượt vào ô trống, tương ứng: B-phía dưới, R-bên phải, L-bên trái. Có thể có những nước đi không hợp cách, ngay cả khi nó được biểu thị bằng những chữ cái trên. Nếu xuất hiện một nước đi không hợp cách thì ô chữ coi như không có cấu hình kết quả. Dãy các nước đi có thể chiếm một số dòng, nhưng nó sẽ được xem là kết thúc ngay khi gặp một số 0.

Out put

Nếu ô chữ không có cấu hình kết quả thì thông báo 'This puzzle has no final configuration.'; ngược lại thì hiển thị cấu hình ô chữ kết quả. Định dạng mỗi dòng kết quả bằng cách đặt một dấu cách vào giữa hai kí tự kế tiếp nhau. Ô trống cũng được xử lý như vậy. Ví dụ nếu ô trống nằm bên trong hàng thì nó được xuất hiện dưới dạng 3 dấu cách: một để ngăn cách nó với kí tự bên trái, một để thể hiện chính ô trống đó , còn một để ngăn cách nó với kí tự bên phải.

Chú ý: Input mẫu đầu tiên tương ứng với ô chữ được minh hoạ trong ví dụ trên.

Sample Input 1

```
TRGSJ
XDOKI
M VLN
WPABE
UQHCF
ARRBBL0
```

Sample Output 1

```
T  R  G S J
```

X O K L I

M D V B N

W P A E

U Q H C F

Sample Input 2

AB C DE

F G H I J

KLMNO

PQRS

TUVWX

AAA

LLLL0

Sample Output 2

A B C D

F G H I E

K L M N J

P Q R S O

T U V W X

Sample Input 3

ABCDE

FGHIJ

KLMNO

PQRS

TUVWX

AAAAABBRRRLL0

Sample Output 3

This puzzle has no final configuration.

BÀI 2.3.8: DI CHUYỂN TRÁI PHẢI

Một robot chỉ có thể di chuyển sang phải hoặc sang trái. Dãy di chuyển của nó được kí hiệu là một chuỗi, nhưng do sơ suất của người ghi chép, nên một số bước đi không ghi lại được. Nhiệm vụ của bạn là xác định xem robot có thể đi cách xa nhất vị trí ban đầu là ban nhiêu?

Input

Một dòng duy nhất chứa chuỗi di chuyển, mỗi kí tự có thể là:

- 'L' nếu di chuyển sang trái
- 'R' nếu di chuyển sang phải
- ' ?' nếu không xác định.

Output

Khoảng cách xa nhất đến vị trí ban đầu.

Ví dụ:

Input:
LLLRLRRR

Output:

3

Input:
R???L

Output:

4

BÀI 2.3.9: LIỆT KÊ DÃY NHỊ PHÂN CÓ ĐỘ DÀI CHO TRƯỚC

Tham khảo bài 2.3.1

BÀI 2.3.10:

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy sử dụng *thuật toán quay lui* viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các số của dãy số A[] và số tự nhiên K, hai số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Các dãy con thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Dayso.in	**Ketqua.out**
5 50 5 10 15 20 25	3 10 15 25 5 20 25 5 10 15 20

BÀI 2.3.11:

Cho ma trận vuông C = (c_{ij}) cấp N ($1 \le i, j \le N \le 100$) gồm N^2 số tự nhiên (*các số không nhất thiết phải khác nhau*) ghi lại trong file matran.in theo khuôn dạng sau :

- Dòng đầu tiên ghi lại số tự nhiên N là cấp của ma trận vuông C;
- N dòng kế tiếp ghi lại ma trận vuông C = (c_{ij}). Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy sử dụng *thuật toán quay lui* viết chương trình lấy trên mỗi hàng, mỗi cột duy nhất một phần tử của ma trận C sao cho tổng các phần tử này là nhỏ nhất. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại tổng giá trị nhỏ nhất của N phần tử tìm được;
- N dòng kế tiếp, mỗi dòng ghi lại ba số i, j, c_{ij} tương ứng với chỉ số hàng, chỉ số cột và giá trị phần tử tương ứng của ma trận. Ba số được viết cách nhau một vài khoảng trống.

Ví dụ về file **matran.in** và **ketqua.out**:

matran.in						**ketqua.out**		
6						82		
10	64	57	29	18	15	1	1	10
34	20	19	30	16	**12**	2	6	12
57	49	40	**16**	11	19	3	4	16
29	21	46	26	**21**	18	4	5	21
28	16	**11**	21	21	37	5	3	11
15	**12**	15	48	37	30	6	2	12

BÀI 2.3.12:

Cho dãy gồm *N* số nguyên phân biệt A[] = {a_1, a_2, .., a_N } và số tự nhiên *K* ($K \le N \le 100$). Hãy sử dụng *thuật toán quay lui* viết chương trình liệt kê tất cả các dãy con *K* phần tử giảm dần tự nhiên của dãy số A[]. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, K. Hai số được viết cách nhau một vài khoảng trống;
- Những dòng kế tiếp ghi lại N số nguyên của dãy số A[], hai số khác nhau được viết cách nhau một vài khoảng trống.

Các dãy con K phần tử tăng dần của dãy số A[] tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số các dãy con K phần tử tăng dần của dãy số A[] tìm được;
- M dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in					**ketqua.out**		
5	3				7		
2	5	15	10	20	2	5	15
					2	5	10
					2	5	20
					2	15	20
					2	10	20
					5	15	20
					5	10	20

1.8. Kỹ thuật nhánh cận

BÀI 2.4.1: *DÃY ABC*

Cho trước một số nguyên dương N (N ≤ 100), hãy tìm một xâu chỉ gồm các ký tự A, B, C thoả mãn 3 điều kiện:

- Có độ dài N

- Hai đoạn con bất kỳ liền nhau đều khác nhau (đoạn con là một dãy ký tự liên tiếp của xâu)

- Có ít ký tự C nhất.

Cách giải:

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

Tại bước thử chọn X[i], nếu ta đã có T[i] ký tự "C" trong đoạn đã chọn từ X[1] đến X[i], thì cho dù các bước đệ quy tiếp sau làm tốt như thế nào chăng nữa, số ký tự "C" sẽ phải chọn thêm bao giờ cũng ≥ (n - i) div 4. Tức là nếu theo phương án chọn X[i] như thế này thì số ký tự "C" trong dãy kết quả (khi chọn đến X[n]) cho dù có làm tốt đến đâu cũng ≥ T[i] + (n - i) div 4. Ta dùng con số này để đánh giá nhánh cận, nếu nó nhiều hơn số ký tự "C" trong BestConfig thì chắc chắn có làm tiếp cũng chỉ được một cấu hình tồi tệ hơn, ta bỏ qua ngay cách chọn này và thử phương án khác.

Input: file văn bản ABC.INP chứa số nguyên dương n ≤ 100

Output: file văn bản ABC.OUT ghi xâu tìm được

Ví dụ:

ABC.INP	**ABC.OUT**
10	ABACABCBAB "C" Letter Count : 2

BÀI 2.4.2: BÀI TOÁN NGƯỜI BÁN HÀNG

Xem bài 2.3.4.

BÀI 2.4.3: BÀI TOÁN TÌM ĐƯỜNG TRONG MÊ CUNG

Xem bài 2.3.3.

BÀI 2.4.4: BÀI TOÁN CÁI TÚI

Xem bài 2.1.10.

BÀI 2.4.5: BÀI TOÁN MÃ ĐI TUẦN

Xem bài 2.3.2.

BÀI 2.4.6: BÀI TOÁN XẾP HẬU

Xem bài 2.3.1.

BÀI 2.4.7:

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy sử dụng *thuật toán quay lui nhánh cận* viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các số của dãy số A[] và số tự nhiên K, hai số được viết cách nhau bởi một vài khoảng trống;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trống.

Các dãy con thoả mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trống.

Dayso.in	**Ketqua.out**

5 50 5 10 15 20 25	3 10 15 25 5 20 25 5 10 15 20

BÀI 2.4.8:

Cho ma trận vuông C = (c_{ij}) cấp N ($1 \le i, j \le N \le 100$) gồm N^2 số tự nhiên (*các số không nhất thiết phải khác nhau*) ghi lại trong file matran.in theo khuôn dạng sau :

- Dòng đầu tiên ghi lại số tự nhiên N là cấp của ma trận vuông C;
- N dòng kế tiếp ghi lại ma trận vuông C = (c_{ij}). Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy sử dụng *thuật toán quay lui nhánh cận* viết chương trình lấy trên mỗi hàng, mỗi cột duy nhất một phần tử của ma trận C sao cho tổng các phần tử này là nhỏ nhất. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại tổng giá trị nhỏ nhất của N phần tử tìm được;
- N dòng kế tiếp, mỗi dòng ghi lại ba số i, j, c_{ij} tương ứng với chỉ số hàng, chỉ số cột và giá trị phần tử tương ứng của ma trận. Ba số được viết cách nhau một vài khoảng trống.

Ví dụ về file **matran.in** và **ketqua.out**:

matran.in	**ketqua.out**
6 **10** 64 57 29 18 15 34 20 19 30 16 **12** 57 49 40 **16** 11 19 29 21 46 26 **21** 18 28 16 **11** 21 21 37 15 **12** 15 48 37 30	82 1 1 10 2 6 12 3 4 16 4 5 21 5 3 11 6 2 12

1.9. Kỹ thuật qui hoạch động

BÀI 2.5.1: XẾP HÀNG

Hàng ngày khi lấy sữa, N con bò của bác John ($1 \le N \le 50000$) luôn xếp hàng theo thứ tự không đổi. Một hôm bác John quyết định tổ chức một trò chơi cho một số con bò. Để đơn giản, bác John sẽ chọn ra một đoạn liên tiếp các con bò để tham dự trò chơi. Tuy nhiên để trò chơi diễn ra vui vẻ, các con bò phải không quá chênh lệch về chiều cao.

Bác John đã chuẩn bị một danh sách gồm Q ($1 \le Q \le 200000$) đoạn các con bò và chiều cao của chúng (trong phạm vi [1, 1000000]). Với mỗi đoạn, bác John muốn xác định chênh lệch chiều cao giữa con bò thấp nhất và cao nhất. Bạn hãy giúp bác John thực hiện công việc này.

Dữ liệu

- Dòng đầu tiên chứa 2 số nguyên N và Q.
- Dòng thứ i trong số N dòng sau chứa 1 số nguyên duy nhất, là độ cao của con bò thứ i.
- Dòng thứ i trong số Q trong tiếp theo chứa 2 số nguyên A, B ($1 \le A \le B \le N$), cho biết đoạn các con bò từ A đến B.

Kết quả

Gồm Q dòng, mỗi dòng chứa 1 số nguyên, là chênh lệch chiều cao giữa con bò thấp nhất và cao nhất thuộc đoạn tương ứng.

Ví dụ:

Input.txt	**Output.txt**
6 3 1 7 3 4 2 5 1 5 4 6 2 2	6 3 0

BÀI 2.5.2: *ĐOẠN TĂNG*

Cho dãy số nguyên A = (a1, a2, …, an). Hãy tìm một đoạn dài nhất gồm các phần tử liên tiếp trong dãy A có thứ tự không giảm.

Quy ước: Đoạn chỉ gồm đúng 1 phần tử trong A cũng được coi là có thứ tự không giảm.

Dữ liệu:

Dòng 1 chứa số nguyên dương $n \le 10^5$

Dòng 2 chứa n số nguyên $a_1, a_2, ..., a_n$ ("i: $|a_i| \le 10^9$) cách nhau ít nhất một dấu cách

Kết quả:một số nguyên duy nhất là số phần tử trong đoạn tìm được

Ví dụ:

INPUT	OUTPUT
11 88 99 11 22 22 33 11 66 33 44 77	4

BÀI 2.5.3: *ĐỘ ĐO*

Hai xâu ký tự được gọi là đảo của nhau nếu ta có thể hoán vị các ký tự của một xâu để được xâu còn lại. Ví dụ: xâu "occurs" là đảo của xâu "succor", tuy nhiên xâu "dear" không phải là đảo của xâu "dared" (vì chữ 'd' xuất hiện 2 lần trong "dared" nhưng chỉ xuất hiện "dear" trong 1 lần).

Độ đo giữa hai xâu ký tự là số ký tự ít nhất cần phải xóa (trên cả hai xâu) để hai xâu còn lại đảo của nhau. Ví dụ độ đo giữa hai xâu "sleep" và "leap" là 3, độ đo giữa hai xâu "dog" và "cat" là 6.

Yêu cầu:

Hãy tính độ đo giữa hai xâu cho trước.

Dữ liệu:

Gồm hai dòng, mỗi dòng chứa một xâu ký tự chỉ gồm các chữ cái tiếng Anh thường, mỗi dòng có không quá 1 triệu ký tự.

Kết quả: một số nguyên duy nhất là độ đo giữa hai xâu trong file dữ liệu.

Ví dụ

INPUT	OUTPUT
Begin end	4

BÀI 2.5.4: DÃY CON TĂNG DẦN TỰ NHIÊN BẬC K

Cho dãy gồm N số phân biệt $A_N = \{a_1, a_2, .., a_N\}$ và số tự nhiên K ($K<=N<=100$). Ta gọi một *dãy con tăng dần bậc K* của dãy số A_N là một dãy các số gồm K phần tử trong dãy đó thỏa mãn tính chất tăng dần. Bài toán được đặt ra là hãy tìm số các *dãy con tăng dần bậc K* của dãy số A_N.

Input:

Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được xây dựng theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số N và K tương ứng với số phần tử của dãy số và bậc của dãy con.
- Dòng kế tiếp ghi lại N số của dãy số A_N, các số trong dãy không lớn hơn 100.

Output:

Với mỗi bộ test, in ra màn hình số các dãy con tăng dần tự nhiên bậc K của dãy số A_N.

Ví dụ:

Input:	**Output:**
2 5 3 2 5 15 10 20 5 3 2 20 10 15 5	7 1

III. Lập trình dựa vào ngăn xếp, hàng đợi

1.10.Kỹ thuật xử lý trên ngăn xếp

BÀI 3.1.1: *THÁP HÀ NỘI (THAM KHẢO)*

Bài toán Tháp Hà Nội được mô tả như sau: cho 3 cột được đánh số lần lượt là 1, 2 và 3. Có n đĩa được sắp theo thứ tự đĩa nhỏ ở bên trên đĩa lớn. Hãy liệt kê các bước thực hiện để chuyển tất cả các đĩa từ cột 1 sang cột 2. Quy luật di chuyển như sau:

1. Mỗi bước chỉ di chuyển 1 đĩa từ cột này sang cột khác.
2. Đĩa có bán kính nhỏ luôn sắp trên đĩa có bán kính lớn.

1 2 3 1 2 3

Yêu cầu:

Viết chương trình nhập vào số đĩa n, thực hiện các bước di chuyển các đĩa, mỗi bước di chuyển cho biết cột nguồn (cột lấy đĩa) và cột đích (cột đặt đĩa). Giải thuật di chuyển không đệ quy, dùng stack để chứa thông tin tạm thời trong quá trình di chuyển.

Sinh viên cài đặt stack dùng danh sách liên kết, mỗi node phần info chứa 3 thông tin {số đĩa di chuyển, cột nguồn, cột đích}.

Hướng dẫn:

Như chúng ta biết bài toán tháp Hanoi thường được giải bằng phương pháp đệ quy. Tuy nhiên có thể giải bằng cách dùng stack để khử đệ quy. Để thực hiện việc lưu trữ tạm trong quá trình di chuyển chúng ta dùng một stack. Trong đó mỗi phần tử của stack này chứa các thông tin gồm: số đĩa di chuyển (N), cột nguồn bắt đầu di chuyển (Nguon) và cột đích là nơi cần di chuyển đến (Dich). Ở đây không cần lưu cột trung gian vì có 3 cột đánh số là 1, 2 và 3 thì cột trung gian để di chuyển là: 6 – (Nguon+Dich).

Đầu tiên đưa vào stack thông tin di chuyển {n, 1, 2}, tức là di chuyển n đĩa từ cột 1 sang cột thứ 2 qua cột trung gian là 6-(1+2) = 3.

Tại mỗi bước khi lấy trong stack ra một phần tử. chúng ta thực hiện như sau: Nếu N = 1: ⇒ di chuyển đĩa từ cột Nguon -> cột Dich.

Ngược lại (nếu N > 1):

- Xác định cột trung gian TrungGian = 6 – (Nguon+Dich)
- Push ⇒ stack thông tin di chuyển {N-1, TrungGian, Dich}
- Push ⇒ stack thông tin di chuyển {1, Nguon, Dich}
- Push ⇒ stack thông tin di chuyển {N-1, Nguon, TrungGian}

Quá trình còn thực hiện khi stack khác rỗng.

Nhận xét: Lưu ý thứ tự khi đưa vào thông tin di chuyển vào stack. Trong phần trên thông tin {N-1, Nguon, TrungGian} được đưa vào stack sau cùng nên chúng sẽ được lấy ra trước tiên, kế đến là thông tin di chuyển {1, Nguon, Dich} và cuối cùng là thông tin di chuyển {N-1, TrungGian, Dich}.

BÀI 3.1.2: SỬ DỤNG CẤU TRÚC STACK ĐỂ CHUYỂN MỘT BIỂU THỨC TRUNG TỐ SANG HẬU TỐ ? (THAM KHẢO)

1. Nhập biểu thức trung tố: toán hạng, toán tử và dấu ngoặc
   VD: (20+5)/5+(7-3)*100
2. Chuyển biểu thức trung tố thành hậu tố (xuất ra màn hình)
   VD: 20 5 + 5 / 7 3 – 100 * +
3. Tính giá trị của biểu thức hậu tố
   VD: (20+5)/5+(7-3)*100 = 405

Yêu cầu:

Sinh viên cài đặt stack dùng danh sách liên kết:
Cài đặt các thao tác: IsEmpty, NewNode, FreeNode, Pop, Push… trên Stack.

Hướng dẫn:

1. Chuyển biểu thức trung tố thành hậu tố:
Duyệt biểu thức trung tố từ trái sang phải

- Nếu gặp toán hạng thì ghi vào chuỗi kết quả
- Nếu gặp dấu mở ngoặc thì push ⇒ stack
- Nếu gặp toán tử gọi là O1 thực hiện các bước sau:
- Chừng nào còn một toán tử O2 ở đỉnh stack và độ ưu tiên của O1 ≤ độ ưu tiên O2 thì lấy O2 ra khỏi stack và ghi vào chuỗi kết quả.
- Push O1 ⇒ stack
- Nếu gặp dấu đóng ngoặc: thì lấy toán tử trong stack ra cho đến khi lấy được

dấu mở ngoặc (lưu ý: pop dấu mở ngoặc ra, nhưng ko xuất ra chuỗi kết quả)

Khi đã duyệt hết biểu thức trung tố, lấy tất cả toán tử trong stack và ghi vào chuỗi kết quả.

2. Tính giá trị biểu thức hậu tố:

Đọc biểu thức từ trái sang phải, Nếu là toán hạng: Push ⇒ stack. Nếu gặp toán tử:

- Lấy 2 toán hạng trong stack ra

- Tính giá trị của 2 toán hạng đó theo toán tử

- Push kết quả ⇒ stack

Khi quá trình kết thúc thì con số cuối cùng còn lại trong stack chính là giá trị của biểu thức đó.

BÀI 3.1.3: BÀI TOÁN ĐỐI SÁNH THẺ HTML

Trong một văn bản HTML thì mỗi thẻ mở phả i đi kèm với 1 thẻ đóng, ví dụ:

```
<div class="header-right">
    <form method="get" id="searchform" action ="http://wordpress.com/" >
        <div>< label class ="hidden" for="s"> Tìm kiếm:</label >
        <input type="text" value ="" name="s" id="s" />
        <input type="submit" id ="searchsubmit" value="Go" /></ div>
    </form >
</ div>
```

Hãy viết chương trình đọc vào 1 file .html và kiểm tra xem trong file đó có thẻ nào bị lỗi hay không?

BÀI 3.1.4: SỬ DỤNG CẤU TRÚC STACK ĐỂ ĐẢO XÂU KÍ TỰ ?

BÀI 3.1.5: SỬ DỤNG CẤU TRÚC STACK ĐỂ CHUYỂN MỘT SỐ THẬP PHÂN SANG DẠNG NHỊ PHÂN ?

BÀI 3.1.6: DẤU NGOẶC ĐÚNG

Cho các đoạn văn chứa các dấu ngoặc, có thể là ngoặc đơn đơn ("()") hoặc ngoặc vuông ("[]"). Một đoạn văn đúng là đoạn mà với mỗi dấu mở ngoặc thì sẽ có dấu đóng ngoặc tương ứng và đúng thứ tự. Nhiệm vụ của bạn kiểm tra xem đoạn văn có đúng hay không.

