**EE450 – Lab1 Report**

**XINYUE LIU**

**1332044343**

**Coding language: C++**

**Self-test: Pass all test cases successfully, including detecting negative loop.**

**1. Summary & Implementation**

**1) Map Storage**

Using an 2D n\*n array to store the graph read from the file

Implementation:

**a. Get Map Size by using function strchr() [in Main()]**

int size = 1;

if(argc > 1){

std::ifstream infile(argv[1]);

std::string line;

if(std::getline(infile, line)){ //get the first line of the file

const char \*p = line.c\_str();

p = strchr(p, ','); //p will point to the first position after the ‘,’

while( p != NULL ) {

size++;

p++;

p = strchr(p, ',');

}

}

BellmanFord(size,argv[1]); //call BellmanFord function to run the algorithm

}

**b. Store data [in** **BellmanFord function]**

After get the node number from step a, declare a corresponding n\*n array, and then store the map info line by line.

std::ifstream infile(filename);

std::string line;

while (std::getline(infile, line)) //read file line by line

{

//read line character by charater

for(std::string::iterator it = line.begin(); it != line.end(); ++it) {

//classify each character and store the useful ones

…

}

}

**c. Implement BellmanFord Algorithm**

c1. Initialize d[size] {0,imax,imax,….}

c2. Iteration from 1 to size-1

store d[] as dPrev[] from last interation

using vector<int> route to store the last nodes before the target:

for (std::vector<int>::iterator it = routeV.begin() ; it != routeV.end(); ++it){

for(m=0; m<size; m++){

if(map[\*it][m] != imax && map[\*it][m] != 0){

tmpV.push\_back(m);

}

}

}

routeV = tmpV;

tmpV.erase (tmpV.begin(),tmpV.end());

update the route (d[] & preNode[]) by comparing new route cost:

for(m=0; m<size; m++){

for (std::vector<int>::iterator it = routeV.begin() ; it != routeV.end(); ++it){

if(map[\*it][m] !=imax && d[m] > (d[\*it]+map[\*it][m])){

d[m] = d[\*it]+map[\*it][m];

preNode[m] = \*it;

}

}

}

check duplication:

if all elements in d[i] equals to dPrev[i], iteration stop

print d[]

print each route based on preNode[]

print current iteration number

end iteration

c3. check negative loop – **do one more loop** to check

Iteration i=size

using vector<int> route to store the last nodes before the target (just like c2)

update the route (d[] & preNode[]) by comparing new route cost (just like c2)

Find the first d[j] < dPrev[j] (the result from i=size-1)

Print that negative loop:

for(m=0; m<size; m++){

if(dPrev[m] > d[m]){

cout<<"Negative Loop Detected" << '\n';

tmp = preNode[m];

std::string s;

std::stringstream out;

out.str("");

out << m;

route = out.str();

while(tmp != m){

out.str("");

out << tmp;

route = out.str() + "->" + route;

tmp = preNode[tmp];

}

out.str("");

out << m;

route = out.str() + "->" + route;

cout<<route<<'\n';

break;

