CHƯƠNG 4

ĐỒ THỊ EULER VÀ ĐỒ THỊ HAMILTON

(tuần 5&6: Tổng cộng có 4 tiết lý thuyết và 4 tiết hướng dẫn bài tập/thực hành)

Trong chương này chúng ra sẽ nghiên cứu hai dạng đồ thị đặc biệt là đồ thị Euler và đồ thị Hamilton. Dưới đây, nếu không có giải thích bổ sung, thuật ngữ đồ thị được dùng để chỉ chung đa đồ thị vô hướng và có hướng, và thuật ngữ cạnh sẽ dùng để chỉ chung cạnh của đồ thị vô hướng cũng như cung của đồ thị có hướng.

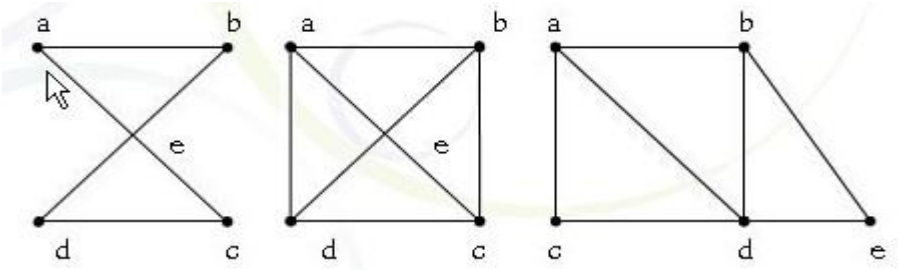
4.1. ĐỒ THỊ EULER

Định nghĩa 1. Chu trình đơn trong đồ thị G đi qua mỗi cạnh của nó một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler, và gọi là đồ thị nửa Euler nếu nó có đường đi Euler.

Rõ ràng mọi đồ thị Euler luôn là nửa Euler, nhưng điều ngược lại không luôn đúng.

Thí dụ 1.

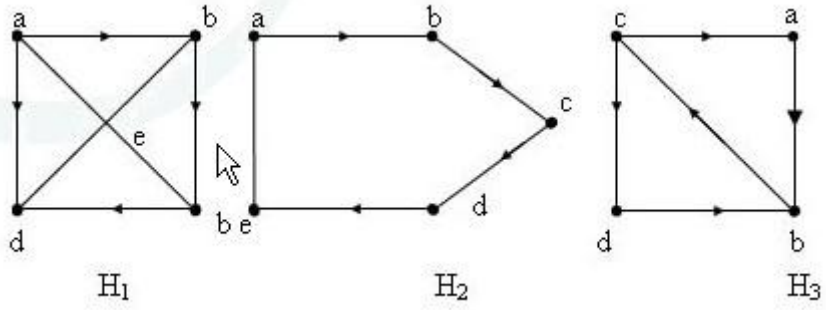
Đồ thị G_1 trong hình 1 là đồ thị Euler vì nó có chu trình Euler a, e, c, d, e, b, a. Đồ thị G_3 không có chu trình Euler nhưng nó có đường đi Euler a, c, d, e, b, d, a, b, vì thế G_3 là đồ thị cửa Euler. Đồ thị G_2 không có chu trình cũng như đường đi Euler.



Hình 1. Đồ thị G_1, G_2, G_3

Thí dụ 2.

Đồ thị H_2 trong hình 2 là đồ thị Euler vì nó có chu trình Euler a, b, c, d, e, a. Đồ thị H_3 không có chu trình Euler nhưng nó có đường đi Euler c, a, b, c, d, b vì thế H_3 là đồ thị nửa Euler. Đồ thị H_1 không có chu trình cũng như đường đi Euler.



Hình 2. Đồ thị H_1, H_2, H_3

Điều kiện cần và đủ để một đồ thị là một đồ thị Euler được Euler tìm ra vào năm 1736 khi ông giải quyết bài toán hóc búa nổi tiếng thế giới thời đó về bảy cái cầu ở thành phố Konigsberg và đây là định lý đầu tiên của lý thuyết đồ thị.

Định lý 1 (Euler).