Input

Gồm nhiều bộ test, mỗi bộ test trên một dòng chứa đoạn văn cần kiểm tra có thể bao gồm: các kí tự trong bảng chữ cái tiếng Anh, dấu cách, và dấu ngoặc (ngoặc đơn hoặc ngoặc vuông). Kết thúc mỗi bộ test là một dấu chấm. Mỗi dòng có không quá 100 kí tự.

Dữ liệu kết thúc bởi dòng chứa duy nhất một dấu chấm.

Output

Với mỗi bộ test, xuất ra trên một dòng "yes" nếu đoạn văn đúng, ngược lại in ra "no".

Ví dụ:

Input:

So when I die (the [first] I will see in (heaven) is a score list).

[first in] (first out).

Half Moon tonight (At least it is better than no Moon at all].

A rope may form)(a trail in a maze.

Help(I[m being held prisoner in a fortune cookie factory)].

([(([([ ]) ( ) (( ))]))])…

Output:

yes
yes
no
no
no
yes
yes

BÀI 3.1.7: CỘNG, TRỪ HAI ĐA THỨC

Cho hai đa thức A bậc n và đa thức B bậc m được ghi lại tương ứng trong file dathuc1.in và dathuc2.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên K là số các số hạng của đa thức;
- K dòng kế tiếp, mỗi dòng ghi lại hệ số và số mũ của số hạng hạng đa thức.

Hãy viết chương trình tính hiệ u hai đa thức A và B và ghi lại đa thức kết quả vào file ketqua.out theo khuôn dạng như trên. Ví dụ với đa thức:

$$P_n(x) = 10x^{30000} + 5x^{1000} + 3x^2 + 3$$
$$Q_m(x) = 8x^{20000} + 3x^{1000} + 3x^{500} + 7x^{100} + 6x$$

sẽ được biểu diễn và tính toán cho ra file kết quả sau

dathuc1.in	**dathuc2.in**	**ketqua.out**
4 10 30000 5 1000 3 2 3 0	5 8 20000 3 1000 3 500 7 100 6 1	10 30000 -8 20000 2 1000 -3 500 -7 100 3 2 -6 1 3 0

BÀI 3.1.8: LẬP TRÌNH LỚP STACK

Viết một lớp với các hàm trong stack như sau:

```
class stack
{
            int data[10];
            int top;
public :
            stack(){top=-1;}
            void push();
            void pop();
}
```

BÀI 3.1.9:

Sử dụng stack để tính biểu thức hậu tố sau và chỉ ra nội dung của stack sau khi chạy mỗi lệnh. Không cần viết code. Giả sử rằng bạn đang sử dụng hàm push và pop của lớp stack.

AB-CD+E*+ (với A=5, B=3, C=5, D =4, và E=2)

BÀI 3.1.10:

Tính biểu thức hậu tố sau đây sử dụng stack và chỉ ra nội dung stack sau mỗi lệnh:

50,40,+,18, 14,-, *,+

BÀI 3.1.11:

Tính biểu thức hậu tố sau đây sử dụng stack và chỉ ra nội dung stack sau mỗi lệnh:

TRUE, FALSE, TRUE, FALSE, NOT, OR, TRUE, OR, OR, AND

BÀI 3.1.12:

Sử dụng cấu trúc dữ liệu *mảng* (*ngăn xếp*), hãy viết chương trình chuyển đổi số tự nhiên n thành số ở hệ cơ số b (2≤b≤32).

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số lượng các test (k≤100).
- K dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một cặp số *n*, *b*. Hai số được viết cách nhau một vài khoảng trống.

Kết quả ra (Output): ghi lại K dòng trong file ketqua.out, mỗi dòng ghi lại bộ ba số *n, k, x*. Trong đó *x* là số ở hệ cơ số *b* được chuyển đổi từ *n*. Ví dự dưới đây minh họa cho file input và output của bài toán.

Input.in	**Output.out**
5 8 2 32 16 255 16 100 10 64 32	8 2 1000 32 16 20 255 16 FF 100 10 100 64 32 20

BÀI 3.1.13:

Sử dụng cấu trúc dữ liệu *ngăn xếp*, hãy viết chương trình tìm số các số tự nhiên N thỏa mãn đồng thời những điều kiện dưới đây ($N \le 2^{31}$):

- N là số có K chữ số (K≤15).
- N là số thuận nghịch (số đối xứng).
- Chuyển đổi N thành số hệ cơ số B cũng là một số thuận nghịch (2≤N≤512).
- Thời gian thực hiện chương trình không quá 1sec.

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số lượng các test (M≤100).
- M dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một cặp số *N, B*. Hai số được viết cách nhau một vài khoảng trống.

Kết quả ra (Output): ghi lại M dòng trong file ketqua.out, mỗi dòng ghi lại bộ ba số *N, B, X*. Trong đó *X* là số các số có N chữ số ở hệ cơ số B thỏa mãn yêu cầu của bài toán. Ví dự dưới đây minh họa cho file input và output của bài toán.

Input.in	**Output.out**
3 8 2	8 2 1000 32 16 20

32	16		255	16	FF
255	16				

BÀI 3.1.14:

Sử dụng cấu trúc dữ liệu *ngăn xếp*, hãy viết chương trình tìm số các số tự nhiên N thỏa mãn đồng thời những điều kiện dưới đây ($N \leq 2^{31}$):

- N là số có K chữ số (K≤15).
- N là số thuận nghịch (số đối xứng).
- Chuyển đổi N thành số hệ cơ số B cũng là một số thuận nghịch (2≤N≤512).
- Thời gian thực hiện chương trình không quá 1sec.

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên M là số lượng các test (M≤100).
- M dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một cặp số *N, B*. Hai số được viết cách nhau một vài khoảng trống.

Kết quả ra (Output): ghi lại M dòng trong file ketqua.out, mỗi dòng ghi lại bộ ba số *N, B, X*. Trong đó *X* là số các số có N chữ số ở hệ cơ số B thỏa mãn yêu cầu của bài toán. Ví dụ dưới đây minh họa cho file input và output của bài toán.

Input.in		**Output.out**		
3		2	2	2
2	2	2	3	0
2	3	5	16	6
5	16			

BÀI 3.1.15:

Cho file dữ liệu trungto.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các biểu thức số học được biểu diễn dưới dạng trung tố;
- N dòng kế tiếp, mỗi dòng ghi lại một biểu thức trung tố.

Hãy sử dụng cấu trúc dữ liệu kiểu ngăn xếp viết chương trình dịch chuyển các biểu thức trung tố trong file trungto.in thành file hauto.out. Các biểu thức hậu tố dịch chuyển được ghi lại trong file hauto.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các biểu thức hậu tố dịch chuyển được;
- N dòng kế tiếp, mỗi dòng ghi lại một biểu thức hậu tố.

Ví dụ dưới đây sẽ minh họa cho file trungto.in và hauto.out.

trungto.in
4
(a + b)
(a - b)
(a / b)
(a * b)

hauto.out
4
a b +
a b –
a b /
a b *

BÀI 3.1.16:

Cho file dữ liệu hauto.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các biểu thức số học được biểu diễn dưới dạng hậu tố;
- N dòng kế tiếp, mỗi dòng ghi lại một biểu thức hậu tố.

Hãy sử dụng cấu trúc dữ liệu kiểu ngăn xếp viết chương trình tính toán giá trị của các biểu thức hậu tố trong file hauto.in. Các biểu thức hậu tố dịch chuyển được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các biểu thức hậu tố;
- N dòng kế tiếp, mỗi dòng ghi lại giá trị của một biểu thức hậu tố trong file.

Ví dụ dưới đây sẽ minh họa cho file hauto.in và ketqua.out.

hauto.out
4
3 2 +
3 2 –
3 2 /
3 2 *

ketqua.out
4
5
1
1
6

1.11.Kỹ thuật xử lý trên hàng đợi

BÀI 3.2.1: LẬP TRÌNH LỚP VỚI CÁC HÀM CHO HÀNG ĐỢI VÒNG

Hoàn thành lớp với các hàm cho hàng đợi vòng như sau:

```
class queue
{
            int data[10];
            int front, rear;
public :
            queue(){front=-1;rear=-1}
            void add();
            void remove();
}
```

BÀI 3.2.2: VIẾT CHƯƠNG TRÌNH QUẢN LÝ KHO

Viết chương trình quản lý kho đơn giản thực hiện các chức năng sau:

1. Cho phép thêm một mặt hàng vào kho
2. Xuất một mặt hàng ra khỏi kho
3. Xem tất cả hàng hoá trong kho
4. Xem mặt hàng nào kế tiếp sẽ được xuất kho

Yêu cầu:

1. Cài đặt cấu trúc dữ liệu HàngHoá: có các dữ liệu nào liệt kê ra
2. Cài đặt một Queue chứa các hàng hoá trong kho
3. Cài đặt các thao tác trên Queue
4. Cài đặt các chức n ăng theo mô tả của bài tập.

BÀI 3.2.3:

Nhập vào queue 1 mảng các nhân viên bao gồm các thông tin sau : mã nhân viên, họ tên nhân viên, hệ số lương và in thông tin các nhân viên này ra màn hình.

BÀI 3.2.4: *HÀNG ĐỢI*

Cho một hàng đợi đã có một số phần tử. Một thao tác di chuyển trong hàng đợi được định nghĩa như sau:

vị trí xuất phát to *vị trí đến*

tức là di chuyển phần tử ở *vị trí xuất phát* đến *vị trí đến*. Ví dụ, hàng đợi ban đầu là:

Item1 Item2 Item3 Item4 Item5

Thì thao tác 5 to 2 sẽ đưa hàng đợi về dạng:

Item1 Item5 Item2 Item3 Item4

Ta có áp dụng nhiều thao tác di chuyển cùng một lúc. Ví dụ nếu hàng đợi là:

Item1 Item2 Item3 Item4 Item5 Item6 Item7 Item8

Và chuỗi thao tác là

2 to 6; 6 to 3; 4 to 5; 5 to 2; 7 to 4; 8 to 1

Thì ta có hàng đợi mới sẽ là:

Item8 Item5 Item6 Item7 Item4 Item2 Item1 Item3

Chú ý: trong chuỗi thao tác thì không được có hai thao tác nào trùng nhau. Các thao tác thực hiện đồng thời và phần tử nào được di chuyển đến vị trí mới thì bắt buộc phải ở vị trí đó sau khi hoàn thành, các phần tử khác không liên quan được đẩy sang các vị trí còn lại nhưng vẫn giữ thứ tự trước sau như lúc đầu. Như trong ví dụ trên, vị trí 7,8 không được phần tử nào di chuyển đến nên hai item còn lại là 1 và 3 *(là các vị trí không bị di chuyển đi)* sẽ ở vị trí đó.

Bài toán đặt ra là cho trước hàng đợi và chuỗi thao tác di chuyển, hãy xác định hàng đợi kết quả.

Input:

Dòng đầu ghi số bộ test không quá 100. Mỗi bộ test gồm:

- Dòng đầu ghi hai số m và n (1<=n,m<=20). Trong đó m là số phần tử trong hàng đợi, n là số thao tác di chuyển.
- Dòng tiếp theo ghi các phần tử trong hàng đợi ban đầu, mỗi phần tử là một chuỗi ký tự ngắn (1 đến 8 ký tự), các phần tử cách nhau một khoảng trống. Không có hai phần tử nào trung nhau.
- n dòng tiếp theo, mỗi dòng ghi một thao tác di chuyển dưới dạng 2 số nguyên dương theo thứ tự là vị trí xuất phát và vị trí đến.

Output

- Với mỗi bộ test, in ra màn hình trạng thái hàng đợi sau khi áp dụng các thao tác di chuyển.

Ví dụ:

Input:

3
5 1
alpha beta gamma delta epsilon
5 2
8 6
a b c d e f g h
2 6
6 3
4 5
5 2
7 4
8 1
3 2
foo bar baz
3 11 3

Output:

alpha epsilon beta gamma delta
h e f g d b a cbaz bar foo

BÀI 3.2.5:

Viết chương trình con thêm một phần tử trong danh sách đã có thứ tự sao cho ta vẫn có một danh sách có thứ tự bằng cách vận dụng các phép toán cơ bản trên danh sách.

BÀI 3.2.6:

Viết chương trình con tìm kiếm và xóa một phần tử trong danh sách có thứ tự.

BÀI 3.2.7:

Viết chương trình con nhận vào từ bàn phím một dãy số nguyên, lưu trữ nó trong một danh sách có thứ tự không giảm, theo cách sau: với mỗi phần tử được nhập vào chương trình con phải tìm vị trí thích hợp để xen nó vào danh sách cho đúng thứ tự. Viết chương trình con trên cho trường hợp danh sách được cài đặt bằng mảng và cài đặt bằng con trỏ và trong trường hợp tổng quát (dùng các phép toán cơ bản trên danh sách).

BÀI 3.2.8:

Viết chương trình con loại bỏ các phần tử trùng nhau (giữ lại duy nhất 1 phần tử) trong một danh sách có thứ tự không giảm, trong hai trường hợp: cài đặt bằng mảng và cài đặt bằng con trỏ.

BÀI 3.2.9:

Viết chương trình con nhận vào từ bàn phím một dãy số nguyên, lưu trữ nó trong một danh sách có thứ tự tăng không có hai phần tử trùng nhau, theo cách sau: với mỗi phần tử được nhập vào chương trình con phải tìm kiếm xem nó có trong danh sách chưa, nếu chưa có thì xen nó vào danh sách cho đúng thứ tự. Viết chương trình con trên cho trường hợp danh sách được cài đặt bằng mảng và cài đặt bằng con trỏ.

BÀI 3.2.10:

Viết chương trình con trộn hai danh sách liên kết chứa các số nguyên theo thứ tự tăng để được một danh sách cũng có thứ tự tăng.

BÀI 3.2.11:

Viết chương trình con xoá khỏi danh sách lưu trữ các số nguyên các phần tử là số nguyên lẻ, cũng trong hai trường hợp: cài đặt bằng mảng và bằng con trỏ.

BÀI 3.2.12:

Viết chương trình con tách một danh sách chứa các số nguyên thành hai danh sách: một danh sách gồm các số chẵn còn cái kia chứa các số lẻ.

BÀI 3.2.13:

Hình dưới đây biểu diễn cho mảng SPACE có 10 phần tử dùng để biểu diễn danh sách bằng con nháy (cursor) và hai danh sách L1 ; L2 đang có trong mảng

		Element	Next
	0	w	9
	1	h	4
	2		8
	3		-1
	4	x	6
$L_1 \rightarrow$	5	g	1
	6	i	-1
$L_2 \rightarrow$	7	y	0
	8		3
	9	u	-1

SPACE

a. Hãy liệt kê các phần tử trong mỗi danh sách L1, L2.

b. Vẽ lại hình đã cho lần lượt sau các lời gọi INSERT_LIST('o',1,L1), INSERT_LIST('m',6,L1), INSERT_LIST('k',9,L1).

c. Vẽ lại hình ở câu b. sau khi xoá : x,y.

BÀI 3.2.14:

Cho cặp số S và T là các số nguyên tố có 4 chữ số (Ví dụ S = 1033, T = 8197 là các số nguyên tố có 4 chữ số). *Sử dụng hàng đợi*, hãy viết chương trình tìm cách dịch chuyển S thành T thỏa mãn đồng thời những điều kiện dưới đây:

- Mỗi phép dịch chuyển chỉ được phép thay đổi một chữ số của số ở bước trước đó (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1733 là hợp lệ);
- Số nhận được cũng là một số nguyên tố có 4 chữ số (ví dụ nếu S=1033 thì phép dịch chuyển S thành 1833 là không hợp lệ, và S dịch chuyển thành 1733 là hợp lệ);
- Số các bước dịch chuyển là ít nhất.

Ví dụ với S = 1033, T = 8179 sẽ có ít nhất 6 phép dịch chuyển như sau:

1033 →1733→3733 →3739 →3779 →8779→81779 .

Input:

Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test viết trên một dòng hai số nguyên tố có 4 chữ số.

Output:

Với mỗi bộ test, viết ra trên một dòng số bước của đường nguyên tố ngắn nhất.

Ví dụ cho Input và Output:

INPUT	OUTPUT
3 1033 8179 1373 8017 **1033 1033**	6 7 **0**

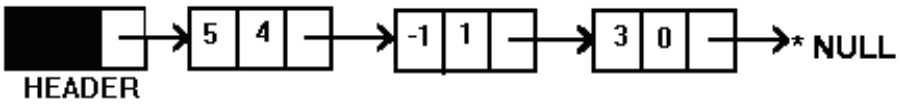
1.12.Kỹ thuật xử lý trên danh sách liên kết

BÀI 3.3.1:

Đa thức $P(x)= a_nx_n+ a_{n-1}x_{n-1}+... + a_1x + a_0$ được lưu trữ trong máy tính dưới dạng một danh sách liên kết mà mỗi phần tử của danh sách là một struct có ba trường lưu giữ hệ số, số mũ,

và trường NEXT trỏ đến phần tử kế tiếp. Chú ý cách lưu trữ đảm bảo thứ tự giảm dần theo số mũ của từng hạng tử của đa thức.

Ví dụ: đa thức 5x4 - x + 3 được lưu trữ trong danh sách có 3 phần tử như sau:



a. Hãy viết chương trình thực hiện được sự lưu trữ này.

b. Dựa vào sự cài đặt ở trên, viết chương trình con thực hiện việc cộng hai đa thức.

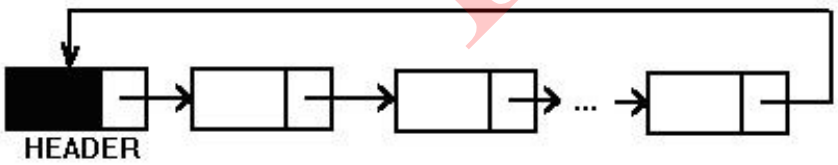
c. Viết chương trình con lấy đạo hàm của đa thức.

BÀI 3.3.2:

Để lưu trữ một số nguyên lớn, ta có thể dùng danh sách liên kết chứa các chữ số của nó. Hãy tìm cách lưu trữ các chữ số của một số nguyên lớn theo ý tưởng trên sao cho việc cộng hai số nguyên lớn là dễ dàng thực hiện. Viết chương trình con cộng hai số nguyên lớn.

BÀI 3.3.3:

Để tiện cho việc truy nhập vào danh sách, người ta tổ chức danh sách liên kết có dạng sau, gọi là danh sách nối vòng:



Hãy viết khai báo và các chương trình con cơ bản để cài đặt một danh sách nối vòng.

BÀI 3.3.4:

Hãy cài đặt một ngăn xếp bằng cách dùng con trỏ.

a. Dùng ngăn xếp để viết chương trình con đổi một số thập phân sang số nhị phân.

b. Viết chương trình con/hàm kiểm tra một chuỗi dấu ngoặc đúng (chuỗi dấu ngoặc đúng là chuỗi dấu mở đóng khớp nhau như trong biểu thức toán học).

BÀI 3.3.5:

Ta có thể cài đặt 2 ngăn xếp vào trong một mảng, gọi là ngăn xếp hai đầu hoạt động của hai ngăn xếp này như sơ đồ sau:

Hình vẽ mảng chứa 2 ngăn xếp

Hãy viết các chương trình con cần thiết để cài đặt ngăn xếp hai đầu.

BÀI 3.3.6:

Mô phỏng việc tạo buffer in một file ra máy in.

Buffer xem như là một hàng, khi ra lệnh in file máy tính sẽ thực hiện một cách lặp quá trình sau cho đến hết file:

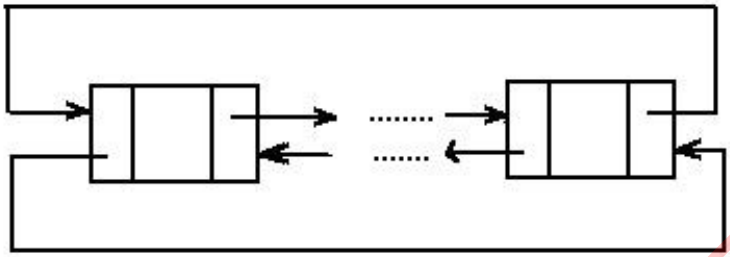
1. Đưa nội dung của tập tin vào buffer cho đến khi buffer đầy hoặc hết file.
2. In nội dung trong buffer ra máy in cho tới khi hàng rỗng.
3. Hãy mô phỏng quá trình trên để in một file văn bản lên từng trang màn hình.

BÀI 3.3.7:

Cài đặt danh sách liên kết kép với các phép toán khởi tạo danh sách rỗng, thêm xoá một phần tử.

BÀI 3.3.8:

Danh sách liên kết kép nối vòng có dạng sau:



Hãy cài đặt danh sách liên kết kép dạng nối vòng như trên.

