HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA VIỄN THÔNG 1

BÁO CÁO TIỂU LUẬN

ĐỒ THỊ HAMILTON VÀ THUẬT TOÁN TÌM CHU TRÌNH HAMILTON

TOÁN RỜI RẠC - NHÓM 01

Giảng viên: Phạm Anh Thư

Bàn báo cáo: 1

Sinh viên thực hiện:

1. Vi Minh Hiếu : B22DCVT197
2. Hoàng Đức Đạt : B22DCVT126
3. Nông Đình Khôi : B22DCVT295
4. Hoàng Thanh Tùng : B22DCVT495
5. Trịnh Đức Chung : B22DCVT083

Hà Nội, 2024

PHÂN CÔNG CÔNG VIỆC

Họ và tên	Nội dung
Vi Minh Hiếu	Thuật toán tìm chu trình Hamilton
Hoàng Đức Đạt	Bài toán Mã đi tuần
Nông Đình Khôi	Tìm hiểu trò chơi hình khối Icosian
Hoàng Thanh Tùng	Quy tắc tìm chu trình Hamilton và định lý
Trịnh Đức Chung	Định nghĩa các thuật ngữ trong đồ thị Hamilton

MỤC LỤC

1. Bài toán hình khối: Trò chơi Icosian 4
1.1) Lịch sử hình thành trò chơi Icosian 4
1.2) Phát biểu bài toán 4
1.3) Câu hỏi đặt ra cho bài toán 6
2. Định nghĩa và ứng dụng của đồ thị Hamilton 6
2.1) Định nghĩa 6
2.2) Ứng dụng của đồ thị Hamilton 7
3. Định lý và Quy luật trong đồ thị Hamilton 8
3.1) Định lý 8
3.2) Quy tắc tìm chu trình Hamilton 9
4. Thuật toán tìm chu trình Hamilton 11
4.1) Đặt bài toán 11
4.2) Mô tả thuật toán 11
4.3) Kiểm nghiệm thuật toán 12
4.4) Cài đặt thuật toán 12
5. Bài toán Mã đi tuần 15
5.1) Bài toán: 16
5.2) Cách di chuyển của một quân mã: 16
5.3) Xây dựng bước đi cho quân mã: 17
5.4) Kiểm tra tính hợp lệ của bước đi: 17
5.5) Toàn bộ bài toán bằng ngôn ngữ C++: 18
6. Những điểm cần ghi nhớ 18

1. Bài toán hình khối: Trò chơi Icosian

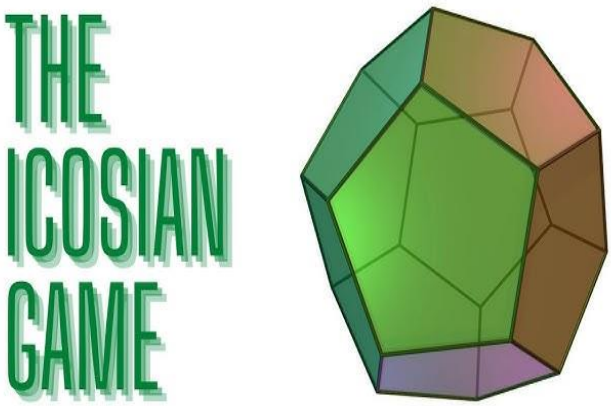
1.1) Lịch sử hình thành trò chơi Icosian

Bài toán hình khối (Hamitlon), còn gọi là trò chơi Icosian được đưa ra bởi nhà toán học người Ireland, William Rowan Hamitlon vào năm 1857, bài toán được coi là sự mở đầu cho lý thuyết đồ thị trong toán học dựa trên việc tìm kiếm chu trình Hamilton.

Hình 1.1. William Rowan Hamilton (1805-1865)

Hình 1.2. Trò chơi Icosian 2D

Trò chơi được thiết kế trên một khối đa diện 12 mặt, 20 đỉnh và 30 cạnh. Mỗi mặt của khối đa diện là một hình ngũ giác tạo nên từ 5 cạnh và 5 đỉnh.

