Graph Traversal

BFS

DFS

Ideas/Issues

Appendix

COMP2521 23T3

Graph Traversal

Kevin Luxa

cs2521@cse.unsw.edu.au

graph traversal

bfs and dfs

path checking

path finding
Common problems on graphs:

- Is there a path between two vertices?
- What is the shortest path between two vertices?
- Is the graph connected?
- If we remove an edge, is the graph still connected?
- Which vertices are reachable from a particular vertex?
- Is there a cycle that passes through all vertices?
All of the above problems can be solved by a **systematic exploration** of a graph via its **edges**.

This systematic exploration is called **traversal** or **search**.
Graph Traversal

PROBLEM
Does a path exist between vertices $src$ and $dest$?

Possible approach:
1. examine vertices adjacent to $src$
2. if any of them is $dest$, we’re done!
3. otherwise, check vertices two edges away from $src$
4. repeat looking further and further away from $src$

The above summarises one form of graph traversal.
Two primary methods for graph traversal/search:

**Breadth-first search (BFS)**
- Prioritises visiting all neighbours over path-following
  - “Go wide”
- Implemented iteratively (using a queue)

**Depth-first search (DFS)**
- Prioritises path-following over visiting all neighbours
  - “Go deep”
- Implemented recursively or iteratively (using a stack)
In what order would BFS and DFS visit the vertices of this graph?
Graph Traversal

BFS vs. DFS

Breadth-first search

Depth-first search
Breadth-first search visits vertices in order of distance from the starting vertex. It visits the starting vertex, then the neighbours of the starting vertex, then the neighbours of those neighbours, etc. BFS is implemented iteratively using a queue.
Data structures used in BFS:

- **Visited array**
  - To keep track of which vertices have been visited

- **Predecessor array**
  - To keep track of the predecessor of each vertex
  - The predecessor of $v$ is the vertex from which we reached $v$
    - i.e., the vertex before $v$ on the path to $v$

- **Queue**
  - First-in-first-out data structure
  - Stores unvisited vertices in the order that they should be visited
Algorithm:

1. Create/initialise data structures:
   - Initialise visited array to false
   - Initialise predecessor array to -1
   - Create empty queue

2. Mark starting vertex as visited and enqueue it

3. While the queue is not empty:
   1. Dequeue a vertex
      - Let this vertex be \( v \)
   2. **Explore** \( v \) - that is, for each of \( v \)'s unvisited neighbours:
      1. Mark it as visited
      2. Set its predecessor to \( v \)
      3. Enqueue it
Breadth-First Search
Example

BFS starting at 0

visited
pred
queue
Breadth-First Search Example

BFS starting at 0

Done

visited
pred
queue

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>visited</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>pred</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>queue</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
bfs($G$, $src$):

**Inputs:** graph $G$, starting vertex $src$

create visited array, predecessor array and queue $Q$

**for each** vertex $v$ in $G$:
  
  visited[$v$] = false
  predecessor[$v$] = -1

visited[$src$] = true
enqueue $src$ into $Q$

**while** $Q$ is not empty:

  $v$ = dequeue from $Q$

  **for each** neighbour $w$ of $v$ where visited[$w$] = false:
    visited[$w$] = true
    predecessor[$w$] = $v$
    enqueue $w$ into $Q$
When using a predecessor array in BFS, the predecessor array can double as a visited array

\[
\text{predecessor}[v] = -1 \text{ means } v \text{ is not visited}
\]
Breadth-First Search
Simplification

\[
\text{bfs}(G, src):
\]

**Inputs:** graph \( G \), starting vertex \( src \)

create predecessor array and queue \( Q \)

\[
\text{for each vertex } v \text{ in } G:\n\]
\[
\text{predecessor}[v] = -1\n\]

\[
\text{predecessor}[src] = src \quad // \quad \text{<- mark src as visited}\n\]
enqueue \( src \) into \( Q \)

\[
\text{while } Q \text{ is not empty:}\n\]
\[
v = \text{dequeue from } Q\n\]
\[
\text{for each neighbour } w \text{ of } v \text{ where } \text{predecessor}[w] = -1:\n\]
\[
\text{predecessor}[w] = v\n\]
enqueue \( w \) into \( Q \)
BFS is $O(V + E)$ when using the adjacency list representation:

- Typical queue implementation has $O(1)$ enqueue and dequeue
- Each vertex is visited at most once $\Rightarrow O(V)$
- For each vertex, all of its edges are considered once $\Rightarrow O(E)$
A BFS finds the shortest path between the starting vertex and all other vertices.

- Shortest path in terms of the number of edges

The shortest path between $src$ and $dest$ can be found by tracing backwards through the predecessor array (from $dest$ to $src$).
Example: Shortest path from 0 to 8

Path-Finding with BFS
Example: Shortest path from 0 to 8

Path-Finding with BFS
Example: Shortest path from 0 to 8

```
0 ←→ 5 → 4 → 8
```

Path Finding

```
pred = [-1 0 0 2 5 0 5 5 4 7]
```
Example: Shortest path from 0 to 8

0 → 4 → 8
Example: Shortest path from 0 to 8

0 → 5 → 4 → 8

```
0 1 2 3 4 5 6 7 8 9
```

```
pred [-1 0 0 2 5 0 5 5 4 7]
```
Example: Shortest path from 0 to 8

0 → 5 → 4 → 8
Example: Shortest path from 0 to 8

0 → 5 → 4 → 8

```
0 -1 0 0 2 5 0 5 5 4
1 0 0 0 0 5 5 4 7
2 0 0 0 0 0 0 0 0 0
3 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0 0
```
findPathBfs($G, src, dest$):

**Inputs:** graph $G$, vertices $src$ and $dest$

... BFS starting from $src$ ...

```python
if predecessor[dest] ≠ -1:
    v = dest
    while $v ≠ src$:
        print $v$, "<-"
        $v = predecessor[v]$
    print $src$
```
Depth-first search goes as far down one path as possible until it reaches a dead end, then backtracks until it finds a new path to take, then repeats.