BÀI 3.3.9: *GIẤY KHAI SINH*

Một buổi họp mặt đại gia đình nhân dịp cụ già Ted tròn 100 tuổi, người ta muốn sắp xếp con cháu của cụ theo thứ tự từ tuổi cao xuống thấp. Giả sử ta có thông tin về giấy khai sinh của từng người đó. Mỗi giấy khai sinh chỉ viết ba thông tin đơn giản gồm: *Tên người cha, Tên người con, Tuổi của người cha lúc sinh con*.

Hãy giúp đại gia đình trên tính ra tuổi của từng người con cháu cụ Ted và viết ra danh sách theo thứ tự từ tuổi cao xuống thấp.

Input

Dòng đầu ghi số bộ test (không quá 100). Với mỗi bộ test:

- Dòng đầu tiên ghi số X (0<X<100) là số người con cháu cần sắp xếp.
- Tiếp theo là X dòng, mỗi dòng ghi thông tin về một giấy khai sinh của từng người (thứ tự ngẫu nhiên) gồm 3 thành phần, mỗi thành phần cách nhau một khoảng trống:
    - Tên người cha: không quá 20 ký tự và không chứa khoảng trống
    - Tên người con: không quá 20 ký tự và không chứa khoảng trống
    - Tuổi của người cha khi sinh con: 1 số nguyên dương, không quá 100.

Output

- Với mỗi bộ test, in ra màn hình thứ tự bộ test (xem thêm trong bộ test ví dụ), sau đó lần lượt là từng người trong danh sách tuổi từ cao xuống thấp (không tính cụ Ted). Mỗi người viết ra hai thông tin: tên, một khoảng trống rồi đến tuổi của người đó.
- Nếu hai người có cùng tuổi thì xếp theo thứ tự từ điển.

Ví dụ:

Input:

2

1

Ted Bill 25

4

Ray James 40

James Beelzebub 17

Ray Mark 75

Ted Ray 20

Output:

DATASET 1

Bill 75

DATASET 2

Ray 80

James 40

Beelzebub 23

Mark 5

BÀI 3.3.10:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ W trong tập văn bản D_1 *và* D_2, ký hiệu là $P(W)$ được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ W trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản data1.in và data2.in. *Sử dụng danh sách liên kết đơn* (kép), hãy tìm tập các từ và tần xuất xuất hiện của mỗi từ trong cả hai tập data1.in và data2.in. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ W tìm được theo yêu cầu của bài toán.
- K dòng kế tiếp, mỗi dòng ghi lại một từ W và tần xuất xuất hiện $P(W)$ thỏa mãn yêu cầu của bài toán. W và $P(W)$ được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	3 AB 0.2 AC 0.2 AD 0.2

BÀI 3.3.11:

Cho tập các số tự nhiên trong file data.in được ghi theo từng dòng, mỗi dòng ghi nhiều nhất 5 số, hai số được viết cách nhau một vài khoảng trống. Biết rằng, mỗi số tự nhiên trong file data.in hoặc là số nguyên tố, hoặc là số thuận nghịch và có thể xuất hiện nhiều lần ở những vị trí khác nhau trong file. *Sử dụng cấu trúc dữ liệu danh sách liên kết đơn (kép)*, hãy viết chương trình tách tập các số và đếm số lần xuất hiện của mỗi số trong file data.in thành 3 file ketqua1.out, ketqua2.out, ketqua3.out thỏa mãn những yêu cầu dưới đây.

a) File ketqua1.out ghi lại các số nguyên tố nhưng không là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

b) File ketqua2.out ghi lại các số thuận nghịch nhưng không là nguyên tố cùng với số lần xuất hiện của nó trong file data.in;

c) File ketqua3.out ghi lại các số vừa là số nguyên tố vừa là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

d) Khuôn dạng của các file kết quả được qui định như sau:

- Dòng đầu tiên của mỗi file ghi lại số các số của mỗi file kết quả;
- Những dòng kế tiếp mỗi dòng ghi lại một số cùng với số lần xuất hiện của nó trong file data.in. Hai số được viết cách nhau một vài khoảng trống.

Ví dụ dưới đây minh họa cho các file data.in, ketqua1.out, ketqua2.out và ketqua3.out.

Data.in
```
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
```

Ketqua1.out
```
2
10007 4
10009 4
```

Ketqua2.out
```
2
10801 4
10901 4
```

Ketqua3.out
```
2
13831 2
34543 2
```

BÀI 3.3.12:

Cho tập các số tự nhiên trong file data.in được ghi theo từng dòng, mỗi dòng ghi nhiều nhất 5 số, hai số được viết cách nhau một vài khoảng trống. Biết rằng, mỗi số tự nhiên trong file data.in hoặc là số nguyên tố, hoặc là số thuận nghịch và có thể xuất hiện nhiều lần ở những vị trí khác nhau trong file. *Sử dụng cấu trúc dữ liệu danh sách liên kết đơn (kép)*, hãy viết chương trình tách tập các số và đếm số lần xuất hiện của mỗi số trong file data.in thành 3 file ketqua1.out, ketqua2.out, ketqua3.out thỏa mãn những yêu cầu dưới đây.

a) File ketqua1.out ghi lại các số nguyên tố nhưng không là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

b) File ketqua2.out ghi lại các số thuận nghịch nhưng không là nguyên tố cùng với số lần xuất hiện của nó trong file data.in;

c) File ketqua3.out ghi lại các số vừa là số nguyên tố vừa là số thuận nghịch cùng với số lần xuất hiện của nó trong file data.in;

d) Khuôn dạng của các file kết quả được qui định như sau:

- Dòng đầu tiên của mỗi file ghi lại số các số của mỗi file kết quả;
- Những dòng kế tiếp mỗi dòng ghi lại một số cùng với số lần xuất hiện của nó trong file data.in. Hai số được viết cách nhau một vài khoảng trống.

Ví dụ dưới đây minh họa cho các file data.in, ketqua1.out, ketqua2.out và ketqua3.out.

Data.in
```
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
10007   10009   10801   10901   13831
10007   10009   10801   10901   34543
```

Ketqua1.out
```
2
10007 4
10009 4
```

Ketqua2.out
```
2
10801 4
10901 4
```

Ketqua3.out
```
2
13831 2
34543 2
```

BÀI 3.3.13:

Cho hai đa thức A bậc n và đa thức B bậc m được ghi lại tương ứng trong file dathuc1.in và dathuc2.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên K là số các số hạng của đa thức;
- K dòng kế tiếp, mỗi dòng ghi lại hệ số và số mũ của số hạng hạng đa thức.

Hãy sử dụng *danh sách liên kết đơn* trình tính tổng hai đa thức A và B và ghi lại đa thức kết quả vào file ketqua.out theo khuôn dạng như trên. Ví dụ với đa thức

$$P_n(x) = 10x^{30000} + 5x^{1000} + 3x^2 + 3$$
$$Q_m(x) = 8x^{20000} + 3x^{1000} + 3x^{500} + 7x^{100} + 6x$$

sẽ được biểu diễn và tính toán cho ra file kết quả sau

dathuc1.in	**dathuc2.in**	**ketqua.out**
4	5	8
10 30000	8 20000	10 30000
5 1000	3 1000	8 20000
3 2	3 500	8 1000
3 0	7 100	3 500
	6 1	7 100
		3 2
		6 1
		3 0

BÀI 3.3.14:

Cho hai đa thức A bậc n và đa thức B bậc m được ghi lại tương ứng trong file dathuc1.in và dathuc2.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên K là số các số hạng của đa thức;
- K dòng kế tiếp, mỗi dòng ghi lại hệ số và số mũ của số hạng hạng đa thức.

Hãy sử dụng *danh sách liên kết đơn* viết chương trình tính hiệu hai đa thức A và B và ghi lại đa thức kết quả vào file ketqua.out theo khuôn dạng như trên. Ví dụ với đa thức

$$P_n(x) = 10x^{30000} + 5x^{1000} + 3x^2 + 3$$
$$Q_m(x) = 8x^{20000} + 3x^{1000} + 3x^{500} + 7x^{100} + 6x$$

sẽ được biểu diễn và tính toán cho ra file kết quả sau

dathuc1.in	dathuc2.in	ketqua.out
4 10 30000 5 1000 3 2 3 0	5 8 20000 3 1000 3 500 7 100 6 1	8 10 30000 -8 20000 3 1000 -3 500 -7 100 3 2 -6 1 3 0

1.13. Khử đệ qui dựa vào ngăn xếp và danh sách liên kết

BÀI 3.4.1: KHỬ ĐỆ QUY THỦ TỤC THÁP HÀ NỘI (THAM KHẢO)

Tham khảo đề bài 3.1.1.

Hướng dẫn:

Dạng đệ quy thủ tục Tháp Hà Nội là:

THN(n , X , Y, Z) ≡ if(n > 0)

{

THN (n – 1 , X , Z , Y) ;

Move (X , Z) ;

THN (n – 1 , Y , X , Z) ;

}

Với n là số đĩa, X là cột đầu, Z là cột cuối, Y là cột giữa, Move(X, Z) là các thao tác chuyển 1 đĩa từ cột X tới cột Z.

Trong trường hợp này:

X là bộ (n, X, Y, Z).

C(X) là biểu thức boolean (n <= 0).

D(X), A(X) là thao tác rỗng.

B(X) = B(n, X, Y, Z) là thao tác Move(X, Z).

g(X) = g(n, X, Y, Z) = (n -1, Y, X, Z).

Giải thuật không đệ quy tương đương là:

```
{ Creat_Stack (S) ;
Push (S ,(n,X,Y,Z,1)) ;
While ( n > 0 ) {
Push (S ,(n,X,Y,Z,2)) ;
n = n – 1 ;
Swap (Y,Z ) ; //(* Swap(a, b) là thủ tục hoán đổi nội dung 2 biến a, b)
}
POP (S,(n,X,Y,Z,k)) ;
if ( k != 1 ) {
Move (X ,Z ) ;
n = n – 1 ;
Swap (X ,Y ) ;
}
until ( k = 1 ) ;
}
```


BÀI 3.4.2: *KHỬ ĐỆ QUI TÍNH N! BẰNG VÒNG LẶP*

F(n) = C Khi n = n_0 (C là một hằng)

= g(n, f(n, n-1)) khi n> n_0

Ví dụ: Hàm giai thừa FAC(n) = n! = 1 khi n = 0

BÀI 3.4.3: *TÌM ƯỚC CHUNG LỚN NHẤT (UCLN) CỦA 2 SỐ NGUYÊN DỰA VÀO THUẬT TOÁN EUCLIDE.*

Giải thuật đệ quy tìm UCLN(m, n) bằng thuật toán Euclide:

UCLN(m , n , var us) ≡ if (n = 0) us = m ;

else USCLN(n , m mod n , us) ;

BÀI 3.4.4: *Khử đệ qui hàm tính tổ hợp chập k của n phần tử*

```
int TH(int k, int n){
// với giả thiết 0<=k<=n
if ((k==0) || (k==n))
return 1;
else
return (TH(k-1,n-1)+TH(k,n-1));
}
```

BÀI 3.4.5: *Khử đệ qui hàm tính dãy Fibonaci theo n*

```
int Fibo(int n){
//với giả thiết n>=0
if ((n==0) || (n=1))
return 1;
else
return (Fibo(n-2)+Fibo(n-1));
}
```

BÀI 3.4.6: *KHỬ ĐỆ QUY THUẬT TOÁN QUICKSORT*

BÀI 3.4.7:

Cho dãy $W_n = x_0x_1...x_n$ với:

- $x_0 = 0$

- Ở mỗi bước kế tiếp, nếu có dãy con W_{n-1} ở bước n-1 thì ghép thêm vào cuối dãy W_{n-1} là dãy phủ định của dãy W_{n-1} để nhận dãy mới là: W_n là W_{n-1} ghép với phủ định của W_{n-1}.

Để xây dựng dãy con phủ định của W_{n-1} bằng cách thay 0 bằng 1, 1 thay bằng 0. Cụ thể như sau:

- $W_0 = 0$
- $W_1 = 01$
- $W_2 = 0110$
- $W_3 = 01101001$

Viết hàm xác định phần tử thứ k bằng đệ qui và không đệ qui.

Ví dụ: k = 0 giá trị thứ 0 là 0, k = 7 giá trị thứ 7 là 1.

Gợi ý:

Bước 1: Xác định W_n ngắn nhất có chứa phần tử thứ k. Nghĩa là xác định n nhỏ nhất sao cho $2^n \geq k + 1$.

Bước 2: Gọi x[k] là giá trị phần tử thứ k. Khi đó phần tử phủ định của x[k] là 1 - x[k].

Trong dãy W_n, các phần tử thứ 2^{n-1}, $2^{n-1}+1$, ..., 2^n-1 là phủ định của các phần thử thứ 0, 1, ..., $2^{n-1}-1$ nên x[k] = 1 - x[k - 2^{n-1}].

Vậy ta có hệ thức đệ qui như sau: **x[k] = 1 - x[k - 2^{n-1}] với x[0] = 0**.

IV.Lập trình trên cây nhị phân

4.1. Cây nhị phân

BÀI 4.1.1:

Xây dựng cấu trúc cây nhị phân với các hàm cơ bản: nhập, xuất, thêm node.

BÀI 4.1.2:

Viết hàm xuất các giá trị chẵn trong cây.

BÀI 4.1.3:

Viết hàm xuất địa chỉ các node trên cây có giá trị lớn hơn a, nhỏ hơn b (a, b là hai giá trị bất kỳ được nhập vào).

BÀI 4.1.4:

Viết hàm xuất các node trên tầng thứ k của cây, với k là số nguyên bất kỳ được nhập vào.

BÀI 4.1.5:

Viết hàm xuất tất cả các nút trên cây theo thứ tự từ tầng 0 đến tầng h-1 của cây (với h là chiều cao của cây)

BÀI 4.1.6:

Đếm số lượng các node có đúng k con (k là số nguyên bất kỳ được nhập vào).

BÀI 4.1.7:

Viết hàm đếm số lượng nút lá mà thông tin tại nút đó là giá trị chẵn.

BÀI 4.1.8:

Viết hàm đếm số lượng nút có đúng 1 con mà thông tin tại nút đó là số nguyên tố.

BÀI 4.1.9:

Viết hàm đếm số lượng nút có đúng 2 con mà thông tin tại nút đó là số chính phương.

BÀI 4.1.10:

Viết hàm đếm số lượng nút trên tầng thứ k của cây

BÀI 4.1.11:

Viết hàm đếm số lượng nút nằm ở tầng cao hơn tầng thứ k của cây

BÀI 4.1.12:

Viết hàm tính tổng các nút trong cây

BÀI 4.1.13:

Viết hàm tính tổng các nút lá trong cây

BÀI 4.1.14:

Viết hàm tính tổng các nút có đúng một con

BÀI 4.1.15:

Viết hàm tính tổng các nút có đúng hai con

BÀI 4.1.16:

Viết hàm tính tổng các nút lá mà thông tin tại nút đó là giá trị chẵn

BÀI 4.1.17:

Viết hàm tính tổng các nút có đúng 1 con mà thông tin tại nút đó là số nguyên tố

BÀI 4.1.18:

Viết hàm tính chiều cao cây

BÀI 4.1.19:

Viết hàm xóa tất cả các nút có bậc cao hơn k

BÀI 4.1.20:

Cho m,n là hai node trên cây. Viết hàm kiểm tra xem liệu m là node bên phải hay bên trái của n trên cây.

BÀI 4.1.21:

Thông tin về một thí sinh bao gồm: số báo danh (kiểu ký tự) và điểm thi (kiểu thực) trong đó số báo danh là trường khóa. CHo danh sách thí sinh như sau:

(G,18);(D,30);(B,15);(M,19);(I,27);(P,30);(F,5);(C,23);(L,15);(N,10); (E,16);(R,25).

a. Khai báo cấu trúc dữ liệu cây nhị phân (dạng con trỏ) để lưu trữ danh sách thí sinh ở trên.

b. xây dựng cây nhị phân tìm kiếm (lần lượt theo từng bước từ danh sách trên

c. Mô tả từng bước tìm thí sinh có số báo danh là E trên cây

d. Viết hàm đếm các thí sinh đủ điểm đỗ (có điểm trên 20)

e. Viết hàm in danh sách thí sinh theo chiều tăng dần của điểm.

4.2. Cây nhị phân tìm kiếm

BÀI 4.2.1:

Xây dựng cấu trúc cây nhị phân tìm kiếm với các hàm sau:

a) Nhập và duyệt cây theo các thứ tự: trước, giữa và sau.
b) Tìm node có giá trị x trên cây.
c) Tìm node có giá trị nhỏ nhất.
d) Tìm node có giá trị lớn nhất.
e) Tính độ cao của cây.
f) Đếm số nút lá của cây.
g) Đếm số nút có đúng 2 cây con.
h) Đếm số nút có đúng 1 cây con.
i) Xóa nút có giá trị x.

BÀI 4.2.2:

Cho một cây nhị phân tìm kiếm. Mỗi node trên cây có info là một số nguyên và liên kết chỉ đến cây con trái và cây con phải.

a. Tính số node, số node lá, số node nhánh trên cây.
b. Tính chiều cao của cây.
c. Tính độ dài của cây.
d. Tính tổng giá trị của các node trên cây.
e. Nhập vào một số nguyên x. Viết thủ tục tìm x trên cây . Nếu tìm thấy hãy in ra màn hình giá trị của các node lớn hơn x.
f. Duyệt cây nhị phân theo các phương pháp NLR, LNR, LRN.

BÀI 4.2.3:

Viết hàm kiểm tra cây nhị phân T có phải là "cây nhị phân tìm kiếm" hay không?

BÀI 4.2.4:

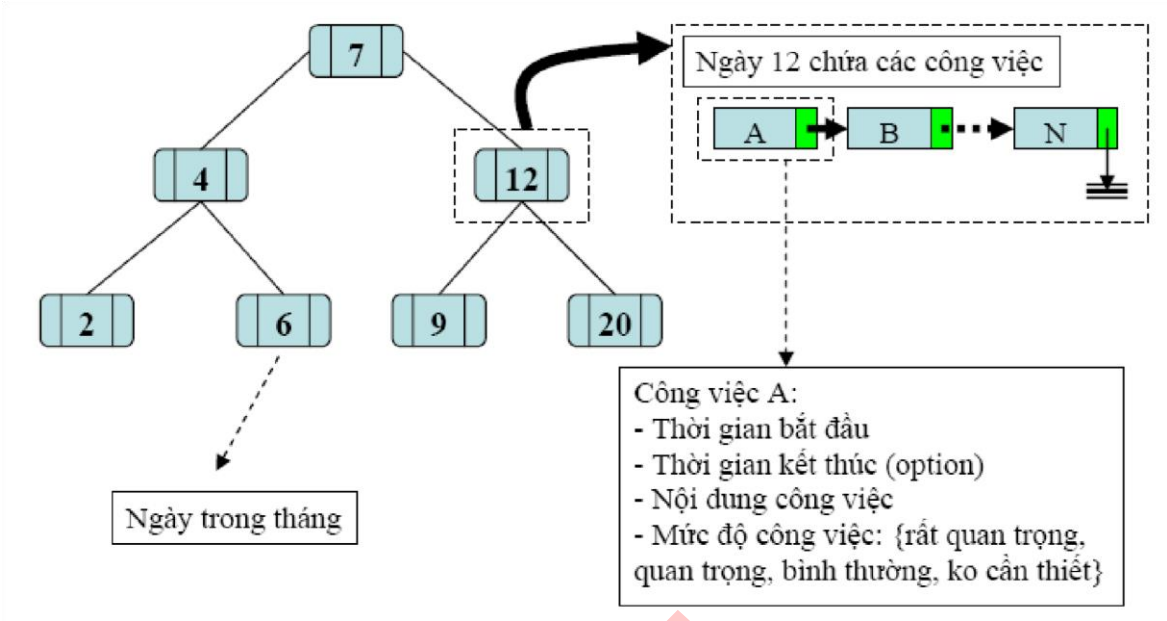
Viết chương trình quản lý lịch công tác trong tháng đơn giản: cho phép nhập vào nội dung công việc cần làm theo ngày, theo giờ. Trong một ngày có thể có nhiều công việc, mỗi công việc có giờ bắt đầu , tên công việc, nội dung công việc, tính chất công việc {rất quan trọng, quan trọng, bình thường, ko cần thiết}…

Chương trình có các chức năng chính như sau:

- Nhập nội dung công việc cần làm theo ngày, theo giờ
- Xem lịch công tác theo ngày yêu cầu
- Xem các công việc theo tính chất: rất quan trọng, quan trọng…
- Xem các công việc đã hoàn tất
- Xem các công việc ch ưa thực hiện
- Xem các công việc từ ngày a đến ngày b
- Xóa hay điều chỉnh lịch công tác. Nếu sau khi điều chỉnh, ngày nào không còn việc phải làm sẽ xóa khỏi lịch công tác.

