

**Depth-first search** goes through the tree branch by branch, going all the way down to the leaf nodes at the bottom of the tree before trying the next branch over. This strategy requires much less memory than breadth-first search, since it only needs to store a single path from the root of the tree down to the leaf node. However, it is potentially incomplete, since it will keep going on down one branch until it finds a dead-end, and it is not optimal. Use stack instead of queue.

Example: Starting a vertex, go to as far as possible. If there is no further way to go, come back. Repeat above until all vertices have been visited.

## Depth-first search Algorithm (page 541)

DFS( $G$ )

```

1  for each vertex  $u \in V[G]$ 
2      do  $color[u] \leftarrow WHITE$ 
3          $\pi[u] \leftarrow NIL$ 
4   $time \leftarrow 0$ 
5  for each vertex  $u \in V[G]$ 
6      do if  $color[u] = WHITE$ 
7         then DFS-VISIT( $u$ )

```

$d[]$ : discovery time.  
 $f[]$ : finishing time  
 $\pi[]$ : parent information.

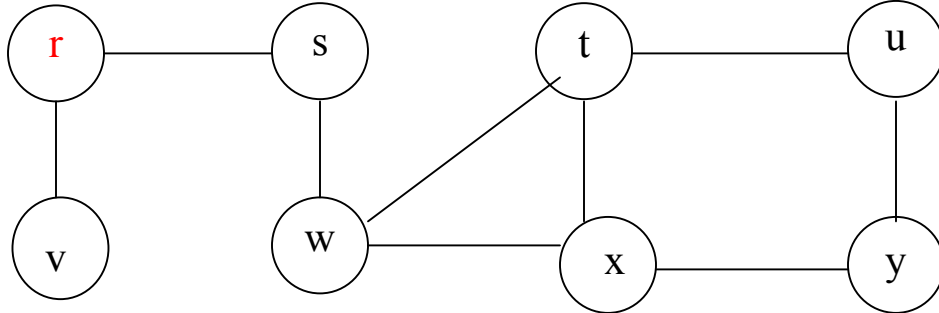
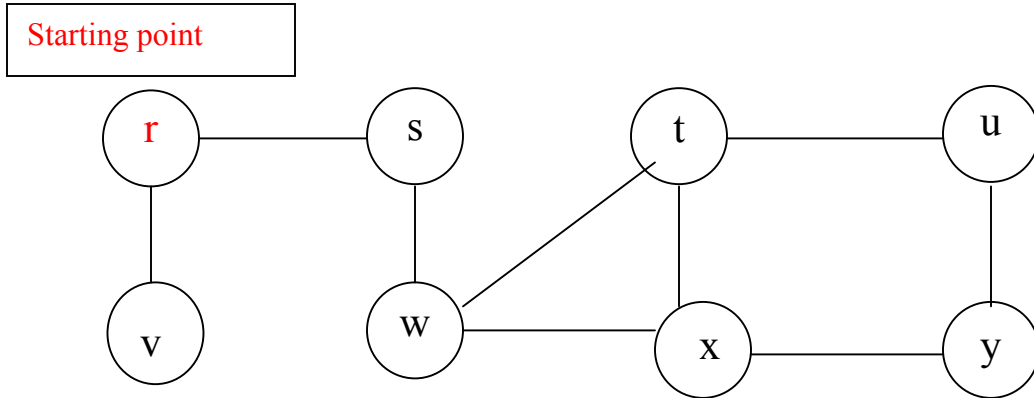
DFS-VISIT( $u$ )

```

1   $color[u] \leftarrow GRAY$       ▷ White vertex  $u$  has just been discovered.
2   $time \leftarrow time + 1$ 
3   $d[u] \leftarrow time$ 
4  for each  $v \in Adj[u]$       ▷ Explore edge  $(u, v)$ .
5      do if  $color[v] = WHITE$ 
6         then  $\pi[v] \leftarrow u$ 
7             DFS-VISIT( $v$ )
8   $color[u] \leftarrow BLACK$     ▷ Blacken  $u$ ; it is finished.
9   $f[u] \leftarrow time \leftarrow time + 1$ 

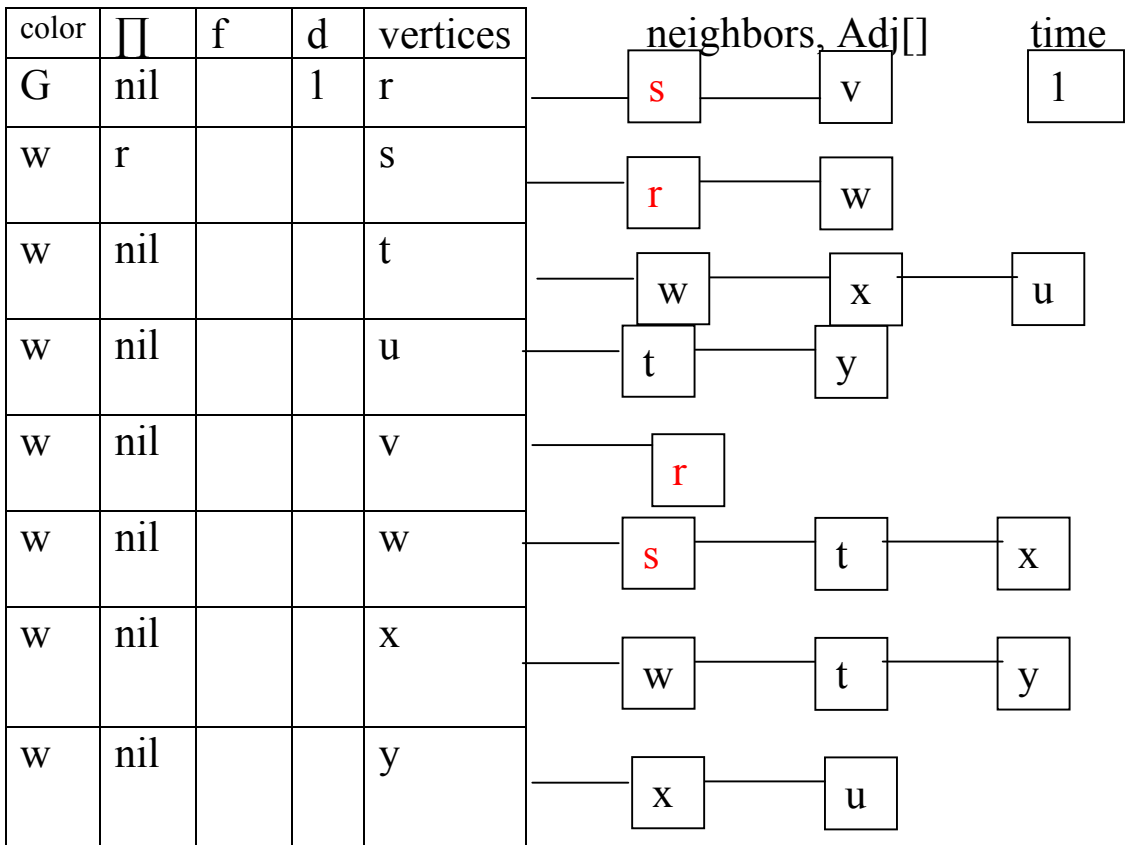
```

Undirected Graph:



color	$\Pi$	f	d	vertices	neighbors	time
w	nil			r	s — v	0
w	nil			s	r — w	
w	nil			t	w — x — u	
w	nil			u	t — y	
w	nil			v	r	
w	nil			w	s — t — x	
w	nil			x	w — t — y	
w	nil			y	x — u	