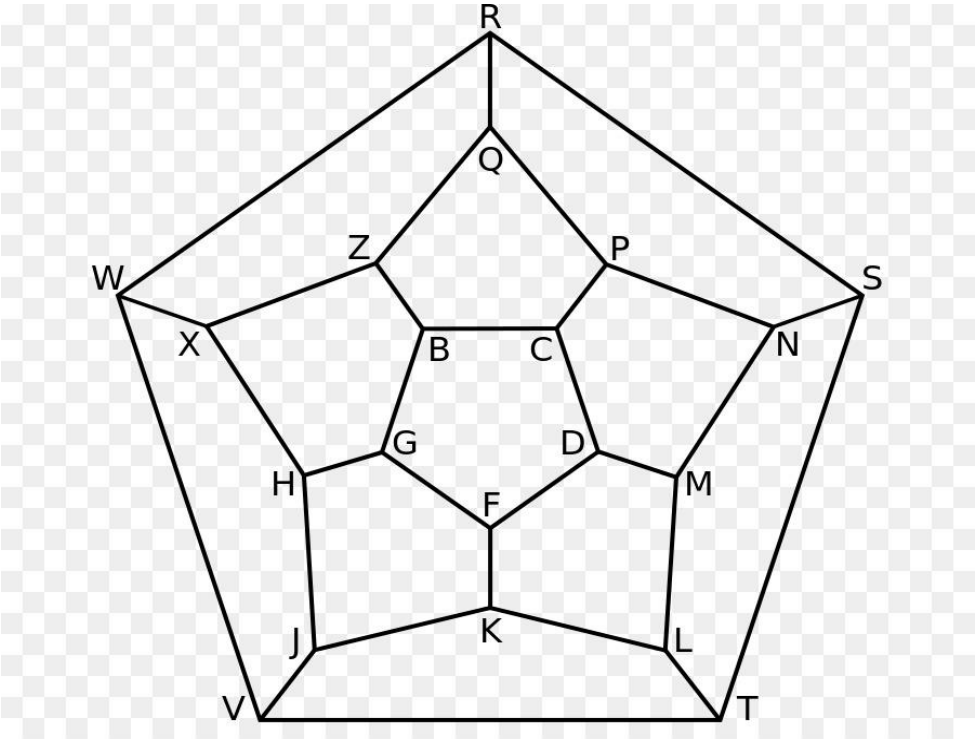


Hình 1.3. Trò chơi Icosian 3D

1.2) Phát biểu bài toán

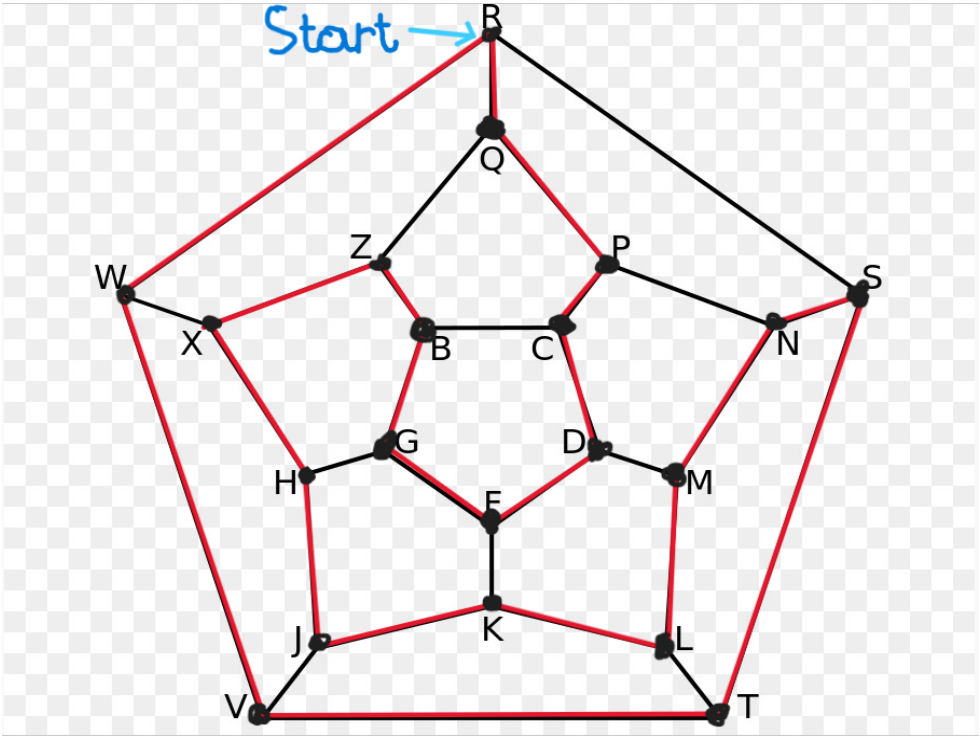
Trong trò chơi này, nhà toán học đã đặt ra yêu cầu đối với người chơi đó là tìm ra chu trình đơn, sao cho tất cả tất cả đỉnh đều được thăm đúng một lần, không có cạnh nào được thăm hai lần và điểm kết thúc của chu trình này chính là quay trở lại điểm ban đầu đã xuất phát.

Ví dụ: Cho đồ thị như Hình 1.4, dựa vào trò chơi Icosian hãy tìm ra chu trình đơn theo các yêu cầu, tất cả các đỉnh đều được thăm một lần và điểm kết thúc sẽ là điểm bắt đầu?



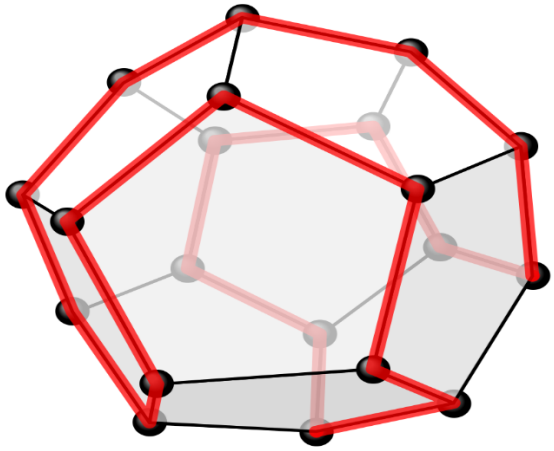
Hình 1.4. Ví dụ về bài toán hình khối

Lời giải:



Hình 1.5. Lời giải bài toán

Đường đi thu được qua trò chơi Icosian: **<R, W, V, T, S, N, M, L, K, J, H, X, Z, B, G, F, D, C, P, Q, R>**



Hình 1.6. Lời giải bài toán trong không gian

Thông qua việc thăm tất cả các đỉnh của khối 12 mặt từ đường đi màu đỏ đã tạo thành một chu trình đơn hay còn được gọi là **chu trình Hamilton.**

1.3) Câu hỏi đặt ra cho bài toán

Chúng ta có thể tìm được bao nhiêu chu trình Hamilton tồn tại trên khối đa diện 12 mặt, đường đi nào là tối ưu nhất?

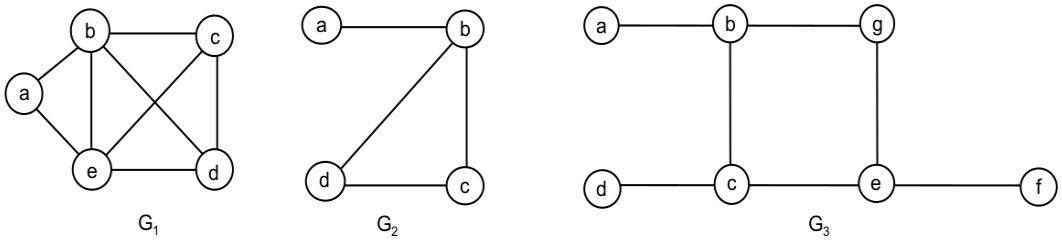
Các kỹ thuật hoặc thuật toán nào có thể được sử dụng để giải bài toán này trên các đồ thị lớn hơn hoặc có cấu trúc phức tạp hơn?

