**Cài đặt thuật toán Ford-Bellman**

**Cách làm**

Ta bắt đầu từ đỉnh nguồn duy nhất đến các đỉnh khác trong đồ thị có hướng có trọng số.Ford-Bellmancó thể hoạt động với đồ thị có trọng số âm nhưng không thể hoạt động với đồ thị có trọng số cạnh âm.

**Các bước**

1. Đầu tiên ta khởi tạo khoảng cách từ nguồn đến tất cả các đỉnh là vô hạn và khoảng cách tới nguồn là 0.

typedef struct edge

{

int src;

int dest;

int wt;

}edge;

void bellman\_ford(int nv,edge e[],int src\_graph,int ne)

{

int u,v,weight,i,j=0;

int dis[MAX];

for(i=0;i<nv;i++)

// khởi tạo biến dis[i] với giá trị là 999 . Giá trị 999 biểu thị cho khoảng cách là vô hạn

{

dis[i]=999;

}

// khoảng cách của đỉnh nguồn từ đỉnh nguồn là 0

dis[src\_graph]=0;

1. Sau đó ta kiểm tra chu kì của đồ thì bằng cách với mỗi đỉnh u -> v của đồ thị ta kiểm tra nếu

Dis[u]!=999 && dis[u] + weight < dis[v] thì dis[v]=dis[u]+weight;

1. Lặp lại việc kiểm tra chi kì với mỗi đỉnh u -> v, nv – 1 lần trong đó nv là số lượng đỉnh của đồ thị

for(i=0;i<nv-1;i++)

{

for(j=0;j<ne;j++)

{

u=e[j].src;

v=e[j].dest;

weight=e[j].wt;

if(dis[u]!=999 && dis[u]+weight < dis[v])

{

dis[v]=dis[u]+weight;

}

}

}

1. Nếu dis[u]+weight < dis[v] thì in ra chu kì âm của đồ thị đã cho

cout<<"\nChu kì đồ thị mới nhất!!\n";

return;

1. Ta bắt đầu nhập số lượng đỉnh và chọn đỉnh nguồn(đỉnh bắt đầu) , nhập số cạnh , nhập đỉnh nguồn và đỉnh kết thúc của cạnh .Cuối cùng ta nhập trọng số của cạnh

int nv,ne,src\_graph;

edge e[MAX];

cout<<"Nhập số đỉnh: ";

cin>>nv;

printf("Nhập đỉnh nguồn của đồ thị : ");

cin>>src\_graph;

cout<<"\n Nhập số cạnh : ";

cin>>ne;

for(int i=0;i<ne;i++)

{

cout<<"\n Cho cạnh : "<<i+1<<"=>";

cout<<"\nNhập đỉnh nguồn :";

cin>>e[i].src;

cout<<"Nhập vào đỉnh kết thúc :";

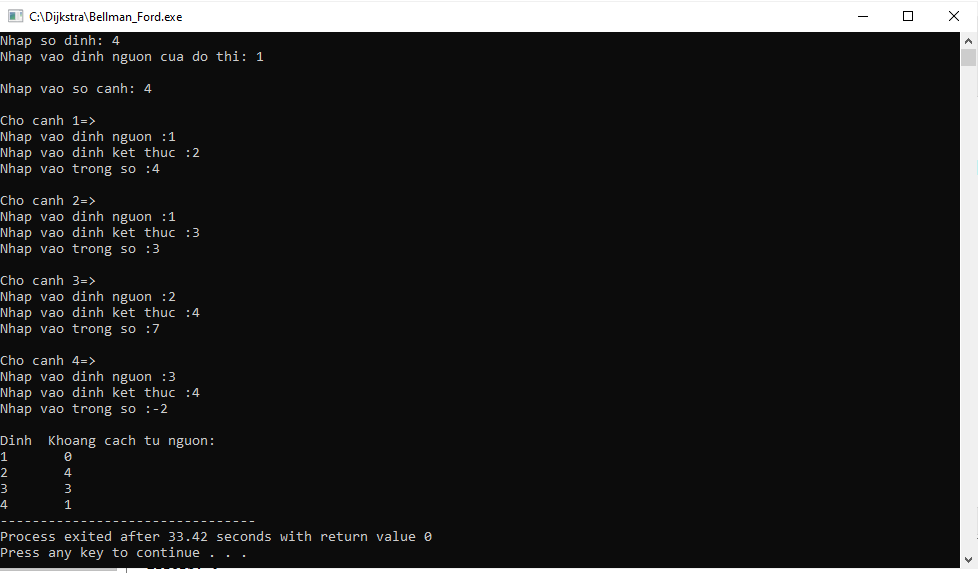
cin>>e[i].dest;

cout<<"Nhập trọng số:";

cin>>e[i].wt;

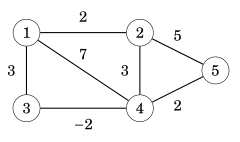
}

1. Sau khi chạy chương trình màn hình cosole sẽ hiện các thông tin cho bạn nhập cũng như in ra kết quả:

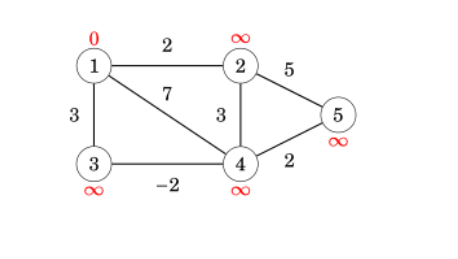


**Bài giải**

Ta có ví dụ minh họa với đồ thị sau:

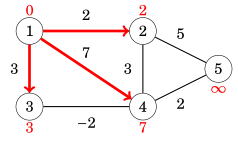


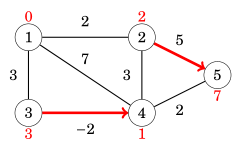
1. Ta chọn đỉnh nguồn là số 1 và khởi tạo khoảng cách đến chình nó là 0 còn đến các đỉnh còn lại là



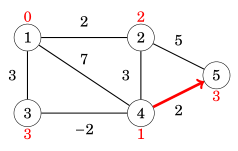
1. Lặp lại việc kiểm tra chi kì với mỗi đỉnh u -> v, nv – 1 lần trong đó nv là số lượng đỉnh của đồ thị.Vậy với ví dụ minh họa này ta sẽ thực hiện 4 vòng lặp:

Ở **vòng lặp đầu tiên**, ta cập nhật được đường đi ngắn nhất thông qua các cạnh (1, 2); (1, 3); (1, 4):



Ở **vòng lặp số 2**, cạnh (2, 5) và (3, 4) là các cạnh tối ưu:

Ở **vòng lặp số 3**, ta chỉ thấy có cạnh (4,5) cải thiện đường đi từ 1 -> 5 :



Ở **vòng lặp số 4**, ta nhận thấy không còn cạnh nào có thể tối ưu được đường đi từ node 1 nữa, vậy nên đồ thị này sẽ không có chu trình âm. Suy ra ta có thể kết thúc thuật toán tại đây.

Và đây là kết quả sau khi chạy chương trình với thuật toán **Ford-Bellman:**

