**Question1**

**A:** Here is the codes of this problem:

#include<iostream>

#include<vector>

#include<limits.h>

#include"Matrix.h"

using namespace std;

using namespace Numeric\_lib;

// p is vertex v's self, d is an upper bound in the weight of a shortest path from source s to v

struct vertex

{

int p;

int d;

};

void initialize\_single\_source(vector<vertex>&G)

{

unsigned int i;

for (i = 1; i < G.size(); i++)

G[i].d = INT\_MAX;

G[0].d = 0;

}

vertex extract\_min(vector<vertex>&Q)

{

unsigned int i;

vertex min;

int min1=0;

min = Q[0];

for (i = 0; i < Q.size(); i++)

{

if (Q[i].d <= min.d)

{

min = Q[i];

min1 = i;

}

}

Q.erase(Q.begin() + min1);

return min;

}

void relax(vertex &u, vertex &v, Matrix<int, 2>&w)

{

if (v.d > u.d + w(u.p, v.p))

v.d = u.d + w(u.p, v.p);

}

void dijkstra(vector<vertex>&G, Matrix<int, 2>&w,vector<vertex>&S)

{

initialize\_single\_source(G);

vector<vertex>Q;

vertex u;

Q = G;

int i;

unsigned int j;

while (Q.size()> 0)

{

u = extract\_min(Q);

S.push\_back(u);

for (i = 0; i < w.dim1(); i++)

{

if (w(u.p, i) != NULL)

{

for (j = 0; j < Q.size(); j++)

{

if (Q[j].p == i)

relax(u,Q[j],w);

}

}

}

}

}

// n is the number of nodes

int main()

{

int n=5;

unsigned int i;

vector<vertex>G(n);

vector<vertex>S;

Matrix<int, 2>w(n, n);

for (i = 0; i < G.size(); i++)

G[i].p = i;

w(0, 1) = 10;

w(0, 2) = 3;;

w(1, 2) = 1;

w(1, 3) = 2;

w(2, 1) = 4;

w(2, 3) = 8;

w(2, 4) = 2;

w(3, 4) = 9;

w(4, 3) = 11;

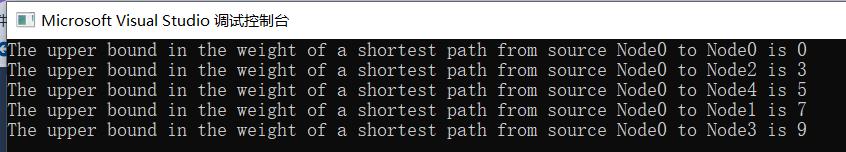
dijkstra(G, w,S);

for (i = 0; i < S.size(); i++)

cout <<"The upper bound in the weight of a shortest path from source Node0 to Node"<< S[i].p << " is " << S[i].d << endl;

}

And the result is below:



**Question2**

**A:** Here is the codes of this problem:

#include<iostream>

#include<vector>

#include<limits.h>

#include"Matrix.h"

using namespace std;

using namespace Numeric\_lib;

// p is vertex v's self, d is an upper bound in the weight of a shortest path from source s to v

struct vertex

{

int p;

int d;

};

void initialize\_single\_source(vector<vertex>&G)

{

unsigned int i;

for (i = 1; i < G.size(); i++)

G[i].d = INT\_MAX;

G[0].d = 0;

}

void relax(vertex &u, vertex &v, Matrix<int, 2>&w)

{

if (v.d > u.d + w(u.p, v.p))

v.d = u.d + w(u.p, v.p);

}

bool bellman\_ford(vector<vertex>&G, Matrix<int, 2>&w)

{

initialize\_single\_source(G);

unsigned int i,j,k,m;

for (i = 0; i < G.size()-1; i++)

{

for (j = 0; j < G.size(); j++)

{

for (k = 0; k < G.size(); k++)

{

if (w(j, k) != NULL)

{

for (m = 0; m < G.size(); m++)

{

if (G[m].p == k)

relax(G[j], G[m], w);

}

}

}

}

}

for (i = 0; i < G.size(); i++)

{

for (j = 0; j < G.size(); j++)

{

if (w(i, j) != NULL)

{

for (k = 0; k < G.size(); k++)

{

if ((G[k].p == j) && (G[k].d > G[i].d + w(G[i].p, G[k].p)))

return false;

}

}

}

}

return true;

}

int main()

{

int n = 5;

unsigned int i;

vector<vertex>G(n);

for (i = 0; i < G.size(); i++)

G[i].p = i;

Matrix<int, 2>w(n,n);

bool c;

w(0, 1) = -1;

w(0, 2) = 4;

w(1, 2) = 3;

w(1, 3) = 2;

w(1, 4) = 2;

w(3, 1) = 1;

w(3, 2) = 5;

w(4, 3) = -3;

c = bellman\_ford(G, w);

if (c)

cout << "There is no neg-weight cycle.";

else

cout << "A neg-weight cycle exists.";

}

And the result is below:

