Bài toán du lịch

Nguyễn Minh Nhật+Trương Trần Tiến

I)Nhắc lại bài Toán

1)Bài toán chung

Bài toán người du lịch(hay bài toán người bán hàng) được phát biểu như sau:

"Có một người giao hàng cần đi giao hàng tại n thành phố. Anh ta xuất phát từ một thành phố nào đó, đi qua các thành phố khác để giao hàng và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, và khoảng cách từ một thành phố đến các thành phố khác đã được biết trước. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất."

Ở một số bài toán ta có thể cho khoảng cách giữa 2 chiều là như nhau.Một số bài toán thì 2 chiều của con đường có thể có độ dài khác nhau.Và sự giống nhau đó có thể giảm đến $\frac{1}{2}$ số lời giải có thể.

2)Nội dung bài toán tìm hiểu

Ở đây thì ta sẽ lần lượt đưa ra các giải pháp với trường hợp "dễ" của bài toán:Ta đã biết được rằng ta sẽ bắt đầu ở thành phố nào (ở đây cụ thể là 1).Và độ dài đoạn đường từ i đến j bằng độ dài đoạn đường từ j đến i.Ta cũng đã biết số tuyến đường.Phát biểu lại bài toán như sau:

"Có một người giao hàng cần đi giao hàng tại n thành phố,có m tuyến đường giữa m thành phố đó. Anh ta xuất phát từ thành phố số 1, đi qua các thành phố khác để giao hàng và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, và khoảng cách từ một thành phố đến các thành phố khác đã được biết trước. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất."

Thật ra xuất phát từ đâu cũng như nhau.Do nếu là đã có 1 đường đi khép kín thoả mãn điều kiện thì bắt đầu tại đâu cũng được.

II)Một số thuật toán giải bài toán trên

Lưu ý:

-Ta sẽ bỏ qua chi phí phần chuẩn bị là có độ phức tạp xấp xỉ O(n^2).

-Chương trình có thuật toán áp dụng cho bài toán chung được chỉnh sửa lại để phù hợp với bài toán tìm hiểu

1)Thuật toán Láng giềng gần nhất

a)Giới thiệu thuật toán

Đây là 1 thuật toán áp dụng tư tưởng tham lam,tư tưởng thuật toán này khá đơn giản,độ phức tạp thấp O(n*m).

Ưu điểm	Khuyết điểm
Chạy nhanh Độ phức tạp thấp Dễ cài đặt	Khả năng chạy sai rất cao. Sai số thường không chấp nhận được (Trung bình là 20% so với lời giải tối ưu).Trong 1 số trường hợp bài toán in nghiệm tệ nhất.Có khi bài toán in ra nghiệm dù không có nghiệm và không in ra nghiệm dù có nghiệm

b)Tư tưởng+Thuật toán

Gọi i là thành phố hiện tại $(1 \le i \le n)$,ta sẽ chọn 1 thành phố thứ j nào đó sao cho khoảng cách từ i đến j là bé nhất

Các bước của thuật toán:

1. Chọn một nút bất kỳ làm nút xuất phát và đây là nút hiện hành
2. Đánh dấu nút hiện hành là đã được đi qua
3. Tìm một nút chưa đi qua có khoảng cách đến nút hiện hành là ngắn nhất, đánh dấu nút này là nút hiện hành mới
4. Nếu chưa đi qua tất cả các nút thì quay lại bước 2

c)Chương trình tham khảo

//Chương trình được trích từ tài liệu giáo khoa chuyên Tin-quyển 1

```
const fi ='DULICH.INP';
      fo ='DULICH.OUT';
var c :array[1..20,1..20]of longint;
    x,d :array[1..20]of longint;
    n,sum,m :longint;
procedure input;
var f :text;
    i,j,a,chiphi :longint;
begin
```

```
        {$I-}
        assign(f,fi);
        reset(f);
        read(f,n,m);
        For i:=1 to n-1 do
        Begin
                For j:=i+1 to n do
                Begin
                        c[i,j]:=1000000;
                        c[j,i]:=1000000;
                End;
        End;
        for i:=1 to m do
        Begin
                read(f,a,j,chiphi);
                C[a,j]:=chiphi;
                c[j,a]:=chiphi;
        End;
        close(f);
        {$I+}
end;
procedure output;
var f :text;
    i :longint;
begin
        {$I-}
        assign(f,fo);
```

```
        rewrite(f);
        for i:=1 to n do write(f,x[i],'->');
        writeln(f,x[1]);
        writeln(f,'Cost: ' ,sum);
        close(f);
end;
procedure Greedy;
var i,j,xi :longint;
    best :longint;
begin
        x[1]:=1;
        d[1]:=1;
        i:=1;
        while i<n do
        begin
                i:=i+1;
                best:=1000000;
                for j:=1 to n do
                begin
                        if (d[j]=0) and (c[x[i-1],j]<best) then
                        begin
                                best:=c[x[i-1],j];
                                xi:=j;
                        end;
                end;
                x[i]:=xi;
                d[xi]:=1;
```

