Depth-First Search / Traversal

Alan G. Labouseur, Ph.D.
Alan.Labouseur@Marist.edu
Graphs
Graphs

Graph . . .

as Linked Objects

<table>
<thead>
<tr>
<th>id</th>
<th>processed</th>
<th>neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>false</td>
<td>[2,5,6]</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>[1,3,5,6]</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>[2,4]</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
<td>[3,5]</td>
</tr>
<tr>
<td>5</td>
<td>false</td>
<td>[1,2,4,6,7]</td>
</tr>
<tr>
<td>6</td>
<td>false</td>
<td>[1,2,5,7]</td>
</tr>
<tr>
<td>7</td>
<td>false</td>
<td>[5,6]</td>
</tr>
</tbody>
</table>
Graphs

Graph . . .

as Linked Objects

Adjacency List

[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6
Graphs

Graph . . .

as Linked Objects

Adjacency List

Matrix

id  processed  false  neighbors
1          [2,5,6]
2          [1,3,5,6]
3          [2,4]
4          [5,6,7]
5          [1,2,4,6,7]
6          [1,2,5,7]
7          [5,6]
Depth-First Search / Traversal

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
Depth-First Search / Traversal

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

Recursion.
Yay!
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack
myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

Runtime Stack
**Depth-First Search / Traversal**

```plaintext
myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1
Depth-First Search / Traversal

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

myGraph.DFS(vertex1)
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack
Depth-First Search / Traversal

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

myGraph.DFS(vertex1)

Adjacency List

```
[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6
```

Adjacency List

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

Runtime Stack
Depth-First Search / Traversal

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

myGraph.DFS(vertex1)

Recursive call.
Push return address on the runtime stack.
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
if (not v.processed)
  print(v.id)
  v.processed := true
endif
for n in v.neighbors[]
  if (not n.processed)
    DFS(n)
  endif
endfor
endproc

1
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```java
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2

RunFme	Stack
return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RunFme Stack

return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

![Graph Diagram]

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

1 2
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

Adjacency List

1 2

RunFme Stack

[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6

Adjacency List

return address

Runtime Stack

1 2
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
 Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1 2
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

Depth-First Search / Traversal

myGraph.DFS(vertex1)
Depth-First Search / Traversal

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

myGraph.DFS(vertex1)
Depth-First Search / Traversal

```
myGraph.DFS(vertex1)
```

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Recursive call. Push return address on the runtime stack.
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack

return	address

return	address

Runtime Stack

1 2
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
**Depth-First Search / Traversal**

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

```
return address
return address
Runtime Stack
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2 3

Runtime Stack

RunFme	Stack
treturn	address
treturn
taddress
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme	Stack

return	address

return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

Adjacency List
[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6

Adjaceny List

Runtime Stack

return address

return address
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack
return	address
treturn	address

Runtime Stack

1 2 3
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```prolog
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

Run Fme Stack

return address

return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
if (not v.processed)
    print(v.id)
    v.processed := true
endif
for n in v.neighbors[]
    if (not n.processed)
        DFS(n)
    endif
endfor
endproc

RunFme	Stack
return	address
treturn	address

Runtime Stack
Depth-First Search / Traversal

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
myGraph.DFS(vertex1)
```

![Graph diagram]
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```java
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack

return	address

return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

Runtime Stack

1 2 3
**Depth-First Search / Traversal**

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

### Sample Execution

1. **Initial State**
   - Runtime Stack:
     - return address

2. **First Frame**
   - Entry to DFS(vertex1)

3. **Second Frame**
   - Entry to DFS(2)
     - Print(2)
     - Mark 2 as processed
     - Push 2 onto stack

4. **Third Frame**
   - Entry to DFS(3)
     - Print(3)
     - Mark 3 as processed
     - Push 3 onto stack

5. **Fourth Frame**
   - Entry to DFS(6)
     - Print(6)
     - Mark 6 as processed
     - Push 6 onto stack

6. **Fifth Frame**
   - Entry to DFS(5)
     - Print(5)
     - Mark 5 as processed
     - Push 5 onto stack

7. **Sixth Frame**
   - Entry to DFS(4)
     - Print(4)
     - Mark 4 as processed
     - Push 4 onto stack

8. **Seventh Frame**
   - Entry to DFS(1)
     - Print(1)
     - Mark 1 as processed
     - Push 1 onto stack

**Final State**

- Runtime Stack:
  - return address

- Stack contents: 1, 2, 3, 4, 5, 6, 7
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
    v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

myGraph.DFS(vertex1)
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4

Runtime Stack

return address
return address
return address
return address
Depth-First Search / Traversal

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
myGraph.DFS(vertex1)
```

```
1 2 3 4
```

