


TRƯỜNG: ĐH CNTP TP.HCM KHOA: CÔNG NGHỆ THÔNG TIN BỘ MÔN: KHOA HỌC MÁY TÍNH MH: TH CẤU TRÚC RỜI RẠC	CHƯƠNG 3. CÁC BÀI TOÁN VỀ ĐƯỜNG ĐI	
---	---	---

A. MỤC TIÊU:

- Cài đặt thuật toán nhập đồ thị từ file
- Cài đặt thuật toán tìm đường đi và chu trình Euler, Hamilton
- Các ứng dụng

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

- Cho đồ thị $G = (V, E)$, trong đó V là tập các đỉnh và E là tập các cạnh của đồ thị G .

1. Đường đi và chu trình Euler

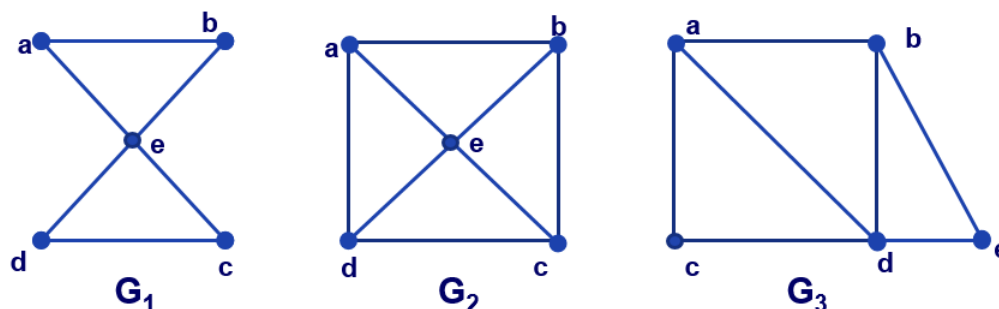
- **Đường đi Euler:**

Đường đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần gọi là đường đi Euler.

- **Chu trình Euler:**

Chu trình đi qua tất cả các cạnh của đồ thị, mỗi cạnh đúng một lần gọi là chu trình Euler.

- **Ví dụ minh họa:** Xét 3 đồ thị G_1, G_2, G_3 sau



G_1 có chu trình Euler a, e, c, d, e, b, a

G_3 không có chu trình Euler nhưng có đường đi Euler a, b, e, d, c, a, d, b .

G_2 không có chu trình và không có đường đi Euler.

- **Điều kiện cần và đủ:** Cho $G = (V, E)$ là đồ thị vô hướng, liên thông
 - G có chu trình Euler \Leftrightarrow Mọi đỉnh của G đều bậc chẵn
 - Nếu G có hai đỉnh bậc lẻ, còn mọi đỉnh khác đều có bậc chẵn thì G có đường đi Euler.

II. Bài tập hướng dẫn mẫu

Đường đi và chu trình Euler

Cho đồ thị G liên thông có n đỉnh, được đánh số từ 1 đến n . Xác định đường đi và chu trình Euler trên đồ thị G (Nếu có).

Trường hợp đồ thị G có thể vô hướng hoặc có hướng. Ví dụ đồ thị G có ma trận kề như sau:

0	1
4	4
0 1 0 1	0 1 0 0
1 0 1 0	0 0 1 0
0 1 0 1	0 0 0 1
1 0 1 0	1 0 0 0

Trong đó dòng đầu tiên thể hiện loại đồ thị giá trị 0 (vô hướng) hoặc 1 (có hướng), dòng thứ 2 thể hiện số đỉnh của đồ thị, các dòng còn lại thể hiện 1 là kề với đỉnh đang xét và 0 là không kề.

Ma trận kề có thể được nhập trực tiếp từ bàn phím hoặc nhập từ file dạng .txt,...

