GREEDY ALGORITHMS 2

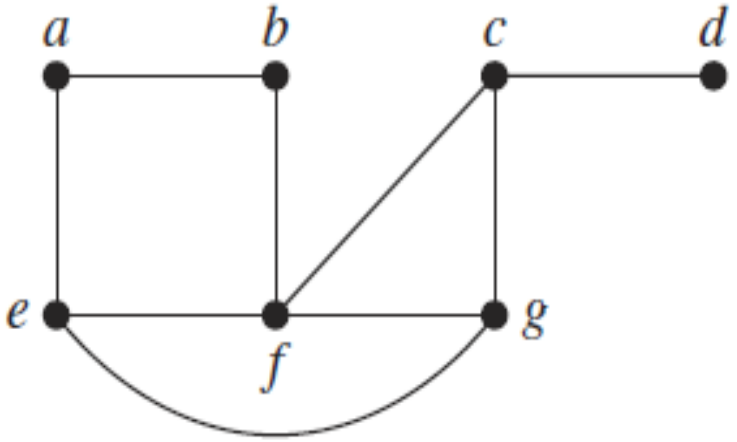
Spanning Tree

- Let *G* be a simple graph. A *spanning tree* of *G* is a subgraph of *G* that is a tree containing every vertex of *G*.

- A simple graph with a spanning tree must be connected, because there is a path in the spanning tree between any two vertices. The converse is also true; that is, every connected simple graph has a spanning tree. We will give an example before proving this result.

Spanning Tree

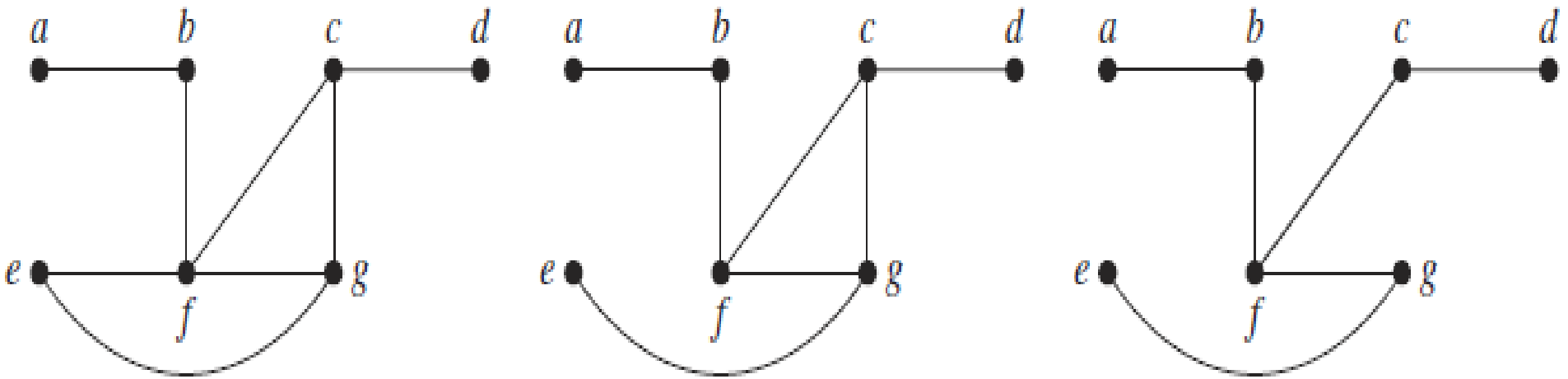
- Find a spanning tree of the figure.



Spanning Tree

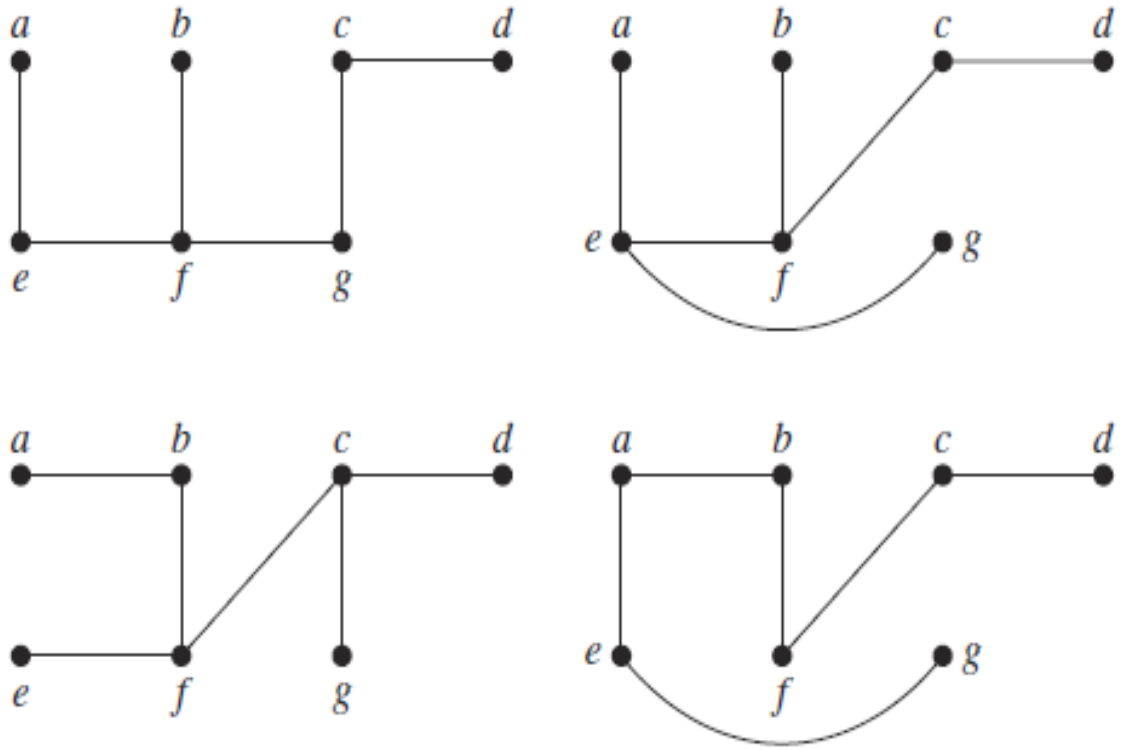
- The graph G is connected, but it is not a tree because it contains simple circuits.
- Remove the edge {a, e}.
- This eliminates one simple circuit, and the resulting subgraph is still connected and still contains every vertex of G. Next remove the edge {e, f } to eliminate a second simple circuit.
- Finally, remove edge {c, g} to produce a simple graph with no simple circuits.
- This subgraph is a spanning tree, because it is a tree that contains every vertex of G. The sequence of edge removals used to produce the spanning tree is illustrated in the figure.

