

Data Structures and Algorithms

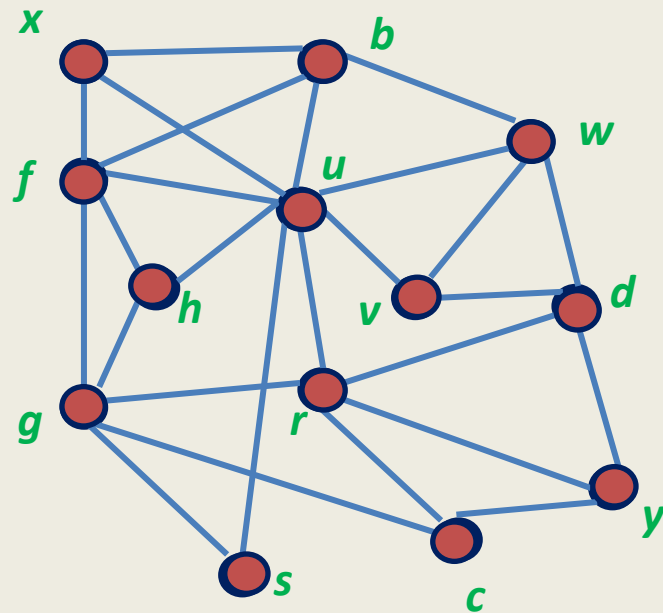
(CS210A)

Semester I – 2014-15

Lecture 24

- A data structure problem for graphs.
- Depth First Search (DFS) Traversal
- Novel application: computing biconnected components of a graph

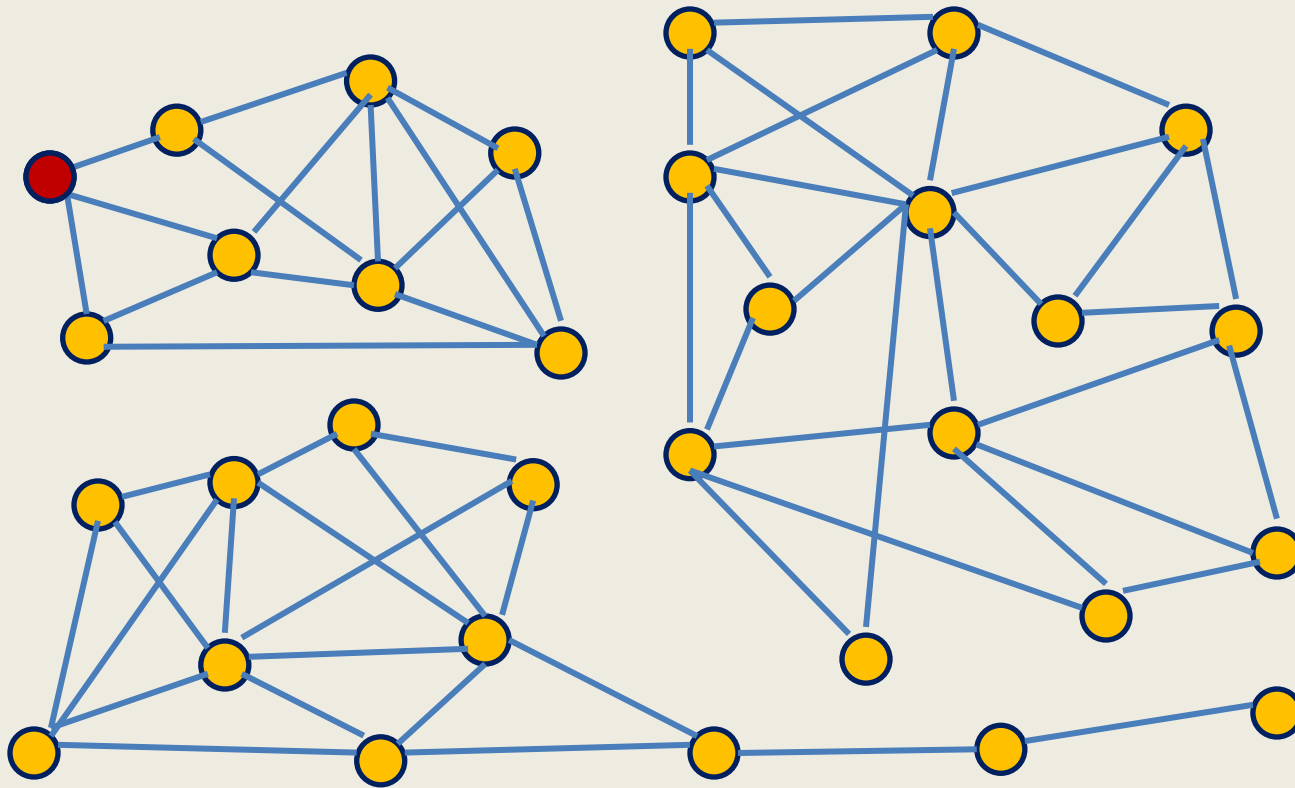
BFS Traversal in Undirected Graphs



Theorem:

BFS Traversal from **x** visits all vertices reachable from **x** in the given graph.