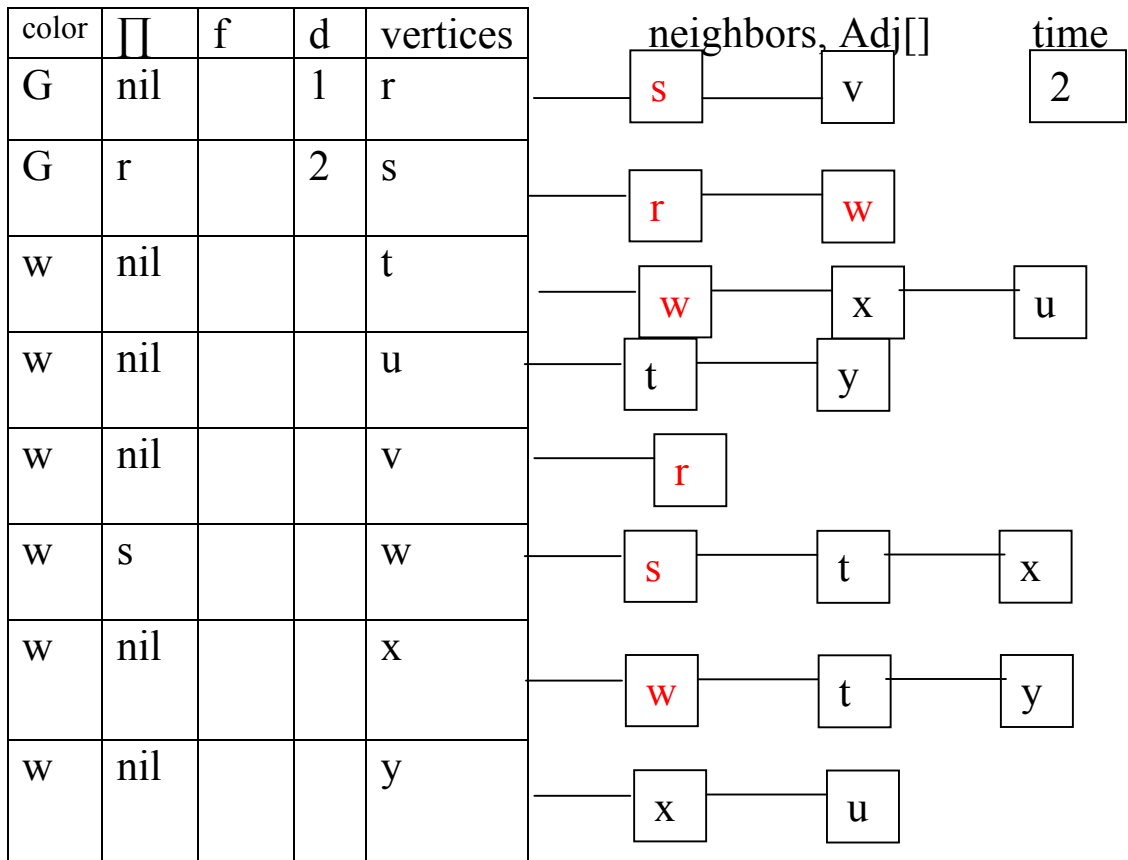
Note: Do lines 1 to 4 of DFS(G)



Note: Do lines 5 to 7 of DFS(G), and call DFS-VISIT(r).  
 Do lines 1 to 6 of DFS-VISIT(r)

DFS-VISIT(r)

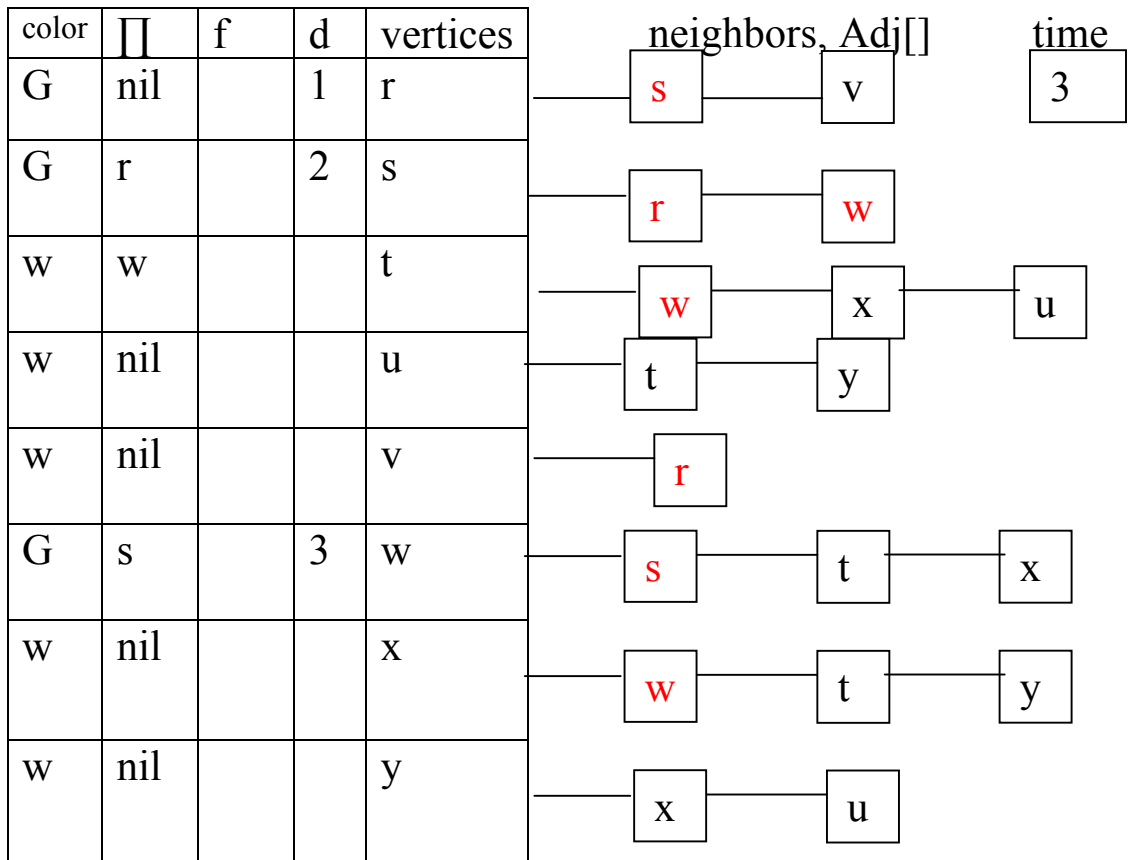
Stack



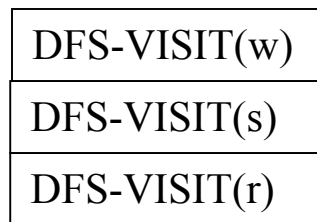
Note: Do line 7 of DFS-VISIT(r) that is calling DFS-VISIT(s).  
 Do lines 1 to 6 of DFS-VISIT(s).



