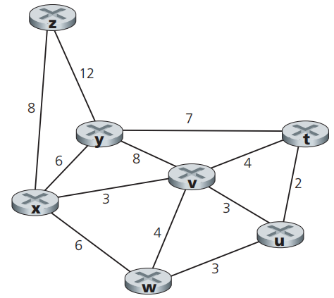
P3. Consider the following network. With the indicated link costs, use Dijkstra’s shortest-path algorithm to compute the shortest path from *x* to all network nodes.

Show how the algorithm works by computing a table similar to Table 5.1.



Answer:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

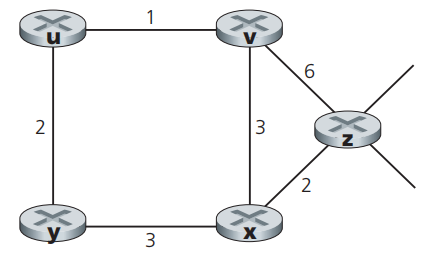
N ′= nodes travelled

D(t) = distance to the given vertex

p(t) = second-last vertex visited in that path

|  |  |  |
| --- | --- | --- |
| **From x to** | **The shortest path** | **Cost** |
| T | x,v,t | 7 |
| U | x,v,u | 6 |
| V | x,v | 3 |
| W | x,w | 6 |
| Y | x,v | 6 |
| z | x,z | 8 |

P5. 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*.



Answer:

The node z will only know the cost to each of its neighbors (v and x) initially:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| From | Cost to | | | | | |
|  | U | V | X | Y | Z |
| V | ∞ | ∞ | ∞ | ∞ | ∞ |
| X | ∞ | ∞ | ∞ | ∞ | ∞ |
| Z | ∞ | 6 | 2 | ∞ | 0 |

We can now check for other neighbors using x and v, as well as look for cheaper costs for existing routes. Through these nodes, we can also find routes to u and y. Red-colored values are for new routes found, and blue-colored values are for cheaper costs of existing routes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| From | Cost to | | | | | |
|  | U | V | X | Y | Z |
| V | 1 | 0 | 3 | ∞ | 6 |
| X | ∞ | 3 | 0 | 3 | 2 |
| Z | 7 | 5 | 2 | 5 | 0 |

Now that we have all of the nodes, we can find the all-optimal routes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| From | Cost to | | | | | |
|  | U | V | X | Y | Z |
| V | 1 | 0 | 3 | 3 | 5 |
| X | 4 | 3 | 0 | 3 | 2 |
| Z | 6 | 5 | 2 | 5 | 0 |

P9. Consider the count-to-infinity problem in the distance vector routing. Will the count-to-infinity problem occur if we decrease the cost of a link? Why?

How about if we connect two nodes which do not have a link?

Answer:

No, because the decreasing link cost won't cause a loop (caused by the next-hop relation of between two nodes of that link). Connecting two nodes with a link is equivalent to decreasing the link weight from infinite to the finite weight.