Trong phần cài đặt code tìm đường đi và chu trình Euler dưới đây sử dụng stack, do đó dễ thuận tiện cho việc cài đặt các thao tác với stack (khai báo) và phần duyệt đồ thị được cài đặt riêng làm thư viện như sau:

```
- Cài đặt: thao_tac_stack.h
#ifdef _cac_thao_tac_voi_stack_h
#define _cac_thao_tac_voi_stack_h
//cac_thao_tac_voi_stack.h
/*
    Khai bao kieu Stack
    void Init_Stack(Stack &S); //Khoi tao Stack
    bool IsEmpty_Stack(Stack S); //Kiem tra tinh rong cua Stack
    void Push_Stack(Stack &S, int v); //Them dinh v vao Stack
    int Pop_Stack(Stack &S); //Xoa (lay ra) mot dinh cua Stack
*/
//Khai bao
typedef struct pStack
{
    int Top;
    pStack *Next;
```

```

}*Stack;
//Khoi tao Stack rong
void Init_Stack(Stack &S)
{
    S = NULL;
}
bool IsEmpty_Stack(Stack S)
{
    if(S == NULL)
        return 1; //Neu Stack rong tra ve 1
    return 0; //Nguoc lai tra ve 0
}
void Push_Stack(Stack &S, int v)
{
    pStack *p; //Khai bao mot node moi
    p = new pStack; //Cap phat bo nho cho node moi

    if(p==NULL)
    {
        std::cout<<"Memory Empty."; //Neu p rong, thông báo hết bộ nhớ
        return;
    }
    p->Top = v;
    p->Next = S; //Tao link liên kết từ p đến Stack
    S = p; //Bây giờ p là Stack
}
int Pop_Stack(Stack &S)
{
    pStack *p; //p để lưu node bị xóa
    if(IsEmpty_Stack(S))
    {
        return 0; //Neu Stack rong thì thoát ra
    }
    p = S; //p là S
    S = p->Next; //S link tới node đang sau nó
    int v = p->Top; //v lưu giá trị của node bị xóa
    delete(p); //Giải phóng bộ nhớ cho node bị xóa
    return v; //Trả về giá trị của node bị xóa
}
#endif
- Cài đặt: duyet_dothi_stack.h
#include<iostream>
#include"thao_tac_stack.h"
#ifndef duyet_dothi_stack_h
#define duyet_dothi_stack_h
using namespace std;
#define TRUE 1
#define FALSE 0
#define MAX 100
int so_TP_lien_thong = 1,n;
int a[MAX][MAX];

```

```

bool daxet[MAX];
/*
    n = so dinh cua do thi
    a[][] = ma tran lien ke bieu dien do thi
    so_TP_lien_thong = dem so thanh phan lien thong cua do thi
    daxet[] = danh dau cac dinh da duoc tham
*/
//*****
//*****
//Khai bao nguyen mau ham
void Init_Stack(Stack &Q);
bool IsEmpty_Stack(Stack Q);
void Push_Stack(Stack &Q, int v);
int Pop_Stack(Stack &Q);
void Init_daxet();
//*****
//*****
void Init_daxet()
{
    for(int i =0;i<=n;++i)
        daxet[i] = FALSE;
}
void duyet_dothi(int v) //duyet đồ thị theo DFS sử dụng stack
{
    Stack P;
    Init_Stack(P);
    daxet[v] = TRUE;
    Push_Stack(P,v);
    int i;
    while(!IsEmpty_Stack(P))
    {
        v = P->Top;
        for(i=1;i<=n;++i)
        {
            if(a[i][v]!=0 && daxet[i] == FALSE)
            {
                Push_Stack(P,i);
                daxet[i] = TRUE;
                v = i;
                i =-1;
            }
        }
        Pop_Stack(P);
    }
}
bool kiem_tra_lien_thong()
{
    for(int i =1;i<=n;++i)
    {
        if(daxet[i] == FALSE)
            return FALSE; //Neu do thi khong lien thong tra ve FALSE
    }
}

```

```

        return TRUE;
    }
#endif

```

1) Đường đi Euler

```

#include "stdio.h"
#include "conio.h"
#include<fstream>
#include<iostream>
#include<stdlib.h>
#include"duyet_dothi_stack.h"
#define TRUE 1
#define FALSE 0
#define MAX 100
using namespace std;
int v;
int co; //đồ thị: 0 là vô hướng, 1 là có hướng
struct dothi
{
    int flag;    //loại đồ thị: 0 là vô hướng, 1 là có hướng
    int n;        //số đỉnh của đồ thị
    int w[100][100]; //ma trận kề hoặc ma trận trọng số
};
//đọc dữ liệu từ file lưu vào cấu trúc đồ thị
void docfile(dothi &g)
{
    FILE *f = fopen("D:\\BaiTap_CTRR\\Euler_input05_vohuong.txt", "r");
    if(f == NULL)
    {
        printf("\nKhong tim thay file");
        printf("\n");
        system("pause");
    }
    else
    {
        fscanf(f, "%d", &g.flag);
        fscanf(f, "%d", &g.n);
        for (int i = 1; i <= g.n; i++)
            for (int j = 1; j <= g.n; j++)
                fscanf(f, "%d", &g.w[i][j]);
    }
}
void xuatdothi(dothi g)
{
    if (g.flag==0)
        printf("\nG la do thi vo huong");
    else
    {
        printf("\nG la do thi co huong");
    }
    printf("\nSo dinh cua G la: %d",g.n); //xuất số đỉnh của đồ thị
    n=g.n; // n số phần tử ma tran ke stack sử dụng
}

