Depth First Search

About this lecture

- Depth First Search
  - DFS Tree and DFS Forest
  - Parenthesis theorem

Depth First Search (DFS)

- An alternative algorithm to find all vertices reachable from a particular source vertex s
- Idea:

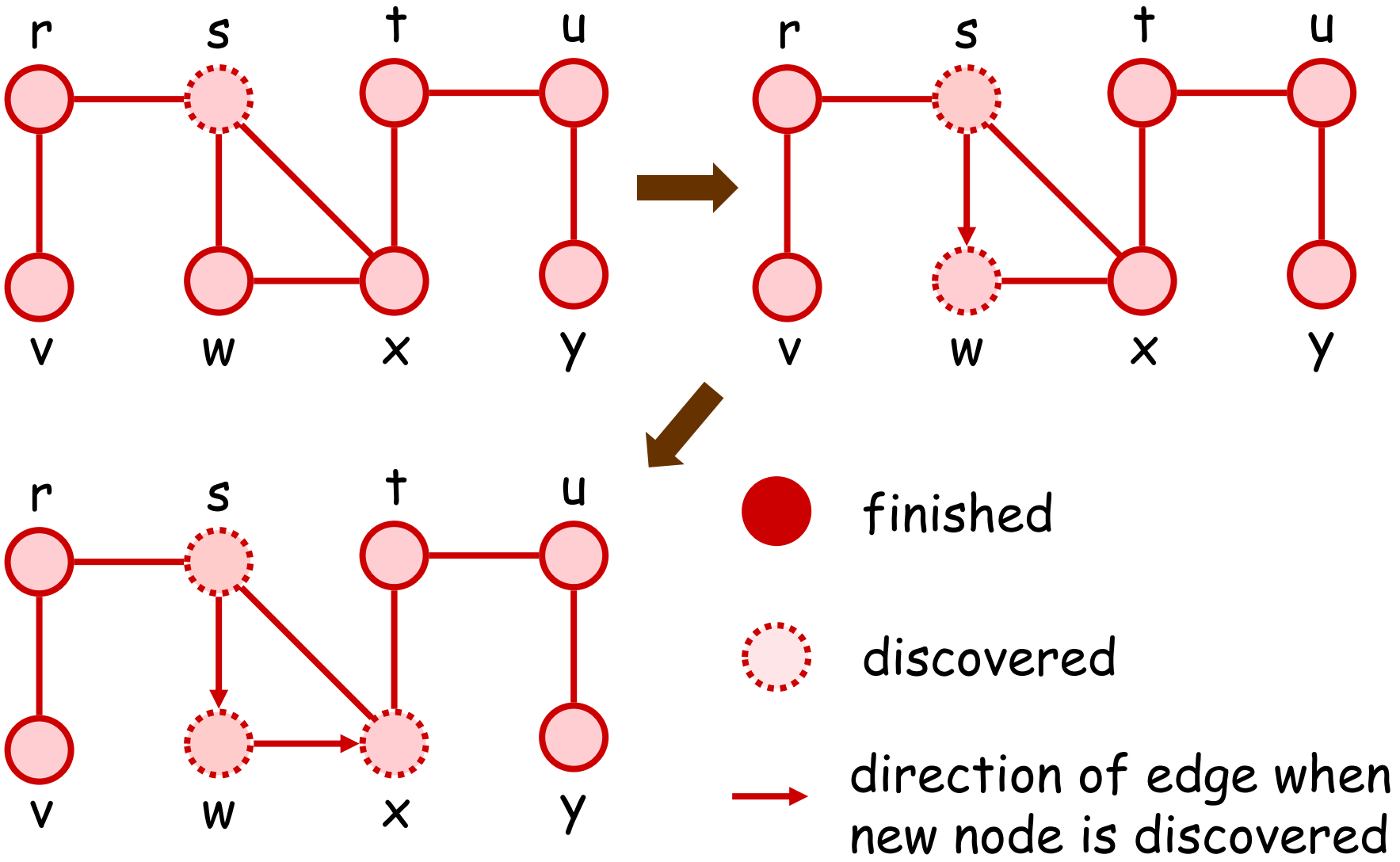
  Explore a branch as far as possible before exploring another branch
- Easily done by recursion or stack

The DFS Algorithm

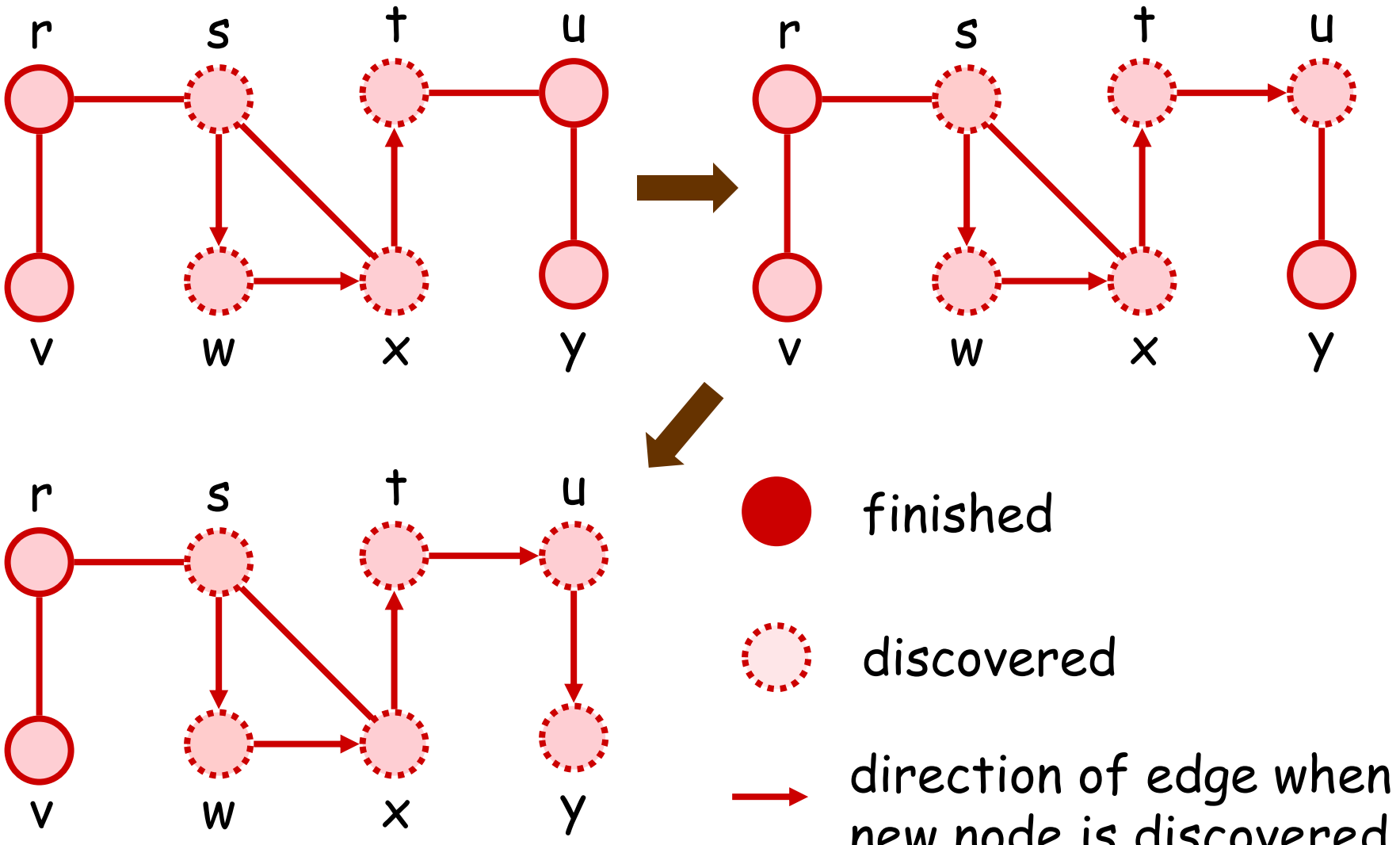
```
DFS(u)
{    Mark u as discovered ;
     while (u has unvisited neighbor v)
              DFS(v);
     Mark u as finished ;
}
```

