EE 355 Unit 18

DFS and Topological Sort

Mark Redekopp
Topological Sort

• Given a graph of dependencies (tasks, prerequisites, etc.) topological sort creates a consistent ordering of tasks (vertices) where no dependencies are violated

• Many possible valid topological orderings exist
  – EE 109L, EE 209, EE 354L, EE 454L, EE 457, MATH 245, PHYS 152, EE 202L,…
  – MATH 245, EE 109L, PHYS 152L, EE 154,…
Topological Sort

- Another example
  - Getting dressed
- More Examples:
  - Project management scheduling
  - Build order in a Makefile or other compile project
  - Cooking using a recipe
  - Instruction execution on an out-of-order pipelined CPU
  - Production of output values in a simulation of a combinational gate network

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/topoSort.htm
Topological Sort

- Does breadth-first search work?
  - No. What if we started at EE200...
  - We'd go to EE 202L before PHYS 152
- All parent nodes need to be completed before any child node
- BFS only guarantees *some* parent has completed before child
- Turns out a Depth-First Search will be part of our solution
Depth First Search

- Explores ALL children before completing a parent
  - Note BFS completes a parent before ANY children
- For DFS let us assign:
  - A start time when the node is first found
  - A finish time when a node is completed
- If we look at our nodes in reverse order of finish time (i.e. last one to finish back to first one to finish) we arrive at a...
  - Topological ordering!!!
DFS & Topological Sort

- Imagine we flipped our dependencies (edge direction) and performed a DFS starting at nodes with no incoming edges.
- Now as nodes “finish/complete” we could enter them in a list and that list would provide a valid topological ordering.
DFS Algorithm

• Visit a node
  – Mark as visited
  – For each visited neighbor, visit it and perform DFS on all of their children
  – Only then, mark as finished

• DFS is recursive!!

• If cycles in the graph, ensure we don’t get caught visiting neighbors endlessly
  – Color them as we go
  – White = unvisited,
  – Gray = visited but not finished
  – Black = finished

DFS (G)
1  for each vertex u
2    u.color = WHITE
3    u.pred = nil
4    time = 0
5    finish_list = empty_list
6  for each vertex u do
7      if u.color == WHITE then
8          DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1    u.color = GRAY
2    u.start = ++time
3    for each vertex v in Adj(u) do
4      if v.color = WHITE then
5        v.pred = u
6        DFS-Visit (G, v)
7    u.color = BLACK
8    finish_list.append(u)
Simplified DFS for Topo-sort

- For topological sort, all we care about is finish time
- If in line 4 of toposort() we simply iterate through goal/output nodes as our start locations then the ordering in the finish_list will be a correct toposort with input nodes first...
- This is a recursive solution
  - We can have an iterative solution shown later...

```
Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)
```

```
DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5    u.color = BLACK
6  finish_list.append(u)
```
## Depth First-Search

Toposort(G)
1. for each vertex u
2. u.color = WHITE
3. finish_list = empty_list
4. for each vertex u do
5. if u.color == WHITE then
6. DFS-Visit (G, u, finish_list)

<table>
<thead>
<tr>
<th>DFS-Visit (G, u)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. u.color = GRAY</td>
</tr>
<tr>
<td>2. for each vertex v in Adj(u) do</td>
</tr>
<tr>
<td>3. if v.color = WHITE then</td>
</tr>
<tr>
<td>4. DFS-Visit (G, v)</td>
</tr>
<tr>
<td>5. u.color = BLACK</td>
</tr>
<tr>
<td>6. finish_list.append(u)</td>
</tr>
</tbody>
</table>

![Graph Diagram](image_url)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Toposort(G)
1  for each vertex u
2   u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5     if u.color == WHITE then
6       DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3     if v.color = WHITE then
4       DFS-Visit (G, v)
5  u.color = BLACK
6  finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list:

DFS-Visit(G,g):
DFS-Visit(G,f):
DFS-Visit(G,d):
DFS-Visit(G,a):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list:
- h, g

DFS-Visit(G,g):
- h, g
DFS-Visit(G,f):
- d, f, g
DFS-Visit(G,d):
- g
DFS-Visit(G,a):
- a
Depth First-Search

Toposort(G)
1  for each vertex u
2   u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5   if u.color == WHITE then
6     DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5   u.color = BLACK
6  finish_list.append(u)

Finish_list:
- h
- g
- f

DFS-Visit(G,f):
DFS-Visit(G,d):
DFS-Visit(G,a):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
# Depth First-Search

**Toposort(G)**
1. for each vertex \( u \)
2. \( u.color = \text{WHITE} \)
3. \( \text{finish_list} = \text{empty_list} \)
4. for each vertex \( u \) do
5. if \( u.color == \text{WHITE} \) then
6. \( \text{DFS-Visit (G, u, finish_list)} \)

