Advanced Algorithms Analysis and Design

By

Nazir Ahmad Zafar

Lecture No 29

Proof (Breadth First Search Algorithm)

Depth First Search

Theorem (Correctness of BFS)

Statement

Let $G = (V, E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source vertex $s \in V$. Then, during its execution, BFS discovers every vertex $v \in V$ that is reachable from the source s, and upon termination

$$d[v] = \delta(s, v) \text{ for all } v \in V.$$

Moreover, for any vertex $v \neq s$ that is reachable from s, one of the shortest paths from s to v is a shortest path from s to $\pi[v]$ followed by edge $(\pi[v], v)$.

Theorem (Correctness of BFS)

Proof

- Assume, for the purpose of contradiction, that some vertex receives a d value not equal to its shortest path distance.

- Let v be the vertex with minimum $\delta(s, v)$ that receives such an incorrect d value; clearly $v \neq s$.

- By Lemma 22.2, $d[v] \geq \delta(s, v)$, and thus we have that $d[v] > \delta(s, v)$. Vertex v must be reachable from s, for if it is not, then $\delta(s, v) = \infty \geq d[v]$.

Theorem (Correctness of BFS)

- Let u be the vertex immediately preceding v on a shortest path from s to v, so that

$$\delta(s, v) = \delta(s, u) + 1.$$

- Because $\delta(s, u) < \delta(s, v)$, and because of how we chose v, we have $d[u] = \delta(s, u)$.

- Putting these properties together, we have

$$d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1 \qquad (22.1)$$

- Now consider the time when BFS chooses to dequeue vertex u from Q in line 11.

Theorem (Correctness of BFS)

- At this time, vertex v is, white, gray, or black.
- We shall show that in each of these cases, we derive a contradiction to inequality (22.1).
- If v is white, then line 15 sets $d[v] = d[u] + 1$, contradicting inequality (22.1).
- If v is black, then it was already removed from the queue and, by Corollary 22.4, we have $d[v] \le d[u]$, again contradicting inequality (22.1).

- If v is gray, then it was painted gray upon dequeuing some vertex w, which was removed from Q earlier than u and, $d[v] = d[w] + 1$.

Theorem (Correctness of BFS)

- By Corollary 22.4, however, $d[w] \le d[u]$, and so we have $d[v] \le d[u] + 1$, once again contradicting inequality (22.1).
- Thus we conclude that $d[v] = \delta(s, v)$ for all $v \in V$. All vertices reachable from s must be discovered, if they were not, they would have infinite d values.
- To conclude the proof of the theorem, observe that if $\pi[v] = u$, then $d[v] = d[u] + 1$.
- Thus, we can obtain a shortest path from s to v by taking a shortest path from s to $\pi[v]$ and then traversing the edge $(\pi[v], v)$

Depth First Search

- The predecessor subgraph of a depth-first search forms a **depth-first forest** composed of several **depth-first trees** defined as

$$G_\pi = (V_\pi, E_\pi), \text{ where}$$
$$E_\pi = \{(\pi[v], v) : v \in V \text{ and } \pi[v] \neq \text{NIL}\}$$

the edges in E_π are called **tree edges.**

- Each vertex is initially white
  - It is grayed when it is **discovered** in the search, and
  - It is blackened when it is **finished**, that is, when its adjacency list has been examined completely.

Discovery and Finish Times

- It guarantees that each vertex ends up in exactly one depth-first tree, so that these trees are disjoint.

- It **timestamps** each vertex
  - 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).

- For every vertex u

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

Algorithm: Depth First Search

DFS(G)

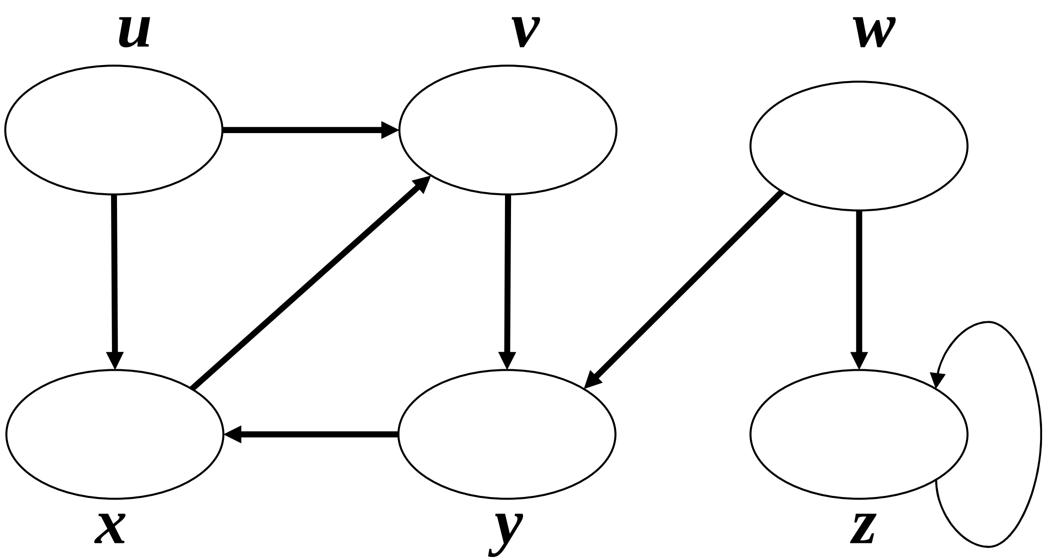
for each vertex $u \in V[G]$
    do $color[u] \leftarrow$ WHITE
        $\pi[u] \leftarrow$ NIL
$time \leftarrow 0$
for each vertex $u \in V[G]$
    do if $color[u] =$ WHITE
        then DFS-Visit (u)

DFS-Visit(u)

$color[u] \leftarrow$ GRAY
$time \leftarrow time + 1$
$d[u] \leftarrow time$
for each $v \in Adj[u]$
    do if $color[v] =$ WHITE
        then $\pi[v] \leftarrow u$
            DFS-Visit (v)
$color[u] \leftarrow$ BLACK
$f[u] \leftarrow time \leftarrow time + 1$