2. Định nghĩa và ứng dụng của đồ thị Hamilton

2.1) Định nghĩa

- **Chu trình Hamilton**: Là một chu trình trong đồ thị đi qua tất cả các đỉnh một lần duy nhất và quay trở về điểm xuất phát.
- **Đường đi Hamilton**: Là một đường đi trong đồ thị mà đi qua tất cả các đỉnh một lần mà không trở lại. Đường đi này không nhất thiết phải khép kín như chu trình Hamilton.
- **Đồ Thị Hamilton**: Là một đồ thị mà tồn tại một chu trình Hamilton, tức là một chu trình đi qua tất cả các đỉnh của đồ thị đúng một lần và quay về đỉnh xuất phát.
- **Đồ Thị Nửa Hamilton**: Là một đồ thị có đường đi Hamilton, tức là một đường đi qua tất cả các đỉnh nhưng không nhất thiết phải trở lại điểm xuất phát.

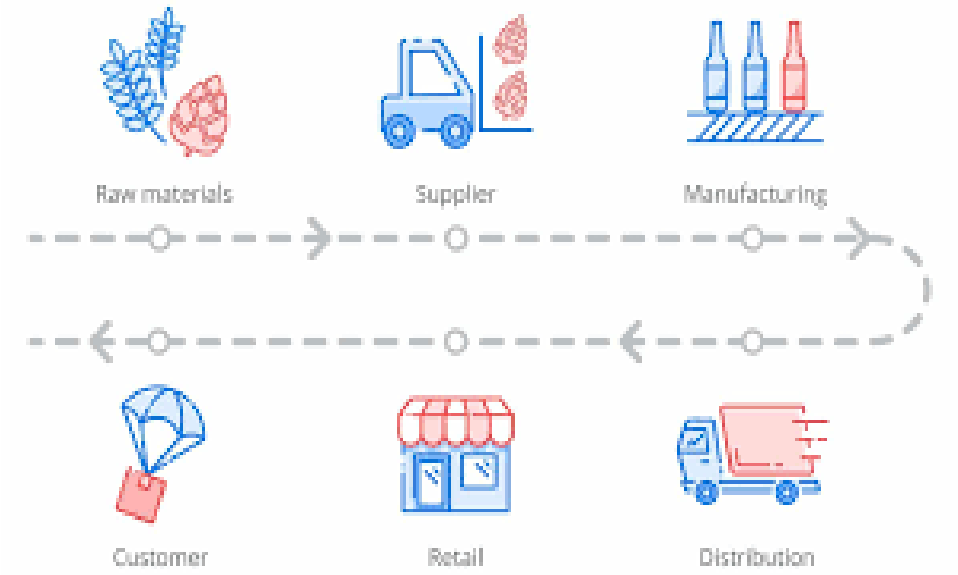
Chú ý: Mọi Đồ Thị Hamilton đều là Nửa Hamilton nhưng điều ngược lại không đúng.



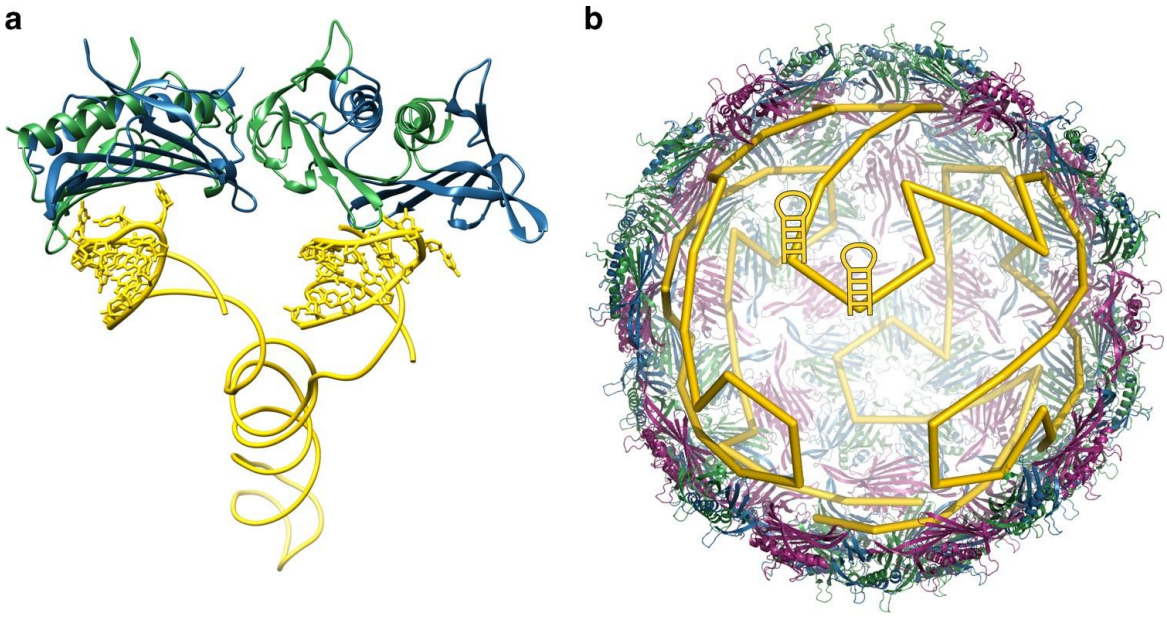
G1.Đồ Thị Hamilton *G2.Đồ thị nửa Hamilton* *G3.Không có đường đi Hamilton*

2.2) Ứng dụng của đồ thị Hamilton

- **Quy hoạch du lịch**:
  - **Mô tả**: Một thương nhân muốn đi thăm một số thành phố và trở về điểm xuất phát, với mục tiêu tìm lộ trình ngắn nhất.
  - **Ứng dụng**: Hữu ích trong logistics để tối ưu hóa tuyến đường giao hàng, giảm chi phí vận chuyển và tiết kiệm thời gian.
- **Mạng Viễn Thông**
  - **Mô tả**: Thiết kế các mạng để truyền tải dữ liệu qua nhiều nút (điểm kết nối) mà không lặp lại.
  - **Ứng dụng**: Sử dụng trong việc xây dựng mạng viễn thông hiệu quả, tối ưu hóa đường truyền và giảm thiểu độ trễ trong việc chuyển tiếp tín hiệu.
- **Trò Chơi và Giải Trí**
  - **Mô tả**: Một số trò chơi chiến thuật yêu cầu người chơi di chuyển qua các ô mà không lặp lại.
  - **Ứng dụng**: Trong thiết kế trò chơi video, để tạo ra các cấp độ thú vị và thách thức cho người chơi.
- **Khai thác dữ liệu**
  - **Mô tả**: Trong nhiều thuật toán, đặc biệt là các thuật toán tìm kiếm và tối ưu hóa, đồ thị Hamilton có thể được sử dụng để phân tích và xử lý dữ liệu hiệu quả hơn.
  - **Ứng dụng**: Trong lĩnh vực khai thác dữ liệu, các nhà nghiên cứu có thể áp dụng lý thuyết đồ thị Hamilton để tối ưu hóa việc tìm kiếm và sắp xếp thông tin, từ đó cải thiện tốc độ và độ chính xác của các thuật toán.
- **Sinh học**
  - **Mô tả**: Trong nghiên cứu DNA, quá trình giải trình tự gen thường bắt đầu bằng cách đọc các đoạn ngắn (reads) của DNA. Để tái tạo chuỗi gen dài, cần kết nối các đoạn này mà không bị trùng lặp.
  - **Ứng dụng**: Các nhà khoa học sử dụng thuật toán đồ thị Hamilton để phân tích và kết hợp các đoạn gen, giúp trong việc nghiên cứu bệnh di truyền hoặc phát triển liệu pháp gen.



Hình 2.1. Tối ưu hóa chuỗi cung ứng.



Hình 2.2. Đồ thị Hamilton trong sinh học

3. Định lý và Quy luật trong đồ thị Hamilton

3.1) Định lý

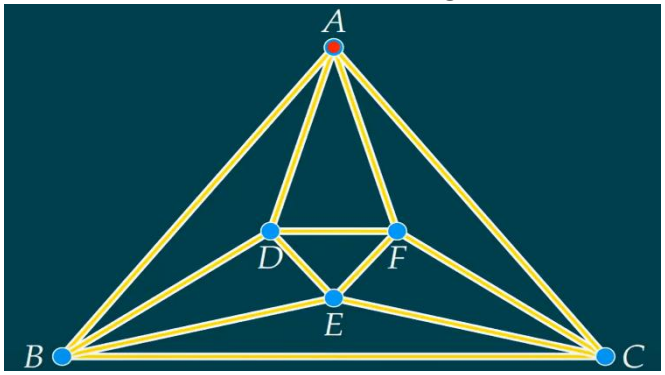
- **Định lý 1**

Xét với đồ thị vô hướng G, trong đó tồn tại k đỉnh sao cho nếu xóa đi k đỉnh này cùng với những cạnh liên thuộc của chúng thì đồ thị nhận được sẽ có nhiều hơn k thành phần liên thông. Khi đó, khẳng định G không phải là đồ thị Hamilton.

- **Định lý 2 (Định lý Dirak, 1952)**

Xét đơn đồ thị vô hướng G=(V,E) có n đỉnh (n≥3). Nếu mọi đỉnh đều có bậc không nhỏ hơn ⌊n/2⌋ thì G là đồ thị Hamilton.

Ví dụ*:* Đồ thị G sau có chu trình Hamilton không?



Hình 3.1. Ví dụ Định lý 2

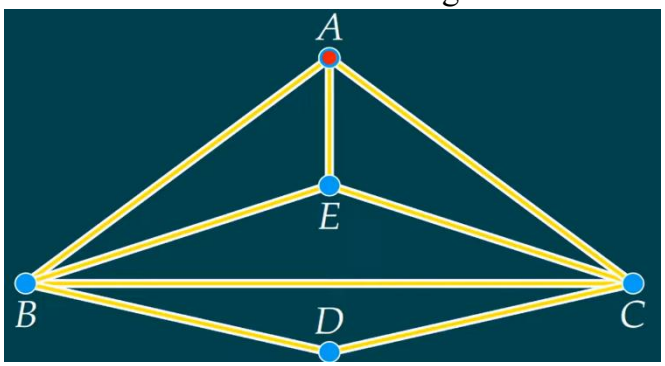
Lời giải; G là đơn đồ thị liên thông 6 đỉnh. Mọi đỉnh của G đều có bậc 4 >(n/2=3). Theo định lý Dirak, G có chu trình Hamilton. (ABCEFDA là một chu trình Hamilton).

- **Định lý 3 (Định lý Ghouila - Houiri, 1960)**

Xét đơn đồ thị có hướng liên thông mạnh G=(V,E) có n đỉnh. Nếu trên phiên bản vô hướng của G, mọi đỉnh đều có bậc không nhỏ hơn n thì G là đồ thị Hamilton.