DFS can be implemented recursively or iteratively.
Depth-first search is described recursively as:

1. Mark current vertex as visited
   - The first time, this is the starting vertex
2. For each neighbour of the current vertex:
   1. If it has not been visited:
      1. Recursively traverse starting from that vertex

The recursion naturally induces backtracking.
Recursive Depth-First Search

Pseudocode

dfs\(\left(G, \ src\right)\):

Inputs: graph \(G\), starting vertex \(src\)

create visited array, initialised to false
dfsRec\(\left(G, \ src, \ visited\right)\)

dfsRec\(\left(G, \ v, \ visited\right)\):

Inputs: graph \(G\), vertex \(v\), visited array

\(visited[v] = \text{true} \ // \ "visit" \ v\)

for each neighbour \(w\) of \(v\):

if visited\(\left[w\right]\) = false:

\(\text{dfsRec}\left(G, \ w, \ visited\right)\)
Recursive Depth-First Search

DFS starting at 0

visit order

visited

call stack
Recursive Depth-First Search

Example

Graph Traversal
BFS
DFS
Recursive Pseudocode
Example
Analysis
Path checking
Path finding
Iterative
Ideas/Issues
Appendix

Recursive Depth-First Search

Example

Graph Traversal
BFS
DFS
Recursive Pseudocode
Example
Analysis
Path checking
Path finding
Iterative
Ideas/Issues
Appendix

Done

0
1
2
3
4
5
6
7
8
9

visited
[0] 1 1 1 1 1 1 1 1 1 1

visit order
0 1 5 3 2 4 7 8 9 6

call stack
Recursive DFS is $O(V + E)$ when using the adjacency list representation:

- Each vertex is visited at most once $\Rightarrow O(V)$
  - Function is called on each vertex at most once
- For each vertex, all of its edges are considered once $\Rightarrow O(E)$
Recursive DFS can be adapted to check if a path exists between two vertices.

Idea:

- To check if a path exists between src and dest:
  - If src = dest, then there is a path (the empty path)
  - Otherwise, for each neighbour of src, recursively check if there is a path from that neighbour to dest
Does there exist a path between 0 and 7 in this graph?
Path-Checking with Recursive DFS

Example

Answer: Yes

Graph:
- Nodes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Edges: (0, 2), (0, 5), (2, 3), (3, 1), (3, 5), (5, 1), (5, 6), (4, 5), (4, 7), (7, 8), (8, 9)

Path checking

No path exists between the given vertices.
hasPath($G, src, dest$):

**Inputs:** graph $G$, vertices $src$ and $dest$

**Output:** true if there is a path from $src$ to $dest$
false otherwise

create visited array, initialised to false

return dfsHasPath($G, src, dest, visited$)

dfsHasPath($G, v, dest, visited$):

**Inputs:** graph $G$, vertices $v$ and $dest$, visited array

visited[$v$] = true

if $v = dest$:
    return true

for each neighbour $w$ of $v$:
    if visited[$w$] = false:
        if dfsHasPath($G, w, dest, visited$):
            return true

return false
Path-Checking with Recursive DFS

Analysis

$O(V + E)$ when using the adjacency list representation:

- Algorithm is just a modified recursive DFS with return statements
Knowing whether a path exists can be useful.

Knowing what the path is can be even more useful.

Idea:

- Record the predecessor of each vertex during the DFS
- Trace backwards through the path after the DFS
**Path-Finding with Recursive DFS**

**Pseudocode**

```python
findPath(G, src, dest):
    **Inputs:** graph G, vertices src and dest

    create predecessor array, initialised to -1
    predecessor[src] = src

    if dfsFindPath(G, src, dest, predecessor):
        v = dest
        while v ≠ src:
            print v, "<-"
            v = predecessor[v]
        print src

    findPath(G, src, dest):
```

**Path checking**

**Path finding**

**Ideas/Issues**

**Appendix**
Path-Finding with Recursive DFS

Pseudocode

dfsFindPath(G, v, dest, predecessor):
    if v = dest:
        return true

    for each neighbour w of v:
        if predecessor[w] = -1:
            predecessor[w] = v
            if dfsFindPath(G, w, dest, predecessor):
                return true

    return false
Find a path from 0 to 7
Path-Finding with Recursive DFS

Example

Path found:

Clearly, DFS is not guaranteed to find the shortest path.
DFS can be implemented iteratively.

- Similar to BFS, with a few crucial differences:
  - DFS uses a stack instead of a queue
  - BFS marks a vertex as visited when enqueuing it
  - DFS marks a vertex as visited after popping it from the stack, instead of when pushing it onto the stack
Iterative Depth-First Search

Pseudocode

\[
dfs(G, \ src):
\]

\textbf{Inputs:} graph \( G \), vertex \( src \)

created visited array, predecessor array and stack \( S \)

\textbf{for each} vertex \( v \) in \( G \):
\[
\begin{align*}
\text{visited}[v] &= \text{false} \\
\text{predecessor}[v] &= -1
\end{align*}
\]

push \( src \) onto \( S \)

\textbf{while} \( S \) is not empty:
\[
\begin{align*}
v &= \text{pop from} \ S \\
\text{if} \ \text{visited}[v] &= \text{true}: \\
\text{continue} \ // \ i.e., \ return \ to \ start \ of \ loop
\end{align*}
\]

\text{visited}[v] = \text{true}

\textbf{for each} neighbour \( w \) of \( v \) where \( \text{visited}[w] = \text{false} \):
\[
\begin{align*}
\text{predecessor}[w] &= v \\
\text{push} \ w \ \text{onto} \ S
\end{align*}
\]
Why mark a vertex as visited after popping it, instead of when pushing it?
Iterative DFS is $O(V + E)$ when using the adjacency list representation.

- Typical stack implementation has $O(1)$ push and pop
- Each vertex visited at most once $\Rightarrow O(V)$
- For each vertex, all of its edges are considered $\Rightarrow O(E)$
The edges traversed in a graph traversal form a spanning tree.

Consider the following graph:
A traversal starting at vertex ‘a’ forms the following spanning trees:

Breadth-first search

Depth-first search
If a graph is not connected, a graph traversal starting from a given vertex will not traverse the entire graph.

**Solution**
After initial traversal is complete, perform traversal again on an unvisited vertex, repeat until all vertices are visited.