Runtime Stack

```
return address
return address
return address
```

Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4

Return address
Return address
Return address

Runtime Stack

Adjacency List

[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6

45
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1 2 3 4
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

1 2 3 4

return address
return address
return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme	Stack

return	address
return	address
return	address

Runtime Stack

1 2 3 4
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme Stack

1 2 3 4

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
return address
return address
return address
return address
Runtime Stack
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

```
return address
return address
return address
return address

1 2 3 4
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
if (not v.processed)
    print(v.id)
endif
for n in v.neighbors[]
    if (not n.processed)
        DFS(n)
    endif
endfor
endproc
```

Run Fme Stack

1 2 3 4 5

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```java
proc DFS(fromVertex v)
   if (not v.processed)
      print(v.id)
      v.processed := true
   endif
   for n in v.neighbors[
      if (not n.processed)
         DFS(n)
      endif
   endfor
endproc
```

Runtime Stack

1 2 3 4 5
Depth-First Search / Traversal

```plaintext
myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
return address
return address
return address
return address

Runtime Stack
```
Depth-First Search / Traversal

proc DFS(fromVertex v)
   if (not v.processed)
      print(v.id)
      v.processed := true
   endif
   for n in v.neighbors[]
      if (not n.processed)
         DFS(n)
      endif
   endfor
endproc

myGraph.DFS(vertex1)

Adjacency List
[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme Stack

Runtime Stack
**Depth-First Search / Traversal**

```
myGraph.DFS(vertex1)
```

```prolog
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack
---
return	address
return	address
return	address
return	address
return	address

**Runtime Stack**

1 2 3 4 5
### Depth-First Search / Traversal

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

myGraph.DFS(vertex1)

#### RunFme	Stack
1 2 3 4 5

```
1 2 3 4 5
```

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

RunFme	Stack

Runtime Stack

1 2 3 4 5
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RUNTIME STACK

1 2 3 4 5
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme	Stack
return	address
return	address
return	address
return	address
return	address
RunFme	Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme	Stack
return	address
return	address
return	address
return	address
return	address
run time stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme	Stack
return	address
return	address
return	address
return	address
return	address

Runtime Stack

1 2 3 4 5 6
Depth-First Search / Traversal

myGraph.DFS(vertex1)

1 2 3 4 5 6

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

Runtime Stack

1 2 3 4 5 6
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1 2 3 4 5 6
### Depth-First Search / Traversal

```plaintext
myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
Adjacency List
[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6
```

### Runtime Stack

```
return address
return address
return address
return address
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RunFme	Stack
return	address
return	address
return	address
return	address
RunFme	Stack
return	address
return	address
return	address
return	address

Runtime Stack

1 2 3 4 5 6
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1 2 3 4 5 6
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

Runtime Stack

1 2 3 4 5 6
Depth-First Search / Traversal

```
myGraph.DFS(vertex1)
```

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Runtime Stack

1 2 3 4 5 6

RunFme	Stack
return	address
return	address
return	address
return	address
return	address
return	address

1 2 3 4 5 6
Depth-First Search / Traversal

```plaintext
myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme
Stack
return
return
return
return
return
return

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

Run-time stack:

1 2 3 4 5 6
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

Run Fme

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4 5 6 7

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack
return	address
return	address
return	address
return	address
return	address
return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

[1] 2 5 6
[2] 1 3 5 6
[3] 2 4
[4] 3 5
[5] 1 2 4 6 7
[6] 1 2 5 7
[7] 5 6

Adjacency List

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RunFme	Stack
return	address
return	address
return	address
return	address
return	address
return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