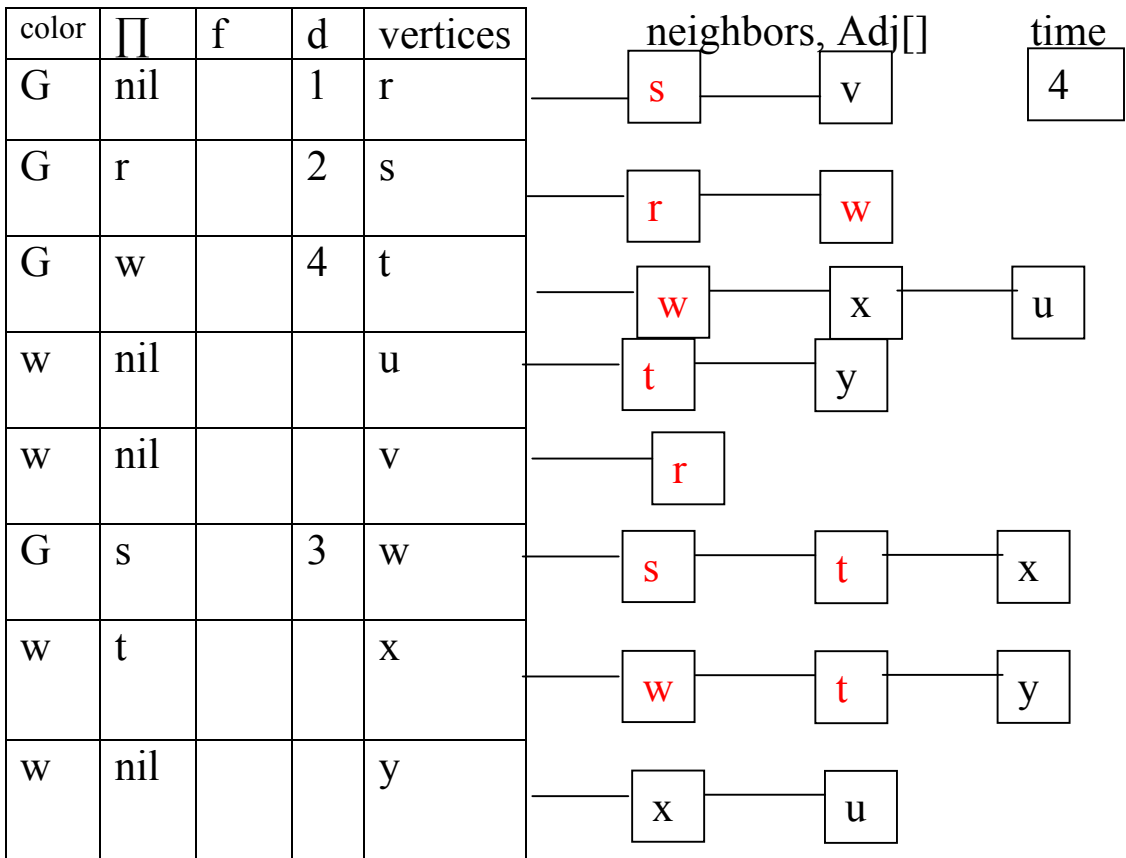
Stack



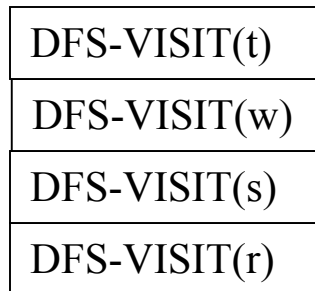
Note: Do line 7 of DFS-VISIT(s) that is calling DFS-VISIT(w).  
 Do lines 1 to 6 of DFS-VISIT(w).



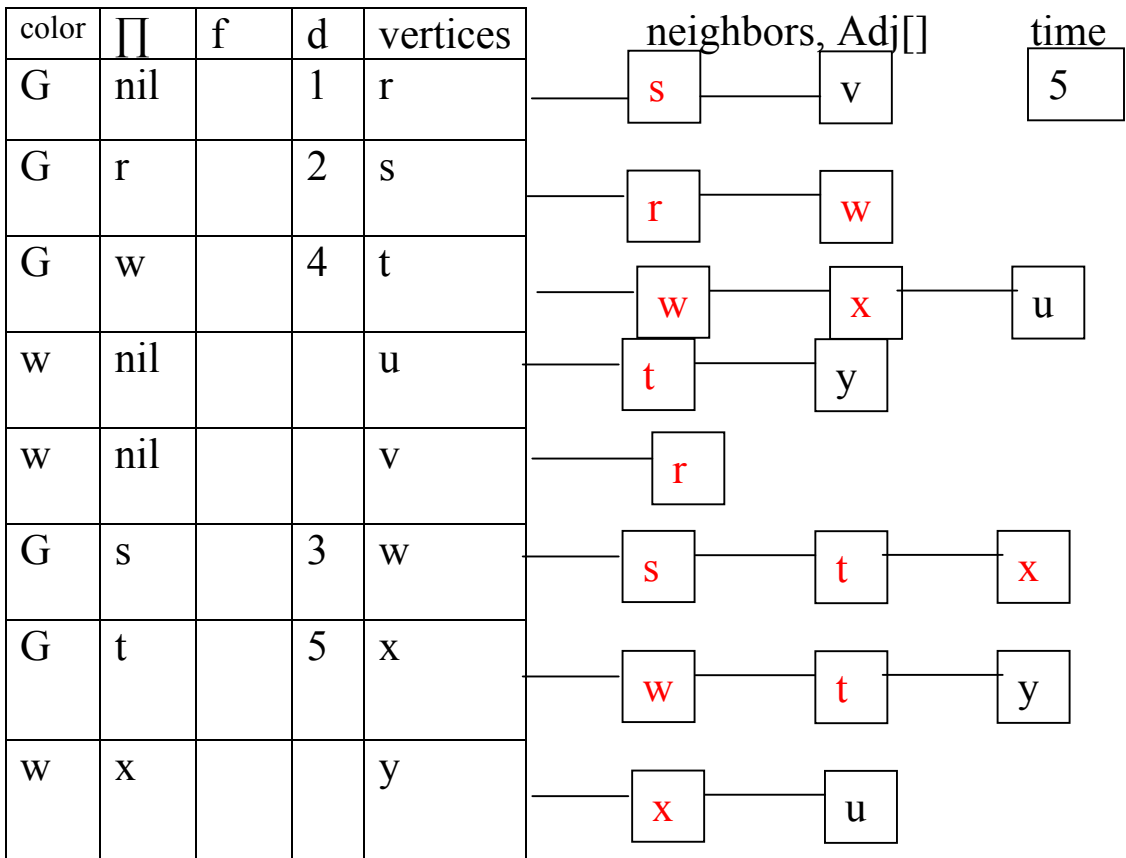
Stack



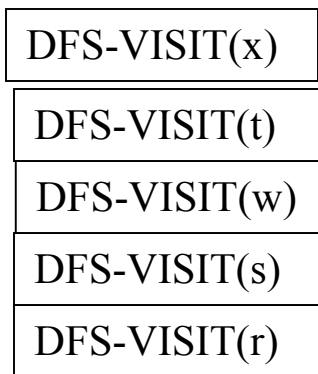
Note: Do line 7 of DFS-VISIT(w) that is calling DFS-VISIT(t).  
 Do lines 1 to 6 of DFS-VISIT(t).



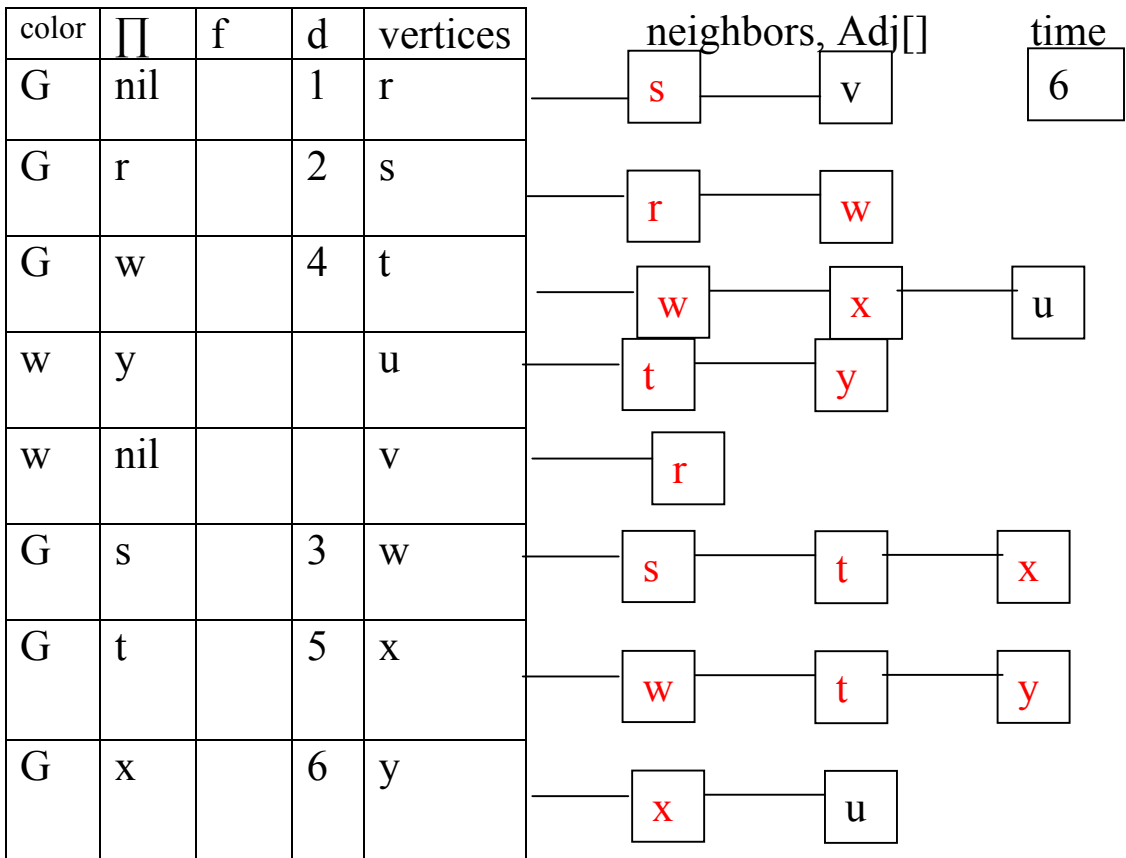
Stack



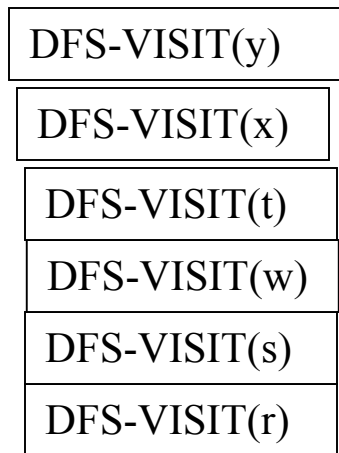
Note: Do line 7 of DFS-VISIT(t) that is calling DFS-VISIT(x).  
 Do lines 1 to 6 of DFS-VISIT(x).



