CMP_SC 4050 FS2024

ASSIGNMENT 3: IMPLEMENT BELLMAN-FORD-MOORE ALGORITHM

DESCRIPTION

You are to implement the Bellman-Ford-Moore dynamic programming algorithm to find the shortest path from all nodes to a terminal node in a directed graph using the provided library to represent the graph. The program will accept input via stdin. Note that you can redirect stdin to come from a file via the command-line and the starter project includes an example file that can be used as input for testing. The input format should be as follows:

```
<1 or 0 to indicate directed or not>
<# of nodes>
<# of edges>
<node 1 of edge1><node 2 of edge 1><weight of edge 1>
<node 1 of edge1><node 2 of edge 2><weight of edge 2>
......
<node 1 of edge n><node 2 of edge n><weight of edge n>
```

For example, the following input describes a directed graph with 3 nodes (indexed 1, 2 and 3) and 2 edges ({1,2} with weight 0.5 and {2,3} with weight 2.2):

```
1
3
2
1 2 0.5
2 3 2.2
```

You are given a main() program, a Makefile, a library, and some starter code. All you need to do is complete the actual code for the functions in Bellman-Ford.cc. Note that you may implement things in either C or C++, and a skeletal test program is included for each.

HOW TO GET STARTED

To get started you should:

```
git clone https://github.com/JimRies1966/cs4050fs2024a3.git
```

You are welcome to work on your code on any platform you like, but the provided library will work on Hellbender (and probably on other Linux on Intel platforms). The provided library will likely not work on MacOS or Windows (except using the Windows Subsystem for Linux) and definitely not on non-intel platforms.

You should be aware that submissions will **only** be evaluated by the TAs on hellbender.rnet.missouri.edu. If, for example, something works on your machine but doesn't compile on hellbender.rnet.missouri.edu, you will get a zero.

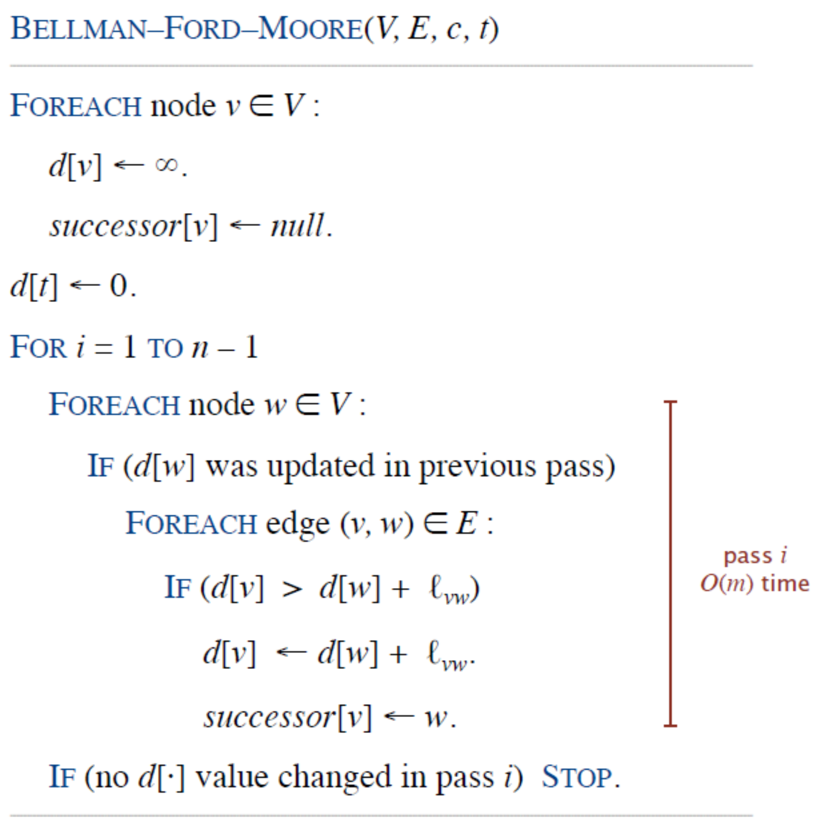
Once you have the starter code in a directory, just type "make". This will build the code and leave you with two executable files called "test" and "testcc". The files are written in C and C++ respectively, and they do the same thing (so you can use either or both). You can run this file with the included sample input files (dijkstraBuster.G, KT.G, and some others) this way:

```
./test <KT.G
Or
./testcc <KT.G
```

