BÀI TẬP CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

PHẦN 1. CẤU TRÚC DỮ LIỆU

1) Xây dựng tập thao tác trên danh sách liên kết đơn
    a. Khởi tạo danh sách liên kết đơn.
    b. Thêm node vào đầu bên trái danh sách liên kết đơn.
    c. Thêm node vào đầu bên phải danh sách liên kết đơn.
    d. Thêm node vào vị trí pos trên danh sách liên kết đơn.
    e. Loại node ở đầu bên trái danh sách liên kết đơn.
    f. Loại node ở đầu bên phải danh sách liên kết đơn.
    g. Loại node ở vị trí pos của danh sách liên kết đơn.
    h. Tìm node có giá trị x trên danh sách liên kết đơn.
    i. Duyệt danh sách liên kết đơn.
    j. Sắp xếp các node của danh sách liên kết đơn.
    k. Đảo ngược các node của danh sách liên kết đơn.
2) Xây dựng tập thao tác trên danh sách liên kết đơn vòng
    a. Khởi tạo danh sách liên kết đơn vòng.
    b. Thêm node vào đầu bên trái danh sách liên kết đơn vòng.
    c. Thêm node vào đầu bên phải danh sách liên kết đơn vòng.
    d. Thêm node vào vị trí pos trên danh sách liên kết đơn vòng.
    e. Loại node ở đầu bên trái danh sách liên kết đơn vòng.
    f. Loại node ở đầu bên phải danh sách liên kết đơn vòng.
    g. Loại node ở vị trí pos của danh sách liên kết đơn vòng.
    h. Tìm node có giá trị x trên danh sách liên kết đơn vòng.
    i. Duyệt danh sách liên kết đơn vòng.
    j. Sắp xếp các node của danh sách liên kết đơn vòng.
    k. Đảo ngược các node của danh sách liên kết đơn vòng.
3) Xây dựng tập thao tác trên danh sách liên kết đơn sắp xếp
    a. Khởi tạo danh sách liên kết đơn sắp xếp.
    b. Thêm node vào đầu bên trái danh sách liên kết đơn sắp xếp.
    c. Loại node ở đầu bên trái danh sách liên kết đơn vòng.
    d. Loại node ở đầu bên phải danh sách liên kết đơn vòng.
    e. Loại node ở vị trí pos của danh sách liên kết đơn vòng.
    f. Tìm node có giá trị x trên danh sách liên kết đơn vòng.
    g. Duyệt danh sách liên kết đơn vòng.
    h. Đảo ngược các node của danh sách liên kết đơn vòng.
4) Xây dựng tập thao tác trên danh sách liên kết kép
    a. Khởi tạo danh sách liên kết kép.
    b. Thêm node vào đầu bên trái danh sách liên kết kép.
    c. Thêm node vào đầu bên phải danh sách liên kết kép.
    d. Thêm node vào vị trí pos trên danh sách liên kết kép.
    e. Loại node ở đầu bên trái danh sách liên kết kép.
    f. Loại node ở đầu bên phải danh sách liên kết kép.
    g. Loại node ở vị trí pos của danh sách liên kết kép.
    h. Tìm node có giá trị x trên danh sách liên kết kép.
    i. Duyệt danh sách liên kết kép.
    j. Sắp xếp các node của danh sách liên kết kép.
    k. Đảo ngược các node của danh sách liên kết kép.

5) Xây dựng tập thao tác trên danh sách liên kết kép vòng
    a. Khởi tạo danh sách liên kết đơn kép vòng.
    b. Thêm node vào đầu bên trái danh sách liên kết kép vòng.
    c. Thêm node vào đầu bên phải danh sách liên kết kép vòng.
    d. Thêm node vào vị trí pos trên danh sách liên kết kép vòng.
    e. Loại node ở đầu bên trái danh sách liên kết kép vòng.
    f. Loại node ở đầu bên phải danh sách liên kết kép vòng.
    g. Loại node ở vị trí pos của danh sách liên kết kép vòng.
    h. Tìm node có giá trị x trên danh sách liên kết kép vòng.
    i. Duyệt danh sách liên kết kép vòng.
    j. Sắp xếp các node của danh sách liên kết kép vòng.
    k. Đảo ngược các node của danh sách liên kết kép vòng.
6) Xây dựng tập thao tác trên danh sách liên kết kép sắp xếp (double sorted linked list)
    a. Khởi tạo danh sách liên kết kép sắp xếp.
    b. Thêm node vào đầu bên trái danh sách liên kết kép sắp xếp.
    c. Loại node ở đầu bên trái danh sách liên kết kép vòng.
    d. Loại node ở đầu bên phải danh sách liên kết kép vòng.
    e. Loại node ở vị trí pos của danh sách liên kết kép vòng.
    f. Tìm node có giá trị x trên danh sách liên kết kép vòng.
    g. Duyệt danh sách liên kết kép vòng.
    h. Đảo ngược các node của danh sách liên kết kép vòng.
7) Xây dựng tập thao tác trên đa thức bằng danh sách liên kết
    a. Tạo lập đa thức Pn(x), Qm(x).
    b. Tìm giá trị Pn(x0), Qm(x0).
    c. Tìm R = P + Q.
    d. Tìm R = P*Q.
    e. Tìm R = P/Q và đa thức dư.
    f. Tìm đạo hàm bậc l của đa thức.
8) Xây dựng các thao tác trên ngăn xếp dựa vào cấu trúc dữ liệu mảng :
    a. Kiểm tra tính rỗng của ngăn xếp.
    b. Kiểm tra tính đầy của ngăn xếp.
    c. Lấy phần tử ở đầu ngăn xếp.
    d. Lấy kích cỡ của ngăn xếp.
    e. Đưa phần tử vào ngăn xếp.
    f. Lấy phần tử ra khỏi ngăn xếp.
    g. Dịch chuyển biểu thức số học trung tố về biểu diễn hậu tố.
    h. Tính toán giá trị của biểu thức hậu tố.
9) Xây dựng các thao tác trên ngăn xếp dựa vào danh sách liên kết đơn :
    a. Kiểm tra tính rỗng của ngăn xếp.
    b. Kiểm tra tính đầy của ngăn xếp.
    c. Lấy phần tử ở đầu ngăn xếp.
    d. Lấy kích cỡ của ngăn xếp.
    e. Đưa phần tử vào ngăn xếp.
    f. Lấy phần tử ra khỏi ngăn xếp.
    g. Dịch chuyển biểu thức số học trung tố về biểu diễn hậu tố.
    h. Tính toán giá trị của biểu thức hậu tố.
10) Xây dựng các thao tác trên ngăn xếp sử dụng STL trong C++:
    a. Kiểm tra tính rỗng của ngăn xếp.
    b. Kiểm tra tính đầy của ngăn xếp.

c. Lấy kích cỡ của ngăn xếp.
d. Lấy phần tử ở đầu ngăn xếp.
e. Đưa phần tử vào ngăn xếp.
f. Dịch chuyển biểu thức số học trung tố về biểu diễn hậu tố.
g. Tính toán giá trị của biểu thức hậu tố.

