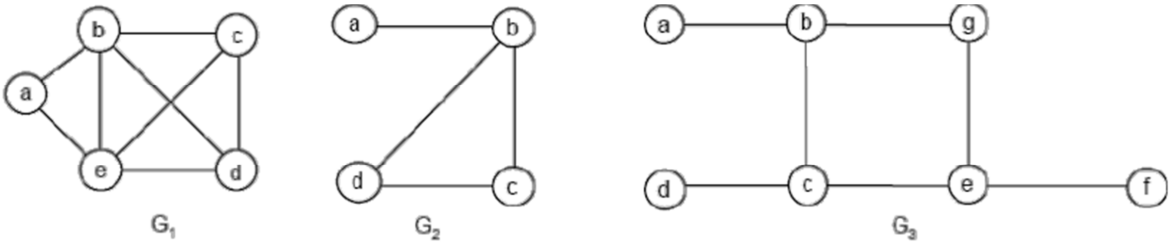
Tìm chu trình, đường đi Hamilton

I. Nhắc lại các khái niệm

- Dây chuyền (đường đi) Hamilton là dây chuyền (đường đi) đi qua tất cả các đỉnh của đồ thị và đi qua mỗi đỉnh đúng một lần.
- Chu trình Hamilton là dây chuyền Hamilton và có một cạnh trong đồ thị nối đỉnh đầu của dây chuyền với đỉnh cuối của nó.

II. Ví dụ

Xét 3 đơn đồ thị G1, G2, G3 sau:



- G1 có chu trình Hamilton (a, b, c, d, e, a).
- G2 không có chu trình Hamilton vì deg(a) = 1 nhưng có đường đi Hamilton (a, b, c, d).
- G3 không có cả chu trình Hamilton lẫn đường đi Hamilton.

III. Thuật toán

Dưới đây là phần hướng dẫn cài đặt thuật toán liệt kê tất cả các chu trình Hamilton xuất phát từ đỉnh 0, các chu trình Hamilton khác có thể có được bằng cách hoán vị vòng quanh. Lưu ý rằng cho tới nay, người ta vẫn chưa tìm ra một phương pháp nào thực sự hiệu quả hơn phương pháp đệ qui quay lui để tìm dù chỉ một chu trình Hamilton cũng như đường đi Hamilton trong trường hợp đồ thị tổng quát.

Tìm chu trình Hamilton

Thuật toán gọi đệ qui hàm tìm trong đó thực hiện những công việc sau:

- B1: Kiểm tra nếu số lần gọi đệ qui > số đỉnh của đồ thị và tồn tại cạnh nối từ *x* tới đỉnh đầu tiên trong đường đi thì kết luận có chu trình Hamilon. Ngược lại làm tiếp B2
- B2: Duyệt qua tất cả các đỉnh *i* (chưa xét) kề với đỉnh *x*
- B3: Lưu lại đỉnh i trong đường đi và đánh dấu *i* đã xét. Gọi đệ qui tìm tiếp.
- B4: Bỏ đánh dấu *i* đã xét.

Lưu ý: ban đầu đỉnh 0 được đánh dấu đã xét và được lưu trong đường đi.

Tìm đường đi Hamilton

Cũng làm tương tự như tìm chu trình chỉ khác trong điều kiện kiểm tra ở B1. Chỉ cần kiểm tra số lần gọi đệ qui == số đỉnh của đồ thị là được.

IV. Tổ chức

Hàm *FindHamiltonPath* là một phương thức của lớp Graph.

```
bool FindHamiltonPath (int nCount, int x, vector<int>& path )
{
      if (nCount == n (số đỉnh của đồ thị))
      {
            <Ta có được một đường đi Hamilton>
            return true;
      }
      for (∀i: (x, i) ∈ E) // duyệt qua các đỉnh i kề với x
            if (tồn tại cạnh nối từ x tới i và i chưa được ghé thăm)
            {
                  path[nCount] = i; // lưu lại đường đi
                  <Đánh dấu đã ghé thăm i>
                  if(FindHamiltonPath (nCount + 1, i, path))
                        return true;
                  <Bỏ đánh dấu đã ghé thăm i>
            }
      return false;
}
```