Yêu cầu : chương trình có cài đặt cây nhị phân tìm kiếm (BST):

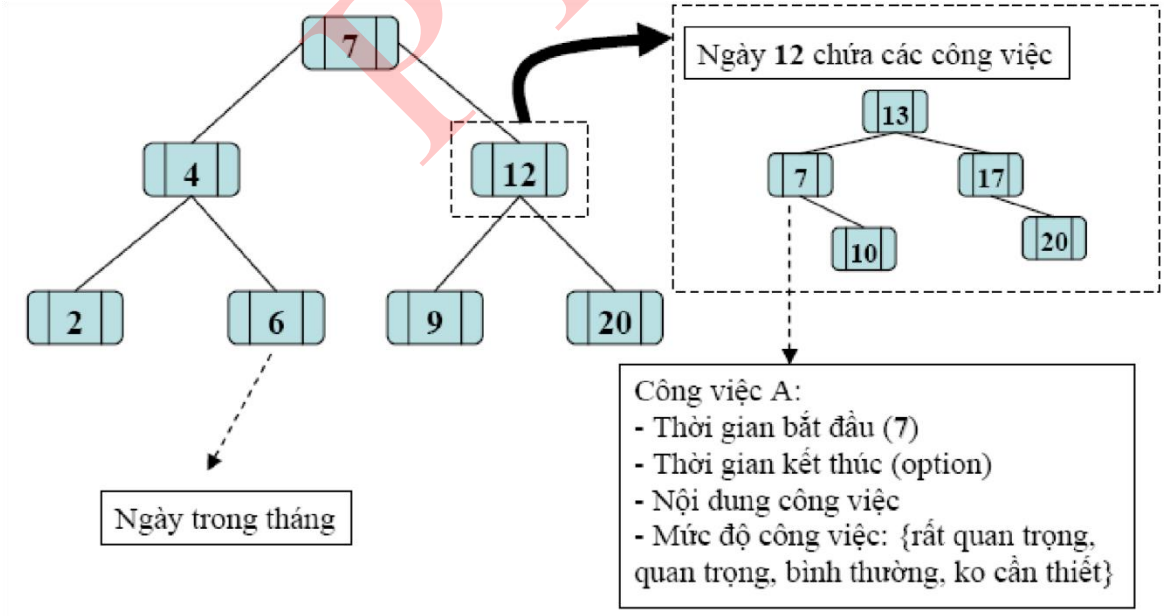
- Mỗi nút trên cây BST là một ngày của lịch công tác
- Trong mỗi nút ngày trên cây lại chứa một **danh sách liên kết** lưu thông tin các công việc.
- Khi thêm một công việc vào một ngày đã tồn tại trên cây, thì công việc này sẽ được đưa vào danh sách liên kết chứa các công việc theo thứ tự tăng dần của **giờ bắt đầu**. Hình vẽ minh họa cấu trúc cây lịch công tác:



Hình: Cấu trúc cây lịch công tác

Nâng cao (**<u>không bắt buộc, dành cho sinh viên khá, giỏi</u>**)

Thay danh sách liên kết chứa công việc trong ngày thành cây nhị phân tìm kiếm, khóa để xây dựng cây BST con là giờ bắt đầu!



Hình: Cấu trúc cây lịch công tác nâng cao

BÀI 4.2.5:

Xét thuật giải tạo cây nhị phân tìm kiếm. Nếu thứ tự các khoá nhập vào là như sau:8, 3, 5, 2, 20, 11, 30, 9, 18, 4 thì hình ảnh cây tạo được như thế nào ? sau đó, nếu hủy lần lượt các nút 5, 20 (theo thứ tự đó) thì cây sẽ thay đổi như thế nào trong từng bước hủy (vẽ lại cây cho mỗi bước).

BÀI 4.2.6:

Hãy đưa ra một cài đặt đệ quy của thuật toán nhị phân tìm kiếm.

BÀI 4.2.7:

Giả sử a[i] = 2i với 1 <= i <= N. Có bao nhiêu vị trí trong bảng được kiểm tra khi dùng tìm kiếm nội suy trong trường hợp tìm kiếm không thành công cho 2k - 1?

BÀI 4.2.8:

Vẽ ra cây nhị phân tìm kiếm có được khi chèn các mẩu tin với các khóa E A S Y Q U E T I O N vào một cây rỗng đã khởi tạo.

BÀI 4.2.9:

Hãy sửa cây nhị phân tìm kiếm sao cho nó lưu tất cả các khóa bằng nhau trong cây. (nếu bấy kỳ node khác trong cây có cùng khóa với một node đã cho thì hoặc cha của nó hoặc một trong các con của nó nên có cùng khóa.)

BÀI 4.2.10:

Viết một chương trình không đệ quy để in ra các khóa trong cây nhị phân tìm kiếm theo thứ tự của khóa.

BÀI 4.2.11:

Vẽ một cây nhị phân tìm kiếm có được khi chèn các mẩu tin với các khóa E A S Y Q U E T I O N vào một cây rỗng đã khởi tạo, và tiếp đó xóa Q.

BÀI 4.2.12:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ W trong tập văn bản D_1 *và* D_2, ký hiệu là $P(W)$ được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ W trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản data1.in và data2.in. Sử dụng *cây nhị phân tìm kiếm*, hãy tìm tập các từ và tần xuất xuất hiện của mỗi từ hoặc xuất hiện trong data1.in hoặc xuất hiện trong data2.in. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ W tìm được theo yêu cầu của bài toán.
- K dòng kế tiếp, mỗi dòng ghi lại một từ W và tần xuất xuất hiện $P(W)$ thỏa mãn yêu cầu của bài toán. W và $P(W)$ được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	7 AB 0.2 AC 0.2 AD 0.2 AE 0.1 AF 0.1 AH 0.1 AK 0.1

BÀI 4.2.13:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ W trong tập văn bản D_1 *và* D_2, ký hiệu là $P(W)$ được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ W trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản *data1.in* và *data2.in*. Sử dụng *cây nhị phân tìm kiếm*, hãy tìm tập các từ và tần xuất xuất hiện của các từ xuất hiện trong file *data1.in* nhưng không xuất hiện trong file *data2.in*. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên *K* là số từ *W* tìm được theo yêu cầu của bài toán.
- *K* dòng kế tiếp, mỗi dòng ghi lại một từ *W* và tần xuất xuất hiện *P*(*W*) thỏa mãn yêu cầu của bài toán. *W* và *P*(*W*) được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	2 AE 0.1 AF 0.1

BÀI 4.2.14:

Ta định nghĩa một từ là dãy các kí tự không chứa khoảng trống (space), dấu tab, dấu xuống dòng (‘\n’), dấu về đầu dòng (‘\r’) và dấu kết thúc dòng (‘\0’). Tần xuất xuất hiện của từ *W* trong tập văn bản D_1 *và* D_2, ký hiệu là *P*(*W*) được tính theo công thức:

$P(W) = \frac{N_1(W) + N_2(W)}{N(D_1) + N(D_2)}$; trong đó $N_i(W)$ là số lần xuất hiện từ *W* trong D_i, $N(D_i)$ là tổng số từ của tập văn bản D_i (i=1, 2).

Cho hai file văn bản data1.in và data2.in. Sử dụng *cây nhị phân tìm kiếm*, hãy tìm tập các từ và tần xuất xuất hiện của mỗi từ trong cả hai tập data1.in và data2.in. Tập các từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên *K* là số từ *W* tìm được theo yêu cầu của bài toán.
- *K* dòng kế tiếp, mỗi dòng ghi lại một từ *W* và tần xuất xuất hiện *P*(*W*) thỏa mãn yêu cầu của bài toán. *W* và *P*(*W*) được viết cách nhau bởi một vài khoảng trống.

Ví dụ với file data1.in và data2.in dưới đây sẽ cho ta file Ketqua.out của bài toán.

data1.in	**data2.in**	**Ketqua.out**
AB AC AD AE AF AB AC AD AE AF	AB AC AD AH AK AB AC AD AH AK	3 AB 0.2 AC 0.2

		AD 0.2

4.3. B-Cây (thuộc tìm kiếm bộ nhớ ngoài)

BÀI 4.3.1:

Hãy cho biết nội dung của B-cây có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một cây khởi động trống với M = 5.

BÀI 4.3.2:

Hãy cho biết nội dung của B-cây có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vòa một cây khởi động trống với M = 6. Dùng phương pháp mà tất cả các mẩu tin đều được lưu trong các node ngoài.

BÀI 4.3.3:

Vẽ một B-cây có được khi chèn mười sáu khóa trùng nhau được chèn vào một cây khởi tạo trống với M=5.

BÀI 4.3.4:

Giả sử rằng trang của cơ sở dữ liệu bị phá hủy. Hãy mô tả cách giải quyết của bạn đối với sự cố này cho mỗi cấu trúc B-cây.

BÀI 4.3.5:

Tại sao các B-cây "từ trên xuống" tốt hơn các B-cây "từ dưới lên" đối với việc truy xuất đồng hành tới dữ liệu? (ví dụ, giả sử hai chương trình chèn cùng một node mới vào cùng một thời điểm.)

4.4. Cây cân bằng

BÀI 4.4.1:

Cho một cây nhị phân T, mỗi nút là một số nguyên. Hãy viết các hàm thực

hiện các yêu cầu sau:

a.Hãy đếm số nút của cây.

b.Cho biết chiều cao của cây AVL

c.Kiểm tra xem T có phải là cây nhị phân tìm kiếm không ?

d.Kiểm tra xem T có phải là cây cân bằng hoàn toàn không ?

e.Kiểm tra xem T có phải là cây nhị phân cân bằng không ?

f.Thêm một phần tử vào cây AVL

g.Hủy một phần tử trên cây AVL.

h.Cân bằng lại một cây vừa bị mất cân bằng.

BÀI 4.4.2:

Hãy vẽ một cây 2-3-4 từ trên xuống có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một cây được khởi tạo trống.

4.5. Cây đỏ đen

BÀI 4.5.1:

Vẽ cây đỏ đen có được khi chèn các ký từ từ A đến K theo thứ tự và mô tả nói chung điều gì xảy ra khi các khóa được chèn vào theo thứ tự tăng.

BÀI 4.5.2:

Thông thường có bao nhiêu liên kết của cây phải bị thay đổi với một thao tác quay kép, và bao nhiêu bị thay đổi trong một cài đặt đã cho.

BÀI 4.5.3:

Hãy phát sinh hai cây đỏ đen ngẫu nhiên 32 node, vẽ chúng và so sánh chúng với với các cây tìm kiếm nhị phân không cân bằng được xây dựng với cùng một tập hợp khóa đó.

BÀI 4.5.4:

Hãy phát sinh 10 cây đỏ đen ngẫu nhiên 1000 node. Tính số phép quay cần thiết để xây dựng các cây và khoảng cách trung bình từ node gốc tới một node ngoài cho mỗi cây. Thảo luận các kết quả.

BÀI 4.5.5:

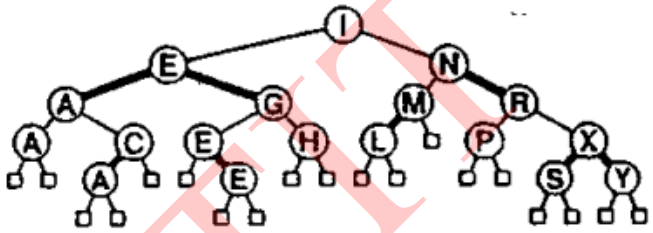
Với một bit màu cho mỗi node, chúng ta có thể biểu diễn 2 node, 3 node, 4 node. Bao nhiêu dạng node khác nhau có thể có nếu chúng ta dùng 2 bit cho mỗi node.

BÀI 4.5.6:

Các thao tác quay được đòi hỏi trong cây đỏ đen khi các 3 node bị đổi thành 4 node bởi một phương pháp "không cân bằng". Tại sao ta không khử bỏ các phép quay bằng cách cho phép các 4 node được biểu diễn như 3 node bất kỳ được nối bằng hai liên kết đỏ (hoàn toàn cân bằng hay không)?

BÀI 4.5.7:

Hãy cho một dãy thao tác chèn mà sẽ xây dựng cây đỏ đen được chỉ trong hình sau:



4.6. Cây quyết định

4.7. Cây mã tiền tố

V.Lập trình trên Đồ thị

5.1. Biểu diễn đồ thị

BÀI 5.1.1:

Hãy biểu diễn đồ thị bằng ma trận kề.

BÀI 5.1.2:

Hãy biểu diễn đồ thị bằng danh sách cạnh.

BÀI 5.1.3:

Hãy biểu diễn đồ thị bằng danh sách kề.

BÀI 5.1.4:

Biểu diễn nào cho đồ thị vô hướng là thích hợp nhất để xem một đỉnh có cô lập (không được nối tới bất kỳ đỉnh nào) hay không?

BÀI 5.1.5:

Cần bao nhiêu bit để lưu trữ trong biểu diễn ma trận kề của đồ thị có hướng V đỉnh và E cạnh? Cần bao nhiêu đối với biểu diễn xâu kề?

BÀI 5.1.6:

Hãy cho ví dụ một đồ thị không thể vẽ được ra giấy mà không có hai cạnh cắt nhau.

BÀI 5.1.7:

Viết một chương trình để xóa đi một cạnh khỏi một đồ thị được biểu diễn bởi xâu kề.

BÀI 5.1.8:

Viết thủ tục adjlist để duy trì các xâu kề theo thứ tự của chỉ số đỉnh. Thảo luận về ưu điểm của cách tiếp cận này.

BÀI 5.1.9:

Cho đồ thị vô hướng G =<V,E> gồm N đỉnh và M cạnh được biểu diễn dưới dạng danh sách kề trong file dske.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;

- N dòng kế tiếp mỗi dòng ghi lại danh sách kề của đỉnh tương ứng. Hai đỉnh trong cùng một danh sách kề được phân biệt với nhau bằng một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy viết chương trình chuyển đổi biểu diễn đồ thị G dưới dạng danh sách kề thành biểu diễn của đồ thị G dưới dạng ma trận kề và danh sách cạnh. Khuôn dạng biểu diễn đồ thị G dưới dạng ma trận kề, danh sách kề được ghi lại trong file mtke.out và dscanh.out theo khuôn dạng sau:

Khuôn dạng file mtke.out:

- Dòng đầu tiên ghi lại số tự nhiên n là số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được ghi cách nhau bởi một vài ký tự trống.

Khuôn dạng file dscanh.out

- Dòng đầu tiên ghi lại số tự nhiên N và M tương ứng với số đỉnh và số cạnh của đồ thị, hai số được ghi cách nhau bởi một vài ký tự trống;
- M dòng kế tiếp mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ đồ thị gồm 5 đỉnh, 5 cạnh được biểu diễn trong file dske.in như dưới đây sẽ cho ta các file mtke.out và dscanh.out tương ứng.

dske.in	***mtke.out***	***dscanh.out***
5 *2 3* *1 4* *1 5* *2* *3*	*5* *0 1 1 0 0* *1 0 0 1 0* *1 0 0 0 1* *0 1 0 0 0* *0 0 1 0 0*	*5 4* *1 2* *1 3* *2 4* *3 5*

BÀI 5.1.10:

Cho đồ thị có hướng G =<V,E> gồm N đỉnh và M cạnh được biểu diễn dưới dạng danh sách kề trong file dske.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;

- N dòng kế tiếp mỗi dòng ghi lại danh sách kề của đỉnh tương ứng. Hai đỉnh trong cùng một danh sách kề được phân biệt với nhau bằng một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy viết chương trình chuyển đổi biểu diễn đồ thị G dưới dạng danh sách kề thành biểu diễn của đồ thị G dưới dạng ma trận kề và danh sách cạnh. Khuôn dạng biểu diễn đồ thị G dưới dạng ma trận kề, danh sách kề được ghi lại trong file mtke.out và dscanh.out theo khuôn dạng sau:

Khuôn dạng file mtke.out:

- Dòng đầu tiên ghi lại số tự nhiên n là số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được ghi cách nhau bởi một vài ký tự trống.

Khuôn dạng file dscanh.out

- Dòng đầu tiên ghi lại số tự nhiên N và M tương ứng với số đỉnh và số cạnh của đồ thị, hai số được ghi cách nhau bởi một vài ký tự trống;
- M dòng kế tiếp mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ đồ thị gồm 5 đỉnh, 5 cạnh được biểu diễn trong file dske.in như dưới đây sẽ cho ta các file mtke.out và dscanh.out tương ứng.

dske.in	***mtke.out***	***dscanh.out***
5 *2* *3 5* *1 5* *5* *4*	*5* *0 1 0 0 0* *0 0 1 0 1* *1 0 0 0 1* *0 0 1 0 0* *0 0 0 1 0*	*5 7* *1 2* *2 3* *2 5* *3 1* *3 5* *4 3* *5 4*

5.2. Kỹ thuật DFS

BÀI 5.2.1:

Viết chương trình tìm kiếm trong đồ thị với thuật toán tìm kiếm theo chiều sâu đệ quy.

BÀI 5.2.2:

Viết chương trình tìm kiếm trong đồ thị với thuật toán tìm kiếm theo chiều sâu không đệ quy.

BÀI 5.2.3:

Giả sử thuật toán tìm kiếm ưu tiên độ sâu DFS được áp dụng vào cây tìm kiếm nhị phân và cạnh bên phải được lấy trước khi rời mỗi nút. Các nút được viếng thăm theo thứ tự như thế nào?

BÀI 5.2.4:

Hãy vẽ ra rừng tìm kiếm ưu tiên độ sâu có được với đồ thị ví dụ bất kỳ khi thủ tục DFS quét qua các đỉnh theo thứ tự đảo ngược (từ V tới 1) cho cả hai cách biểu diễn.

BÀI 5.2.5:

Thủ tục visit được gọi chính xác bao nhiêu lần trong tìm kiếm DFS trên một đồ thị vô hướng (đếm theo số đỉnh V, số cạnh E, số thành phần liên thông C)?

BÀI 5.2.6:

Cho một đồ thị bất kỳ. Viết thủ tục thực hiện xóa một cạnh và thêm một cạnh bất kỳ. Hãy tìm các điểm bản lề và các thành phần liên thông của đồ thị mới này sử dụng thuật toán DFS.

BÀI 5.2.7:

Vẽ cây DFS cho một đồ thị bất kỳ.

BÀI 5.2.8:

Trong đồ thị song liên có V đỉnh, số cạnh nhỏ nhất là bao nhiêu?

BÀI 5.2.9:

Viết chương trình in ra các thành phần song liên của một đồ thị.

BÀI 5.2.10:

Viết chương trình sinh một đồ thị liên thông ngẫu nhiên có V đỉnh bằng cách sinh ra các cặp số nguyên từ 1..V. Ước lượng số cạnh cần thiết để tạo ra một đồ thị liên thông theo V.

BÀI 5.2.11:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán DFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, M và u tương ứng với số đỉnh, số cạnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Ba số được viết cách nhau bởi một vài khoảng trống.
- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

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

cay.out

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

BÀI 5.2.12:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán DFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới danh sách kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp, mỗi dòng ghi lại danh sách kề của đỉnh tương ứng, hai đỉnh khác nhau của cùng một danh sách kề được ghi cách nhau bởi một vài ký tự trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, M tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- M dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống. Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5    1
2    3    4    5

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

cay.out

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

5.3. Kỹ thuật BFS

Hãy giải các bài tập lập trình phần 5.2 sử dụng kỹ thuật BFS.

BÀI 5.3.1: *TRUYỀN TIN*

Một lớp gồm N học sinh, mỗi học sinh cho biết những bạn mà học sinh đó có thể liên lạc được (chú ý liên lạc này là liên lạc một chiều : u có thể gửi tin tới v nhưng v thì chưa chắc đã có thể gửi tin tới u).

Thầy chủ nhiệm đang có một thông tin rất quan trọng cần thông báo tới tất cả các học sinh. Để tiết kiệm thời gian, thầy chỉ nhắn tin tới 1 số học sinh rồi sau đó nhờ các học sinh này nhắn lại cho tất cả các bạn mà các học sinh đó có thể liên lạc được, và cứ lần lượt như thế làm sao cho tất cả các học sinh trong lớp đều nhận được tin .

Hãy tìm một số ít nhất các học sinh mà thầy chủ nhiệm cần nhắn.

Input

- Dòng đầu là N, M (N <= 800, M là số lượng liên lạc 1 chiều)
- Một số dòng tiếp theo mỗi dòng gồm 2 số u , v cho biết học sinh u có thể gửi tin tới học sinh v

Output

- Gồm 1 dòng ghi số học sinh cần thầy nhắn tin.