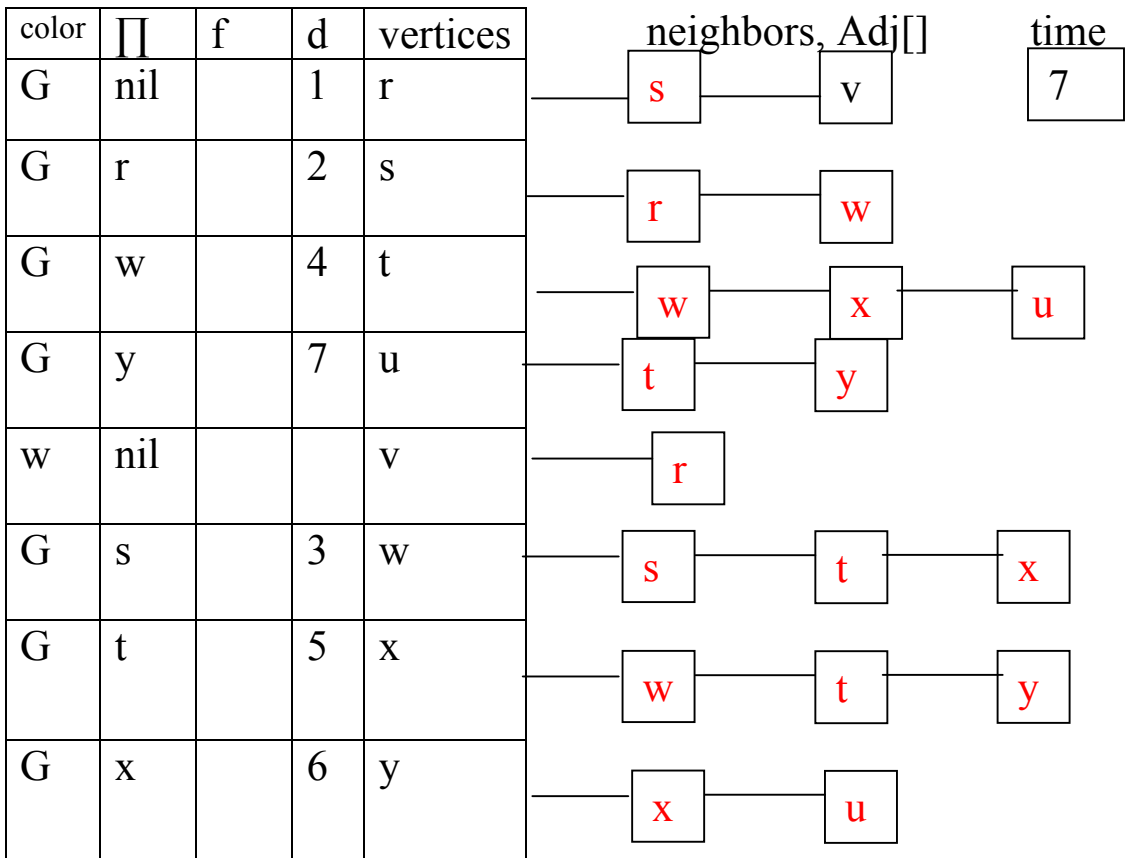
Stack



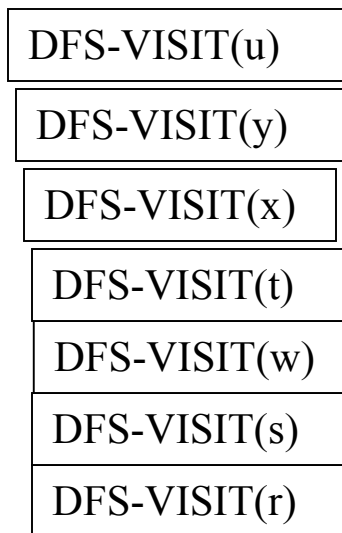
Note: Do line 7 of DFS-VISIT(x) that is calling DFS-VISIT(y).  
 Do lines 1 to 6 of DFS-VISIT(y).



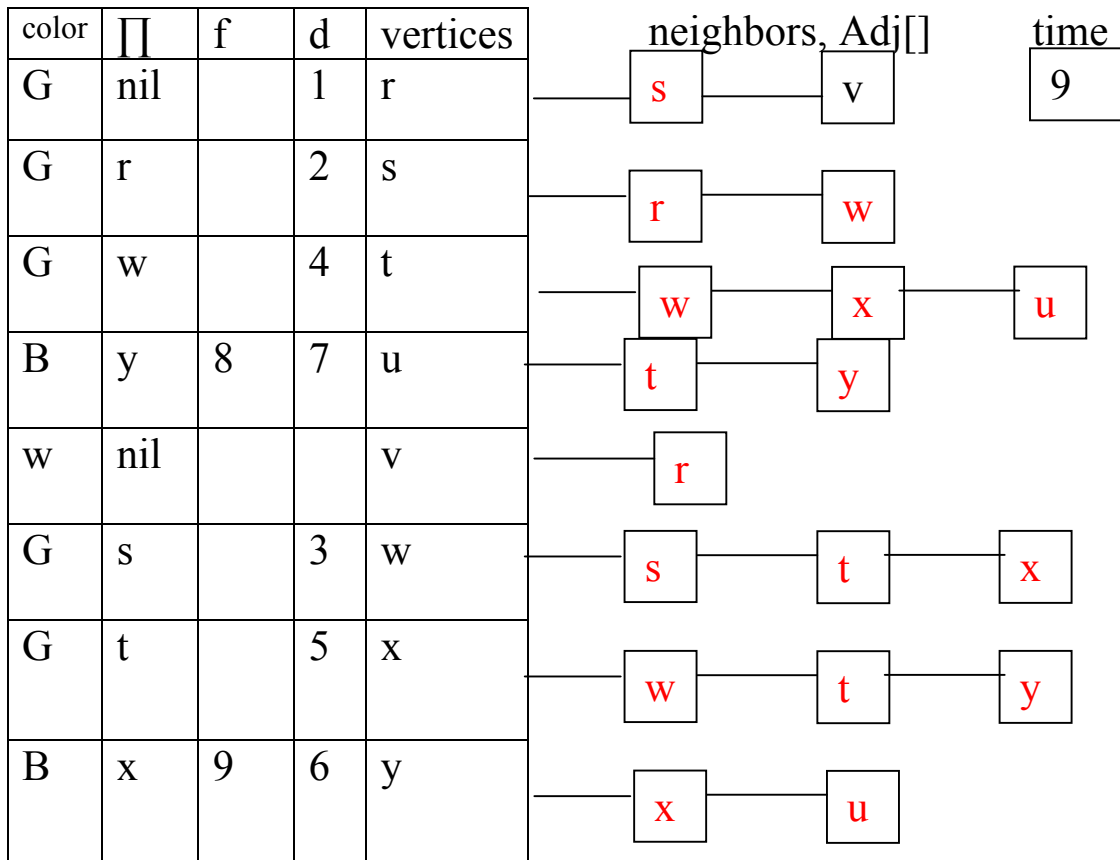
Stack



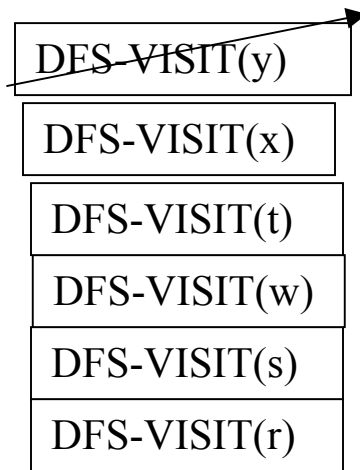
Note: Do line 7 of DFS-VISIT(y) that is calling DFS-VISIT(u).  
 Do lines 1 to 6 of DFS-VISIT(u).