**DFS-Visit (G, u)**
1. \( u.color = \text{GRAY} \)
2. for each vertex \( v \) in \( \text{Adj}(u) \) do
3. if \( v.color = \text{WHITE} \) then
4. \( \text{DFS-Visit (G, v)} \)
5. \( u.color = \text{BLACK} \)
6. \( \text{finish_list.append (u)} \)

Finish_list:
- h, g, f, d

DFS-Visit(G,a):

```
1  for each vertex u
2   u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1   u.color = GRAY
2   for each vertex v in Adj(u) do
3     if v.color = WHITE then
4       DFS-Visit (G, v)
5   u.color = BLACK
6  finish_list.append(u)
```
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list:
- h,
- g,
- f,
- d

DFS-Visit(G,e):

DFS-Visit(G,c):

DFS-Visit(G,a):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5  u.color = BLACK
6  finish_list.append(u)

Finish_list:
- h, g, f, d, e, c

DFS-Visit(G,c):
- a, b, c
DFS-Visit(G,a):
- c, d, e, f, g, h, b, a
Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list: h, g, f, d, e, c, a

May iterate through many complete vertices before finding b to launch a new search from

DFS-Visit(G,b):
Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list:

h, g, f, d, e, c, a, b

DFS-Visit(G,b):

Depth First-Search
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

Finish_list:

h, g, f, d, e, c, a, b
With Cycles in the graph

ANOTHER EXAMPLE
Toposort(G)
1  for each vertex u
2   u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5   if u.color == WHITE then
6     DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1   u.color = GRAY
2   for each vertex v in Adj(u) do
3     if v.color = WHITE then
4       DFS-Visit (G, v)
5   u.color = BLACK
6  finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5  u.color = BLACK
6  finish_list.append(u)
Depth First-Search

Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5    u.color = BLACK
6  finish_list.append(u)

DFS-Visit(G,b):
DFS-Visit(G,c):
DFS-Visit(G,a):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit(G, u, finish_list)

DFS-Visit(G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit(G, v)
5 u.color = BLACK
6 finish_list.append(u)

<table>
<thead>
<tr>
<th>DFS-Visit(G,h):</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS-Visit(G,b):</td>
</tr>
<tr>
<td>DFS-Visit(G,c):</td>
</tr>
<tr>
<td>DFS-Visit(G,a):</td>
</tr>
</tbody>
</table>
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5  u.color = BLACK
6  finish_list.append(u)
**Depth First-Search**

Toposort(G)
1. for each vertex u
2. u.color = WHITE
3. finish_list = empty_list
4. for each vertex u do
5.   if u.color == WHITE then
6.     DFS-Visit (G, u, finish_list)

**DFS-Visit (G, u)**
1. u.color = GRAY
2. for each vertex v in Adj(u) do
3.   if v.color = WHITE then
4.     DFS-Visit (G, v)
5. u.color = BLACK
6. finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

Toposort(G)
1  for each vertex u
2    u.color = WHITE
3  finish_list = empty_list
4  for each vertex u do
5    if u.color == WHITE then
6      DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1  u.color = GRAY
2  for each vertex v in Adj(u) do
3    if v.color = WHITE then
4      DFS-Visit (G, v)
5  u.color = BLACK
6  finish_list.append(u)

DFS-Visit(G,d):
DFS-Visit(G,f):
DFS-Visit(G,g):
DFS-Visit(G,h):
DFS-Visit(G,b):
DFS-Visit(G,c):
DFS-Visit(G,a):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)

DFSQ:

DFS-Visit(G,a):
DFS-Visit(G,b):
DFS-Visit(G,c):
DFS-Visit(G,d):
DFS-Visit(G,e):
DFS-Visit(G,f):
DFS-Visit(G,g):
DFS-Visit(G,h):
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
**Depth First-Search**

Toposort(G)
1. for each vertex u
2. u.color = WHITE
3. finish_list = empty_list
4. for each vertex u do
5.    if u.color == WHITE then
6.        DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1. u.color = GRAY
2. for each vertex v in Adj(u) do
3.    if v.color = WHITE then
4.        DFS-Visit (G, v)
5.    u.color = BLACK
6.    finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
Depth First-Search

**Toposort(G)**
1. for each vertex u
2. u.color = WHITE
3. finish_list = empty_list
4. for each vertex u do
5. if u.color == WHITE then
6. DFS-Visit (G, u, finish_list)

**DFS-Visit (G, u)**
1. u.color = GRAY
2. for each vertex v in Adj(u) do
3. if v.color = WHITE then
4. DFS-Visit (G, v)
5. u.color = BLACK
6. finish_list.append(u)
Depth First-Search