This produces a spanning forest.
dfs($G$):

Inputs: graph $G$

create predecessor array, initialised to -1

for each vertex $v$ in $G$:
    if predecessor[$v$] = -1:
        dfsRec($G$, $v$, predecessor)

...
https://forms.office.com/r/aPF09YHZ3X
Appendix
BFS Example

BFS starting at 0

Graph
Traversals
DFS
BFS
Ideas/Issues
Appendix

BFS Example
DFS Example
Path-Checking Example
BFS Example

BFS starting at 0

Mark 0 as visited

visited

pred
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

queue
0

BFS starting at 0

Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

Mark 0 as visited
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

Mark 0 as visited
Mark 1 as visited
Mark 2 as visited
Mark 5 as visited
Done exploring 0
p0 is already visited
p5 is already visited
Done exploring 1
p0 is already visited
p1 is already visited
Mark 3 as visited
Done exploring 2
p0 is already visited
p1 is already visited
p3 is already visited
Mark 4 as visited
Mark 6 as visited
Mark 7 as visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
Mark 3 as visited
Mark 4 as visited
Mark 6 as visited
Mark 7 as visited
p5 is already visited
Mark 8 as visited
Done exploring 3
p3 is already visited
p5 is already visited
Mark 9 as visited
Done exploring 4
p3 is already visited
Mark 8 as visited
p5 is already visited
Mark 9 as visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
Mark 8 as visited
p5 is already visited
Mark 9 as visited
p5 is already visited
Mark 8 as visited
Mark 9 as visited
Done exploring 6
p4 is already visited
p5 is already visited
Mark 8 as visited
p5 is already visited
Mark 9 as visited
p5 is already visited
Mark 8 as visited
p7 is already visited
Mark 9 as visited
Done exploring 7
p4 is already visited
p7 is already visited
Mark 9 as visited
p8 is already visited
Mark 9 as visited
Done exploring 8
p7 is already visited
p8 is already visited
Mark 9 as visited
p8 is already visited
Mark 9 as visited
Done exploring 9
Done
BFS Example

Dequeue 0

visited

0 0 0 0 0 0 0 0 0 0 0

pred

-1 -1 -1 -1 -1 -1 -1 -1 -1 -1

queue

0
BFS Example

Explore 0

```
visited  1  0  0  0  0  0  0  0  0  0  0  0
pred    -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
queue   0
```

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done
BFS Example

Explore 0

![Graph](image)

**visited**

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**pred**

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0</strong></td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**queue**

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0</strong></td>
<td>0</td>
</tr>
</tbody>
</table>
**BFS Example**

Explore 0

Mark 1 as visited

```
visited [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
1 1 0 0 0 0 0 0 0 0


queue 0 1
```
BFS Example

Explore 0

Graph:
- Node 0 is visited first.
- Node 2 is visited next.
- Node 5 is visited next.
- Node 3 is visited next.
- Node 1 is visited next.
- Node 6 is visited next.
- Node 7 is visited next.
- Node 8 is visited next.
- Node 9 is visited next.

Visited Table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>visited</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Pred Table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Queue:

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
BFS Example

Explore 0

Mark 2 as visited

```
visited | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
pred    | -1| 0 | 0 | -1| -1| -1| -1| -1| -1| -1| -1
queue   | 0 | 1 | 2
```
**BFS Example**

Explore 0

![BFS Diagram]

- **visited**:
  - 1 1 1 0 0 0 0 0 0 0 0

- **pred**:
  - -1 0 0 -1 -1 -1 -1 -1 -1 -1

- **queue**:
  - 0 1 2
**Graph Traversal**

**BFS Example**

- **Explore 0**
- **Mark 5 as visited**

- **Visited**:
  - 1 1 1 0 0 1 0 0 0 0 0

- **Pred**:
  - -1 0 0 -1 -1 0 -1 -1 -1 -1

- **Queue**:
  - 0 1 2 5
BFS Example

Explore 0

Done exploring 0

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
BFS Example

Dequeue 1

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9

visited

1 1 1 0 0 1 0 0 0 0 0

pred

-1 0 0 -1 -1 0 -1 -1 -1 -1 -1

queue

0 1 2 5
Explore 1

BFS Example

visited:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue: 0 1 2 5
BFS Example

Explore 1

0 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue

0 1 2 5
BFS Example

Explore 1

0 -> 1 -> 5 is already visited

visited: 1 1 1 0 0 1 0 0 0 0 0

pred: -1 0 0 -1 -1 0 -1 -1 -1 -1

queue: 0 1 2 5
BFS Example

Explore 1

Done exploring 1

visited

path:

queue

pred

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

pMark 0 as visited
pMark 1 as visited
pMark 2 as visited
pMark 5 as visited
Done exploring 0
p0 is already visited
p5 is already visited
pMark 3 as visited
Done exploring 1
p0 is already visited
p1 is already visited
pMark 4 as visited
pMark 6 as visited
pMark 7 as visited
Done exploring 2
p2 is already visited
p4 is already visited
p5 is already visited
pMark 8 as visited
Done exploring 3
p3 is already visited
p5 is already visited
pMark 9 as visited
Done exploring 4
p5 is already visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
pMark 8 as visited
Done exploring 6
p4 is already visited
pMark 9 as visited
Done exploring 7
p7 is already visited
p9 is already visited
Done exploring 8
p7 is already visited
p8 is already visited
Done exploring 9
Dequeue 2

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

BFS Example
**BFS Example**

Explore 2

![Graph](https://example.com/graph.png)

- **visited**:
  - [0] 1 1 1 0 0 1 0 0 0 0 0

- **pred**:
  - [-1] 0 0 -1 -1 0 -1 -1 -1 -1

- **queue**:
  - 0 1 2 5
BFS Example

Explore 2

0 is already visited

visited
0 1 1 1 0 0 0 1 0 0 0 0 0

pred
-1 0 0 -1 -1 0 -1 -1 -1 -1 -1

queue
0 1 2 5
BFS Example

Explore 2

0 → 2 → 3

visited:
0 1 1 1 0 0 1 0 0 0 0 0

pred:
-1 0 0 -1 -1 0 -1 -1 -1 -1

queue:
0 1 2 5

BFS starting at 0:
Dequeue 0, explore 0
Dequeue 1, explore 1
Dequeue 2, explore 2
Dequeue 5, explore 5
Dequeue 3, explore 3
Dequeue 4, explore 4
Dequeue 6, explore 6
Dequeue 7, explore 7
Dequeue 8, explore 8
Dequeue 9, explore 9