The while-loop explores a branch as far as possible before the next branch

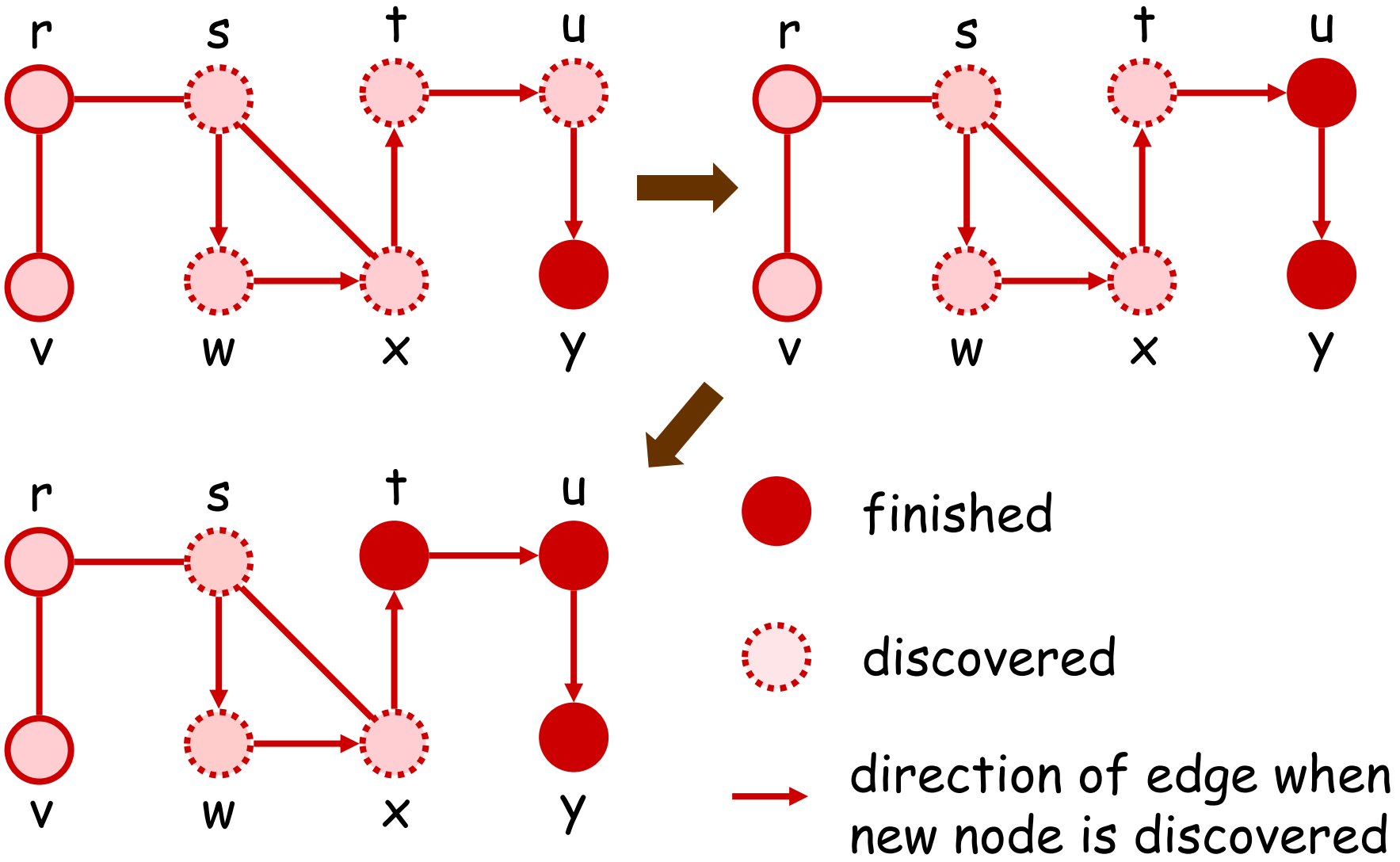
Example (s = source)



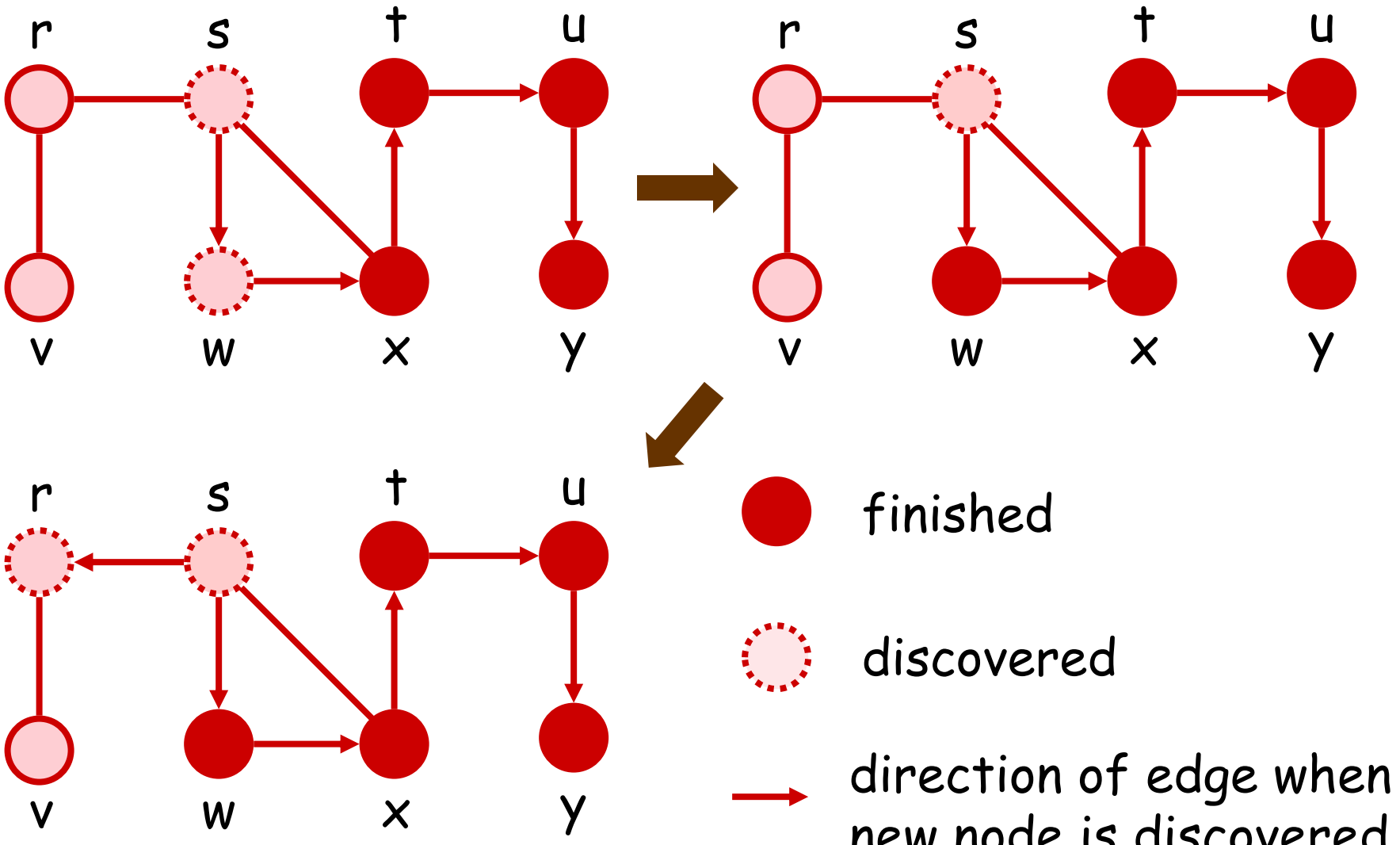
Example (s = source)