BELLMAN-FORD-MOORE FROM SLIDES

```
SHORTEST-PATHS($V, E, \ell, t$)

FOREACH node $v \in V$ :
    $M[0, v] \leftarrow \infty$.
$M[0, t] \leftarrow 0$.
FOR i = 1 TO $n - 1$
    FOREACH node $v \in V$ :
        $M[i, v] \leftarrow M[i-1, v]$.
        FOREACH edge $(v, w) \in E$ :
            $M[i, v] \leftarrow \min \{ M[i, v],\ M[i-1, w] + \ell_{vw} \}$.
```

BELLMAN-FORD-MOORE (EFFICIENT) FROM SLIDES



```
BELLMAN–FORD–MOORE($V, E, c, t$)

FOREACH node $v \in V$ :
    $d[v] \leftarrow \infty$.
    $successor[v] \leftarrow null$.
$d[t] \leftarrow 0$.
FOR $i = 1$ TO $n - 1$
    FOREACH node $w \in V$ :
        IF ($d[w]$ was updated in previous pass)
            FOREACH edge $(v, w) \in E$ :
                IF ($d[v] > d[w] + \ell_{vw}$)
                    $d[v] \leftarrow d[w] + \ell_{vw}$.
                    $successor[v] \leftarrow w$.
    IF (no $d[\cdot]$ value changed in pass $i$) STOP.
```

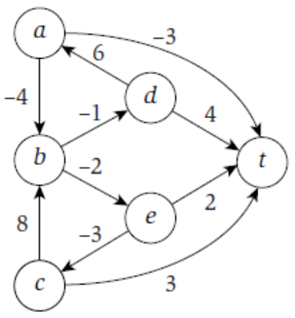
pass i
$O(m)$ time

Note that you can use the "efficient" algorithm or the "inefficient" algorithm as you choose. I implemented mine using the "efficient" algorithm.

SAMPLE OUTPUT#1

```
jimr@jimrsurfacepro9:~/CS4050/FS2024/assignments/A3$ ./testcc <KT.G
Enter 1 for directed or 0 for undirected: Enter the number of vertices: Enter the
number of edges:
Vertex 1:
        1->2 (-4.00)
        1->6 (-3.00)

Vertex 2:
        2->4 (-1.00)
        2->5 (-2.00)

Vertex 3:
        3->2 (8.00)
        3->6 (3.00)

Vertex 4:
        4->1 (6.00)
        4->6 (4.00)

Vertex 5:
        5->3 (-3.00)
        5->6 (2.00)

Vertex 6:


successor (1)=2 distance to t=-6
successor (2)=5 distance to t=-2
successor (3)=6 distance to t=3
successor (4)=1 distance to t=0
successor (5)=3 distance to t=0
successor (6)=NULL distance to t=0
```

DISCUSSION OF THE ABOVE EXAMPLE FROM THE BOOK



(a)

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

(b)

Figure 6.23 For the directed graph in (a), the Shortest-Path Algorithm constructs the dynamic programming table in (b).

SAMPLE OUTPUT #2

```
jimr@jimrsurfacepro9:~/CS4050/FS2024/assignments/A3$ ./testcc <dijkstraBuster.G
Enter 1 for directed or 0 for undirected: Enter the number of vertices: Enter the
number of edges:
Vertex 1:
        1->2 (6.00)
        1->3 (4.00)
        1->4 (2.00)

Vertex 2:
        2->3 (-8.00)

Vertex 3:
        3->4 (3.00)

Vertex 4:


successor (1)=2 distance to t=1
successor (2)=3 distance to t=-5
successor (3)=4 distance to t=3
successor (4)=NULL distance to t=0
```

SAMPLE OUTPUT #3

```
jimr@jimrsurfacepro9:~/CS4050/FS2024/assignments/A3$ ./testcc <ex3.G
Enter 1 for directed or 0 for undirected: Enter the number of vertices: Enter the
number of edges:
Vertex 1:
        1->2 (9.00)
        1->3 (5.00)
        1->4 (8.00)

Vertex 2:
        2->4 (5.00)
        2->5 (-3.00)
        2->8 (10.00)

Vertex 3:
        3->4 (4.00)
        3->6 (-2.00)
        3->7 (12.00)

Vertex 4:
        4->7 (7.00)
        4->5 (-1.00)

Vertex 5:
        5->7 (-6.00)
        5->8 (13.00)

Vertex 6:
        6->8 (17.00)

Vertex 7:
        7->6 (-5.00)
        7->8 (11.00)

Vertex 8:


successor (1)=2 distance to t=11
successor (2)=5 distance to t=2
successor (3)=4 distance to t=8
successor (4)=5 distance to t=4
successor (5)=7 distance to t=5
successor (6)=8 distance to t=17
successor (7)=8 distance to t=11
successor (8)=NULL distance to t=0
```