Total Running Time = Θ (V + E)

Depth First Search



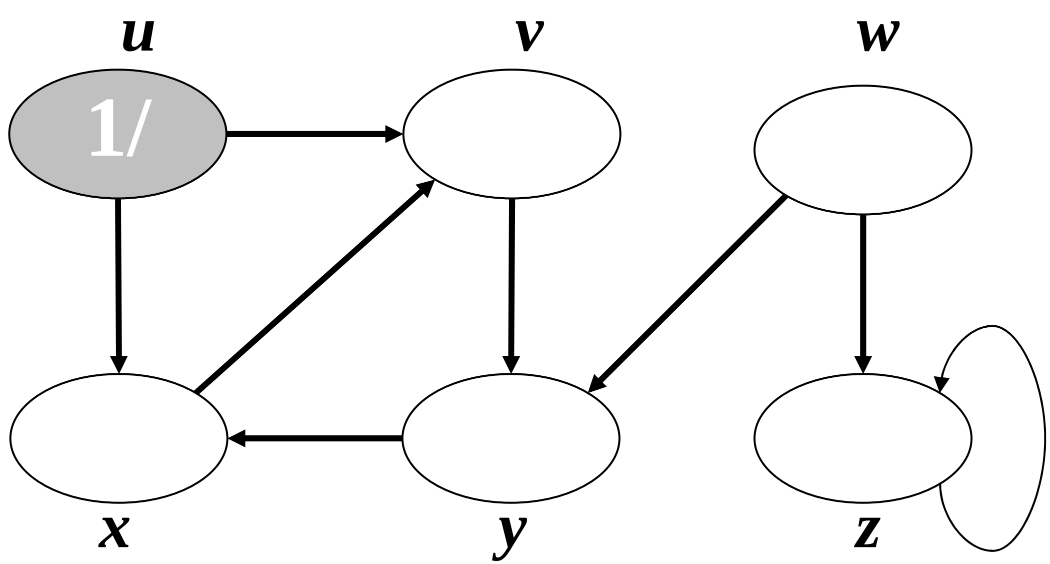
For each vertex u $\in V(G)$

$color\ [u] \leftarrow$ WHITE

$\pi\ [u] \leftarrow NIL$

$time \leftarrow 0$

Depth First Search



Considering white vertex u

$color[u] \leftarrow \text{GRAY}$

$d[u] \leftarrow time + 1 = 0 + 1 = 1$

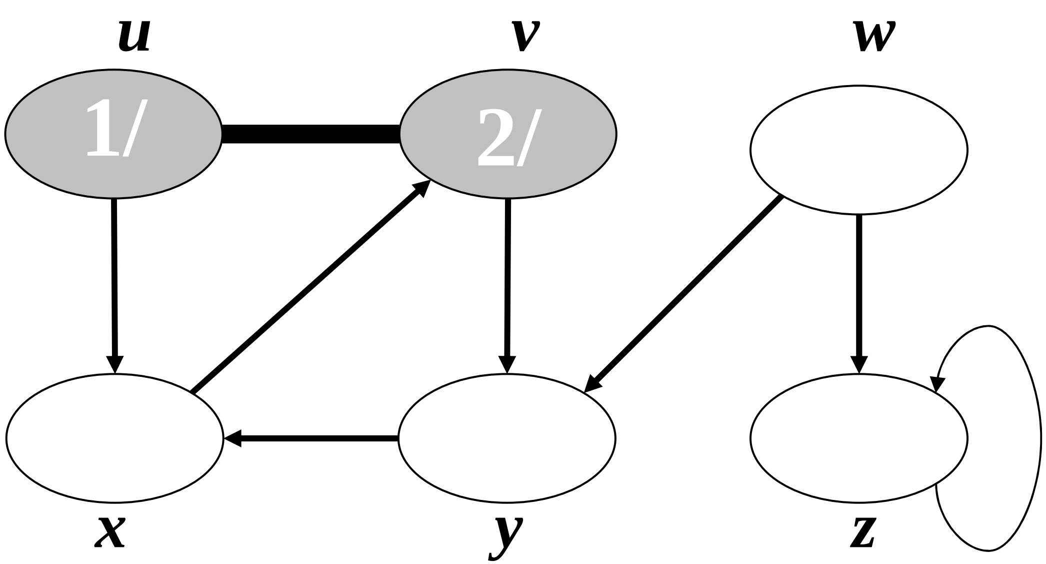
$Adj[u] = v, x$

$color[v] = \text{WHITE}$

$\pi[v] \leftarrow u$

DFS-VISIT (v)

Depth First Search



$color\ [v] \leftarrow$ GRAY

$d[v] \leftarrow time + 1 = 1 + 1 = 2$

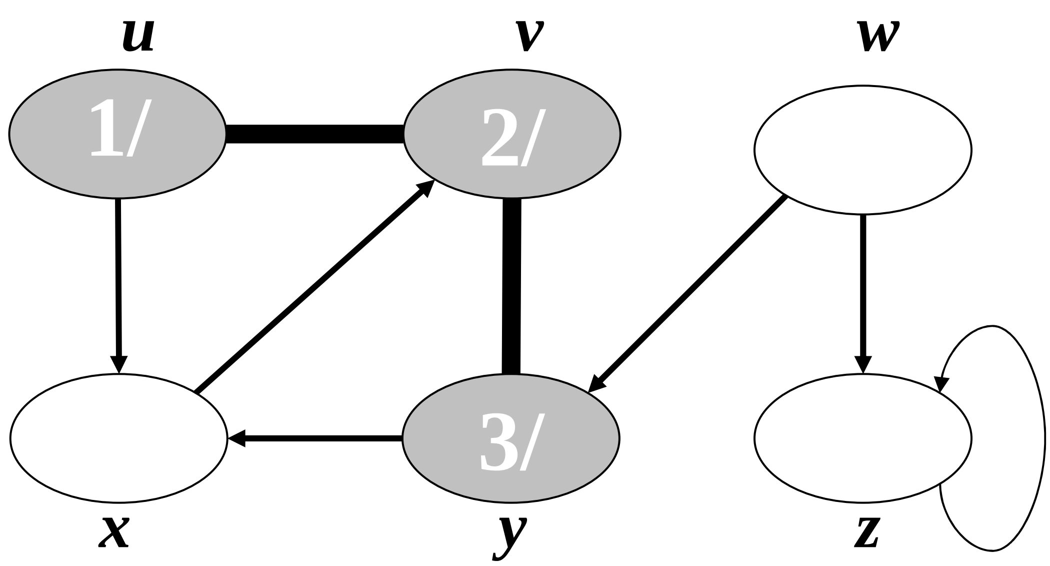
$Adj[v] = y$

$color\ [y] =$ WHITE

$\pi[y] \leftarrow v$

DFS-VISIT (y)