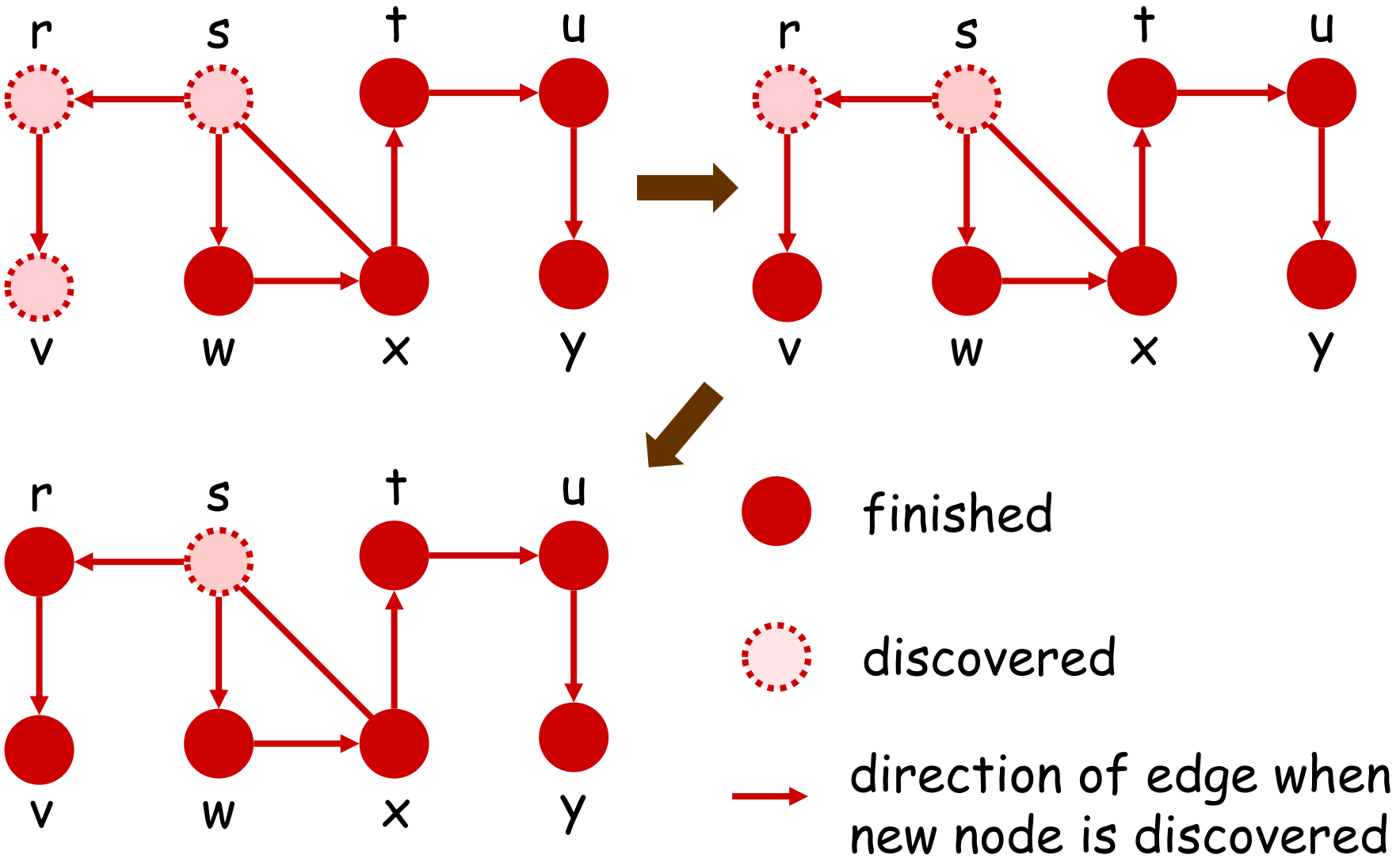
Example (s = source)



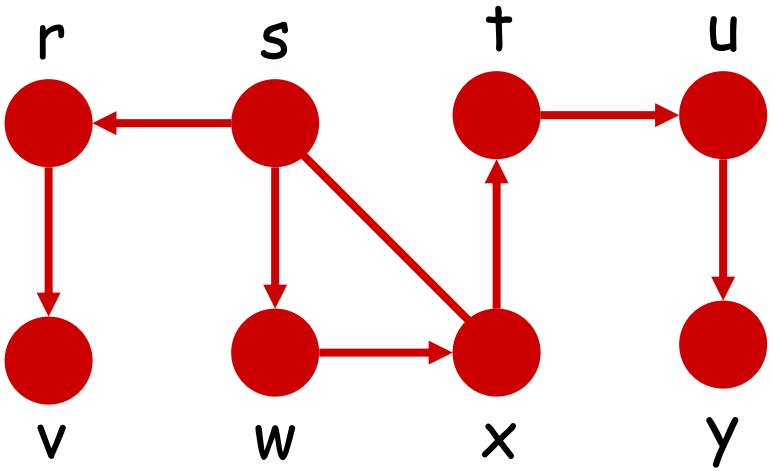
Example (s = source)



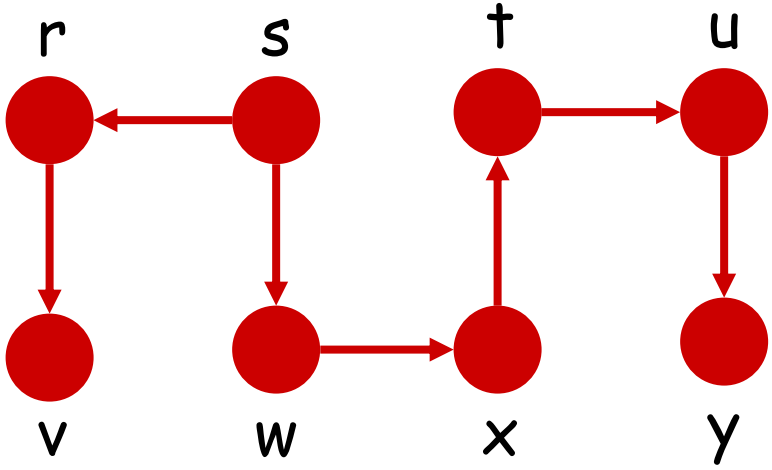
Example (s = source)



Example (s = source)



