BFS Graph Traversal

<table>
<thead>
<tr>
<th>d</th>
<th>p</th>
<th>v</th>
<th>Adjacent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>A</td>
<td>C B D</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>A C E</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>C</td>
<td>B A D E F</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>D</td>
<td>A C F H</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>E</td>
<td>B C G</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>F</td>
<td>C D G</td>
</tr>
<tr>
<td>3</td>
<td>E</td>
<td>G</td>
<td>E F H</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>H</td>
<td>D G</td>
</tr>
</tbody>
</table>

BST Graph Observations

1. Does our implementation handle disjoint graphs? How?
   a. How can we modify our code to count components?

2. Can our implementation detect a cycle? How?
   a. How can we modify our code to store update a private member variable `cycleDetected_`?

3. What is the running time of our algorithm?

4. What is the shortest path between A and H?

5. What is the shortest path between E and H?
   a. What does that tell us about BFS?

6. What does a cross edge tell us about its endpoints?

7. What structure is made from discovery edges in G?
Big Ideas: Utility of a BFS Traversal

**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: \( |d(u) - d(v)| = 1 \)

---

Depth First Search – A Modification to BFS

Two types of edges: 1. ...

---

Running Time of DFS:

**Labeling:**
- **Vertex:**
- **Edge:**

**Queries:**
- **Vertex:**
- **Edge:**

---

Pseudocode for DFS

```
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)
```

---

CS 225 – Things To Be Doing:

1. Exam #11 (theory) is ongoing
2. MP7 released (+14 EC due on Monday!)
3. lab_dictionary due Wednesday at 7:00pm
4. lab_graphs starts Wednesday
5. Multi-day “puzzle” POTDs available M/W/F

---

*“The Muddy City” by CS Unplugged, Creative Commons BY-NC-SA 4.0*