Spanning Tree



Edge removed: $\{a, e\}$ $\{e, f\}$ $\{c, g\}$

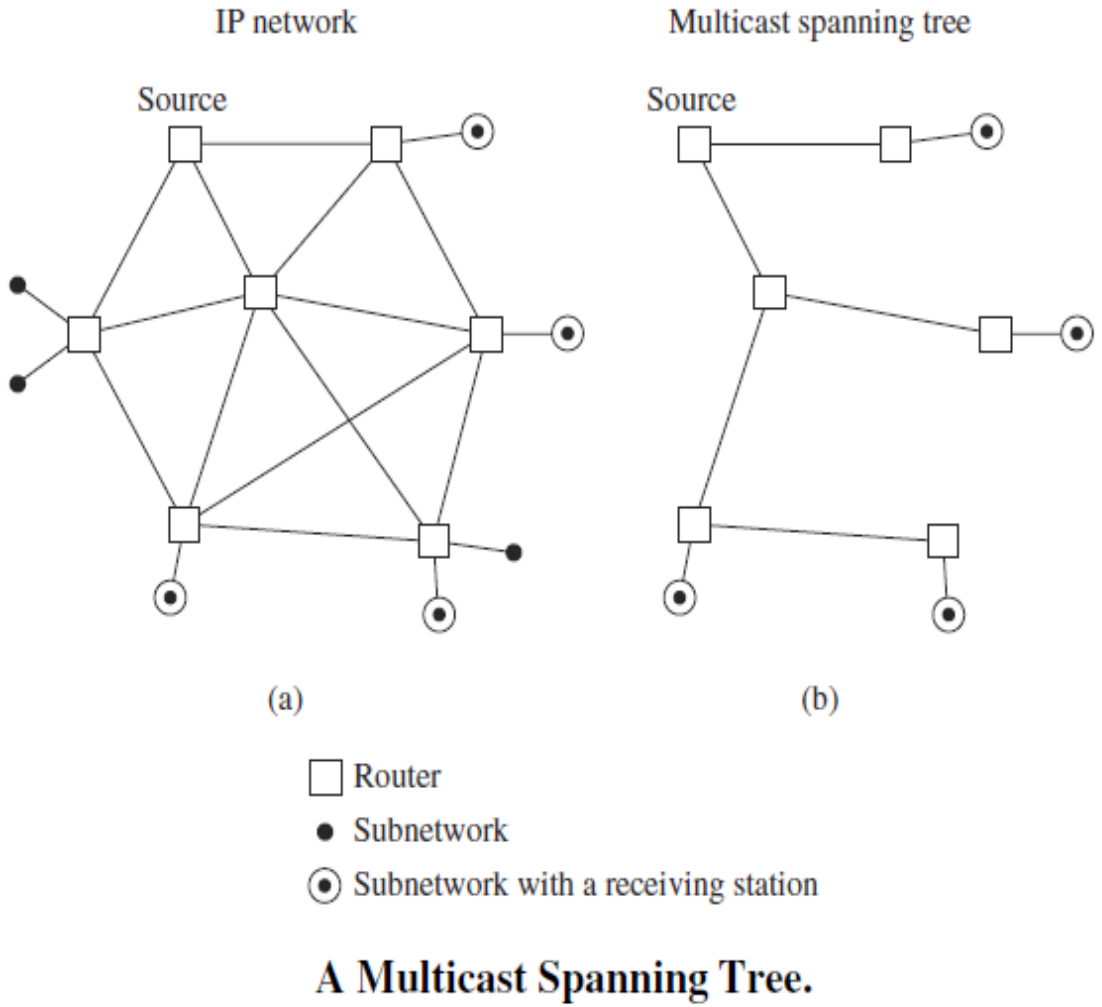
Spanning Tree



Spanning Tree

- For data to reach receiving computers as quickly as possible, there should be no loops (which in graph theory terminology are circuits or cycles) in the path that data take through the network.
- That is, once data have reached a particular router, data should never return to this router.
- To avoid loops, the multicast routers use network algorithms to construct a spanning tree in the graph that has the multicast source, the routers, and the subnetworks containing receiving computers as vertices, with edges representing the links between computers and/or routers.
- The root of this spanning tree is the multicast source. The subnetworks containing receiving computers are leaves of the tree. (Note that subnetworks not containing receiving stations are not included in the graph.) This is illustrated in figure.

Spanning Tree



A Multicast Spanning Tree.