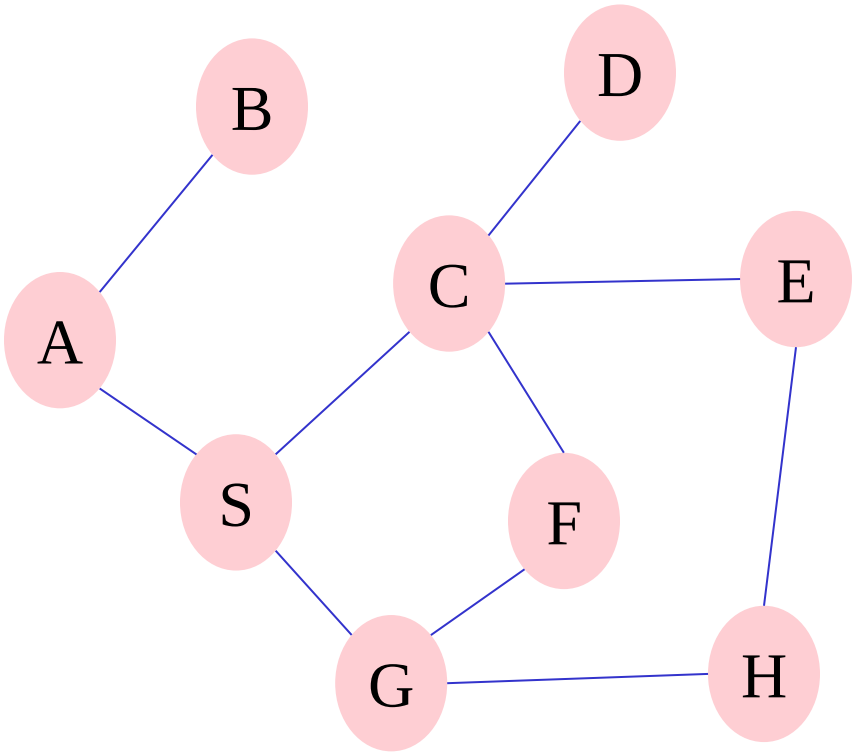
Done when s is discovered

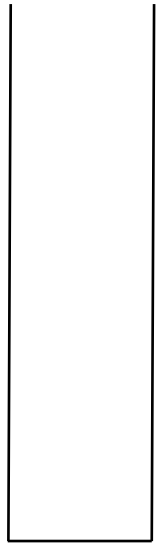


The directed edges form a tree that contains all nodes reachable from s

Called DFS tree of s

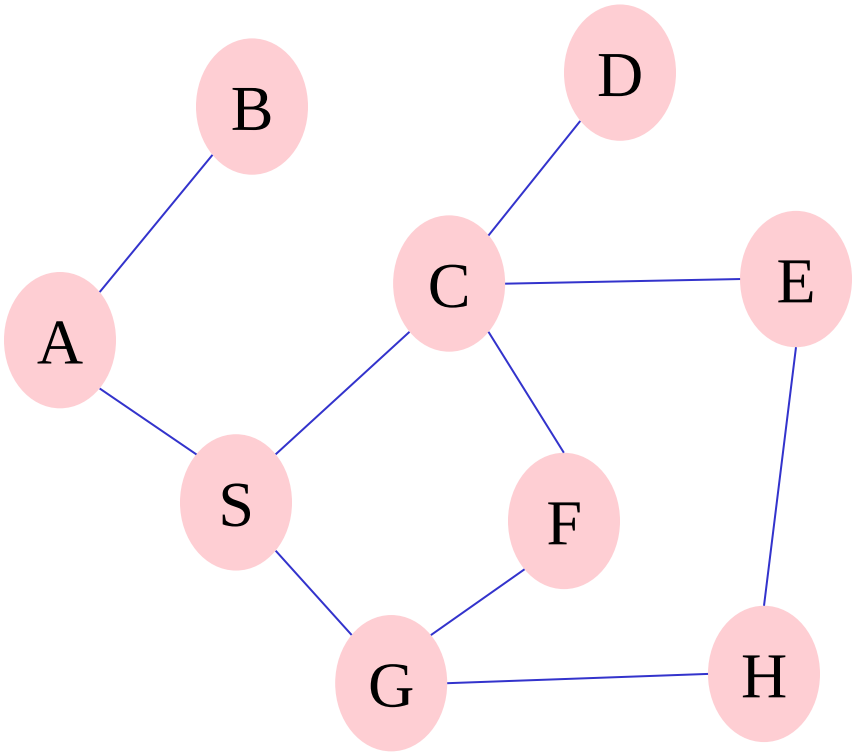
DFS using Stacks

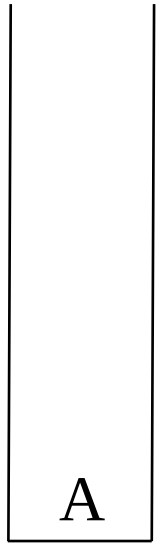




Output:

DFS using Stacks

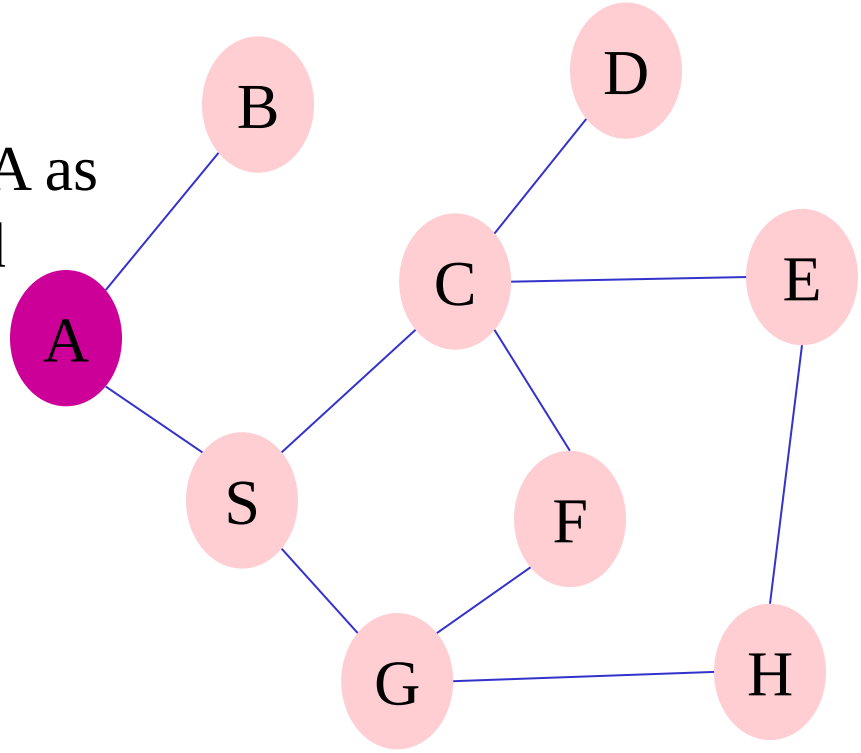


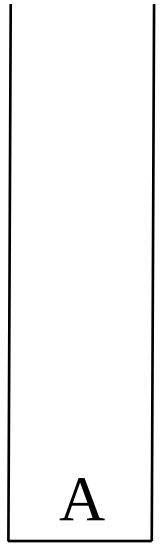


Output: A

DFS using Stacks

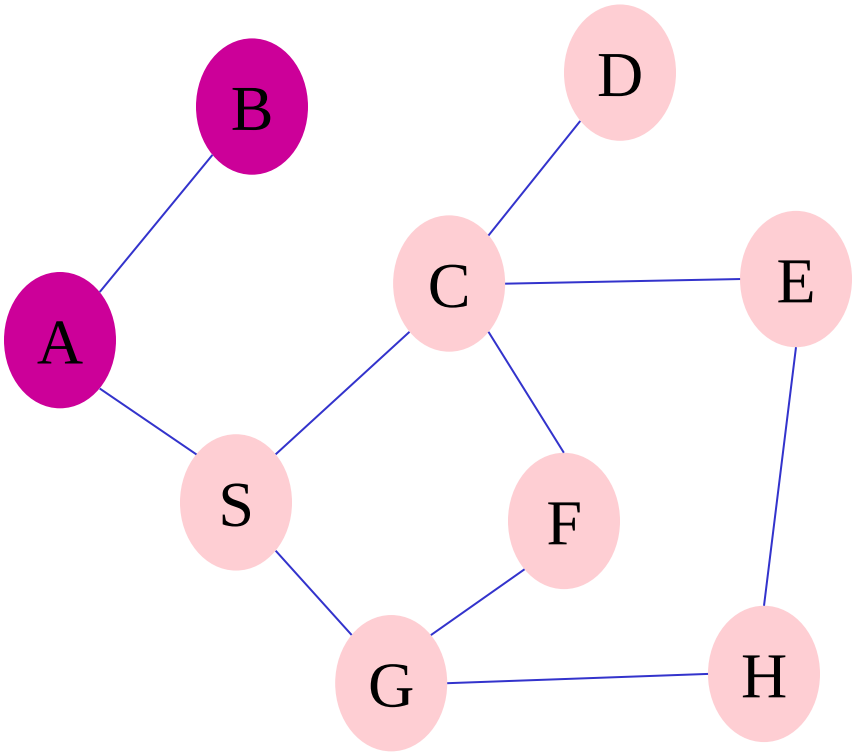
Mark A as visited

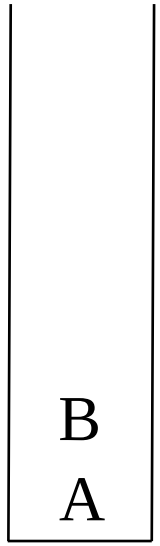




Output: A

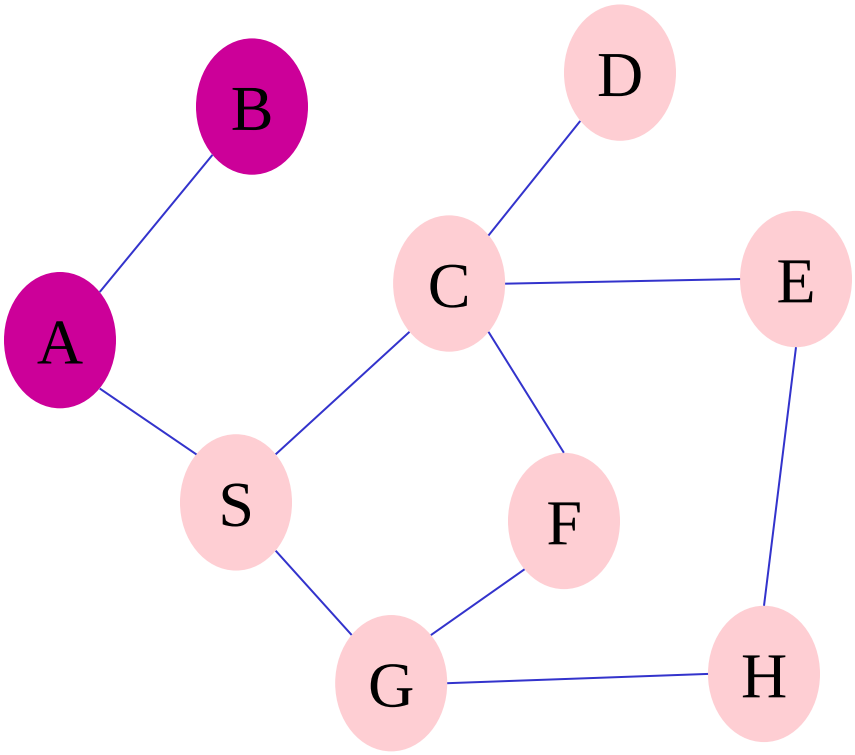
DFS using Stacks

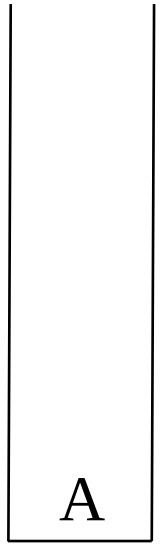




Output: A B

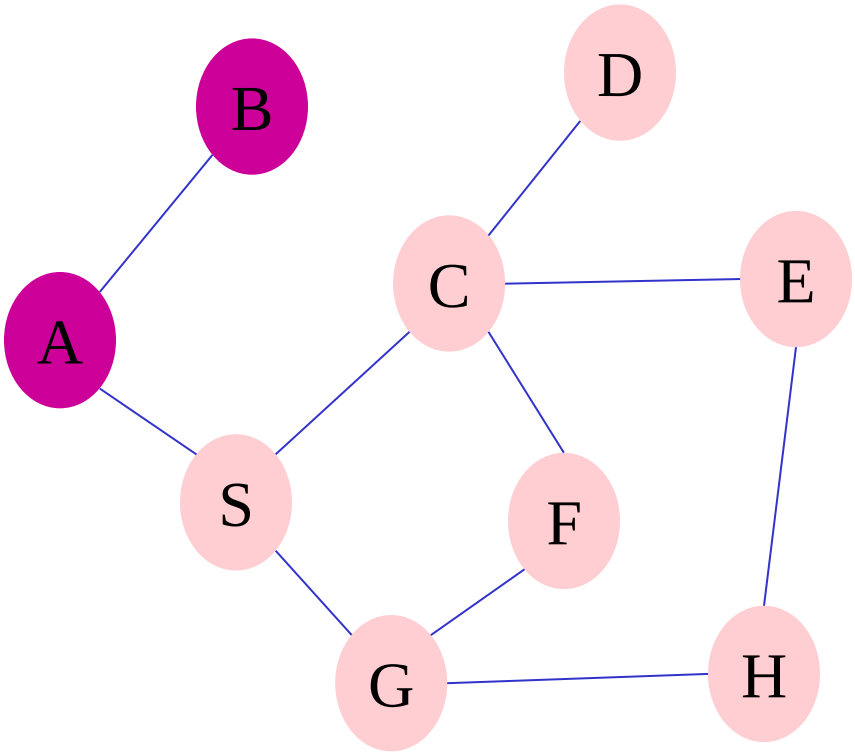
DFS using Stacks

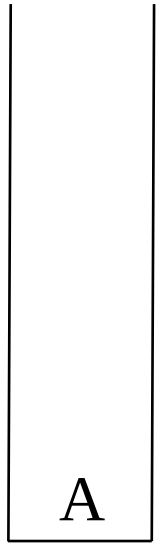




Output: A B S

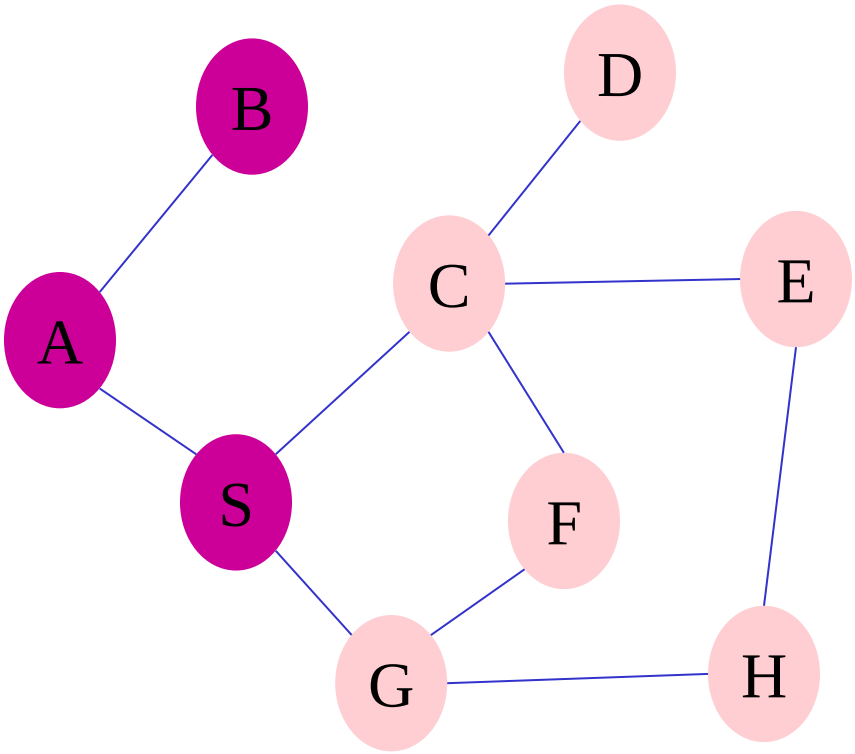
DFS using Stacks

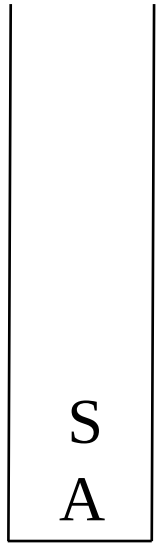




Output: A B S

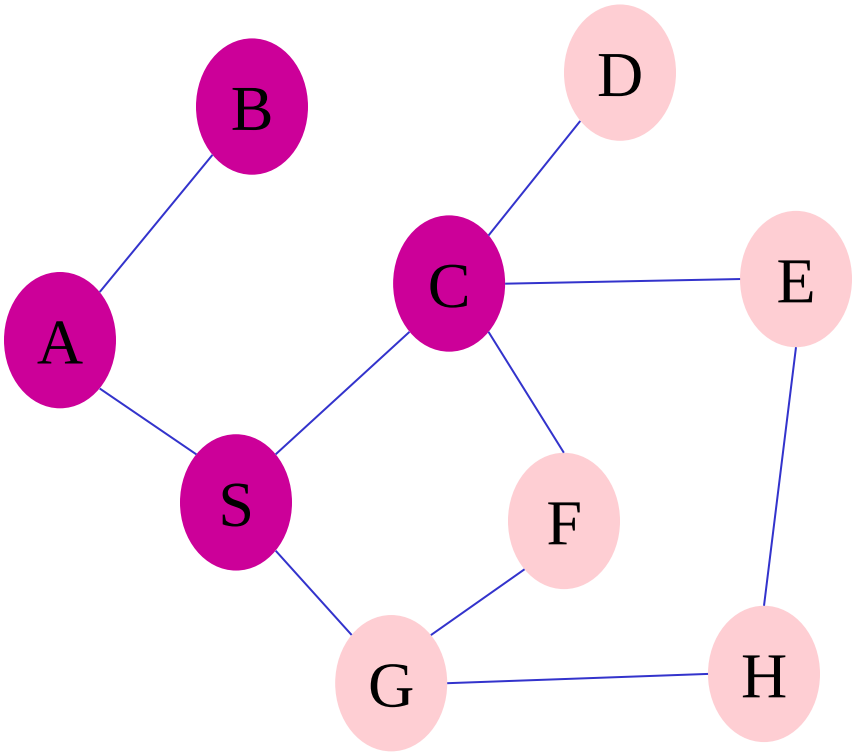
DFS using Stacks

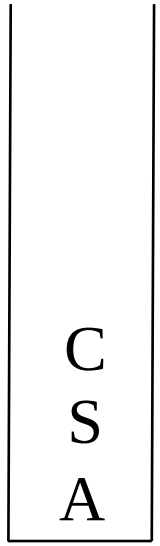