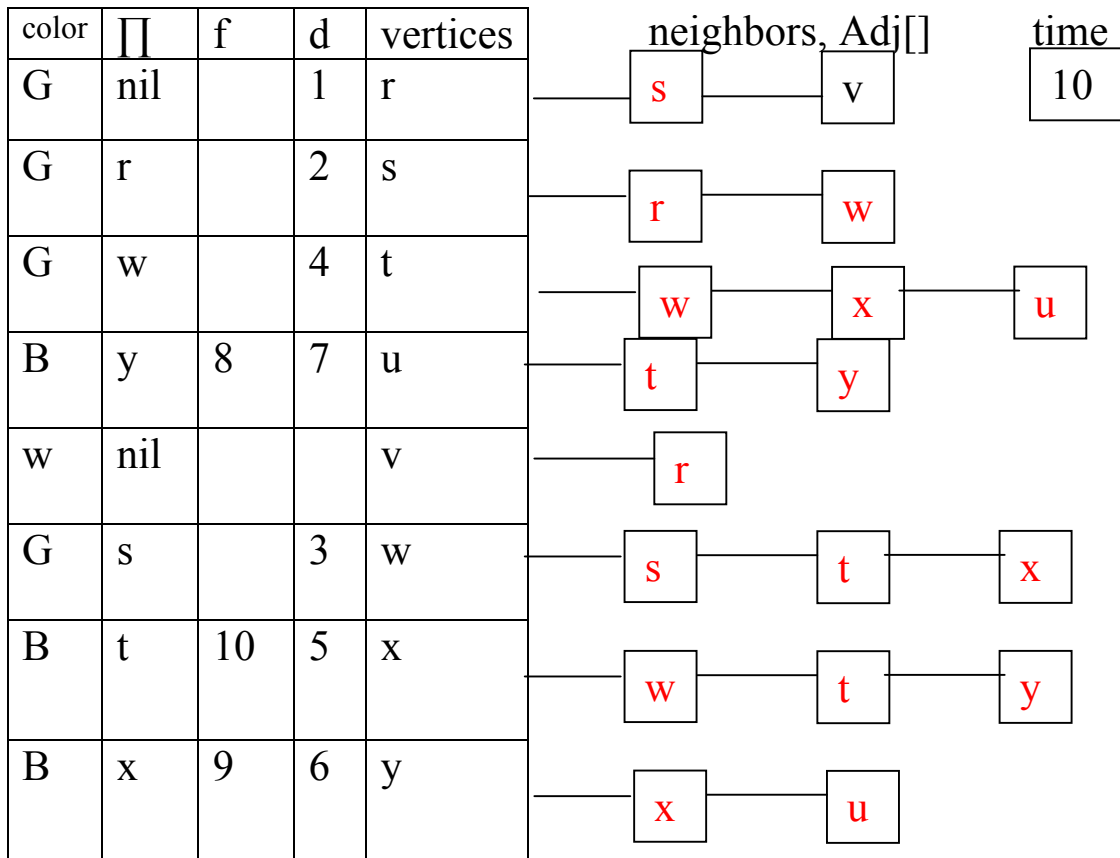
Stack



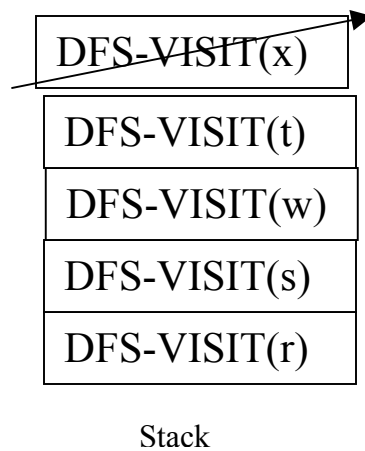
Note: Do lines 8 & 9 of DFS-VISIT(y) and pop DFS-VISIT(u).

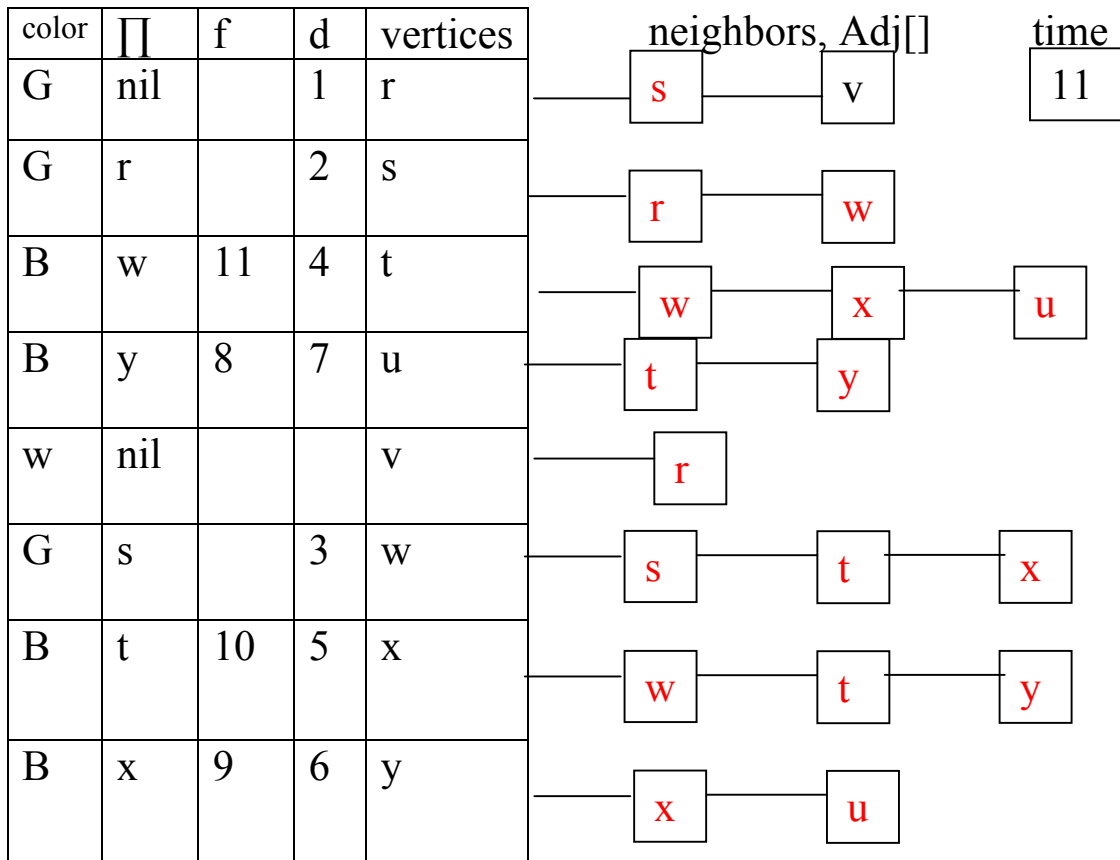


Stack

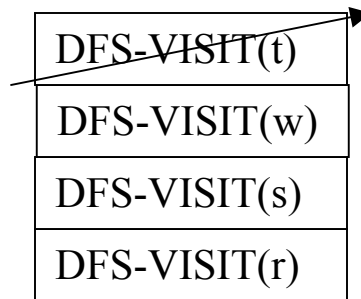


Note: Do lines 8 & 9 of DFS-VISIT(x) and pop DFS-VISIT(x).