}

}

**2. Experimental Results**

cs-server.usc.edu(47): more N7.csv

0,\*,100,10,\*,32,\*

4,0,\*,\*,17,\*,5

5,\*,0,30,\*,42,\*

\*,23,3,0,14,\*,\*

\*,10,\*,26,0,2,\*

\*,\*,9,13,3,0,\*

\*,6,\*,\*,12,12,0

cs-server.usc.edu(48): ./BellmanFord N7.csv

0,33,13,10,24,26,38

0

0->3->1

0->3->2

0->3

0->3->4

0->3->4->5

0->3->1->6

Iteration:4

cs-server.usc.edu(51): more N10.csv

0,\*,3,\*,2,\*,\*,1,\*,\*

4,0,3,\*,\*,3,8,\*,2,\*

2,\*,0,\*,5,\*,4,8,\*,\*

5,\*,\*,0,\*,4,\*,\*,7,4

\*,3,8,\*,0,\*,\*,3,\*,\*

\*,1,\*,\*,4,0,\*,\*,\*,\*

\*,\*,5,3,\*,\*,0,\*,\*,1

\*,\*,7,\*,2,\*,\*,0,\*,\*

\*,2,\*,\*,\*,6,7,1,0,\*

\*,\*,4,\*,\*,3,1,2,\*,0

cs-server.usc.edu(52): ./BellmanFord N10.csv

0,5,3,10,2,8,7,1,7,8

0

0->4->1

0->2

0->2->6->3

0->4

0->4->1->5

0->2->6

0->7

0->4->1->8

0->2->6->9

Iteration:4

cs-server.usc.edu(54): more N20.csv

0,6,9,\*,6,4,\*,\*,9,7,\*,\*,\*,\*,\*,3,\*,\*,4,\*

\*,0,2,9,5,\*,\*,\*,9,3,9,\*,\*,\*,\*,\*,2,\*,8,2

\*,\*,0,2,\*,\*,\*,\*,8,\*,\*,\*,\*,5,1,\*,\*,7,2,5

\*,6,\*,0,\*,4,\*,6,\*,6,8,4,\*,\*,\*,\*,\*,9,\*,2

4,\*,\*,4,0,7,5,7,\*,1,\*,4,\*,2,8,6,\*,2,7,9

8,\*,\*,6,\*,0,\*,3,\*,\*,8,9,\*,\*,1,5,\*,4,4,1

\*,\*,\*,9,\*,4,0,\*,\*,5,\*,\*,\*,\*,\*,\*,\*,\*,5,\*

\*,7,1,\*,1,\*,3,0,\*,9,\*,\*,\*,\*,4,\*,\*,7,\*,4

4,\*,2,\*,\*,5,\*,\*,0,\*,8,\*,1,\*,3,9,4,\*,\*,2

\*,3,7,\*,8,\*,\*,\*,3,0,5,7,\*,\*,5,5,8,3,\*,7

\*,\*,\*,\*,\*,6,\*,\*,\*,\*,0,\*,\*,\*,3,\*,3,\*,\*,\*

8,\*,\*,\*,1,\*,\*,\*,2,8,\*,0,\*,\*,4,2,1,\*,3,3

4,\*,9,\*,\*,4,\*,\*,\*,\*,7,3,0,\*,1,3,\*,\*,\*,\*

9,6,\*,\*,9,6,8,\*,\*,\*,\*,3,\*,0,\*,3,7,\*,\*,7

\*,\*,\*,5,\*,2,\*,\*,\*,9,9,\*,\*,1,0,4,6,6,\*,6

3,1,\*,\*,\*,\*,9,\*,\*,1,\*,\*,5,3,\*,0,7,9,4,2

\*,9,\*,6,7,\*,\*,5,8,7,\*,4,\*,\*,\*,\*,0,\*,\*,6

5,7,7,\*,\*,\*,\*,3,\*,9,4,\*,\*,\*,\*,2,\*,0,6,1

8,4,3,\*,\*,9,6,3,\*,6,\*,3,\*,\*,\*,\*,1,\*,0,\*

\*,4,\*,2,1,\*,\*,2,\*,4,\*,2,\*,5,9,\*,\*,3,5,0

cs-server.usc.edu(55): ./BellmanFord N20.csv

0,4,6,7,6,4,10,7,7,4,9,7,8,6,5,3,5,7,4,5

0

0->15->1

0->15->1->2

0->5->19->3

0->4

0->5

0->18->6

0->5->7

0->15->9->8

0->15->9

0->15->9->10

0->18->11

0->15->12

0->15->13

0->5->14

0->15

0->18->16

0->15->9->17

0->18

0->5->19

Iteration:4

**Negative loop test:**

cs-server.usc.edu(56): more neg.txt

0,-1,\*

\*,0,-1

-1,\*,0

cs-server.usc.edu(57): ./BellmanFord neg.txt

Negative Loop Detected

0->1->2->0