Output: A B S

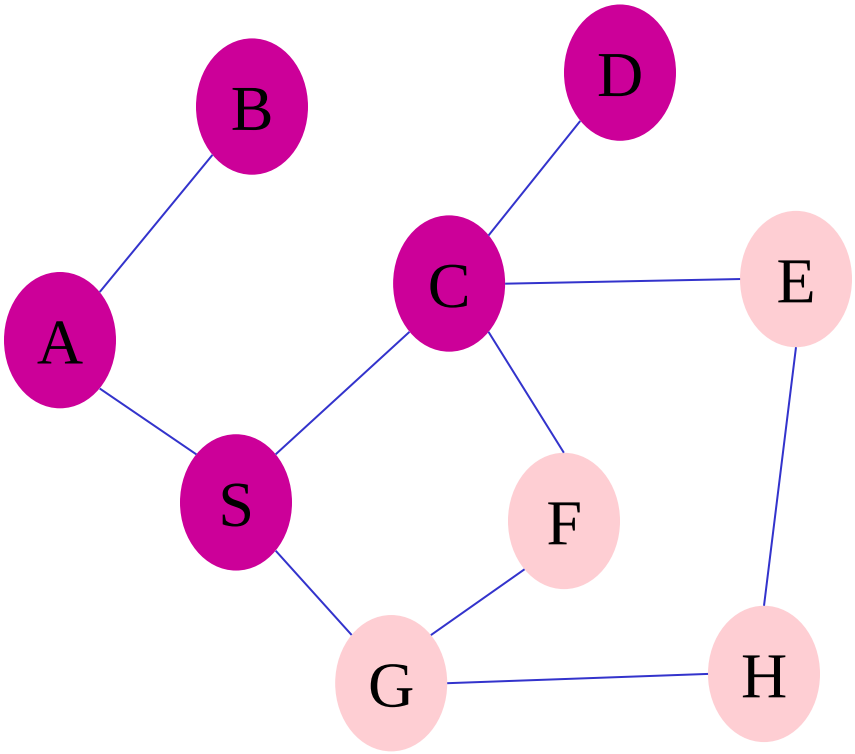
DFS using Stacks

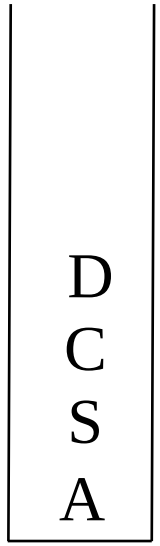




Output: A B S C

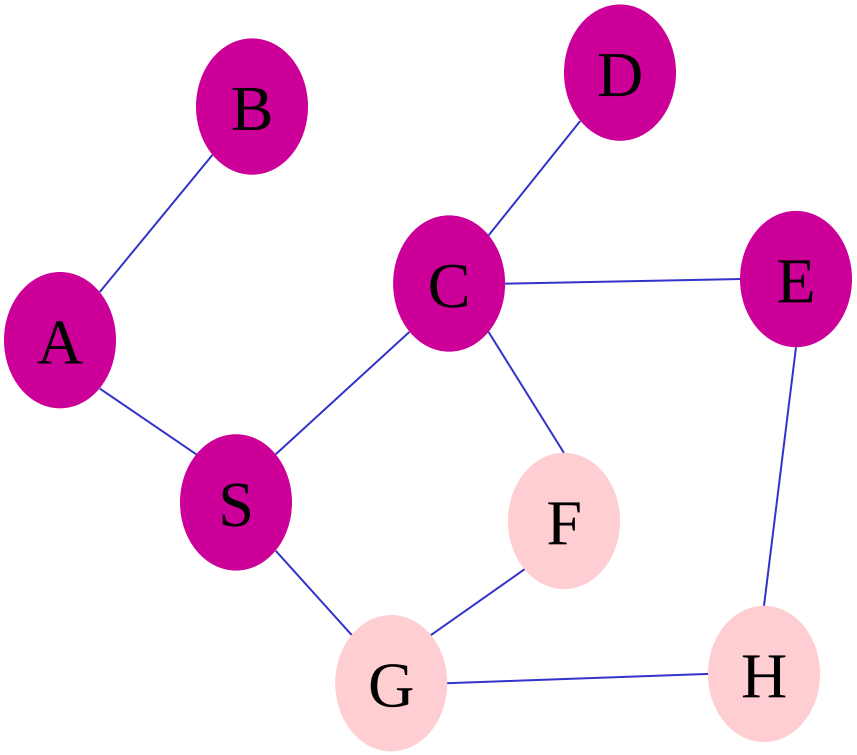
DFS using Stacks

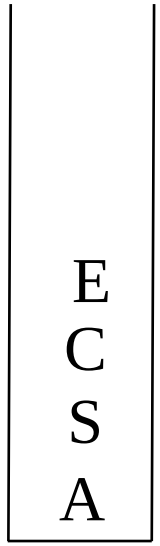




Output: A B S C D

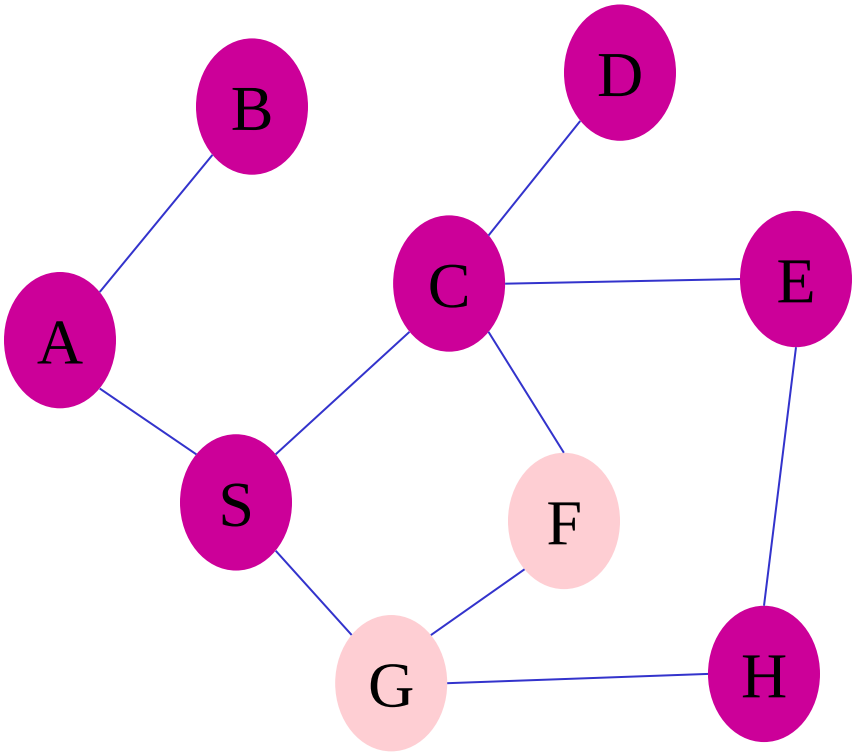
DFS using Stacks

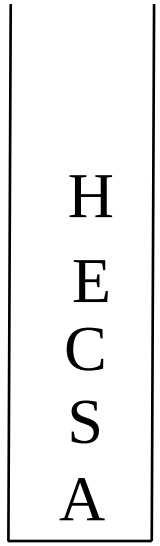




Output: A B S C D E

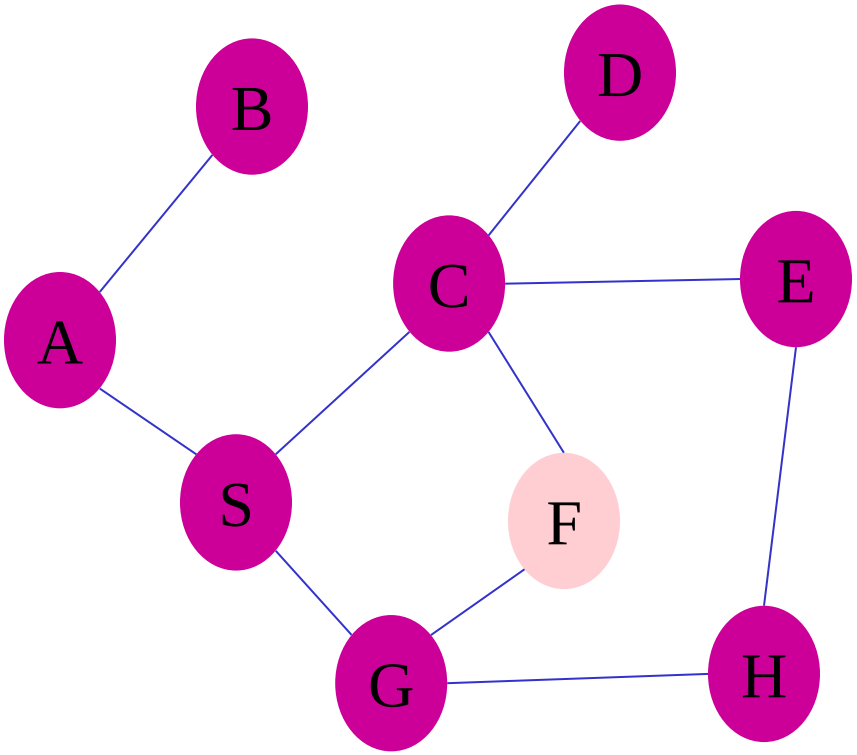
DFS using Stacks

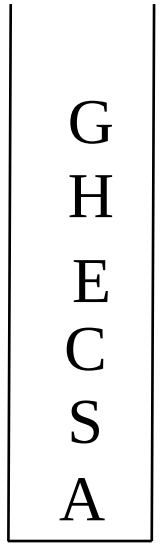




Output: A B S C D E H

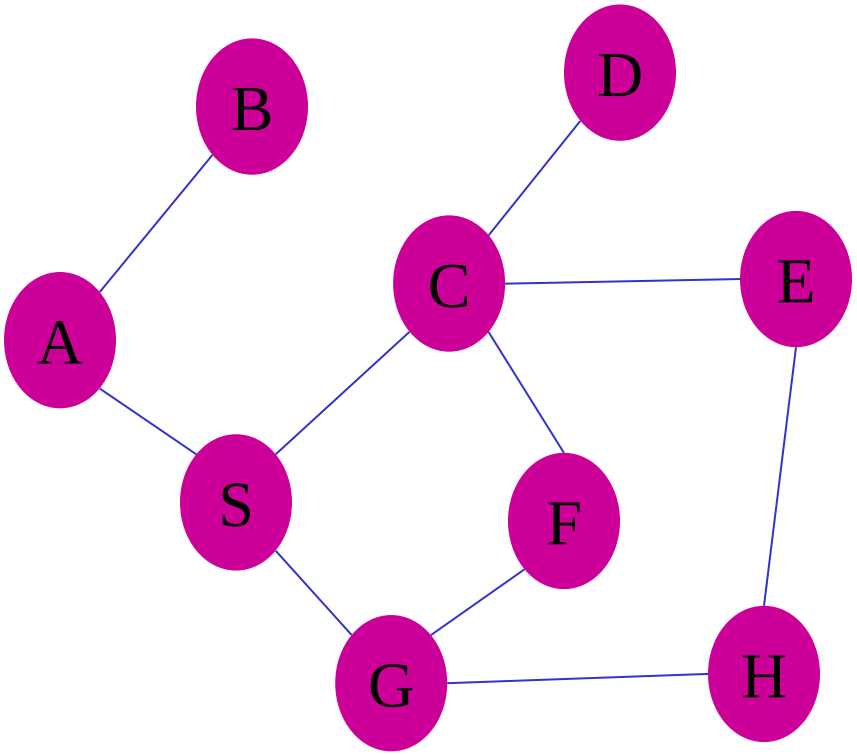
DFS using Stacks

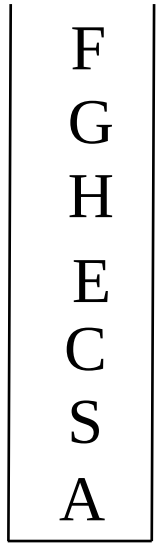




Output: A B S C D E H G

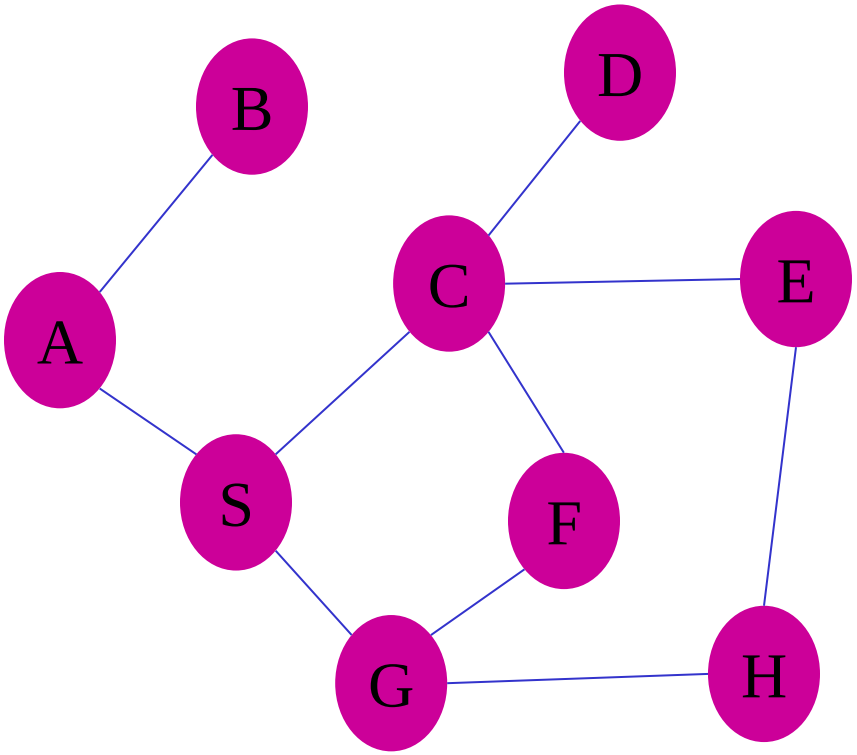