Connectivity problem in a Graph

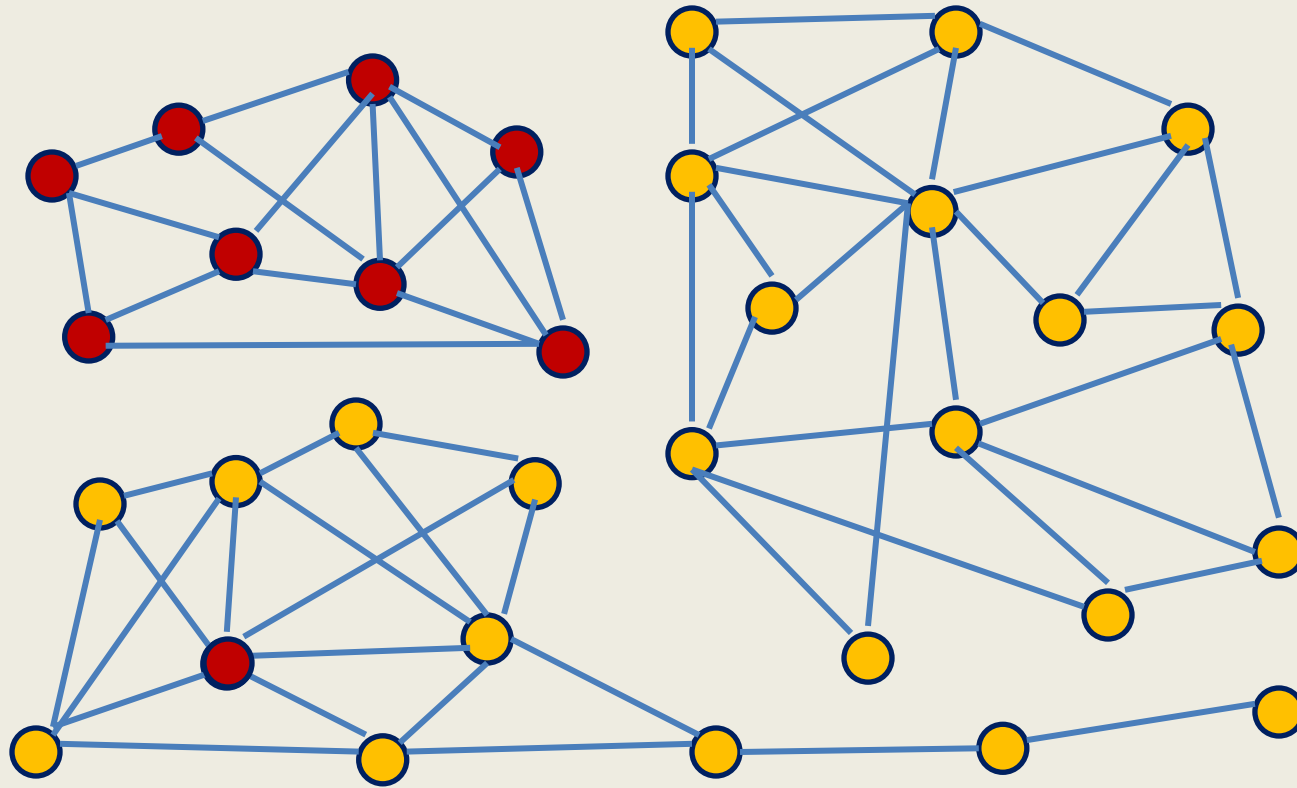


Problem:

Build an $O(n)$ size data structure for a given undirected graph s.t. the following query can be answered in $O(1)$ time.

Is vertex i reachable from vertex j ?