- **Định lý 4 (Định lý Ore, 1960)**

Xét đơn đồ thị vô hướng G=(V,E) có n đỉnh (n≥3). Xét mọi cặp đỉnh không kề nhau, nếu không có cặp đỉnh nào có tổng bậc nhỏ hơn n thì G là đồ thị Hamilton.
Ví dụ**:** Đồ thị G sau có chu trình Hamilton không?



Hình 3.2. Ví dụ Định lý 4

Lời giải: G là đơn đồ thị liên thông có 5 đỉnh. G chỉ có 2 cặp đỉnh không kề nhau là A, D và E, D. Ta có: deg(A)+deg(B)=deg(E)+deg(D)=3+2=5. Theo định lý ore=> G có chu trình Hamilton. (ABDCEA là một chu trình Hamilton)

- **Định lý 5 (Định lý Meynie, 1973)**

Xét đơn đồ thị có hướng liên thông mạnh G=(V,E) có n đỉnh. Nếu trên phiên bản vô hướng của G mọi cặp đỉnh không kề nhau đều có tổng bậc không nhỏ hơn 2n−1 thì G là đồ thị Hamilton.

- **Định lý 6 (Định lý Bondy - Chvátal, 1972)**

Xét đơn đồ thị vô hướng G=(V,E) có n đỉnh, với mỗi cặp đỉnh không kề nhau (u,v) mà deg(u)+deg(v)≥n, ta thêm một cạnh nối u và v. Cứ làm như vậy tới khi không thêm được cạnh nào nữa, thì ta thu được một bao đóng của đồ thị G, kí hiệu cl(G). Khi đó, G là đồ thị Hamilton nếu và chỉ nếu cl(G) là đồ thị Hamilton.

3.2) Quy tắc tìm chu trình Hamilton

Cho đồ thị G = <V,E>, *v* V

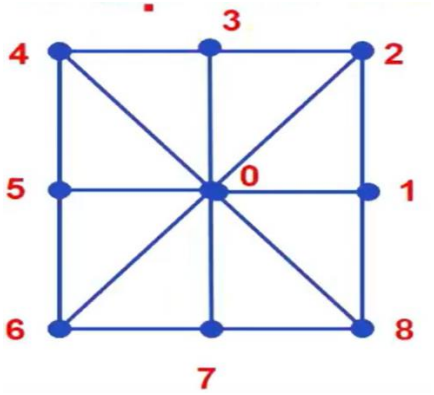
• **Quy tắc 1:** Nếu tồn tại một đỉnh của *G* có bậc <=1 thì *G* không có chu trình Hamilton.

• **Quy tắc 2:** Nếu đỉnh *v* có bạc là 2 thì cả 2 cạnh tới *v* đều phải thuộc chu trình Hamilton.

• **Quy tắc 3:** Chu trình Hamilton không chứa bất kỳ chu trình con thực sự nào.

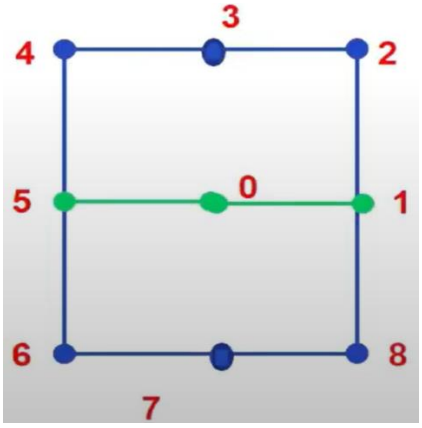
• **Quy tắc 4:** Trong quá trình xây dựng chu trình Hamilton, sau khi đã lấy 2 cạnh tới đỉnh *v* đặt vào chu trình *H* rồi thì không thể lấy thêm cạnh nào tới *v* nữa, do đó có thể xóa mọi cạnh còn lại tới *v*.

Ví dụ: Tìm chu trình Hamilton của đồ thị sau?



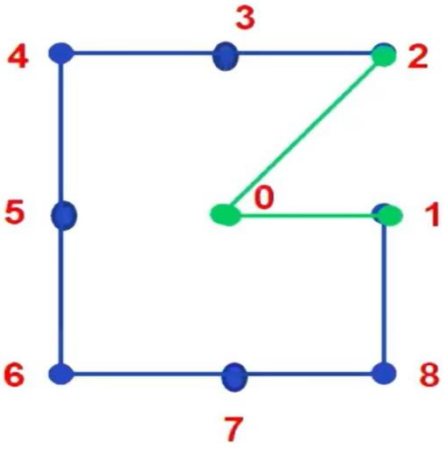
Lời giải:

- Áp dụng quy tắc 1, ta thấy tất cả các đỉnh của đồ thị đều có bậc >1. Vậy khả năng sẽ có chu trình Hamilton.
- **Xét đỉnh 0**: Ta chọn 2 cạnh tới đỉnh này.
- **TH1: 01 và 05**: Xóa các cạnh tới 0 khác theo quy tắc (4) ta còn lại đồ thị.



Các đỉnh 2,3,4 còn lại là bậc 2. Vậy lấy các cạnh 12,23,34,45 theo quy tắc (2), nhưng như vậy tạo ra chu trình con thực sự -> vi phạm quy tắc(3) -> **vô lý.**

-**TH2: 01 và 02**: Xóa các cạnh tới 0 khác theo quy tắc (4) và xóa các cạnh 12 theo quy tắc (3) ta còn lại đồ thị.



Các đỉnh 3,4,5,6,7,8 còn lại là bậc 2. Vậy lấy các cạnh 23 34 45 56 67 78 81 theo quy tắc (2) , như vậy ta nhận được chu trình Hamilton **0 2 3 4 5 6 7 8 1 0.**

-TH3: Lập luận tương tự các trường hợp chọn cạnh 01 và 03, 01 và 04,... đều không được. Tóm lại mọi chu trình Hamilton của đồ thị này đều phải chứa 2 cạnh tới đỉnh 0 hợp với nhau góc 45 độ.