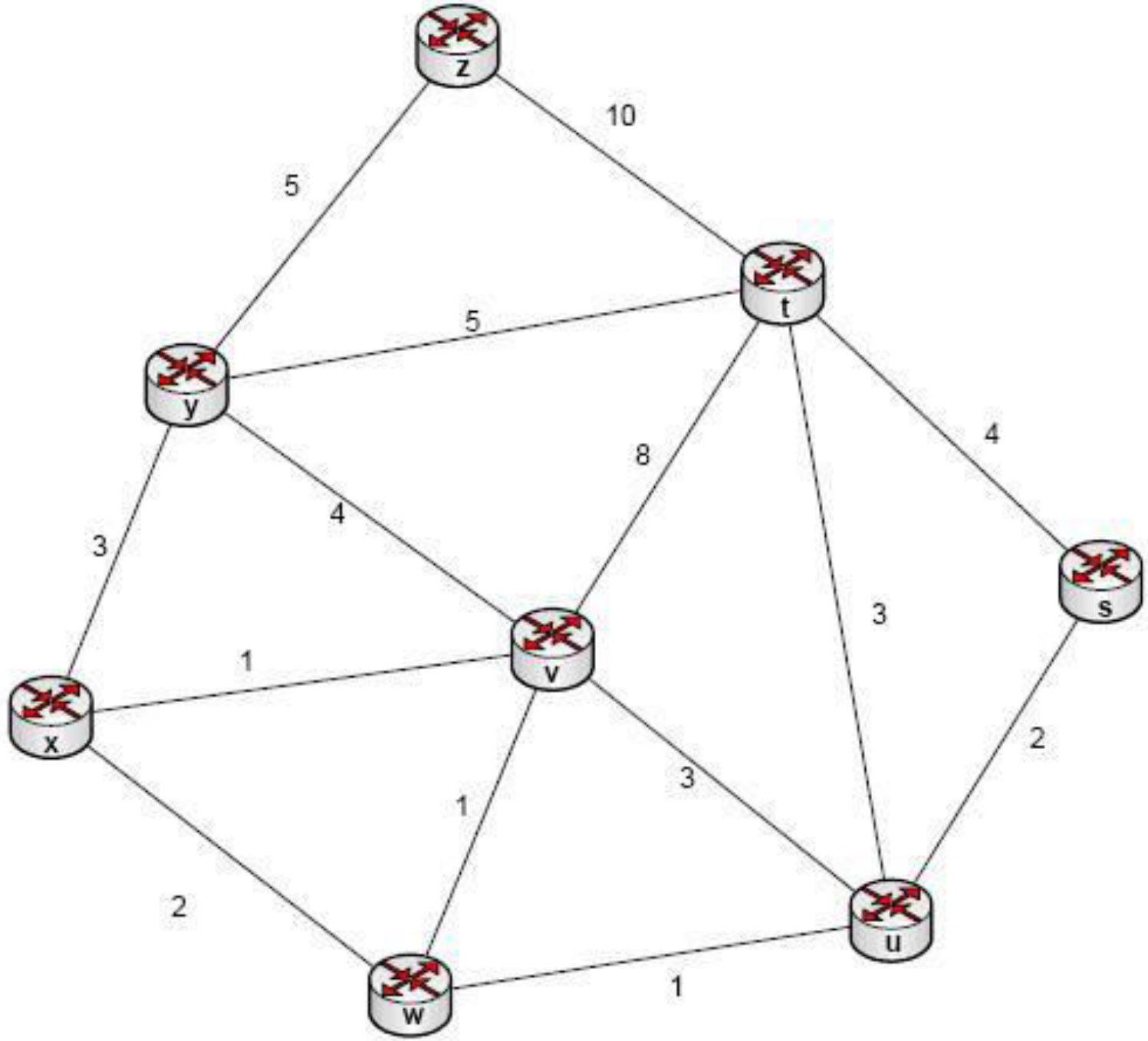
Prim Algorithm

- Prim Algorithm is a greedy algorithm who finds a spanning tree from a tree.

Prim Algorithm

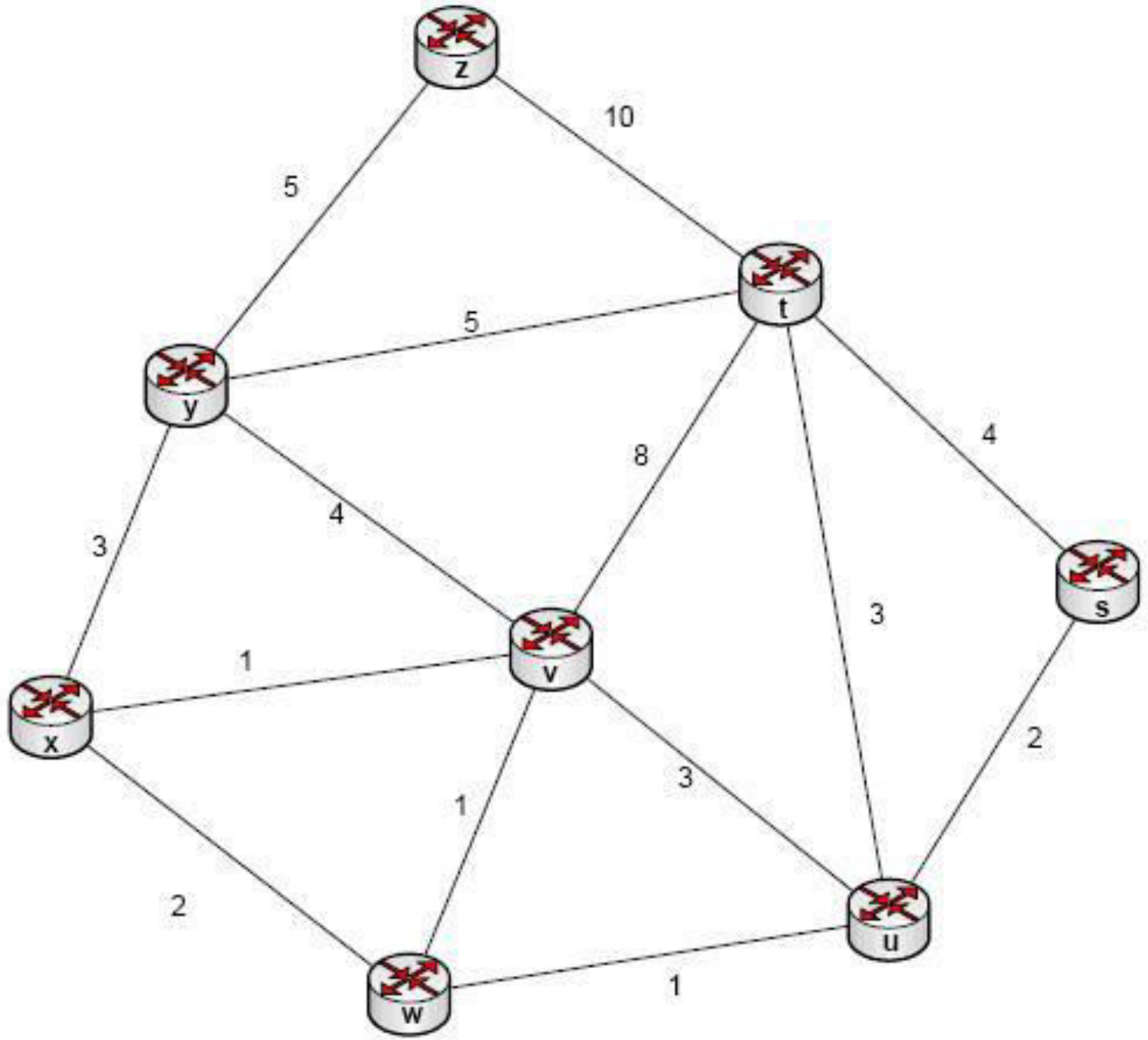
- Prim algorithm chooses the nearest node to signed neighbourhood.
- There is an example below.