Connectivity problem in a Graph

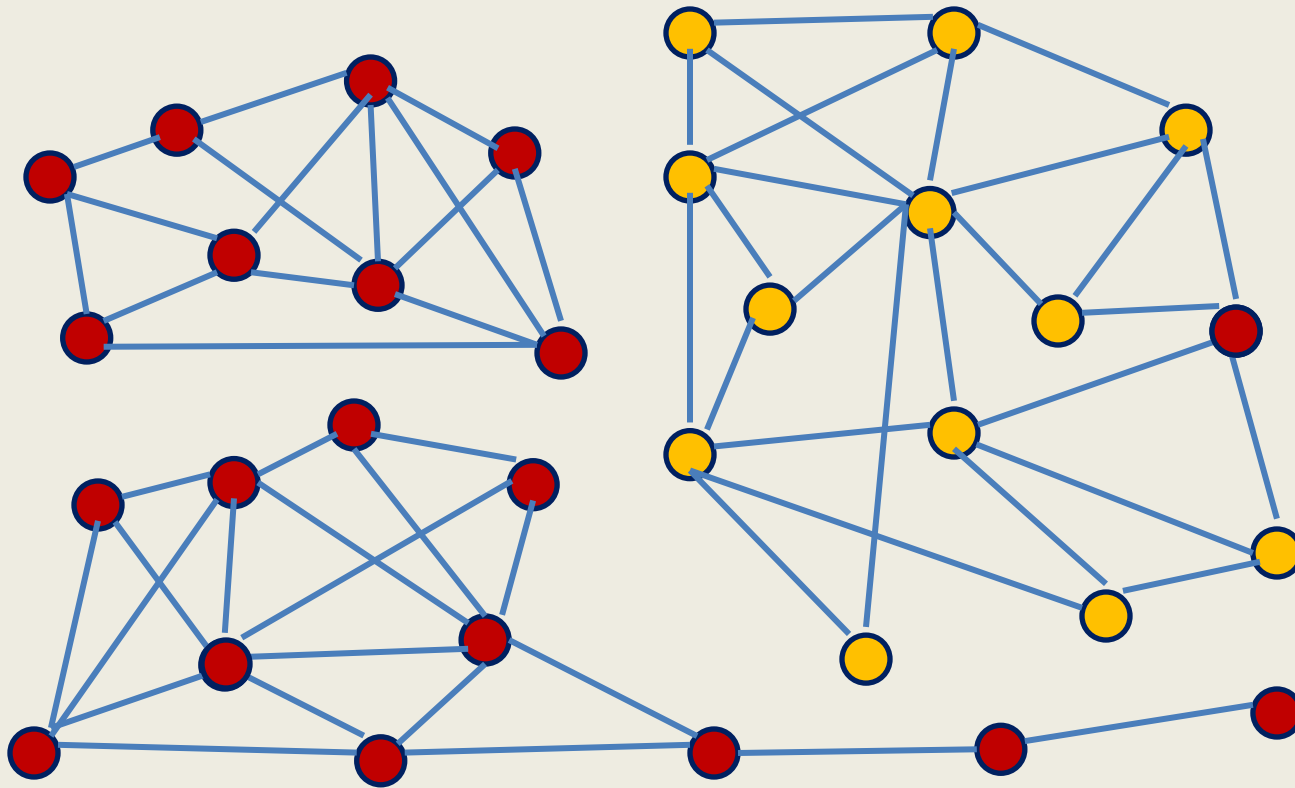


Problem:

Build an $O(n)$ size data structure for a given undirected graph s.t. the following query can be answered in $O(1)$ time.

Is vertex i reachable from vertex j ?