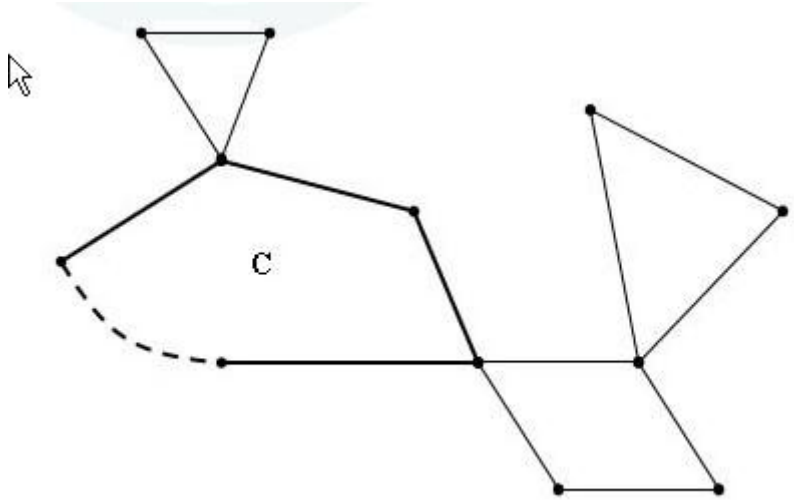
Đồ thị vô hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều có bậc chẵn.

Hệ quả.

Đồ thị vô hướng liên thông G là nửa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

Thuật tóan 1: Xây dựng một chu trình Euler (mở rộng chu trình):

Bắt đầu từ một đỉnh a, đi theo các cạnh của đồ thị một cách ngẫu nhiên nhưng không lặp lại cạnh nào đã đi qua cho đến khi không thể đi tiếp được nữa, nghĩa là phải dừng ở một đỉnh b nào đó. Lúc này mọi cạnh tới đỉnh b đều đã đi qua, nếu đỉnh a khác đỉnh b thì dễ thấy rằng số lần đi đến đỉnh b nhiều hơn số lần rời khỏi đỉnh b là 1: vô lý. Vậy phải có đỉnh b trùng với đỉnh a. Nói cách khác khi không thể di tiếp được là các cạnh đã đi qua đã tạo thành một chu trình. Nếu có một đỉnh c trong chu trình này là đỉnh đầu của một cạnh chưa đi qua thì ta sẽ mở rộng chu trình này thành một chu trình lớn hơn bằng cách khởi hành lại từ đỉnh c, rồi tiếp tục đi theo cạnh tới đỉnh c chưa đi qua nói ở trên cho đến khi không thể đi tiếp được nữa, ta sẽ tạo được một chu trình mói chứa chu trình cũ. Cứ tiếp tục như thế cho đến khi được một chu trình mà không thể tiếp tục mở rộng ra được nữa, đều này xảy ra khi mọi đỉnh trong chu trình hiện có đều không còn cạnh nào chưa đi qua.



Hình 4. Minh hoạ cho chứng minh định lý 1

Giả sử G là đồ thị Euler, thuật toán đơn giản sau đây cho phép xác định chu trình Euler khi làm bằng tay.

```
void Euler_Cycle()
{
    STACK=∅; CE=∅;
    Chon u la mot dinh nao do cua do thi;
    STACK⇐u;
    while STACK<>∅
    {
        x=top(STACK);  //x la phan tu dau STACK
        if Ke(x)<>∅
        {
            y=dinh dau tien trong danh sach Ke(x);
            STACK⇐y;
            // loai bo canh (x,y) khoi do thi
            Ke(x)=Ke(x)\{y};
            Ke(y)=Ke(y)\{x};
        }
        else
        {
            x⇐STACK;
            CE⇐x;
        }
    }
}
```

Thuật toán 2: (Thuật toán Flor)

Xuất phát từ một đỉnh u nào đó của G ta đi theo các cạnh của nó một cách tuỳ ý chỉ cần tuân thủ 2 qui tắc sau:

(1) Xoá bỏ cạnh đã đi qua đồng thời xoá bỏ cả những đỉnh cô lập tạo thành.

(2) Ở mỗi bước ta chỉ đi qua cầu khi không còn cách lựa chon nào khác.

Định lý 2.

Đồ thị có hướng liên thông mạnh là đồ thị Euler khi và chỉ khi

$deg^+(v)=deg^-(v)$, □ $v \in V$.