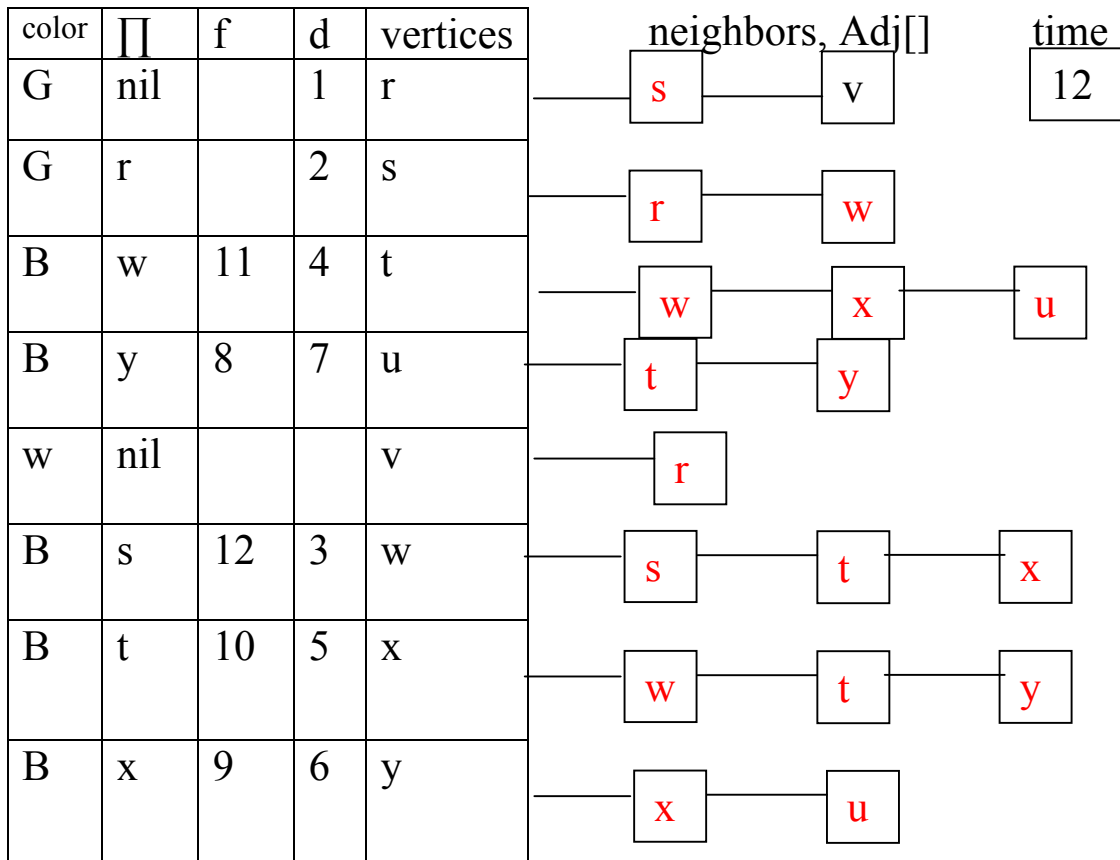




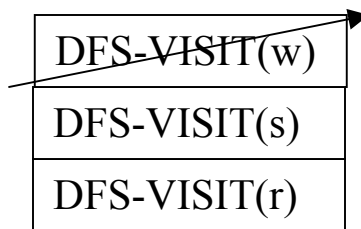
Note: Do lines 8 & 9 of DFS-VISIT(t) and pop DFS-VISIT(t).



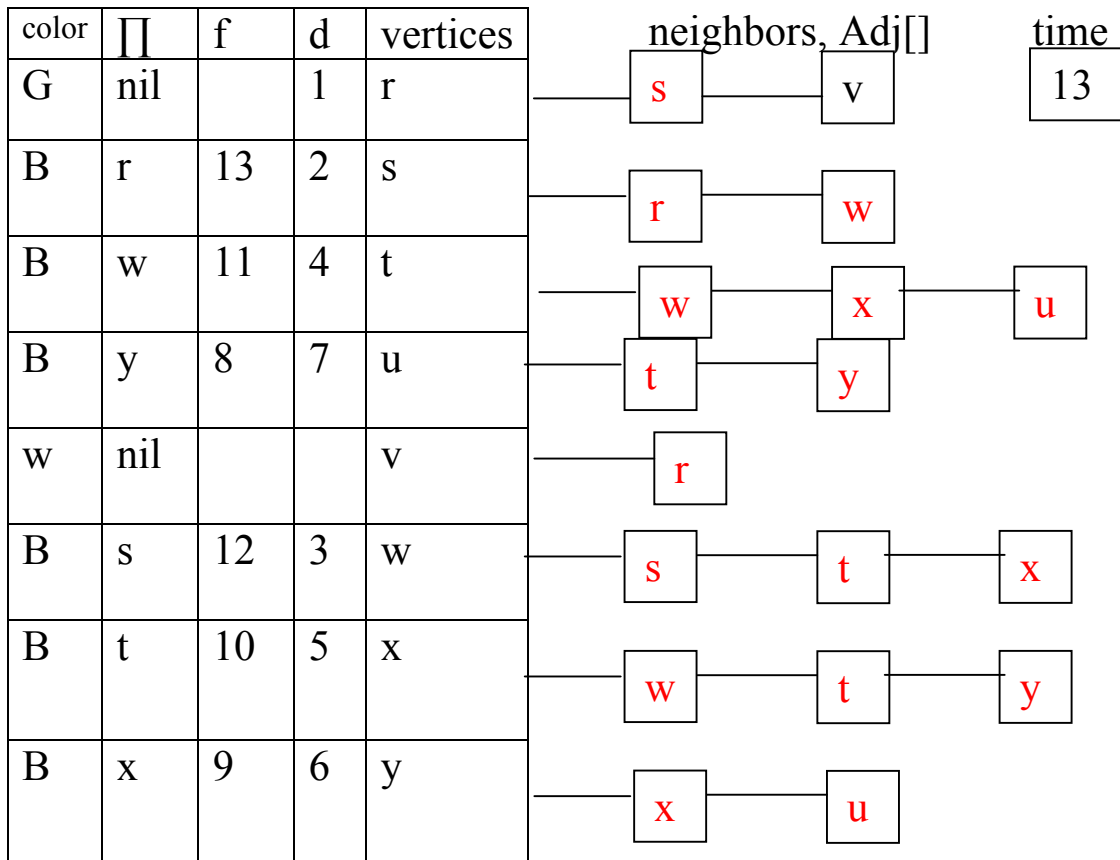
Stack



Note: Do lines 8 & 9 of DFS-VISIT(w) and pop DFS-VISIT(w).



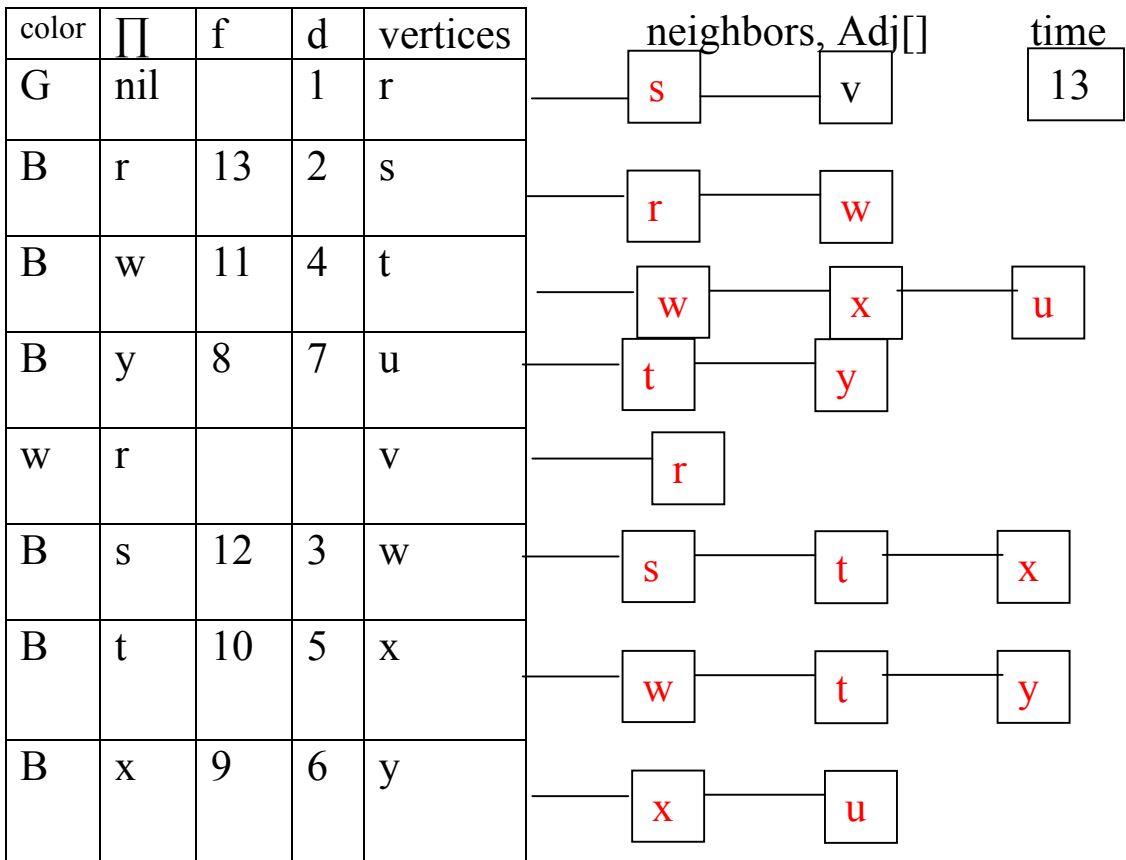
Stack



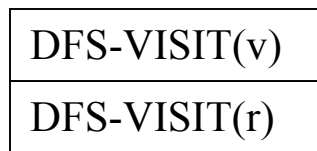
Note: Do lines 8 & 9 of DFS-VISIT(s) and pop DFS-VISIT(s).



