# BÀI THỰC HÀNH SỐ 2

**Bài 1.** Viết chương trình kiểm tra một đồ thi cho trước có phải là đồ thị đặc biệt dạng Đồ thị hình bánh xe (*Wheel graph*) hay không? Nếu phải thì cho biết số cạnh của đồ thị này. Nếu không thì trả trả lời: "NO".

Mô tả kỹ thuật:

- File "*Wheelgraph.inp*" chứa dữ liệu đầu vào được định dạng như sau:
  + Dòng đầu tiên là số N: là cấp của một ma trận vuông.
  + N dòng tiếp theo là ma trận mô tả đồ thị đầu vào.
- Kết quả lưu ở file có tên *Wheelgraph*.out".

*Gợi ý làm bài:*

- Xem lại lý thuyết về đồ thị đặc biệt ở slide bài giảng trên elearning.

**Bài 2.** Cho trước một đồ thị đặc biệt G thuộc 1 trong 4 loại sau:

1) Đồ thì đầy đủ (*Complete graph*)
2) Đồ thị vòng (*Cycle graph*)
3) Đồ thị hình bánh xe (*Wheel graph*)
4) Đồ thì đều (*Regular graph*)

Yêu cầu: Hãy lập trình xác định xem G thuộc loại nào trong 4 loại trên và cho biết số cạnh của G.

Mô tả kỹ thuật:

- File *Dothidacbiet.inp* mô tả dữ liệu của đồ thị G, được định dạng như sau:
  - Dòng đầu tiên là số N.
  - N dòng tiếp theo là ma trận cấp N.
- File *Dothidacbiet.out* lưu trữ kết quả của chương trình:
  - Nếu dữ liệu đầu vào không mô tả chính xác 1 trong 4 loại đồ thị nêu trên thì xuất kết quả "DỮ LIỆU ĐẦU VÀO KHÔNG CHÍNH XÁC".
  - Ngược lại, trả lời:
    - Dòng đầu tiên là 1 số nguyên dương: có giá trị từ 1 – 4 tương ứng với 4 loại đồ thì trên.
    - Dòng thứ 2 là số cạnh của đồ thị tìm được.

**Bài 3.** Cho trước đồ thị vô hướng G = (V, E) cấp N (số nguyên dương). Hãy viết chương trình tìm một đường đi từ đỉnh u tới đỉnh v cho trước.

Input:

- Tên file: *pathvohuong.inp*.
- Trong file này, dòng đầu tiên là số N, N dòng tiếp theo là ma trận kề cấp N.

- Dòng sau ma trận là 2 đỉnh u và v, mỗi đỉnh cách nhau bởi khoảng trắng.

Output: in kết quả ra file "*pathvohuong.out*"

- Kết quả được in ra trên một dòng duy nhất, mỗi đỉnh của đường đi cách nhau bởi 1 mũi tên (→).
- Nếu không có đường đi giữa u và v thì trả lời "NO PATH".

**Bài 4.** Cho trước đồ thị có hướng G = (V, E) cấp N (số nguyên dương). Hãy viết chương trình tìm một đường đi từ đỉnh u tới đỉnh v cho trước.

Input:

- Tên file: *pathcohuong.inp*.
- Trong file này, dòng đầu tiên là số N, N dòng tiếp theo là ma trận kề cấp N.
- Dòng sau ma trận là 2 đỉnh u và v, mỗi đỉnh cách nhau bởi khoảng trắng.

Output: in kết quả ra file "*pathcohuong.out*"

- Kết quả được in ra trên một dòng duy nhất, mỗi đỉnh của đường đi cách nhau bởi 1 mũi tên (→).

Nếu không có đường đi giữa u và v thì trả lời "NO PATH".

**Bài 5.** Cho trước một đồ thị vô hướng G dưới dạng một ma trận kề có bậc N. Hãy kiểm xem G có liên thông hay không.

Input:

- Tên file: *lienthong.inp*.
- Trong file này, dòng đầu tiên là số N, N dòng tiếp theo là ma trận kề cấp N.

Output:

- Kết quả được in ra file *lienthong.out,* in ra số 1 nếu G liên thông, số 0 ngược lại.

**Bài 6.** Cho trước một đồ thị có hướng G dưới dạng một ma trận kề có bậc N. Hãy kiểm xem G có liên thông mạnh hay không.

Input:

- Tên file: *lienthongmanh.inp*.
- Trong file này, dòng đầu tiên là số N, N dòng tiếp theo là ma trận kề cấp N.

Output:

- Kết quả được in ra file *lienthongmanh.out,* in ra số 1 nếu G liên thông mạnh, số 0 ngược lại.

**Bài 7.** Cho trước một đồ thị G dưới dạng một ma trận kề có bậc N. Hãy kiểm xem trong G có đỉnh cô lập hay không.

Input:

- Tên file: *dinhcolap.inp*.
- Trong file này, dòng đầu tiên là số N, N dòng tiếp theo là ma trận kề cấp N.

Output:

Kết quả được in ra file *dinhcolap.out,* nếu có đỉnh cô lập thì in ra tên đỉnh đó (*trường hợp có nhiều đỉnh cô lập thì in ra đỉnh đầu tiên tìm gặp*), ngược lài thì in ra số -1.

./.