4.2.ĐỒ THỊ HAMILTON

Trong mục này chúng ta xét bài toán tương tự như trong mục trước chỉ khác là ta quan tâm đến đường đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần. Sự thay đổi tưởng chừng như là không đáng kể này trên thực tế đã dẫn đến sự phức tạp hoá vấn đề cần giải quyết.

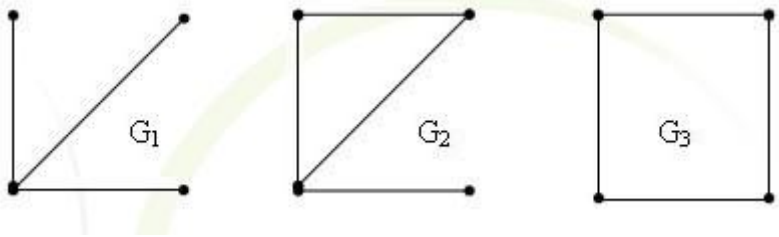
Định nghĩa 2.

Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Chu trình bắt đầu từ một đỉnh v nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần rồi quay trở về v được gọi là chu trình Hamilton. Đồ thị G được gọi là đồ thị Hamilton nếu nó chứa chu trình Hamilton và gọi là đồ thị nữa Hamilton nếu nó có đường đi Hamilton.

Rõ ràng đồ thị Hamilton là nửa Hamilton, nhưng điều ngược lại không còn đúng.

Thí dụ 3.

Trong hình 3: G_3 là Hamilton, G_2 là nửa Hamilton còn G_1 không là nửa Hamilton.



Hình 3. Đồ thị Hamilton G_3, nửa Hamilton G_2 , và G_1.

Cho đến nay việc tìm một tiêu chuẩn nhận biết đồ thị Hamilton vẫn còn là mở, mặc dù đây là một vấn đề trung tâm của lý thuyết đồ thị. Hơn thế nứa, cho đến nay cũng chưa có thuật toán hiệu quả để kiểm tra một đồ thị có là Hamilton hay không. Các kết quả thu được phần lớn là điều kiện đủ để một đồ thị là đồ thị Hamilton. Phần lớn chúng điều có dạng "nếu G có số cạnh đủ lớn thì G là Hamilton". Một kết quả như vậy được phát biểu trong định lý sau đây.

Định lý 3 (Dirak 1952). Đơn đồ thị vô hướng G với n>2 đỉnh, mỗi đỉnh có bậc không nhỏ hơn n/2 là đồ thị Hamilton.

Định lý 4.

Giả sử G là đồ có hướng liên thông với n đỉnh. Nếu $\deg^+(v) \geq n/2$, $\deg^-(v) \geq n/2$, $\forall v$ thì G là Hamilton.

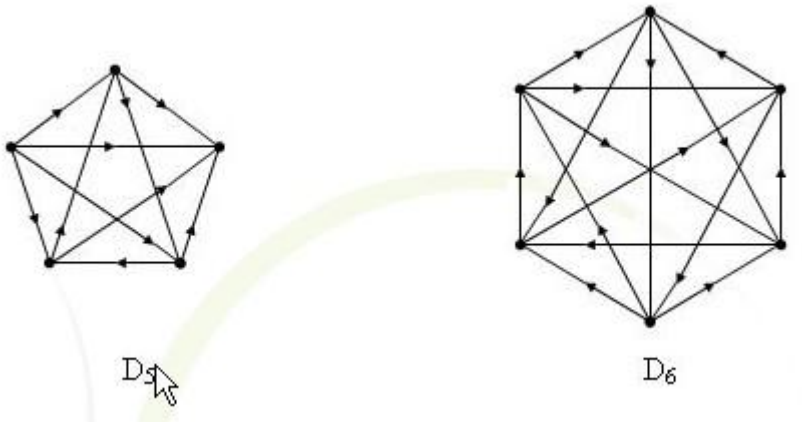
Có một số dạng đồ thị mà ta có thể biết khi nào là đồ thị Hamilton. Một ví dụ như vậy là đồ thị đấu loại. Đồ thị đấu loại là đồ thị có hướng mà trong đó hai đỉnh bất kỳ của nó được nối với nhau bởi đúng một cung. Tên đấu loại xuất hiện như vậy vì đồ thị như vậy có thể dùng để biểu diễn kết quả thi đấu bóng chuyền, bóng bàn hay bất cứ một trò chơi nào mà không cho phép hoà. Ta có định lý sau:

Định lý 5.

i) Mọi đồ thị đấu loại là nửa Hamilton.

ii) Mọi đồ thị đấu loại liên thông mạnh là Hamilton.

