Depth-First Search (DFS)

- Another archetype for many important graph algorithms

Depth-First Search (DFS)

- Another archetype for many important graph algorithms
- Methodically explore *every* vertex and *every* edge

Depth-First Search (DFS)

- Another archetype for many important graph algorithms
- Methodically explore *every* vertex and *every* edge
- Input: $G = (V, E)$

Depth-First Search (DFS)

- Another archetype for many important graph algorithms
- Methodically explore *every* vertex and *every* edge
- Input: $G = (V, E)$

  Output: (1) two timestamps for every $v \in V$
  $d[v] =$ when v is first discovered.
  $f[v] =$ when v is finished.
  (2) classification of edges

DFS

- Basic idea:

DFS

- Basic idea:
  - *go as deep as possible, then "back up",*

DFS

- Basic idea:
  - *go as deep as possible, then "back up",*
  - edges are explored out of the most recently discovered vertex v that still have unexplored edges leaving,

DFS

- Basic idea:
  - *go as deep as possible, then "back up",*
  - edges are explored out of the most recently discovered vertex v that still have unexplored edges leaving,
  - when all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discoverd.

DFS

- Basic idea:
  - *go as deep as possible, then "back up",*
  - edges are explored out of the most recently discovered vertex v that still have unexplored edges leaving,
  - when all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discoverd.
- Three-color code for search status of vertices

DFS

- Basic idea:
  - *go as deep as possible, then "back up",*
  - edges are explored out of the most recently discovered vertex v that still have unexplored edges leaving,
  - when all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discoverd.
- Three-color code for search status of vertices
  - **White** = a vertex is undiscovered
  - **Gray** = a vertex is discovered, but its processing is incomplete
  - **Black** = a vertex is discovered, and its processing is complete

DFS

```
DFS(G)    // main routine    :    DFS-Visit(u)    // subroutine
for each vertex u in V       :    color[u] = ‘‘gray’’
   color[u] = ‘‘white’’      :    time = time + 1
endfor                       :    d[u] = time
time = 0                     :    for each v in Adj[u]
for each vertex u in V       :       if color[v] = ‘‘white’’
   if color[u] = ‘‘white’’   :          DFS-Visit(v)
      DFS-Visit(u)           :       endif
   endif                     :    end for
endfor                       :    color[u] = ‘‘black’’
// end of main routine       :    time = time + 1
                                   f[u] = time
                                   // end of subroutine
```

DFS

```
DFS(G)    // main routine    :   DFS-Visit(u)    // subroutine
for each vertex u in V       :   color[u] = ‘‘gray’’
   color[u] = ‘‘white’’      :   time = time + 1
endfor                       :   d[u] = time
time = 0                     :   for each v in Adj[u]
for each vertex u in V       :      if color[v] = ‘‘white’’
   if color[u] = ‘‘white’’   :         DFS-Visit(v)
      DFS-Visit(u)           :      end if
   endif                     :   end for
endfor                       :   color[u] = ‘‘black’’
// end of main routine       :   time = time + 1
                             :   f[u] = time
                             :   // end of subroutine
```

DFS

Remarks:

- Vertices, from which exploration is incomplete, are proceessed in a LIFO stack.

DFS

Remarks:

- Vertices, from which exploration is incomplete, are proceessed in a LIFO stack.
- Running time: $\Theta(|V| + |E|)$

DFS

Remarks:

- Vertices, from which exploration is incomplete, are proceessed in a LIFO stack.
- Running time: $\Theta(|V| + |E|)$

  not big-O since guaranteed to examine every vertex and edge.

DFS

Remarks:

- Vertices, from which exploration is incomplete, are proceessed in a LIFO stack.
- Running time: $\Theta(|V| + |E|)$

  not big-O since guaranteed to examine every vertex and edge.
- For more properties of DFS, see pp.606-608 of [CLRS,3rd ed.]

DFS

Classification of edges:

- **T** = Tree edge = encounter new vertex (*gray to white*)
- **B** = Back edge = from descendant to ancestor (*gray to gray*)
- **F** = Forward edge = from ancestor to descendant (*gray to black*)
- **C** = Cross edge = any other edges (between trees and subtrees) (*gray to black*)

Note: In an undirected graph, there may be some ambiguity since edge (u,v) and (v,u) are the same edge. Classify by the first type that matches.

DFS

```
DFS(G)    // main routine    :   DFS-Visit(u)    // subroutine
for each vertex u in V       :   color[u] = ‘‘gray’’
   color[u] = ‘‘white’’      :   time = time + 1
endfor                       :   d[u] = time
time = 0                     :   for each v in Adj[u]
for each vertex u in V       :      if color[v] = ‘‘white’’
   if color[u] = ‘‘white’’   :         DFS-Visit(v)
      DFS-Visit(u)           :      end if
   endif                     :   end for
endfor                       :   color[u] = ‘‘black’’
// end of main routine       :   time = time + 1
                             :   f[u] = time
                             :   // end of subroutine
```

Classification of edges

T = Tree edge = encounter new vertex (*gray to white*)

B = Back edge = from descendant to ancestor (*gray to gray*)

F = Forward edge = from ancestor to descendant (*gray to black*)

C = Cross edge = any other edges (between trees and subtrees) (*gray to black*)

DFS vs. BFS

1. DFS: vertices from which the exploring is incomplete are processed in a LIFO order (stack)

DFS vs. BFS

1. DFS: vertices from which the exploring is incomplete are processed in a LIFO order (stack)

   BFS: vertices to be explored are organized in a FIFO order (queue)

DFS vs. BFS

1. DFS: vertices from which the exploring is incomplete are processed in a LIFO order (stack)

   BFS: vertices to be explored are organized in a FIFO order (queue)

2. DFS contains two processing opportunities for each vertex v, when it is "discovered" and when it is "finished"

DFS vs. BFS

1. DFS: vertices from which the exploring is incomplete are processed in a LIFO order (stack)

   BFS: vertices to be explored are organized in a FIFO order (queue)

2. DFS contains two processing opportunities for each vertex v, when it is "discovered" and when it is "finished"

   BFS contains only one processing opportunity for each vertex v, and then it is dequeued