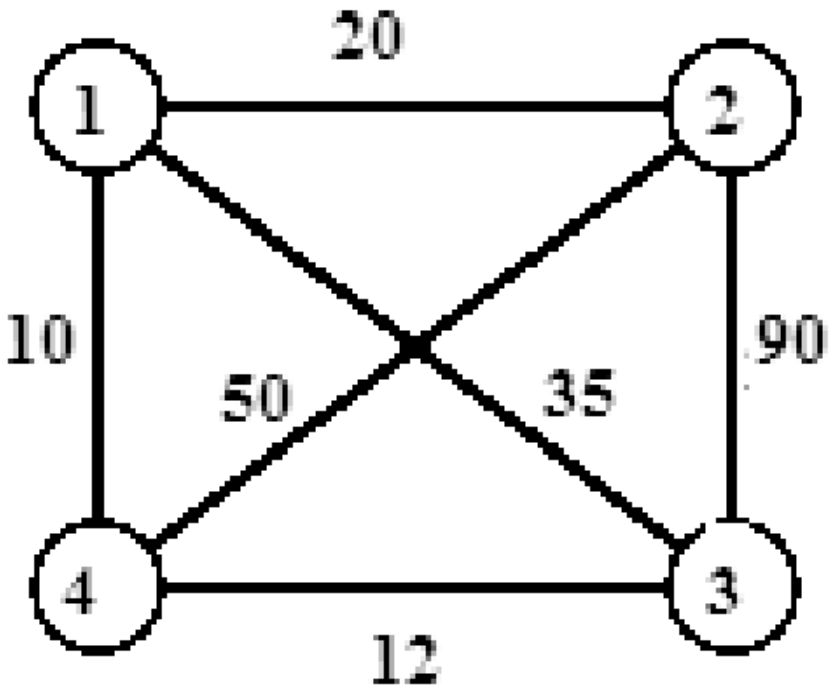
```
                sum:=sum+c[x[i-1],x[i]];
        end;
        sum:=sum+c[x[n],x[1]];
end;
BEGIN
        input;
        Greedy;
        output;
END.
```