Thí dụ 4. Đồ thị đấu loại D_5, D_6 được cho trong hình 4.



Hình 4. Đồ thị đấu loại D_5, đấu loại liên thông mạnh D_6

Liệt kê các chu trình Hamilton

(đối với đồ thị không có trọng số và được biểu diễn bằng ma trận kề)

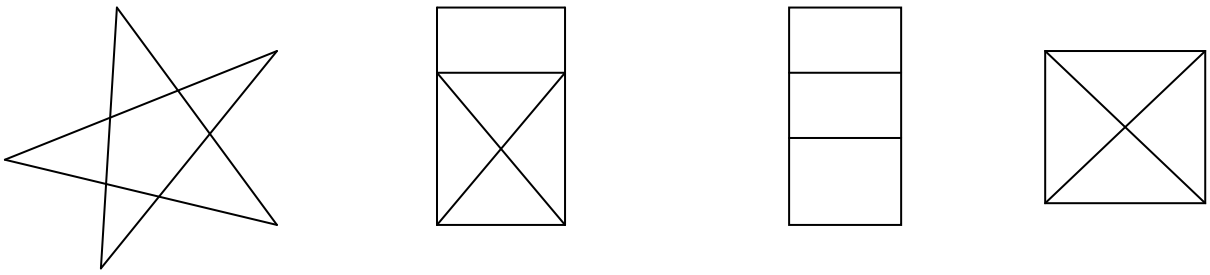
Gợi ý thuật tóan sau :

```
void Hamilton(k);
(*Liệt kê các chu trình Hamilton thu được bằng việc phát triển dãy X[1],X[2],…,X[K-1]
của đồ thị G=(V,E) cho bởi  danh sách kề *)
{
        for y ∈ ke(X[k-1])
                if (k==n-1 && y=vo)
                        ghinhan(X[1],X[2],..,X[N],Vo)
                else
```

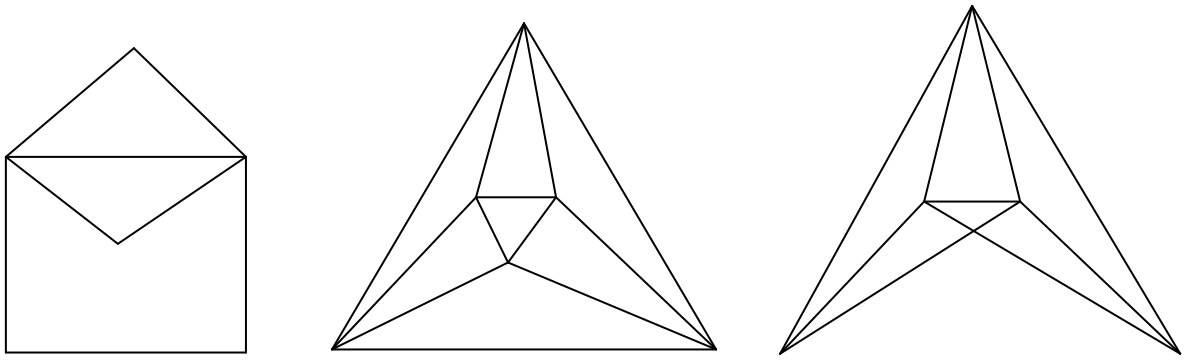
```
                if (chuaxet(y))
                {
                        X[k]=y;
                        chuaxet[y]=0;
                        Hamilton(k+1)
                        chuaxet[y]=1;
                }
}
void main()
{
        for v ∈ V chuaxet(v)=1
        X[1]=vo; (*vo là một đỉnh nào đó của đồ thị*)
        chuaxet(vo)=0;
        Hamilton(2);
}
```

Bài tập lý thuyết:

4-1. Kiểm tra xem các đồ thị sau có chứa chu trình hoặc đường đi Euler hay không ?