![Runtime Stack](image)
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

End of most recent recursive call. Pop return address off the runtime stack and return.
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
   if (not v.processed)
      print(v.id)
      v.processed := true
   endif
   for n in v.neighbors[]
      if (not n.processed)
         DFS(n)
      endif
   endfor
endproc

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
**Depth-First Search / Traversal**

```plaintext
myGraph.DFS(vertex1)
```

**proc DFS(fromVertex v)**

```plaintext
if (not v.processed)
    print(v.id)
    v.processed := true
endif
for n in v.neighbors[
    if (not n.processed)
        DFS(n)
    endif
endfor
endproc
```

Pop return address off the runtime stack and return.
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

Runtime Stack

1 2 3 4 5 6 7
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RunFme	Stack
return	address
return	address
return	address
return	address

1 2 3 4 5 6 7
Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RunFme	Stack
return	address
return	address
return	address
return	address

Runtime Stack

1 2 3 4 5 6 7
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

RUNTIME STACK

1 2 3 4 5 6 7


Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2 3 4 5 6 7

Runtime Stack

```
return address
return address
return address
return address
return address
```

Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
if (not v.processed)
    print(v.id)
    v.processed := true
endif
for n in v.neighbors[]
    if (not n.processed)
        DFS(n)
    endif
endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

RUNTIME STACK

1 2 3 4 5 6 7

return address
return address
return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack

return	address

return	address

1 2 3 4 5 6 7

Runtime Stack
myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

RunFme Stack

Runtime Stack

1 2 3 4 5 6 7
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4 5 6 7

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack
return	address
test
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2 3 4 5 6 7

return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack

return address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4 5 6 7

return address
Runtime Stack
**Depth-First Search / Traversal**

```latex
myGraph.DFS(vertex1)

\[
\begin{array}{c}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]
```

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

```
return address

Runtime Stack

```
**Depth-First Search / Traversal**

```plaintext
proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc
```

myGraph.DFS(vertex1)

1 2 3 4 5 6 7

RunFme	Stack
return	address

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

1 2 3 4 5 6 7

proc DFS(fromVertex v)
  if (not v.processed)
    print(v.id)
    v.processed := true
  endif
  for n in v.neighbors[]
    if (not n.processed)
      DFS(n)
    endif
  endfor
endproc

return address
Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2 3 4 5 6 7
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```plaintext
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

1 2 3 4 5 6 7

Runtime Stack
Depth-First Search / Traversal

\[\text{proc } \text{DFS(fromVertex } v)\]
\[\begin{aligned}
&\text{if (not } v.\text{processed)} \\
&\quad \text{print}(v.\text{id}) \\
&\quad v.\text{processed} := \text{true} \\
&\text{endif} \\
&\text{for } n \text{ in } v.\text{neighbors[]} \\
&\quad \text{if (not } n.\text{processed)} \\
&\quad\quad \text{DFS}(n) \\
&\quad \text{endif} \\
&\text{endfor} \\
&\text{endproc}
\]

\[\text{myGraph.DFS(vertex1)}\]

1 2 3 4 5 6 7

Run Fme Stack

1 2 3 4 5 6 7

Runtime Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

```java
proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc
```

RunFme	Stack

1 2 3 4 5 6 7

Runtime Stack
null
Depth-First Search / Traversal

myGraph.DFS(vertex1)

Proc DFS(fromVertex v)
  If (not v.processed)
    Print(v.id)
    V.processed := true
  Endif
  For n in v.neighbors[]
    If (not n.processed)
      DFS(n)
    Endif
  Endfor
Endproc

RunFme
Stack
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4 5 6 7

Runtime Stack
null
Depth-First Search / Traversal

myGraph.DFS(vertex1)

proc DFS(fromVertex v)
    if (not v.processed)
        print(v.id)
        v.processed := true
    endif
    for n in v.neighbors[]
        if (not n.processed)
            DFS(n)
        endif
    endfor
endproc

1 2 3 4 5 6 7