Toposort(G)
1 for each vertex u
2 u.color = WHITE
3 finish_list = empty_list
4 for each vertex u do
5 if u.color == WHITE then
6 DFS-Visit (G, u, finish_list)

DFS-Visit (G, u)
1 u.color = GRAY
2 for each vertex v in Adj(u) do
3 if v.color = WHITE then
4 DFS-Visit (G, v)
5 u.color = BLACK
6 finish_list.append(u)
BFS vs. DFS Algorithm

• BFS and DFS are more similar than you think
  – Do we use a FIFO/Queue (BFS) or LIFO/Stack (DFS) to store vertices as we find them

**BFS-Visit (G, start_node)**

```plaintext
1 for each vertex u
2 u.color = WHITE
3 u.pred = nil
4 bfsq = new deque
5 bfsq.push_back(start_node)
6 while bfsq not empty
7 u = bfsq.pop_front()
8 if u.color == WHITE
9 u.color = GRAY
10 foreach vertex v in Adj(u) do
11 bfsq.push_back(v)
```

**DFS-Visit (G, start_node)**

```plaintext
1 for each vertex u
2 u.color = WHITE
3 u.pred = nil
4 dfsq = new deque
5 dfsq.push_back(start_node)
6 while dfsq not empty
7 u = dfsq.pop_back()
8 if u.color == WHITE
9 u.color = GRAY
10 foreach vertex v in Adj(u) do
11 dfsq.push_back(v)
```
ITERATIVE VERSION
Depth First-Search

DFS \((G,s)\)
1. for each vertex \(u\)
2. \(u.\text{color} = \text{WHITE}\)
3. \(\text{dfsq} = \text{new deque}\)
4. \(\text{dfsq}.\text{push}\_\text{back}(s)\)
5. while \(\text{dfsq}\) not empty
6. \(u = \text{dfsq}.\text{back}()\)
7. if \(u.\text{color} == \text{WHITE}\) then
8. \(u.\text{color} = \text{GRAY}\)
9. foreach vertex \(v\) in \(\text{Adj}(u)\) do
10. if \(v.\text{color} == \text{WHITE}\)
11. \(\text{dfsq}.\text{push}\_\text{back}(v)\)
12. else if \(u.\text{color} == \text{GRAY}\)
13. \(u.\text{color} = \text{BLACK}\)
14. \(\text{dfsq}.\text{pop}\_\text{back}()\)

DFSQ: \(a\)
Depth First-Search

DFS (G,s)
1    for each vertex u
2        u.color = WHITE
3    dfsq = new deque
4    dfsq.push_back(s)
5    while dfsq not empty
6        u = dfsq.back()
7        if u.color == WHITE then
8            u.color = GRAY
9            foreach vertex v in Adj(u) do
10               if v.color == WHITE
11                   dfsq.push_back(v)
12        else if u.color == GRAY
13            u.color = BLACK
14        dfsq.pop_back()

DFSQ:  

a c e
Depth First-Search

DFS (G,s)

1. for each vertex u
2. u.color = WHITE
3. dfsq = new deque
4. dfsq.push_back(s)
5. while dfsq not empty
6. u = dfsq.back()
7. if u.color == WHITE then
8. u.color = GRAY
9. foreach vertex v in Adj(u) do
10. if v.color == WHITE
11. dfsq.push_back(v)
12. else if u.color == GRAY
13. u.color = BLACK
14. dfsq.pop_back()
Depth First-Search

DFS (G,s)
1  for each vertex u
2    u.color = WHITE
3  dfsq = new deque
4  dfsq.push_back(s)
5  while dfsq not empty
6    u = dfsq.back()
7    if u.color == WHITE then
8      u.color = GRAY
9      foreach vertex v in Adj(u) do
10         if v.color == WHITE
11            dfsq.push_back(v)
12      else if u.color == GRAY
13        u.color = BLACK
14    dfsq.pop_back()
Depth First-Search

DFS \((G,s)\)

\begin{verbatim}
1  for each vertex u
2      u.color = WHITE
3  dfsq = new deque
4  dfsq.push_back(s)
5  while dfsq not empty
6      u = dfsq.back()
7      if u.color == WHITE then
8          u.color = GRAY
9      foreach vertex v in Adj(u) do
10         if v.color == WHITE
11            dfsq.push_back(v)
12      else if u.color == GRAY
13          u.color = BLACK
14      dfsq.pop_back()
\end{verbatim}