11) Xây dựng các thao tác trên hàng đợi dựa vào mảng:
a. Kiểm tra tính rỗng của hàng đợi.
b. Kiểm tra tính đầy của hàng đợi.
c. Lấy kích cỡ của hàng đợi.
d. Lấy phần tử ở đầu hàng đợi.
e. Đưa phần tử vào hàng đợi.

12) Xây dựng các thao tác trên hàng đợi vòng dựa vào mảng:
a. Kiểm tra tính rỗng của hàng đợi vòng.
b. Kiểm tra tính đầy của hàng đợi vòng.
c. Lấy kích cỡ của hàng đợi vòng.
d. Lấy phần tử ở đầu hàng đợi vòng.
e. Đưa phần tử vào hàng đợi vòng.

13) Xây dựng các thao tác trên hàng đợi ưu tiên dựa vào mảng:
a. Kiểm tra tính rỗng của hàng đợi ưu tiên.
b. Kiểm tra tính đầy của hàng đợi ưu tiên.
c. Lấy kích cỡ của hàng đợi ưu tiên.
d. Lấy phần tử ở đầu hàng đợi ưu tiên.
e. Đưa phần tử vào hàng đợi ưu tiên.

14) Xây dựng các thao tác trên hàng đợi hai điểm cuối dựa vào mảng:
a. Kiểm tra tính rỗng của hàng đợi ưu tiên.
b. Kiểm tra tính đầy của hàng đợi ưu tiên.
c. Lấy kích cỡ của hàng đợi ưu tiên.
d. Lấy phần tử ở đầu hàng đợi ưu tiên.
e. Đưa phần tử vào hàng đợi ưu tiên.

15) Xây dựng các thao tác trên hàng đợi dựa vào danh sách liên kết đơn:
a. Kiểm tra tính rỗng của hàng đợi.
b. Kiểm tra tính đầy của hàng đợi.
c. Lấy kích cỡ của hàng đợi.
d. Lấy phần tử ở đầu hàng đợi.
e. Đưa phần tử vào hàng đợi.

16) Xây dựng các thao tác trên hàng đợi vòng dựa vào danh sách liên kết đơn:
a. Kiểm tra tính rỗng của hàng đợi vòng.
b. Kiểm tra tính đầy của hàng đợi vòng.
c. Lấy kích cỡ của hàng đợi vòng.
d. Lấy phần tử ở đầu hàng đợi vòng.
e. Đưa phần tử vào hàng đợi vòng.

17) Xây dựng các thao tác trên hàng đợi ưu tiên dựa vào danh sách liên kết đơn:
a. Kiểm tra tính rỗng của hàng đợi ưu tiên.
b. Kiểm tra tính đầy của hàng đợi ưu tiên.
c. Lấy kích cỡ của hàng đợi ưu tiên.
d. Lấy phần tử ở đầu hàng đợi ưu tiên.
e. Đưa phần tử vào hàng đợi ưu tiên.

18) Xây dựng các thao tác trên hàng đợi hai điểm cuối dựa vào danh sách liên kết đơn:

a. Kiểm tra tính rỗng của hàng đợi ưu tiên.
b. Kiểm tra tính đầy của hàng đợi ưu tiên.
c. Lấy kích cỡ của hàng đợi ưu tiên.
d. Lấy phần tử ở đầu hàng đợi ưu tiên.
e. Đưa phần tử vào hàng đợi ưu tiên.

19) Xây dựng tập thao tác trên cây nhị phân dựa vào mảng:
a. Tạo node gốc cho cây.
b. Thêm node lá bên trái cho node x.
c. Thêm node lá bên phải cho node x.
d. Loại node lá trái của node x.
e. Loại node lá phải của node x.
f. Tìm node có giá trị x trên cây.
g. Duyệt cây theo thứ tự trước bằng đệ qui.
h. Duyệt cây theo thứ tự giữa bằng đệ qui.
i. Duyệt cây theo thứ tự sau bằng đệ qui.
j. Duyệt cây theo thứ tự trước không đệ qui.
k. Duyệt cây theo thứ tự trước không đệ qui.
l. Duyệt cây theo thứ tự trước không đệ qui.

20) Xây dựng tập thao tác trên cây nhị phân dựa vào danh sách liên kết:
a. Tạo node gốc cho cây.
b. Thêm node lá bên trái cho node x.
c. Thêm node lá bên phải cho node x.
d. Loại node lá trái của node x.
e. Loại node lá phải của node x.
f. Tìm node có giá trị x trên cây.
g. Duyệt cây theo thứ tự trước bằng đệ qui.
h. Duyệt cây theo thứ tự giữa bằng đệ qui.
i. Duyệt cây theo thứ tự sau bằng đệ qui.
j. Duyệt cây theo thứ tự trước không đệ qui.
k. Duyệt cây theo thứ tự trước không đệ qui.
l. Duyệt cây theo thứ tự trước không đệ qui.

21) Xây dựng tập thao tác trên MAX HEAP:
a. Tạo node cho MAX HEAP.
b. Thêm node vào MAX HEAP.
c. Loại node trên MAX HEAP.
d. Tìm node trên MAX HEAP.
e. Duyệt MAX HEAP.

22) Xây dựng tập thao tác trên MIN HEAP:
a. Tạo node cho MIN HEAP.
b. Thêm node vào MIN HEAP.
c. Loại node trên MIN HEAP.
d. Tìm node trên MIN HEAP.
e. Duyệt MIN HEAP.