Prim Algorithm



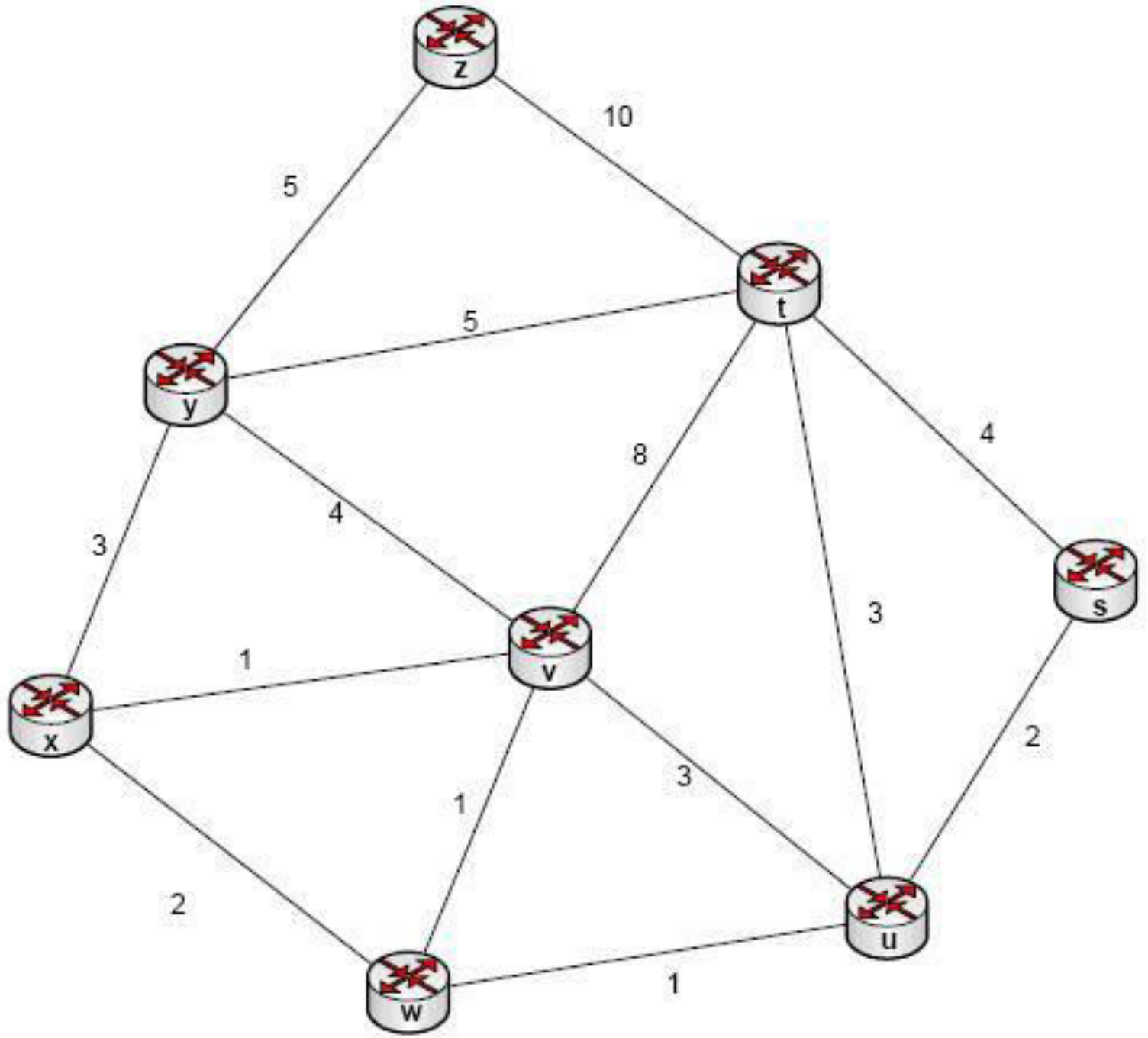
Prim Algorithm

- In this graph, every node has a representative letter and a weight values.



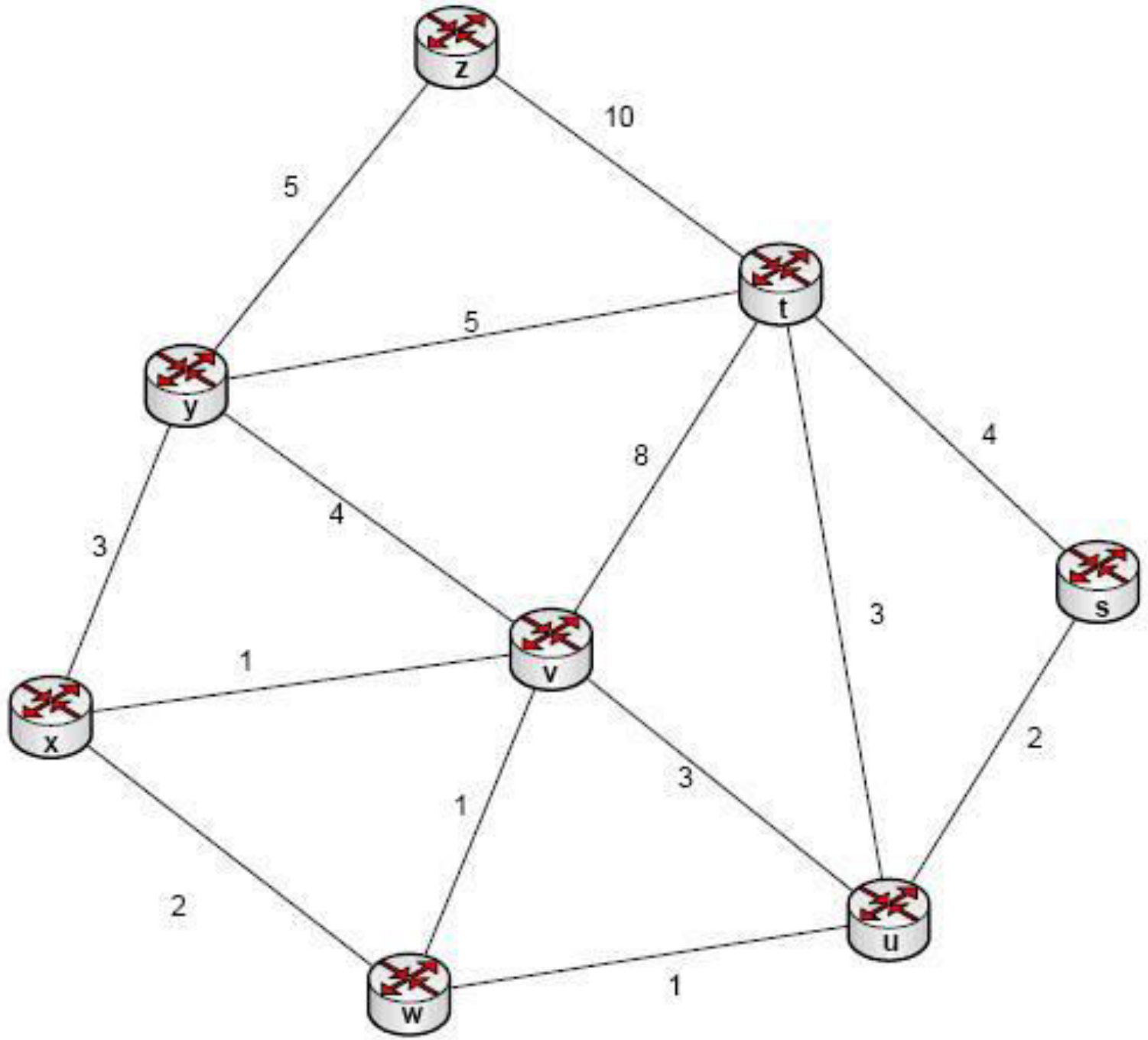
Prim Algorithm

- In Prim algorithm, first of all a random node is chosen.
- For example, let our beginning node be z.
- In this situation, we will first investigate the neighbourhood from z to any other node.
- Also we will investigate the costs from z.



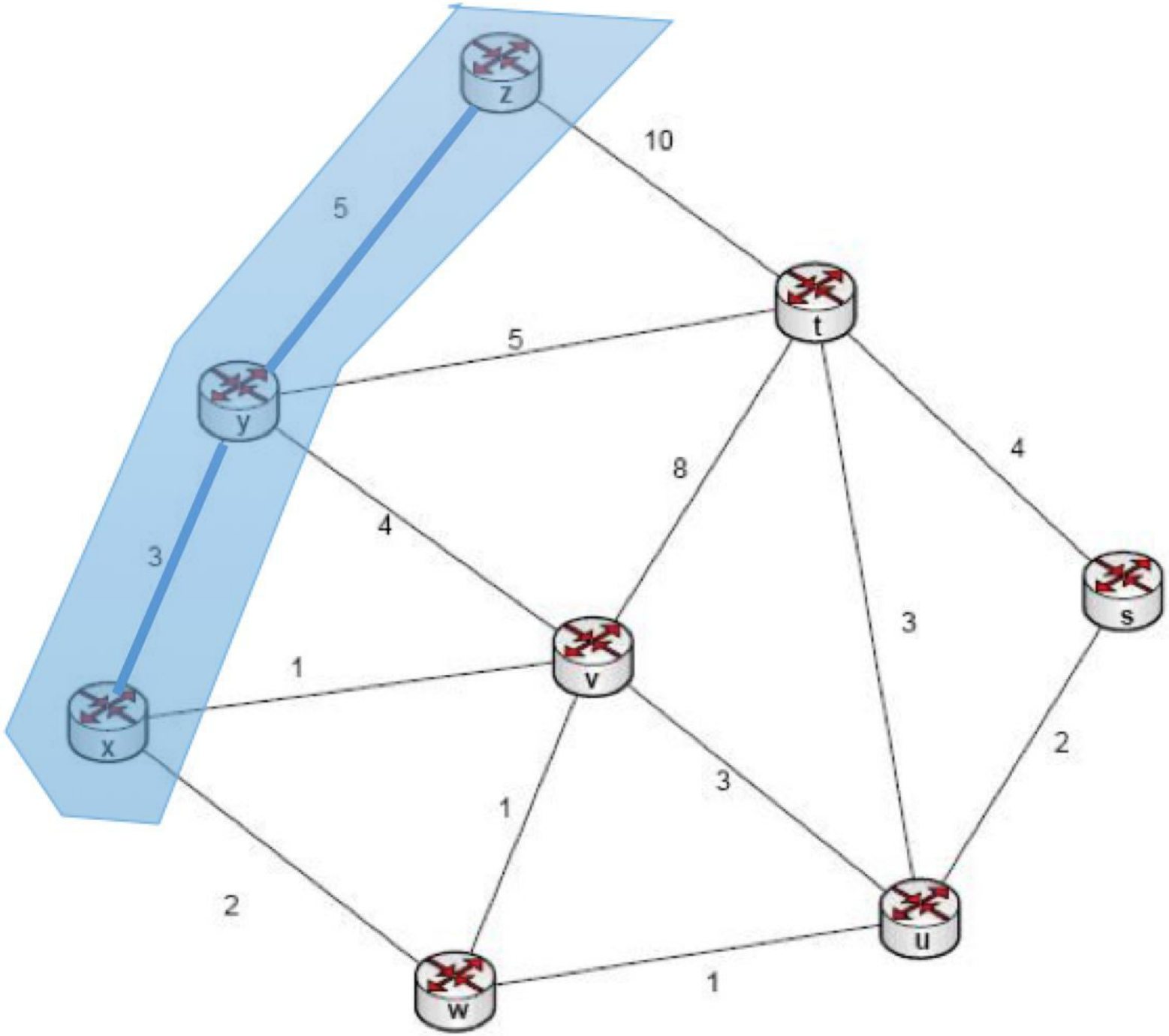
Prim Algorithm

- The nodes we can go from z and the costs are listed below.
- y:5
  t:10



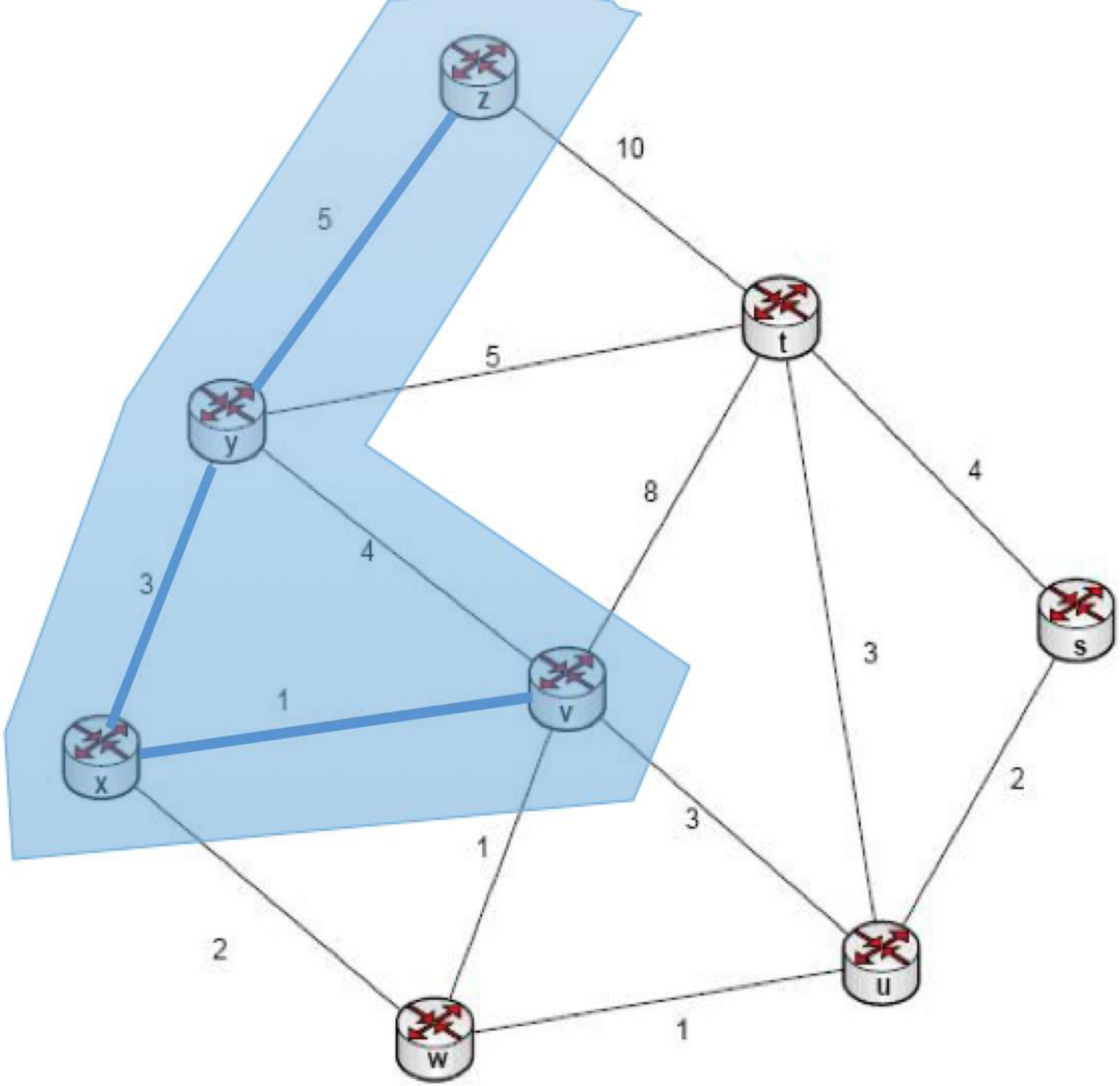
Prim Algorithm

- Prim is a greedy algorithm and chooses the minimum cost node. So the new members will be {z,y} and the weights will be {z-y:5}



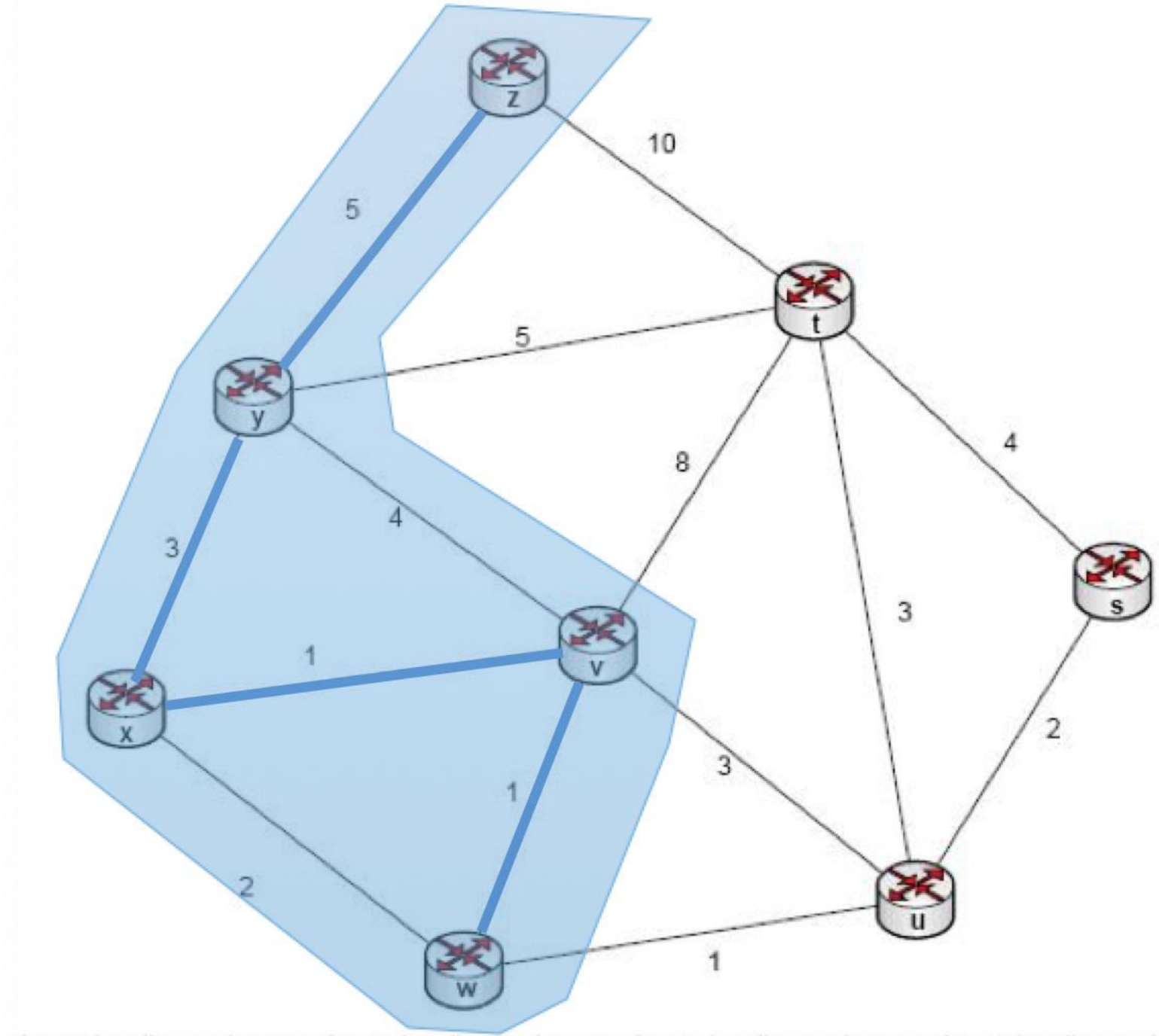
Prim Algorithm

- We will again list the neighbours.
- t:5
  v:1
  w:2
- From these nodes, we will chose the minimum one – v.
- So the member will be {z,y,x,v} and the paths will be {z-y:5,y-x:3,x-v:1}.



Prim Algorithm

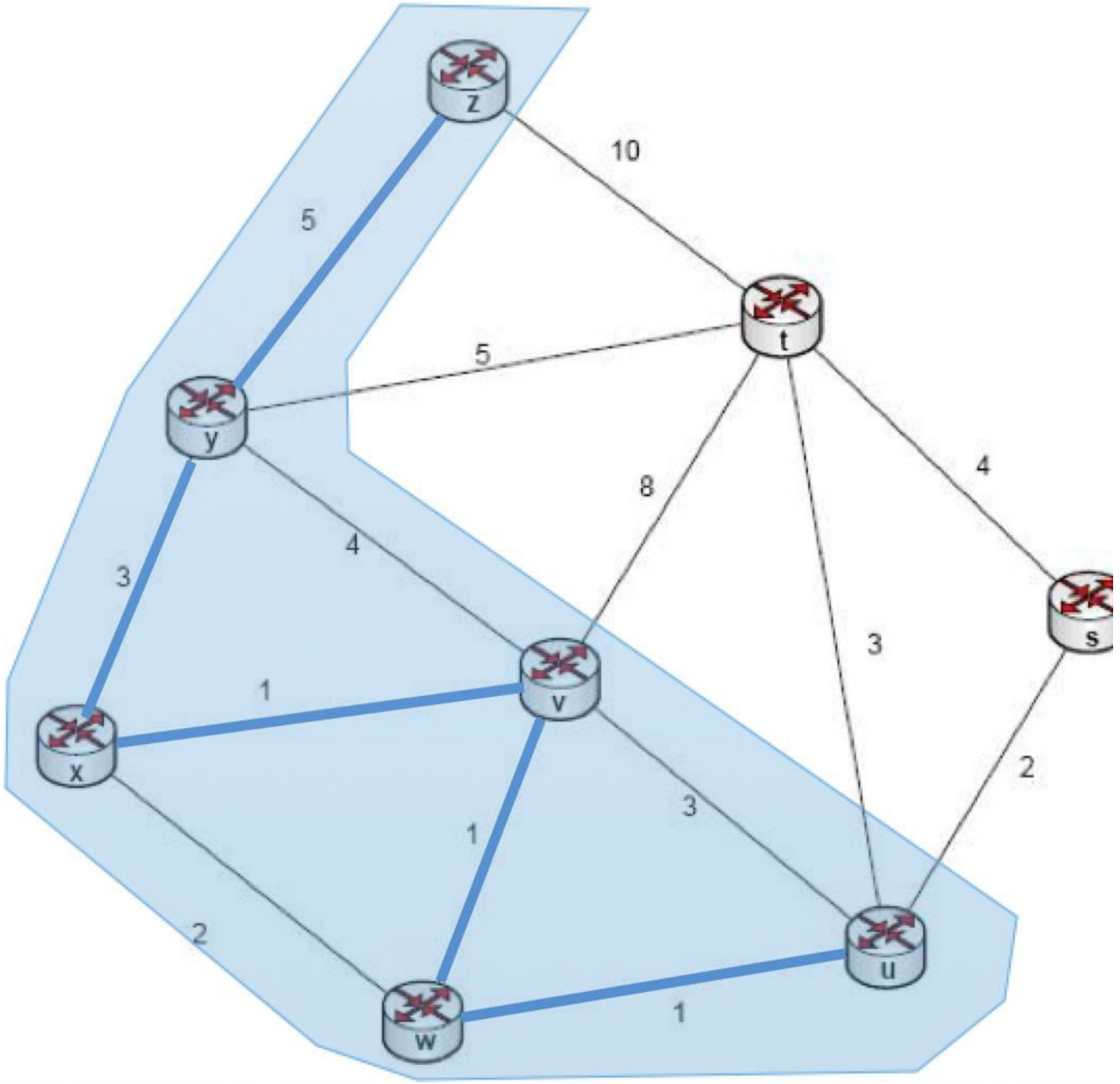
- So we again list the neighbours now.
- t:5
  u:3
  w:1
- The algorithm will choose w:1.
- The members will become {z,y,x,v,w} and the path will be {z-y:5,y-x:3,x-v:1,v-w:1}