```

```

// Xuất ra ma tran ke đã input để kiểm tra
printf("\nMa tran ke G :");
for (int i = 1; i <= g.n; i++)
{
    printf("\n");
    for (int j = 1; j <= g.n; j++)
    {
        printf("%4d", g.w[i][j]);
        a[i][j]=g.w[i][j]; //mảng được dùng trong stack
        co=g.flag; // loại đồ thị
    }
}
}
//Kiểm tra xem số đỉnh bậc lẻ của đồ thị có nhiều hơn 2 hay không
int test_dinh()
{
    v = 1;
    int i,j,k,z,dem = 0;
    int tongbacvao, tongbacra;
    if(co==0) // loại đồ thị vô hướng
    {
        for(i=1;i<=n;++i)
        {
            k = 0;
            for(j=1;j<=n;++j)
            {
                k += a[i][j];
            }
            if(k%2 == 1)
            {
                v = i;
                ++dem;
            }
        }
    }
    else //loại đồ thị có hướng
    { // đếm số đỉnh có |tổng bậc ra - tổng bậc vào| =1
        for(i=1; i<=n; ++i)
        {
            tongbacvao=0;
            tongbacra=0;
            for(z=1; z<=n; ++z) //Tính tổng bậc vào (cột)
                tongbacvao+=a[z][i];
            for(j=1; j<=n; ++j) //Tính tổng bậc ra (hàng)
                tongbacra+=a[i][j];
            if(tongbacvao-tongbacra!=0)
            {
                v=i;
                ++dem;
            }
        }
    }
    return dem;
}
//Kiểm tra xem đồ thị có liên thông hay không

```

```

bool test()
{
    duyet_dothi(1);
    if(co==0) //loại đồ thị vô hướng
    {
        if(kiem_tra_lien_thong()== FALSE || test_dinh(>2)
            return FALSE;
    }
    else // loại đồ thị có hướng
    {
        if(test_dinh(>2)
            return FALSE;
    }
    return TRUE;
}
//Duyet tu dinh v
void duong_Euler()
{
    int i;
    Stack Q; // Q = NULL
    Init_Stack(Q);
    Push_Stack(Q,v); // đưa v vào stack Q
    while(!IsEmpty_Stack( Q))
    {
        v = Q->Top; // lấy đỉnh ở đầu stack
        for(i =1;i<=n;++i)
        {
            if(a[i][v]>0 )
            {
                Push_Stack(Q,i);
                //Xoa bot 1 canh noi giua i va v
                if(co==0) // loại đồ thị vô hướng
                {
                    --a[i][v];
                    --a[v][i];
                }
                else // loại đồ thị có hướng
                    --a[i][v];
                v = i; //Bay gio dinh cua Stack la i
                i = -1; //Duyet lai tu dau
            }
        }
        cout<<Pop_Stack(Q)<<" ";
    }
}
void main()
{
    dothi g;
    docfile(g);
    xuatdothi(g);
    if(test()== FALSE)
    {
        cout<<endl<<"Do thi da cho khong co duong di Euler."<<endl;
    }
}

```

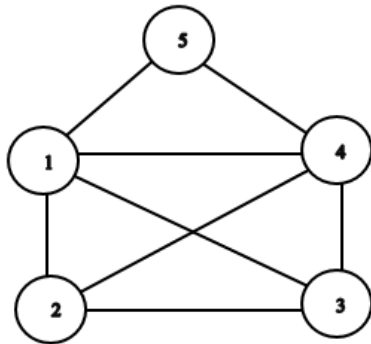
```

        system("pause");
        return;
    }
    else
    {
        cout<<endl<<"Duong di Euler cua do thi da cho la: "<<endl;
        duong_Euler();
    }
    _getch();
}