Ví dụ

Input:

12 15

1 3

3 6

6 1

6 8

8 12

12 9

9 6

2 4

4 5

5 2

4 6

7 10

10 11

11 7

10 9

Output:

2

Chọn các học sinh 7 và 2.

BÀI 5.3.2:

Cho đồ thị vô hướng G =<V,E>, trong đó V là tập đ ỉnh, E là tập cạnh. Ta gọi cạnh e thuộc E là cầu nếu khi loại bỏ cạnh sẽ làm tăng số thành phần liên thông của đồ thị. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp. Hãy viết chương trình tìm tất cả các cạnh cầu của đồ thị. Dữ liệu vào cho bởi file Dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên n là số đỉnh của đồ thị.
- n dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau bởi một hoặc vài kí tự trống.

Kết quả ra ghi lại trong file Canhcau.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các cạnh cầu mà bạn tìm được.
- Những dòng kế tiếp mỗi dòng ghi lại một cạnh cầu, đỉnh đầu và đỉnh cuối của cạnh được viết cách nhau bởi một vài kí tự trống.

Ví dụ dưới đây sẽ minh họa cho file Dothi.in và Canhcau.out:

dothi.in	canhcau.out
5 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 0	4 1 2 1 3 2 4 3 5

BÀI 5.3.3:

Cho đồ thị vô hướng G =<V,E>, trong đó V là tập đỉnh, E là tập cạnh. Ta gọi đỉnh v thuộc V là "*trụ*" nếu khi loại bỏ đỉnh u cùng các cạnh nối với u sẽ làm tăng số thành phần liên thông của đồ thị. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp, viết chương trình tìm tất cả đỉnh "trụ" của đồ thị. Dữ liệu vào cho bởi file Dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên n là số đỉnh của đồ thị.
- n dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau bởi một hoặc vài kí tự trống.

Kết quả ra ghi lại trong file Canhcau.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số đỉnh trụ mà bạn tìm được.
- Dòng kế tiếp ghi lại các đỉnh trụ tìm được, hai đỉnh trụ khác nhau được viết cách nhau bởi một vài kí tự trống.

Ví dụ dưới đây sẽ minh họa cho file Dothi.in và Canhcau.out:

dothi.in	dinhtru.out

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

BÀI 5.3.4: *BÃI CỎ NGON NHẤT*

Bessie dự định cả ngày sẽ nhai cỏ xuân và ngắm nhìn cảnh xuân trên cánh đồng của nông dân John, cánh đồng này được chia thành các ô vuông nhỏ với R (1 <= R <= 100) hàng và C (1 <= C <= 100) cột.

Bessie ước gì có thể đếm được số khóm cỏ trên cánh đồng. Mỗi khóm cỏ trên bản đồ được đánh dấu bằng một ký tự '#' hoặc là 2 ký tự '#' nằm kề nhau (trên đường chéo thì không phải). Cho bản đồ của cánh đồng, hãy nói cho Bessie biết có bao nhiêu khóm cỏ trên cánh đồng.

Ví dụ như cánh đồng dưới dây với R=5 và C=6:

.#....

..#...

..#..#

...##.

.#....

Cánh đồng này có 5 khóm cỏ: một khóm ở hàng đầu tiên, một khóm tạo bởi hàng thứ 2 và thứ 3 ở cột thứ 2, một khóm là 1 ký tự nằm riêng rẽ ở hàng 3, một khóm tạo bởi cột thứ 4 và thứ 5 ở hàng 4, và một khóm cuối cùng ở hàng 5.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: R và C
- Dòng 2..R+1: Dòng i+1 mô tả hàng i của cánh đồng với C ký tự, các ký tự là '#' hoặc '.' .

Kết quả

- Dòng 1: Một số nguyên cho biết số lượng khóm cỏ trên cánh đồng.

Ví dụ

Dữ liệu

5 6

.#....

..#...

..#..#

...##.

.#....

Kết quả

5

BÀI 5.3.5: *PHƯƠNG ÁN BẮN PHÁO*

Một hệ thống phòng thủ của địch gồm N điểm (N<=100), giữa các điểm bất kỳ của hệ thống đều có thể đi lại trực tiếp hoặc gián tiếp với nhau thông qua hệ thống các đường hầm. Bài toán được đặt ra là cho trước một hệ thống phòng thủ, hãy giúp bộ đội sử dụng đúng 1 quả pháo bắn vào một trong N điểm sao cho hệ thống bị chia cắt thành nhiều mảnh nhất.

Input:

Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N không lớn hơn 100 là số điểm của hệ thống phòng thủ.
- Những dòng kế tiếp ghi lại ma trận $G = (g_{ij})$ biểu diễn hệ thống phòng thủ, trong đó $g_{ij}=0$ được hiểu là không có đường hầm trực tiếp nối giữa điểm i và j, $g_{ij}=1$ được hiểu có đường hầm nối trực tiếp giữa điểm i và điểm j (1<=i, j<=N).

Output:

Với mỗi bộ test, in ra màn hình trên một dòng một số duy nhất là đỉnh bị phá hủy thỏa mãn yêu cầu bài toán (nếu có nhiều đỉnh cùng thỏa mãn yêu cầu thì in ra đỉnh có giá trị nhỏ nhất). Nếu không thể chia cắt được hệ thống, hãy in ra số 0.

Example

Input:

```
2
5
0 1 1 0 0
1 0 1 0 0
1 1 0 1 1
0 0 1 0 0
0 0 1 0 0
5
0 1 1 0 0
1 0 1 0 1
1 1 0 1 1
0 0 1 0 1
0 1 1 1 0
```

Output:

```
3
0
```


BÀI 5.3.6: *CHÚ BÒ HƯ HỎNG*

Nông dân John có N (1<=N<=250) con bò đánh số từ 1..N chơi trên bãi cỏ.

Để tránh bị lạc mất các con bò, mỗi con bò có thể được nối với một số con bò khác bằng dây thừng.

Có tất cả M (1 <= M <= N*(N-1)/2) dây thừng nối các con bò. Tất nhiên, không có 2 con bò mà có nhiều

hơn 1 dây thừng nối giữa chúng. Dữ liệu cho biết mỗi cặp con bò c1 và c2 là nối với nhau (1 <= c1 <= N; 1 <= c2 <= N; c1 != c2).

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

Input:

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

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

Output:

* Nếu không có con bò hư hỏng, in ra 0.

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

Ví dụ:

Input:

6 4

1 3

2 3

1 2

4 5

Output:

4

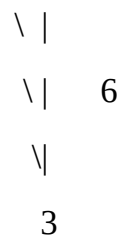
5

6

Giải thích:

Input có thể mô tả như hình dưới đây:

1---2 4---5



Dễ thấy, các con bò 4, 5, và 6 không nối tới con bò 1.

BÀI 5.3.7:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng ma trận kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5
0     1     1
      1     1
1     0     1
      0     1
1     1     0
      1     0
1     0     1
      0     1
1     1     0
      1     0
```

cay.out

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

BÀI 5.3.8:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, M và u tương ứng với số đỉnh, số cạnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Ba số được viết cách nhau bởi một vài khoảng trống.
- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

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

cay.out

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


BÀI 5.3.9:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới danh sách kề theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N, u tương ứng với số đỉnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp, mỗi dòng ghi lại danh sách kề của đỉnh tương ứng, hai đỉnh khác nhau của cùng một danh sách kề được ghi cách nhau bởi một vài ký tự trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, M tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- M dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng.

dothi.in

```
5    1
2    3    4    5

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

cay.out

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

BÀI 5.3.10:

Cho đồ thị có hướng G =<V,E> gồm N đỉnh và M cạnh được biểu diễn dưới dạng danh sách kề trong file dske.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;
- N dòng kế tiếp mỗi dòng ghi lại danh sách kề của đỉnh tương ứng. Hai đỉnh trong cùng một danh sách kề được phân biệt với nhau bằng một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy viết chương trình kiểm tra và đưa ra thông báo:

a) "*Đồ thị liên thông mạnh*" nếu G liên thông mạnh;
b) "*Đồ thị liên thông yếu*" nếu G không liên và G liên thông yếu;
c) "*Đồ thị không liên thông mạnh, không liên thông yếu*" trong những trường hợp còn lại.

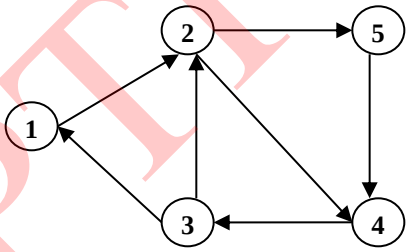
Ví dụ với đồ thị được biểu diễn dưới dạng danh sách kề dưới, kết quả thực hiện của chương trình là " *Đồ thị liên thông mạnh*".

dske.in

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

BÀI 5.3.11:

Cho hệ thống giao thông gồm N điểm. Biết giữa hai điểm bất kỳ của hệ thống đều tồn tại đường đi trực tiếp hoặc gián tiếp thông qua một số điểm trung gian. Ta gọi điểm giao thông *s* là điểm *thắt* của cặp nút giao thông *u, v* nếu mọi đường đi từ *s* đến *t* đều phải đi qua *s*. Ví dụ với cặp nút *1, 3* của hệ thống giao thông gồm *5* điểm dưới đây sẽ có đỉnh *thắt s = 2* và *s =4*.



Nhiệm vụ của bạn là viết một chương trình tìm số lượng các đỉnh *thắt s* của cặp điểm *u, v* của hệ thống giao thông.

Dữ liệu vào:

Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu tiên chứa một số nguyên không lớn hơn 100 là số lượng các bộ dữ liệu. Các dòng tiếp theo chứa các bộ dữ liệu. Mỗi bộ dữ liệu gồm một nhóm dòng theo khuôn dạng:

- Dòng 1 chứa 4 số nguyên N, M, u, v (u, v, N≤100; M≤1000).
- M dòng sau, mỗi dòng ghi lại đường nối trực tiếp giữa các cặp điểm (i, j) của hệ thống giao thông (1≤i,j≤N).

Dữ liệu ra:

Với mỗi bộ dữ liệu, ghi ra trên một dòng một số là số lượng đỉnh thắt của mỗi cặp điểm u, v trong mỗi test.

Ví dụ dữ liệu vào	**Ví dụ dữ liệu ra**
2 5 7 1 3 1 2 2 4 2 5 3 1 3 2 4 3 5 4 4 5 1 4 1 2 1 3 2 3 2 4 3 4	2 0

BÀI 5.3.12:

Cho một hệ thống giao thông gồm N điểm (N≤250). Biết giữa hai điểm bất kỳ của hệ thống đều có đường đi trực tiếp hoặc gián tiếp thông qua các điểm trung gian. Hãy cho biết ta có thể định chiều lại toàn bộ hệ thống giao thông sao cho giữa hai điểm bất kỳ của hệ thống đều tồn tại đường đi và bằng một chiều hay không?

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức như sau:

- Dòng thứ nhất ghi lại số N là số điểm node giao thông của hệ thống;
- N dòng tiếp theo ghi lại ma trận vuông A = (a_{ij}) cấp N là biểu diễn của hệ thống giao thông. Trong đó, a_{ij} = 0 nếu không có đường đi trực tiếp từ điểm i đến j, a_{ij} = 1 nếu có đường đi trực tiếp từ điểm i đến j.

Output: Với mỗi bộ test, viết ra trên một dòng giá trị "YES" nếu hệ thống giao thông có thể định chiều được, giá trị "NO" nếu hệ thống giao thông không thể định chiều được.

Ví dụ cho Input và Output:

INPUT	OUTPUT
2 **5** **0 1 1 0 0** **1 0 1 0 1** **1 1 0 1 0** **0 0 1 0 0** **0 1 0 0 0** **5** **0 1 1 0 0** **1 0 1 1 1** **1 1 0 1 0** **0 1 1 0 1** **0 1 0 1 0**	**NO** **YES**

BÀI 5.3.13:

Cho hệ thống giao thông gồm N node ($1 \le N \le 100$) được tổ chức trong file DATA.IN theo khuôn dạng sau: dòng đầu tiên ghi lại số tự nhiên N là số node của hệ thống; N dòng kế tiếp ghi lại ma trận vuông A_{ij} ($1 \le i,j \le N$) là biểu diễn của hệ thống giao thông, trong đó A_{ij} =1 biểu thị hệ thống có đường đi trực tiếp từ node i đến node j, A_{ij}=0 biểu thị hệ thống không có đường đi trực tiếp từ node i đến node j.

Biết giữa hai node bất kỳ của hệ thống đều có đường đi trực tiếp hoặc gián tiếp thông qua một số node trung gian. Để khắc phục tình trạng tắc nghẽn giao thông, nhà chức trách muốn định chiều lại toàn bộ hệ thống giao thông sao cho những điều kiện sau được thỏa mãn:

a. Không xây dựng thêm mới bất kỳ một tuyến đường nào (bảo toàn các tuyến đường cũ);
b. Tất cả các tuyến đường đi từ node i đến node j bất kỳ của hệ thống chỉ đi bằng một chiều.

Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình định chiều lại toàn bộ hệ thống giao thông thỏa mãn những điều kiện trên. Ghi lại kết quả định chiều các tuyến đường trong file KETQUA.OUT theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên M là số các đường nối một chiều trực tiếp từ node i đến node j của hệ thống hoặc ghi lại thông báo "Vô nghiệm" trong trường hợp hệ thống không thể định chiều được;
- M dòng kế tiếp mỗi dòng ghi lại hướng của mỗi tuyến đường trực tiếp (trong trường hợp bài toán định chiều thành công).

Ví dụ hệ thống gồm 4 node được biểu diễn trong file DATA.IN dưới đây sẽ cho ta kết quả trong file KETQUA.OUT như sau:

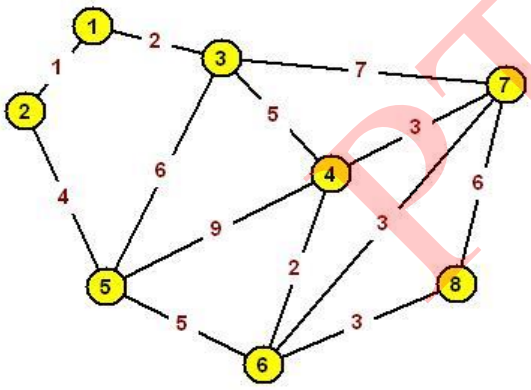
DATA.IN			
4			
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

KETQUA.OUT	
6	
1	2
2	3
3	4
3	1
4	1

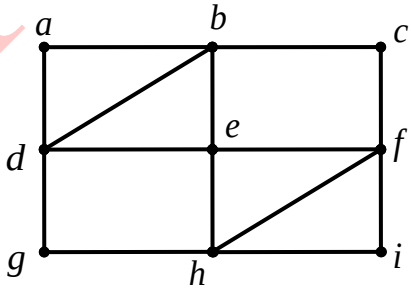
5.4. Đồ thị Euler

BÀI 5.4.1:

Cho các đồ thị sau. Đồ thị nào là đồ thị nửa Euler, đồ thị Euler? Tại sao? Chỉ ra một đường đi, chu trình Euler nếu có.



Hình 1



Hình 2

BÀI 5.4.2:

Cho đồ thị vô hướng liên thông G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị nửa Euler hay không? Nếu G là đồ thị nửa Euler hãy xây dựng một đường đi Euler của đồ thị, ngược lại đưa ra thông báo "G không là đồ thị nửa Euler"? Các kết quả xuất ra màn hình.

Ví dụ với đồ thị dưới đây sẽ cho ta đường đi Euler : 2 - 1 - 3 - 2 - 4 - 3

dothi.in	**Màn hình**
4 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0	2 - 3 - 4 - 1 - 2 – 4

BÀI 5.4.3:

Cho đồ thị có hướng liên thông yếu G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị nửa Euler hay không? Nếu G là đồ thị nửa Euler hãy xây dựng một đường đi Euler của đồ thị, ngược lại đưa ra thông báo "G không là đồ thị nửa Euler"?. Các kết quả xuất ra màn hình.

Ví dụ với đồ thị dưới đây sẽ cho ta đường đi Euler : 2 - 3 - 4 - 1 - 2 – 4:

dothi.in	**Màn hình**
4	2 - 3 - 4 - 1 - 2 – 4

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

BÀI 5.4.4:

Cho ma trận vuông $A = (a_{ij})$ cấp N ($N \le 100$). Các các phần tử trên đường chéo chính của ma trận $A = (a_{ij})$ luôn có giá trị 0 ($a_{ii} = 0$), các phần tử còn lại có giá trị hoặc bằng 0 hoặc bằng 1 ($1 \le i, j \le N$). Ta nói ma trận $A = (a_{ij})$ là biểu diễn của đồ thị Euler nếu và chỉ nếu $A = (a_{ij})$ thỏa mãn tính chất (*a*); ma trận $A = (a_{ij})$ là biểu diễn của đồ thị nửa Euler nếu và chỉ nếu $A = (a_{ij})$ thỏa mãn các tính chất (*b*) dưới đây.

Tính chất (*a*): Ma trận $A=(a_{ij})$ là ma trận đối xứng ($a_{ij} = a_{ji}$) và tổng các phần tử trên mỗi hàng của ma trận là những số chẵn.

Tính chất (*b*): Ma trận $A=(a_{ij})$ là ma trận đối xứng ($a_{ij} = a_{ji}$) và tồn tại đúng hai hàng *u*, *v* sao cho tổng các phần tử của ma trận trên hàng *u* và tổng các phần tử của ma trận trên hàng *v* là những số lẻ.

Bài toán đặt ra là, cho ma trận vuông $A = (a_{ij})$ cấp N như trên, hãy đưa ra giá trị *1* nếu $A = (a_{ij})$ thỏa mãn tính chất (*a*), giá trị *-1* nếu $A = (a_{ij})$ thỏa mãn tính chất (*b*), giá trị *0* trong các trường hợp khác.

Input:

Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N là cấp của ma trận vuông;
- Những dòng kế tiếp ghi lại ma trận vuông $A = (a_{ij})$. Hai phần tử khác nhau của ma trận vuông được ghi cách nhau một vài khoảng trống.

Output:

Với mỗi bộ test, viết ra trên một dòng giá trị 1, -1 hoặc 0 theo yêu cầu bài toán.

Ví dụ cho Input và Output:

INPUT	OUTPUT
3 5 0 1 1 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 5 0 1 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 5 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 0	1 -1 0

BÀI 5.4.5:

Cho đồ thị có hướng liên thông yếu G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị nửa Euler hay không? Nếu G là đồ thị nửa Euler hãy xây dựng một đường đi Euler của đồ thị, ngược lại đưa ra thông báo "G không là đồ thị nửa Euler"?

Ví dụ với đồ thị dưới đây sẽ cho ta đường đi Euler : 2 - 3 - 4 - 1 - 2 - 4

dothi.in

```
4
0       1
        0
        0
0       0
        1
        1
0       0
        0
        1
1       0
        0
        0
```

BÀI 5.4.6:

Cho đồ thị vô hướng liên thông G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị nửa Euler hay không? Nếu G là đồ thị nửa Euler hãy xây dựng một đường đi Euler của đồ thị, ngược lại đưa ra thông báo "G không là đồ thị nửa Euler"?

BÀI 5.4.7:

Cho đồ thị vô hướng liên thông G = <V,E> gồm N đỉnh được biểu diễn dưới dạng danh sách kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp, mỗi dòng ghi lại danh sách kề của đỉnh tương ứng, hai đỉnh khác nhau của cùng một danh sách kề được ghi cách nhau bởi một vài ký tự trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị Euler hay không? Nếu G là đồ thị Euler, hãy xây dựng một chu trình Euler của đồ thị bắt đầu tại đỉnh u (u được nhập từ bàn phím), ngược lại đưa ra thông báo "G không là đồ thị Euler"?

Ví dụ với đồ thị dưới đây sẽ cho ta chu trình Euler bắt đầu tại đỉnh số 1 là : 1 - 2 - 3 - 4 - 1

dothi.in
```
5
2   4
1   3
2   4
1   3
```

BÀI 5.4.8:

Cho đồ thị có hướng liên thông yếu G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị Euler hay không? Nếu G là đồ thị Euler hãy xây dựng một chu trình Euler của đồ thị bắt đầu tại đỉnh u (u được nhập từ bàn phím), ngược lại đưa ra thông báo "G không là đồ thị Euler"?

Ví dụ với đồ thị dưới đây sẽ cho ta chu trình Euler bắt đầu tại đỉnh số 1 là : 1 - 2 - 3 - 4 - 1

dothi.in
```
5
0   1
    0
    0
0   0
    1
    0
0   0
    0
    1
1   0
    0
    0
```


BÀI 5.4.9:

Cho đồ thị vô hướng liên thông G = <V,E> gồm N đỉnh được biểu diễn dưới dạng ma trận kề trong file dothi.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N tương ứng với số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị, hai phần tử khác nhau của ma trận kề được viết cách nhau một vài khoảng trống.

