GIẢI THUẬT QUAY LUI

Lời giải của bài toán thường biểu diễn bằng một vec tơ gồm n thành phần x=(x1, x2,...,xn) phải thỏa mãn điều kiện nào đó. Để chỉ ra lời giải x, ta phải xây dựng dần các thành phần lời giải xi.

Tại mỗi bước i:

- Đã xây dựng xong các thành phần x1,...,xi-1

- Xây dựng thành phần xi bằng cách lần lượt thử tất cả các khả năng mà xi có thể lựa chọn:

* Nếu chọn một khả năng j nào đó phù hợp cho xi thì xác định xi theo khả năng j. Thường phải có thêm các thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu i=n thì ta có được một lời giải, ngược lại thì tiến hành bước i+1 để xác định xi+1.
* Nếu không có một khả năng nào chấp nhận được cho xi thì ta lùi lại bước trước để xác định lại thành phần xi-1

Mô hình của phương pháp quay lui có thể viết bằng thủ tục sau, với n là số bước cần phải thực hiện, k là số khả năng mà xi có thể lựa chọn.

Try(i)

For(j=1→k)

If(xi chấp nhận được khả năng j)

{

Xác nhận xi theo khả năng j;

Ghi nhận trạng thái mới;

If(i<n)

Try(i+1);

Else

Ghi nhận nghiệm;

Trả lại trạng thái cũ cho bài toán;

}

BÀI TOÁN NGƯỜI ĐI DU LỊCH

1. Nêu bài toán người đi du lịch

Có n thành phố( được đánh số từ 1 đến n), chi phí đi từ thành phố i đến thành phố j là C[i,j].

Một người đi du lịch xuất phát từ một thành phố muốn đi thăm các thành phố khác, mỗi thành phố đúng một lần rồi quay về nơi xuất phát.

Hãy tìm một hành trình cho người đi du lịch để tổng chi phí theo hành trình này là ít nhất.

2. Mô tả chi tiết thuật toán

• Mỗi hành trình là một chu trình đi qua các thành phố {1, 2, .., n}.

• Cấu trúc lời giải là dãy X[1..n] trong đó:

– X[1] = 1 được xem là thành phố xuất phát.

– X[2..n] chứa hoán vị của dãy {2, 3, .. ,n}

• Tổng chi phí của một hành trình bằng: T=(∑C[Xi,Xi+1] ) + C[Xn,X1]

\* Thuật toán xác định phần tử Xi

void Try(int i) {

for (int j = 2; j <= n; j++)

if (F[ j ] == 0) {

X[ i ] = j;

T[i]=T[i-1]+C[X[i-1][j];

if ( i <n ) {

F[ j ] = 1;

Try( i + 1 );

F[ j ] = 0;

}

Else

<Cập nhật hành trình tối ưu> ;

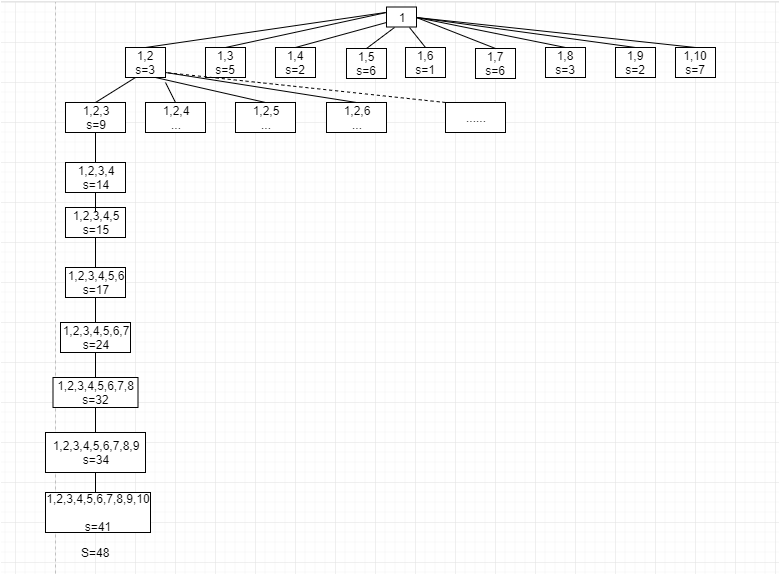
}

}

3. Kiểm tra với 5 bộ dữ liệu

3.1. Bộ dữ liệu 1

| 0 | 3 | 5 | 2 | 6 | 1 | 6 | 3 | 2 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 6 | 7 | 3 | 1 | 5 | 7 | 1 | 7 |
| 5 | 6 | 0 | 5 | 4 | 6 | 5 | 6 | 3 | 2 |
| 2 | 7 | 5 | 0 | 1 | 3 | 7 | 6 | 2 | 6 |
| 6 | 3 | 4 | 1 | 0 | 2 | 1 | 3 | 2 | 1 |
| 1 | 1 | 6 | 3 | 2 | 0 | 7 | 4 | 5 | 6 |
| 6 | 5 | 5 | 7 | 1 | 7 | 0 | 8 | 4 | 3 |
| 3 | 7 | 6 | 6 | 3 | 4 | 8 | 0 | 2 | 5 |
| 2 | 1 | 3 | 2 | 2 | 5 | 4 | 2 | 0 | 7 |
| 7 | 7 | 2 | 6 | 1 | 6 | 3 | 5 | 7 | 0 |