Depth First Search



$color\ [y] \leftarrow$ GRAY

$d[y] \leftarrow time + 1 = 2 + 1 = 3$

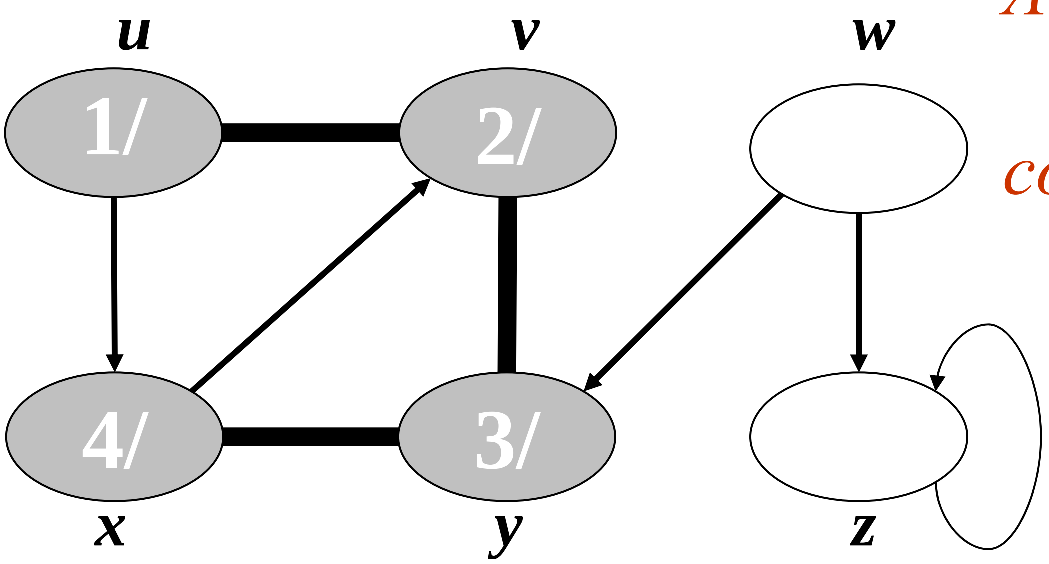
$Adj[y] = x$

$color\ [x]$ = WHITE

$\pi[x] \leftarrow y$

DFS-VISIT (x)