Connectivity problem in a Graph

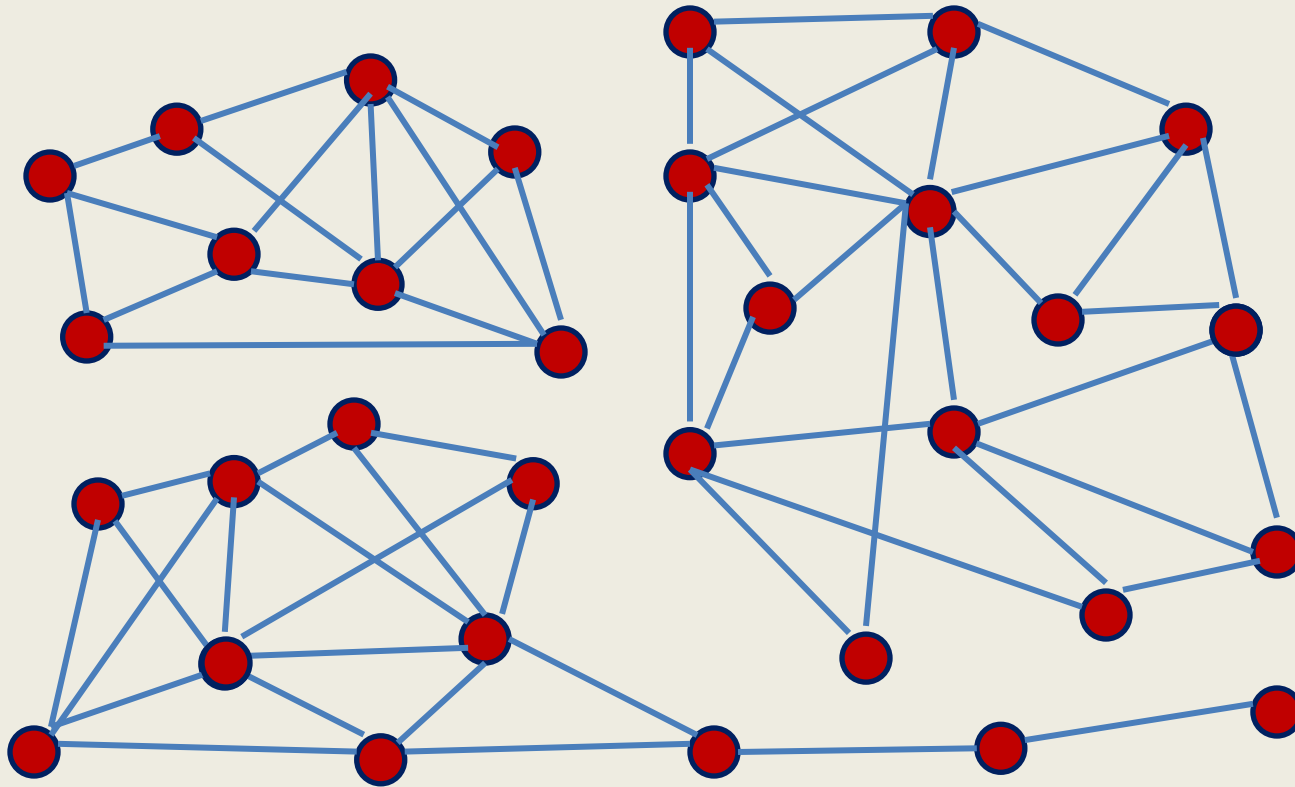


Problem:

Build an $O(n)$ size data structure for a given undirected graph s.t. the following query can be answered in $O(1)$ time.

Is vertex i reachable from vertex j ?

Connectivity problem in a Graph



Problem:

Build an $O(n)$ size data structure for a given undirected graph s.t. the following query can be answered in $O(1)$ time.

Is vertex i reachable from vertex j ?

Connectivity problem in a Graph

BFS(x)

CreateEmptyQueue(Q);

Visited(x) \leftarrow **true**; **Label**[x] \leftarrow x ;

Enqueue(x, Q);

While(**Not IsEmptyQueue**(Q))

{ $v \leftarrow$ **Dequeue**(Q);

For each neighbor w **of** v

 { **if** (**Visited**(w) = **false**)

 { **Visited**(w) \leftarrow **true** ; **Label**[w] \leftarrow x ;

Enqueue(w, Q);

 }

 }

}

Connectivity(G)

{ **For each vertex** x **Visited**(x) \leftarrow **false**; **Create an array** **Label**;

For each vertex v **in** V

If (**Visited**(v) = **false**) **BFS**(x);

return **Label**;

}

Analysis of the algorithm

Output of the algorithm:

Array **Label**[] of size $O(n)$ such that

Label[**x**]=**Label**[**y**] if and only if **x** and **y** belong to same connected component.

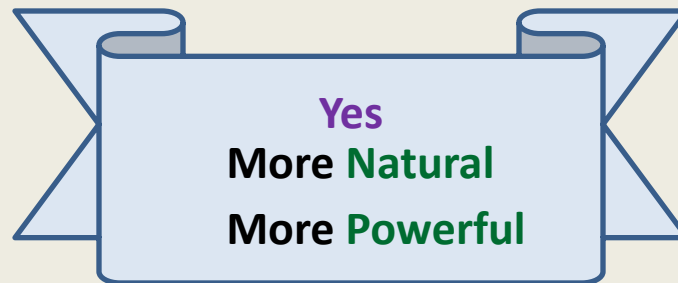
Running time of the algorithm :

$$O(n + m)$$

Theorem:

An undirected graph can be processed in $O(n + m)$ time to build an $O(n)$ size data structure which can answer any connectivity query in $O(1)$ time.

Is there **alternate way** to traverse a graph ?

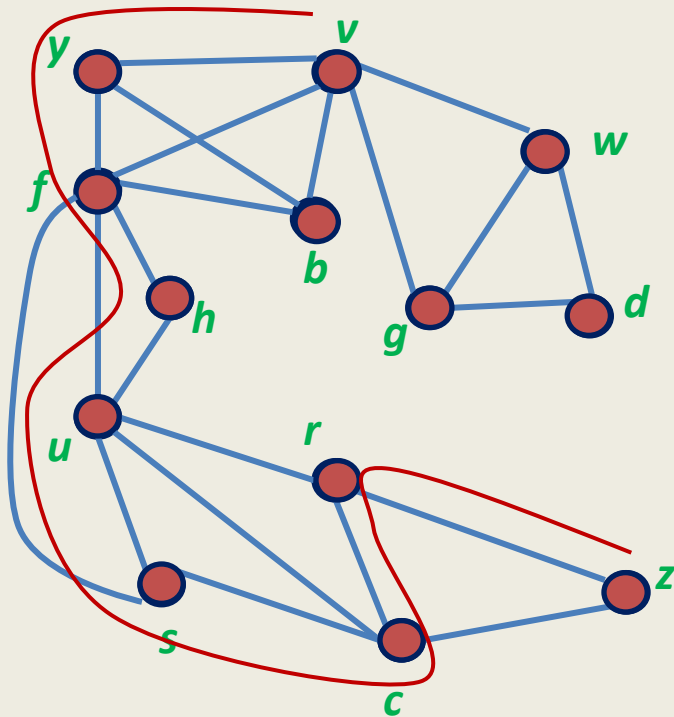


Try to get inspiration from your “human executable method”
to design
“a machine executable algorithm” for traversing a graph.



How will you do it without any map or
asking any one for directions ?