DFSQ:  \(\text{DFSQ: } a \ c \ e \ c \ f \ d \ g \ c \ h\)
Depth First-Search

DFS (G,s)
1 for each vertex u
2 \text{u.color} = \text{WHITE}
3 \text{dfsq} = \text{new deque}
4 \text{dfsq.push_back}(s)
5 while \text{dfsq} not empty
6 \text{u} = \text{dfsq.back}()
7 if \text{u.color} == \text{WHITE} then
8 \text{u.color} = \text{GRAY}
9 foreach vertex v in Adj(u) do
10 if v.color == \text{WHITE}
11 \text{dfsq.push_back}(v)
12 else if u.color == \text{GRAY}
13 \text{u.color} = \text{BLACK}
14 \text{dfsq.pop_back}()

\text{DFSQ:} \quad \text{a} \quad \text{c} \quad \text{e} \quad \text{c} \quad \text{f} \quad \text{d} \quad \text{g} \quad \text{c} \quad \text{h} \quad \text{b}
**Depth First-Search**

**DFS (G,s)**

1. for each vertex u
2. u.color = WHITE
3. dfsq = *new* deque
4. dfsq.push_back(s)
5. while dfsq not empty
6. u = dfsq.back()
7. if u.color == WHITE then
8.   u.color = GRAY
9.   foreach vertex v in Adj(u) do
10.   if v.color == WHITE
11.     dfsq.push_back(v)
12.   else if u.color == GRAY
13.     u.color = BLACK
14.   dfsq.pop_back()
**Depth First-Search**

DFS (G,s)
1. for each vertex u
2. u.color = WHITE
3. dfsq = *new deque*
4. dfsq.push_back(s)
5. while dfsq not empty
6. u = dfsq.back()
7. if u.color == WHITE then
8. u.color = GRAY
9. foreach vertex v in Adj(u) do
10. if v.color == WHITE
11. dfsq.push_back(v)
12. else if u.color == GRAY
13. u.color = BLACK
14. dfsq.pop_back()

DFSQ: a c e c f d g c h b c d
Depth First-Search

DFS (G,s)
1   for each vertex u
2       u.color = WHITE
3   dfsq = new deque
4   dfsq.push_back(s)
5   while dfsq not empty
6       u = dfsq.back()
7       if u.color == WHITE then
8           u.color = GRAY
9           foreach vertex v in Adj(u) do
10              if v.color == WHITE
11                  dfsq.push_back(v)
12       else if u.color == GRAY
13           u.color = BLACK
14       dfsq.pop_back()

DFSQ:  

```
 a b c d e f g h
```
Depth First-Search

DFS (G,s)
1. for each vertex u
2. u.color = WHITE
3. dfsq = new deque
4. dfsq.push_back(s)
5. while dfsq not empty
6. u = dfsq.back()
7. if u.color == WHITE then
8. u.color = GRAY
9. foreach vertex v in Adj(u) do
10. if v.color == WHITE
11. dfsq.push_back(v)
12. else if u.color == GRAY
13. u.color = BLACK
14. dfsq.pop_back()
Depth First-Search

DFS (G,s)
1  for each vertex u
2    u.color = WHITE
3  dfsq = new deque
4  dfsq.push_back(s)
5  while dfsq not empty
6    u = dfsq.back()
7    if u.color == WHITE then
8        u.color = GRAY
9        foreach vertex v in Adj(u) do
10           if v.color == WHITE
11              dfsq.push_back(v)
12        else if u.color == GRAY
13            u.color = BLACK
14        dfsq.pop_back()
Depth First-Search

DFS (G,s)
1  for each vertex u
2      u.color = WHITE
3  dfsq = new deque
4  dfsq.push_back(s)
5  while dfsq not empty
6      u = dfsq.back()
7      if u.color == WHITE then
8          u.color = GRAY
9      foreach vertex v in Adj(u) do
10         if v.color == WHITE
11             dfsq.push_back(v)
12         else if u.color == GRAY
13            u.color = BLACK
14     dfsq.pop_back()
Depth First-Search

DFS \((G, s)\)

1. for each vertex \(u\)
2. \(u\).color = WHITE
3. \(\text{dfsq} = \text{new deque}\)
4. \(\text{dfsq}.\text{push_back}(s)\)
5. while \(\text{dfsq}\) not empty
6. \(u = \text{dfsq}.\text{back}()\)
7. if \(u\).color == WHITE then
8. \(u\).color = GRAY
9. foreach vertex \(v\) in \(\text{Adj}(u)\) do
10. if \(v\).color == WHITE
11. \(\text{dfsq}.\text{push_back}(v)\)
12. else if \(u\).color == GRAY
13. \(u\).color = BLACK
14. \(\text{dfsq}.\text{pop_back}()\)

DFSQ: [Diagram of a graph with vertices labeled a, b, c, d, e, f, g, h]