d)Nhận xét

Chương trình chạy dưới 1s với $n, m \leq 10^3$

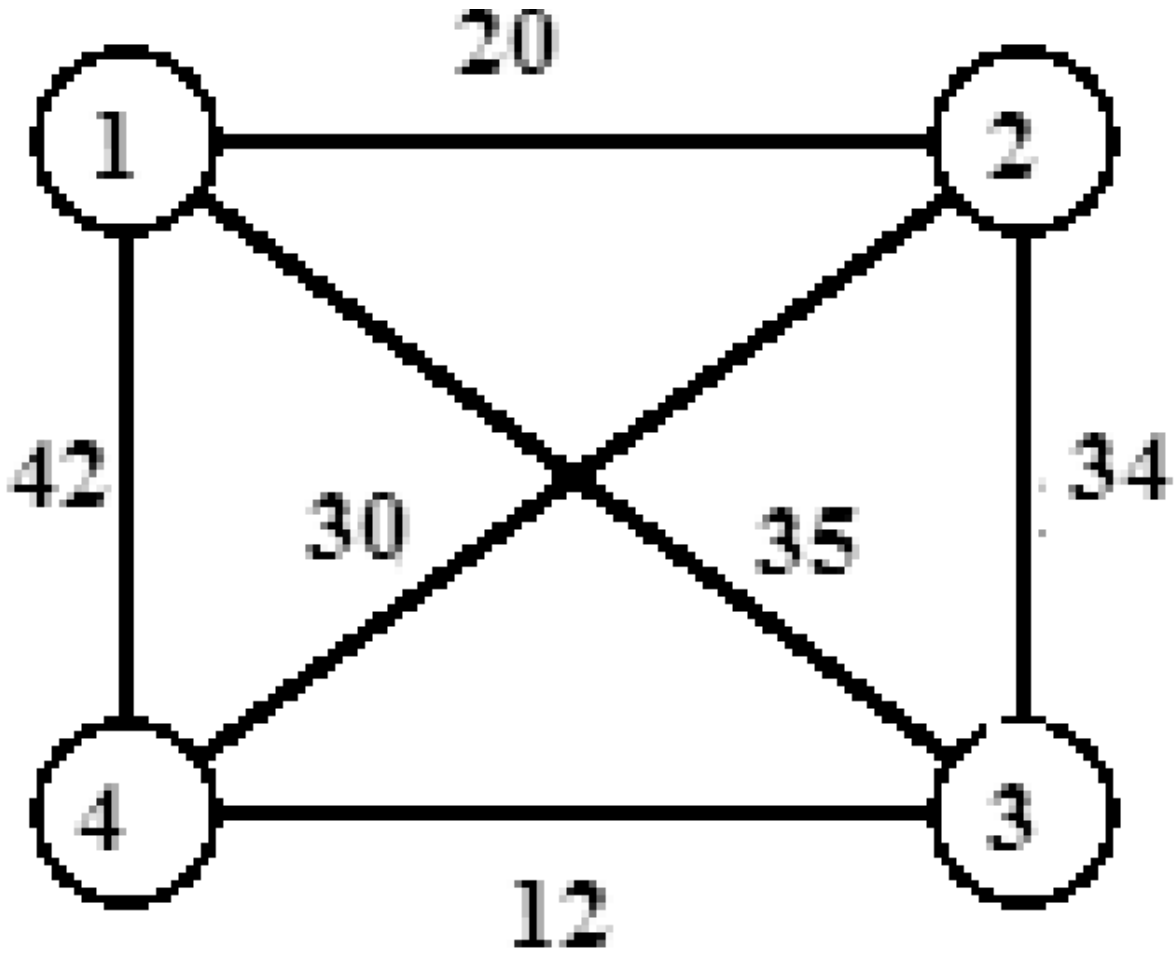
Chạy thử 1 số test

Test 1:



Với ví dụ trên,chương trình sẽ cho hành trình là 1→4→3→2→1 với chi phí 132, nhưng kết quả tối ưu là 117(1→2→4→3→1)

Test 2:



Chương trình cho hành trình là 1→2→4→3→1 với chi phí là 97.Đây là phương án tối ưu

2)Thuật toán vét cạn(Quay lui-đệ quy)

a)Giới thiệu thuật toán

Tìm hết tất cả các nghiệm(đường đi),chọn nghiệm tốt nhất(đường đi ngắn nhất).Do thứ tự đường đi là 1 dãy hoán vị của (1..n) do đó nếu duyệt hết sẽ phải duyệt $n!$đường đi.Độ phức tạp là O(n!)

Ưu điểm	Khuyết điểm
Luôn luôn cho kết quả đúng	Chạy chậm

a)Chương trình tham khảo

```
var     C: array [1..100,1..100] of LongInt;
        x,best: array[1..100] of Integer;
        t: array[1..100] of LongInt;
     free: array[1..100] of Boolean;
      maxC,maxC1: LongInt;
        n,m: Integer;
        f: Text;
procedure DocFile;
Var a,b,chiphi,i:Integer;
begin
        assign(f,'DULICH.INP');
        reset(f);
        readln(f,n,m);
        For i:=1 to m do
        begin
              read(f,a,b,chiphi);
              c[a,b]:=chiphi;
              c[b,a]:=chiphi;
        end;
        close(f);
end;
procedure ChuanBi;
var i,j: Integer;
begin
    for i:=1 to n-1 do
```

```
    Begin
      c[i,i]:=0;
      for j:=1 to n do
      begin
            C[i,j]:= maxC1;
            C[j,i]:= macC1;
      end;
      FillChar(free,100,True);
end;
procedure Duyet(i: Integer);
var u: Integer;
begin
      for u:=2 to n do
      begin
            if free[u]= true then
            begin
                  x[i]:=u;
                  t[i]:=t[i-1] + C[x[i-1],u];
                  if i<n then
                  begin
                        free[u]:= False;
                        DuyetNhanhCan(i+1);
                        free[u]:= true;
                  end
                  else
```

```
                begin
                    if (t[n] + C[x[n],1] < maxC) then
                    Begin
                        maxC:= t[n] + C[x[n],1];
                        best:=x;
                    End;
                end;
            end;
        end;
end;
procedure XuLi;
begin
    free[1]:= False;
    x[1]:=1;
    t[1]:=0;
    Duyet(2);
end;
procedure GhiFile;
var i: Integer;
begin
    assign(f,'DULICH.OUT');
    rewrite(f);
    For i:=1 to n do Write(f,best[i],'=>');
    Writeln(f,1);
    write(f,'Cost: ',maxC);
```

```
    close(f);
end;
begin
    maxC := 1000000;
    maxC1:= 1000000;
    ChuanBi;
    DocFile;
    XuLi;
    GhiFile;
end.
```