DFS using Stacks



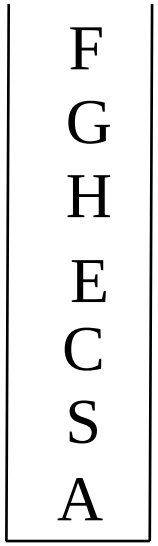


Output: A B S C D E H G F

DFS using Stacks

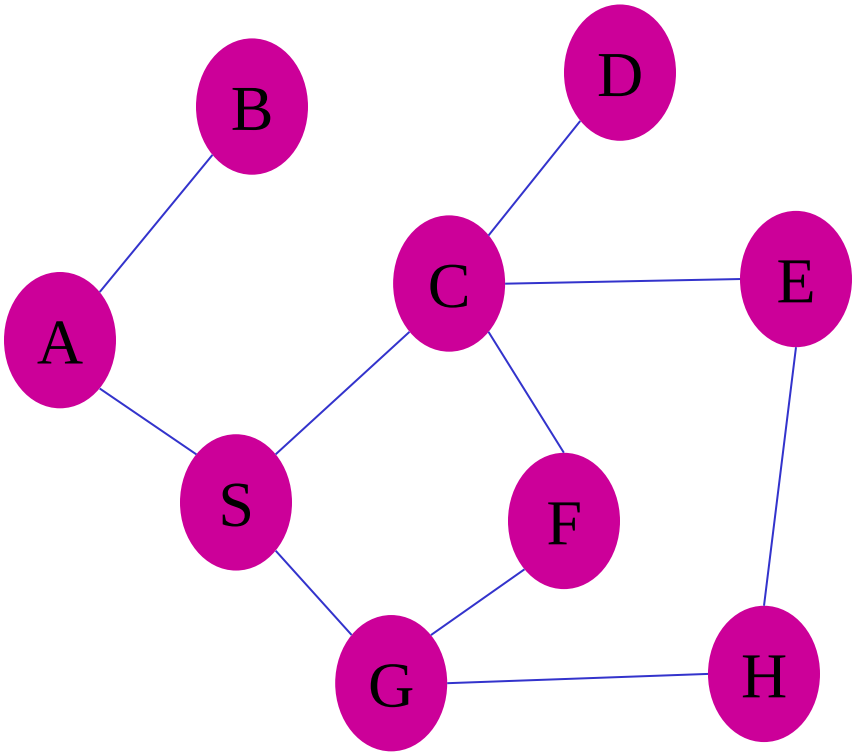


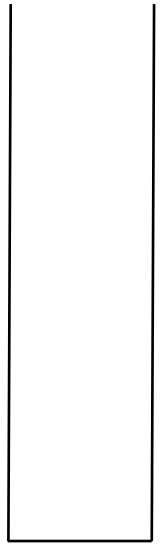
Pop every node with no unvisited child



Output: A B S C D E H G F

DFS using Stacks





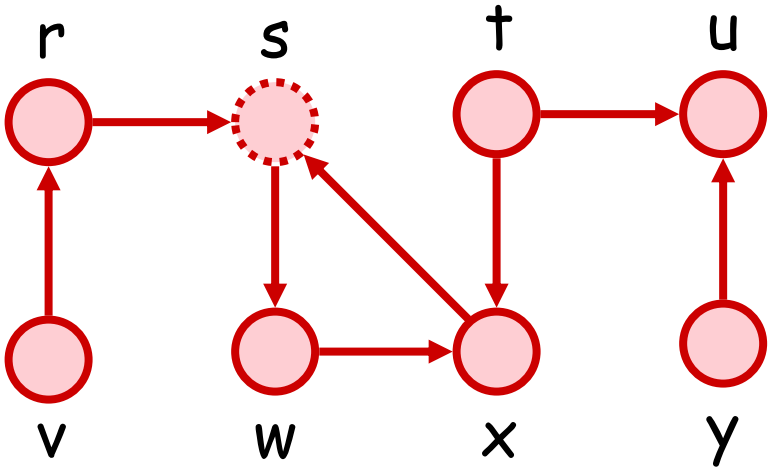
Output: A B S C D E H G F

Generalization

- Just like BFS, DFS may not visit all the vertices of the input graph G, because :
  - G may be disconnected
  - G may be directed, and there is no directed path from s to some vertex
- In most application of DFS (as a subroutine) , once DFS tree of s is obtained, we will continue to apply DFS algorithm on any unvisited vertices …