Hãy viết chương trình kiểm tra G có phải là đồ thị Euler hay không? Nếu G là đồ thị Euler hãy xây dựng một chu trình Euler của đồ thị bắt đầu tại đỉnh u (u được nhập từ bàn phím), ngược lại đưa ra thông báo "G không là đồ thị Euler"?

Ví dụ với đồ thị dưới đây sẽ cho ta chu trình Euler bắt đầu tại đỉnh số 1 là : 1 - 2 - 3 - 4 - 1

dothi.in

```
4
0   1
    0
    1
1   0
    1
    0
0   1
    0
    1
1   0
    1
    0
```

BÀI 5.4.10:

An đưa cho Bình một tập bản vẽ trên mặt phẳng. Mỗi bản vẽ chỉ bao gồm các điểm và các đoạn thẳng nối giữa các điểm. An đố Bình, hãy ghi lại "*Yes*" trên các bản vẽ mà Bình có thể tô lại tất cả các đoạn thẳng trên bản vẽ đó bằng một nét bút, hãy ghi lại "*No*" nếu Bình không làm được điều đó.

Yêu cầu: Hãy cho biết bằng cách nào Bình có thể ghi lại các giá trị "Yes", "*No*" trên mỗi bản vẽ với thời gian nhanh nhất.

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức như sau:

- Dòng đầu tiên ghi lại hai số N, M là số điểm trên bản vẽ và số đoạn thẳng nối giữa các điểm;
- M dòng kế tiếp mỗi dòng ghi lại một đoạn thẳng nối giữa điểm *i* và điểm *j*.

Output: Với mỗi bộ test, output đưa ra giá trị "Yes" hoặc "*No*" trên mỗi dòng theo yêu cầu của bài toán.

Ví dụ cho Input và Output:

INPUT	OUTPUT
2 ***5 6*** 1 2 1 3 2 3 2 4 3 4 4 5 ***5 5*** 1 2 1 3 2 4 3 4 4 5	No **Yes**

BÀI 5.4.11:

Token Ring là một trong những phương pháp khá phổ biến trong việc kiểm tra tín hiệu các kênh truyền tin trong các mạng máy tính. Token Ring có thể được mô tả như sau.

Cho một mạng gồm N máy tính. Giữa hai máy tính bất kỳ có thể nối với một hoặc nhiều kênh truyền tin khác nhau. Token Ring là một tín hiệu xuất phát từ một máy tính bất kỳ đi qua tất cả các kênh truyền tin nối giữa các máy tính khác nhau rồi trở về chính nó.

Yêu cầu: Cho một mạng máy tính bất kỳ. Hãy cho biết có thể tạo nên một Token Ring từ một máy tính bất kỳ đi qua tất cả các kênh truyền, mỗi kênh đúng một lần rồi trở về chính nó hay không?

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức như sau:

- Dòng thứ nhất ghi lại số N, M là số máy tính và số kênh truyền sử dụng trong mạng.
- M dòng tiếp theo, mỗi dòng ghi lại một kênh truyền trực tiếp giữa hai máy tính.

Output: Với mỗi bộ test, output đưa ra một giá trị duy nhất "YES" nếu có thể tạo ra một Token Ring theo yêu cầu của bài toán, "NO" trong trường hợp ngược lại.

Ví dụ cho Input và Output:

INPUT	OUTPUT
2 4 4 **1 2** **1 4** **2 3** **3 4** 4 5 **1 2** **1 3** **1 4** **2 3** **3 4**	**YES** **NO**

BÀI 5.4.12:

Một từ được hiểu là dãy các ký tự không chứa khoảng trống, dấu tab, dấu xuống dòng và dấu về đầu dòng. Ta nói, từ X có thể nối được với từ Y nếu ký tự đầu tiên của từ X trùng với ký tự cuối cùng của từ Y (Ví dụ từ X = ”ABC” có thể nối được với từ Y =”EFA”).

Yêu cầu: Cho xâu ký tự S có không quá 80 từ (các từ trong S có thể lặp lại). Hãy cho biết có thể nối tất cả các từ trong S để tạo nên một từ mới theo nguyên tắc nêu trên hay không?

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test là một xâu ký tự S được viết trên một dòng..

Output: Viết ra trên mỗi dòng giá trị “YES” hoặc “NO” nếu ta có thể thực hiện được hoặc không thực hiện được yêu cầu đặt ra của bài toán.

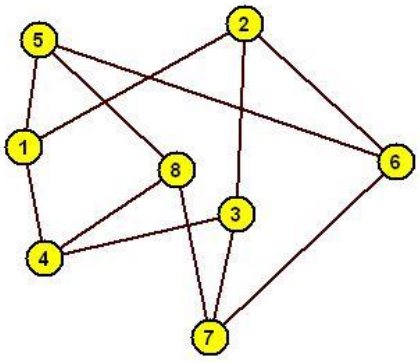
Ví dụ cho Input và Output:

INPUT	OUTPUT
2 **ABC DEA ABC EAD DEA EAD** **ABC ABD ABF ABG ADH ADA**	**YES** **NO**

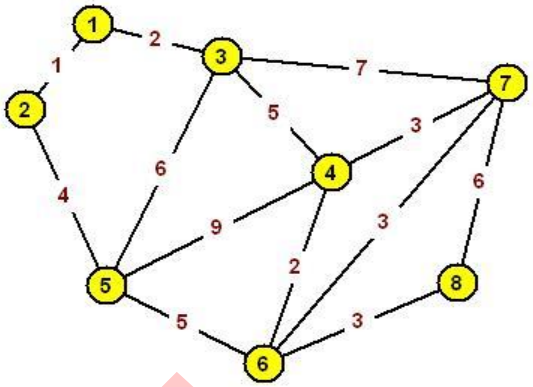
5.5. Đồ thị Hamilton

BÀI 5.5.1:

Tìm một chu trình Hamilton của đồ thị cho ở hình dưới.



Hình 1



Hình 2

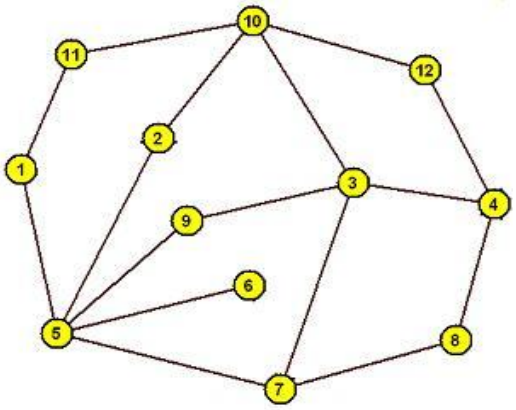
BÀI 5.5.2:

Chứng minh rằng: Nếu đồ thị lưỡng phân G = (V_1, V_2, E) có chu trình Hamilton thì $|V_1| = |V_2|$

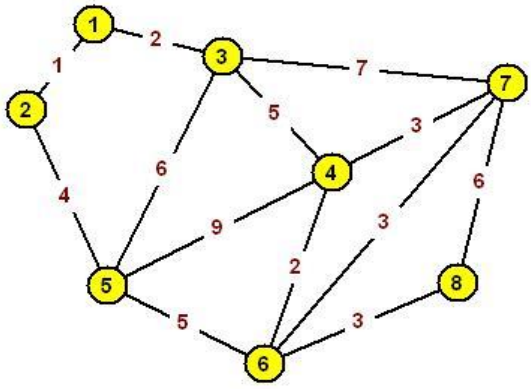
5.6. Cây khung đồ thị

BÀI 5.6.1:

Xây dựng cây khung của đồ thị sau theo thuật toán tìm kiếm theo chiều rộng và chiều sâu xuất phát từ đỉnh 1 (trong danh sách các đỉnh kề của một đỉnh luôn ưu tiên chọn đỉnh có số thứ tự nhỏ)



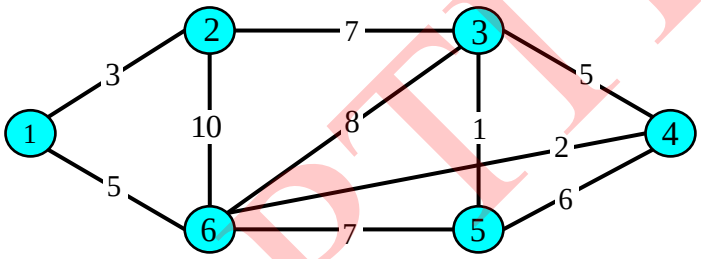
Hình 1



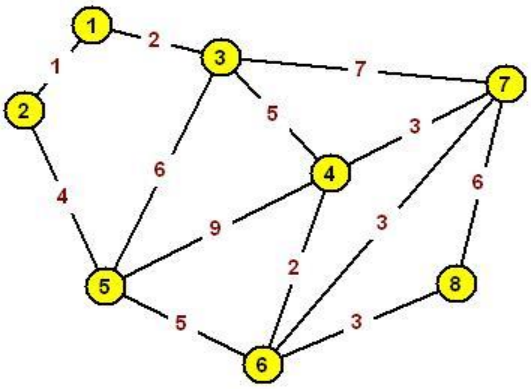
Hình 2

BÀI 5.6.2:

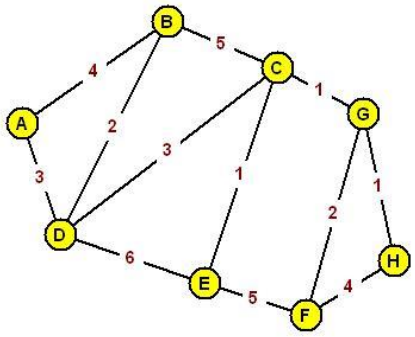
Tìm cây khung nhỏ nhất cho các đồ thị sau bằng hai phương pháp PRIM, KRUSKAL



Hình 1



Hình 2



Hình 3

BÀI 5.6.3:

Tìm một cây bao trùm tối thiểu của một đồ thị.

BÀI 5.6.4:

Hãy đưa ra thuật toán tìm cây bao trùm tối thiểu của một đồ thị liên thông.

BÀI 5.6.5:

Có đồi thị nào V đỉnh và E cạnh mà cần thời gian xấp xỉ (E+V)logV khi tìm cây bao trùm tối thiểu bằng phương pháp tìm kiếm theo độ sâu ưu tiên hay không?

BÀI 5.6.6:

Tìm một phản ví dụ để cho thấy chiến lược "tham lam" sau đây sẽ không đúng khi giải bài toán cây bao trùm tối thiểu hay đường đi ngắn nhất: "trong mỗi bước, thăm đỉnh chưa được thăm gần đỉnh vừa thăm nhất".

BÀI 5.6.7:

Làm thế nào để tìm cây bao trùm tối thiểu của đồ thị "khổng lồ" (quá lớn đến nỗi không thể chứa nó trong bộ nhớ chính).

BÀI 5.6.8:

Viết chương trình để phát sinh các đồ thị liên thông ngẫu nhiên V đỉnh, kế đó tìm đường đi ngắn nhất và cây bao trùm tối thiểu. Hãy dùng trọng số ngẫu nhiên từ 1 đến V. So sánh trọng số các cây có được tùy theo các giá trị của V.

BÀI 5.6.9:

Viết chương trình để phát sinh đồ thị đầy đủ có trọng số ngẫu nhiên gồm V đỉnh bằng cách đặt các giá trị trong ma trận kề bởi các số ngẫu nhiên từ 1 đến V. Chạy chương trình để so sánh thuật toán tìm cây bao trùm tối thiểu của Kruskal và Prim khi V = 10, 25, 100.

BÀI 5.6.10:

Cho một phản ví dụ để thấy thuật toán tìm cây bao trùm tối thiểu sau đây sẽ không đúng: " Sắp xếp các điểm theo hoành độ, tìm cây bao trùm tối thiểu cho một nửa đầu tiên và một nửa thứ hai, kế đến tìm cạnh ngắn nhất nối chúng với nhau".

BÀI 5.6.11:

Cho đồ thị vô hướng liên thông gồm N đỉnh G = <V,E>. Sử dụng thuật toán BFS, hãy viết chương trình xây dựng một cây khung của đồ thị bắt đầu tại đỉnh u. Dữ liệu vào cho bởi file dothi.in là biểu diễn của đồ thị dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số tự nhiên N, M và u tương ứng với số đỉnh, số cạnh của đồ thị và đỉnh bắt đầu xây dựng cây khung. Ba số được viết cách nhau bởi một vài khoảng trống.
- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị, đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Cây khung xây dựng từ đỉnh u tìm được ghi lại trong file cay.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N, K tương ứng với số đỉnh và số cạnh của cây khung. Hai số được viết cách nhau một vài ký tự trống;
- K dòng kế tiếp ghi lại một cạnh của cây khung, đỉnh đầu và đỉnh cuối của mỗi cạnh được ghi cách nhau bởi một vài ký tự trống.

Ví dụ với đồ thị G=<V,E> được tổ chức trong file dothi.in dưới đây sẽ cho ta file cay.out tương ứng:

dothi.in	**cay.out**
5 8 1 1 2 1 3 1 4 1 5 2 3 2 5 3 4 4 5	5 4 1 2 1 3 1 4 1 5

BÀI 5.6.12:

Trên một nền phẳng với hệ tọa độ trực chuẩn đặt n máy tính, máy tính thứ i được đặt ở tọa độ (xi, yi). Đã có sẵn một số dây cáp mạng nối giữa một số cặp máy tính. Cho phép nối thêm các dây cáp mạng nối giữa từng cặp máy tính. Chi phí nối một dây cáp mạng tỉ lệ thuận với khoảng cách giữa hai máy cần nối. Hãy tìm cách nối thêm các dây cáp mạng để cho các máy tính trong toàn mạng là liên thông và chi phí nối mạng là nhỏ nhất.

BÀI 5.6.13:

Hệ thống điện trong thành phố được cho bởi n trạm biến thế và các đường dây điện nối giữa các trạm biến thế. Mỗi đường dây điện e có độ an toàn là p(e) ϵ (0,1]. Độ an toàn của cả lưới điện là tích độ an toàn trên các đường dây. Hãy tìm cách bỏ đi một số dây điện để cho các trạm biến thế vẫn liên thông và độ an toàn của mạng là lớn nhất có thể.

Gợi ý: bằng kỹ thuật lấy lô ga rít, độ an toàn trên lưới điện trở thành tổng độ an toàn trên các đường dây.

BÀI 5.6.14:

Cho đồ thị vô hướng liên thông có trọng số G = <V,E> trong file dothi.in được biểu diễn dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N, M tương ứng với số đỉnh và số cạnh của đồ thị.
- M dòng kế tiếp mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số của cạnh tương ứng.

BÀI 5.6.15:

Hãy sử dụng thuật toán Prim, viết chương trình tìm cây khung nhỏ nhất của đồ thị bắt đầu tại đỉnh u=1. Cây khung nhỏ nhất tìm được ghi lại trong file caykhung.out theo khuôn dạng:

- Dòng đầu tiên ghi lại độ dài cây khung nhỏ nhất;
- Những dòng kế tiếp, mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số cạnh tương ứng của cây khung.

Ví dụ dưới đây sẽ minh họa cho file dothi.in và caykhung.out của đồ thị.

dothi.out	**ketqua.out**
5 1 2 2 1 3 4 1 4 6 1 5 8 2 3 7 2 5 5 3 4 3 4 5 1	10 1 2 1 3 3 4 4 5

BÀI 5.6.16:

Cho đồ thị vô hướng liên thông có trọng số G = <V,E> trong file dothi.in được biểu diễn dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N, M tương ứng với số đỉnh và số cạnh của đồ thị.
- M dòng kế tiếp mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số của cạnh tương ứng.

Hãy sử dụng thuật toán Prim, viết chương trình tìm cây khung nhỏ nhất của đồ thị bắt đầu tại đỉnh u=1. Cây khung nhỏ nhất tìm được ghi lại trong file caykhung.out theo khuôn dạng:

- Dòng đầu tiên ghi lại độ dài cây khung nhỏ nhất;

- Những dòng kế tiếp, mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số cạnh tương ứng của cây khung.

Ví dụ dưới đây sẽ minh họa cho file dothi.in và caykhung.out của đồ thị.

dothi.in

```
5
1   2   2
1   3   4
1   4   6
1   5   8
2   3   7
2   5   5
3   4   3
4   5   1
```

ketqua.out

```
10
1   2
1   3
3   4
4   5
```

BÀI 5.6.17:

Cho đồ thị vô hướng liên thông có trọng số G = <V,E> trong file dothi.in được biểu diễn dưới dạng danh sách cạnh theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N, M tương ứng với số đỉnh và số cạnh của đồ thị.
- M dòng kế tiếp mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số của cạnh tương ứng.

Hãy sử dụng thuật toán Kruskal, viết chương trình tìm cây khung nhỏ nhất của đồ thị. Cây khung nhỏ nhất tìm được ghi lại trong file caykhung.out theo khuôn dạng:

- Dòng đầu tiên ghi lại độ dài cây khung nhỏ nhất;
- Những dòng kế tiếp, mỗi dòng ghi lại ba số i, j, w tương ứng với đỉnh đầu, đỉnh cuối và trọng số cạnh tương ứng của cây khung.

Ví dụ dưới đây sẽ minh họa cho file dothi.in và caykhung.out của đồ thị.

dothi.in

```
5
1   2   2
1   3   4
1   4   6
1   5   8
2   3   7
2   5   5
3   4   3
4   5   1
```

ketqua.out

```
10
1   2
1   3
3   4
4   5
```

BÀI 5.6.18:

Cho mạng gồm N máy tính. Biết giữa hai máy tính đều được nối với nhau bằng hệ thống cable trực tiếp hoặc gián tiếp thông qua một số máy tính trung gian. Để tiết kiệm cable nối, người ta nghĩ cách loại bỏ đi một số đường cable sao cho ta vẫn nhận được một mạng máy tính liên thông. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình bỏ các đường cable cho mạng máy tính sao cho hai điều kiện sau được thỏa mãn:

(i) Số các đường cable loại bỏ nhiều nhất có thể được;

(ii) Số các đường cable đi vào hoặc đi ra máy tính thứ K ($1 \le K \le N$) là ít nhất.

Dữ liệu vào cho bởi file mang.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N và K. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp ghi lại ma trận vuông A_{ij} (i, j = 1, 2, ..., N) là biểu diễn các tuyến cable nối. Trong đó, $A_{ij} = 1$ biểu thị từ máy tính thứ i và máy tính thứ j có đường cable nối trực tiếp; $A_{ij} = 0$ biểu thị từ máy tính thứ i và máy tính thứ j không có đường cable nối trực tiếp;

Mạng máy tính liên thông với tối thiểu các đường cable nối tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N là số máy tính của mạng và M và số các đường cable còn lại nối các máy tính;
- M dòng kế tiếp ghi lại mỗi đường cable nối trực tiếp từ máy tính i đến máy tính j. Giá trị i và j được viết cách nhau một vài khoảng trống.

Ví dụ với mạng máy tính được cho trong file mang.in sẽ cho ta file ketqua.out tương ứng.

mang.in

```
5 1
0 1 1 1 1
1 0 1 0 1
1 1 0 1 0
1 0 1 0 1
1 1 0 1 0
```

ketqua.out

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

BÀI 5.6.19:

Cho mạng gồm N máy tính. Biết giữa hai máy tính đều được nối với nhau bằng hệ thống cable trực tiếp hoặc gián tiếp thông qua một số máy tính trung gian. Để tiết kiệm cable nối, người ta nghĩ cách loại bỏ đi một số đường cable sao cho ta vẫn nhận được một mạng máy tính liên thông. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình bỏ các đường cable cho mạng máy tính sao cho hai điều kiện sau được thỏa mãn:

(iii) Số các đường cable loại bỏ nhiều nhất có thể được;

(iv) Số các đường cable đi vào hoặc đi ra máy tính thứ K ($1 \le K \le N$) là nhiều nhất.

Dữ liệu vào cho bởi file mang.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên N và K. Hai số được viết cách nhau bởi một vài khoảng trống.
- N dòng kế tiếp ghi lại ma trận vuông A_{ij} (i, j = 1, 2, ..., N) là biểu diễn các tuyến cable nối. Trong đó, $A_{ij} = 1$ biểu thị từ máy tính thứ i và máy tính thứ j có đường cable nối trực tiếp; $A_{ij} = 0$ biểu thị từ máy tính thứ i và máy tính thứ j không có đường cable nối trực tiếp;

Mạng máy tính liên thông với tối thiểu các đường cable nối tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số N là số máy tính của mạng và M và số các đường cable còn lại nối các máy tính;
- M dòng kế tiếp ghi lại mỗi đường cable nối trực tiếp từ máy tính i đến máy tính j. Giá trị i và j được viết cách nhau một vài khoảng trống.