A recursive way to traverse a graph



We need a **mechanism** to

- **Avoid** visiting a vertex multiple times

We can solve it by keeping a label
“**Visited**” for each vertex
like in BFS traversal.

- **Trace back** in case we reach a dead end.

Recursion takes care of it 😊

DFS traversal of G

DFS(v)

```
{ Visited( $v$ )  $\leftarrow$  true;
```

```
  For each neighbor  $w$  of  $v$ 
```

```
  {   if (Visited( $w$ ) = false)
```

```
    { DFS( $w$ ) ;
```

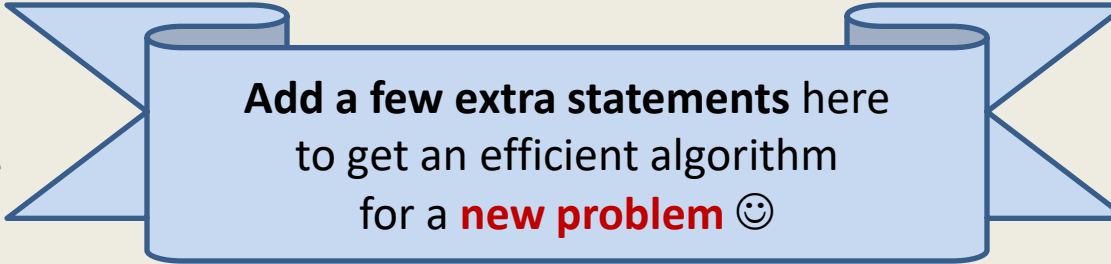
```
      .....; 
```

```
    }
```

```
  .....; 
```

```
}
```

```
}
```



Add a few extra statements here
to get an efficient algorithm
for a **new problem** 😊

DFS-traversal(G)

```
{ For each vertex  $v \in V$  {   Visited( $v$ )  $\leftarrow$  false;   }
```

```
  For each vertex  $v \in V$  {
```

```
    If (Visited( $v$ ) = false) DFS( $v$ );
```

```
  }
```

```
}
```

DFS traversal

a **milestone** in the area of graph algorithms



Invented by **Robert Endre Tarjan** in **1972**

- One of the **pioneers** in the field of data structures and algorithms.
- Got the **Turing award** (equivalent to **Nobel prize**) for his fundamental contribution to data structures and algorithms.
- **DFS traversal** has proved to be a very elegant and powerful tool for graph algorithms.

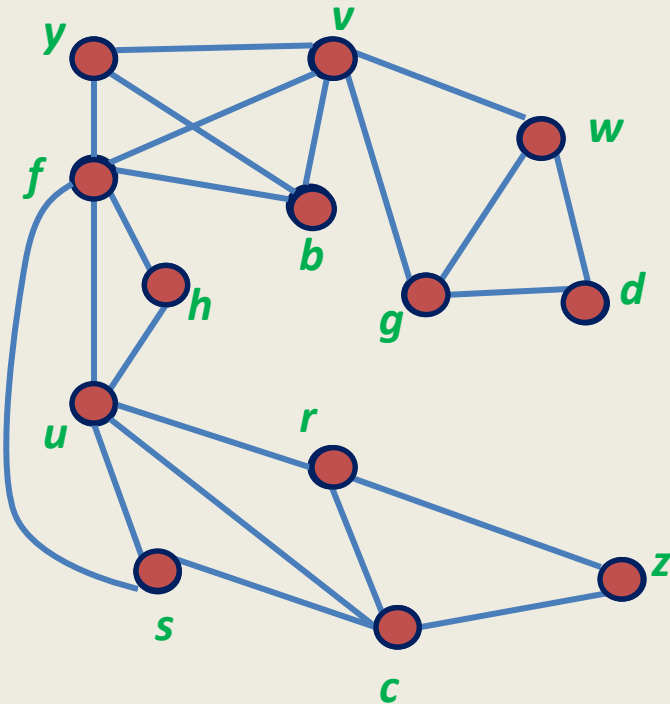
DFS traversal

a **milestone** in the area of graph algorithms

Applications:

- **Connected** components of a graph.
- **Biconnected** components of a graph.
(Is the connectivity of a graph robust to failure of any node ?)
- Finding **bridges** in a graph.
(Is the connectivity of a graph robust to failure of any edge)
- **Planarity testing** of a graph
(Can a given graph be embedded on a plane so that no two edges intersect ?)
- **Strongly connected** components of a directed graph.
(the extension of connectivity in case of directed graphs)