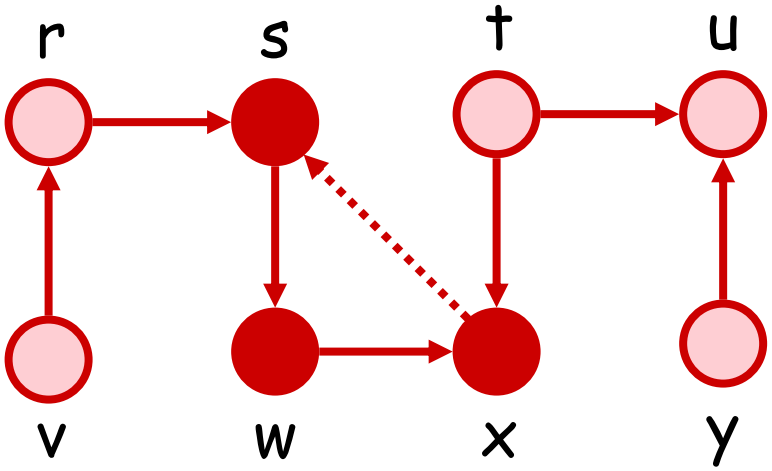
Generalization (Example)

Suppose the input graph is directed



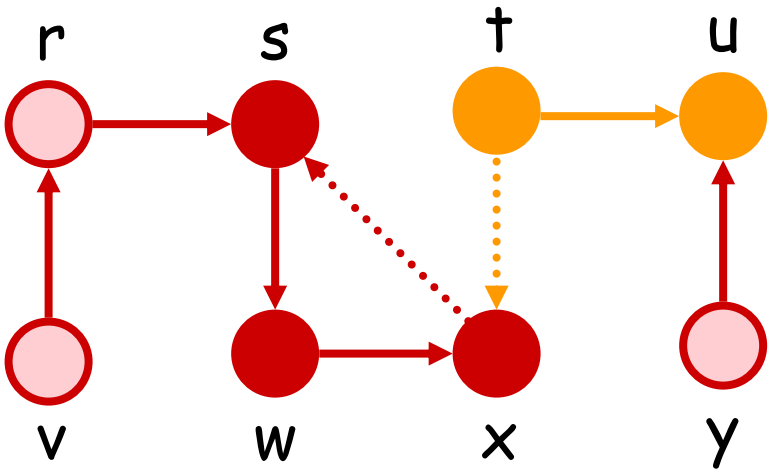
Generalization (Example)

1. After applying DFS on s



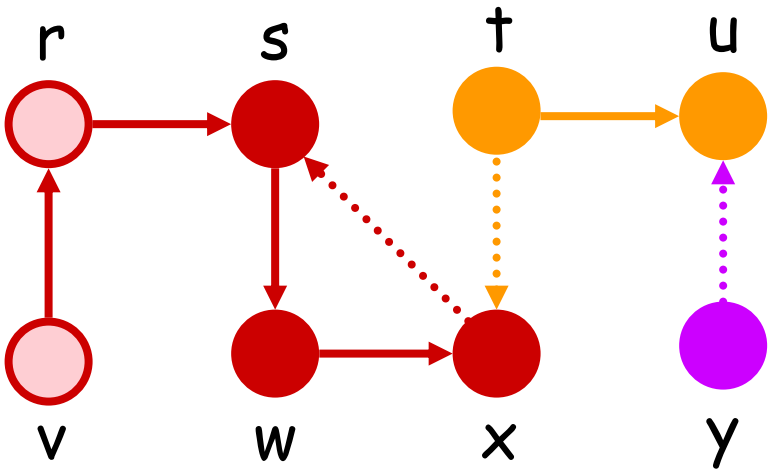
Generalization (Example)

2. Then, after applying DFS on t



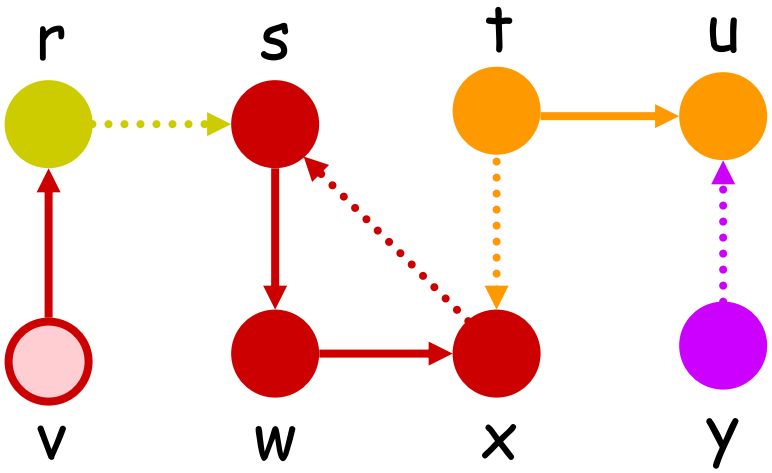
Generalization (Example)

3. Then, after applying DFS on y



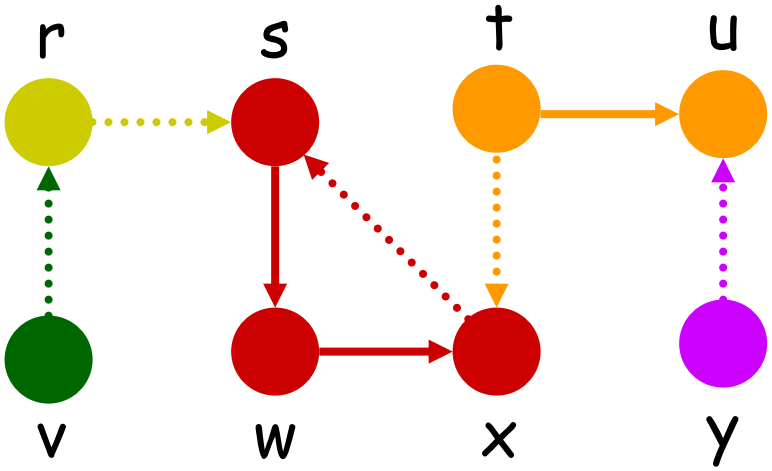
Generalization (Example)

4. Then, after applying DFS on r



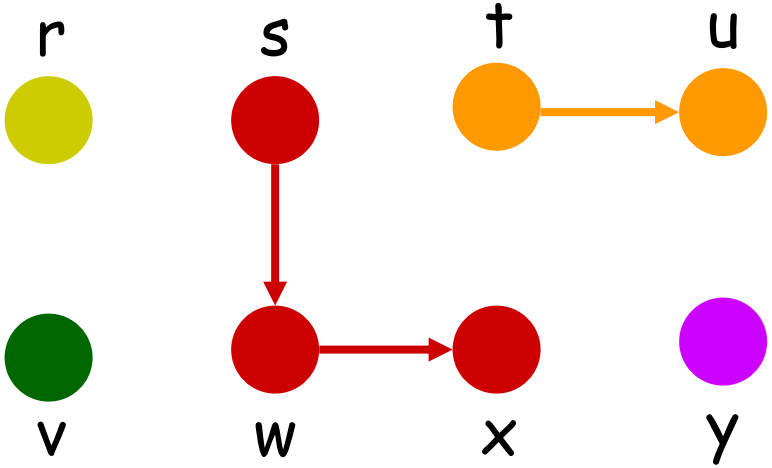
Generalization (Example)

5. Then, after applying DFS on v



Generalization (Example)

Result : a collection of rooted trees
called DFS forest



Performance

- Since no vertex is discovered twice, and each edge is visited at most twice (why?)

  ➔ Total time: $O(|V|+|E|)$

- As mentioned, apart from recursion, we can also perform DFS using a LIFO stack (Do you know how?)