Ví dụ với mạng máy tính được cho trong file mang.in sẽ cho ta file ketqua.out tương ứng.

mang.in

```
5 1
0 1 1 1 1
1 0 1 0 1
1 1 0 1 0
1 0 1 0 1
1 1 0 1 0
```

ketqua.out

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

BÀI 5.6.20:

Cho một mạng gồm N máy tính. Biết giữa hai máy tính bất kỳ đều có thể truy nhập trực tiếp hoặc gián tiếp với nhau thông qua hệ thống cable nối giữa các máy tính khác nhau. Biết độ dài đường cable nối giữa máy tính *i* và máy tính *j* là C[*i*][*j*] mét (*i, j =1, 2, .., N*). Để tiết kiệm đường cable nối giữa các máy tính, người ta xây dựng phương án loại bỏ đi một số đoạn cable nối sao cho giữa hai máy tính bất kỳ vẫn có thể truy nhập lẫn nhau.

Yêu cầu: Hãy cho biết số mét cable nối tối đa có thể tiết kiệm được sau khi thực hiện phương án tiết kiệm cable nối.

Input: Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test được tổ chức như sau:

- Dòng đầu tiên ghi lại N là số máy tính trong mạng và M là số các đoạn cable nối giữa những máy tính khác nhau.
- M dòng kế tiếp, mỗi dòng ghi lại bộ ba i, j, C[i][j] biểu diễn đường cable nối từ máy i đến máy j với độ dài C[i][j].

Output: Với mỗi bộ test, output đưa ra một số duy nhất là số mét cable tối đa có thể tiết kiệm được.

Ví dụ cho Input và Output:

INPUT	OUTPUT
2 5 6 **1 2 5** **1 3 4** **2 3 11** **2 4 7** **3 5 9** **4 5 17** 5 4 **1 2 5** **1 3 4** **2 4 7** **3 5 9**	**28** **0**

5.7. Bài toán tìm đường đi ngắn nhất

BÀI 5.7.1:

Hãy vẽ các đường đi ngắn nhất cho mỗi đỉnh của một đồ thị.

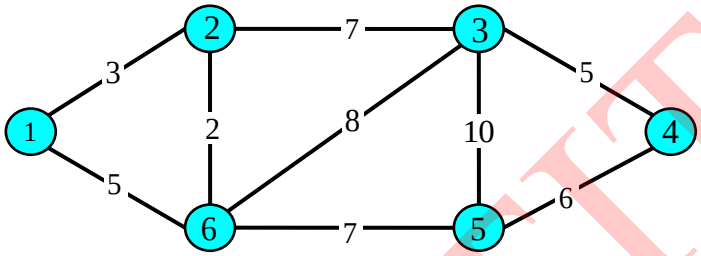
BÀI 5.7.2:

Tìm đường đi ngắn nhất trên đồ thị cho bởi hình dưới:

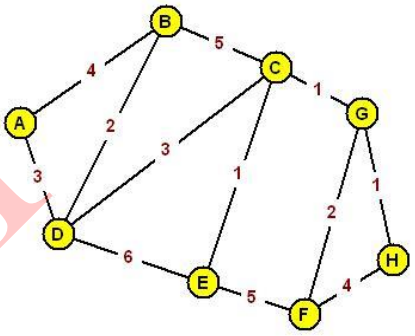
a) Từ đỉnh 1 đến đỉnh 4 trong hình 1

b) Từ đỉnh A đến đỉnh H trong hình 2

c) Từ đỉnh B đến đỉnh F trong hình 2



Hình 1



Hình 2

BÀI 5.7.3:

Các thuật toán tìm đường đi ngắn nhất đối với đồ thị vô hướng đã học có đúng với đồ thị có hướng hay không? Giải thích tại sao và cho một ví dụ nếu nó sai.

BÀI 5.7.4:

Cho một bảng các số tự nhiên kích thước m x n. Từ một ô có thể di chuyển sang một ô kề cạnh với nó. Hãy tìm một cách đi từ ô (x,y) ra một ô biên sao cho tổng các số ghi trên các ô đi qua là nhỏ nhất.

BÀI 5.7.5:

Cho một dãy số nguyên A = (a1, a2, …, an). Hãy tìm một dãy con gồm nhiều nhất các phần tử của dãy đã cho mà tổng của 2 phần tử liên tiếp là số nguyên tố.

BÀI 5.7.6:

Một công trình lớn được chia làm n công đoạn. Công đoạn i phải thực hiện mất thời gian t_i. Quan hệ giữa các công đoạn được cho bởi bảng A trong đó A= $\{a_{ij}\}_{n\times n}$ trong đó a_{ij}= 1 nếu công đoạn j chỉ được bắt đầu khi mà công đoạn i đã hoàn thành và a_{ij} = 0 trong trường hợp ngược lại. Mỗi công đoạn khi bắt đầu cần thực hiện liên tục cho tới khi hoàn thành, hai công đoạn độc lập nhau có thể tiến hành song song. Hãy bố trí lịch thực hiện các công đoạn sao cho thời gian hoàn thành cả công trình là sớm nhất, cho biết thời gian sớm nhất đó.

Gợi ý:

Dựng đồ thị có hướng G = (V,E), mỗi đỉnh tương ứng với một công đoạn, đỉnh u có cung nối tới đỉnh v nếu công đoạn u phải hoàn thành trước khi công đoạn v bắt đầu. Thêm vào G một đỉnh s và cung nối từ s tới tất cả các đỉnh còn lại. Gán trọng số mỗi cung (u, v) của đồ thị bằng t_v. Nếu đồ thị có chu trình, không thể có cách xếp lịch, nếu đồ thị không có chu trình (DAG) tìm đường đi dài nhất xuất phát từ s tới tất cả các đỉnh của đồ thị, khi đó nhãn khoảng cách d[v] chính là thời điểm hoàn thành công đoạn v, ta chỉ cần xếp lịch để công đoạn v được bắt đầu vào thời điểm d[v] - t_v là xong.

BÀI 5.7.7: <u>*BIN LADEN*</u>

Trùm khủng bố Bin Laden trốn trong 1 căn hầm được đào sâu xuống mặt đất M tầng, mỗi tầng có N phòng. Các phòng được ngăn cách bằng các cửa rất khó phá. Các phòng có cửa xuống phòng ngay phía dưới và 2 phòng ở 2 bên. Từ trên mặt đất có N cửa xuống N phòng tầng -1. Bin Laden ở tầng dưới cùng (tầng -M) phòng thứ N (phòng ở bên phải nhất). Mỗi cửa được làm bằng một kim loại khác nhau với độ dày khác nhau nên việc phá cửa cần thời gian khác nhau.

Bạn hãy tìm cách đi từ mặt đất xuống phòng của Bin Laden nhanh nhất không hắn thoát mất.

Dữ liệu:

- Dòng 1 ghi M và N
- Dòng 2 đến 2M + 1, dòng chẵn ghi N số, dòng lẻ ghi N - 1 số là chi phí để phá cửa.

Kết quả:

Ghi ra 1 số là thời gian nhỏ nhất để đến được phòng của Bin Laden

Ví dụ

Dữ liệu

4 2

99 10

1

10 99

1

99 10

1

10 99

1

Kết quả

44

```
+--99--+--10--+
|      |      |
|      1      |
|      |      |
+--10--+--99--+
|      |      |
|      1      |
|      |      |
+--99--+--10--+
|      |      |
|      1      |
|      |      |
+--10--+--99--+
```

```
|      |      |
|      1      |
|      |      |
+------+------+
Đi theo đường zigzac
```

Giới hạn

- 1 <= M <= 2222
- 1 <= N <= 10
- Chi phí của các cánh cửa thuộc [0, 1000].

5.8. Bài toán luồng cực đại trên mạng

BÀI 5.8.1:

Hãy đưa ra một thuật toán để giải quyết bài toán dòng chảy trong mạng lưới trong trường hợp mạng lưới có dạng một cây nhờ vào xóa đi đỉnh đích.

BÀI 5.8.2:

Những đường đi nào được đi qua khi tìm dòng chảy tối đa trong mạng lưới có được bằng cách thêm các cạnh từ B đến C và từ E đến D với trọng lượng 3.

BÀI 5.8.3:

Khẳng định sau đây đứng hay sai: không thuật toán nào có thể tìm được dòng chảy cực đại mà không kiểm tra mỗi cạnh trong mạng lưới.

BÀI 5.8.4:

Điều gì xảy ra đối với thuật toán Ford-Fullkerson khi mạng lưới có một chu trình có hướng.

BÀI 5.8.5:

Tìm một phản ví dụ cho thấy tại sao tìm kiếm ưu tiên độ sâu không thích hợp đối với bài toán dòng chảy trong mạng lưới.

BÀI 5.8.6: *BẢO VỆ*

Một mạng lưới gồm N thành phố, và một số đường một chiều nối các cặp thành phố (giữa hai thành phố có thể có nhiều đường nối một chiều).

Quân địch đang tập trung ở thành phố N, định tiến công ta ở thành phố 1, và chúng sẽ tiến công trên tất cả các con đường chưa được bảo vệ để tiến vào thành phố 1. Bộ chỉ huy ta cần xác định số quân ít nhất trên các con đường để chặn địch tiến về thành phố 1.

Input

Dòng đầu ghi N ($N \leq 5000$)

Các dòng tiếp theo cho đến hết file, mỗi dòng một tả 1 đường gồm u, v, s cho biết có đoạn đường một chiều từ u đến v, và phải cần ít nhất s quân để chặn địch trên đường này. ($s \leq 65000$)

Có không quá 10000 đường.

Output

Số quân ít nhất cần điều động

Ví dụ:

Input:

```
10
10 7 25050
6 1 12564
10 4 23916
5 1 61054
10 9 50950
9 1 35558
10 2 60941
3 1 22203
8 2 2853
5 7 31422
3 7 41491
8 7 27235
4 8 55965
```

```
8 6 41980
3 6 47707
2 3 45320
3 8 11237
7 6 38734
5 6 7561
3 5 8844
```

Output:

```
79169
```

BÀI 5.8.7: *HỆ ĐẠI DIỆN PHÂN BIỆT*

(Hệ đại điện phân biệt) Một lớp học có n bạn nam và n bạn nữ. Nhân ngày 8/3, lớp có mua m món quà để các bạn nam tặng các bạn nữ. Mỗi món quà có thể thuộc sở thích của một số bạn trong lớp. Hãy lập chương trình tìm cách phân công tặng quả thỏa mãn:

Mỗi bạn nam phải tặng quà cho đúng một bạn nữ và mỗi bạn nữ phải nhận quà của đúng một bạn nam. Món quà được tặng phải thuộc sở thích của cả hai người.

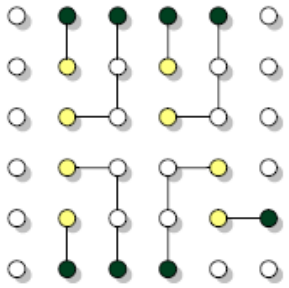
Món quà nào đã được một bạn nam chọn để tặng thì bạn nam khác không được chọn nữa.

Gợi ý:

- *Xây dựng một mạng trong đó tập đỉnh V gồm 3 lớp đỉnh S, X và T:*
- *Lớp đỉnh phát $S = (s_1,s_2,\ldots,s_n)$, mỗi đỉnh tương ứng với một bạn nam.*
- *Lớp đỉnh $X = (x_1, x_2,\ldots,x_n)$, mỗi đỉnh tương ứng với một món quà.*
- *Lớp đỉnh thu $T = (t_1,t_2,\ldots,t_n)$, mỗi đỉnh tương ứng với một bạn nữ.*
- *Nếu bạn nam i thích món quà k, ta cho cung nối từ si tới xk, nếu bạn nữ j thích món quà k, ta cho cung nối từ xk tới tj và sức chứa của các cung đặt bằng 1 và sức chứa của các đỉnh v1,v2,…,vn cũng đặt bằng 1. Tìm luồng nguyên cực đại trên mạng G có n đỉnh phát, n đỉnh thu, đồng thời có cá ràng buộc sức chứa trên các đỉnh, những cung có luồng 1 sẽ nối giữa một món quà và người tặng/nhận tương ứng.*

BÀI 5.8.8: *MẠNG ĐIỆN*

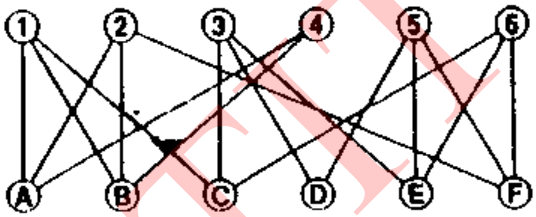
Cho mạng điện gồm m x n điểm nằm trên một lưới m hàng, n cột. Một số điểm nằm trên biên của lưới là nguồn điện, một số điểm trên lưới là các thiết bị sử dụng điện. Người ta chỉ cho phép nối dây điện giữa hai điểm nằm cùng hàng hoặc cùng cột. Hãy tìm cách đặt các dây điện nối các thiết bị sử dụng điện với nguồn điện sao cho hai đường dây bất kỳ nối hai thiết bị sử dụng điện với nguồn điện tương ứng của chúng không được có điểm chung.



5.9. Đồ thị hai phía

BÀI 5.9.1:

Tìm tất cả các bộ đối sánh năm cạnh của đồ thị hai phần như hình dưới:



BÀI 5.9.2:

Tìm ra các bộ đối sánh cực đại cho các đồ thị hai phần có 50 đỉnh và 100 cạnh. Có khoảng bao nhiêu cạnh trong mỗi bộ đối sánh.

BÀI 5.9.3:

Xây dựng một đồ thì 2 phần gồm 6 nút và 8 cạnh mà có một bộ đối sánh 3 cạnh. Nếu không được hãy chứng minh không tồn tại đồ thị như thế.

BÀI 5.9.4:

Giả sử rằng các đỉnh trong một đồ thị hai phần biểu diễn các công việc và con người, mỗi người được giao cho 2 việc. Có thể dùng thuật toán dòng chảy trong mạng lưới để giải bài toán này hay không? Chứng minh trả lời của bạn.

BÀI 5.9.5:

Hãy viết một chương trình hiệu quả để xác định xem một phương án của bài toán hôn nhân có bền vững hay không?

BÀI 5.9.6:

Có thể để cho hai chàng trai chọn cô gái cuối cùng của danh sách của mình trong thuật toán hôn nhân bền vững hay không? Chứng minh trả lời của bạn.

BÀI 5.9.7:

Xây dựng một tập danh sách các sở thích với N = 4 cho bài toán hôn nhân bền vững trong đó mỗi người chọn được người thứ 2 trong danh sách sở thích của mình. Hãy chứng minh rằng không tồn tại tập hợp như thế.

BÀI 5.9.8:

Cho biết cấu hình bền vững của bài toán hôn nhân bền vững trong trường hợp các danh sách sở thích của các chàng trai và các cô gái là như nhau theo thứ tự tăng.

BÀI 5.9.9:

Viết chương trình hôn nhân bền vững với N = 50, sử dụng các hoán vị ngẫu nhiên cho các danh sách sở thích. Có khoảng bao nhiêu cuộc hôn nhân trong suốt quá trình thực hiện thuật toán.?

BÀI 5.9.10:

Có p thợ và q việc. Mỗi thợ cho biết mình có thể làm được những việc nào, và mỗi việc khi giao cho một thợ thực hiện sẽ được hoàn thành xong trong đúng 1 đơn vị thời gian. Tại một thời điểm, mỗi thợ chỉ thực hiện không quá một việc.

Hãy phân công các thợ làm công việc sao cho:

- Mỗi việc chỉ giao cho đúng một thợ thực hiện.
- Thời gian hoàn thành tất cả các công việc là nhỏ nhất. Chú ý là các thợ có thể thực hiện song song các công việc được giao, việc của ai người nấy làm, không ảnh hưởng tới người khác.

VI.Các kỹ thuật sắp xếp và tìm kiếm

6.1. Các phương pháp sắp xếp cơ bản

BÀI 6.1.1:

Cho một dãy số nguyên dương: 6, 8, 12, 9, 2, 2, 98, 23, 23. Hãy mô phỏng sắp xếp tăng dần bằng các thuật toán: sắp xếp chọn, sắp xếp chèn, sắp xếp nổi bọt, sắp xếp Shellsort.

BÀI 6.1.2:

Cho dãy n số nguyên a[0], a[1],…, a[n-1] đã được sắp xếp tăng dần và một số nguyên x.

a. Hãy viết hàm tìm kiếm nhị phân kiểm tra xem x có thuộc dãy số trên hay không? Nếu tìm thấy trả về giá trị i nhỏ nhất mà a[i] = x, nếu không trả về giá trị -1.

b.Cho dãy n = 8 số nguyên như sau:

1 2 4 4 4 9 10 15

Nếu x = 4 thì phương pháp tìm kiếm nhị phân cho ra kết quả gì ?

c.Cho biết k số phần tử lớn nhất của dãy.

Ví dụ với n = 12

9 6 2 7 9 9 6 5 7 9 6 7

Nếu k = 5 thì kết quả là 9, 9, 9, 9, 7

BÀI 6.1.3:

Cho mảng 1 chiều n phần tử. Sắp xếp các số nguyên tố tăng dần, các số khác giữ nguyên giá trị và vị trí.

BÀI 6.1.4:

Cho mảng vuông n. Hãy tìm phần tử lớn nhất trên mỗi đường chéo song song với đường chéo chính.

BÀI 6.1.5:

Cho ma trận 2 chiều m dòng, n cột.. Hãy sắp tăng dần các phần tử theo chiều từ trái qua phải và từ trên xuống dưới.

BÀI 6.1.6:

Sắp xếp các phần tử trên các đường chéo song song với đường chéo chính tăng dần.

BÀI 6.1.7:

Cho mảng một chiều gồm n phần tử là các số nguyên. Sắp xếp các số chẵn trong mảng theo thứ tự tăng, sắp xếp các số lẻ theo thứ tự giảm dần, các số 0 giữ nguyên vị trí..

BÀI 6.1.8:

Cho mảng một chiều gồm n phần tử là các số nguyên. Tìm k giá trị lớn nhất khác nhau của mảng

BÀI 6.1.9:

Cho mảng một chiều gồm n phần tử là các số nguyên.

a. Chỉ giữ lại một giá trị trong số các giá trị giống nhau.

b. Sắp xếp các số chẵn trong mảng theo thứ tự tăng, sắp xếp các số lẻ theo thứ tự giảm dần, các số 0 giữ nguyên vị trí.

BÀI 6.1.10:

Cho tập tin văn bản songuyen.inp chứa các số nguyên. Hãy ghi các số nguyên tố trong tập tin songuyen.inp vào tập tin nguyento.out theo thứ tự tăng dần mỗi dòng ghi 10 số, các số cách nhau ít nhất một khoảng cách.

BÀI 6.1.11:

Cho 2 file số nguyên được sắp tăng dần. Hãy trộn 2 file để được một file cũng được sắp tăng dần (không dùng mảng).

BÀI 6.1.12:

Trong 3 phương pháp sắp xếp cơ bản là phương pháp chọn, chèn và nổi bọt thì phương pháp nào là nhanh nhất đối với một tập tin đã được sắp rồi?

BÀI 6.1.13:

Trong 3 phương pháp sắp xếp cơ bản là phương pháp chọn, chèn và nổi bọt thì phương pháp nào là nhanh nhất đối với một tập tin đã được sắp thứ tự đảo ngược?

BÀI 6.1.14:

Hãy kiểm chứng 2 câu hỏi trên bằng lập trình với dãy số nguyên.

BÀI 6.1.15:

Hãy đưa ra một lý do hợp lý tại sao không thuận tiện khi dùng một khóa cầm canh (đứng gác) cho phương pháp sắp xếp chèn (trừ phương pháp phát sinh từ cài đặt của Shellsort).

BÀI 6.1.16:

Có bao nhiêu phép so sánh được dùng bởi Shellsort với sắp-7, rồi sắp-3 với các khóa E A S Y Q U E S T I O N?

BÀI 6.1.17:

Hãy cho một ví dụ để minh họa tại sao 8, 4, 2, 1 sẽ không là một cách tốt để kết thúc một dãy tăng của Shellsort.

BÀI 6.1.18:

Phương pháp sắp xếp chọn có ổn định không? Tương tự đối với chèn và nổi bọt?

BÀI 6.1.19:

Hãy thử nghiệm với các dãy tăng khác nhau cho Shellsort, tìm một dãy mà nó chạy nhanh hơn dãy được cho bởi một tập tin ngẫu nhiên gồm 1000 phần tử.

