DFS:

- The strategy followed by depth-first search is to search "deeper" in the graph whenever possible.
- In depth-first search, edges are explored out of the most recently discovered vertex v that still has unexplored edges leaving it. When all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered. This process continues until we have discovered all the vertices that are reachable from the original source vertex.
- If any undiscovered vertices remain, then one of them is selected as a new source and the search is repeated from that source. This entire process is repeated until **all vertices are** discovered.
- As in breadth-first search, whenever a vertex v is discovered during a scan of the adjacency list of an already discovered vertex u, depth-first search records this event by setting v's predecessor field $\pi[v]$ to u.

- Unlike breadth-first search, whose predecessor subgraph forms a tree, the predecessor subgraph produced by a depth-first search may be composed of several trees, because the search may be repeated from multiple sources.
- The predecessor subgraph of a depth-first search is therefore defined slightly differently from that of a breadth-first search:

  $G\pi = (V, E\pi)$, where

  $E\pi = \{(\pi[v], v) : v \in V \text{ and } \pi[v] \neq NIL\}$.

- As in breadth-first search, vertices are colored during the search to indicate their state. Each vertex is **initially white**, is grayed when it is discovered in the search, and is blackened when it is finished, that is, when its adjacency list has been examined completely.
- Besides creating a depth-first forest, depth-first search also timestamps each vertex. Each vertex v has two timestamps: the first timestamp d[v] records when v is first discovered (and grayed), and the second timestamp f [v] records when the search finishes examining v's adjacency list (and blackens v).
  (These timestamps are used in many graph algorithms and are generally helpful in reasoning about the behavior of depth-first search.)
- These timestamps are integers between 1 and 2 |V|, since there is one discovery event and one finishing event for each of the |V| vertices. For every vertex u,