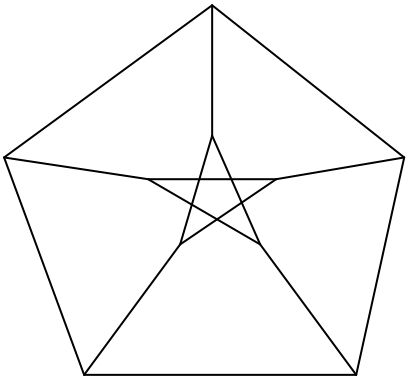
4-2. Kiểm tra xem các đồ thị sau có chứa chu trình hoặc đường đi Euler hay không ?



4-3. Cho đồ thị PeterSon(P)

a.Tìm một đường đi Hamilton trong P

b.P có chứa chu trình Hamilton hay không ?



4-4.Hãy cho ví dụ về:

a.Đồ thị có một chu trình, vừa là chu trình Euler vừa là chu trình Hamilton

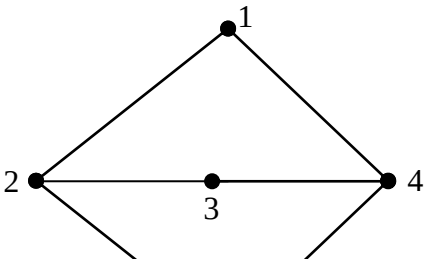
b.Đồ thị có 1 chu trình Euler , 1 chu trình Hamilton , nhưng hai chu trình đó không trùng nhau

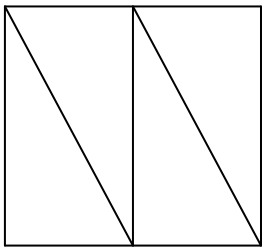
c.Đồ thị có 6 đỉnh, là đồ thị Hamilton, nhưng không là đồ thị Euler

d.Đồ thị có 6 đỉnh, là đồ thị Euler, nhưng không là đồ thị Hamilton

4-5. Kiểm tra xem các đồ thị sau có chứa chu trình hoặc đường đi Hamilton hay không ?

`





Bài tập thực hành

4-6.Bài toán người đưa thư:

Cho n thành phố, các thành phố được đánh số từ 1 đến n. Một người đưa thư muốn đi qua tất cả các con đường, mỗi con đường đúng 1 lần. Hãy lập lộ trình cho người đưa thư hoặc thông báo không tồn tại đường đi.

4-7. Cho đồ thị được biểu diễn bằng ma trận kề. Hãy liệt kê tất cả các chu trình Hamilton của đồ thị.

4-8.Bài toán hành trình người bán hàng (TRAVELING SALESMAN PROBLEM)

Có n thành phố (được đánh số từ 1 đến n), một người bán hàng xuất phát từ một thành phố, muốn đi qua các thành phố khác, mỗi thành phố một lần rồi quay về thành phố xuất phát.. Giả thiết biết được chi phí đi từ thành phố i đến thành phố j là c[I,j]. Hãy tìm một hành trình cho người bán hàng sao cho tổng chi phí theo hành trình này là thấp nhất.

4-9.Kiểm tra đường

Một trạm quảng đường giao thông phải chịu trách nhiêm về tình trạng của một mạng lưới giao thông nối giữa các điểm dân cư. Hàng tháng, họ phải cử một đội đi kiểm tra một vòng qua khắp mạng lưới để xem xét tình trạng hiện thời của các đường giao thông nhằm báo sửa chữa kịp thời nếu có nhu cầu. Hãy viết chương trình nhập vào mạng lưới giao thông và giúp trạm quyết định lộ trình của đội kiểm tra sao cho có thể thăm tất cả các con đường mà tổng chiều dài đoạn đường đi qua là nhỏ nhất.

4-10.Hội nghị bàn tròn

Tổng thư ký Đại hội đồng Liên hợp quốc triệu tập một cuộc họp có N nhà ngoại giao của N tổ chức tham gia. Các đại diện ngoại giao được bố trí ngồi quanh một bàn tròn. Giữa một số tổ chức có quan hệ căng thẳng, vì vậy không thể xếp họ ngồi cạnh nhau được. Thông tin về quan hệ giữa các tổ chức được cho dưới dạng cặp số nguyên i, j nếu giữa 2 tổ chức này có quan hệ căng thẳng.