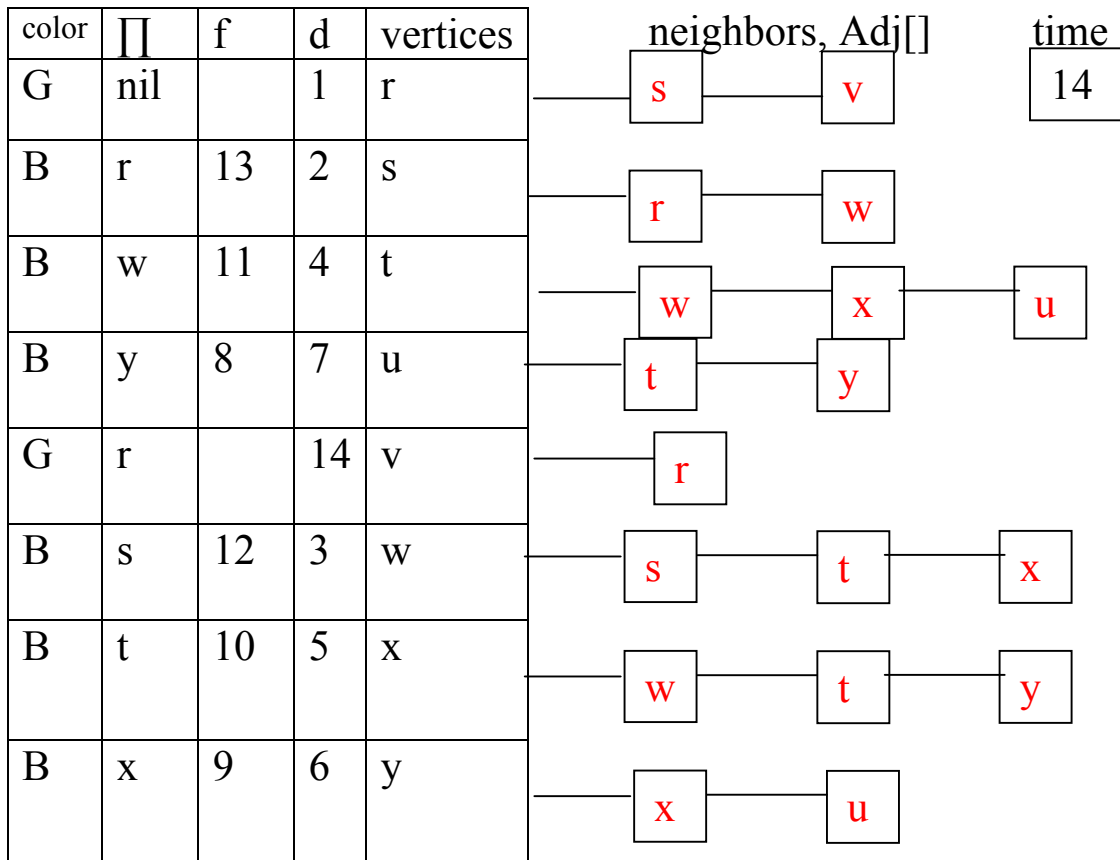
Stack



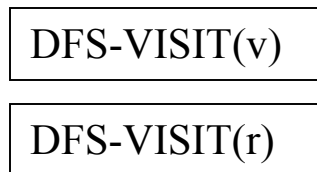
Note: Do lines 4 to 7 of DFS-VISIT(r) and call DFS-VISIT(v)



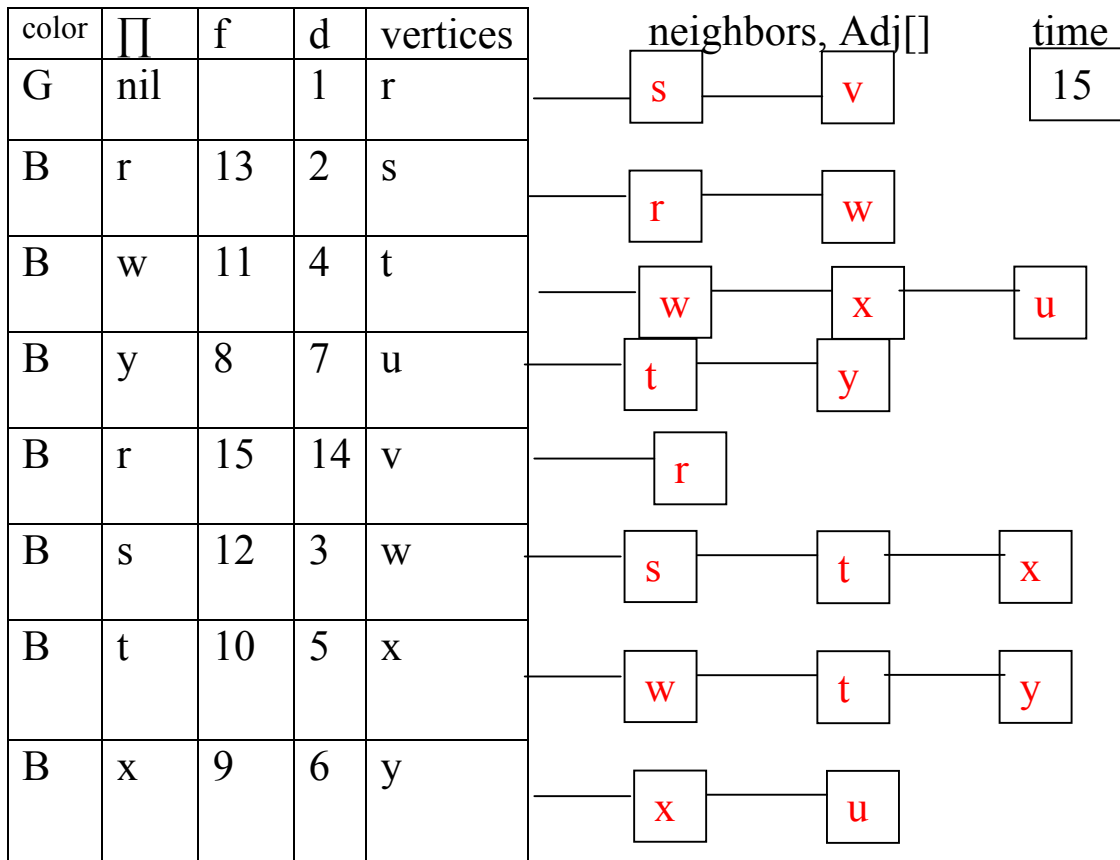
Stack



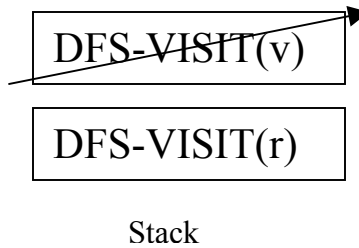
Note: Do lines 1 to 7 of DFS-VISIT(v)