23) Xây dựng tập thao tác trên cây nhị phân tìm kiếm dựa vào mảng:
a. Tạo node gốc cho cây nhị phân tìm kiếm.
b. Thêm node vào cây nhị phân tìm kiếm.
c. Loại node trên cây nhị phân tìm kiếm.
d. Tìm node trên cây nhị phân tìm kiếm.

e. Soay phải node trên cây nhị phân tìm kiếm.
f. Soay trái node trên cây nhị phân tìm kiếm.
g. Duyệt cây theo thứ tự trước bằng đệ qui.
h. Duyệt cây theo thứ tự giữa bằng đệ qui.
i. Duyệt cây theo thứ tự sau bằng đệ qui.
j. Duyệt cây theo thứ tự trước không đệ qui.
k. Duyệt cây theo thứ tự trước không đệ qui.
l. Duyệt cây theo thứ tự trước không đệ qui.

24) Xây dựng tập thao tác trên cây nhị phân tìm kiếm dựa vào danh sách liên kết:
a. Tạo node gốc cho cây nhị phân tìm kiếm.
b. Thêm node vào cây nhị phân tìm kiếm.
c. Loại node trên cây nhị phân tìm kiếm.
d. Tìm node trên cây nhị phân tìm kiếm.
e. Soay phải node trên cây nhị phân tìm kiếm.
f. Soay trái node trên cây nhị phân tìm kiếm.
g. Duyệt cây theo thứ tự trước bằng đệ qui.
h. Duyệt cây theo thứ tự giữa bằng đệ qui.
i. Duyệt cây theo thứ tự sau bằng đệ qui.
j. Duyệt cây theo thứ tự trước không đệ qui.
k. Duyệt cây theo thứ tự trước không đệ qui.
l. Duyệt cây theo thứ tự trước không đệ qui.

25) Sử dụng các cấu trúc dữ liệu dưới đây để giải bài toán tìm tập từ và số lần xuất hiện mỗi từ trong file. Phân tích ưu điểm và nhược điểm mỗi cấu trúc dữ liệu:
a. Danh sách liên kết đơn.
b. Danh sách liên kết đơn vòng.
c. Danh sách liên đơn sắp xếp.
d. Danh sách liên kết kép.
e. Danh sách liên kết kép vòng.
f. Danh sách liên kép sắp xếp.
g. Cây nhị phân.
h. Cây nhị phân tìm kiếm.

Cây nhị phân tìm kiếm cân bằng.

26) Xây dựng tập thao tác trên cây nhiều nhánh
a. Xây dựng các thao tác trên cây biểu thức Preoder.
b. Xây dựng các thao tác trên cây biểu thức Postoder.
c. Xây dựng các thao tác trên cây biểu thức Intoder.
d. Xây dựng các thao tác trên cây tiền tố.
e. Xây dựng các thao tác trên cây Top-Down.
f. Xây dựng các thao tác trên cây Btree.

27) Biểu diễn đồ thị:
a. Chuyển đổi biểu diễn đồ thị dưới dạng ma trận kề thành biểu diễn danh sách cạnh.
b. Chuyển đổi biểu diễn đồ thị dưới dạng ma trận kề thành biểu diễn danh sách kề.
c. Chuyển đổi biểu diễn đồ thị dưới dạng danh sách cạnh thành biểu diễn ma trận kề.
d. Chuyển đổi biểu diễn đồ thị dưới dạng danh sách cạnh thành biểu diễn danh sách kề.
e. Chuyển đổi biểu diễn đồ thị dưới dạng danh sách kề thành biểu diễn ma trận kề.
f. Chuyển đổi biểu diễn đồ thị dưới dạng danh sách kề thành biểu diễn danh sách kề.

PHẦN 2. GIẢI THUẬT

28) Các thuật toán sắp xếp
   a) Thuật toán Selection Sort.
   b) Thuật toán Bubble Sort.
   c) Thuật toán Insertion Sort.
   d) Thuật toán Quick Sort.
   e) Thuật toán Merge Sort.
   f) Thuật toán Sell-Sort.
   g) Thuật toán Radix Sort.
   h) Thuật toán Shaker Sort.

29) Các thuật toán tìm kiếm
   a) Tìm kiếm tuyến tính.
   b) Tìm kiếm nhị phân.
   c) Tìm kiếm nội suy.
   d) Tìm kiếm Fibonacci.
   e) Tìm kiếm trên cây nhị phân.
   f) Tìm kiếm trên cây tìm kiếm.
   g) Tìm kiếm dựa vào hàm băm (Hash Function).

30) **Thuật toán sinh.** Sử dụng thuật toán sinh, hãy thực hiện:
   a. Liệt kê các xâu nhị phân có độ dài n.
   b. Liệt kê các tổ hợp chập k của 1,..,n.
   c. Liệt kê các hoán vị của 1, 2,.., n.
   d. Liệt kê các cách chia số n thành tổng các số tự nhiên nhỏ hơn n.
   e. Liệt kê các dãy con của dãy số An sao cho tổng các phần tử của dãy con đúng bằng K.
   f. Liệt kê các dãy con K phần tử của dãy số An sao cho tổng các phần tử của dãy con đúng bằng M.
   g. Giải bài toán cái túi bằng thuật toán sinh.
   h. Giải bài toán người du lịch bằng thuật toán sinh.
   i. Giải bài toán n quân hậu bằng thuật toán sinh.

31) **Thuật toán đệ qui.** Sử dụng thuật toán đệ qui, hãy thực hiện:
   a. Đổi số tự nhiên n thành số ở hệ cơ số b.
   b. Tìm tổng tất cả các chữ số của số tự nhiên n.
   c. Tìm tổng tất cả các phần tử của dãy số An.
   d. Đảo xâu có độ dài n.
   e. Thuật toán Selection Sort đệ qui.
   f. Thuật toán Insertion Sort đệ qui.
   g. Thuật toán Sequential Search đệ qui.
   h. Tìm kích cỡ của cây nhị phân bằng đệ qui.
   i. Xác định hai cây nhị phân giống nhau bằng đệ qui.
   j. Xác định độ cao của cây nhị phân bằng đệ qui.
   k. Tạo nên cây phản chiếu bằng đệ qui.
   l. Kiểm tra cây tổng bằng đệ qui.

