April 19 – Graph Traversals
Brad Solomon
Mid-Project Check-ins this week!

Discuss:

Current Progress (First deliverable done?)

Future Progress (What do you have left to do?)

Group Cohesion (Any issues or concerns?)
Learning Objectives

• Discuss pseudo-code for BFS and DFS on graphs

• Analyze and contrast BFS/DFS algorithm runtime and utility

• If time: Introduce Minimum Spanning Tree (MST) problem
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
- Clear End

- Any Order
- Any Start
- End is not obvious
Traversal: 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>
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>0</td>
<td>-</td>
<td>B C 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>A B 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>
Traversal: 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>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>A B 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>

G H F E D B C A
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?
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)
Running time of BFS

While-loop at :19?

For-loop at :21?
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?

<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 Observations

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

**Obs. 2:** BFS 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:

$$|d(u) - d(v)| \leq 1$$
Traversal: DFS

Diagram showing a graph with nodes labeled A, B, C, D, E, F, G, H, K, and J, connected by edges.
Traversal: DFS

Discovery Edge

Back Edge
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)
DFS(G):
Input: Graph, G
Output: A labeling of the edges on G as discovery and back edges

dfsMainFunction(v)

foreach (Vertex v : G.vertices()):
    setLabel(v, UNEXPLORED)
echo for each edge e in G edges():
    setLabel(e, UNEXPLORED)
echo for each vertex v in G vertices():
    if getLabel(v) == UNEXPLORED:
        dfsMainFunction(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)
            dfsMainFunction(w)
echo else if getLabel(v, w) == UNEXPLORED:
            setLabel(v, w, BACK)