Graph visits

Programmazione Avanzata 2025-26

Visit algorithms

- Visit
  - Systematic exploration of a graph
  - Starting from a “source” vertex
  - Reaching all reachable vertices
- Main strategies
  - Breadth-First Visit
  - Depth-First Visit

Breadth-First Visit

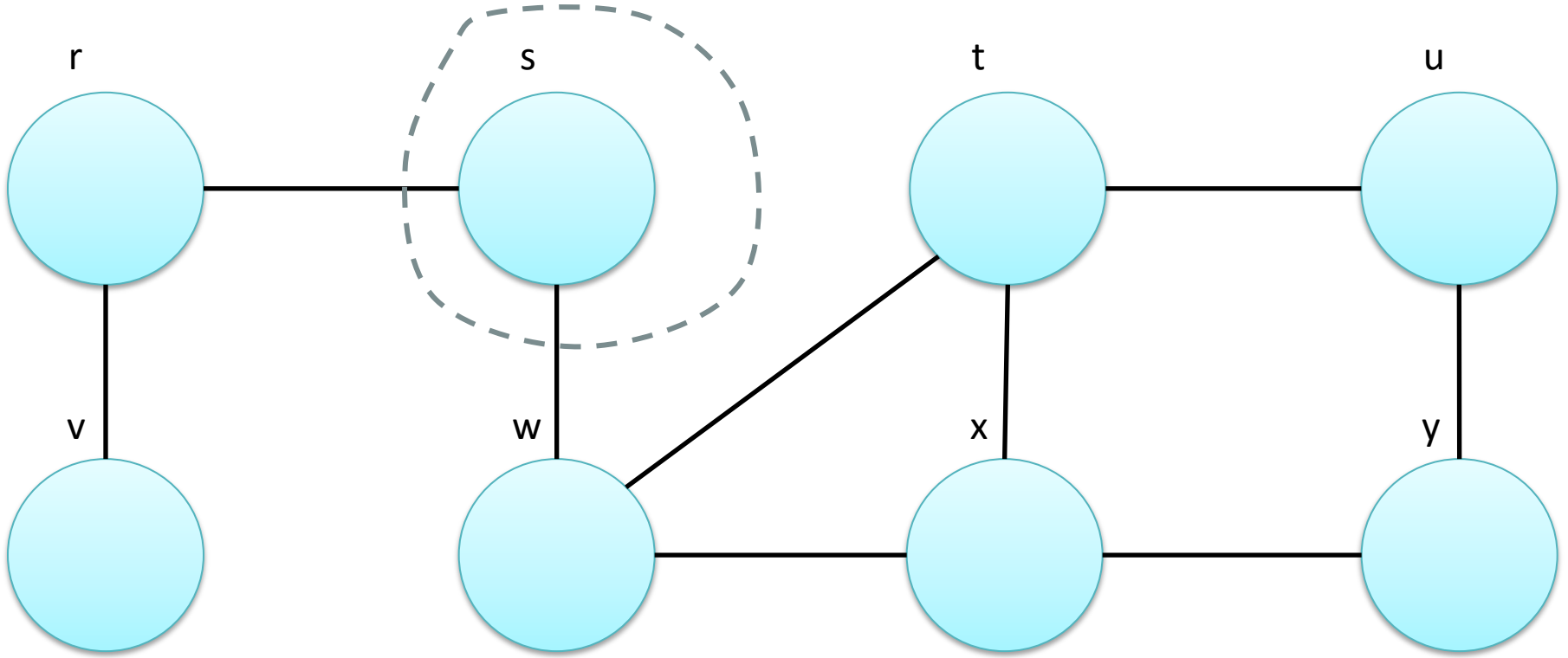
- Also called Breadth-First Search (BFV or BFS)
- All reachable vertices are visited “by levels”
  - L, level of the visit
  - S_L, set of vertices in level L
  - L=0, $S_0=\{v_{source}\}$

BFS algorithm

- Repeat while S_L is not empty:
  - S_{L+1}, set of all vertices:
    - Not visited yet, and
    - Adjacent to at least one vertex in S_L
  - L=L+1

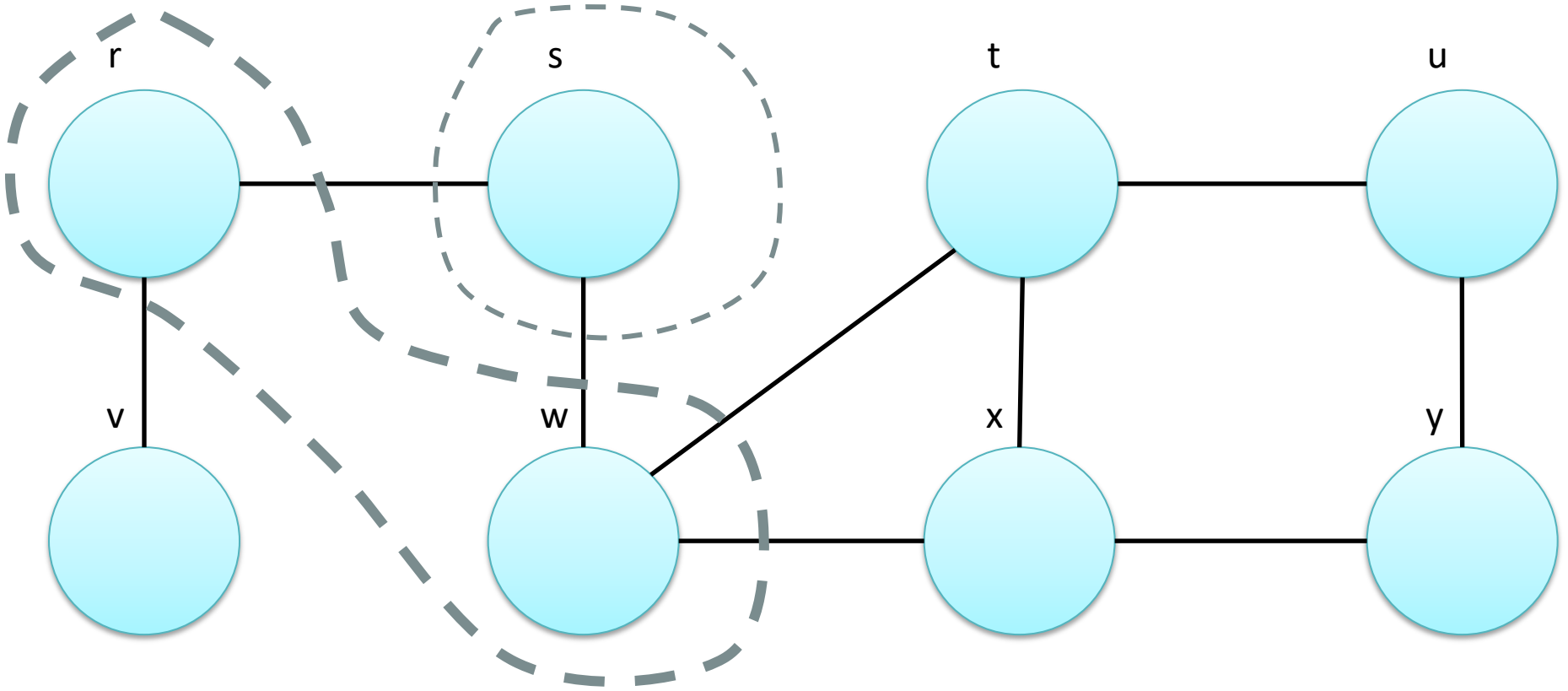
Example

Source = s
L = 0
S_0 = {s}



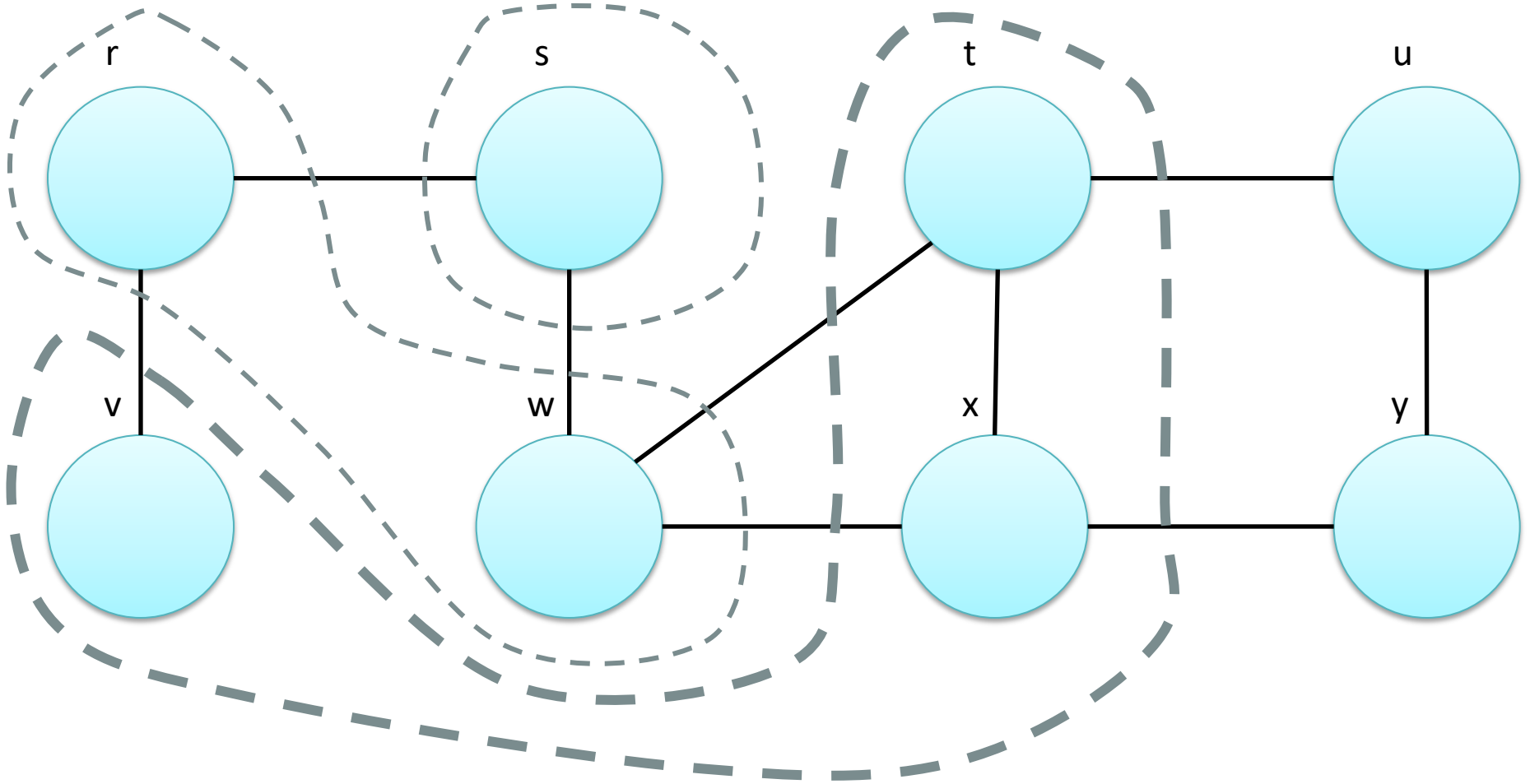
Example

L = 1
S_0 = {s}
S_1 = {r, w}



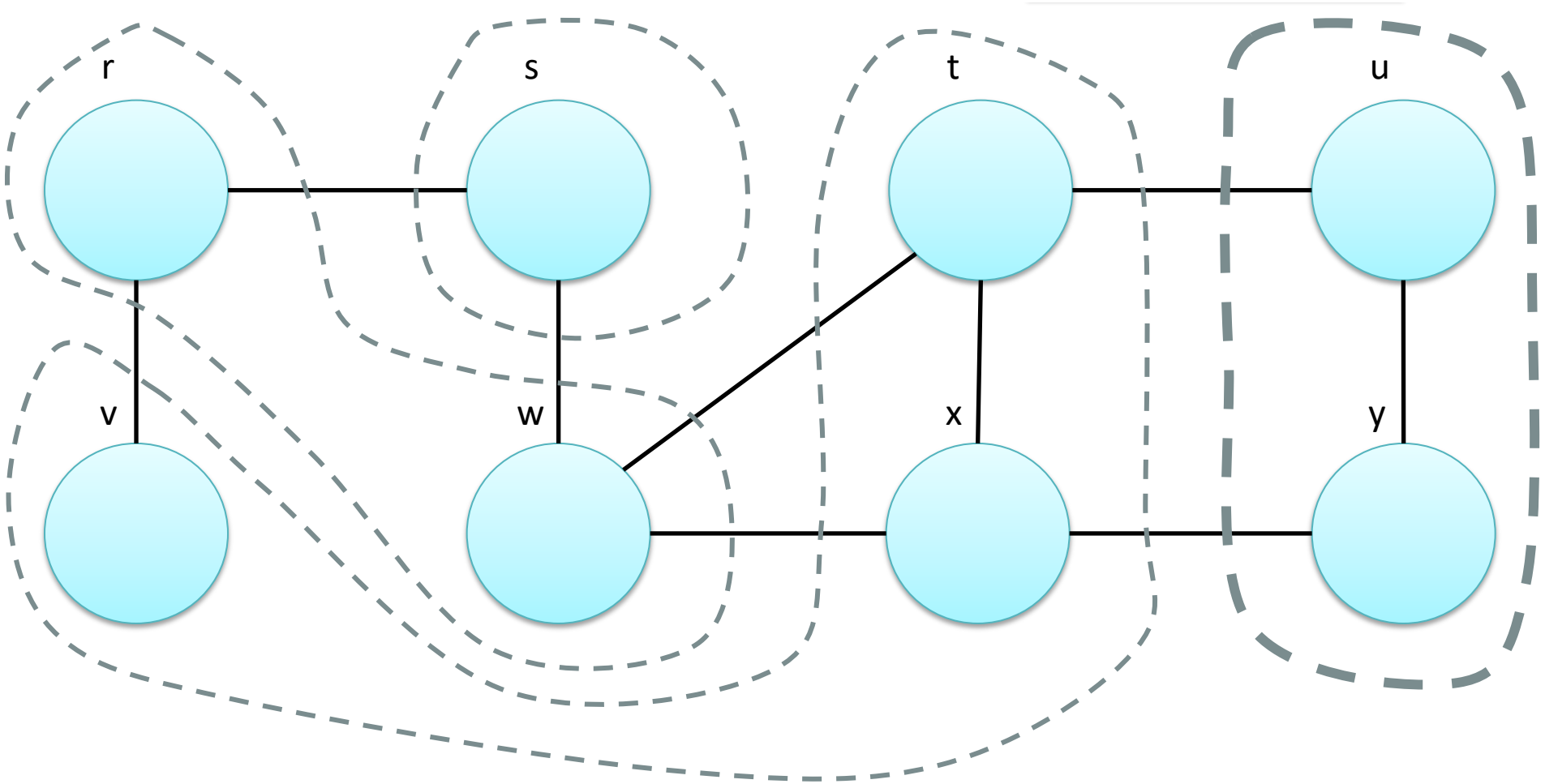
Example

L = 2
S_1 = {r, w}
S_2 = {v, t, x}



Example

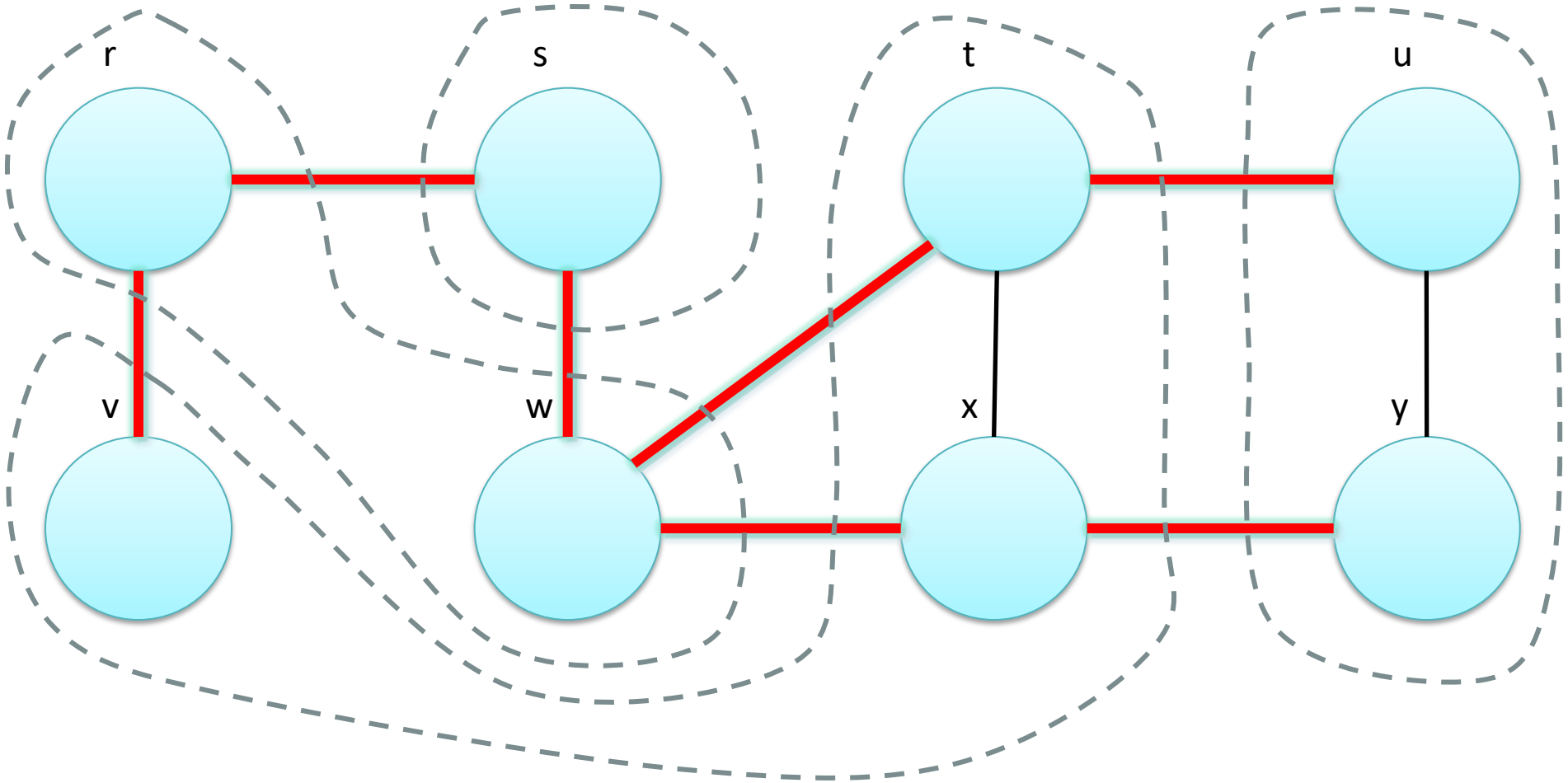
L = 3
S_2 = {v, t, x}
S_3 = {u, y}



BFS Tree

- The result of a BFV identifies a “visit tree” in the graph:
  - The tree root is the source vertex
  - Tree nodes are all graph vertices
    - (In the same connected component of the source)
  - Tree edges are a subset of graph edges
    - Those edges that have been used to “discover” new vertices

BFS Tree



Minimum (shortest) paths

- Shortest path: the minumum number of edges on any path between two vertices
- The BFS algorithm computes all minimum paths for all vertices, starting from the source vertex
- Unweighted graphs: path length = number of edges

Depth First Visit

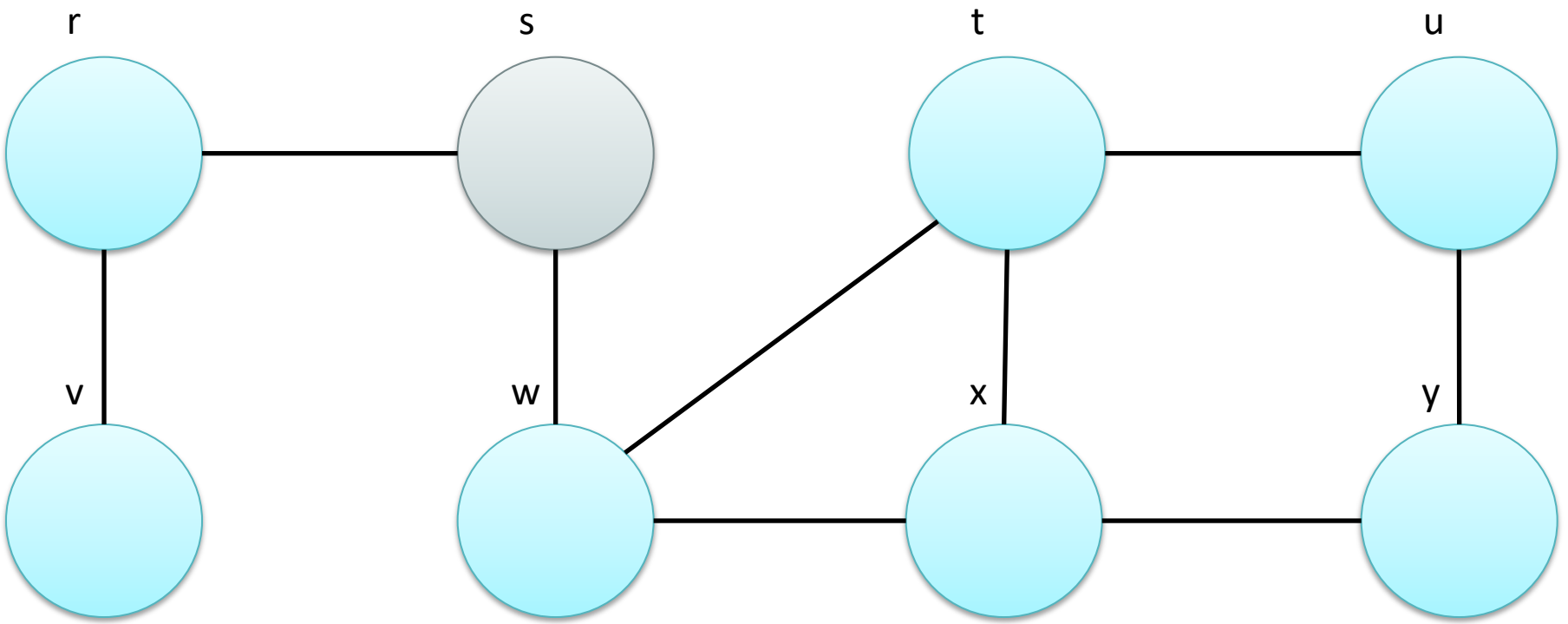
- Also called Depth-first search (DFV or DFS)
- Opposite approach to BFS
  - At every step, visit one (yet unvisited) vertex, adjacent to the last visited one
  - If no such vertex exist, go back one step to the previously visited vertex
  - Lends itself to recursive implementation
    - Similar to tree visit procedures

DFS algorithm

- DFS(Vertex v)
  - For all (w : adjacent_to(v))
  - If(not visited (w))
    - Visit (w)
    - DFS(w)
- Start with: DFS(v_{source})

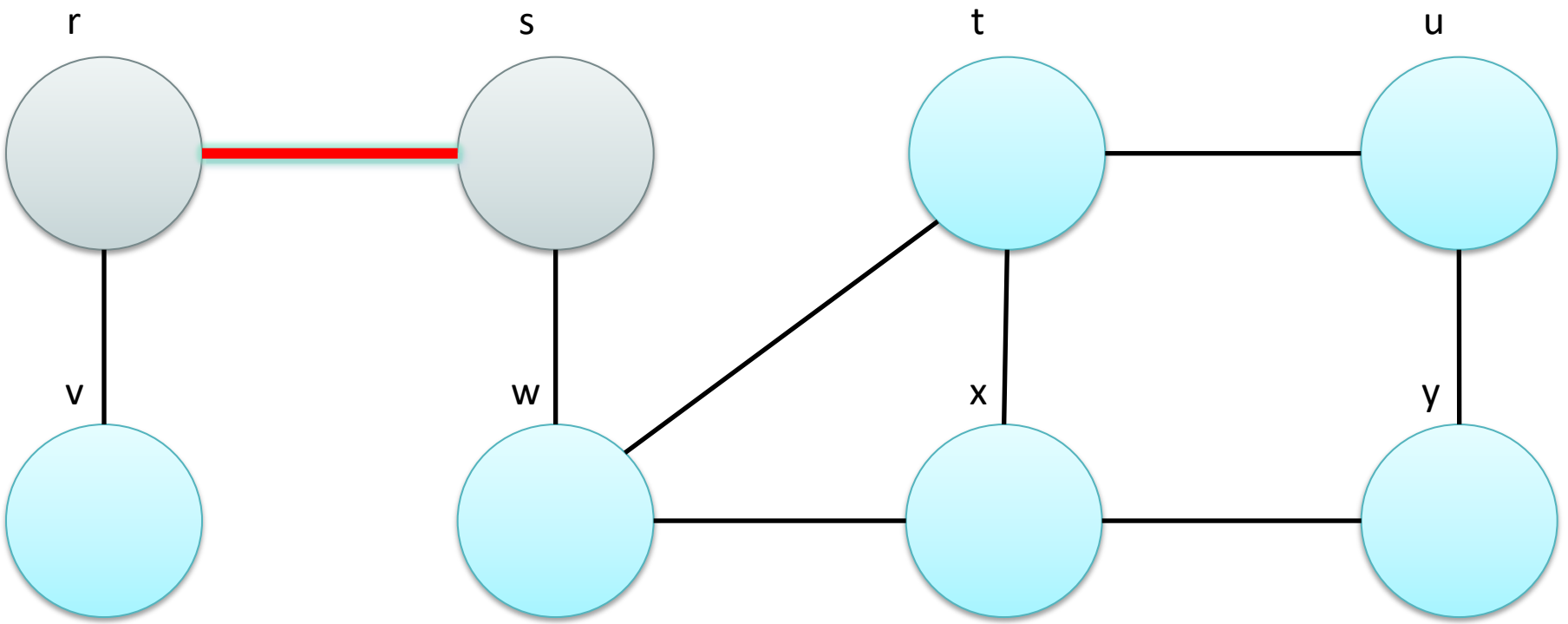
Example

Source = s



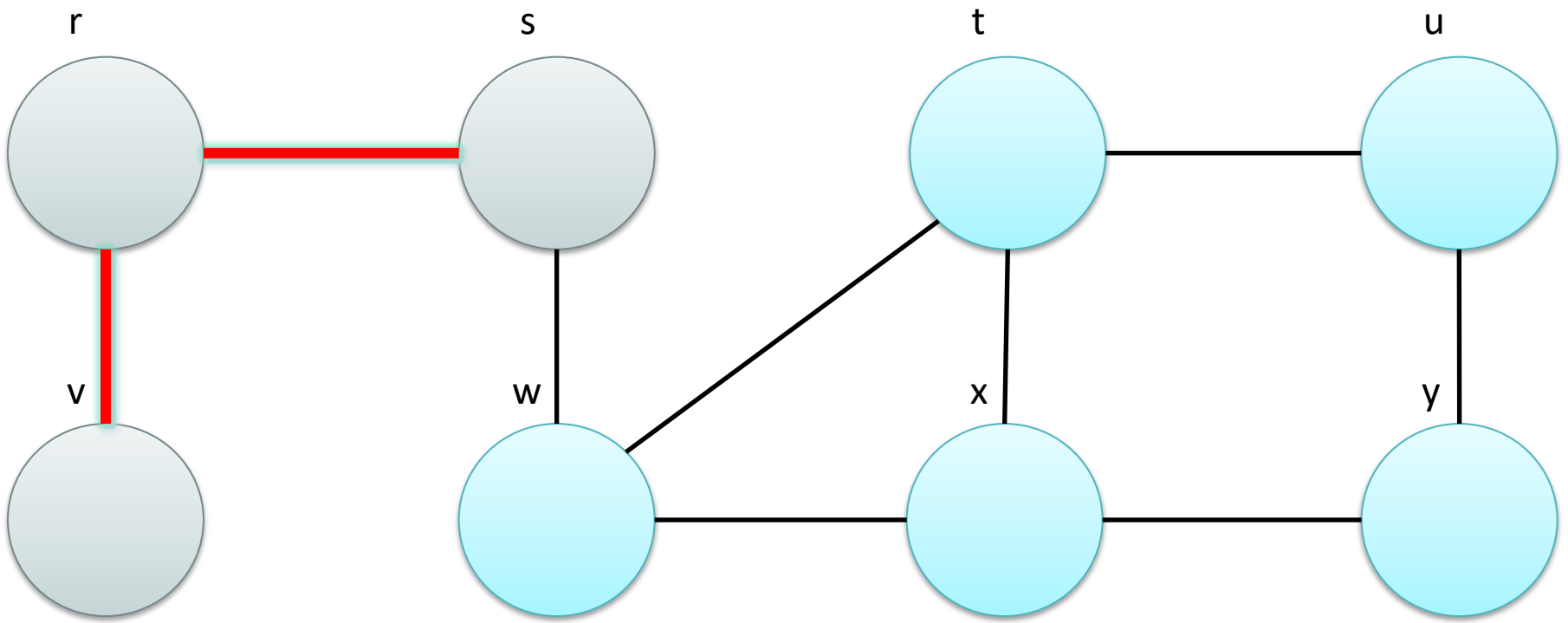
Example

Source = s
Visit r



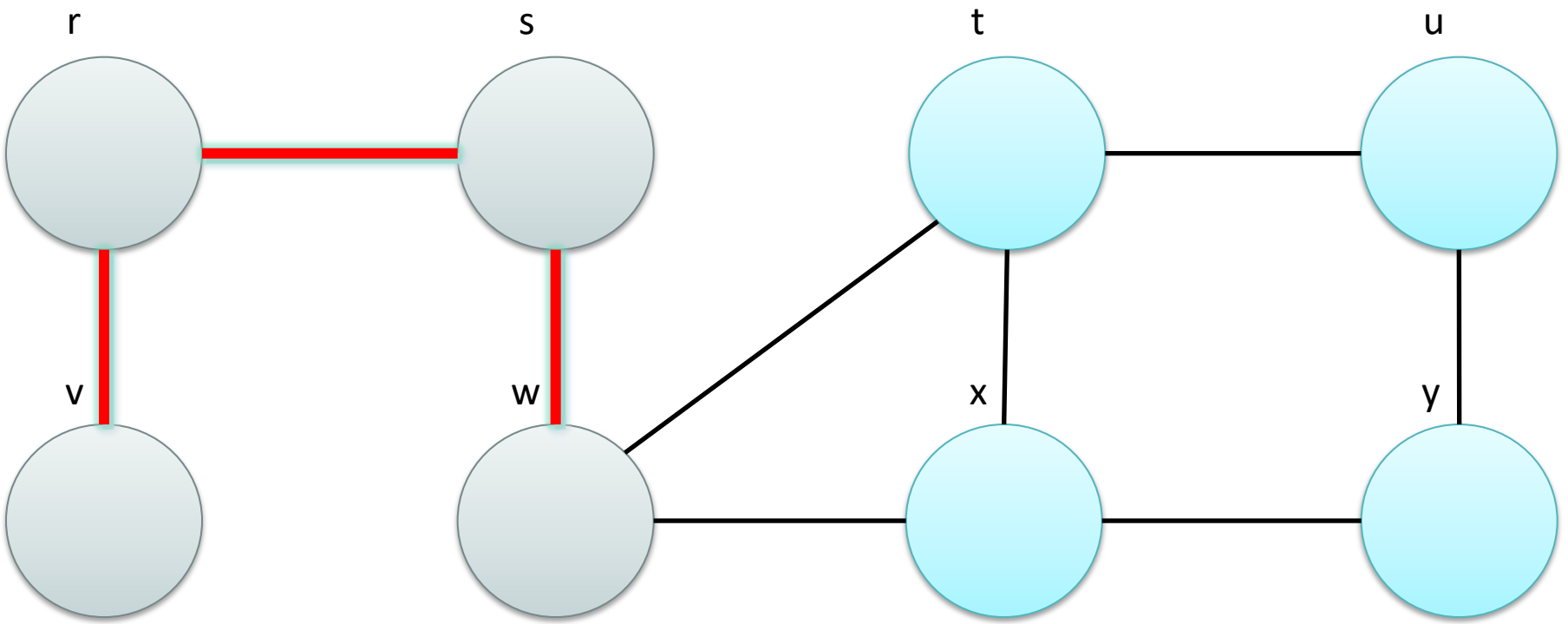
Example

Source = s
Visit r
Visit v



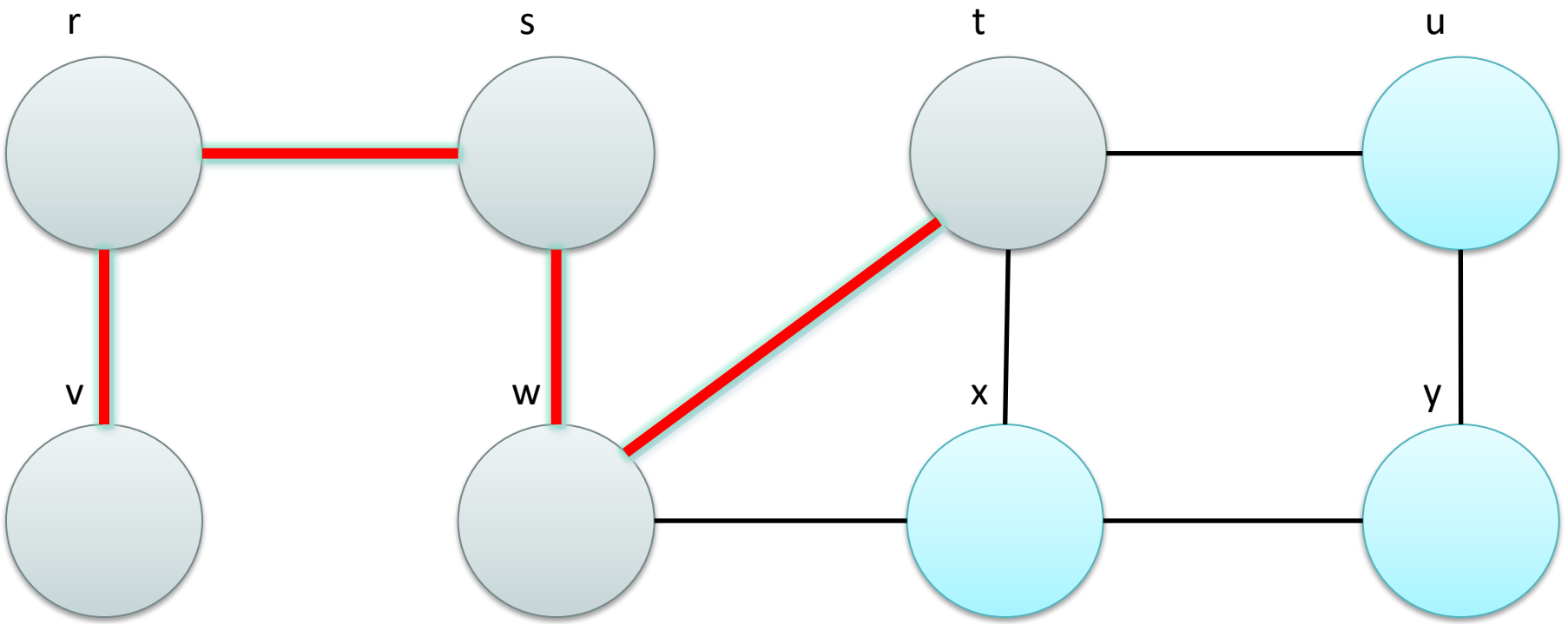
Example

Source = s
Visit w



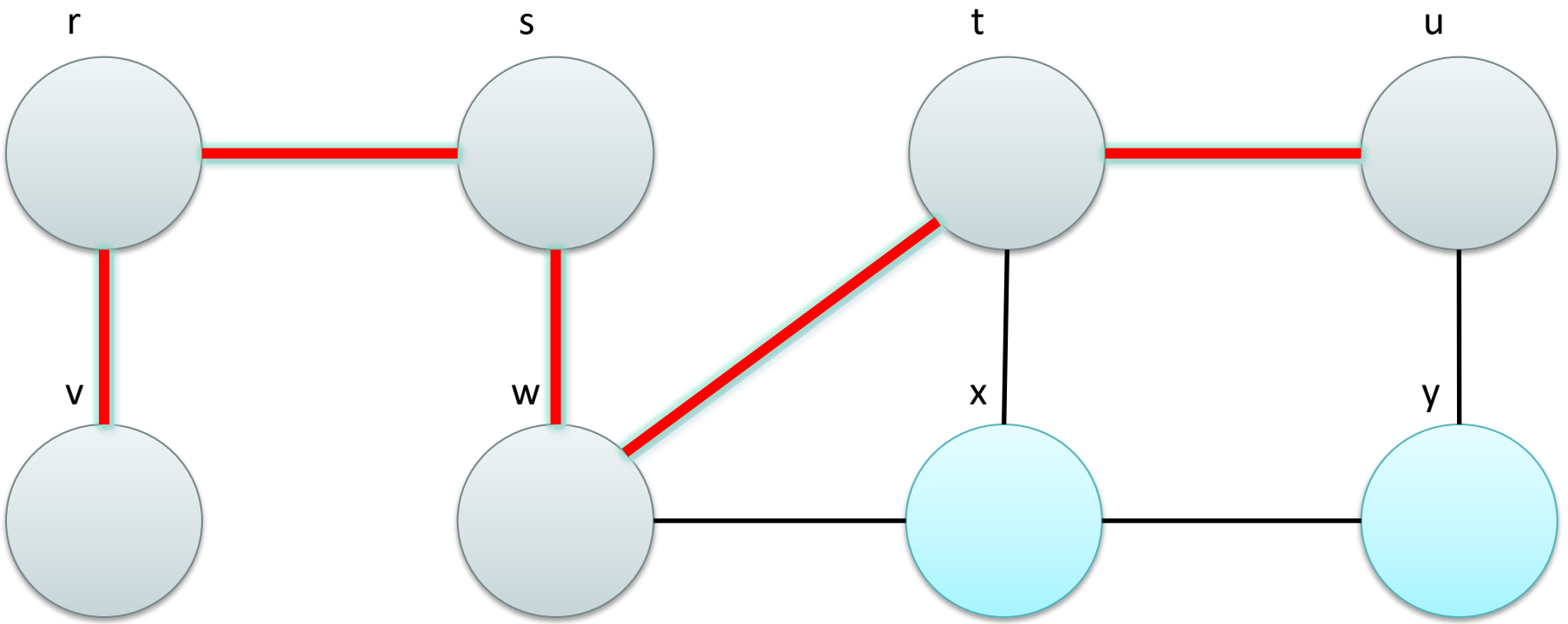
Example

Source = s
Visit w
Visit t



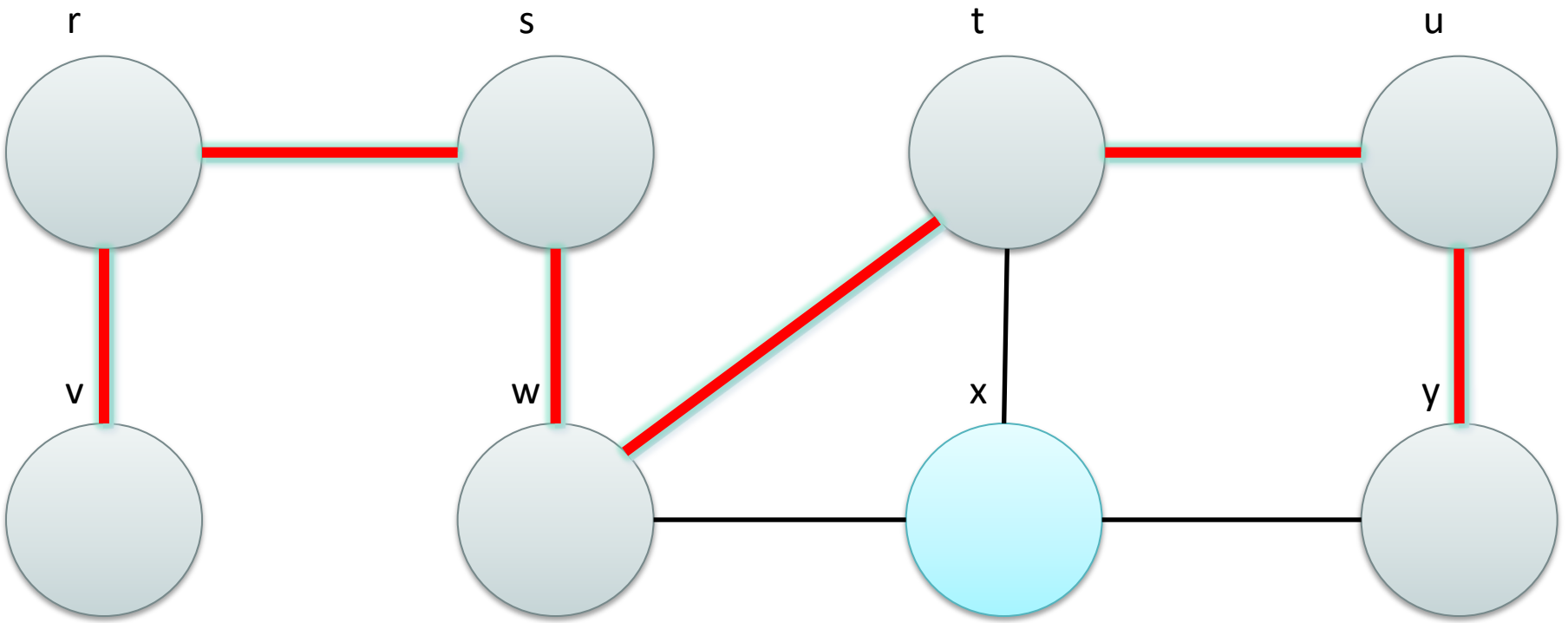
Example

Source = s
Visit w
Visit t
Visit u



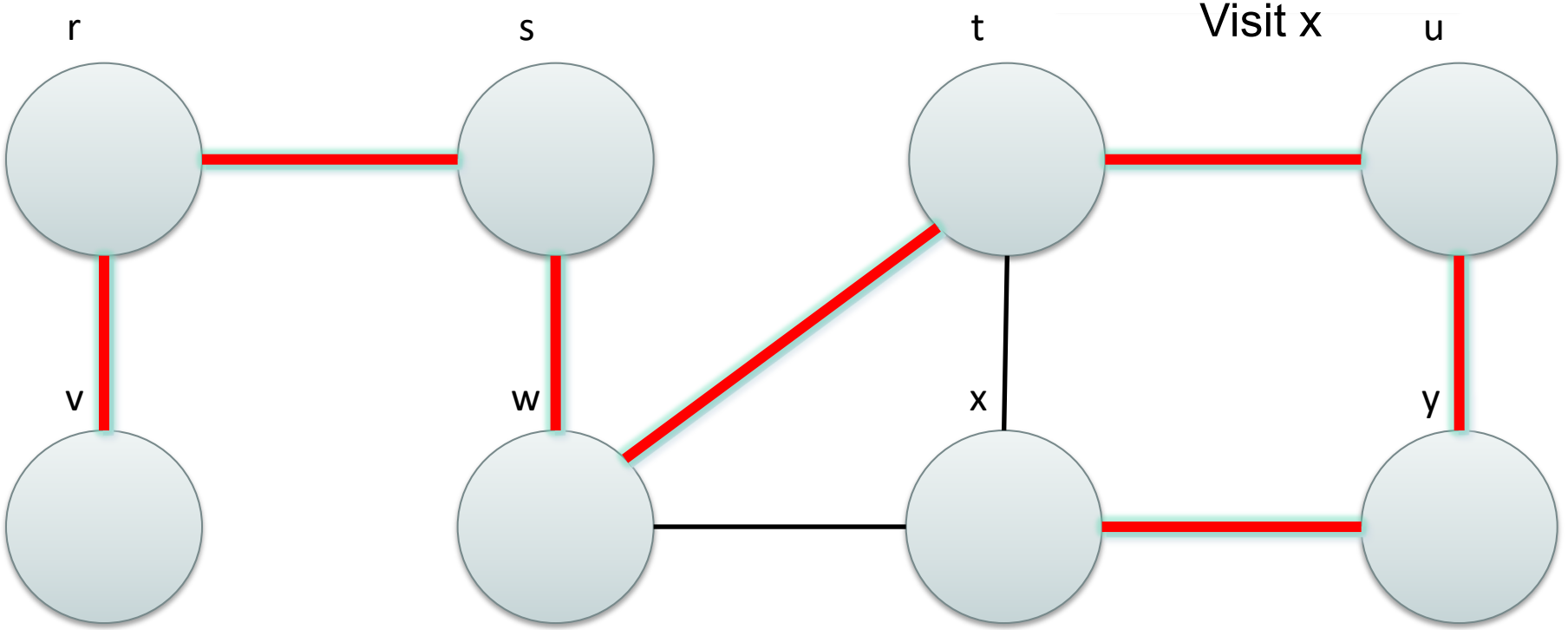
Example

Source = s
Visit w
Visit t
Visit u
Visit y



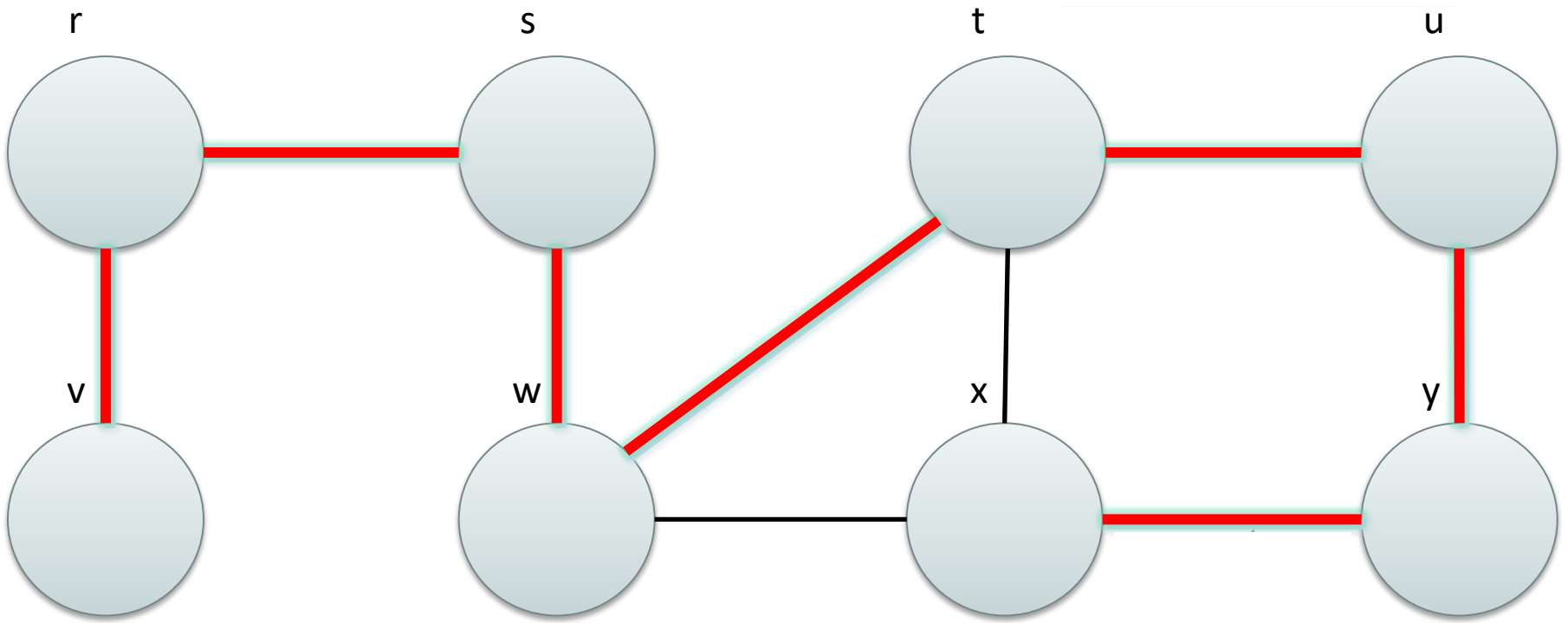
Example

Source = s
Visit w
Visit t
Visit u
Visit y
Visit x



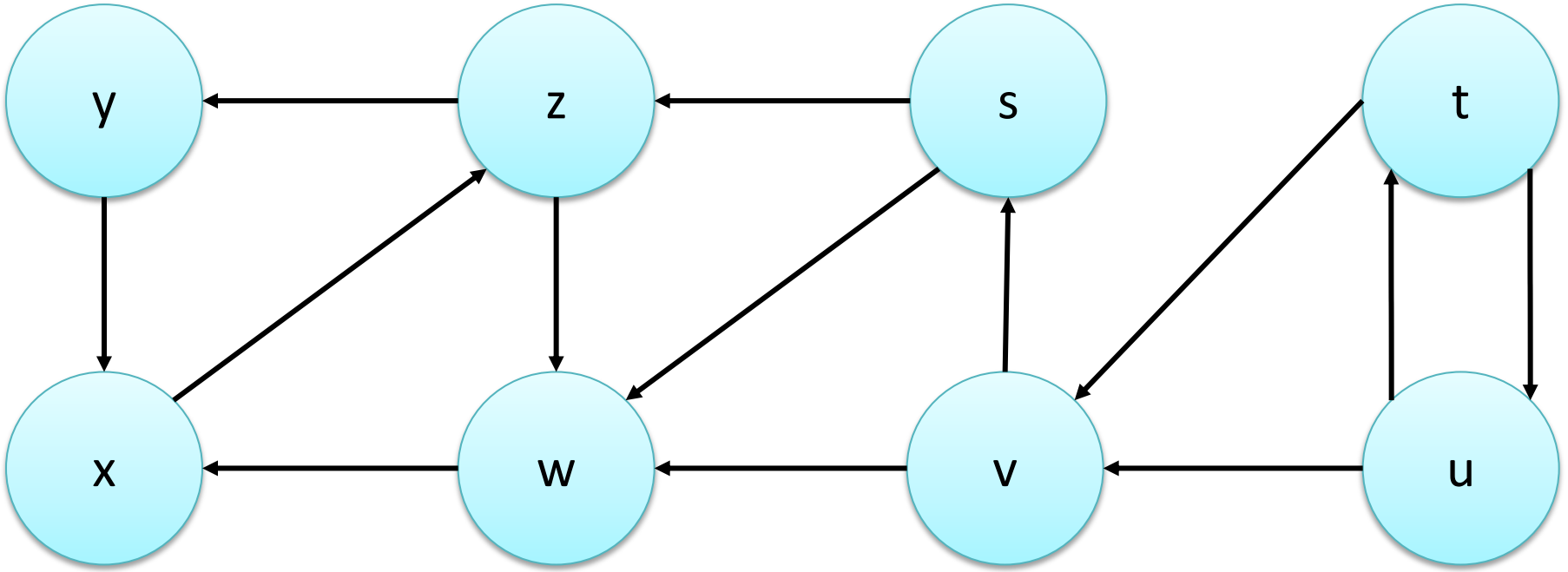
DFS tree

Back to s and stop



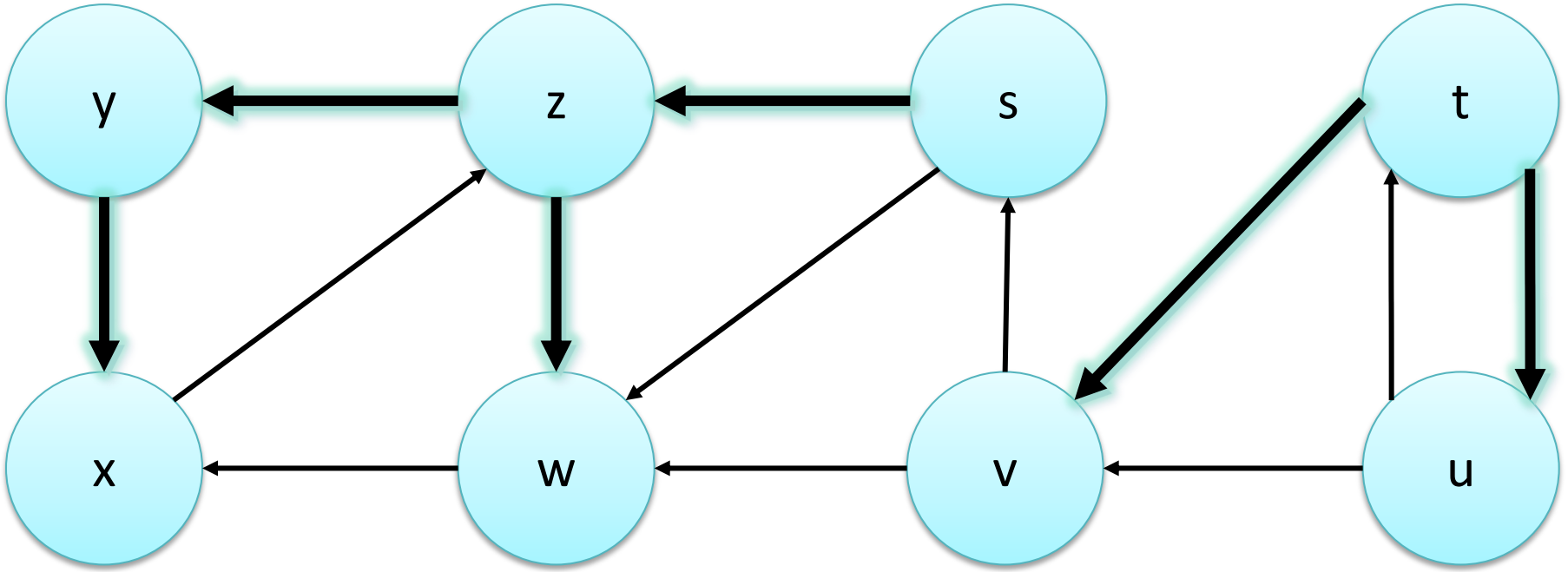
Example

Directed graph



Example

DFS visit
Sources: s, t



Complexity

- Visits have linear complexity in the graph size
  - BFS : O(V+E)
  - DFS : Q(V+E)
- For dense graphs, $E = O(V^2)$

Visits in NetworkX

- Visits are called “traversals”
- NetworkX already provides implementations for BFV and DFV, together with other visits strategies
  - https://networkx.org/documentation/stable/reference/algorithms/traversal.html

Graph traversal methods

Breadth First Search

Basic algorithms for breadth-first searching the nodes of a graph.

`bfs_edges` (G, source[, reverse, depth_limit, ...])	Iterate over edges in a breadth-first-search starting at source.
`bfs_layers` (G, sources)	Returns an iterator of all the layers in breadth-first search traversal.
`bfs_tree` (G, source[, reverse, depth_limit, ...])	Returns an oriented tree constructed from of a breadth-first-search starting at source.
`bfs_predecessors` (G, source[, depth_limit, ...])	Returns an iterator of predecessors in breadth-first-search from source.
`bfs_successors` (G, source[, depth_limit, ...])	Returns an iterator of successors in breadth-first-search from source.
`descendants_at_distance` (G, source, distance)	Returns all nodes at a fixed `distance` from `source` in `G`.
`generic_bfs_edges` (G, source[, neighbors, ...])	Iterate over edges in a breadth-first search.

Depth First Search

Basic algorithms for depth-first searching the nodes of a graph.

`dfs_edges` (G[, source, depth_limit, ...])	Iterate over edges in a depth-first-search (DFS).
`dfs_tree` (G[, source, depth_limit, ...])	Returns oriented tree constructed from a depth-first-search from source.
`dfs_predecessors` (G[, source, depth_limit, ...])	Returns dictionary of predecessors in depth-first-search from source.
`dfs_successors` (G[, source, depth_limit, ...])	Returns dictionary of successors in depth-first-search from source.
`dfs_preorder_nodes` (G[, source, depth_limit, ...])	Generate nodes in a depth-first-search pre-ordering starting at source.
`dfs_postorder_nodes` (G[, source, ...])	Generate nodes in a depth-first-search post-ordering starting at source.
`dfs_labeled_edges` (G[, source, depth_limit, ...])	Iterate over edges in a depth-first-search (DFS) labeled by type.