32) **Thuật toán quay lui.** Sử dụng thuật toán quay lui, hãy thực hiện:
   a. Liệt kê các xâu nhị phân có độ dài n.

b. Liệt kê các tổ hợp chập k của 1,..,n.
c. Liệt kê các hoán vị của 1, 2,.., n.
d. Giải bài toán cái túi bằng thuật toán quay lui.
e. Giải bài toán người du lịch bằng thuật toán quay lui.
f. Giải bài toán n quân hậu bằng thuật toán quay lui.
g. Giải bài toán Mã đi tuần bằng thuật toán quay lui.
h. Giải bài toán "Tug of War" bằng thuật toán quay lui.
i. Giải bài toán Sudoku bằng thuật toán quay lui.
j. Giải bài toán "Đường đi con rắn trong mê cung" bằng thuật toán quay lui

33) **Thuật toán tham lam**. Sử dụng thuật toán tham lam giải các bài toán dưới đây:
a. Giải bài toán n-ropes.
b. Giải bài toán "Activity Selection" không trọng số.
c. Giải bài toán "Job Sequences" có trọng số.
d. Giải bài toán cái túi bằng phương pháp tham lam.
e. Giải bài toán người du lịch bằng phương pháp tham lam.
f. Giải bài toán tìm thời gian nhỏ nhất thực thi tất cả các jop với ràng buộc cho trước.
g. Giải bài toán sắp đặt lại các ký tự trong xâu sao cho tất cả các kỹ tự giống nhau đều có khoảng cách d cho trước.
h. Sinh mã Hufffman bằng phương pháp tham lam.
i. Giải bài toán Graph Coloring bằng phương pháp tham lam.
j. Thuật toán PRIM tìm cây khung nhỏ nhất bằng phương pháp tham lam.
k. Thuật toán Kruskal tìm cây khung nhỏ nhất bằng phương pháp tham lam.
l. Thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại trên đồ thị bằng phương pháp tham lam.

34) **Thuật toán chia và trị.** Sử dụng thuật toán chia và trị giải các bài toán dưới đây:
a. Cài đặt thuật toán nhân nhanh Karatsuba.
b. Cài đặt thuật toán Strassen nhân hai ma trận.
c. Cài đặt thuật toán nhân hai đa thức.
d. Tìm dãy con có tổng lớn nhất trong một dãy số.
e. Cài đặt thuật toán Binary-Search .
f. Cài đặt thuật toán Quick-Sort .
g. Cài đặt thuật toán Merge-Sort .

35) **Thuật toán qui hoạch động**. Sử dụng thuật toán qui hoạch động giải các bài toán dưới đây:
a. Nhân hai ma trận.
b. Đếm số cách chia số tự nhiên n thành tổng các số nhỏ hơn n.
c. Đếm các xâu nhị phân có độ dài n không chứa dãy k số 1 liên tiếp.
d. Đếm số các xâu nhị phân độ dài n chứa ít nhất một dãy k số 1 liên tiếp.
e. Đếm số các nghiệm nguyên không âm của phương trình x1 + ..+Xn =k.
f. Tìm dãy con trong dãy số có tổng các phần tử bằng K.
g. Giải bài toán cái túi.
h. Giải bài toán cho thuê máy.
i. Giải bài toán tìm dãy con tăng dài nhất.
j. Giải bài toán biến đổi xâu.
k. Tìm xâu con chung dài nhất giữa xâu X và xâu Y

36) **Thuật toán đối sánh mẫu**:
a. Thuật toán Knuth-Morris-Partt.
b. Thuật toán Karp-Rabin.

c. Thuật toán Shift-or.
d. Thuật toán Simon.
e. Thuật toán Colussi.
f. Thuật toán Boyer-Moore.
g. Thuật toán Not So Naive.
h. Thuật toán Zhu-Takaoka.
i. Thuật toán Raita.
j. Thuật toán KMP Search.
k. Thuật toán Alpha Skip Search.

37) Sử dụng ngăn xếp xây dựng thuật toán tìm kiếm theo chiều sâu bắt đầu tại đỉnh u (DFS(u)) và thực hiện:
a) Biểu diễn đồ thị bằng danh sách liên kết đơn.
b) Duyệt tất cả các đỉnh của đồ thị bằng thuật toán tìm DFS(u).
c) Xác định các thành phần liên thông của đồ thị bằng thuật toán DFS(u).
d) Tìm đường đi từ đỉnh s đến đỉnh t dựa thuật toán bằng thuật toán DFS(u).
e) Kiểm tra tính liên thông mạnh của đồ thị dựa bằng thuật toán DFS(u).
f) Xác định các đỉnh trụ của đồ thị bằng thuật toán DFS (u).
g) Xác định các cạnh cầu của đồ thị bằng thuật toán DFS (u).
h) Xác định các đỉnh rẽ nhánh của cặp đỉnh u,v của đồ thị bằng thuật toán DFS(u).
i) Kiểm tra tính định chiều được của đồ thị bằng thuật toán DFS(u).
j) Xây dựng cây khung của đồ thị bằng thuật toán DFS(u).

38) Sử dụng ngăn hàng đợi dựng thuật toán tìm kiếm theo chiều rộng bắt đầu tại đỉnh u (BFS(u)) và thực hiện:
a) Biểu diễn đồ thị bằng danh sách liên kết đơn.
b) Duyệt tất cả các đỉnh của đồ thị bằng thuật toán tìm BFS(u).
c) Xác định các thành phần liên thông của đồ thị bằng thuật toán BFS(u).
d) Tìm đường đi từ đỉnh s đến đỉnh t dựa thuật toán bằng thuật toán BFS(u).
e) Kiểm tra tính liên thông mạnh của đồ thị dựa bằng thuật toán BFS(u).
f) Xác định các đỉnh trụ của đồ thị bằng thuật toán BFS (u).
g) Xác định các cạnh cầu của đồ thị bằng thuật toán BFS (u).
h) Xác định các đỉnh rẽ nhánh của cặp đỉnh u,v của đồ thị bằng thuật toán BFS(u).
i) Kiểm tra tính định chiều được của đồ thị bằng thuật toán BFS(u).
j) Xây dựng cây khung của đồ thị bằng thuật toán BFS(u).