4. Thuật toán tìm chu trình Hamilton

4.1) Đặt bài toán

Bài toán tìm chu trình Hamilton được đặt ra như sau:

- Input: Đồ thị vô hướng G=<V, E> gồm n đỉnh, m cạnh
- Output: Liệt kê tất cả chu trình Hamilton có trong đồ thị

4.2) Mô tả thuật toán

- Cho đến nay, việc tìm ra một tiêu chuẩn để nhận biết một đồ thị là đồ thị Hamilton vẫn còn mở mặc dù đây là vấn đề trung tâm của lý thuyết đồ thị.
- Cho đến nay, cũng vẫn chưa có thuật toán hiệu quả để kiểm tra một đồ thị có phải là đồ thị Hamilton hay không.
- Thông thường ta sẽ đi vào bài toán liệt kê dùng **phương pháp quay lui** để tìm tất cả chu trình Hamilton xuất phát từ đỉnh k. Nhưng do phương pháp quay lui sử dụng đệ quy nên bài toán chỉ áp dụng cho không gian tìm kiếm rất nhỏ (n<=20)
- Trong chu trình Hamilton, **mỗi đỉnh chỉ xuất hiện 1 lần**, cho nên chúng ta phải đánh dấu những đỉnh đã có trong chu trình trong quá trình tìm kiếm bằng 1 mảng visited[].

Hình dưới đây mô tả thuật toán liệt kê tất cả chu trình Hamilton bắt đầu tại đỉnh k.

```
Thuật toán Hamilton( int k){
/* Liệt kê các chu trình Hamilton của đồ thị bằng cách phát triển dãy đỉnh
(X[1], X[2], ..., X[k-1]) của đồ thị G=<V, E>  */
   for v ∈ Ke(X[k-1]){  //duyệt các đỉnh kề với X[k-1]
         if (k == n+1) and (v == v0) then
               <Ghi nhận chu trình X[1], X[2], ..., X[n], v0>
         else if(visited[v] == false){
               X[k] = v;
               visited[v] = true; //Xác nhận đỉnh v đã duyệt
               Hamilton(k+1);
               visited[v] = false; //Back-tracking
         }
   }
}
```

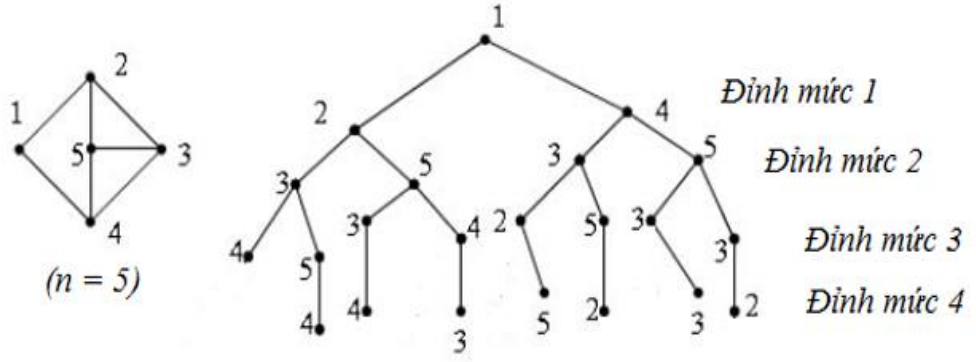
Hình 4.1. Thuật toán liệt kê các chu trình Hamilton bắt đầu tại đỉnh k

Khi đó, việc liệt kê chu trình Hamilton được thực hiện như sau:

```
Begin
        for(v ∈ V ) visited[v] = false;
        X[1] = v0;
        visited[v0] = true;
        Hamilton(2);
End.
```

4.3) Kiểm nghiệm thuật toán

Ví dụ: Đồ thị G=<V, E> dưới đây sẽ cho ta cây tìm kiếm chu trình Hamilton thể hiện thuật toán trên được mô tả như trong hình



Hình 4.2. Cây tìm kiếm chu trình Hamilton

Chọn **đỉnh 1** làm *gốc*, ghép tất cả các đỉnh kề liên tiếp từ đỉnh này sao cho không tạo thành chu trình.

Từ các **đỉnh 4** và **2** *(mức 4)* có đường đi tới gốc nên ta có các chu trình Hamilton <1,2,3,5,4,1>, <1,2,5,3,4,1>, <1,4,3,5,2,1>, <1,4,5,3,2,1>.

Vậy đồ thị đã cho là đồ thị Hamilton. (Có hai chu trình Hamilton khác nhau là <1,4,3,5,2,1> và <1,4,5,3,2,1>.

4.4) Cài đặt thuật toán

(Sử dụng ngôn ngữ lập trình C++11 để cài đặt thuật toán)

```
#include<bits/stdc++.h>
using namespace std;
vector<int> Adj[30];
int n, a[30][30], visited[30], X[30], check=0;
void Result(){
   check++;
   for(int i=1; i<=n; i++) cout<<X[i]<<" ";
   cout<<X[1]<<endl;
```

Chương trình liệt kê các chu trình Hamilton được thể hiện như sau:

```
}
int main(){
    memset(visited,0,sizeof(visited));
    cout<<"Nhap so dinh cua do thi: ";
    cin>>n;
    cout<<"Nhap ma tran ke:\n";
    for(int i=1; i<=n; i++){
    for(int j=1; j<=n; j++){
            cin>>a[i][j];
            if(a[i][j]) Adj[i].push_back(j);
            }
    }
    X[1]=1; visited[1]=1;
    Hamilton(2);
    if(!check) cout<<"Khong co chu trinh Hamilton\n";
    return 0;
}
```

Hình 4.3. Chương trình C++ sử dụng thuật toán tìm chu trình Hamilton

Ví dụ:

Input	Output
5 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0	1 2 3 5 4 1 1 2 5 3 4 1 1 4 3 5 2 1 1 4 5 3 2 1

Kết quả chạy chương trình:

```
Nhap so dinh cua do thi: 5
Nhap ma tran ke:
0 1 0 1 0
1 0 1 0 1
0 1 0 1 1
1 0 1 0 1
0 1 1 1 0
1 2 3 5 4 1
1 2 5 3 4 1
1 4 3 5 2 1
1 4 5 3 2 1
```

Hình 4.4. Kết quả tìm chu trình Hamilton

Cũng giống như thuật toán tìm chu trình Hamilton, thuật toán tìm đường đi Hamilton được cài đặt như sau:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> Adj[30];
int n, a[30][30], visited[30], X[30], check=0;
void Result(){
    check++;
    for(int i=1; i<=n; i++) cout<<X[i]<<" ";
    cout<<endl;
}
void Hamilton( int k){
    for(int v:Adj[X[k-1]]){
            if(!visited[v]){
                    X[k]=v;
                    visited[v]=1;
                    if(k==n) Result();
                    else Hamilton(k+1);
                    visited[v]=0;
            }
    }
}
int main(){
    memset(visited,0,sizeof(visited));
    cout<<"Nhap so dinh cua do thi: ";
    cin>>n;
    cout<<"Nhap ma tran ke:\n";
    for(int i=1; i<=n; i++){
            for(int j=1; j<=n; j++){
                    cin>>a[i][j];
                    if(a[i][j]) Adj[i].push_back(j);
            }
    }
    X[1]=1; visited[1]=1;
    Hamilton(2);
    if(!check) cout<<"Khong co duong di Hamilton\n";
}
```

Hình 4.5. Chương trình C++ sử dụng thuật toán tìm đường đi Hamilton

Ví dụ:

Input	Output
5 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0	1 2 3 4 5 1 2 3 5 4 1 2 5 3 4 1 2 5 4 3 1 4 3 2 5 1 4 3 5 2 1 4 5 2 3 1 4 5 3 2

Kết quả chạy chương trình:

```
Nhap so dinh cua do thi: 5
Nhap ma tran ke:
0 1 0 1 0
1 0 1 0 1
0 1 0 1 1
1 0 1 0 1
0 1 1 1 0
1 2 3 4 5
1 2 3 5 4
1 2 5 3 4
1 2 5 4 3
1 4 3 2 5
1 4 3 5 2
1 4 5 2 3
1 4 5 3 2
```

Hình 4.6. Kết quả tìm đường đi Hamilton

5. Bài toán Mã đi tuần

Bài toán Mã đi tuần (The Knight's Tour) là dạng tổng quát bài toán tìm đường đi Hamilton trong lĩnh vực toán rời rạc nói chung và lý thuyết đồ thị nói riêng.

INPUT:

Cho vị trí quân mã bắt đầu tại 1 vị trí.

OUTPUT :

Tìm ra hành trình đóng của quân mã sao cho đi hết 64 ô và quay trở lại ô bắt đầu trở thành 1 chu trình hamilton.

Ví Dụ:

INPUT	OUTPUT
Nhap n: 5 Nhap vi tri ban dau . X: 1 Y: 1	Cac buoc di la: 19 8 13 2 25 14 1 18 7 12 9 20 3 24 17 4 15 22 11 6 21 10 5 16 23

5.1) Bài toán:

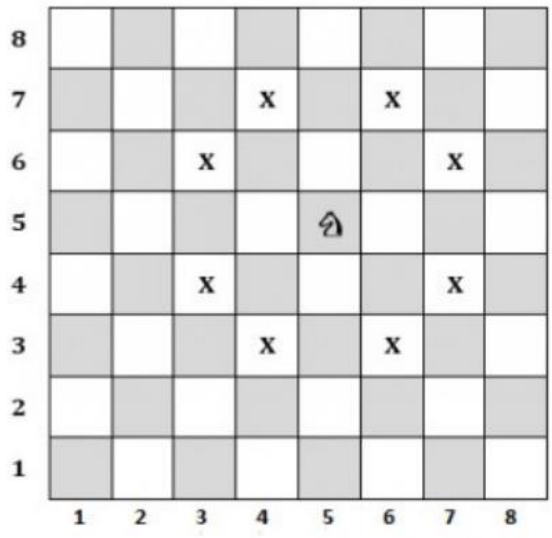
• Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.

• Nếu một quân mã đi hết 64 vị trí và tại vị trí cuối cùng có thể di chuyển đến vị trí bắt đầu thông qua một nước cờ thì đó gọi là một hành trình đóng. (chu trình Hamilton)

• Có những hành trình, trong đó quân mã sau khi đi hết tất cả 64 ô của bàn cờ và từ ô cuối của hành trình không thể đi về ô xuất phát chỉ bằng một nước đi. Những hành trình như vậy được gọi là hành trình mở. (đường đi Hamilton)

5.2) Cách di chuyển của một quân mã:

Nước đi của một quân mã giống hình chữ L và nó có thể di chuyển tất cả các hướng. Ở một vị trí thích hợp thì quân mã có thể di chuyển đến được 8 vị trí.



Hình 5.1. Cách di chuyển của một quân mã

5.3) Xây dựng bước đi cho quân mã:

• Gọi x,y là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ của quân mã sẽ như sau: |x| + |y| = 3 (Với |x|,|y| > 0).

• Khi đó ở một vị trí bất kì quân mã có có 8 đường có thể di chuyển. Chưa xét đến bước đi đó có hợp lệ hay không.

• Các bước đi đó là: (-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1), (2, 1)

• Để đơn giản ta sẽ tạo hay mảng X[], Y[ ] để chứa các giá trị trên. Với mỗi X[i], Y[i] sẽ là một cách di chuyển của quân mã($0 \le i \le 7$).

5.4) Kiểm tra tính hợp lệ của bước đi:

• Ta sẽ dùng một mảng hai chiều A[n*n] để lưu vị trí của từng ô trong bàn cờ. Tất cả mảng đều khởi tạo giá trị là 0(quân mã chưa đi qua).

• Gọi x,y là vị trí hiện tại của quân mã, thì vị trí tiếp theo mà quân mã đi sẽ có dạng x+X[i], y+Y[i]. Một vị trí được gọi là hợp lệ thì sẽ thỏa mãn tính chất sau: $+ 0 \le x+X[i] \le n-1$. $+ 0 \le y+Y[i] \le n-1$.

• Nếu bước đi đó là bước đi đúng thì ta sẽ lưu thứ tự của bước đi đó vào mảng A[x+X[i], y+Y[i]]

Sử dụng ngôn ngữ C++ để giải bài toán mã đi tuần :

Ta sử dụng hàm **diChuyen()** chính là hàm quan trọng nhất vì nó quyết định cách di chuyển của một quân mã. Ý tưởng của hàm là từ một vị trí ban đầu ta sẽ tìm một bước di chuyển hợp lệ và di chuyển đến vị trí đó. Tiếp tục tìm một bước đi và di chuyển tiếp đến khi không thể di chuyển được nữa (vào thế bí) thì sẽ xóa bước đi đó. Chương trình chỉ dừng khi tìm được một hành trình hoặc đã thử di chuyển hết tất cả các bước đi nhưng vẫn không tìm thấy hành trình.

