Bellman Test 1: directed bellman (P2)

Bellman Test 2: if remove one edge, **directed** bellman show negative loop (P3)

Bellman Test 3: if remove one edge, **undirected** bellman (P4)

Dijkstra Test 1: undirected Dijkstra (P5)

Dijkstra Test 2: directed Dijkstra (P6)

Astar Test 1: all 1-weight nodes with Astar (P7)

Astar Test 2: change some weights so Astar path change (P8)

* All the test commands are saved in sample_usage.txt and can be copied to cmd directely.

<u>To run the program, open the folder of routing.exe, hold shift+click mouse right button in the folder, select “open cmd window here” or “open powershell window here”,**</u> and paste the commands from sample_usage.txt

```
Bellman Ford test 1
in this test, have direction, search every node to node f, use
routing_table_01.txt as input.
.\routing.exe --with_direction --destination_node=f --
data_path=./routing_table_01.txt bellmanford

output:
Loading data from path ./routing_table_01.txt ...
totally 6 nodes detected from dataset, they have direction and no
weight:
a b c d e f
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
a       c       2.0     1.0     1.0
a       b       3.0     1.0     1.0
a       d       5.0     1.0     1.0
b       d       1.0     1.0     1.0
b       e       4.0     1.0     1.0
c       f       1.0     1.0     1.0
d       e       3.0     1.0     1.0
d       c       2.0     1.0     1.0
e       f       2.0     1.0     1.0
------
Selected bellmanford algorithm. This algorithm receives directed, no
weight graph and calculates the minimum distance from each node to
destination node.
You selected destination node: f
---------------------------------------------
Testing if there exists negtive weighted loop
>>> No Negtive weighted loop!
---------------------------------------------
Distance to vertice f
Node    Next    Distance
a       c       3.0
b       d       4.0
c       f       1.0
d       c       3.0
e       f       2.0

*Same as PPT P58 result
```

```
BellmanFord test 2
If remove edge between c and f (see routing_table_02.txt for detail)
First test with direction, to the node f, the result should be one
negative loop:
.\routing.exe --with_direction --destination_node=f --
data_path=./routing_table_02.txt bellmanford

output:
Loading data from path ./routing_table_02.txt ...
totally 6 nodes detected from dataset, they have direction and no
weight:
a b c d e f
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
a       c       2.0     1.0     1.0
a       b       3.0     1.0     1.0
a       d       5.0     1.0     1.0
b       d       1.0     1.0     1.0
b       e       4.0     1.0     1.0
d       e       3.0     1.0     1.0
d       c       2.0     1.0     1.0
e       f       2.0     1.0     1.0
------
Selected bellmanford algorithm. This algorithm receives directed, no
weight graph and calculates the minimum distance from each node to
destination node.
You selected destination node: f
---------------------------------------------
Testing if there exists negtive weighted loop
Negtive weighted loop tested!


*Same as PPT P59 result
 Because its directed edges so negative loop exists
```

```
Bellman Ford test 3
If remove edge between c and f (see routing_table_02.txt for detail)
Compare with test 2, if we set it as no direction, node c still can go
to node f, thus there should be no negative loop:
.\routing.exe --destination_node=f --
data_path=./routing_table_02.txt bellmanford

output:
Loading data from path ./routing_table_02.txt ...
totally 6 nodes detected from dataset, they have no direction and no
weight:
a b c d e f
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
a       c       2.0     1.0     1.0
a       b       3.0     1.0     1.0
a       d       5.0     1.0     1.0
b       d       1.0     1.0     1.0
b       e       4.0     1.0     1.0
c       d       2.0     1.0     1.0
d       e       3.0     1.0     1.0
e       f       2.0     1.0     1.0
------
Selected bellmanford algorithm. This algorithm receives directed, no
weight graph and calculates the minimum distance from each node to
destination node.
You selected destination node: f
---------------------------------------------
Testing if there exists negtive weighted loop
>>> No Negtive weighted loop!
---------------------------------------------
Distance to vertice f
Node    Next    Distance
a       b       9.0
b       e       6.0
c       d       7.0
d       e       5.0
e       f       2.0


*Same as PPT P59 result
```

```
Dijkstra test 1
If all nodes available and un-directed, from node a to all other nodes:
.\routing.exe --start_node=a --
data_path=./routing_table_01.txt Dijkstra

output:
Loading data from path ./routing_table_01.txt ...
totally 6 nodes detected from dataset, they have no direction and no
weight:
a b c d e f
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
a       c       2.0     1.0     1.0
a       b       3.0     1.0     1.0
a       d       5.0     1.0     1.0
b       d       1.0     1.0     1.0
b       e       4.0     1.0     1.0
c       d       2.0     1.0     1.0
c       f       1.0     1.0     1.0
d       e       3.0     1.0     1.0
e       f       2.0     1.0     1.0
------
Selected dijkstra algorithm. This algorithm receives directed, no
weight graph and calculates the minimum distance from start node to
every other notde.
You selected start node: a
---------------------------------------------
Distance from vertice a
Node    From    Distance
b       a       3.0
c       a       2.0
d       c       4.0
e       f       5.0
f       c       3.0

*same as PPT P66 result
```

```
Dijkstra test 2
In this test all edges have direction. Compare with test 1: the node e
have longer path cause direction.
.\routing.exe --with_direction --start_node=a --
data_path=./routing_table_01.txt dijkstra

output:
Loading data from path ./routing_table_01.txt ...
totally 6 nodes detected from dataset, they have direction and no
weight:
a b c d e f
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
a       c       2.0     1.0     1.0
a       b       3.0     1.0     1.0
a       d       5.0     1.0     1.0
b       d       1.0     1.0     1.0
b       e       4.0     1.0     1.0
c       f       1.0     1.0     1.0
d       e       3.0     1.0     1.0
d       c       2.0     1.0     1.0
e       f       2.0     1.0     1.0
------
Selected dijkstra algorithm. This algorithm receives directed, no
weight graph and calculates the minimum distance from start node to
every other notde.
You selected start node: a
---------------------------------------------
Distance from vertice a
Node    From    Distance
b       a       3.0
c       a       2.0
d       b       4.0
e       b       7.0
f       c       3.0
```

```
Astar test 1
Standard Astar from 1 to 6
In this case all the node have weight 1, so the shortest path is same
as dijkstra
.\routing.exe --with_direction --with_weight --start_node=1 --
destination_node=6 --data_path=./routing_table_03.txt astar

output:
Loading data from path ./routing_table_03.txt ...
totally 6 nodes detected from dataset, they have direction and nodes
weight:
1 2 3 4 5 6
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
1       3       2.0     1.0     1.0
1       2       3.0     1.0     1.0
1       4       5.0     1.0     1.0
2       4       1.0     1.0     1.0
2       5       4.0     1.0     1.0
3       6       1.0     1.0     1.0
4       3       2.0     1.0     1.0
4       5       3.0     1.0     1.0
5       6       2.0     1.0     1.0
------
Selected astar algorithm. This algorithm receives directed, weighted
graph and calculates the minimum distance from start node to
destination node.
You selected start node: 1 and destination node: 6
---------------------------------------------
The shortest path with weighted nodes is:
['1', '3', '6']


*1 go to 6 via 3
```

```
Astar test 2
If decrease weight of node 5 and increase weight of node 6 (see routing
_table_04.txt for detail)
Compare with Astar test 1, it should change the path cause now if go to
6 via 3 will cost more.
.\routing.exe --with_direction --with_weight --start_node=1 --
destination_node=6 --data_path=./routing_table_04.txt astar

output:
Loading data from path ./routing_table_04.txt ...
totally 6 nodes detected from dataset, they have direction and nodes
weight:
1 2 3 4 5 6
The edges between nodes and nodes' weight:
Node_a  Node_b  Dist    Weighta Weightb
1       3       2.0     5.0     10.0
1       2       3.0     5.0     3.0
1       4       5.0     5.0     2.0
2       4       1.0     3.0     2.0
2       5       4.0     3.0     1.0
3       6       1.0     10.0    1.0
4       3       2.0     2.0     10.0
4       5       3.0     2.0     1.0
5       6       2.0     1.0     1.0
------
Selected astar algorithm. This algorithm receives directed, weighted
graph and calculates the minimum distance from start node to
destination node.
You selected start node: 1 and destination node: 6
---------------------------------------------
The shortest path with weighted nodes is:
['1', '2', '5', '6']

*As expected, 1 to 6 change to via 2 and 5 rather than 3.
```