Thực hiện từng bước của thuật toán ta liệt kê lần lượt các đường đi của người đi du lịch và chi phí tương ứng.

Sau khi cập nhật hành trình tối ưu có đường đi ngắn nhất ta được hành trình tối ưu với đỉnh xuất phát là 1

1->4->5->7->10->3->8->9->2->6->1

Cost : 20

3.2. Bộ dữ liệu 2

| 0 | 2 | 1 | 7 | 1 | 2 | 6 | 5 | 7 | 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 0 | 4 | 2 | 3 | 5 | 8 | 1 | 8 | 9 |
| 1 | 4 | 0 | 6 | 9 | 0 | 8 | 2 | 6 | 8 |
| 7 | 2 | 6 | 0 | 5 | 2 | 3 | 9 | 4 | 3 |
| 1 | 3 | 9 | 5 | 0 | 6 | 4 | 7 | 2 | 2 |
| 2 | 5 | 0 | 2 | 6 | 0 | 4 | 3 | 1 | 6 |
| 6 | 8 | 8 | 3 | 4 | 4 | 0 | 2 | 7 | 5 |
| 5 | 1 | 2 | 9 | 7 | 3 | 2 | 0 | 2 | 3 |
| 7 | 8 | 6 | 4 | 2 | 1 | 7 | 2 | 0 | 5 |
| 4 | 9 | 8 | 3 | 2 | 6 | 5 | 3 | 5 | 0 |

Với bộ dữ liệu 2 với đỉnh xuất phát là 2:

2->3->1->6->9->5->10->4->7->8->2

Cost : 21

3.3. Bộ dữ liệu 3

| 0 | 3 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 2 | 0 | 3 | 4 | 0 | 0 | 0 | 0 |
| 1 | 2 | 0 | 7 | 2 | 0 | 0 | 0 | 5 | 0 |
| 4 | 0 | 7 | 0 | 4 | 0 | 0 | 0 | 7 | 5 |
| 0 | 3 | 2 | 4 | 0 | 1 | 7 | 4 | 6 | 0 |
| 0 | 4 | 0 | 0 | 1 | 0 | 8 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 7 | 8 | 0 | 3 | 8 | 0 |
| 0 | 0 | 0 | 0 | 4 | 0 | 3 | 0 | 2 | 6 |
| 0 | 0 | 5 | 7 | 6 | 0 | 8 | 2 | 0 | 1 |
| 0 | 0 | 0 | 5 | 0 | 0 | 0 | 6 | 1 | 0 |

1->3->2->6->5->4->10->9->8->7->1

Cost : 23

3.4. Bộ dữ liệu 4

| 0 | 1 | 8 | 3 | 7 | 2 | 6 | 5 | 3 | 4 | 2 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 4 | 2 | 2 | 8 | 7 | 8 | 5 | 9 | 1 | 4 |
| 8 | 4 | 0 | 6 | 9 | 7 | 8 | 2 | 1 | 7 | 4 | 7 |
| 3 | 2 | 6 | 0 | 5 | 2 | 0 | 1 | 4 | 3 | 5 | 8 |
| 7 | 2 | 9 | 5 | 0 | 6 | 4 | 7 | 6 | 1 | 6 | 9 |
| 2 | 8 | 7 | 2 | 6 | 0 | 4 | 3 | 1 | 2 | 7 | 2 |
| 6 | 7 | 8 | 0 | 4 | 4 | 0 | 2 | 0 | 5 | 3 | 7 |
| 5 | 8 | 2 | 1 | 7 | 3 | 2 | 0 | 2 | 3 | 5 | 1 |
| 3 | 5 | 1 | 4 | 6 | 1 | 0 | 2 | 0 | 5 | 2 | 8 |
| 4 | 9 | 7 | 3 | 1 | 2 | 5 | 3 | 2 | 0 | 8 | 3 |
| 2 | 1 | 4 | 5 | 6 | 7 | 3 | 5 | 2 | 8 | 0 | 6 |
| 3 | 4 | 7 | 8 | 9 | 2 | 7 | 1 | 8 | 3 | 6 | 0 |

