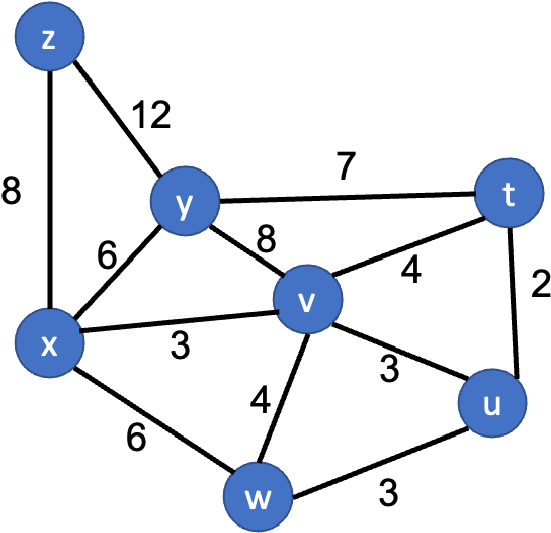
**Homework 3**

CSE 310 Fall 2021

Due date: **November 9, 2021; 11:59 PM** Submission via Blackboard.

**1. Dijkstra’s algorithm**

1. Consider the following network. With the indicated link costs, use Dijkstra’s shortest-path algorithm to compute the shortest path from *x* to all network nodes. Show how the algorithm works by computing a table similar to the ones discussed in class. **(15 points)**



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | N’ | D(t),p(t) | D(u),p(u) | D(v),p(v) | D(w),p(w) | D(y),p(y) | D(z),p(z) |
| 0 | x | ∞ | ∞ | **3,x** | 6,x | 6,x | 8,x |
| 1 | xv | 7,v | **6,v** | 3,x | 6,x | 6,x | 8,x |
| 2 | xvu | 7,v | 6,v | 3,x | **6,x** | 6,x | 8,x |
| 3 | xvuw | 7,v | 6,v | 3,x | 6,x | **6,x** | 8,x |
| 4 | xvuwy | **7,v** | 6,v | 3,x | 6,x | 6,x | 8,x |
| 5 | xvuwyt | 7,v | 6,v | 3,x | 6,x | 6,x | **8,x** |
| 6 | xvuwytz | 7,v | 6,v | 3,x | 6,x | 6,x | 8,x |

x to t: xvt,7(path,distance); x to u: xvu,6; x to v: xv, 3; x to w: xw, 6; x to y: xy, 6; x to z: xz, 8

1. For the same network as part A, using Dijkstra’s algorithm, and showing your work similar to part A, compute the shortest path from *t* to all network nodes. **(10 points)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | N’ | D(z),p(z) | D(x),p(x) | D(w),p(w) | D(u),p(u) | D(v),p(v) | D(y),p(y) |
| 0 | t | ∞ | ∞ | ∞ | **2,t** | 4,t | 7,t |
| 1 | tu | ∞ | ∞ | 5,u | 2,t | **4,t** | 7,t |
| 2 | tuv | ∞ | 7,v | **5,u** | 2,t | 4,t | 7,t |
| 3 | tuvw | ∞ | **7,v** | 5,u | 2,t | 4,t | 7,t |
| 4 | tuvwx | 15,x | 7,v | 5,u | 2,t | 4,t | **7,**t |
| 5 | tuvwxy | **15,x** | 7,v | 5,u | 2,t | 4,t | 7,t |
| 6 | tuvwxyz | 15,x | 7,v | 5,u | 2,t | 4,t | 7,t |

t to z: tvxz,15;

t to x: tvx, 7;

t to w: tuw,5;

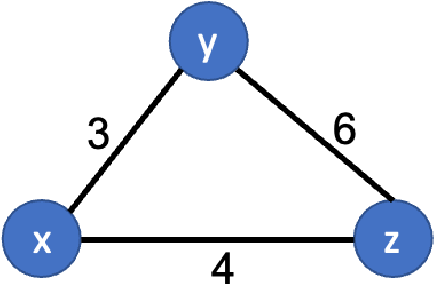
t to u: tu, 2;

t to v: tv,4;

t to y: ty,7

# Distance vector

1. Consider the three-node topology shown below. Compute the distance tables after the initialization step and after each iteration of a synchronous version of the distance-vector algorithm, similar to our discussion in class. **(10 points)**



to

Node x:

After initialization: After first iteration: After second iteration:

from

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | ∞ | ∞ | ∞ |
| z | ∞ | ∞ | ∞ |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

Node y:

After initialization: After first iteration: After second iteration:

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | ∞ | ∞ | ∞ |
| y | 4 | 0 | 6 |
| z | ∞ | ∞ | ∞ |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

Node z:

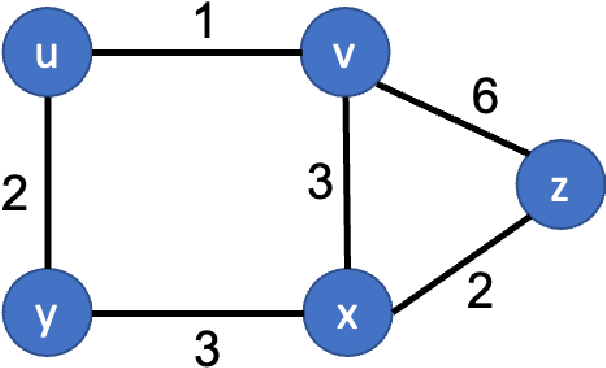
After initialization: After first iteration: After second iteration:

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 4 | 6 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | x | y | z |
| x | 0 | 3 | 4 |
| y | 3 | 0 | 6 |
| z | 4 | 6 | 0 |

1. Consider the network shown below and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node *z* for each iteration. **(15 points)**



to

from

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | u | v | x | y | z |
| v | ∞ | ∞ | ∞ | ∞ | ∞ |
| x | ∞ | ∞ | ∞ | ∞ | ∞ |
| z | ∞ | 6 | 2 | ∞ | 0 |
|  | | | | | |
|  | u | v | x | y | z |
| v | 1 | 0 | 3 | ∞ | 6 |
| x | ∞ | 3 | 0 | 3 | 2 |
| z | 7 | 5 | 2 | 5 | 0 |
|  | | | | | |
|  | u | v | x | y | z |
| v | 1 | 0 | 3 | 3 | 5 |
| x | 4 | 3 | 0 | 3 | 2 |
| z | 6 | 5 | 2 | 5 | 0 |
|  | | | | | |
|  | u | v | x | y | z |
| v | **1** | **0** | **3** | **3** | **5** |
| x | **4** | **3** | **0** | **3** | **2** |
| z | **6** | **5** | **2** | **5** | **0** |