Depth First Search



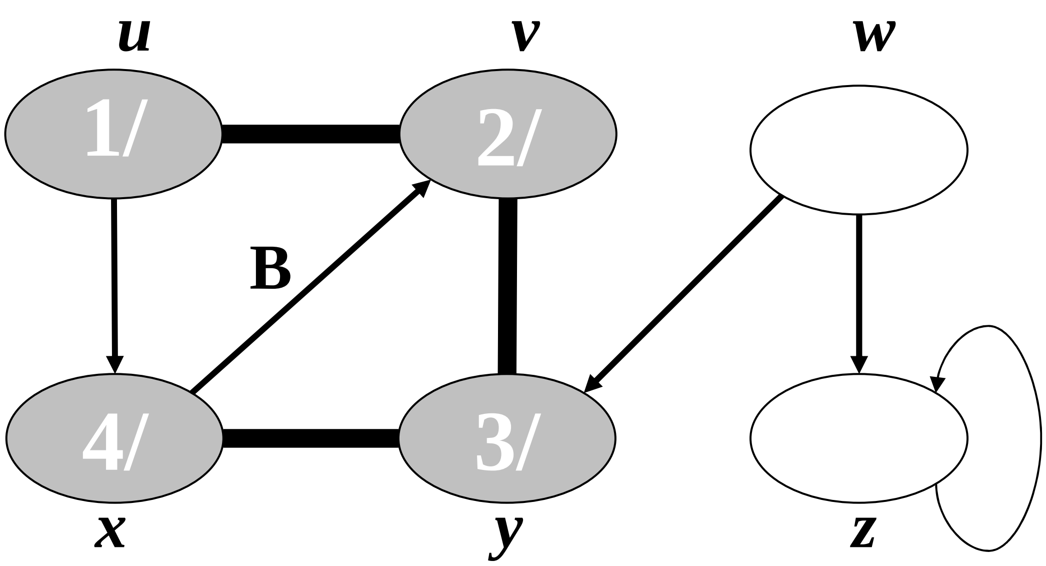
$color\ [x] \leftarrow \text{GRAY}$

$d[x] \leftarrow time + 1 = 3 + 1 = 4$

$Adj[x] = v$

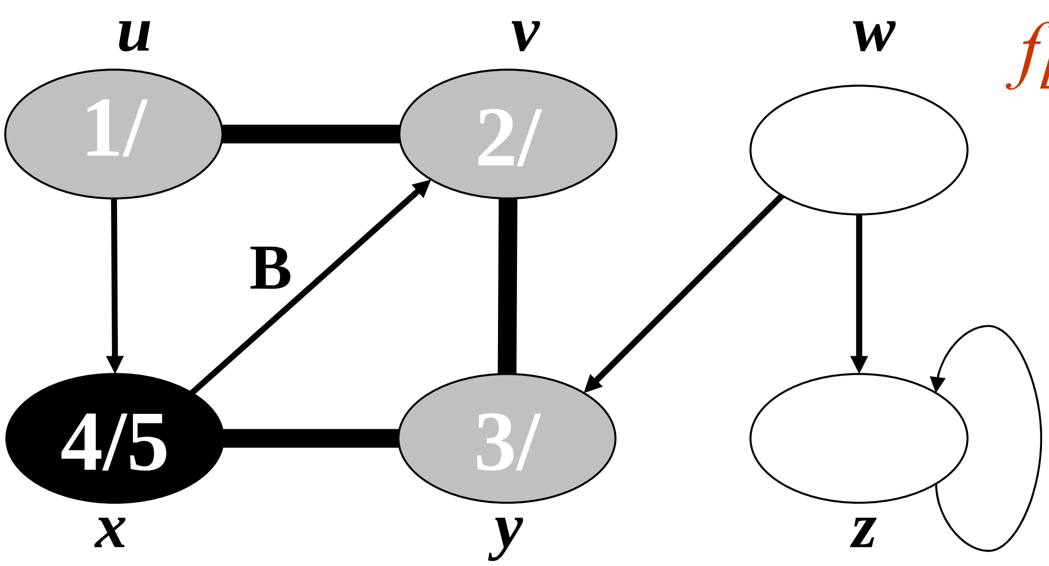
$color\ [v] \neq \text{WHITE}$

Depth First Search



The edge *(x, v)* is a back edge that is a non tree edge and is labeled as B

Depth First Search

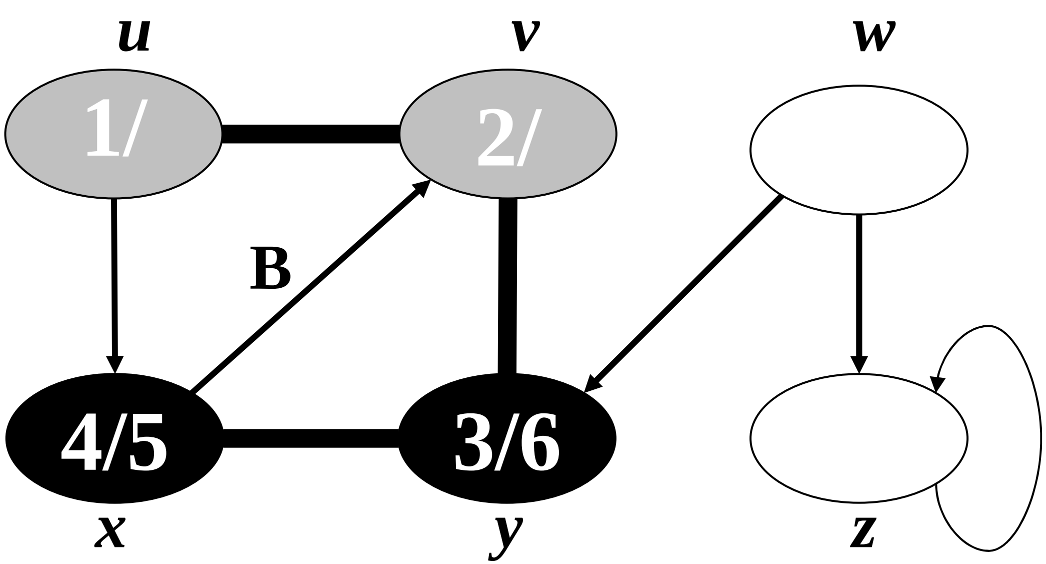


The vertex x is finished.

$color\ [x] \leftarrow$ BLACK

$f[x] \leftarrow time + 1 = 4 + 1 = 5$

Depth First Search

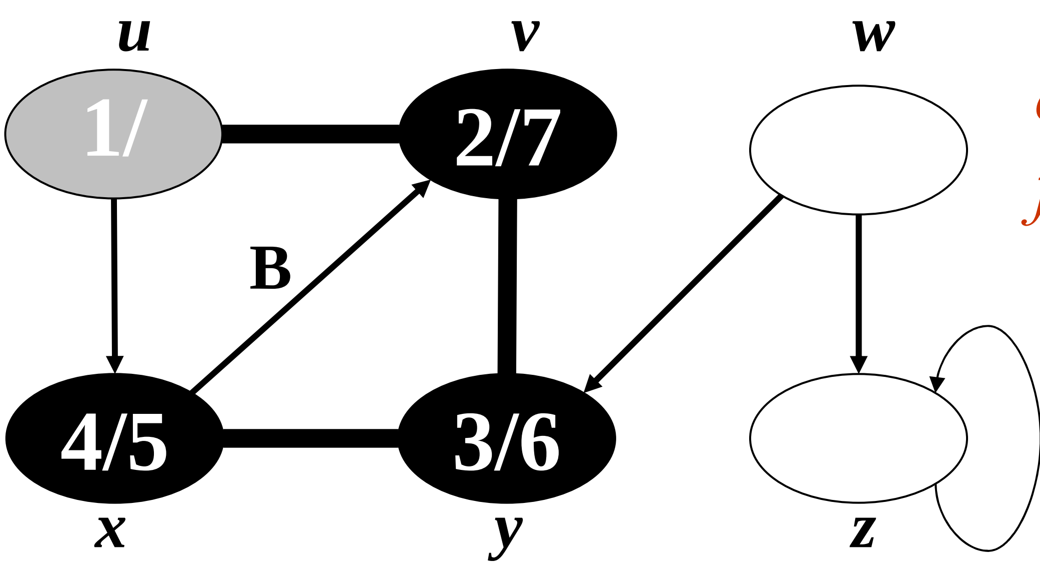


The vertex y is finished.