d)Nhận xét

Chương trình chạy dưới 1s với $n, m \leq 10$

3)Thuật toán Nhánh-cận

a)Giới thiệu thuật toán

Trong thực tế, có nhiều bài toán yêu cầu tìm ra một phương án thoả mãn một số điều kiện nào đó, và phương án đó là tốt nhất theo một tiêu chí cụ thể. Các bài toán như vậy được gọi là bài toán tối ưu.

Ta sẽ tiến hành Quay lui để vét nghiệm.Giả sử ta hiện có nghiệm tối ưu nhất là A{a_1, a_2 a_3, a_4,…, a_n} có "độ tốt" là i và ta cần một nghiệm nào đó "tốt" nhất.Thì với nghiệm b{b_1, b_2 b_3, b_4,…, b_m} khi đang xét đến thành phần thứ b_j $(1 \leq j \leq m)$mà độ tốt của nghiệm b hiện tại đã giảm còn k mà k "không tốt" bằng i thì ta không cần phải xét tiếp để có toàn nghiệm b đó.(Do là càng mở rộng nghiệm ra thì "độ tốt" càng giảm).Và khi gặp 1 nghiệm hoàn chỉnh c{c_1,c_2 c_3, c_4,…, c_x} "tốt" hơn nghiệm A (nghĩa là c là nghiệm tối ưu nhất) thì ta sẽ cập nhật lại "độ tốt" i của A = "độ tốt" của c(ta cập nhật lại độ "tốt" của nghiệm "tốt nhất" A).Và vì có 1 số nghiệm ta không xét(hoặc không xét hết) nên số bước phải làm sẽ giảm đi 1 cách đáng kể

Ưu điểm	Khuyết điểm
Cũng như Quay lui nhưng sẽ nhanh hơn quay lui	Rất khó để quyết định cách nhánh- cận do đặc thù đề bài
Có thể rất nhanh O(n) và cũng có thể rất chậm O(3*n!) tuỳ trường hợp do trong 1 số trường hợp thì đó vẫn là quay lui(nếu nghiệm được sắp từ "tốt" đến "tốt nhất") mà mỗi bước duyệt phải tốn thêm O(1) để kiểm tra và cả phần cập nhật nghiệm O(1)	

b)Tư tưởng+Thuật toán

Tóm tắt lại các bước của thuật toán nhánh cận cũng giống như thuật toán vét cạn nhưng sẽ có thêm:

-Gọi A{a_1, a_2 a_3, a_4,…, a_n} là nghiệm tối ưu nhất mà ta đang có.Bước đầu tiên là khởi tạo độ tốt i của A= ∞ để đề phòng bài toán không có nghiệm và để A ban đầu là nghiệm tệ nhất (bất kỳ nghiệm nào cũng tốt hơn A)

-Phần kiểm tra "độ tốt" của thành phần đầu đến thứ i của nghiệm d so với "độ tốt" của nghiệm tốt nhất đang có là A để ta đánh giá có mở rộng tiếp nghiệm d cho đến khi d thành nghiệm hoàn chỉnh hay không ?