```

Chạy minh họa chương trình tìm đường đi Euler:

- Cho đồ thị G vô hướng như sau



Ma trận kề

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

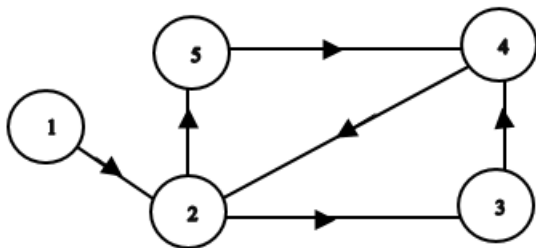
Kết quả chương trình:

```

G la do thi vo huong
So dinh cua G la: 5
Ma tran ke G :
0 1 1 1 1
1 0 1 1 0
1 1 0 1 0
1 1 1 0 1
1 0 0 1 0
Duong di Euler cua do thi da cho la:
2 4 5 1 4 3 2 1 3

```

- Cho đồ thị G có hướng như sau



Ma trận kề

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

Kết quả chương trình

```
G la do thi co huong
So dinh cua G la: 5
Ma tran ke G :
  0  1  0  0  0
  0  0  1  0  1
  0  0  0  1  0
  0  1  0  0  0
  0  0  0  1  0
Duong di Euler cua do thi da cho la:
1 2 5 4 2 3 4
```

Lưu ý: Một đồ thị có thể không hoặc có nhiều đường đi Euler.

Bài tập: Hãy cải tiến đoạn chương trình trên để có thể in ra tất các đường đi Euler (nếu có) trong đồ thị G.

2) Chu trình Euler

```
#include "stdio.h"
#include "conio.h"
#include<fstream>
#include<iostream>
#include<stdlib.h>
#include"duyet_dothi_stack.h"

#define MAX 100
#define TRUE 1
#define FALSE 0
using namespace std;
int co; //đồ thị: 0 là vô hướng, 1 là có hướng
struct dothi
{
    int flag;    //loại đồ thị: 0 là vô hướng, 1 là có hướng
    int n;      //số đỉnh của đồ thị
    int w[100][100]; //ma trận kề hoặc ma trận trọng số
};
//đọc dữ liệu từ file lưu vào cấu trúc đồ thị
void docfile(dothi &g)
{
    FILE *f = fopen("D:\\BaiTap_CTRR\\Euler_input04_cohuong2.txt", "r");
    if (f == NULL)
    {
        printf("\nKhong tim thay file");
        printf("\n");
        system("pause");
    }
    else
    {
        fscanf(f, "%d", &g.flag);
        fscanf(f, "%d", &g.n);
```

```

        for (int i = 1; i <= g.n; i++)
            for (int j = 1; j <= g.n; j++)
                fscanf(f, "%d", &g.w[i][j]);
    }
}
void xuatdothi(dothi g)
{
    if (g.flag==0)
        printf("\nG la do thi vo huong");
    else
    {
        printf("\nG la do thi co huong");
    }
    printf("\nSo dinh cua G la: %d",g.n); //xuất số đỉnh của đồ thị
    n=g.n; // n số phần tử ma tran ke stack sử dụng
    // Xuất ra ma tran ke đã input để kiểm tra
    printf("\nMa tran ke cua G:");
    for (int i = 1; i <= g.n; i++)
    { printf("\n");
        for (int j = 1; j <= g.n; j++)
        { printf("%4d", g.w[i][j]);
            a[i][j]=g.w[i][j]; //mảng được dùng trong stack
            co=g.flag; // cờ thể hiện loại đồ thị
        }
    }
}
//Kiểm tra xem do thi co dinh nao co bac le khong
bool test_bac()
{
    int i,j,t,z;
    int tongbacvao, tongbacra;
    if(co==0) // kiểm tra loại đồ thị, nếu loại vô hướng
    {
        for(i =1;i<=n;++i)
        {
            t = 0;
            for(j=1;j<=n;++j)
                t+=a[i][j];
            if(t % 2 == 1)
                return FALSE; //Neu do thi co bac le tra ve FALSE
        }
        return TRUE; //Neu do thi cho co bac chan tra ve TRUE
    }
    else //co=1: loại đồ thị có hướng
    {
        // kiểm tra tổng bậc ra = tổng bậc vào của 1 đỉnh => true, ngược lại
false
        for(i=1; i<=n; ++i)
        { tongbacvao=0;
            tongbacra=0;
            //Tính tổng bậc vào (cột)
            for(z=1; z<=n; ++z)