```
void diChuyen(int x, int y) {
        ++dem;//Tăng giá trị bước đi
        A[x][y] = dem;//Đánh dấu đã đi
        for (int i = 0; i < 8; i++)     {
                //Kiểm tra xem mã đã đi hết bàn cờ chưa
                if (dem == n * n) {
                        cout << "Cac buoc di la: \n";
                        xuat();
                        exit(0);//kết thúc chương trình
                }
                //Nếu chưa đi hết bàn cờ thì tạo bước đi mới
                int u = x + X[i];//tạo một vị trí x mới
                int v = y + Y[i];//tạo một vịi trí y mới
                //Nếu hợp lệ thì tiến hành di chuyển
                if (u >= 0 && u < n&&v >= 0 && v < n&& A[u][v] == 0)
                        diChuyen(u, v);
        }
        //Nếu không tìm được bước đi thì ta phải trả lại các giá trị ban đầu
        --dem;
        A[x][y] = 0;
}
```

Hình 5.2. Thuật toán di chuyển

5.5) Toàn bộ bài toán bằng ngôn ngữ C++:

```
#include<iostream>
#include<stdio.h>
#define MAX 8
using namespace std;

int A[MAX][MAX] = { 0 };//Khởi tạo mảng giá trị 0
int X[8] = { -2,-2,-1,-1, 1, 1, 2, 2};
int Y[8] = { -1, 1,-2, 2,-2, 2,-1, 1};
int dem = 0;//Số bước đi
int n;

void xuat() {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++)
			printf("%2d ", A[i][j]);
		cout << endl;
	}
	cout << endl;
}

void diChuyen(int x, int y) {
	++dem;//Tăng giá trị bước đi
	A[x][y] = dem;//Đánh dấu đã đi
	for (int i = 0; i < 8; i++)	{
		//Kiểm tra xem mã đã đi hết bàn cờ chưa
		if (dem == n * n) {
			cout << "Cac buoc di la: \n";
			xuat();
			exit(0);//kết thúc chương trình
		}
		//Nếu chưa đi hết bàn cờ thì tạo bước đi mới
		int u = x + X[i];//tạo một vị trí x mới
		int v = y + Y[i];//tạo một vịi trí y mới
		//Nếu hợp lệ thì tiến hành di chuyển
		if (u >= 0 && u < n&&v >= 0 && v < n&& A[u][v] == 0)
			diChuyen(u, v);
	}
	//Nếu không tìm được bước đi thì ta phải trả lại các giá trị ban đầu
	--dem;
	A[x][y] = 0;
}
int main() {
	cout << "Nhap n: ";
	cin >> n;
	int a, b;
	cout << "Nhap vi tri ban dau.\nx: ";
	cin>>a;
	cout << "y: ";
	cin >> b;
	diChuyen(a, b);
	//Nếu không tìm được bước đi thì sẽ thông báo
	cout << "Khong tim thay duong di.";
}
```

Hình 5.3. Cài đặt thuật toán mã đi tuần bằng C++

6. Những điểm cần ghi nhớ

- Định nghĩa và định lý về đồ thị Hamilton và đồ thị nửa Hamilton.
- Nắm vững và phân biệt rõ sự khác biệt giữa chu trình (đường đi) Hamilton và chu trình (đường đi) Euler.
- Phương pháp hiểu rõ bản chất của thuật toán là cài đặt kiểm chứng thuật toán bằng cách viết chương trình.
- Thuật toán tìm kiếm chu trình Hamilton cho đến nay vẫn chưa được tối ưu với không gian tìm kiếm lớn do sử dụng phương pháp quay lui.
- Các dạng bài toán mở liên quan đến lý thuyết đồ thị Hamilton

Tài liệu tham khảo:

1. Nguyễn Duy Phương, *Bài giảng toán rời rạc 1*, Học viện Công nghệ Bưu Chính viễn thông, 2013
2. Nguyễn Duy Phương, *Bài giảng toán rời rạc 2*, Học viện Công nghệ Bưu Chính viễn thông, 2016
3. Nguyễn Đức Nghĩa - Nguyễn Tô Thành, *Giáo trình toán rời rạc*, NXB Đại học Quốc gia Hà Nội, 2009
4. Kenneth H. Rosen. Discrete Mathematics and its Applications, sixth Edition. McGraw-Hill, 2007.