Done.
BFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix

BFS Example

Explore 2
Mark 3 as visited

0
1
2
3
4
5
6
7
8
9

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

visited 1 1 1 1 0 1 0 0 0 0

pred -1 0 0 2 -1 0 -1 -1 -1 -1

queue 0 1 2 5 3
BFS Example

```
0 1 1 1 1 0 1 0 0 0 0
visited
-1 0 0 2 -1 0 -1 -1 -1 -1
pred
0 1 2 5 3
queue
```

Explore 2

0 1 2 3 4 5 6 7 8 9

Done exploring 2
Dequeue 5

visited:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue:

0 1 2 5 3
Explore 5

**visited**

| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

**pred**

| -1 | 0 | 0 | 2 | -1 | 0 | -1 | -1 | -1 | -1 | -1 |

**queue**

0 1 2 5 3
Explore 5

0 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue

0 1 2 5 3

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

Mark 0 as visited

Mark 1 as visited

Mark 2 as visited

Mark 5 as visited

Done exploring 0

p0 is already visited

p5 is already visited

Done exploring 1

p0 is already visited

Mark 3 as visited

Done exploring 2

p0 is already visited

p1 is already visited

p3 is already visited

Mark 4 as visited

Mark 6 as visited

Mark 7 as visited

Done exploring 5

p2 is already visited

p4 is already visited

p5 is already visited

Done exploring 3

p3 is already visited

p5 is already visited

p7 is already visited

Mark 8 as visited

Done exploring 4

p5 is already visited

Done exploring 6

p4 is already visited

p5 is already visited

p8 is already visited

Mark 9 as visited

Done exploring 7

p4 is already visited

p7 is already visited

p9 is already visited

Done exploring 8

p7 is already visited

p8 is already visited

Done exploring 9

BFS Example
BFS Example

Explore 5

1 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

pred

|   | -1 | 0   | 0   | 2   | -1 | 0   | -1 | -1 | -1 | -1 |

queue

0 1 2 5 3
BFS Example

Explore 5

3 is already visited

visited 1 1 1 1 0 1 0 0 0 0 0

pred -1 0 0 2 -1 0 -1 -1 -1 -1

queue 0 1 2 5 3

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

Mark 0 as visited
Mark 1 as visited
Mark 2 as visited
Mark 5 as visited
Done exploring 0
p0 is already visited
p5 is already visited
Done exploring 1
p0 is already visited
p1 is already visited
p3 is already visited
Done exploring 2
p0 is already visited
p1 is already visited
p3 is already visited
Done exploring 3
p3 is already visited
p5 is already visited
p7 is already visited
Mark 4 as visited
Mark 6 as visited
Mark 7 as visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
Done exploring 6
p4 is already visited
p5 is already visited
p8 is already visited
Mark 8 as visited
Done exploring 7
p4 is already visited
p7 is already visited
p9 is already visited
Done exploring 8
p7 is already visited
p8 is already visited
Done exploring 9
p7 is already visited
Explore 5

Graph
Traversals
BFS
DFS
Ideas/Issues
Appendix
BFS Example
DFS Example
Path-Checking Example

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

pred

queue

0 1 2 5 3
BFS Example

Explore 5

Mark 4 as visited

```
visited  1 1 1 1 1 1 0 0 0 0 0
pred   -1 0 0 2 5 0 -1 -1 -1 -1
queue   0 1 2 5 3 4
```
### BFS Example

#### Explore 5

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>visited</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pred</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>queue</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix
BFS Example
DFS Example
Path-Checking Example

BFS Example

Explore 5
Mark 6 as visited

0
1
2
3
4
5
6
7
8
9

visited
pred
queue

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

visited: 1 1 1 1 1 1 1 0 0 0
pred: -1 0 0 2 5 0 5 -1 -1 -1
queue: 0 1 2 5 3 4 6
BFS Example

Explore 5

0 2 3 4 8 6 7 9

visited: 1 1 1 1 1 1 1 0 0 0
pred: -1 0 0 2 5 0 5 -1 -1 -1
queue: 0 1 2 5 3 4 6
BFS Example

Explore 5

Mark 7 as visited

visited: 1 1 1 1 1 1 1 1 0 0

pred: -1 0 0 2 5 0 5 5 -1 -1

queue: 0 1 2 5 3 4 6 7
Explore 5

BFS starting at 0

Dequeue 0

Explore 0

Dequeue 1

Explore 1

Dequeue 2

Explore 2

Dequeue 5

Explore 5

Dequeue 3

Explore 3

Dequeue 4

Explore 4

Dequeue 6

Explore 6

Dequeue 7

Explore 7

Dequeue 8

Explore 8

Dequeue 9

Explore 9

Done

visited

pred

queue
BFS Example

Dequeue 3

visited: 1 1 1 1 1 1 1 1 1 0 0
pred: -1 0 0 2 5 0 5 5 -1 -1
queue: 0 1 2 5 3 4 6 7
BFS Example

Explore 3

```
visited  1  1  1  1  1  1  1  1  1  0  0
pred    -1  0  0  2  5  0  5  5 -1 -1
queue   0  1  2  5  3  4  6  7
```
BFS Example

Explores the graph starting from node 0. The visited nodes are marked with 1 in the visited array, and the predecessors are recorded in the pred array.

The queue starts with 0, and the visited nodes are explored in the order of the queue. Each node is visited only once, and the process continues until all nodes are visited.

For example, when exploring node 3, node 2 is already visited, so exploration stops at node 2.
**BFS Example**

Explore 3

4 is already visited

```
BFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix
BFS Example
DFS Example
Path-Checking Example
```
BFS Example

Explore 3

5 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

queue

0 1 2 5 3 4 6 7
BFS Example

Explore 3

0
1
2

3
4
5
6

7
8
9

Done exploring 3

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

queue

0 1 2 5 3 4 6 7
BFS Example

Dequeue 4

```
visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
1   1   1   1   1   1   1   1   1   0   0

pred
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
-1   0   0   2   5   0   5   5   -1   -1

queue   0   1   2   5   3   4   6   7
```
Explore 4

BFS Example

```
visited [1 1 1 1 1 1 1 1 1 1 0 0]
pred  [-1 0 0 2 5 0 5 5 -1 -1]
queue [0 1 2 5 3 4 6 7]
```
BFS Example

Explore 4

3 is already visited

0 1 2 3 4 5 6 7 8 9

visited 1 1 1 1 1 1 1 1 1 0 0

pred -1 0 0 2 5 0 5 5 -1 -1

queue 0 1 2 5 3 4 6 7
BFS Example

Explore 4

5 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

queue

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
BFS Example

Explore 4

7 is already visited

visited
[0] 1 1 1 1 1 1 1 1 1 0 0

pred
-1 0 0 2 5 0 5 5 -1 -1

queue
0 1 2 5 3 4 6 7
BFS Example

Explore 4

```
visited:  1  1  1  1  1  1  1  1  1  0  0
pred:    -1  0  0  2  5  0  5  5  -1  -1
queue:  0  1  2  5  3  4  6  7
```
BFS Example

Explore 4

Mark 8 as visited

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
visited
[1] [1] [1] [1] [1] [1] [1] [1] [1] [1]

pred
[0] [1] [0] [2] [5] [0] [5] [5] [4] [-1]

queue
[0] [1] [2] [5] [3] [4] [6] [7] [8]
Explore 4

0

2

3

4

5

6

7

8

9

Done exploring 4

visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

0 1 1 1 1 1 1 1 1 1 1

pred

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

-1 0 0 2 5 0 5 5 4 -1

queue

0 1 2 5 3 4 6 7 8
BFS Example

Dequeue 6

visited
0 1 1 1 1 1 1 1 1 1 1 1 0
pred
-1 0 0 2 5 0 5 5 4 -1
queue
0 1 2 5 3 4 6 7 8
BFS Example

Explore 6

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

Mark 0 as visited
Mark 1 as visited
Mark 2 as visited
Mark 5 as visited
Done exploring 0
p0 is already visited
p5 is already visited
Done exploring 1
p0 is already visited
p1 is already visited
Mark 3 as visited
Done exploring 2
p0 is already visited
p1 is already visited
p3 is already visited
Mark 4 as visited
Mark 6 as visited
Mark 7 as visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
Done exploring 3
p3 is already visited
p5 is already visited
p7 is already visited
pMark 8 as visited
Done exploring 4
p5 is already visited
Done exploring 6
p4 is already visited
p5 is already visited
p8 is already visited
pMark 9 as visited
Done exploring 7
p4 is already visited
p7 is already visited
p9 is already visited
Done exploring 8
p7 is already visited
p8 is already visited
Done exploring 9

visited: 1 1 1 1 1 1 1 1 1 1 0
pred: -1 0 0 2 5 0 5 5 4 -1
queue: 0 1 2 5 3 4 6 7 8
BFS Example

Explore 6

5 is already visited

![Graph with BFS traversal example]

- Graph representation showing BFS traversal from vertex 0 to 9.
- Visited vertices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Pred predecessor matrix:
  - [0]: -1
  - [1]: 0
  - [2]: 0
  - [3]: 2
  - [4]: 5
  - [5]: 0
  - [6]: 5
  - [7]: 5
  - [8]: 4
  - [9]: -1
- Queue: 0 1 2 5 3 4 6 7 8

BFS starting at 0:
- Dequeue 0: Explore 0
- Dequeue 1: Explore 1
- Dequeue 2: Explore 2
- Dequeue 5: Explore 5
- Dequeue 3: Explore 3
- Dequeue 4: Explore 4
- Dequeue 6: Explore 6
- Dequeue 7: Explore 7
- Dequeue 8: Explore 8
- Dequeue 9: Explore 9

Done.
BFS Example

Explore 6

Done exploring 6

visited

0 1 1 1 1 1 1 1 1 1 1 0

pred

-1 0 0 2 5 0 5 5 4 -1

queue

0 1 2 5 3 4 6 7 8
Dequeue 7

visited 1 1 1 1 1 1 1 1 1 1 0
pred -1 0 0 2 5 0 5 5 4 -1
queue 0 1 2 5 3 4 6 7 8
BFS Example

Explore 7

0
1
2
3
4
5
6
7
8
9

visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

1 1 1 1 1 1 1 1 1 1 0

pred

-1 0 0 2 5 0 5 5 4 -1

queue

0 1 2 5 3 4 6 7 8
BFS Example

Explore 7

4 is already visited

visited

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>-1</td>
</tr>
</tbody>
</table>

queue

0 1 2 5 3 4 6 7 8
BFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix

BFS Example

Explore 7
5 is already visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

visited 1 1 1 1 1 1 1 1 1 1 0
pred -1 0 0 2 5 0 5 5 4 -1
queue 0 1 2 5 3 4 6 7 8

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done
Explore 7

8 is already visited

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

queue

0 1 2 5 3 4 6 7 8
BFS Example

Explore 7

0  2  3  5  8
1  4  6  7  9

visited: 1 1 1 1 1 1 1 1 1 1 0
pred:    -1 0 0 2 5 0 5 5 4 -1
queue: 0 1 2 5 3 4 6 7 8
BFS Example

0

2

3

4

5

6

7

8

Explore 7

Mark 9 as visited

visited

0

1

2

3

4

5

6

7

8

9

pred

-1

0

0

2

5

0

5

5

4

7

queue

0

1

2

5

3

4

6

7

8

9
BFS Example

Explore 7

Done exploring 7

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
visited 1 1 1 1 1 1 1 1 1 1

pred
-1 0 0 2 5 0 5 5 4 7

queue
0 1 2 5 3 4 6 7 8 9
BFS Example

Dequeue 8

```
visited  1  1  1  1  1  1  1  1  1  1  1
pred    -1  0  0  2  5  0  5  5  4  7
queue   0  1  2  5  3  4  6  7  8  9
```
Explore 8