39) Cho đồ thị (có hướng + vô hướng) G = <V,E> được biểu diễn dưới dạng danh sách kề. Hãy thực hiện :
a) Biểu diễn đồ thị bằng danh sách liên kết đơn.
b) Tìm một chu trình Euler trên đồ thị vô hướng bắt đầu tại đỉnh $u \in V$ dựa vào ngăn xếp.
c) Tìm một chu trình Euler trên đồ thị có hướng bắt đầu tại đỉnh $u \in V$ dựa vào ngăn xếp.
d) Tìm một đường đi Euler trên đồ thị vô hướng bắt đầu tại đỉnh $u \in V$ dựa vào ngăn xếp.
e) Tìm một đường đi Euler trên đồ thị có hướng bắt đầu tại đỉnh $u \in V$ dựa vào ngăn xếp.
f) Liệt kê tất cả chu trình Hamilton trên đồ thị bắt đầu tại đỉnh $u \in V$.
g) Liệt kê tất cả đường đi Hamilton trên đồ thị bắt đầu tại đỉnh $u \in V$.

40) Cho đồ thị (có hướng + vô hướng) G = <V,E> có trọng số được biểu diễn dưới dạng danh sách trọng số. Hãy thực hiện :
   a) Tìm một cây khung nhỏ nhất trên đồ thị vô hướng có trọng số bằng thuật toán Kruskal.
   b) Tìm một cây khung nhỏ nhất trên đồ thị vô hướng có trọng số bằng thuật toán PRIM.
   c) Tìm đường đi ngắn nhất từ đỉnh u đến tất cả các đỉnh còn lại trên đồ thị có trọng số không âm bằng thuật toán Dijkstra.
   d) Tìm đường đi ngắn nhất từ đỉnh u đến tất cả các đỉnh còn lại trên đồ thị có không tồn tại chu trình âm bằng thuật toán Bellman-Ford.
   e) Tìm đường đi ngắn nhất từ đỉnh u đến tất cả các đỉnh còn lại trên đồ thị có không tồn tại chu trình âm bằng thuật toán Floyed.

PHẦN 3. MỘT SỐ BÀI TẬP LÀM THÊM

1. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:
   (i) K là số có N chữ số (N≤15);
   (ii) K là số nguyên tố;
   (iii) K là số thuận nghịch (k là số thuận nghịch nếu đọc xuôi hay đọc ngược các chữ số của k ta đều nhận được một số như nhau. Ví dụ số: 30303);
   (iv) Biểu diễn của K ở hệ cơ số B (B bất kỳ được nhập từ bàn phím cũng là một số thuận nghịch. Ví dụ số k=30303 có biểu diễn ở hệ cơ số 8 là 73137 cũng là một số thuận nghịch;

2. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:
   (i) K là số có N chữ số;
   (ii) K là số nguyên tố;
   (iii) Đảo ngược các chữ số trong của K cũng là một số nguyên tố;
   (iv) Tổng các chữ số của K cũng là một số nguyên tố;
   (v) Mỗi chữ số trong K cũng là những số nguyên tố.

3. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:
   (i) K là số có 5 chữ số;
   (ii) K là số nguyên tố;
   (iii) Mỗi chữ số của K cũng là những số nguyên tố;
   (iv) Tổng các chữ số của K là một số thuận nghịch hai chữ số;
   (v) Tích các chữ số của K là một số thuận nghịch ba chữ số.

4. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:
   (i) K là số có 5 chữ số;
   (ii) K là số nguyên tố;
   (iii) Mỗi chữ số của K cũng là các số nguyên tố;
   (iv) Tổng các chữ số của K là một số chia hết cho P (P được nhập từ bàn phím);
   (v) Tích các chữ số của K là một số chia hết cho Q (Q được nhập từ bàn phím);
   (vi) Các chữ số của K không chứa số R (được nhập từ bàn phím).

5. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:
   (i) K là số có 5, 7 hoặc 9 chữ số;
   (ii) K là số thuận nghịch;
   (iii) Tổng các chữ số của K là một số chia hết cho P (P được nhập từ bàn phím);
   (iv) Tích các chữ số của K là một số chia hết cho Q (Q được nhập từ bàn phím);
   (v) Các chữ số của K không chứa số R (được nhập từ bàn phím).

6. Cho số tự nhiên N, B được nhập từ bàn phím (N≥10000 ; B≥255). Hãy viết chương trình thực hiện:

(i) Tính tổng các chữ số của N ;
(ii) Phân tích N thành tích các thừa số nguyên tố ;
(iii) Biểu diễn N ở hệ cơ số B ;
(iv) Liệt kê các số hoàn hảo nhỏ hơn N ;

7. Số điện thoại di động của một hãng viễn thông được đánh số theo qui cách 091N. XXX.XXX. Trong đó, N là số từ 2 đến 8, X là một số từ 0 đến 9. Ta định nghĩa các loại số điện thoại sau:

- **Số điện thoại loại I (Loại I):** Là những số có sáu số cuối cùng của nó tạo thành một số thuận nghịch sáu chữ số. Ví dụ số: 0913.558855.
- **Số điện thoại loại II (Loại II):** Là những số điện thoại Loại I có tổng sáu chữ số cuối cùng là một số chia hết cho 10 . Ví dụ số: 0913.104.401 (1+0+4+4+0+1=10).
- **Số điện thoại loại III (Loại III):** Là những số điện thoại Loại II có sáu chữ số cuối cùng không chứa bất kỳ một số 0 nào. Ví dụ số: 0913. 122.2211.

Hãy viết chương trình thực hiện:

- Liệt kê tất cả các số điện thoại Loại I không chứa các số điện thoại Loại II. Ghi lại các số Loại I vào file Loai1.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.
- Liệt kê tất cả các số điện thoại Loại II không chứa các số điện thoại Loại III. Ghi lại các số Loại II vào file Loai2.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.
- Liệt kê tất cả các số điện thoại Loại III. Ghi lại các số Loại III vào file Loai3.out theo từng dòng, mỗi dòng không quá 8 số điện thoại.

8. 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). Hãy viết chương trình tìm cách dịch chuyển S thành T thỏa mãn đồng thời những điều kiện dưới đây:

a) 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);
b) 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);
c) Số các bước dịch chuyển là ít nhất.