$color\ [y] \leftarrow \text{BLACK}$

$f[y] \leftarrow time + 1 = 5 + 1 = 6$

Depth First Search



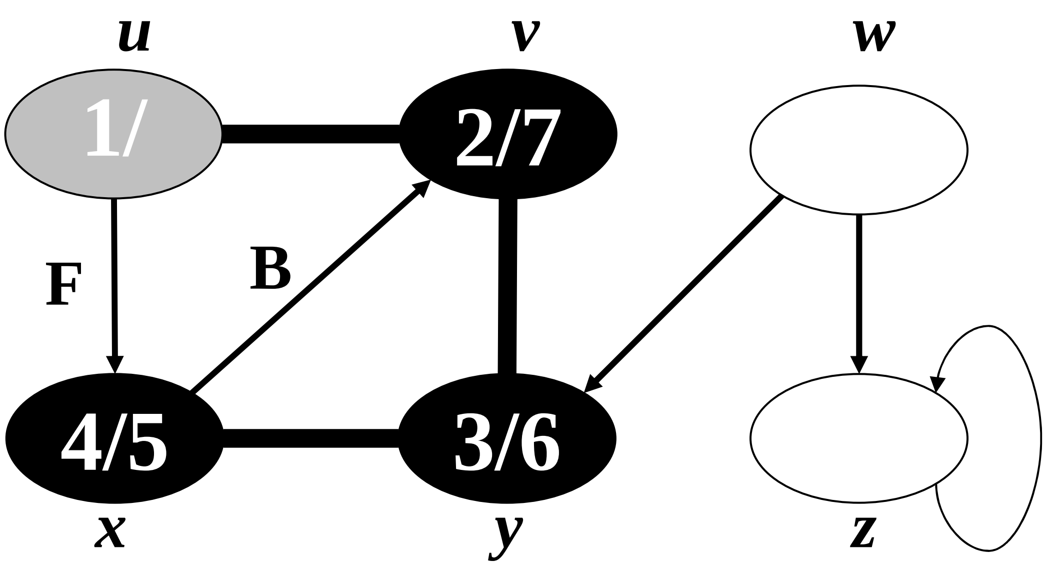
The vertex *v* is finished.

$color\,[v] \leftarrow \text{BLACK}$

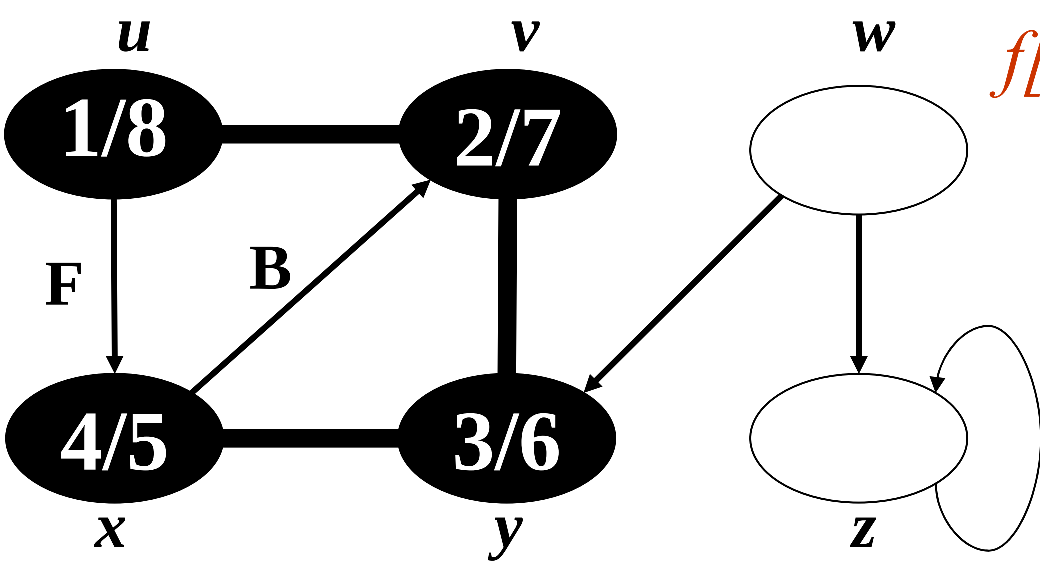
$f[v] \leftarrow time + 1 = 6 + 1 = 7$

Depth First Search

The edge *(u, x)* is a forward edge that is a non tree edge and is labeled as F



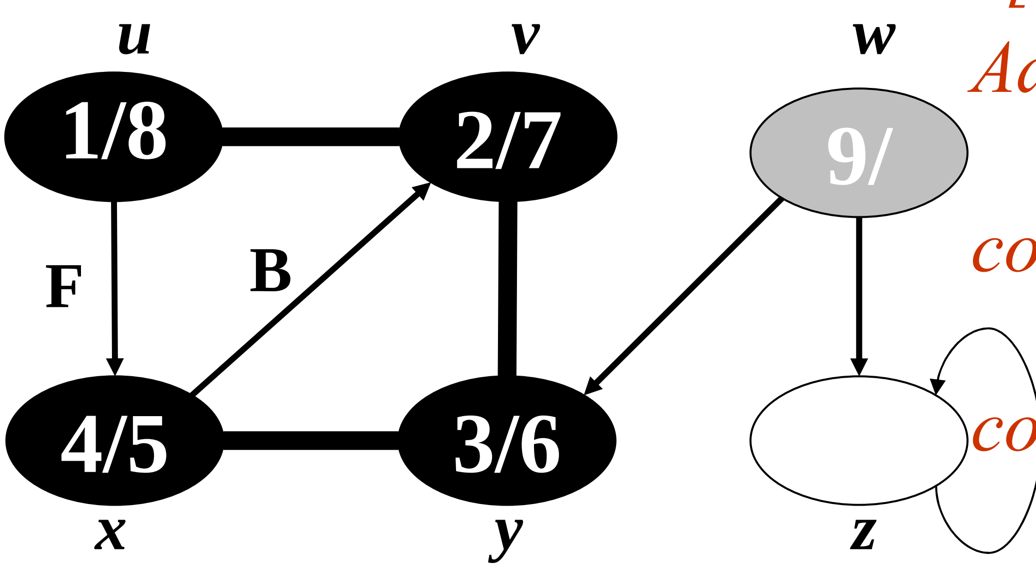
Depth First Search



The vertex u is finished.