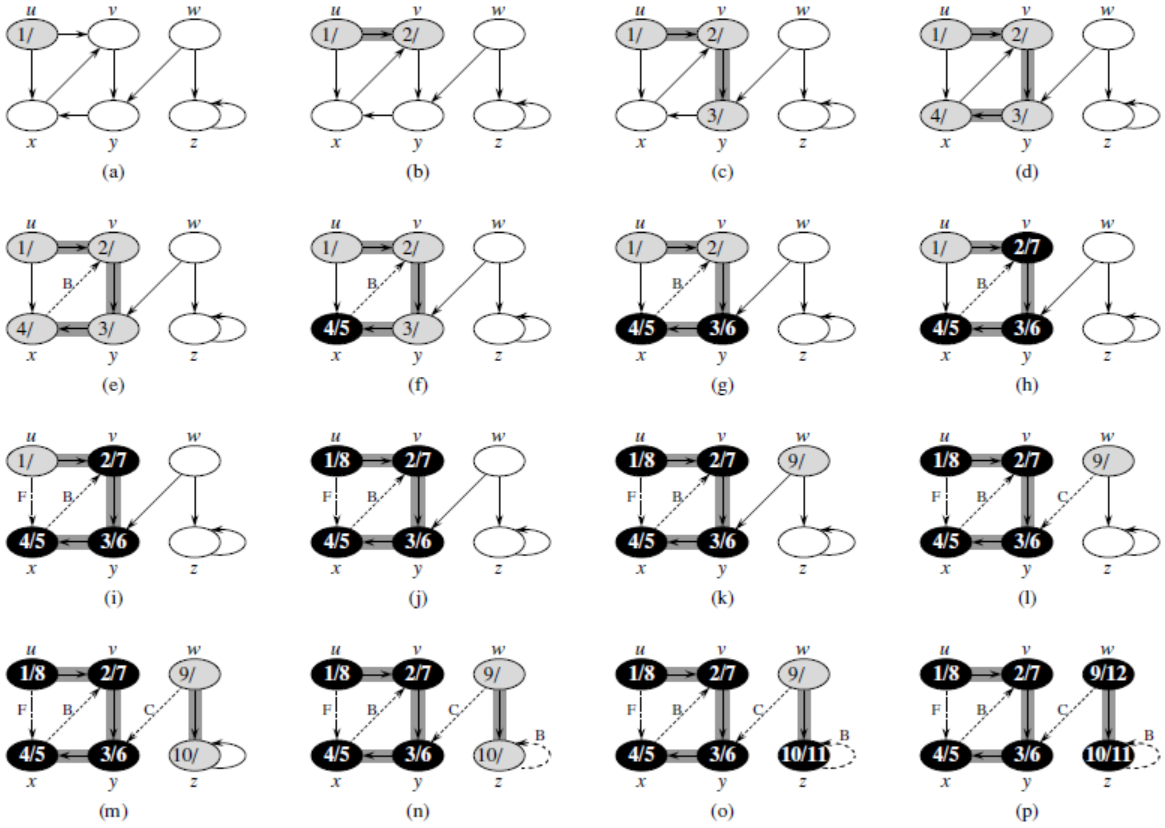
  $$d[u] < f[u].$$

- Vertex u is WHITE before time d[u], GRAY between time d[u] and time f [u], and BLACK thereafter.