Hai số nguyên tố có 4 chữ số S và T được ghi trong file nguyento.in; hai số được ghi cách nhau một vài khoảng trống. Dãy các phép dịch chuyển thỏa mãn điều kiện ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các phép dịch chuyển nhỏ nhất;
- Dòng kế tiếp ghi lại các số nguyên tố được dịch chuyển theo chiều ngược lại từ T về S theo từng bước.

Ví dụ với S = 1033, T = 8179 trong file nguyento.in dưới đây sẽ cho ta file ketqua.out như sau:

nguyento.in	*ketqua.out*
1033 8179	6
	8179 8779 3779 3739 3733 1733 **1033**

9. Một chuỗi ký tự S = (s_1, s_2, .., s_n) được gọi là xâu AB độ dài n nếu với mọi $s_i \in S$ thì s_i = 'A' hoặc s_i = 'B'. Ví dụ xâu S ="ABABABAB" là một chuỗi AB độ dài 8. Cho số tự nhiên N và số tự nhiên K (1≤K<N≤32 được nhập từ bàn phím), hãy viết chương trình liệt kê tất cả các chuỗi AB có độ dài N chứa duy nhất một dãy K kí tự A liên tiếp. Các xâu AB tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các xâu AB thỏa mãn yêu cầu bài toán;
- Những dòng kế tiếp, mỗi dòng ghi lại một xâu AB có độ dài N chứa duy nhất một dãy K kí tự A liên tiếp.

Ví dụ với N =5, K = 3 sẽ cho ta 5 xâu kí tự AB chứa duy nhất dãy 3 kí tự A liên tiếp như sau.

ketqua.out

5
AAABA
AAABB
ABAAA
BAAAB
BBAAA

10. Một dãy số tự nhiên bất kỳ A_N = {a_1, a_2,.., a_N} được gọi là một *đường nguyên tố bậc K* nếu tổng *K* phần tử liên tiếp bất kỳ của dãy số A_N là một số nguyên tố (*K≤N*). Ví dụ dãy số A_N = {*3, 27, 7, 9, 15*} là một *đường nguyên tố bậc 3*.

Cho dãy số A_N được tổ chức trong file dayso.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên *N* , *K* tương ứng với số các phần tử dủa dãy số A_N (*N≤10*) và bậc *của đường nguyên tố*. Hai số được viết cách nhau một vài khoảng trống;
- Dòng kế tiếp ghi lại *N* số tự nhiên là các phần tử của dãy số A_N (a_i*≤16000; i=1, 2, .., N*). Hai số khác nhau được ghi cách nhau một vài khoảng trống.

Hãy viết chương trình liệt kê tất cả các *đường nguyên tố bậc K* có thể có được tạo ra bằng cách tráo đổi các phần tử khác nhau của dãy số A_N. Các *đường nguyên tố bậc K* tìm được ghi lại trong file ketqua.out theo khuông dạng:

- Dòng đầu tiên ghi lại số các đường nguyên tố tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một *đường nguyên tố bậc K* được tạo ra theo nguyên tắc nêu trên. Hai phần tử khác nhau của một *đường nguyên tố bậc K* được viết cách nhau một vài khoảng trống.

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

dayso.in

```
5 3
3 7 9 15 27
```

ketqua.out

```
4
3 27 7 9 15
15 9 7 3 27
15 9 7 27 3
27 3 7 9 15
```

11. Dãy kí tự $S=(s_1,s_2,..,s_n): s_i='A' \vee s_i='B'; i=1,2,..,n.$ được gọi là dãy AB độ dài N. Dãy *S* độ dài *N* được gọi là dãy *AB* bậc *K* theo *A* và bậc *M* theo *B* nếu *S* có đúng *K* dãy, mỗi dãy có *K* kí tự *A* liên tiếp và *M* dãy, mỗi dãy có *M* kí tự *B* liên tiếp. Ví dụ dãy *S =”AAAAAB”* là dãy *AB* độ dài *6* có bậc *3* theo *A* và bậc *1* theo *B* vì *S* có đúng *3* dãy chứa *3* kí tự *A* liên tiếp là (s_0,s_1,s_2), (s_1,s_2,s_3), (s_2,s_3,s_4) và có *1* dãy có *1* kí tự *B* là (s_5).

Cho số tự nhiên *N, M, K* (*1≤K, M≤N<32* được nhập từ bàn phím), hãy viết chương trình liệt kê tất cả các dãy *AB* bậc *K* theo *A* và bậc *M* theo *B* có độ dài *N*. Các dãy tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các dãy *AB* bậc *K* theo *A* và bậc *M* theo *B* có độ dài *N*;
- Những dòng kế tiếp ghi lại mỗi dòng một dãy *AB* bậc *K* theo *A* và bậc *M* theo *B* có độ dài *N*;

Ví dụ với N=8, K=2, M=3 sẽ cho ta 2 dãy AB bậc 2 theo A và bậc 3 theo B như dưới đây.

ketqua.out

2

B B B B B A A A

A A A B B B B B

12. Cho số tự nhiên N, M, K (1≤K, M<N≤100) được nhập từ bàn phím. Hãy viết chương trình liệt kê tất cả các xâu nhị phân có độ dài N chứa duy nhất một dãy K bít 1 liên tiếp và duy nhất một dãy M bít 0 liên tiếp. Các xâu nhị phân tìm được thỏa mãn yêu cầu của bài toán được ghi lại trong file *ketqua.out* theo khuôn dạng:

- Dòng đầu tiên ghi lại số các xâu nhị phân thỏa mãn yêu cầu của bài toán;
- Những dòng kế tiếp, mỗi dòng ghi lại một xâu nhị phân độ dài N thỏa mãn yêu cầu của bài toán. Hai bít khác nhau của xâu nhị phân được viết cách nhau một vài khoảng trống.

Ví dụ với N=6, K=2, M=3 sẽ cho ta 4 xâu nhị phân thỏa mãn điều kiện được ghi lại trong file ketqua.out như sau:

ketqua.out

4

0 0 0 1 1 0

0 1 1 0 0 0

1 0 0 0 1 1

1 1 0 0 0 1

13. Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp 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
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

14. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp 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;
- 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.