visited

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

pred

| -1 | 0 | 0 | 2 | 5 | 0 | 5 | 5 | 4 | 7 |

queue

| 0 | 1 | 2 | 5 | 3 | 4 | 6 | 7 | 8 | 9 |
BFS Example

Explore 8

4 is already visited

0 1 2 3 4 5 6 7 8 9

visited

<table>
<thead>
<tr>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

pred

| -1  | 0   | 0   | 2   | 5   | 0   | 5   | 5   | 4   | 7   |

queue

| 0   | 1   | 2   | 5   | 3   | 4   | 6   | 7   | 8   | 9   |
BFS Example

Explore 8

7 is already visited

visited
0 1 1 1 1 1 1 1 1 1

pred
-1 0 0 2 5 0 5 5 4 7

queue
0 1 2 5 3 4 6 7 8 9
BFS Example

Explore 8

9 is already visited

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix

BFS Example
DFS Example
Path-Checking Example

visited 1 1 1 1 1 1 1 1 1 1

pred -1 0 0 2 5 0 5 5 4 7

queue 0 1 2 5 3 4 6 7 8 9
BFS Example

Graph Traversal

BFS

DFS

Ideas/Issues

Appendix

BFS Example

DFS Example

Path-Checking Example

BFS Example

Explore 8

Done exploring 8

visited

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

pred

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

queue

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
BFS Example

Dequeue 9

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
1 1 1 1 1 1 1 1 1 1

pred
-1 0 0 2 5 0 5 5 4 7

queue
0 1 2 5 3 4 6 7 8 9
BFS Example

Explore 9

0 - 2 - 3

5 - 4 - 8

1 - 5 - 6

7

visited: 1 1 1 1 1 1 1 1 1 1 1

pred: -1 0 0 2 5 0 5 5 4 7

queue: 0 1 2 5 3 4 6 7 8 9
BFS Example

Explore 9

7 is already visited

0

2

3

4

5

6

1

5

8

3

7

9

0

1

2

3

4

5

6

7

8

9

visited

1 1 1 1 1 1 1 1 1 1

pred

-1 0 0 2 5 0 5 5 4 7

queue

0 1 2 5 3 4 6 7 8 9
BFS Example

Explore 9

8 is already visited

visited: 1 1 1 1 1 1 1 1 1 1

pred: -1 0 0 2 5 0 5 5 4 7

queue: 0 1 2 5 3 4 6 7 8 9
Explore 9

visited

pred

queue

BFS starting at 0
Dequeue 0
Explore 0
Dequeue 1
Explore 1
Dequeue 2
Explore 2
Dequeue 5
Explore 5
Dequeue 3
Explore 3
Dequeue 4
Explore 4
Dequeue 6
Explore 6
Dequeue 7
Explore 7
Dequeue 8
Explore 8
Dequeue 9
Explore 9
Done

pMark 0 as visited
pMark 1 as visited
pMark 2 as visited
pMark 5 as visited
Done exploring 0
p0 is already visited
p5 is already visited
Done exploring 1
p0 is already visited
pMark 3 as visited
Done exploring 2
p0 is already visited
p1 is already visited
p3 is already visited
pMark 4 as visited
pMark 6 as visited
pMark 7 as visited
Done exploring 5
p2 is already visited
p4 is already visited
p5 is already visited
Done exploring 3
p3 is already visited
p5 is already visited
p7 is already visited
pMark 8 as visited
Done exploring 4
p5 is already visited
Done exploring 6
p4 is already visited
p5 is already visited
p8 is already visited
pMark 9 as visited
Done exploring 7
p4 is already visited
p7 is already visited
p9 is already visited
Done exploring 8
p7 is already visited
p8 is already visited
Done exploring 9
**BFS Example**

Graph Traversal

BFS

DFS

Ideas/Issues

Appendix

---

**BFS Example**

Done

```
0  1  2  3  4  5  6  7  8  9
```

**visited**

```
0  1  1  1  1  1  1  1  1  1  1
```

**pred**

```
-1  0  0  2  5  0  5  5  4  7
```

**queue**

```
0  1  2  5  3  4  6  7  8  9
```
DFS starting at 0

0

1

2

3

4

5

6

7

8

9

DFS Example

visited

0 0 0 0 0 0 0 0 0 0 0 0

visit order
DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

DFS Example

visit order
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
visited
0 0 0 0 0 0 0 0 0 0
dfs(0)
call stack
Mark 0 as visited

DFS Example

DFS starting at 0

Mark 0 as visited

1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited

0 1 0 0 0 0 0 0 0 0

visit order 0
DFS Example

1 has not been visited

0

1

0 has not been visited

5

3

4

6

2

7

8

9

visited

0 1 2 3 4 5 6 7 8 9

visit order

0

call stack

dfs(0)
DFS Example

Recurse into 1

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

visit order

0

call stack

dfs(0)
dfs(1)
DFS Example

Mark 1 as visited

 DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
0 1 0 0 0 0 0 0 0 0

visit order
0 1

call stack
dfs(1)
dfs(0)
DFS Example

5 has not been visited

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

visit order

0 1

call stack
dfs(0)
dfs(1)
Recuse into 5

DFS Example

Graph
Traversal
BFS
DFS
Ideas/Issues
Appendix
BFS Example
DFS Example
Path-Checking
Example
DFS Example

Mark 5 as visited

0 1 5

visited 1 1 0 0 0 0 1 0 0 0 0

visit order 0 1 5

call stack
dfs(0)
dfs(1)
dfs(5)
DFS Example

3 has not been visited

0

1

2

3

4

5

6

7

8

9

visited

0 1 1 0 0 0 1 0 0

visit order 0 1 5
DFR Example

Recursion into 3

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited:

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

visit order: 0 1 5

call stack:
- dfs(0)
- dfs(1)
- dfs(3)
- dfs(5)
DFS Example

Mark 3 as visited

DFS Example

Mark 3 as visited
2 has not been visited

DFS Example

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

visit order 0 1 5 3
DFS Example

Recure into 2

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

visit order
0 1 5 3

call stack
dfs(0)
dfs(1)
dfs(2)
dfs(3)
dfs(4)
dfs(5)
dfs(6)
dfs(7)
dfs(8)
dfs(9)
DFS Example

Mark 2 as visited

visited

visit order

0 1 5 3 2
DFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix
Path-Checking Example

