Graph 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 – this is different:

<table>
<thead>
<tr>
<th>BST</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="BST Diagram" /></td>
<td><img src="image2" alt="Graph Diagram" /></td>
</tr>
</tbody>
</table>

**BFS Graph Traversal:**

```plaintext
BFS(G):
# Pseudocode for BFS
1. BFS(G):
2. Input: Graph, G
3. Output: A labeling of the edges on G as discovery and cross edges
4. foreach (Vertex v : G.vertices()):
5. setLabel(v, UNEXPLORED)
6. foreach (Edge e : G.edges()):
7. setLabel(e, UNEXPLORED)
8. foreach (Vertex v : G.vertices()):
9. if getLabel(v) == UNEXPLORED:
10. BFS(G, v)
11. BFS(G, v):
12. Queue q
13. setLabel(v, VISITED)
14. q.enqueue(v)
15. while !q.empty():
16. v = q.dequeue()
17. foreach (Vertex w : G.adjacent(v)):
18. if getLabel(w) == UNEXPLORED:
19. setLabel(v, w, DISCOVERY)
20. setLabel(w, VISITED)
21. q.enqueue(w)
22. elseif getLabel(v, w) == UNEXPLORED:
23. setLabel(v, w, CROSS)
```

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

---

**DFS Graph Traversal**

Two types of edges:

1. 

2. 

---

**Modifying BFS to create DFS**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BFS(G):</td>
</tr>
<tr>
<td>2</td>
<td>Input: Graph, G</td>
</tr>
<tr>
<td>3</td>
<td>Output: A labeling of the edges on G as discovery and cross edges</td>
</tr>
<tr>
<td>4</td>
<td>foreach (Vertex v : G.vertices()):</td>
</tr>
<tr>
<td>5</td>
<td>setLabel(v, UNEXPLORED)</td>
</tr>
<tr>
<td>6</td>
<td>foreach (Edge e : G.edges()):</td>
</tr>
<tr>
<td>7</td>
<td>setLabel(e, UNEXPLORED)</td>
</tr>
<tr>
<td>8</td>
<td>foreach (Vertex v : G.vertices()):</td>
</tr>
<tr>
<td>9</td>
<td>if getLabel(v) == UNEXPLORED:</td>
</tr>
<tr>
<td>10</td>
<td>BFS(G, v)</td>
</tr>
<tr>
<td>11</td>
<td>BFS(G, v):</td>
</tr>
<tr>
<td>12</td>
<td>Queue q</td>
</tr>
<tr>
<td>13</td>
<td>setLabel(v, VISITED)</td>
</tr>
<tr>
<td>14</td>
<td>q.enqueue(v)</td>
</tr>
<tr>
<td>15</td>
<td>while !q.empty():</td>
</tr>
<tr>
<td>16</td>
<td>v = q.dequeue()</td>
</tr>
<tr>
<td>17</td>
<td>foreach (Vertex w : G.adjacent(v)):</td>
</tr>
<tr>
<td>18</td>
<td>if getLabel(w) == UNEXPLORED:</td>
</tr>
<tr>
<td>19</td>
<td>setLabel(v, w, DISCOVERY)</td>
</tr>
<tr>
<td>20</td>
<td>setLabel(w, VISITED)</td>
</tr>
<tr>
<td>21</td>
<td>q.enqueue(w)</td>
</tr>
<tr>
<td>22</td>
<td>elseif getLabel(v, w) == UNEXPLORED:</td>
</tr>
<tr>
<td>23</td>
<td>setLabel(v, w, CROSS)</td>
</tr>
</tbody>
</table>

---

**Minimum Spanning Tree**

---

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

---

**CS 225 – Things To Be Doing:**

1. Programming Exam C: Thursday, April 18 – Sunday, April 21
2. lab_ml this week!
3. MP6 due tonight; MP7 released tomorrow
4. Daily POTDs resume today!