Insight into DFS through an example



DFS(**v**) begins

v visits **y**

DFS(**y**) begins

y visits **f**

DFS(**f**) begins

f visits **b**

DFS(**b**) begins

all neighbors of **b** are already visited

DFS(**b**) ends

control returns to DFS(**f**)

f visits **h**

DFS(**h**) begins

.... and so on

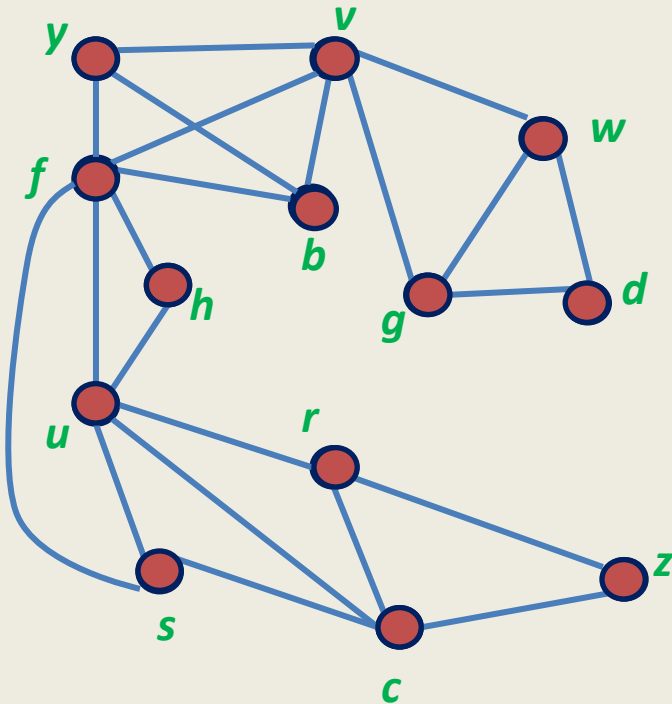
After visiting **z**, control returns to **r** → **c** → **s** → **u** → **h** → **f** → **y** → **v**

v visits **w**

DFS(**w**) begins

.... and so on

Insight into DFS through an example

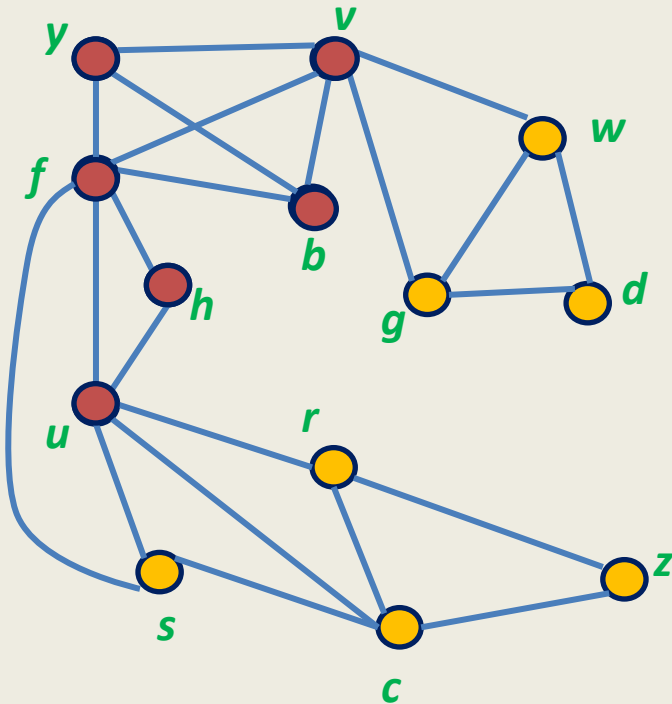


Observation1: (Recursive nature of **DFS**)

If **DFS**(v) invokes **DFS**(w), then

DFS(w) finishes **before** **DFS**(v).

Insight into DFS through an example

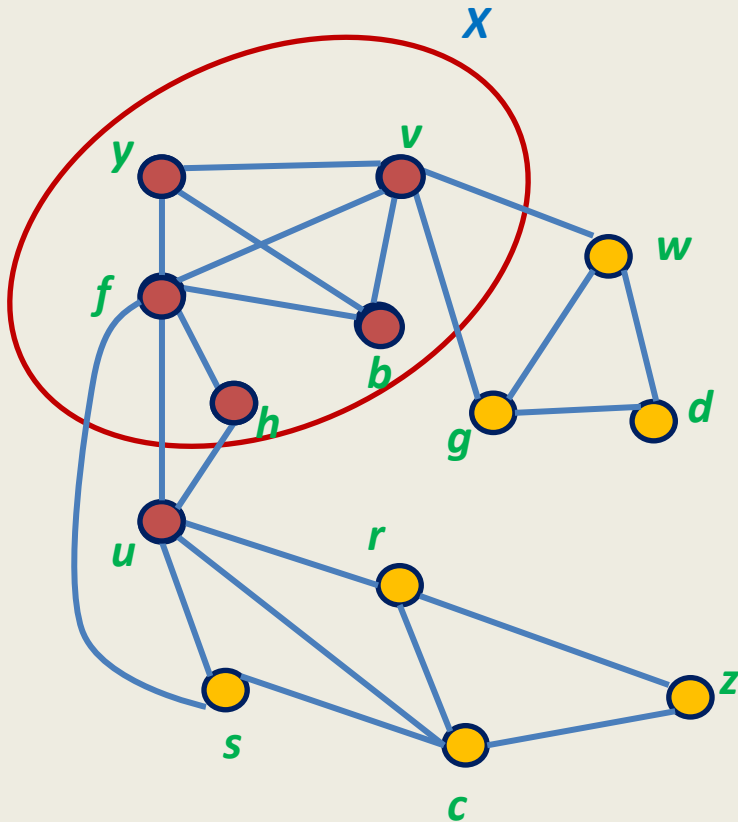


Question :

When **DFS** reaches a vertex **u**, what is the role of vertices already visited ?