Hãy lập trình giúp Tổng thư ký Liên hợp quốc bố trí chỗ ngồi quanh bàn họp. Các tổ chức được đánh số từ 1 tới N, $0 < N <= 500$.

Dữ liệu vào: từ file CONF.INP, dòng đầu tiên chứa số nguyên N, các dòng sau, mỗi dòng một cặp số i, j cho biết các đại diện i và j không ngồi cạnh nhau được. Kết thúc là một dòng chứa 2 số 0.

Kết quả: đưa ra file CONF.OUT. Nếu không có cách bố trí thỏa mãn yêu cầu thì đưa ra thông báo KHONG CO, trong trường hợp ngược lại – đưa ra dãy N số nguyên xác định vị trí ai ngồi cạnh ai quanh bàn tròn.

Ví dụ:

CONF.INP CONF.OUT

11 1 9 7 4 11 5 8 2 10 3 6

1 4

1 7

5 7

10 7

10 8

10 9

3 4

0 0

Cài đặt một số thuật toán căn bản quan trọng

1.Tìm chu trình Euler (bài toán người đưa thư).

(Lưu ý chương trình sau được cài đặt theo thuật toán quay lui/vét cạn)

```
1.  #include <conio.h>
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.  #include <iostream.h>
5.  int  m[20][20];
6.  int  a[20];
7.  int  b[20];
8.  int  tc,dau,n;
9.  int  co;
10. void nhap()
11. {
12. FILE *f;
13. int  i,j;
14. f=fopen("D:\\DOTHI\\EULER1.inp","rt");
15. fscanf(f,"%d",&n);
16. cout<<n<<endl;
17. for (i=1;i<=n;i++)
18. {
19. for (j=1;j<=n;j++)
20. {
21. fscanf(f,"%d",&m[i][j]);
22. cout<<m[i][j]<<" ";
23. }
24. cout<<endl;
25. }
26. tc=0;
27. for (i=1;i<=n-1;i++)
28. for (j=i+1;j<=n;j++)
29. if (m[i][j]==1)
30. tc++;
31. cout<<"Chu trinh euler bat dau tu dinh : ";cin>>dau;
```

```
co=0;
// cout<<tc;
}
void xuat()
{
int i;
cout<<dau;
for (i=1;i<=tc;i++)
{
cout<<"--->"<<a[i];
co=1;
}
cout<<endl;
getch();
exit(1);
}
void euler(int d,int i)
{
int j;
for (j=1;j<=n;j++)
if (m[d][j]==1)
{
a[i]=j;
m[d][j]=0;
m[j][d]=0;
if (i == tc)  xuat();
else          euler(j,i+1);
m[d][j]=1;
m[j][d]=1;
}
}
void main()
{
nhap();
```

```
66. euler(dau,1);
67. if (!co) cout<<"khong co duong di :";
68. getch();
69. }
```

EULER1.inp

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

EULER2.inp

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

EULER3.inp

```
4
0 1 1 1
1 0 0 0
1 0 0 0
1 0 0 0
```

Yêu cầu sinh viên cài đặt thuật toán tìm chu trình Euler như đã nêu trong phần lý thuyết.

2.Tìm chu trình Hamilton (bài toán người bán hàng)

Đi qua mỗi đỉnh một lần sao cho tổng chi phí là thấp nhất.

Thuật toán tối ưu cho bài toán hành trình người bán hàng

Besttour=0;

Bestcost=maxint;

while tìm thấy một hoán vị {xi} của tập {1,2,…,n}

{

với mỗi hoán vị {xi} ta được một hành trình, tính chi phí cho từng hành trình này

cost=0;

for i:=1 to n-1

cost+=c[xi,x_{i+1}];

cost+=c[xn,x1];