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 1 9 3 5 3 6 4	12 0 0 1 1

15. Cho dãy gồm n số tự nhiên phân biệt $a_1, a_2, .., a_n$ và số tự nhiên *B*. Hãy 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  30  35
```

Ketqua.Out
```
  3
  0  0  0  0  1  0  0
  1  0  0  1  0  0  0
  0  1  1  0  0  0  0
```

16. 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 biểu diễn dữ liệu và thuật toán thích hợp viết chương trình liệt kê tất cả các dãy con *K* phần tử tăng dần 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
```
5   3
2   5   15   10   20
```

ketqua.out
```
7
2   5   15
2   5   10
2   5   20
2   15  20
2   10  20
5   15  20
5   10  20
```

17. Cho dãy $A_N = \{a_1, a_2, .., a_N\}$ gồm N số tự nhiên phân biệt. Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình liệt kê tất cả các dãy con K phần tử của dãy số A_N ($K \le N$) sao cho tổng các phần tử của dãy con đó là một số đúng bằng B. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại ba số N, K, B. Các số được ghi cách nhau một vài khoảng trống;
- Những dòng kế tiếp ghi lại N số của dãy số A_N. Hai số khác nhau được ghi cách nhau một vài khoảng trống.

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

- Dòng đầu tiên ghi lại số các dãy con tìm được;
- Những dòng kế tiếp, mỗi dòng ghi lại một dãy con thỏa mãn điều kiện. Hai số khác nhau được ghi cách nhau một vài khoảng trống.

Ví dụ dưới đây sẽ minh họa cho khuôn dạng của file dayso.in và ketqua.out

dayso.in

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

ketqau.out

```
2
5  20  25
10  15  25
```

18. 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 biểu diễn dữ liệu và thuật toán 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

```
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
```

ketqua.out

```
82
1    1    10
2    6    12
3    4    16
4    5    21
5    3    11
6    2    12
```

19. Cho dãy gồm *N* số nguyên phân biệt A[] = {a_1, a_2, .., a_N } và số tự nhiên *K* (*K≤N≤100*). Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình liệt kê tất cả các dãy con *K* phần tử giảm dần 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 giảm 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 giảm 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

```
5 3
20 10 15 5 3
```

ketqua.out

```
7
20 10 5
20 10 3
20 15 5
20 15 3
20 5 3
10 5 3
15 5 3
```

20. Cho ma trận vuông C = (c_{ij}) cấp N (*1≤ i, j ≤ N≤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 biểu diễn dữ liệu và thuật toán 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 lớn 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 viec.in và ketqua.out:

matran.in

```
6
10 14 27 29 18 27
34 20 19 34 16 12
57 37 40 57 11 19
29 21 46 26 21 18
27 37 11 21 21 37
55 12 15 48 37 35
```

ketqua.out

```
238
1 6 27
2 1 34
3 4 57
4 3 46
5 2 37
6 5 37
```

21. Một người du lịch cần đi qua *N* thành phố (*N≤100*). Xuất phát tại thành phố số 1, người du lịch muốn qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần rồi trở lại thành phố ban đầu. Biết chi phí đi lại từ thành phố thứ *i* đến thành phố thứ *j* là c_{ij}. Hãy viết chương trình tìm hành trình có chi phí nhỏ nhất cho người du lịch.

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

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

Hành trình có chi phí nhỏ nhất tìm được ghi lại trong file **cuctieu.out** theo khuôn dạng:

- Dòng đầu tiên ghi lại giá trị chi phí nhỏ nhất của hành trình tìm được;
- Dòng kế tiếp, ghi lại hành trình của người du lịch. Hai thành phố khác nhau của hành trình được ghi cách nhau một vài khoảng trống.

Ví dụ về dưới đây sẽ minh họa cho file **chiphi.in** và **cuctieu.out** của bài toán.

chiphi.in

```
5
00  48  43  58  31
20  00  30  63  22
29  64  00  04  17
06  19  02  00  08
01  28  07  18  00
```

cuctieu.out

```
81
1  5  3  4  2  1
```

22. Một người du lịch cần đi qua *N* thành phố (*N≤100*). Xuất phát tại thành phố số 1, người du lịch muốn qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần rồi trở lại thành phố ban đầu. Biết chi phí đi lại từ thành phố thứ *i* đến thành phố thứ *j* là c_{ij}. Hãy viết chương trình tìm hành trình có chi phí lớn nhất cho người du lịch.

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

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

Hành trình có chi phí lớn nhất tìm được ghi lại trong file **cucdai.out** theo khuôn dạng:

- Dòng đầu tiên ghi lại giá trị chi phí lớn nhất của hành trình tìm được;
- Dòng kế tiếp, ghi lại hành trình của người du lịch. Hai thành phố khác nhau của hành trình được ghi cách nhau một vài khoảng trống.

Ví dụ về dưới đây sẽ minh họa cho file **chiphi.in** và **cucdai.out** của bài toán.

chiphi.in

```
5
00  48  43  58  31
20  00  30  63  22
29  64  00  04  17
06  19  02  00  08
01  28  07  18  00
```

cucdai.out

```
179
1  3  2  4  5  1
```

23. Cho ma trận vuông $C_{i,j}$ cấp N (*$1 \le i, j \le N \le 100$*) gồm N^2 số tự nhiên và số tự nhiên K(*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 và K. Hai số được viết cách nhau một vài khoảng trống;
- N dòng kế tiếp ghi lại ma trận vuông Ci,j; Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình lấy mỗi hàng, mỗi cột duy nhất một phần tử của ma trận C sao cho tổng các phần tử này đúng bằng K. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các nghiệm tìm được của bài toán.
- Những dòng kế tiếp, mỗi dòng ghi lại N số là một phương án của bài toán, số thứ i ghi lại giá trị j tương ứng với chỉ số cột của phần tử được lựa chọn. Các số được viết cách nhau một vài khoảng trống.

Ví dụ về file viec.in và ketqua.out:

matran.in

```
6  180
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
```

ketqua.out

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

24. Một người du lịch cần đi qua *N* thành phố (*$N \le 100$*). Xuất phát tại thành phố số 1, người du lịch muốn qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần rồi trở lại thành phố ban đầu. Biết chi phí đi lại từ thành phố thứ *i* đến thành phố thứ *j* là c_{ij}. Hãy viết chương trình tìm hành trình có chi phí lớn nhất cho người du lịch.

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

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

Hành trình có chi phí lớn nhất tìm được ghi lại trong file **cucdai.out** theo khuôn dạng:

- Dòng đầu tiên ghi lại giá trị chi phí lớn nhất của hành trình tìm được;
- Dòng kế tiếp, ghi lại hành trình của người du lịch. Hai thành phố khác nhau của hành trình được ghi cách nhau một vài khoảng trống.

Ví dụ về dưới đây sẽ minh họa cho file **chiphi.in** và **cucdai.out** của bài toán.

chiphi.in

