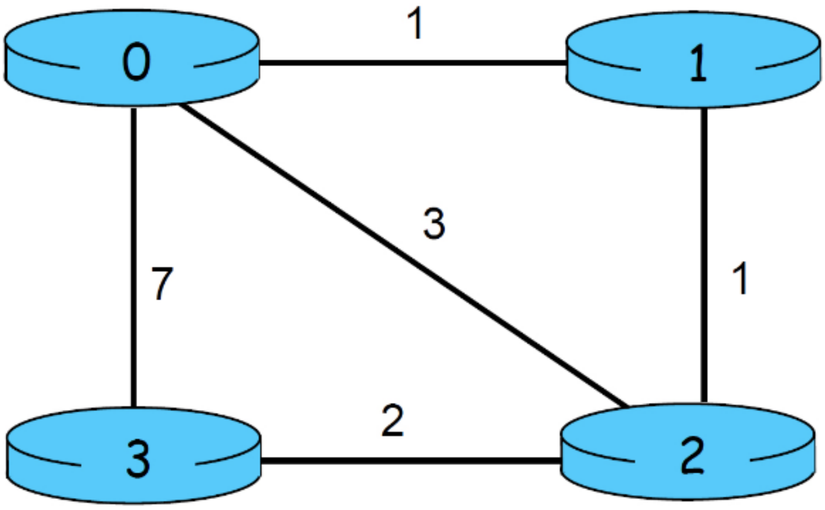
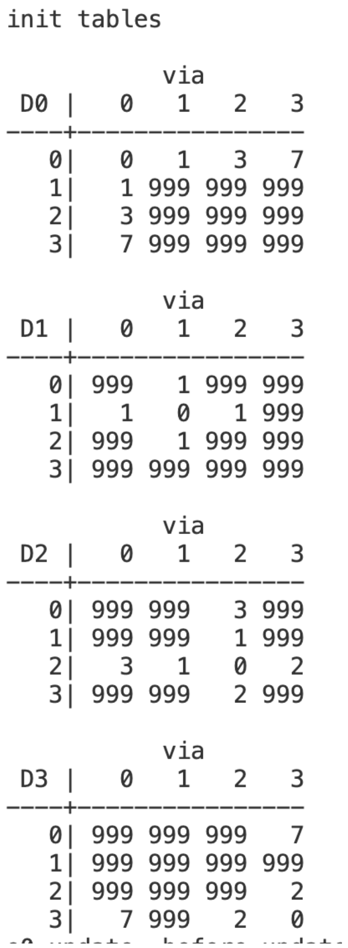
Implementation Details & Analysis
Part 1: Routing Algorithm

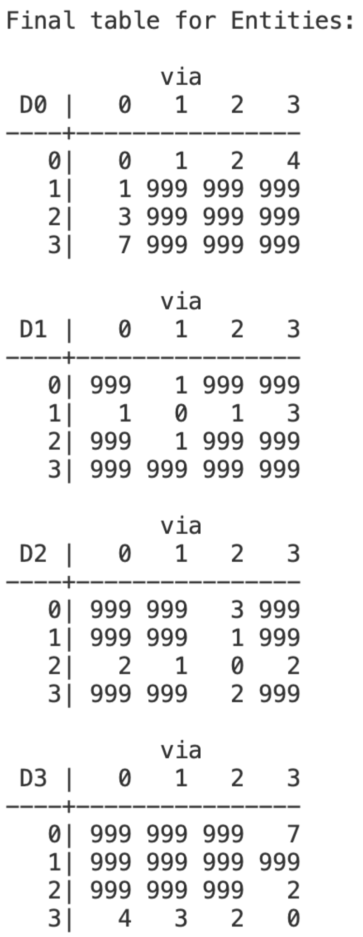


Distance vector in java

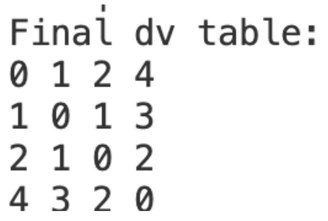
1. Starter code: https://gaia.cs.umass.edu/kurose_ross/programming/DV/
2. Implementation: init(), update() for each entity/node
    - constructor: initiate state of dv
    - update(): update dv upon receiving neighbor’s dv
3. Run Result



```
init tables

             via
 D0 |    0     1    2    3
----+-----------------
   0|    0     1    3    7
   1|    1   999  999  999
   2|    3   999  999  999
   3|    7   999  999  999

             via
 D1 |    0     1    2    3
----+-----------------
   0|  999     1  999  999
   1|    1     0    1  999
   2|  999     1  999  999
   3|  999   999  999  999

             via
 D2 |    0     1    2    3
----+-----------------
   0|  999   999    3  999
   1|  999   999    1  999
   2|    3     1    0    2
   3|  999   999    2  999

             via
 D3 |    0     1    2    3
----+-----------------
   0|  999   999  999    7
   1|  999   999  999  999
   2|  999   999  999    2
   3|    7   999    2    0
```