$color\,[u] \leftarrow \text{BLACK}$
$f[u] \leftarrow time + 1 = 7 + 1 = 8$

Depth First Search



Considering white vertex w

$color\ [w] \leftarrow \text{GRAY}$

$d[w] \leftarrow time + 1 = 8 + 1 = 9$

$Adj[w] = y, z$

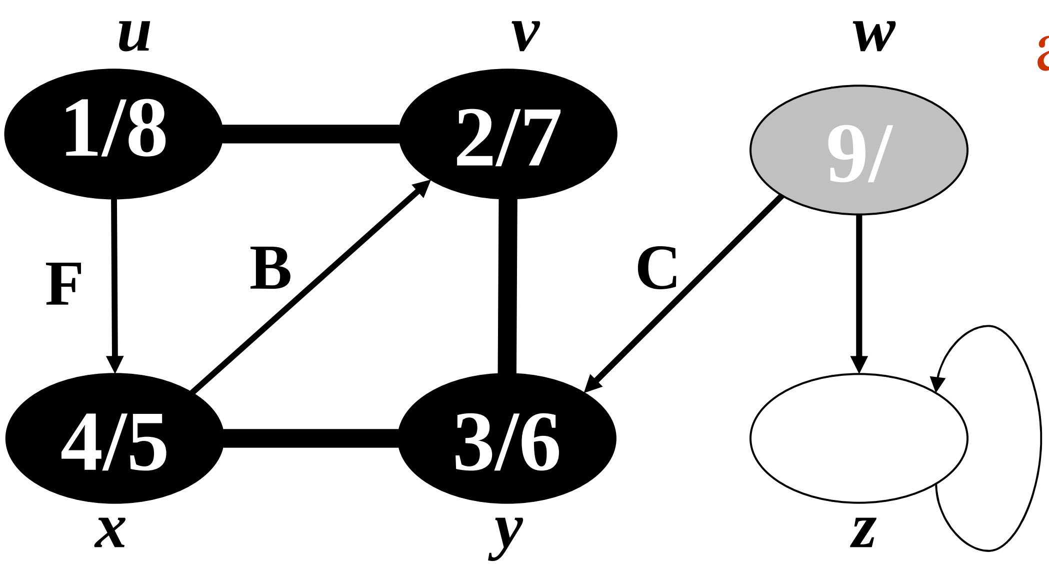
$color\ [y] \neq \text{WHITE}$

$color\ [z] = \text{WHITE}$

$\pi[z] \leftarrow w$

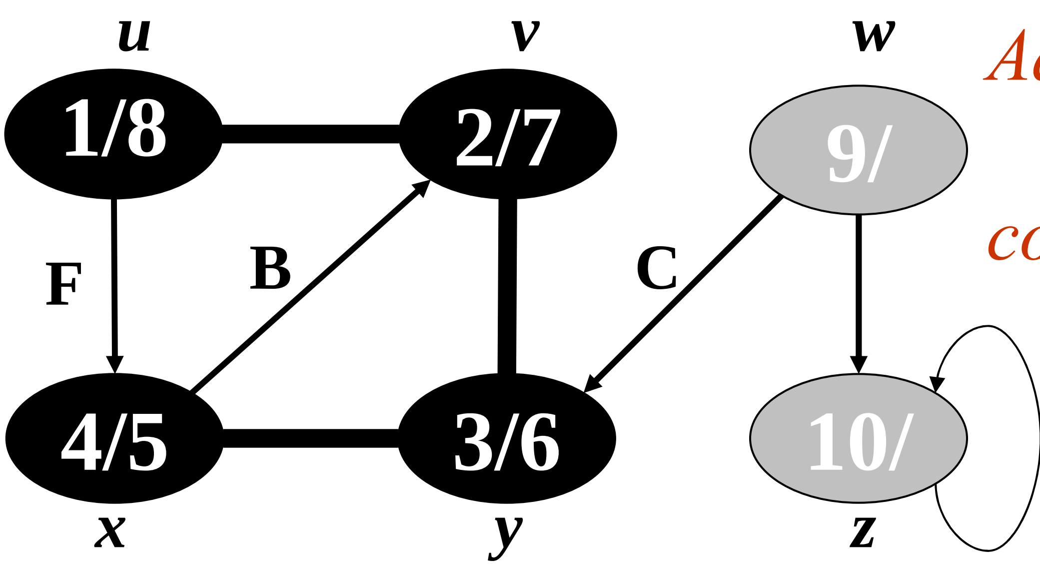
DFS-VISIT (z)