Return

0 1 5 3 2

visited

0 1 1 1 1 0 1 0 0 0 0

call stack

dfs(0)
dfs(1)
dfs(3)
dfs(5)

dfs(9)

dfs(6)

dfs(7)

dfs(8)

visit order

0 1 5 3 2

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

BFS Example

Path-Checking Example
DFS Example

4 has not been visited

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return

Visit order:
0 1 5 3 2

Visited:
1 1 1 1 0 1 0 0 0 0

Call stack:
dfs(0)
dfs(1)
dfs(3)
dfs(5)
dfs(8)
DFS Example

Recursively into 4

DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return
Mark 4 as visited

DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

[1] [1] [1] [1] [1] [1] [1] [0] [0] [0]

visit order 0 1 5 3 2 4

call stack
dfs(0)
dfs(1)
dfs(2)
dfs(3)
dfs(4)
dfs(5)
dfs(6)
dfs(7)
dfs(8)
dfs(9)
DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

visit order: 0 1 5 3 2 4

visited: [0] 1 1 1 1 1 1 1 0 0 0 0

call stack:
- dfs(4)
- dfs(3)
- dfs(5)
- dfs(1)
- dfs(0)
DFS Example

Recure into 7

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

[1] [1] [1] [1] [1] [1] [1] [0] [0] [0]

visit order

0 1 5 3 2 4
Mark 7 as visited

dfs(0)
dfs(1)
dfs(5)
dfs(3)
dfs(2)
dfs(4)
dfs(7)
dfs(8)
dfs(9)
dfs(6)

visited

[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

visit order 0 1 5 3 2 4 7

call stack
DFS Example

8 has not been visited

0 1 5 3 2 4 7

visited: [1 1 1 1 1 1 1 0 1 0 0]

call stack: dfs(0), dfs(1), dfs(3), dfs(5), dfs(4), dfs(7), dfs(8), dfs(9)

DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

Return

Return

Return

Return

Return

Return

6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Recursion into 8

BFS Example

DFS Example

Graph Traversal
BFS
DFS
Ideas/Issues
Appendix
Path-Checking Example
Mark 8 as visited

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

call stack
dfs(8)
dfs(7)
dfs(4)
dfs(3)
dfs(5)
dfs(1)
dfs(0)

visit order 0 1 5 3 2 4 7 8
visited 1 1 1 1 1 1 1 0 1 1 1 0
DFS Example

9 has not been visited

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

DFS Example

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

visit order

0 1 5 3 2 4 7 8
DFS Example

Recursively through graph:

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

Visit order:
0 1 5 3 2 4 7 8

Visited:
[0] 1 1 1 1 1 1 1 0 1 1 1 0

Call stack:
dfs(0)
dfs(1)
dfs(3)
dfs(4)
dfs(5)
dfs(7)
dfs(8)
dfs(9)
DFS Example

Mark 9 as visited

DFS starting at 0
- Mark 0 as visited
- 1 has not been visited
  - Recurse into 1
  - Mark 1 as visited
    - 5 has not been visited
      - Recurse into 5
      - Mark 5 as visited
        - 3 has not been visited
          - Recurse into 3
          - Mark 3 as visited
            - 2 has not been visited
              - Recurse into 2
              - Mark 2 as visited
              - Return
            - 4 has not been visited
              - Recurse into 4
              - Mark 4 as visited
              - 7 has not been visited
                - Recurse into 7
                - Mark 7 as visited
                - 8 has not been visited
                  - Recurse into 8
                  - Mark 8 as visited
                  - 9 has not been visited
                    - Recurse into 9
                    - Mark 9 as visited
                    - Return
            - 6 has not been visited
              - Recurse into 6
              - Mark 6 as visited
              - Return
          - Return
        - Return
      - Return
    - Return
  - Return
- 4 has not been visited
  - Recurse into 4
  - Mark 4 as visited
- 7 has not been visited
  - Recurse into 7
  - Mark 7 as visited
- 8 has not been visited
  - Recurse into 8
  - Mark 8 as visited
- 9 has not been visited
  - Recurse into 9
  - Mark 9 as visited
- Return

Call stack:
- dfs(0)
- dfs(1)
- dfs(3)
- dfs(4)
- dfs(5)
- dfs(7)
- dfs(8)
- dfs(9)

Visit order:
0 1 5 3 2 4 7 8 9

Visited:
[1 1 1 1 1 1 1 0 1 1 1]
DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

visited

<table>
<thead>
<tr>
<th></th>
<th>[0]</th>
<th>[1]</th>
<th>[2]</th>
<th>[3]</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
<th>[7]</th>
<th>[8]</th>
<th>[9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

visit order

0 1 5 3 2 4 7 8 9
DFS Example

Return

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

0 1 5 3 2 4 7 8 9

visited

[0] 1 1 1 1 1 1 1 0 1 1 1 1

visit order

[0] 1 5 3 2 4 7 8 9

call stack

dfs(0)
dfs(1)
dfs(3)
dfs(5)
dfs(4)
dfs(7)
DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

Return

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example

DFS Example
DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

Visit order: 0 1 5 3 2 4 7 8 9

Visited: [1 1 1 1 1 1 1 0 1 1 1]

Call stack:
- dfs(0)
- dfs(1)
- dfs(3)
- dfs(5)
- dfs(9)
- dfs(7)
- dfs(8)
DFS Example

DFS starting at 0

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return

6 has not been visited
Recurse into 6
Mark 6 as visited
Return

Return

Return

Return

Return

Return

Return

Return

Return

Return
DFS Example

6 has not been visited

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

visit order
0 1 5 3 2 4 7 8 9

call stack
dfs(0)
dfs(1)
dfs(5)

Graph
Traversal
BFS
DFS
Ideas/Issues
Appendix
BFS Example
DFS Example
Path-Checking Example
DFS Example

Recurse into 6

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
0 1 1 1 1 1 1 0 1 1 1 1

visit order
0 1 5 3 2 4 7 8 9

call stack
dfs(6)
dfs(5)
dfs(1)
dfs(0)
### DFS Example

Mark 6 as visited

- **Graph**:
  - Vertices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Edges: 0-2, 2-1, 3-5, 5-4, 4-7, 7-8, 8-9, 6-5, 5-6

- **Visited Nodes**:
  - Order: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Marked as visited:
    - 0
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 9

- **Call Stack**:
  - dfs(0) → dfs(1) → dfs(2) → dfs(3) → dfs(4) → dfs(5) → dfs(6) → dfs(7) → dfs(8) → dfs(9)

- **Visit Order**:
  - 0, 1, 5, 3, 2, 4, 7, 8, 9, 6
DFS Example

Return

0

1

2

3

4

5

6

7

8

9

dfs(0)
dfs(1)
dfs(5)
call stack

visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

0 1 1 1 1 1 1 1 1 1

visit order

0 1 5 3 2 4 7 8 9 6
DFS Example

DF Example

Return

visited

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

DFS Example

0
1
2
3
4
5
6
7
8
9
0
1
5
3
2
4
7
8
9
6

DF Example

call stack

dfs(0)
dfs(1)
DFS Example

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return

Return

Return
DFS Example

Return

DFS starting at 0
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
8 has not been visited
Recurse into 8
Mark 8 as visited
9 has not been visited
Recurse into 9
Mark 9 as visited
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return
Return
Return
Return
Return
6 has not been visited
Recurse into 6
Mark 6 as visited
Return

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited

1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
1 has not been visited

Is there a path between 0 and 7?

Mark 0 as visited

1 has not been visited
Recurse into 1
Mark 1 as visited

5 has not been visited
Recurse into 5
Mark 5 as visited

3 has not been visited
Recurse into 3
Mark 3 as visited

2 has not been visited
Recurse into 2
Mark 2 as visited

Return false

4 has not been visited
Recurse into 4
Mark 4 as visited

7 has not been visited
Recurse into 7
Mark 7 as visited

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited

1 has not been visited

Recurse into 1

Mark 1 as visited

5 has not been visited

Recurse into 5

Mark 5 as visited

3 has not been visited

Recurse into 3

Mark 3 as visited

2 has not been visited

Recurse into 2

Mark 2 as visited

Return false

4 has not been visited

Recurse into 4

Mark 4 as visited

7 has not been visited

Recurse into 7

Mark 7 as visited

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?
Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Path-Checking with Recursive DFS

Example

Recursion example:

Is there a path between 0 and 7?

Mark 0 as visited

1 has not been visited

Recurse into 1

Mark 1 as visited

5 has not been visited

Recurse into 5

Mark 5 as visited

3 has not been visited

Recurse into 3

Mark 3 as visited

2 has not been visited

Recurse into 2

Mark 2 as visited

Return false

4 has not been visited

Recurse into 4

Mark 4 as visited

7 has not been visited

Recurse into 7

Mark 7 as visited

Return true

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Path-Checking with Recursive DFS

Example

3 has not been visited

Is there a path between 0 and 7?

Mark 0 as visited

1 has not been visited

Recurse into 1

Mark 1 as visited

5 has not been visited

Recurse into 5

Mark 5 as visited

3 has not been visited

Recurse into 3

Mark 3 as visited

2 has not been visited

Recurse into 2

Mark 2 as visited

Return false

4 has not been visited

Recurse into 4

Mark 4 as visited

7 has not been visited

Recurse into 7

Mark 7 as visited

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Mark 3 as visited

Is there a path between 0 and 7?

- Mark 0 as visited
- 1 has not been visited
- Recurse into 1
- Mark 1 as visited
- 5 has not been visited
- Recurse into 5
- Mark 5 as visited
- 3 has not been visited
- Recurse into 3
- Mark 3 as visited
- 2 has not been visited
- Recurse into 2
- Mark 2 as visited
- Return false
- 4 has not been visited
- Recurse into 4
- Mark 4 as visited
- 7 has not been visited
- Recurse into 7
- Mark 7 as visited
- Return true

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Path-Checking with Recursive DFS

Example

Mark 2 as visited

Is there a path between 0 and 7?

1. Mark 0 as visited
2. 1 has not been visited
3. Recurse into 1
4. Mark 1 as visited
5. 5 has not been visited
6. Recurse into 5
7. Mark 5 as visited
8. 3 has not been visited
9. Recurse into 3
10. Mark 3 as visited
11. 2 has not been visited
12. Recurse into 2
13. Mark 2 as visited
14. Return false
15. 4 has not been visited
16. Recurse into 4
17. Mark 4 as visited
18. 7 has not been visited
19. Recurse into 7
20. Mark 7 as visited
21. Return true

Answer: Yes

visited:

| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

call stack:

path(0, 7)?
path(1, 7)?
path(2, 7)?
path(3, 7)?
path(4, 7)?
path(5, 7)?
path(6, 7)?
path(7, 7)?
Return false

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Path-Checking with Recursive DFS

Example

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes

visited
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
1 1 1 1 0 1 0 0 0 0

call stack
path(3, 7)?
path(5, 7)?
path(1, 7)?
path(0, 7)?
Path-Checking with Recursive DFS

Example

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Path-Checking with Recursive DFS Example

Mark 4 as visited

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes

visited

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

call stack

path(0, 7)?
path(1, 7)?
path(2, 7)?
path(3, 7)?
path(4, 7)?
path(5, 7)?
path(6, 7)?
path(7, 7)?
Path-Checking with Recursive DFS Example

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true
Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited

Return true
Path-Checking with Recursive DFS Example

Return true

Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited

Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true

Answer: Yes
Is there a path between 0 and 7?

- Mark 0 as visited
- 1 has not been visited
  - Recurse into 1
  - Mark 1 as visited
    - 5 has not been visited
      - Recurse into 5
      - Mark 5 as visited
        - 3 has not been visited
          - Recurse into 3
          - Mark 3 as visited
            - 2 has not been visited
              - Recurse into 2
              - Mark 2 as visited
                - Return false
            - 4 has not been visited
              - Recurse into 4
              - Mark 4 as visited
                - 7 has not been visited
                  - Recurse into 7
                  - Mark 7 as visited
                - Return true
- Answer: Yes
Is there a path between 0 and 7?

Mark 0 as visited
1 has not been visited
Recurse into 1
Mark 1 as visited
5 has not been visited
Recurse into 5
Mark 5 as visited
3 has not been visited
Recurse into 3
Mark 3 as visited
2 has not been visited
Recurse into 2
Mark 2 as visited
Return false
4 has not been visited
Recurse into 4
Mark 4 as visited
7 has not been visited
Recurse into 7
Mark 7 as visited
Return true

Answer: Yes
Path-Checking with Recursive DFS

Example

Return true

visited

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

0 1 1 1 1 1 1 0 1 0 0

path(0, 7)? call stack
Path-Checking with Recursive DFS

Answer: Yes

Call stack:

visited: 1 1 1 1 1 1 1 0 1 0 0