if cost<bestcost

{ bestcots=cost

besttour=x

}

}

//TRAVELING SALESMAN PROBLEM-

(thuật toán tối ưu cho bài toán TSP)

```
1. #include <iostream.h>
2. #include <conio.h>
3. #include <stdio.h>
4. #include <values.h>
5. void output();
6. void readfile();
7. void permute(int i);
8. void result();
9. const max=50;
10. int   c[max][max];
11. int   tourbest[max],b[max],x[50],n,costbest;
12. void main()
13. {
14. readfile();
```

```
15. permute(1);
16. result();
17. }
18. void  output()
19. {
20. int cost=0;
21. for (int i=1;i<n;i++)
22. cost=cost+c[x[i]] [x[i+1]];
23. cost=cost+c[x[n]] [x[1]];
24. if (cost<costbest)
25. {
26. costbest=cost;
27. for (int j=1;j<=n;j++)
28. tourbest[j]=x[j];
29. }
30. }
31. void permute(int i)
32. {
33. for (int j=1;j<=n;j++)
34. if (b[j]==1)
35. {
36. x[i]=j;        b[j]=0;
37. if (i==n) output();
38. else      permute(i+1);
39. b[j]=1;
40. }
41. }
42. void readfile()
43. {
44. clrscr();
45. FILE *f;
46. f=fopen("d:\\dothi\\tsp1.inp","rt");
47. fscanf(f,"%d",&n);
48. for (int i=1;i<=n;i++)
```

```
49. for (int j=1;j<=n;j++)
50. {
51. fscanf(f,"%d",&c[i][j]);
52. if (i==j) c[i][j]=MAXINT;
53. }
54. fclose(f);
55. costbest=MAXINT;
56. for (i=1;i<=n;i++) b[i]=1;
57. }
58. void result()
59. {cout<<costbest<<endl;
60. for (int i=1;i<=n;i++)       cout<<tourbest[i]<<" ";
61. getch();
62. }
```

Tsp1.inp

```
6
0 3 5 7 5 6
2 0 2 3 1 7
4 7 0 2 2 4
3 7 4 0 5 2
4 6 2 4 0 2
5 6 2 5 3 0
```

Tsp2.inp

```
5
0 1 2 7 4
1 0 4 4 3
2 4 0 1 2
7 4 1 0 4
5 3 2 4 0
```

Tsp3.inp

```
5
```

0 21 40 32 28

15 0 18 37 25

19 17 0 6 7

9 50 28 0 4

25 6 5 10 0

SỬ DỤNG NGUYÊN LÝ THAM LAM GIẢI BÀI TÓAN NGƯỜI BÁN HÀNG:

Với những bài toán mà không gian trạng thái có thể phát sinh cực lớn thì việc dùng phương pháp vét cạn là điều không thể. Nguyên lý tham lam lấy tiêu chuẩn tối ưu toàn cục để làm tiêu chuẩn chọn lựa hành động trong phạm vi cục bộ. Một số ví dụ có thể áp dụng nguyên lý này như các bài toán có mô hình toán học là bài toán người bán hàng, bài toán tô màu đồ thị. Hơn nữa nếu có một chiến lược tham lam hợp lý, thì phương pháp này sẽ tìm được lời giải tối ưu (thuật toán Kruskal, thuật toán Prim).

Lược đồ của phương pháp tham lam

```
Procedure Greedy(A,S)        { là tập các ứng cử viên, S là tập nghiệm}
begin
        S=ϕ
        While A ≠ ϕ do
                Begin
                        X=select(A); { chọn phần tử tốt nhất trong A}
                        A=A-{x}
                        If S ∪ {x} chấp nhận được then        S= S ∪ {x}
                End;
End;
```

Thuật giải GTS1(u) Thuật giải GTS1 (Greedy Traveling Saleman)

INPUT: số thành phố là n đỉnh xuất phát u và ma trận chi phí c

OUTPUT: tour (thứ tự các thành phố đi qua), cost – chí phí ứng với tour tìm được

```
v:=u;
tour:={u};
cost=0;
for i=1 to n
{đặt w là thành phố kề sau thành phố v.
        tour=tour + {w};
```

```
            cost=cost+c[v,w]
            v=w;
        }
        tour=tour + {u};
        cost=cost+c[v,u]
```

THUẬT GIẢI GTS2

```
Input       n, c, p,Vi (i=1..p)// vi là các thành phố được chọn ngẫu
            //  nhiên trong tập n thành phố ban đầu
Output:     besttour, bestcost
        Bestcost=0
        Besttour={}
        For i=1 to p
        {       GTS1(vk); // suy ra được tour(vk) và  cost(vk)
                If cost(vk)<bestcost
                        {       bestcost=cost(vk)
                                besttour=tour(vk)
                        }
        }
```