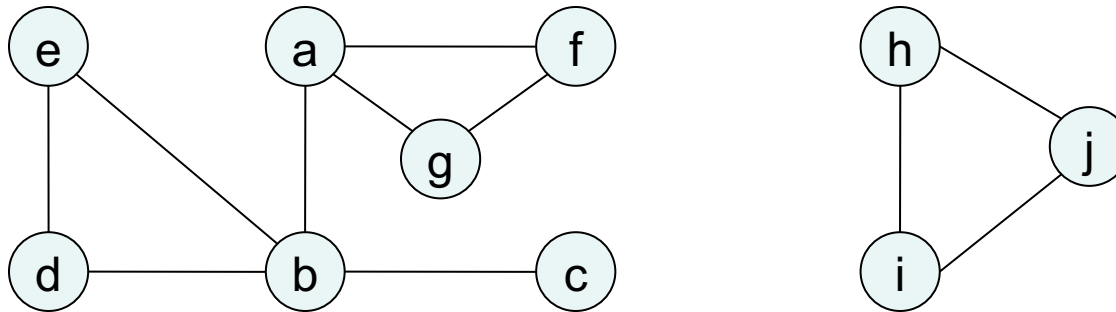


Reachability in undirected graphs

What parts of a graph are reachable from a given vertex?



With an adjacency list representation, this is like navigating a maze...

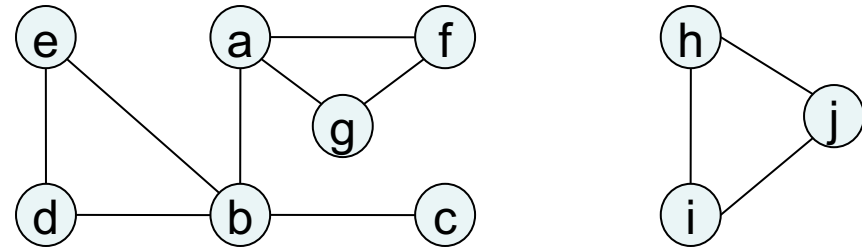
Potential difficulty	Don't go round in circles	Don't miss anything
Classical solution	Piece of chalk to mark visited junctions	Ball of string – leads back to starting point
Cyber-analog	Boolean variable for each vertex: visited or not	STACK

An exploration procedure

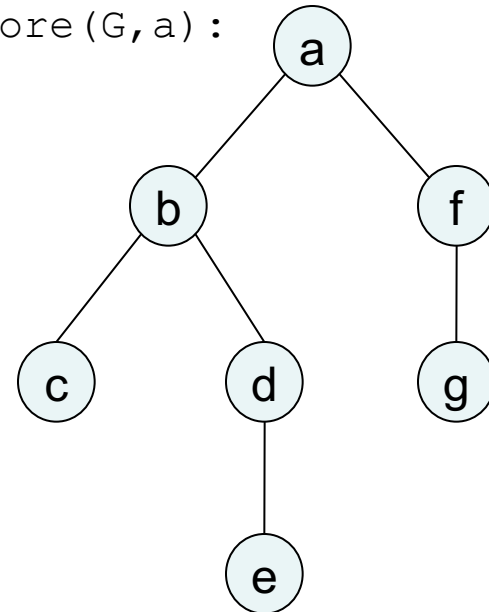
```
procedure explore(G,v)
```

```
input: graph  $G = (V,E)$ ; node  $v$  in  $V$   
output: visited[u] is set to true  
        for all  $u$  reachable from  $v$ 
```

```
visited[v] = true  
for each edge  $(v,u)$  in  $E$ :  
    if not visited[u]:  
        explore(G,u)
```



explore(G,a):



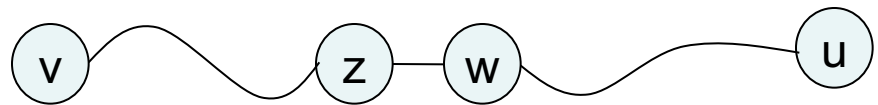
Does “explore” work?

```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
    if not visited[u]:
        explore(G,u)
```

Does it visit everything reachable from v?

Suppose it misses node u reachable from v;
we'll derive a contradiction.

Pick any path from v to u, and let z be the last node on the path that was visited.



But w would not have been overlooked during $\text{explore}(G,z)$; this is a contradiction.

Does it actually halt?

For any node u, $\text{explore}(G,u)$ is called at most once;
thereafter $\text{visited}[u]$ is set.