-Phần cập nhật "độ tốt",nhằm để cập nhật A liên tục để A luôn là "tốt nhất".Lưu ý là trong phần này có thể luôn cả phần cập nhật lại nghiệm A nếu chương trình yêu cầu in ra nghiệm "tốt nhất" đó.

c)Chương trình tham khảo

```
var     C: array [1..100,1..100] of LongInt;

       x,best: array[1..100] of Integer;

       t: array[1..100] of LongInt;

     free: array[1..100] of Boolean;

      maxC,maxC1: LongInt;

       n,m: Integer;

       f: Text;

procedure DocFile;

Var a,b,chiphi,i:Integer;

begin

       assign(f,'DULICH.INP');

       reset(f);

       readln(f,n,m);

       for i:=1 to n-1 do

       Begin

            C[i,i]:=0;

            for a:=1 to n do
```

```
            begin
                C[i,a]:=maxC1;
                C[a,i]:=maxC1;
            end;
        end;
        FillChar(free,100,True);
        For i:=1 to m do
        begin
            read(f,a,b,chiphi);
            c[a,b]:=chiphi;
            c[b,a]:=chiphi;
        end;
        close(f);
end;
procedure DuyetNhanhCan(i: Integer);
var u: Integer;
begin
        for u:=2 to n do
        begin
            if free[u]= true then
            begin
                x[i]:=u;
                t[i]:=t[i-1] + C[x[i-1],u];
                if t[i]< maxC then
                begin
```

```
                    if i<n then
                    begin
                      free[u]:= False;
                      DuyetNhanhCan(i+1);
                      free[u]:= true;
                    end
                    else
                    begin
                       if (t[n] + C[x[n],1] < maxC) then
                       Begin
                          maxC:= t[n] + C[x[n],1];
                          best:=x;
                       End;
                    end;
                end;
            end;
      end;
end;
procedure XuLi;
begin
      free[1]:= False;
      x[1]:=1;
      t[1]:=0;
      DuyetNhanhCan(2);
end;
```

```
procedure GhiFile;

var i: Integer;

begin

    assign(f,'DULICH.OUT');

    rewrite(f);

    For i:=1 to n do Write(f,best[i],'=>');

    Writeln(f,1);

    write(f,'Cost: ',maxC);

    close(f);

end;

begin

     maxC := 1000000;

     maxC1:= 1000000;

     DocFile;

     XuLi;

     GhiFile;

end.
```

d)Nhận xét

Chương trình trên là một giải pháp nhánh cận rất thô sơ, có thể có nhiều cách đánh giá nhánh cận chặt hơn nữa làm tăng hiệu quả của chương trình.Một cải tiến xin được đề xuất là:

If(t[i]+(n-i)*A_{min}< maxC)then …

Trong đó A_{min} là chi phí nhỏ nhất trong bảng chi phí di chuyển

4)Thuật toán Quy Hoạch Động

a)Giới thiệu thuật toán

Ta sẽ chia nhỏ bài toán thành các hành trình nhỏ hơn.Bắt đầu từ thành phố thứ i ($1 \le i \le n$¿ ta sẽ lần lượt thăm thành phố j ($1 \le j \le n$¿ và tương tự với j ta sẽ thăm các thành phố khác.Ta sẽ lưu các hành trình nhỏ vào bảng phương án.

Ưu điểm	Khuyết điểm
-Độ phức tạp thấp O($n^2 * 2^n$¿ -Chạy nhanh hơn đáng kể so với quay lui+nhánh cận(trong đa số các trường hợp)	-Khó cài đặt -Tiêu tốn bộ nhớ rất nhiều **O**$(n * 2^n)$ ô nhớ.

b)Tư tưởng+thuật toán

Gọi S là trạng thái của hành trình.S sẽ là 1 dãy gồm n phần tử.Trong đó:

-S[i]=1:Thành phố thứ i đã được thăm

-S[i]=0:Thành phố thứ i chưa được thăm

Ta rút ra nhận xét như sau:Ta sẽ ngốn thêm một lượng lớn ô nhớ nữa với mảng nếu ta khai báo S là 1 mảng để lưu trữ(Ta sẽ phải khai báo bảng phương án là nhiều chiều hoặc là mảng đánh số chuỗi).Điều đó rất là phi thực tế.

Do đó ta sẽ áp dụng xử lý bit để phần nào giảm lại kích thước yêu cầu của chương trình.Ta nhớ lại 1 điều.Giả sử 1 số được biểu diễn bằng n bit thì các bit đó sẽ được đánh dấu từ 0→n-1 bit.Do đó S sẽ là 1 dãy bit trong đó:

-S[i]=1:Thành phố thứ i+1 đã được thăm

-S[i]=0:Thành phố thứ i+1 chưa được thăm

Ví dụ, trạng thái S = 10101 = 21 thể hiện qua 3 thành phố 1, 3, 5.Ta không cần quan tâm đến thứ tự.