The traversal will not proceed along the vertices which are **already visited**. Hence the visited vertices act as a **barrier** for the traversal from **u**.

Insight into DFS through an example



Observation 2:

Let X be the set of vertices visited before **DFS** traversal reaches vertex u for the first time.

The **DFS**(u) pursued now is like

fresh **DFS**(u) executed in graph $G \setminus X$.

NOTE:

$G \setminus X$ is the graph G after removal of all vertices X along with their edges.

Proving that
DFS(v) visits all vertices reachable from **v**

By **induction** on the
size of connected component of v

Can you figure out the **inductive assertion** now?

Think over it. It is given on the following slide...

Inductive assertion

$A(i)$: If a connected component has **size** = i , then **DFS** from any of its vertices **will visit** all its vertices.

PROOF:

Base case: $i = 1$.

The component is $\{v\}$ and the first statement of **DFS**(v) marks it visited.

So $A(1)$ holds.

Induction hypothesis:

If a connected component has **size** < i , then **DFS** from any of its vertices **will visit** all its vertices.

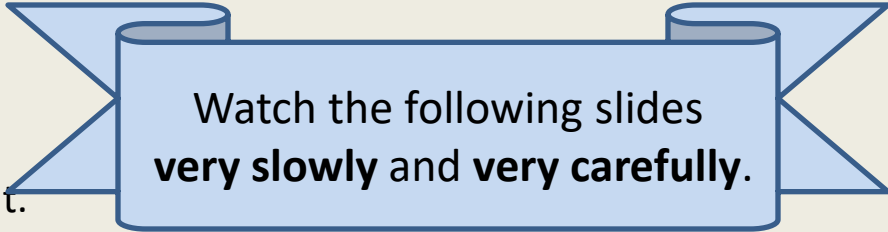
Induction step:

We have to prove that $A(i)$ holds.

Consider any connected component of size i .

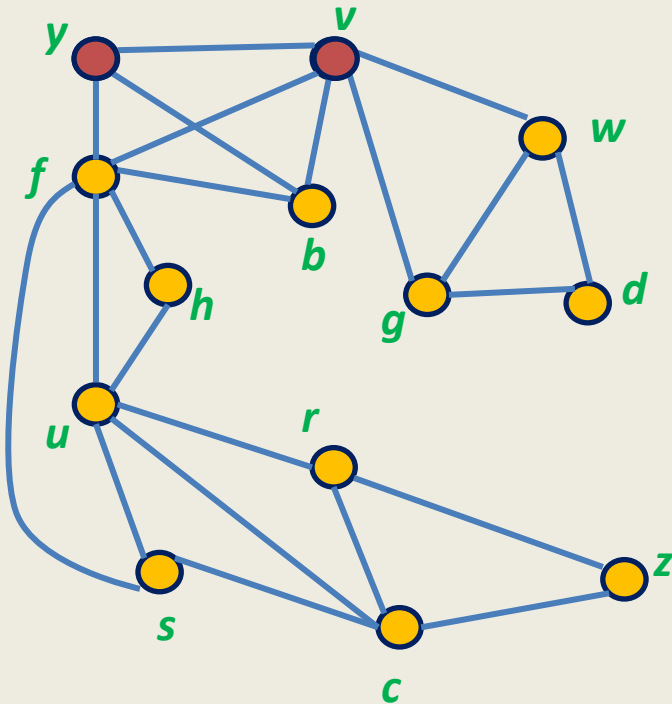
Let V^* be the set of its vertices. $|V^*| = i$.

Let v be any vertex in the connected component.



Watch the following slides
very slowly and **very carefully**.

DFS(v)



Let y be the first neighbor visited by v .

A = the set of vertices such that **every path** from y to them passes through v .

$B = V^* \setminus A$.