Hàm *HamiltonPath* là một phương thức của lớp Graph. Trả về danh sách các đỉnh trong chu trình Hamilton.

```
vector<int> HamiltonPath ()
{
      // Khởi tạo mảng lưu đường đi
      vector<int> path;
      // …
      // lần lượt xét các đỉnh của đồ thị
      for (∀x: x ∈ V)
      {
            <Đánh dấu đã ghé thăm x>
            Path[0] = x;
            if(FindHamiltonPath(1, x, path)) // tìm thấy đường đi
                  return path;
            <Xóa đánh dấu đã ghé thăm x>
      }
}
```

Bài tập thực hành

Mã đi tuần – Hamilton

I. Quy định

Thời gian làm bài: **2 tuần** (*dealine xem trên moodle*)

Loại bài tập: **cá nhân**

Cấu trúc bên trong thư mục **MSSV** bao gồm các thư mục

- **Source**: thư mục chứa toàn bộ source code.

- **Document**: chứa báo cáo (**MSSV.doc** hoặc **MSSV.docx**).

- **Release**: thư mục chứa file thực thi (**MSSV.exe**).

Nén toàn bộ thư mục thành file **MSSV.zip** hoặc **.rar**

Nộp bài lên moodle.

Lưu ý: tất cả các bài làm sai qui định sẽ không được chấm tức là 0 điểm.

II. Đề bài

Trên bàn cờ $n \times n$ ($3 \le n$) có một số ô bị đục thủng, một con mã muốn đi khắp các ô còn lại mà không có bước đi nào lặp lại 2 lần, các bước đi nối tiếp nhau. Viết chương trình xuất ra một **đường đi** có thể của con mã này.

Chương trình được viết dưới dạng Console với **2** tham số dòng lệnh lần lượt là đường dẫn *tập tin đầu vào* và đường dẫn *tập tin đầu ra*:

Ví dụ tham số dòng lệnh:

MSSV.exe input.txt Output.txt

Hoặc

MSSV.exe C:\input.txt D:\Output.txt

Lưu ý: đường dẫn/tên tập tin đầu vào và đầu ra có thể thay đổi (không cố định)

Cấu trúc file dữ liệu đầu vào:

Biểu diễn bàn cờ quốc tế $n \times n$ bằng một đồ thị có $n \times n = N$ đỉnh. Các ô trên bàn cờ được đánh số bắt đầu từ 0, theo thứ tự từ trái sang phải và từ trên xuống dưới.

- Dòng đầu tiên: **số đỉnh đồ thị (*N*)**
- *N* dòng tiếp theo: **ma trận kề** của đồ thị với quy ước:
- A[*i*][*j*] = 1: con mã đi được từ *i* đến *j*
- A[*i*][*j*] = 0: con mã không đi được từ *i* đến *j*

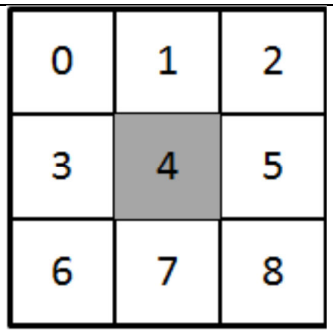
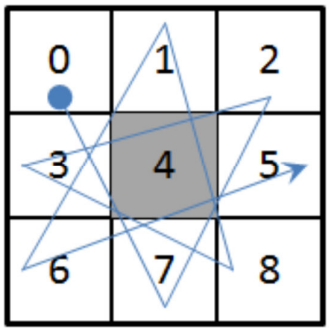
Các đỉnh được đánh chỉ số từ 0

Cấu trúc file dữ liệu đầu ra:

Nếu tìm được đường đi thì lần lượt xuất ra các *đỉnh* trên đường đi đó (cách nhau bởi **khoảng trắng**).

Nếu không tìm được ghi là 0

Ví dụ:

Tập tin đầu vào	**Bàn cờ**	**Tập tin đầu ra**
9 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0	 **Ô số 4 (màu xám) bị đục thủng**	**0 7 2 3 8 1 6 5**  **Đường màu xanh biểu diễn đường đi của con mã**

III. Nội dung báo cáo

Báo cáo cần thể hiện các nội dung sau:

- Cách tổ chức/thiết kế các cấu trúc dữ liệu và thuật toán.
- Ý tưởng thuật toán tìm đường đi Hamilton (**KHÔNG** ghi source code, chỉ ghi ý tưởng)