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:

Pseudocode for BFS

```plaintext
BFS(G):
1. Input: Graph, G
2. Output: A labeling of the edges on G as discovery and cross edges
   3. foreach (Vertex v : G.vertices()):
   4.     setLabel(v, UNEXPLORED)
   5.     foreach (Edge e : G.edges()):
   6.         setLabel(e, UNEXPLORED)
   7.     foreach (Vertex v : G.vertices()):
   8.         if getLabel(v) == UNEXPLORED:
   9.             BFS(G, v)

BFS(G, v):
10. Queue q
11. setLabel(v, VISITED)
12. q.enqueue(v)
13. while !q.empty():
14.     v = q.dequeue()
15.     foreach (Vertex w : G.adjacent(v)):
16.         if getLabel(w) == UNEXPLORED:
17.             setLabel(v, w, DISCOVERY)
18.         setLabel(w, VISITED)
19.         q.enqueue(w)
20.     elseif getLabel(v, w) == UNEXPLORED:
21.         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:** 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: \[|d(u) - d(v)| = 1\]

---

**DFS Graph Traversal**

Two types of edges:

1. 

2. 

---

```java
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)
```

---

**Minimum Spanning Tree**

---

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

---

**CS 225 – Things To Be Doing:**

1. Final Project proposals being reviewed.
2. mp_mazes final due date on Monday, Nov. 15
3. POTD ongoing