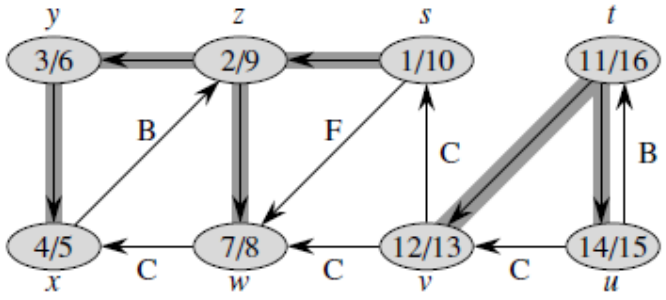
```
DFS(G)
1  for each vertex u ∈ V[G]
2       do color[u] ← WHITE
3          π[u] ← NIL
4  time ← 0
5  for each vertex u ∈ V[G]
6       do if color[u] = WHITE
7             then DFS-VISIT(u)

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

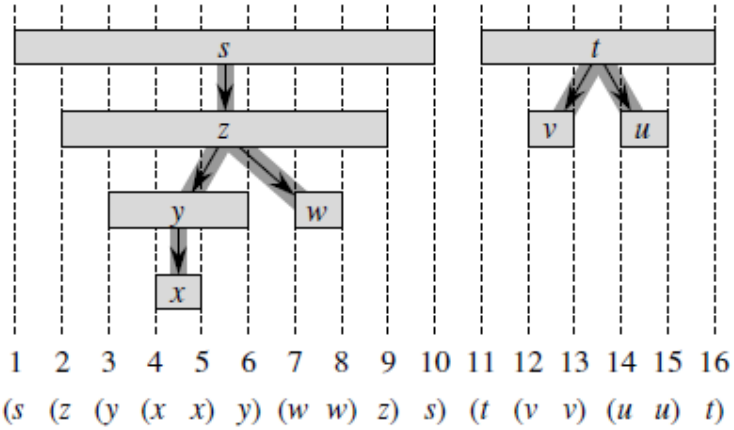
Example:


u v w
1/
x y z
(a)
1/ 2/
(b)
1/ 2/ 3/
(c)
1/ 2/ 4/ 3/
(d)
B
(e)
4/5 3/
(f)
4/5 3/6
(g)
2/7
(h)
F
(i)
1/8 2/7
(j)
9/
(k)
C
(l)
10/
(m)
(n)
10/11
(o)
9/12
(p)

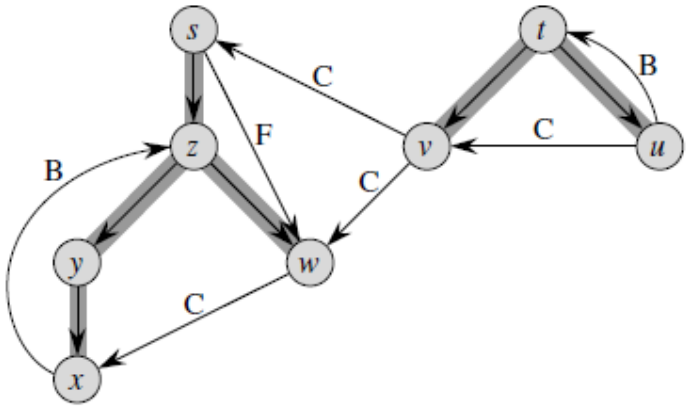
(a)


y
z
s
t
3/6
2/9
1/10
11/16
B
F
C
B
4/5
7/8
12/13
14/15
C
C
C
x
w
v
u

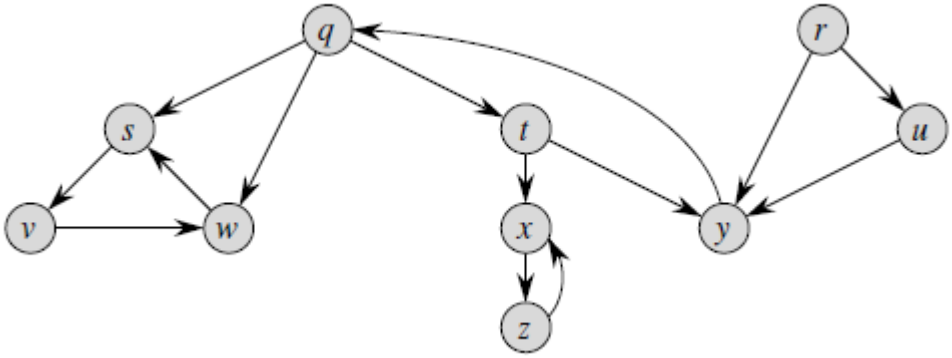
(b)


s
t
z
v
u
y
w
x
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(s (z (y (x x) y) (w w) z) s) (t (v v) (u u) t)

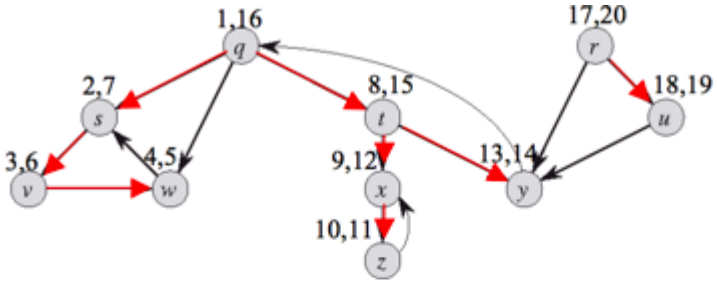
(c)


s
t
C
B
F
C
z
v
u
B
C
y
w
C
x

Q 1. Show how depth-first search works on the graph below. Assume that the for loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge. (**submit it on Google classroom** using pen & paper)



Soln



Tree edges: (q, s), (s, v), (v, w), (q, t), (t, x), (x, z), (t, y), (r, u)

Back edges: (w, s), (z, x), (y, q)

Forward edges: (q, w)

Cross edges: (r, y), (u, y)

(Parenthesis theorem)

- An important property of depth-first search is that discovery and finishing times have **parenthesis structure**.
- If we represent the discovery of vertex u with a left parenthesis "(u" and represent its finishing by a right parenthesis "u)", then the history of discoveries and finishings makes a well-formed expression in the sense that the parentheses are properly nested.
  Eg:
  Valid: ( ) [ ] ([ ]) [( )]
  Invalid : ([ )] [( ])

- Another way of stating the condition of parenthesis structure is given in the following theorem.

Theorem: In any depth-first search of a (directed or undirected) graph G = (V, E), for any two vertices u and v, **exactly** one of the following three conditions holds:

• the intervals [d[u], f [u]] and [d[v], f [v]] are entirely disjoint, and neither u nor v is a descendant of the other in the depth-first forest,

• the interval [d[u], f [u]] is contained entirely within the interval [d[v], f [v]], and u is a descendant of v in a depth-first tree, or

• the interval [d[v], f [v]] is contained entirely within the interval [d[u], f [u]], and v is a descendant of u in a depth-first tree.

Eg: For example, d[u] < d[v] < f [u] < f [v] cannot happen.

Classification of edges

- Another interesting property of depth-first search is that the search can be used to classify the edges of the input graph G = (V, E).
- This edge classification can be used to glean important information about a graph.
  Eg: For example, we shall see that a directed graph is acyclic if and only if a depth-first search yields no "back" edges
- We can define four edge types in terms of the depth-first forest Gπ produced by a depth-first search on G.

1. ***Tree edges*** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).
2. ***Back edges*** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. Self-loops, which may occur in directed graphs, are considered to be back edges.
3. ***Forward edges*** are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.
4. ***Cross edges*** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.