Prim Algorithm

- So we again list the neighbours now.
- t:5
  u:1

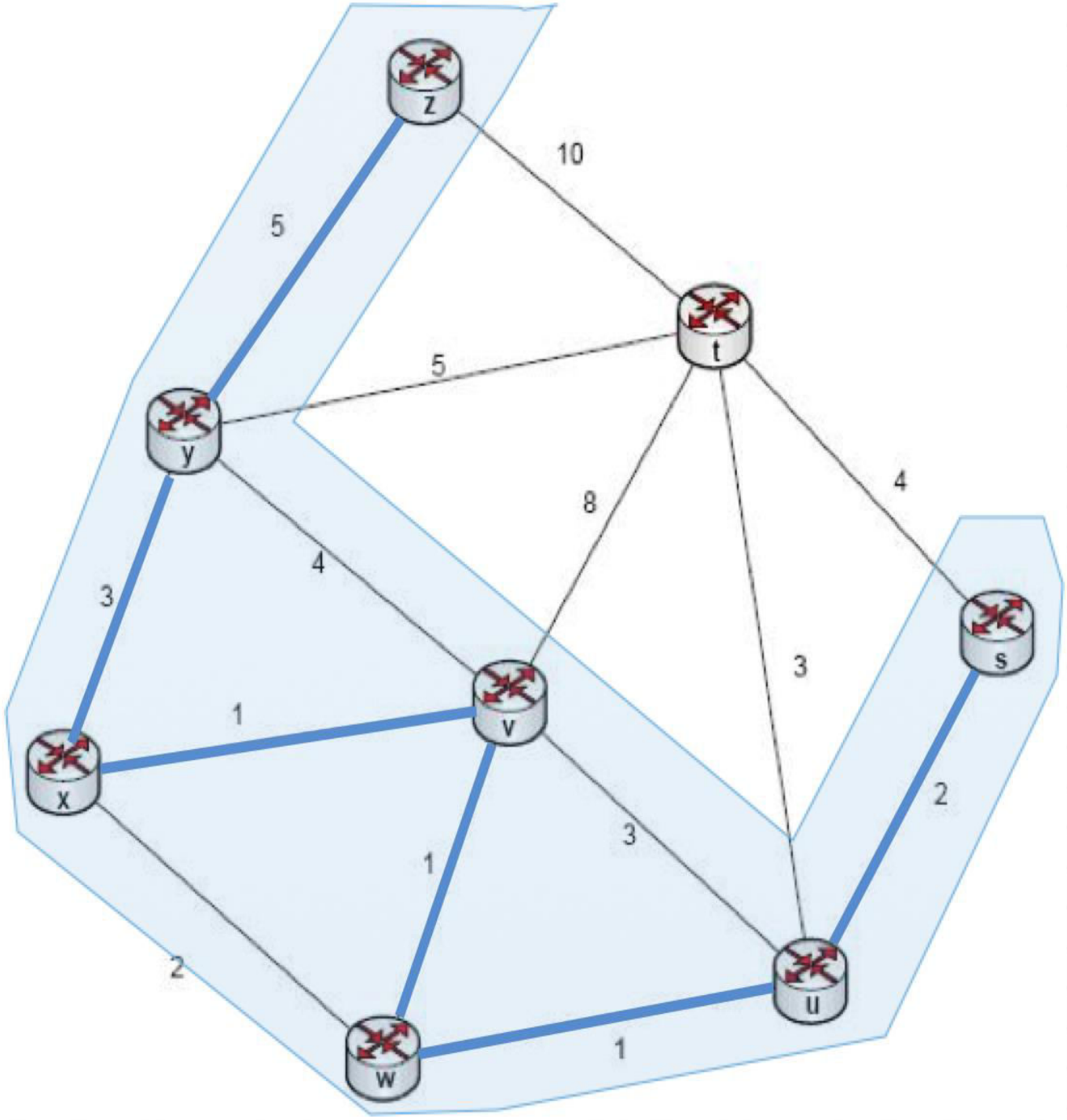
- The algorithm will choose u:1.
- The members will become {z,y,x,v,w,u} and the path will be {z-y:5,y-x:3,x-v:1,v-w:1,w-u:1}



Prim Algorithm

- So we again list the neighbours now.
- t:3
  s:2

- The algorithm will choose s:2.
- The members will become {z,y,x,v,w,u,s} and the path will be {z-y:5,y-x:3,x-v:1,v-w:1,w-u:1,u-s:2}



Prim Algorithm

- For the last node t, our minimum weight is t:3 so the members become {z,y,x,v,w,u,s,t} and the path becomes
- {z-y:5,y-x:3,x-v:1,v-w:1,w-u:1,u-s:2,u-t:3}
- So the spanning tree is given below.