```
5
00  48  43  58  31
20  00  30  63  22
29  64  00  04  17
06  19  02  00  08
01  28  07  18  00
```

cucdai.out

```
179
1  3  2  4  5  1
```

25. 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
```

26. Cho đồ thị vô hướng G =<V,E> được biểu diễn dưới dạng danh sách kề trong file *dske.in* theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh và số cạnh 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 ghi cách nhau một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình kiểm tra và đưa ra kết quả sau:

a) Đưa ra thông báo "*G là đồ thị Euler*" và một chu trình Euler của đồ thị nếu G là đồ thị Euler;
b) Đưa ra thông báo "*G là đồ thị nửa Euler*" và một đường đi Euler của đồ thị nếu G là đồ thị nửa Euler;
c) "*G không là đồ thị Euler/Semi-Euler*" nếu G liên thông nhưng không là đồ thị Euler hoặc nửa Euler;
d) "*G không liên thông*" nếu G không liên thông.

Ví dụ với đồ thị 4 đỉnh được cho trong file dske.in dưới đây sẽ cho ta kết quả "*G là đồ thị Euler*" với chu trình Euler tương ứng là: 1- 2 - 4 - 3 - 1.

dske.in

4
2 3
1 4
1 4
2 3

27. Cho đồ thị có hướng G =<V,E> được biểu diễn dưới dạng danh sách kề trong file *dske.in* theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh và số cạnh 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 ghi cách nhau một hoặc vài kí tự trống, đỉnh không có cạnh nối với nó (đỉnh cô lập) được ghi giá trị 0.

Hãy sử dụng biểu diễn dữ liệu và thuật toán thích hợp viết chương trình kiểm tra và đưa ra kết quả sau:

a) Đưa ra thông báo *"G là đồ thị Euler"* và một chu trình Euler của đồ thị nếu G là đồ thị Euler;
b) Đưa ra thông báo *"G là đồ thị nửa Euler"* và một đường đi Euler của đồ thị nếu G là đồ thị nửa Euler;
e) *"G không là đồ thị Euler"* nếu G liên thông mạnh nhưng không là đồ thị Euler;
d) *"G không liên thông mạnh "* nếu G liên thông yếu nhưng không là đồ thị nửa Euler.

Ví dụ với đồ thị 4 đỉnh được cho trong file dske.in dưới đây sẽ cho ta kết quả "*G là đồ thị Euler*" với chu trình Euler tương ứng là: 1- 2 - 4 - 3 - 1.

dske.in

4
2
4
1
3

28. Cho đồ thị vô hướng G =<V,E>, trong đó V là tập đỉnh, E là tập cạnh. Ta gọi cạnh e∈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	4

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

29. Cho đồ thị vô hướng G =<V,E>, trong đó V là tập đỉnh, E là tập cạnh. Ta gọi đỉnh v∈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

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

Dinhtru.out

3		
1	2	3

30. 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'). Cho file văn bản DATA.IN. Hãy sử dụng danh sách liên kết kép tìm tập các từ và số lần xuất xuất hiện của mỗi từ trong file văn bản DATA.IN. Tập từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ xuất hiện trong file DATA.IN;
- K dòng kế tiếp, mỗi dòng ghi lại một từ và số lần xuất hiện của từ đó trong file DATA.IN.

Ví dụ dưới đây sẽ minh họa cho file DATA.IN và Ketqua.out của bài toán.

Data.in

A	AB	AC	AD	AE	A	AB	AE	AF

Ketqua.out

6	
A	2
AB	2
AC	1
AD	1
AE	2
AF	1

31. 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’). Cho file văn bản DATA.IN. Hãy sử dụng cây nhị phân tìm kiếm tìm tập các từ và số lần xuất xuất hiện của mỗi từ trong file văn bản DATA.IN. Tập từ tìm được ghi lại trong file Ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên K là số từ xuất hiện trong file DATA.IN;
- K dòng kế tiếp, mỗi dòng ghi lại một từ và số lần xuất hiện của từ đó trong file DATA.IN.

Ví dụ dưới đây sẽ minh họa cho file DATA.IN và Ketqua.out của bài toán.

```
Data.in
A   AB  AC  AD  AE  A   AB  AE  AF
```

```
Ketqua.out
6
A       2
AB      2
AC      1
AD      1
AE      2
AF      1
```

32. 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   *
3   2   +   3   2   -   *
```

```
ketqua.out
4
5
1
1
6
5
```

33. 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 biểu diễn dữ liệu và thuật toán thích hợp viết chương 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

```
4
10 30000
5 1000
3 2
3 0
```

dathuc2.in

```
5
8 20000
3 1000
3 500
7 100
6 1
```

ketqua.out

```
8
10 30000
8 20000
8 1000
3 500
7 100
3 2
6 1
3 0
```

34. 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 biểu diễn dữ liệu và thuật toán thích hợp 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

```
4
10 30000
5 1000
3 2
3 0
```

dathuc2.in

```
5
8 20000
3 1000
3 500
7 100
6 1
```

ketqua.out

```
8
10 30000
-8 20000
3 1000
-3 500
-7 100
3 2
-6 1
3 0
```

35. 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
```

36. 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
```

37. 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
```

38. 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
```

39. 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 viết chương trình tìm cách sắp xếp 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

40. 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
```

41. 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
```

42. 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
```

43. 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
```

44. 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
```

45. 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
```

46. 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
```

47. 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
```

48. 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
```

49. 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"?

Ví dụ với đồ thị dưới đây sẽ cho ta đường đi Euler : 2 - 1 - 3 - 2 - 4 - 3

dothi.in

```
5
0   1   1   0
1   0   1   1
1   1   0   1
0   1   1   0
```

50. 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

```
5
0 1 0 0
0 0 1 1
0 0 0 1
1 0 0 0
```

51. 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

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

mtke.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
```

dscanh.out

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

52. 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

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

mtke.out

```
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
```

dscanh.out

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