$A = \{v, g, w, d\}$

$B = \{y, b, f, h, u, s, c, r, z\}$

$|A| < i$ since $y \notin A$

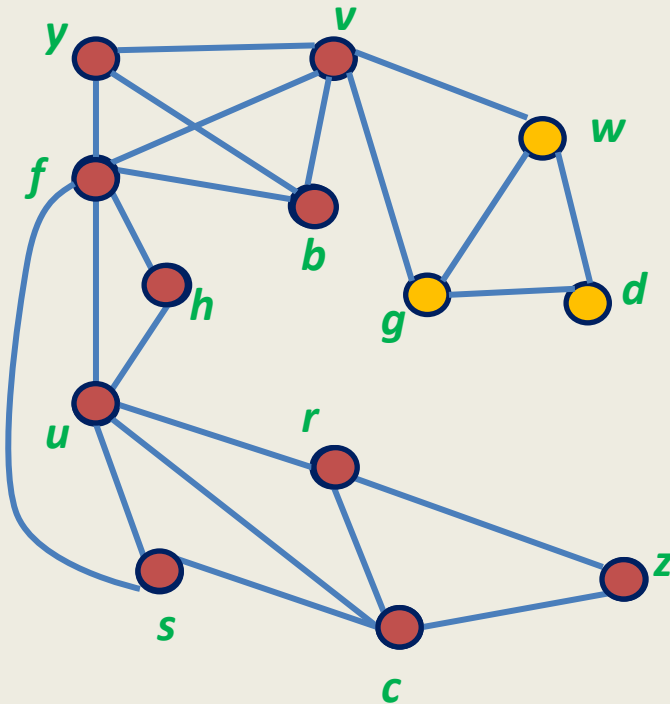
$|B| < i$ since $v \notin B$

Question: What is $\text{DFS}(y)$ like ?

Answer: $\text{DFS}(y)$ in $G \setminus \{v\}$.

Question: What is the connected component of y in $G \setminus \{v\}$?

DFS(v)



Let y be the first neighbor visited by v .

A = the set of vertices such that **every path** from y to them passes through v .

$B = V^* \setminus A$.

$A = \{v, g, w, d\}$

$B = \{y, b, f, h, u, s, c, r, z\}$

$|A| < i$ since $y \notin A$

$|B| < i$ since $v \notin B$

Question: What is $\text{DFS}(y)$ like ?

Answer: $\text{DFS}(y)$ in $G \setminus \{v\}$.

Question: What is the connected component of y in $G \setminus \{v\}$?

Answer: B .

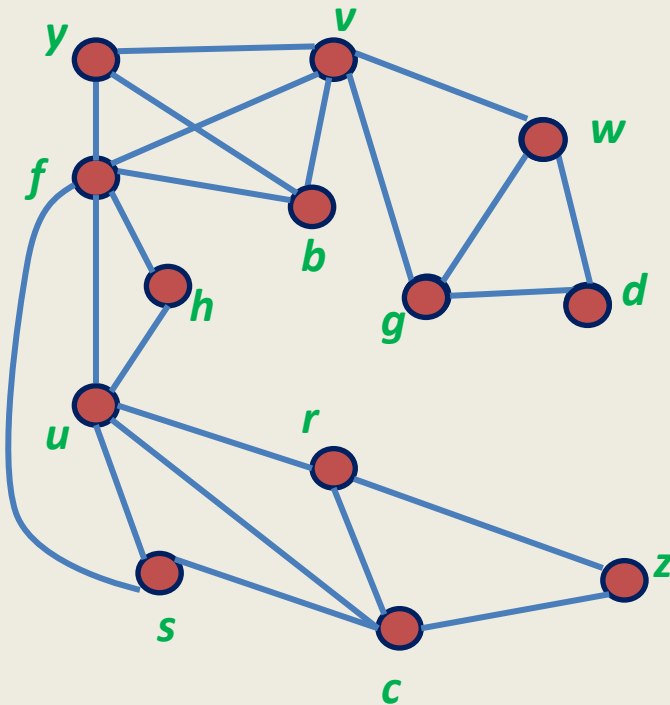
$|B| < i$, so by I.H., $\text{DFS}(y)$ visits entire set B & we return to v .

Question: What is $\text{DFS}(v)$ like when $\text{DFS}(y)$ finishes ?

Answer: $\text{DFS}(v)$ in $G \setminus B$.

Question: What is the connected component of v in $G \setminus B$?

DFS(v)



Hence entire component
of v gets visited

Let y be the first neighbor visited by v .

A = the set of vertices such that **every path** from y to them passes through v .

$B = V^* \setminus A$.

$A = \{v, g, w, d\}$

$B = \{y, b, f, h, u, s, c, r, z\}$

$|A| < i$ since $y \notin A$

$|B| < i$ since $v \notin B$

Question: What is $\text{DFS}(y)$ like ?

Answer: $\text{DFS}(y)$ in $G \setminus \{v\}$.

Question: What is the connected component of y in $G \setminus \{v\}$?

Answer: B .

$|B| < i$, so by I.H., $\text{DFS}(y)$ visits entire set B & we return to v .

Question: What is $\text{DFS}(v)$ like when $\text{DFS}(y)$ finishes ?

Answer: $\text{DFS}(v)$ in $G \setminus B$.

Question: What is the connected component of v in $G \setminus B$?

Answer: A .

$|A| < i$, so by I.H., $\text{DFS}(v)$ pursued after finishing $\text{DFS}(y)$ visits entire set A .

Theorem: $\text{DFS}(v)$ visits all vertices of the connected component of v .

Exercise:

Use **DFS** traversal to compute all connected components of a given G in time $O(m + n)$.