1->2->4->6->9->3->8->12->10->5->7->11->1

Cost : 23

3.5. Bộ dữ liệu 5

| 0 | 3 | 1 | 7 | 8 | 2 | 6 | 5 | 3 | 4 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | 4 | 2 | 2 | 8 | 7 | 8 | 5 | 9 | 4 |
| 1 | 4 | 0 | 6 | 9 | 7 | 8 | 2 | 1 | 7 | 7 |
| 7 | 2 | 6 | 0 | 5 | 2 | 0 | 1 | 4 | 3 | 8 |
| 8 | 2 | 9 | 5 | 0 | 6 | 4 | 7 | 6 | 1 | 9 |
| 2 | 8 | 7 | 2 | 6 | 0 | 4 | 3 | 1 | 2 | 2 |
| 6 | 7 | 8 | 0 | 4 | 4 | 0 | 2 | 0 | 5 | 7 |
| 5 | 8 | 2 | 1 | 7 | 3 | 2 | 0 | 2 | 3 | 1 |
| 3 | 5 | 1 | 4 | 6 | 1 | 0 | 2 | 0 | 5 | 8 |
| 4 | 9 | 7 | 3 | 1 | 2 | 5 | 3 | 2 | 0 | 3 |
| 3 | 4 | 7 | 8 | 9 | 2 | 7 | 1 | 8 | 3 | 0 |

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

Cost : 21

4. Viết chương trình

| #include <iostream>  #include <fstream>  using namespace std;  #define MAX 100  #define maxE 1000  #define INF 99999  int C[MAX + 1][MAX + 1]; //ma tran chi phi  int X[MAX + 2]; // X de thu cac kha nang  int BestWay[MAX + 2]; //BestWay de ghi nhan nghiem  int T[MAX + 1]; //T[i] Luu chi phi di tu X[1] den X[i]  bool Free[MAX + 1]; //danh dau, true neu chua di qua thanh pho i  int minSpending; // chi phi hanh trinh toi uu (nho nhat)  int m, n;  int start;// lua chon diem bat dau cho  void Input()  {  int i, j;  fstream infile;  infile.open("input2.txt");  infile >> n;  cout << "- So thanh pho la: " << n;  cout << "\n- Ma tran ke la:";  for (i = 1; i <= n; i++)  {  cout << "\n";  for (j = 1; j <= n; j++)  {  infile >> C[i][j];  cout << "\t" << C[i][j];  }  }  cout << "\n";  infile.close();  }  void Init()  {  for (int i = 1; i <= n; i++)  Free[i] = true;//cac thanh pho deu danh dau la chua di qua  Free[start] = false; //tru thanh pho 1  X[1] = start; //vị trí bat dau  T[start] = 0; //chi phi tai thanh pho xuat phat la 0  minSpending = INF;  }  void PrintResult()  {  if (minSpending == INF)  cout << "NO SOLUTION ";  else  {  for (int i = 1; i <= n; i++)  cout << BestWay[i] << "->";  cout << start << endl;  cout << "Cost : " << minSpending;  }  }  void Try(int i)  {  for (int idx = start + 1; idx <= n+start-1; idx++) //thu cac tp tu 2 den n  {  int j=idx;  if(j>n)j=j-n;  if (Free[j]) // neu gap thanh pho chua di qua  {  X[i] = j; //thu thanh pho tiep theo  T[i] = T[i - 1] + C[X[i - 1]][j]; //chi phi=chi phi buoc truoc+ chi phi di truc tiep  if (C[X[i - 1]][j] > 0)// co duong di truc tiep tu X[i-1] den j  {  if (i < n) //neu chua den duoc X[n]  {  Free[j] = false; //danh dau thanh pho  Try(i + 1); //tim cac kha nang chon  Free[j] = true; //bo qua danh dau de quay len nhanh khac  }  else  {  if ((T[n] + C[X[n]][start]) < minSpending)// khi chay den vi tri n so sanh tong gia tri trươc do và gtri(vitridau->vtri cuoi) neu nho hon thi cap nhat  {  for (int i = 0; i <= n; i++)  BestWay[i] = X[i];// luu cac dinh da xet  minSpending = T[n] + C[X[n]][start];  }  }  }  }  }  }  int main()  {  cout<<"Nhap dinh bat dau:";  cin>>start;  Input();  Init();  Try(2);  PrintResult();  return 0;  } |
| --- |

5. Đánh giá độ phức tạp của thuật toán theo lý thuyết và thực nghiệm

- Độ phức tạp xấp xỉ của thuật toán người đi du lịch theo phương pháp quay lui là nn (với n là số đỉnh).