Gọi T[i, S] là chi phí nhỏ nhất để đến thành phố i với trạng thái S, ta có công thức quy hoạch động như sau:

T[i, S] = min(T[j,P] + C[j, i])

Với P là trạng thái liền trước của trạng thái S, là bản sao của S, chỉ khác ở bit i – 1 (thành phố i chưa được thăm); j là vị trí của các bit 1 trong trạng thái P (các thành phố đã được thăm ở trạng thái P)

Nghĩa là từ thành phố i $(1 \le i \le n)$ ta sẽ đến thành phố thứ j $(1 \le j \le n)$ với k thành phố đã được thăm $(1 \le k \le n)$.

Hay ta sẽ thử từng trạng thái S từ bé đến lớn (do ta phải có từng hành trình nhỏ trước).Và để có được trạng thái P ta sẽ tắt bit thứ i $(1 \le i \le n)$ của S.

Và trước đó ta phải khởi tạo mảng T[$\forall i, S$]=∞ để xác định là chưa có đường đi nào được tìm thấy

c)Chương trình tham khảo

```
Const fi='DULICH.INP';

    fo='DULICH.OUT';
```

```
Var c:array[1..4,1..4]of Word;
   T,Mark:array[1..4,0..15]of Word;
   n:Byte;
   b:array[1..4]of Byte;
   KetQua:Word;
Procedure DocFile;
Var f:Text;
   m,a,j,chiphi,i:Byte;
Begin
      {$I-}
      Assign(f,fi);
      Reset(f);
      Readln(f,n,m);
      For i:=1 to n-1 do
      Begin
            For j:=i+1 to n do
            Begin
                  c[i,j]:=MaxInt;
                  c[j,i]:=MaxInt;
            End;
      End;
      For i:=1 to m do
      Begin
            Read(f,a,j,chiphi);
            c[a,j]:=chiphi;
```

```
            c[j,a]:=chiphi;
        End;
        Close(f);
        {$I+}
End;
Function GetBit(state,i:Byte):Byte;
Begin
        GetBit:=(state shr i)and 1;
End;
Function TurnOff(state,i:Byte):Byte;
Begin
        TurnOff:=state and not (1 shl i);
End;
Procedure QHD;
Var u,i,j,k,state,prestate,first,last:Byte;
Begin
        first:=1;
        last:=1 shl n-1;
        KetQua:=MaxInt;
        For u:=1 to n do For state:=0 to last do T[u,state]:=MaxInt;
        For u:=1 to n do
        Begin
            T[u,1 shl (u-1)]:=0;
            Mark[u,1 shl (u-1)]:=u;
        End;
```

```
k:=0;
For state:=first to last do
Begin
    FillChar(b,k,0);
    k:=0;
    For i:=1 to n do
    Begin
        If(GetBit(state,i-1)=1)then
        Begin
            Inc(k);
            b[k]:=i;
        End;
    End;
    For i:=1 to k do
    Begin
        u:=b[i];
        prestate:=TurnOff(state,b[i]-1);
        For j:=1 to k do
        Begin
            If(T[u,state]>T[b[j],prestate]+c[b[j],u])then
            Begin
                T[u,state]:=T[b[j],prestate]+c[b[j],u];
                Mark[u,state]:=Mark[b[j],prestate];
                If(state=last)and(T[u,last]+c[u,Mark[u,state]]<KetQua)then KetQua:=T[u,last]
+c[u,Mark[u,state]];
            End;
```

```
                End;
            End;
        End;
End;
Procedure GhiFile;
Var f:Text;
Begin
    {$I-}
    Assign(f,fo);
    Rewrite(f);
    Writeln(f,KetQua);
    Close(f);
    {$I+}
End;
Begin
    DocFile;
    QHD;
    GhiFile;
End.
```

d)Nhận xét

-Chạy được với $n \leq 16$

-Chương trình trên có thể được cải tiến theo 2 hướng:

-Giảm thiểu mảng Mark(đánh dấu) bằng cách không đánh dấu bit bắt đầu hoặc dùng “thành phố phụ” có đầy đủ các tính chất của thành phố (bắt đầu).

-Cải thiện mảng Mark để truy vết hành trình.