BÀI 6.1.20: *SẮP XẾP 2*

Cho một danh sách chứa cả các số và các từ. Yêu cầu bạn hãy sắp xếp danh sách này tăng dần sao cho các từ theo thứ tự từ điển, các số theo thứ tự số. Hơn nữa, nếu phần tử thứ n là số thì danh sách sau khi sắp xếp phần tử thứ n cũng phải là số, nếu là từ thì vẫn là từ.

Lưu ý: Các từ chỉ gồm các chữ in thuờng trong bảng chữ cái tiếng Anh.

Input

Gồm nhiều dòng, mỗi dòng là một danh sách. Mỗi phần tử của danh sách cách nhau bởi dấu phẩy (“,”) theo sau là dấu cách, và danh sách được kết thúc bằng dấu chấm (“.”).

Dữ liệu kết thúc bởi dòng chỉ chứa một dấu chấm.

Output

Với mỗi danh sách trong dữ liệu, xuất ra danh sách đã sắp xếp thỏa mãn yêu cầu đề bài (có định dạng như trong dữ liệu).

Ví dụ:

Input:

0.

banana, strawberry, orange.

banana, strawberry, orange.

10, 8, 6, 4, 2, 0.

x, 30, -20, z, 1000, 1, y.

50, 7, kitten, puppy, 2, orangutan, 52, -100, bird, worm, 7, beetle.

Output:

0.

banana, orange, strawberry.

banana, orange, strawberry.

0, 2, 4, 6, 8, 10.

x, -20, 1, y, 30, 1000, z.

-100, 2, beetle, bird, 7, kitten, 7, 50, orangutan, puppy, 52, worm.

BÀI 6.1.21: CHỨNG KHOÁN

Cho trước lịch sử giao dịch của một mã chứng khoán trong n ngày. Hãy xác định k1 ngày có giá thấp nhất và k2 ngày có giá cao nhất.

Input

Mỗi bộ test gồm 2 dòng

- Dòng 1 ghi 3 số n, k1, k2 với n<=106. k1+k2<=n và k1,k2<=100.
- Dòng tiếp theo ghi n số nguyên theo thứ tự là giá của mã chứng khoán trong n ngày liên tiếp.

Bộ test cuối cùng chứa 3 số 0

Output

Với mỗi bộ test, ghi ra màn hình 3 dòng gồm:

- Dòng 1 ghi số thứ tự bộ test
- Dòng 2 ghi k1 ngày có giá thấp nhất theo thứ tự các ngày tăng dần. Nếu có nhiều danh sách cho kết quả giống nhau thì chọn danh sách thấp nhất theo thứ tự từ điển.
- Dòng 3 ghi k2 ngày có giá cao nhất theo thứ tự các ngày giảm dần. Nếu có nhiều danh sách cho kết quả giống nhau thì chọn danh sách cao nhất theo thứ tự từ điển.

Ví dụ:

Input:

10 3 2

1 2 3 4 5 6 7 8 9 10

10 3 2

10 9 8 7 6 5 4 3 2 10 0 0

Output:

Case 1

1 2 3

10 9

Case 2

8 9 102 1

BÀI 6.1.22: CHẠY ĐUA MARATHON

John cho các con bò của mình chạy đua marathon! Thời gian bò N (1 <= N <= 5,000) về đích được biểu diễn theo dạng Số giờ (0 <= Số giờ <= 99), Số phút (0 <= Số phút <= 59), và số giây (0 <= Số giây <= 59). Để xác định nhà vô địch, John phải sắp xếp các thời gian (theo số giờ, số phút, và số giây) theo thứ tự tăng dần, thời gian ít nhất xếp đầu tiên.

Ví dụ: Với 3 thời gian như sau:

11:20:20

11:15:12

14:20:14

Kết quả sau khi sắp xếp là:

11:15:12

11:20:20

14:20:14

INPUT FORMAT:

* Line 1: 1 số nguyên: N

* Lines 2..N+1: Dòng i+1 chứa thời gian bò i được mô tả bởi 3 số nguyên cách bởi dấu cách : Số Giờ , Số Phút, Số giây.

OUTPUT FORMAT:

* Dòng 1..N: Mỗi dòng chứa thời gian của 1 con bò là 3 số nguyên cách nhau bởi dấu cách sau khi đã sắp xếp.

SAMPLE INPUT:

3

11 20 20

11 15 12

14 20 14

SAMPLE OUTPUT:

11 15 12

11 20 20

14 20 14

BÀI 6.1.23: *REPLACING DIGITS*

Cho số nguyên dương a có N chữ số và số dãy s có M chữ số. Chữ số ở vị trí j (1<=j<=M) của dãy s có thể chọn bất kì vị trí i (1<=i<=N) trong số a và thay thế bằng s_j. Mỗi chữ số của dãy s chỉ được thay thế không quá một lần.

Nhiệm vụ của bạn là hãy tìm cách thay sao cho số a đạt giá trị lớn nhất. Bạn có thể không cần sử dụng tất cả các chữ số trong s.

Input

Dòng đầu chứa số nguyên dương a có độ dài N (không bắt đầu bằng chữ số 0).

Dòng 2 chứa dãy s có độ dài M

(1≤N,M≤105)

Output

Số a lớn nhất có thể thay thế được.

Ví dụ:

Input:

1024

010

Output:

1124

Input:

987

1234567

Output:

987

6.2. Quicksort

BÀI 6.2.1:

Hãy vẽ cây phân hoạch đệ qui của thuật toán Quick-Sort trong trường hợp xấu nhất. Từ đó, chứng tỏ rằng chi phí thuật toán Quick-sort trong trường hợp này là O(n2).

BÀI 6.2.2:

Cài đặt một thuật toán Quicksort đệ quy với sự cắt xén bớt phép sắp xếp chèn cho các tập tin con có ít hơn M phần tử và xác đinh theo kinh nghiệm giá trị của M mà nó sẽ chạy nhanh nhất trên một tập tin có 1000 phần tử.

BÀI 6.2.3:

Giải bài toán trên đối bằng khử đệ quy.

BÀI 6.2.4:

Giải bài toán trên có bổ sung phép chọn phần tử giữa của 3 phần tử.

BÀI 6.2.5:

Quicksort sẽ thực hiện bao lâu để sắp một tập tin gồm N phần tử bằng nhau?

BÀI 6.2.6:

Số lần tối đa mà phần tử lớn nhất có thể được di chuyển trong lúc thi hành Quicksort là bao nhiêu?

BÀI 6.2.7:

Hãy chỉ ra làm thế nào tập tin A B A B A B A được phân hoạch, sử dụng các phương pháp đã học?

BÀI 6.2.8:

Quicksort dùng bao nhiêu phép so sánh để sắp xếp các khóa E A S Y Q U E S T I O N?

BÀI 6.2.9:

Cần bao nhiêu khóa "cầm canh" nếu phương pháp sắp xếp chèn được gọi một cách trực tiếp từ Quicksort?

BÀI 6.2.10:

Có hợp lý không khi dùng một hàng đợi thay vì một ngăn xếp cho một bản cài đặt không đệ quy của Quicksort? Tại sao có và tại sao không?

BÀI 6.2.11:

Viết một chương trình để tổ chức lại một tập tin sao cho tất cả các phần tử với các khóa bằng với giá trị trung bình thì nằm tại chỗ, với các phần tử nhỏ hơn thì nằm bên trái và các phần tử lớn hơn thì nằm bên phải.

6.3. Heapsort

BÀI 6.3.1:

Hãy cho biết số phần tử tối thiểu và tối đa trong một heap có chiều cao h ?

BÀI 6.3.2:

Vẽ heap có được khi các thao tác sau thực hiện trên một heap rỗn ban đầu: insert(10), insert(5), insert(2), replace(4), insert(6), insert(8), remove, insert(7), insert(3).

BÀI 6.3.3:

Một tập tin sắp theo thứ tự ngược có phải là một heap không?

BÀI 6.3.4:

Hãy cho heap được tạo bởi việc áp dụng liên tiếp phép chèn trên các khóa E A S Y Q U E S T I O N.

BÀI 6.3.5:

Các vị trí nào có thể bị chiếm bởi khóa nhỏ thứ ba trong một heap kích thước 32.

BÀI 6.3.6:

Tại sao không dùng một biến cầm canh để tránh phép kiểm tra j < N trong downheap?

BÀI 6.3.7:

Hãy minh họa làm thế nào để nhận được các hàm của ngăn xếp và hàng đợi chuẩn như là trường hợp đặc biệt của các hàng đợi có ưu tiên.

BÀI 6.3.8:

Số khóa tối thiểu phải được di chuyển trong một thao tác “hủy cái lớn nhất: trong 1 heap là bao nhiêu? Hãy vẽ 1 heap có kích thước 15 mà số tối thiểu đạt được.

BÀI 6.3.9:

Viết một chương trình để xóa phần tử ở vị trí thứ k trong 1 heap.

BÀI 6.3.10:

Hãy so sánh theo kinh nghiệm, việc xây dựng heap từ dưới lên với việc xây dựng heap từ trên xuống, bằng cách nào tạo các heap với 1000 khóa ngẫu nhiên.

6.4. Mergesort

BÀI 6.4.1:

Viết chương trình cho phương pháp sắp xếp trộn (Merge sort) đệ quy bằng cách cắt xén bớt phương pháp sắp xếp chèn cho các tập tin con nhỏ hơn M phần tử; hãy xác định theo kinh nghiệm giá trị của M để nó chạy nhanh nhất trên một tập tin ngẫu nhiên gồm 1000 phần tử.

BÀI 6.4.2:

So sánh theo kinh nghiệm sắp xếp trộn đệ quy và không đệ quy cho các xâu liên kết và N = 1000.

BÀI 6.4.3:

Cài đặt phép sắp xếp trộn đệ quy cho một mảng gồm N số nguyên, sử dụng một mảng phụ trợ có kích thước nhỏ hơn N/2.

BÀI 6.4.4:

Phát biểu "thời gian chạy của sắp xếp trộn không phụ thuộc vào giá trị của các khóa trong tập tin nhập" là đúng hay sai? Hãy giải thích câu trả lời của bạn.

BÀI 6.4.5:

Số bước tối thiểu mà sắp xếp trộn có thể dùng là bao nhiêu?

BÀI 6.4.6:

Hãy chỉ ra các phép trộn được thực hiện khi dùng đệ quy để sắp xếp các khóa E A S Y Q U E S T I O N.

BÀI 6.4.7:

Hãy cho biết nội dung của xâu liên kết ở mỗi bước lặp khi dùng sắp xếp trộn không đệ quy để sắp xếp các khóa E A S Y Q U E S T I O N.

BÀI 6.4.8:

Hãy thử nghiệm một phép sắp xếp trộn đệ quy, sử dụng mảng, lấy ý tưởng thực hiện các phép trộn 3-way thay vì 2-way

6.5. Sắp xếp bằng cơ số

BÀI 6.5.1:

So sánh số hoán vị được dùng bởi sắp xếp hoán vị cơ số với số hoán vị được dùng bởi Quicksort cho tập tin gồm các khóa 001, 011, 101, 110, 000, 001,010,111, 110, 010.

BÀI 6.5.2:

Tại sao lại không quan trọng khi khử đệ quy từ phương pháp sắp xếp hoán vị cơ số như là đối với Quicksort.

BÀI 6.5.3:

Hãy sửa đổi phương pháp sắp xếp hoán vị cơ số để nhảy qua các bit dẫn đầu giống nhau trên tất cả các khóa. Trong những trường hợp nào thì điều này trở nên phung phí vô ích?

BÀI 6.5.4:

Điều sau đây đúng hay sai: thời gian thực hiện của phương pháp sắp xếp cơ số trực tiếp không phụ thuộc vào thứ tự các khóa trong tập tin nhập. Hãy giải thích câu trả lời của bạn.

BÀI 6.5.5:

Phương pháp nào sẽ nhanh hơn đối với tập tin gồm các khóa bằng nhau: sắp xếp hoán vị cơ số hay sắp xếp cơ số trực tiếp?

BÀI 6.5.6:

Điều sau đây đúng hay sai: cả hai phương pháp sắp xếp hoán vị cơ số và sắp xếp cơ số trực tiếp sẽ kiểm tra tất cả các bit của tất cả các khóa trong tập tin. Hãy giải thích câu trả lời của bạn.

BÀI 6.5.7:

Trừ yêu cầu về bộ nhớ bổ sung, hãy cho biết bất tiện chính của chiến lược thực hiện phép sắp cơ số trực tiếp trên các bitđi đầu của các khóa, rồi tiếp tục bằng phương pháp chèn sau đó.

BÀI 6.5.8:

Cần dùng bao nhiêu bộ nhớ để thực hiện một phép sắp cơ số trực tiếp 4-đường của N khóa b-bit?

BÀI 6.5.9:

Kiểu nào của tập tin nhập sẽ khiến cho phép sắp hoán vị cơ số chạy chậm nhất (N rất lớn)?

BÀI 6.5.10:

Hãy so sánh theo kinh nghiệm phép sắp xếp cơ số trực tiếp với phép sắp hoán vị cơ số đối với một tập tin gồm 1000 khóa 32-bit.

6.6. Các phương pháp tìm kiếm cơ bản

BÀI 6.6.1:

Cài đặt một thuật toán tìm kiếm tuần tự mà đòi hỏi trung bình khoảng N/2 bước cho cả hai trường hợp tìm kiếm thành công và không thành công, thuật toán này lưu các mẩu tin trong một mảng được sắp xếp.

BÀI 6.6.2:

Hãy đưa ra một cài đặt đệ quy của thuật toán tìm kiếm nhị phân.

Giả sử a[i] = 2i với 1<= i <= N. Có bao nhiêu vị trí trong bảng được kiểm tra khi dùng tìm kiếm nội suy trong trường hợp tìm kiếm không thành công cho 2k – 1?

BÀI 6.6.3: *THAY THẾ TỪ*

Hai file INPUT1.TXT và INPUT2.TXT được cho như sau: File INPUT1.TXT chứa một đoạn văn bản bất kì. File INPUT2.TXT chứa không quá 50 dòng, mỗi dòng gồm hai từ: từ đầu là từ đích và từ sau là từ nguồn. Hãy tìm trong file INPUT1.TXT tất cả các từ là từ đích và thay thế chúng bằng các từ nguồn tương ứng. Kết quả ghi vào file KQ.OUT (sẽ là một đoạn văn bản tương tự như trong file INPUT1.TXT nhưng đã được thay thế từ đích bởi từ nguồn).

Ví dụ:

Input:

- File INPUT1.TXT chứa đoạn văn bản sau:

  Nam moi sap den roi, ban co vui khong?

  Chuc cac ban don mot cai Tet that vui ve va hanh phuc.

  Chuc ban luon hoc gioi!

- File INPUT2.TXT chứa các dòng sau:

  ban em

  zui vui

Output:

- File KQ.OUT sẽ chứa đoạn văn bản sau:

  Nam moi sap den roi, em co vui khong?

  Chuc cac em don mot cai Tet that vui ve va hanh phuc.

  Chuc em luon hoc gioi!

BÀI 6.6.4: *DÃY SỐ NGUYÊN*

Dãy các số tự nhiên được viết ra thành một dãy vô hạn trên đường thẳng:

12345678910111213 14..... (1)

Hỏi số ở vị trí thứ 1000 trong dãy trên là số nào?

Hãy làm bài này theo hai cách: Cách 1 dùng suy luận logic và cách 2 viết chương trình để tính toán và so sánh hai kết quả với nhau.

Tổng quát bài toán trên: Chương trình yêu cầu nhập số K từ bàn phím và in ra trên màn hình kết quả là số nằm ở vị trí thứ K trong dãy (1) trên. Yêu cầu chương trình chạy càng nhanh càng tốt.

BÀI 6.6.5: BIRTHDATES

Viết chương trình tìm người trẻ nhất và già nhất trong lớp.

Input

- Dòng 1 chứa số n (1<=n<=100), số người trong lớp.
- N dòng sau, mỗi dòng là thông tin 1 người có dạng:

  personName dd mm yyyy

  Trong đó: personName là tên không quá 15 chữ cái, dd,mm,yyyy lần lượt là ngày, tháng, và năm sinh.

Output

Dòng 1: tên người trẻ nhất

Dòng 2: tên người già nhất

Ví dụ:

Input:

5

Garfield 20 9 1990

5

Mickey 1 10 1991

Alice 30 12 1990

Tom 15 8 1993

Jerry 18 9 1990

Garfield 20 9 1990

Output:

Tom

Jerry

6.7. Phép băm

BÀI 6.7.1:

Mô tả làm thế nào để cài đặt một hàm băm bằng cách dùng một bộ phát sinh số ngẫu nhiên tốt. Có ý nghĩa hay không nếu cài đặt một bộ phát sinh ngẫu nhiên bằng cách dùng một hàm băm?

BÀI 6.7.2:

Lượng giá trường hợp xấu nhất khi chèn N khóa vào một bảng được khởi tạo trống bằng cách dùng xích ngăn cách với các danh sách không thứ tự.

BÀI 6.7.3:

Cho biết nội dung của bảng băm có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một bảng được khởi tạo trống kích thước bằng 13 bằng phương pháp dò tuyến tính.

BÀI 6.7.4:

Cho biết nội dung của bảng băm có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một bảng được khởi tạo trống kích thước bằng 13 bằng phương pháp băm kép. (Trong đó h1(k) lấy từ câu hỏi trước, h2(k) = 1 + (k mod 11).)

BÀI 6.7.5:

Cần khoảng bao nhiêu lần dò khi dùng phương pháp băm kép để xây dựng một bảng với N khóa bằng nhau?

BÀI 6.7.6:

Nên dùng phương pháp băm nào cho một ứng dụng mà có nhiều trường hợp khóa trùng nhau?

BÀI 6.7.7:

Giả sử rằng cho biết trước số phần tử sẽ được đặt vào bảng băm. Với những điều kiện nào thì phương pháp xích ngăn cách thích hợp hơn phương pháp băm kép?

BÀI 6.7.8:

Giả sử một lập trình viên có một lỗi trong chương trình dùng phương pháp băm kép mà một trong các hàm luôn trả về cùng một giá trị (khác 0). Mô tả điều gì sẽ xảy ra trong mỗi tình huống (khi hàm thứ nhất bị sai và khi hàm thứ hai bị sai).

BÀI 6.7.9:

Nên dùng hàm băm nào nếu biết trước rằng các giá trị khóa rơi vào một phạm vi tương đối nhỏ.

BÀI 6.7.10:

Phê bình thuật toán sau đây, thuật toán này nhằm mục đích xóa khóa khỏi một bảng băm được xây dựng bằng phương pháp dò tuyến tính. Quét qua phải kể từ phần tử được xóa để ra một vị trí trống, kế đến quét trái để tìm một phần tử có cùng giá trị băm, sau cùng thay thế phần tử được xóa bởi phần tử vừa tìm được.

6.8. Tìm kiếm dựa vào cơ số

BÀI 6.8.1:

Vẽ cây tìm kiếm số học có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một cây được khởi tạo trống.

BÀI 6.8.2:

Phát sinh một cây tìm kiếm số học 1000 nút và so sánh độ cao và số nút mỗi tầng của nó với cây tìm kiếm nhị phân chuẩn và cây tìm kiếm đỏ đen được xây dựng từ cùng một tập khóa.

BÀI 6.8.3:

Hãy tìm một tập hợp 12 khóa mà chúng tạo nên một cây tìm kiếm số học cân bằng yếu.

BÀI 6.8.4:

Vẽ cây tìm kiếm cơ số có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một cây được khởi tạo trống.

BÀI 6.8.5:

Một vấn đề xảy ra đối với các cây tìm kiếm số học 26-hướng (way) là một số ký tự trong bảng chữ cái thì lại được sử dụng rất thường xuyên. Hãy đề nghị một phương pháp giải quyết vấn đề này.

BÀI 6.8.6:

Mô tả phương pháp xóa một phần tử khỏi cây tìm kiếm cơ số đa hướng.

BÀI 6.8.7:

Vẽ cây Patricia có được khi chèn các khóa E A S Y Q U E S T I O N theo thứ tự đó vào một cây được khởi tạo trống.

BÀI 6.8.8:

Hãy tìm một tập hợp 12 khóa mà chúng tạo nên một cây Patricia cân bằng yếu.

BÀI 6.8.9:

Viết chương trình in ra tất cả các khóa trong cây Patricia mà có t bit khởi đầu giống với một khóa tìm kiếm đã cho.

BÀI 6.8.10:

Trong các phương pháp cơ số thì phương pháp nào thích hợp để viết chương trình in ra các khóa theo thứ tự? Phương pháp nào không thích hợp?

TÀI LIỆU THAM KHẢO

1. Lê Minh Hoàng. **Giải thuật và lập trình**, Đại học Sư phạm Hà Nội, 2010.

2. Robert Sedgewick. **Algorithms 2nd edition**, ISBN: 0201066734, Addison Wesley, 1988.

3. PTIT Online Judge. http://www.spoj.com/PTIT/