Depth First Search



The edge *(w, y)* is a cross edge that is a non tree edge and is labeled as C

Depth First Search



$color\ [z] \leftarrow \text{GRAY}$

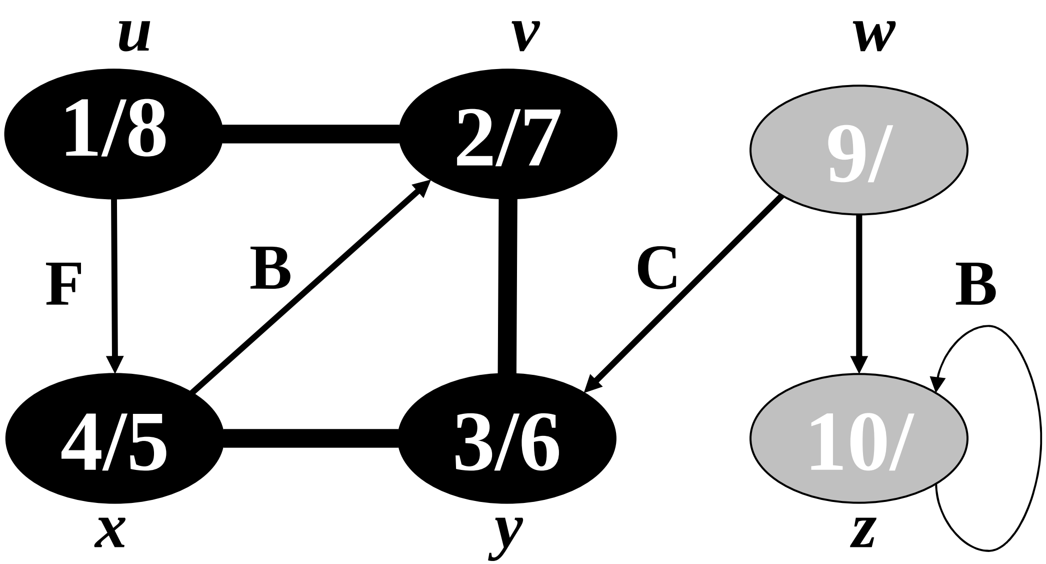
$d[z] \leftarrow time + 1 = 9 + 1 = 10$

$Adj[z] = z$

$color\ [z] \neq \text{WHITE}$

Depth First Search

The edge *(z, z)* is a back edge that is a non tree edge and is labeled as B

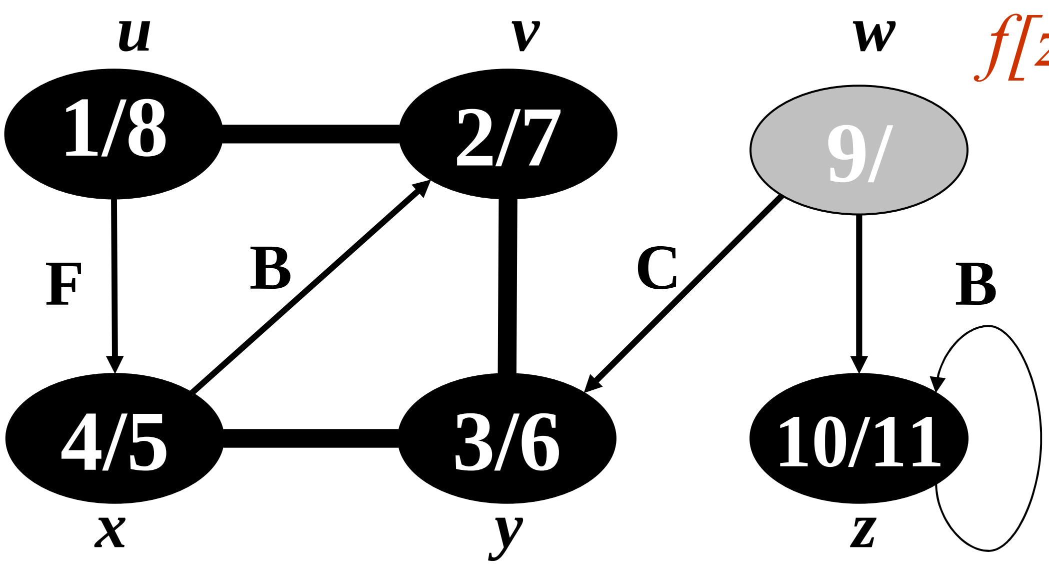


Depth First Search

The vertex z is finished.

$color\,[z] \leftarrow \text{BLACK}$

$f[z] \leftarrow time + 1 = 10 + 1 = 11$

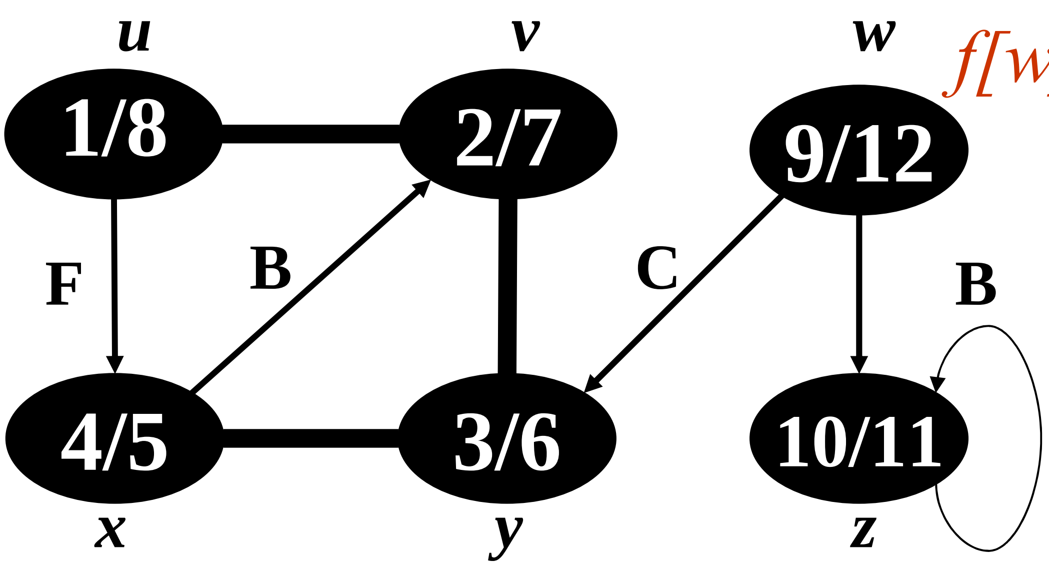


Depth First Search

The vertex w is finished.