Alternative proof

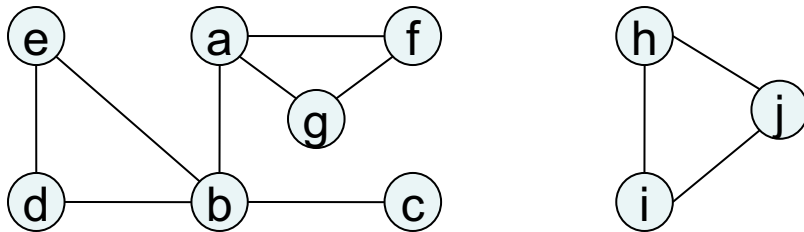
```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
    if not visited[u]:
        explore(G,u)
```

Does `explore(G,v)` visit everything
reachable from `v`?

Do a proof by induction.

Undirected connectivity

An undirected graph is *connected* if there is a path between any pair of nodes.



This graph has 2 *connected components*.

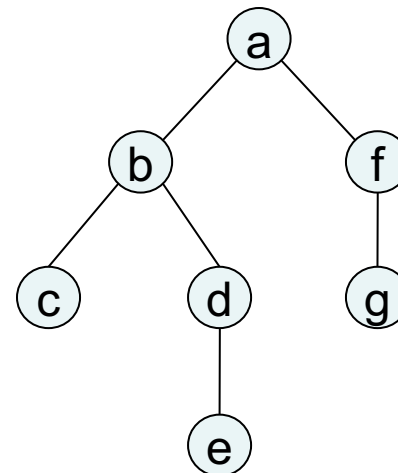
`explore(G,v)` returns the connected component containing `v`.

To explore the rest of the graph, restart `explore()` elsewhere.

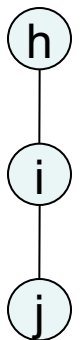
DFS decomposes an undirected graph into its connected components!

```
procedure dfs(G)
  for all v in V:
    visited[v] = false
  for all v in V:
    if not visited[v]:
      explore(G,v)
```

`explore(G,a)`



`explore(G,h)`



Running time analysis

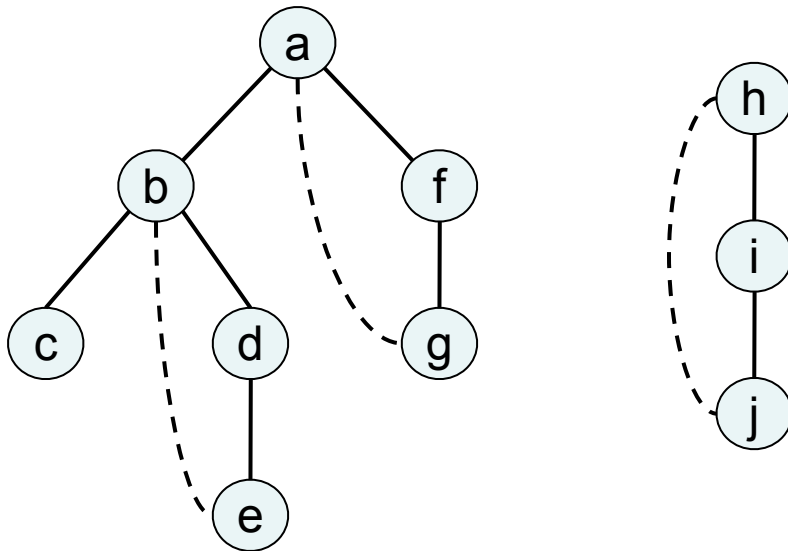
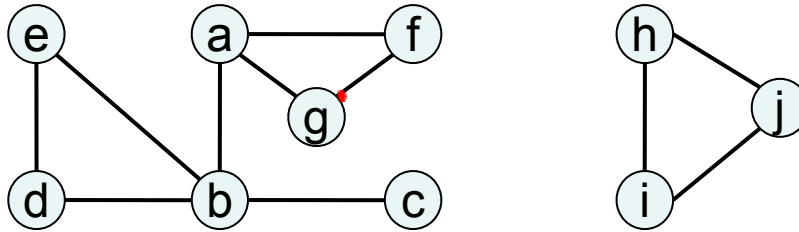
```
procedure explore(G,v)
visited[v] = true
for each edge (v,u) in E:
    if not visited[u]:
        explore(G,u)
```

```
procedure dfs(G)
for all v in V:
    visited[v] = false
for all v in V:
    if not visited[v]:
        explore(G,v)
```

How long does dfs(G) take?

explore(G,v) is called exactly once for each node v.

DFS search forest



Terminology:

DFS search forest consisting of
two *DFS search trees*

- *tree edge*: traversed by DFS
- - - *back edge*: not traversed (led to a node already visited)