```

```

        tongbacvao+=a[z][i];
    for(j=1; j<=n; ++j)
        tongbacra+=a[i][j];
    if(tongbacvao!=tongbacra)
        return FALSE;
    }
    return TRUE; // đồ thị có hướng các đỉnh có bậc vào bằng bậc ra
}
}
//Kiểm tra xem đồ thị có chu trình Euler không
bool test()
{
    //Duyệt theo chiều sâu
    duyet_dothi(1);
    if(kiem_tra_lien_thong() && test_bac())
        return TRUE;
    else
        return FALSE;
}
//Tìm chu trình Euler
void chutrinh_Euler()
{
    int i,v,t=0;
    Stack p;
    Init_Stack(p);

    //Đưa đỉnh 1 vào Stack
    Push_Stack(p,1);
    while(!IsEmpty_Stack(p))
    {
        v = p->Top;
        for(i=1;i<=n;++i)
        {
            if(a[i][v]>0)
            {
                Push_Stack(p,i);
                //Bo bot 1 canh noi giua i va v
                if(co==0) // loại đồ thị vô hướng
                {
                    --a[i][v];
                    --a[v][i];
                }
                else // loại đồ thị có hướng
                    --a[i][v];
                //Duyệt lại từ đỉnh i
                i=-1;
                v = p->Top; // v = đầu của Stack
            }
        }
        //Loại một đỉnh ở đầu stack
        cout<<Pop_Stack(p)<<" ";
    }
}

```

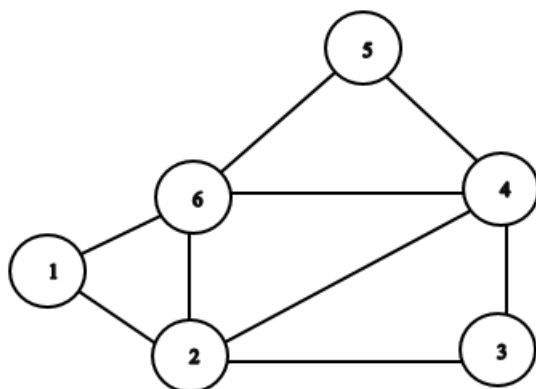
```

}
void main()
{
    dothi g;
    docfile(g);
    xuatdothi(g);
    if(test())
    {
        cout<<endl<<"Chu trình Euler tìm được là: "<<endl;
        chutrinh_Euler();
    }
    else
    {
        cout<<endl<<"Đồ thị đã cho không có chu trình Euler";
        //return;
    }
    cout<<endl;
    _getch();
}

```

Chạy minh họa chương trình tìm chu trình Euler:

- Cho đồ thị G vô hướng như sau



Ma trận kề

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

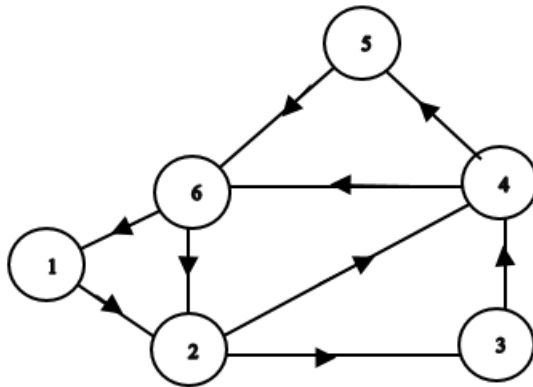
Kết quả chương trình:

```

G là đồ thị vô hướng
Số đỉnh của G là: 6
Ma trận kề của G:
  0  1  0  0  0  1
  1  0  1  1  0  1
  0  1  0  1  0  0
  0  1  1  0  1  1
  0  0  0  1  0  1
  1  1  0  1  1  0
Chu trình Euler tìm được là:
1 6 5 4 6 2 4 3 2 1

```

- Cho đồ thị G có hướng như sau



Ma trận kề:

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

Kết quả chương trình:

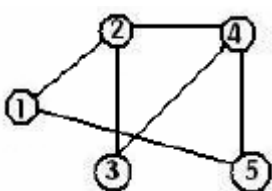
```
G là đồ thị có hướng
Số đỉnh của G là: 6
Ma trận kề của G:
0 1 0 0 0 0
0 0 1 1 0 0
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 0 1
1 1 0 0 0 0
Chu trình Euler tìm được là:
1 2 3 4 5 6 2 4 6 1
```

Lưu ý: Một đồ thị có thể không hoặc có nhiều chu trình Euler.