$color\ [w] \leftarrow$ BLACK

$f[w] \leftarrow time + 1 = 11 + 1 = 12$

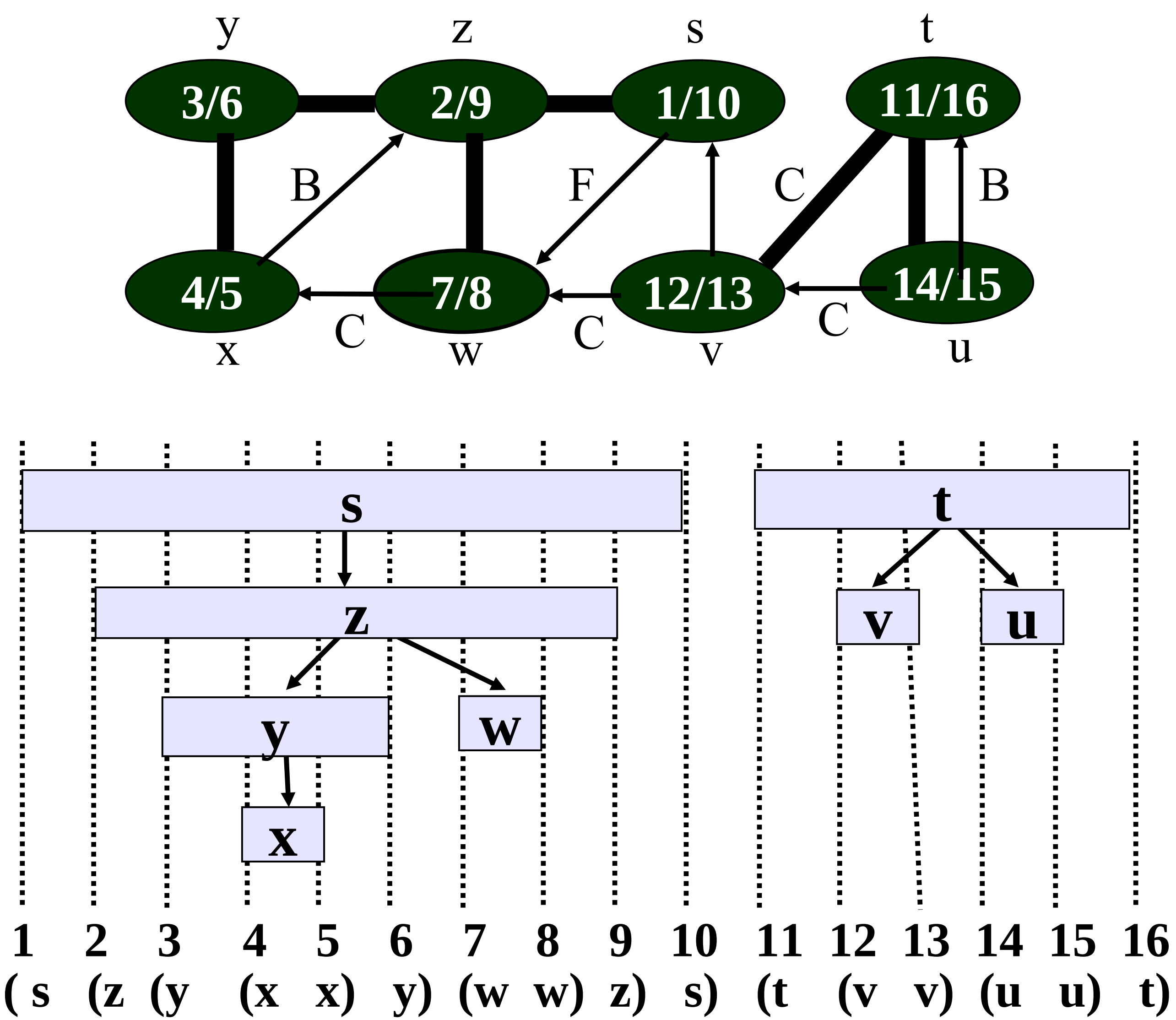


Properties of Depth First Search

- It yields valuable information about structure of a graph.
  - Predecessor subgraph G_π does indeed form a forest of trees, since the structure of the depth-first trees exactly mirrors the structure of recursive calls of DFS-VISIT.

- 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
  - history of discoveries and finishing makes well-formed expression in a sense that parentheses properly nested.

Parenthesis Structure



Theorem : Parenthesis 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:

1. 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,
2. 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
3. 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.

Theorem: Parenthesis Theorem (Cont.)

Proof

- We begin with case in which $d[u] < d[v]$.
- There are two sub-cases, either
  $d[v] < f[u]$ or $d[v] > f[u]$.

Case 1

- $d[v] < f[u] \Rightarrow v$ discovered while *u* was still gray.
- This means v is a descendant of *u*.
- Since *v* was discovered more recently than *u*, all of its outgoing edges are explored, and *v* is finished, before search finishes *u*.
- *Hence* $d[u] < d[v] < f(v) < f(u)$ (part 3 is proved)

Theorem:Parenthesis Theorem (Cont.)

Case 2

- $d[u] < d[v]$ (supposed)
- and $f[u] < d[v]$ (by case 2)
- Hence intervals $[d[u], f[u]]$ and $[d[v], f[v]]$ disjoint.
- Because intervals are disjoint, neither vertex was discovered while the other was gray, and so neither vertex is a descendant of the other.

- Now if we suppose $d[v] < d[u]$, then again either
- Intervals will be disjoint OR
- Interval of v will contain interval of u.

Corollary (Nesting of Descendants' Intervals)

Vertex v is a proper descendant of vertex u in the depth-first forest for a (directed or undirected) graph G if and only if $d[u] < d[v] < f[v] < f[u]$

Proof

- Immediate from the above Theorem

Classification of Edges

- The depth-first search can be used to classify the edges of the input graph $G = (V, E)$.

  1. **Tree edges**
     - These 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**
     - 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.

Classification of Edges

3. Forward edges

- Those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.

4. Cross edges

- These 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.

Theorem

In a depth-first search of an undirected graph *G*, every edge of *G* is either a tree edge or back edge.

Proof

- Let (u, v) be an arbitrary edge of G, and suppose without loss of generality that d[u] < d[v].

- Then, *v* must be discovered and finished before we finish *u* (while *u* is gray), since *v* is on *u*'s adjacency list.

Theorem (Cont..)

- If the edge (u, v) is explored first in direction from u to v, then v is undiscovered (white) until that time, for otherwise we would have explored this edge already in the direction from v to u.

- Thus, (u, v) becomes a tree edge.

- If (u, v) is explored first in the direction from v to u, then (u, v) is a back edge, since u is still gray at the time the edge is first explored.

Conclusion

- Depth First Search Techniques is discussed
- Algorithms is designed
- Correctness of Depth First Search is given
- Topological sort and its benefits
- Computing strongly connected components
- Applications and Conclusion