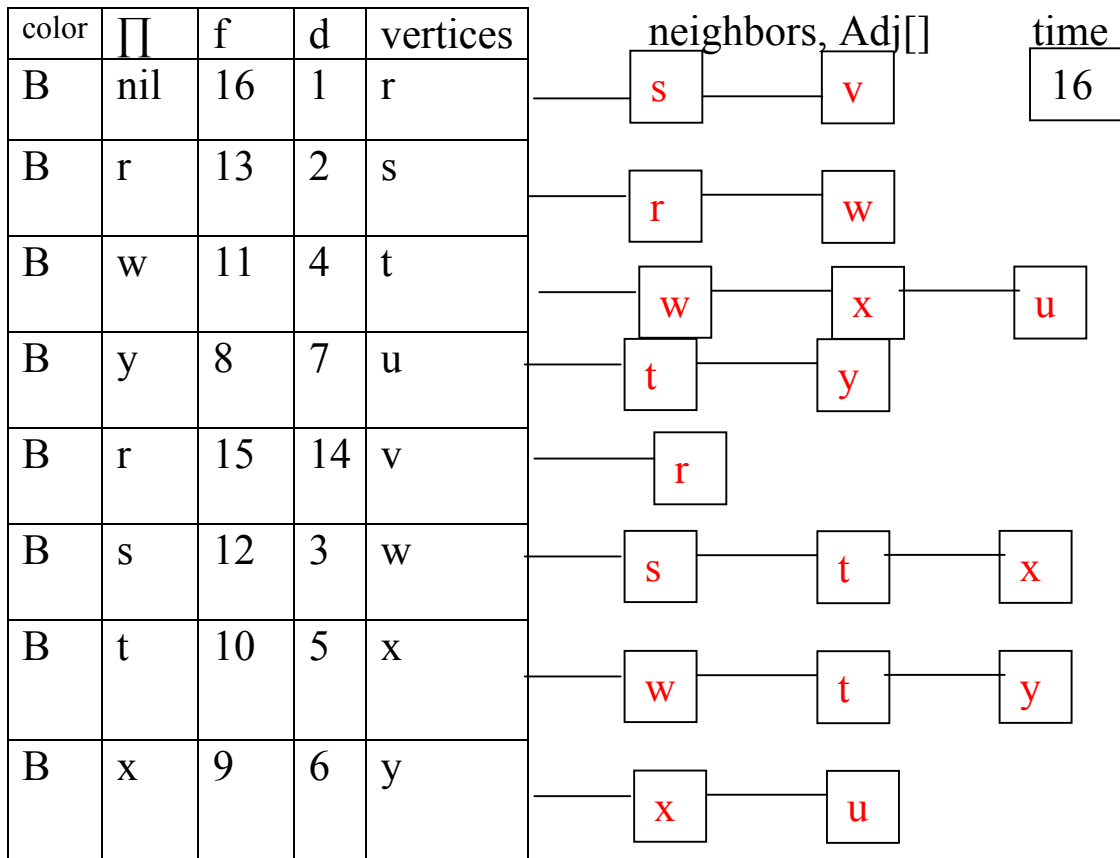


Stack



Note: Do lines 8 & 9 of DFS-VISIT(v) and pop DFS-VISIT(v).





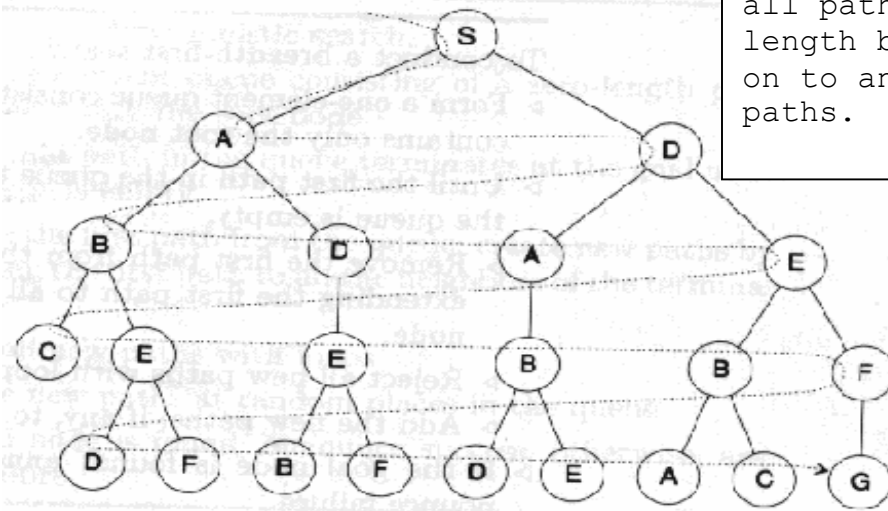
Note: Do lines 8 & 9 of DFS-VISIT(r) and pop DFS-VISIT(r).



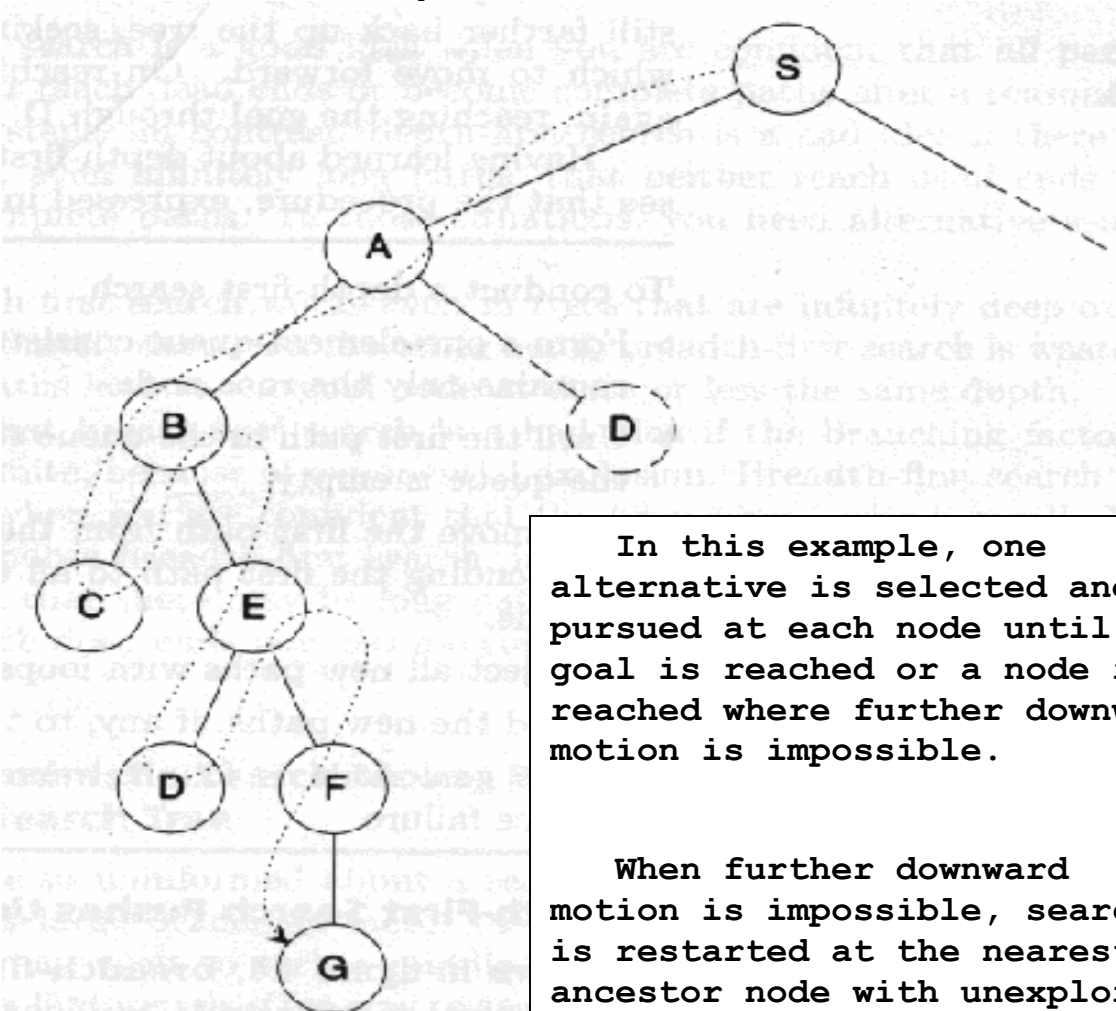
Stack

BFS: search level by level.

As shown in the example, *breadth-first search* checks all paths of a given length before moving on to any longer paths.



DFS: search branch by branch.



In this example, one alternative is selected and pursued at each node until the goal is reached or a node is reached where further downward motion is impossible.

When further downward motion is impossible, search is restarted at the nearest ancestor node with unexplored children.