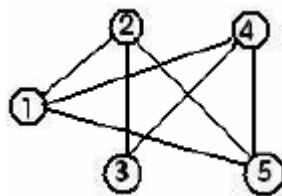
Bài tập: Hãy cải tiến đoạn chương trình trên để có thể in ra tất các chu trình Euler (nếu có) trong đồ thị G.

III. Bài tập ở lớp

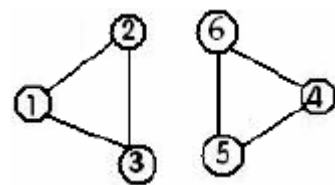
Bài 1. Có n thành phố, biết rằng đường đi giữa 2 thành phố (nếu có) là đường đi 2 chiều. Sơ đồ mạng lưới giao thông trong thành phố được cho bởi 1 trong những đồ thị sau:



G1



G2



G3

Yêu cầu:

- Xây dựng file dữ liệu đầu vào đặt tên Buoi4.txt theo cấu trúc sau cho từng trường hợp trên
- Dòng đầu tiên ghi 1 con số thể hiện loại đồ thị : 0 (vô hướng) hoặc 1 (có hướng)

- Dòng thứ 2 ghi số đỉnh của đồ thị (n).
- Dòng thứ 3 -> dòng n+2 lưu ma trận kề của đồ thị

VD : G1 được lưu trữ như sau

```

0
5
0 1 0 0 0
1 0 1 1 1
0 1 0 1 0
0 1 1 0 1
0 1 0 1 0

```

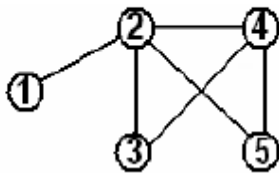
- b) Lần lượt kiểm tra các đồ thị G1, G2, G3 có đường đi và chu trình Euler/Hamilton hay không ? In ra màn hình các đường đi và chu trình này

Bài 2. Cho một mạng lưới giao thông hai chiều giữa n địa điểm, một người đưa thư cần đi qua tất cả các con đường này để phát thư, mỗi đường chỉ qua 1 lần. Hãy xác định lộ trình của người đưa thư này (nếu có, ngược lại hiện thông báo không tồn tại lộ trình)

Kiểm nghiệm thuật toán cho đồ thị sau :

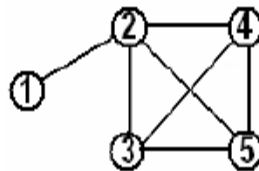
Kết quả:

1->2->3->4->2->5->4



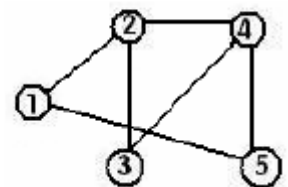
Kết quả:

KHONG TON TAI DUONG DI



Kết quả:

2->1->5->4->2->3->4



IV. Bài tập về nhà

Bài 3. Tương tự các bài trên, hãy cài đặt code tìm đường đi và chu trình Hamilton cho đồ thị G có n đỉnh.

Bài 4. Hãy dùng phương pháp đệ quy để cài đặt lại chương trình tìm đường đi và chu trình Euler.

Bài 5. Một công ty lữ hành muốn dẫn đoàn khách du lịch tham quan các địa điểm nổi tiếng trong thành phố (giả sử có n địa điểm), điểm xuất phát đầu tiên (điểm số 1) là Nhà thờ Đức Bà tại trung tâm thành phố, mỗi địa điểm chỉ ghé 1 lần, sau đó đến địa điểm kế tiếp và sau cùng quay về điểm xuất phát. Giả sử coi lộ trình của đoàn khách du lịch là 1 đồ thị, mỗi điểm đến là 1 đỉnh của đồ thị, từ điểm này đến điểm kia là cạnh của đồ thị. Hãy xác định lộ trình phù hợp với yêu cầu trên cho hướng dẫn viên và tài xế công ty lữ hành thực hiện dẫn đoàn khách tham quan.