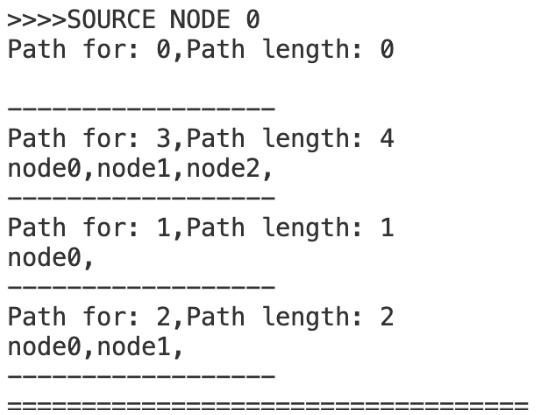
```
Final table for Entities:

             via
 D0 |    0     1    2    3
----+-----------------
   0|    0     1    2    4
   1|    1   999  999  999
   2|    3   999  999  999
   3|    7   999  999  999

             via
 D1 |    0     1    2    3
----+-----------------
   0|  999     1  999  999
   1|    1     0    1    3
   2|  999     1  999  999
   3|  999   999  999  999

             via
 D2 |    0     1    2    3
----+-----------------
   0|  999   999    3  999
   1|  999   999    1  999
   2|    2     1    0    2
   3|  999   999    2  999

             via
 D3 |    0     1    2    3
----+-----------------
   0|  999   999  999    7
   1|  999   999  999  999
   2|  999   999  999    2
   3|    4     3    2    0
```



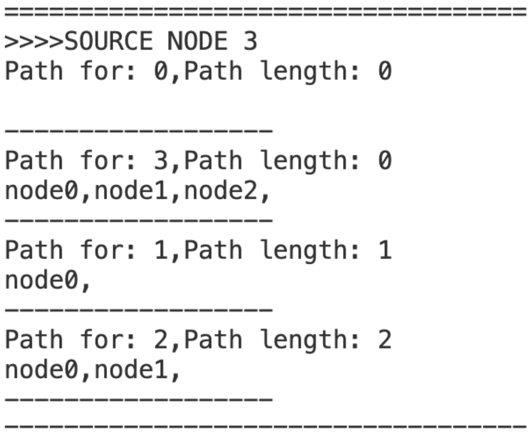
```
Final dv table:
0 1 2 4
1 0 1 3
2 1 0 2
4 3 2 0
```

Verify results with link state algorithm

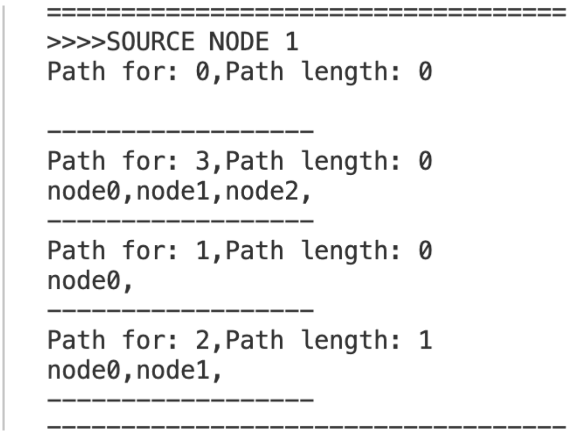
- code reference https://www.baeldung.com/java-dijkstra
- entry point: Dijkstra.java
- run result



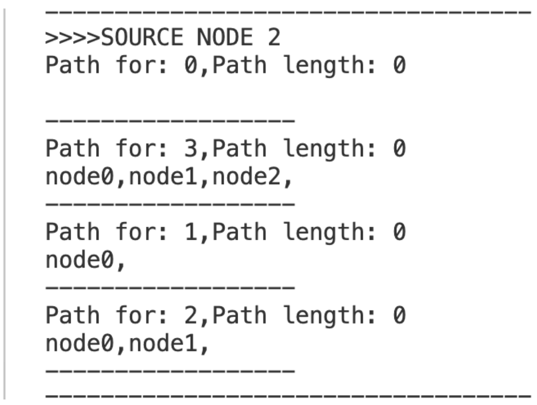
```
>>>>SOURCE NODE 0
Path for: 0,Path length: 0

------------------
Path for: 3,Path length: 4
node0,node1,node2,
------------------
Path for: 1,Path length: 1
node0,
------------------
Path for: 2,Path length: 2
node0,node1,
------------------
===================================
```



```
===================================
>>>>SOURCE NODE 3
Path for: 0,Path length: 0

------------------
Path for: 3,Path length: 0
node0,node1,node2,
------------------
Path for: 1,Path length: 1
node0,
------------------
Path for: 2,Path length: 2
node0,node1,
------------------
-----------------------------------
```



```
===================================
>>>>SOURCE NODE 1
Path for: 0,Path length: 0

------------------
Path for: 3,Path length: 0
node0,node1,node2,
------------------
Path for: 1,Path length: 0
node0,
------------------
Path for: 2,Path length: 1
node0,node1,
------------------
-----------------------------------
```



```
-----------------------------------
>>>>SOURCE NODE 2
Path for: 0,Path length: 0

------------------
Path for: 3,Path length: 0
node0,node1,node2,
------------------
Path for: 1,Path length: 0
node0,
------------------
Path for: 2,Path length: 0
node0,node1,
------------------
-----------------------------------
```

4. reproduce result
    - checkout code @ https://github.com/ZyeG/dv-dijk

Discussion

Link state (LS) is centralized (link costs are known to all nodes at all time), while distance vector (DV) algorithm is distributed (each node notifies neighbor only when its dv changes). Convergence time for DV varies; when a link cost goes up, it may lead to count-to-infinity issues (i.e. routing loop).