April 15 – Graph Traversals
Wade Fagen-Ulmschneider, Craig Zilles
Graphs

To study all of these structures:
1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms
Traversal:

Objective: Visit every vertex and every edge in the graph.

Purpose: Search for interesting sub-structures in the graph.

We’ve seen traversal before ....but it’s different:

- Ordered
- Obvious Start
Traversals: BFS

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
<th>Adjacent Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Traversing the graph using BFS (Breadth-First Search), we get the following sequence of visited vertices:

A, B, C, D, E, F, G, H,

The adjacent edges for each vertex are:

<table>
<thead>
<tr>
<th>v</th>
<th>d</th>
<th>P</th>
<th>Adjacent Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-</td>
<td>C B D</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>A</td>
<td>A C E</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>A</td>
<td>B A D E F</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>A</td>
<td>A C F H</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>C</td>
<td>B C G</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>C</td>
<td>C D G</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>E</td>
<td>E F H</td>
</tr>
<tr>
<td>H</td>
<td>2</td>
<td>D</td>
<td>D G</td>
</tr>
</tbody>
</table>
BFS(G):
Input: Graph, G
Output: A labeling of the edges on G as discovery and cross edges

foreach (Vertex v : G.vertices()):
setLabel(v, UNEXPLORED)

foreach (Edge e : G.edges()):
setLabel(e, UNEXPLORED)

foreach (Vertex v : G.vertices()):
if getLabel(v) == UNEXPLORED:
BFS(G, v)

BFS(G, v):
Queue q
setLabel(v, VISITED)
q.enqueue(v)

while !q.empty():
  v = q.dequeue()
  foreach (Vertex w : G.adjacent(v)):
    if getLabel(w) == UNEXPLORED:
      setLabel(v, w, DISCOVERY)
      setLabel(w, VISITED)
      q.enqueue(w)
    elseif getLabel(v, w) == UNEXPLORED:
      setLabel(v, w, CROSS)
BFS Analysis

Q: Does our implementation handle disjoint graphs? If so, what code handles this?
   • *How do we use this to count components?*

Q: Does our implementation detect a cycle?
   • *How do we update our code to detect a cycle?*

Q: What is the running time?
Running time of BFS

While-loop at :19?

For-loop at :21?
BFS(G):
Input: Graph, G
Output: A labeling of the edges on G as discovery and cross edges

g {foreach (Vertex v : G.vertices()):
    setLabel(v, UNEXPLORED)
}{foreach (Edge e : G.edges()):
    setLabel(e, UNEXPLORED)
}{foreach (Vertex v : G.vertices()):
    if getLabel(v) == UNEXPLORED:
        BFS(G, v)
}

BFS(G, v):
Queue q
setLabel(v, VISITED)
q.enqueue(v)

while !q.empty():
    v = q.dequeue()
    {foreach (Vertex w : G.adjacent(v)):}
        if getLabel(w) == UNEXPLORED:
           setLabel(v, w, DISCOVERY)
           setLabel(w, VISITED)
            q.enqueue(w)
        elseif getLabel(v, w) == UNEXPLORED:
           setLabel(v, w, CROSS)
BFS Observations

**Q:** What is a shortest path from **A** to **H**?

**Q:** What is a shortest path from **E** to **H**?

**Q:** How does a cross edge relate to **d**?

**Q:** What structure is made from discovery edges?
BFS Observations

**Obs. 1:** Traversals can be used to count components.

**Obs. 2:** Traversals can be used to detect cycles.

**Obs. 3:** In BFS, \(d\) provides the shortest distance to every vertex.

**Obs. 4:** In BFS, the endpoints of a cross edge never differ in distance, \(d\), by more than 1:

\[|d(u) - d(v)| = 1\]
Traversal: DFS
Traversals: DFS

- **Discovery Edge**
- **Back Edge**
BFS(G):

Input: Graph, G
Output: A labeling of the edges on G as discovery and cross edges

does

foreach (Vertex v : G.vertices()):
    setLabel(v, UNEXPLORED)

foreach (Edge e : G.edges()):
    setLabel(e, UNEXPLORED)

foreach (Vertex v : G.vertices()):
    if getLabel(v) == UNEXPLORED:
        BFS(G, v)

BFS(G, v):

Queue q
setLabel(v, VISITED)
q.enqueue(v)

while !q.empty():
    v = q.dequeue()
    foreach (Vertex w : G.adjacent(v)):
        if getLabel(w) == UNEXPLORED:
            setLabel(v, w, DISCOVERY)
            setLabel(w, VISITED)
            q.enqueue(w)
        elseif getLabel(v, w) == UNEXPLORED:
            setLabel(v, w, CROSS)
DFS(G):
Input: Graph, G
Output: A labeling of the edges on G as discovery and back edges

foreach (Vertex v : G.vertices()):
    setLabel(v, UNEXPLORED)

foreach (Edge e : G.edges()):
    setLabel(e, UNEXPLORED)

foreach (Vertex v : G.vertices()):
    if getLabel(v) == UNEXPLORED:
        DFS(G, v)

DFS(G, v):
Queue q
setLabel(v, VISITED)
q.enqueue(v)

while !q.empty():
    v = q.dequeue()
    foreach (Vertex w : G.adjacent(v)):
        if getLabel(w) == UNEXPLORED:
            setLabel(v, w, DISCOVERY)
            setLabel(w, VISITED)
            DFS(G, w)
        elseif getLabel(v, w) == UNEXPLORED:
            setLabel(v, w, BACK)
Running time of DFS

Labeling:
• Vertex:
  • Edge:

Queries:
• Vertex:
  • Edge: