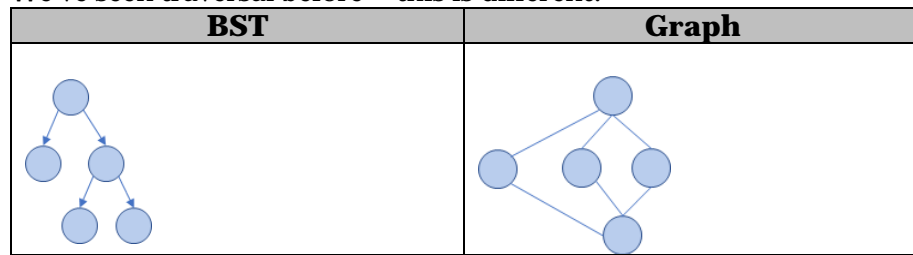


Graph Traversal

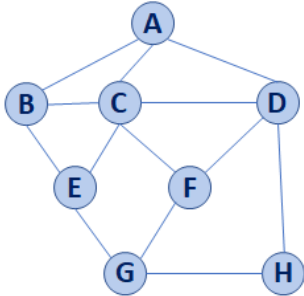
Objective: Visit every vertex and every edge in the graph.

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

We've seen traversal before – this is different:



BFS Graph Traversal:

Pseudocode for BFS	
<pre> 1 BFS(G): 2 Input: Graph, G 3 Output: A labeling of the edges on 4 G as discovery and cross edges 5 6 foreach (Vertex v : G.vertices()): 7 setLabel(v, UNEXPLORED) 8 foreach (Edge e : G.edges()): 9 setLabel(e, UNEXPLORED) 10 foreach (Vertex v : G.vertices()): 11 if getLabel(v) == UNEXPLORED: 12 BFS(G, v) 13 14 BFS(G, v): 15 Queue q 16 setLabel(v, VISITED) 17 q.enqueue(v) 18 19 while !q.empty(): 20 v = q.dequeue() 21 foreach (Vertex w : G.adjacent(v)): 22 if getLabel(w) == UNEXPLORED: 23 setLabel(v, w, DISCOVERY) 24 setLabel(w, VISITED) 25 q.enqueue(w) 26 elseif getLabel(v, w) == UNEXPLORED: 27 setLabel(v, w, CROSS) </pre>	

Vertex (v)	Distance (d)	Prev. (p)	Adjacent
A			
B			
C			
D			
E			
F			
G			
H			

BFS Graph Observations

1. Does our implementation handle disjoint graphs? How?

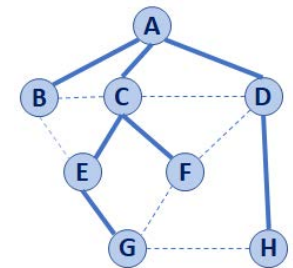
a. How can we modify our code to count components?

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**?

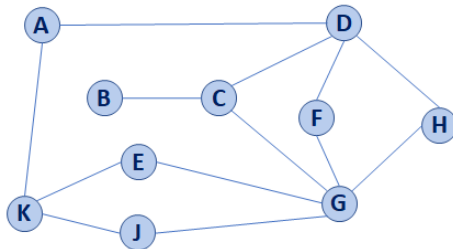
Big Ideas: Utility of a BFS Traversal

- 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.



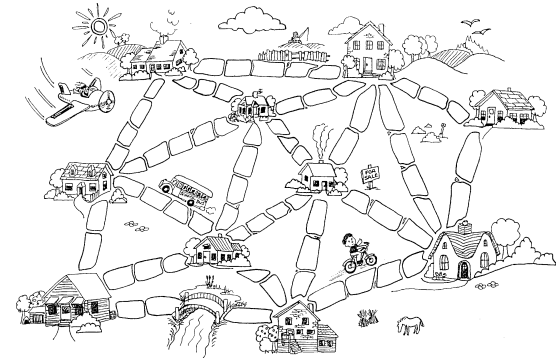
Modifying BFS to create DFS

```

1 BFS(G):
2   Input: Graph, G
3   Output: A labeling of the edges on
4           G as discovery and cross edges
5
6   foreach (Vertex v : G.vertices()):
7     setLabel(v, UNEXPLORED)
8   foreach (Edge e : G.edges()):
9     setLabel(e, UNEXPLORED)
10  foreach (Vertex v : G.vertices()):
11    if getLabel(v) == UNEXPLORED:
12      BFS(G, v)
13
14  BFS(G, v):
15    Queue q
16    setLabel(v, VISITED)
17    q.enqueue(v)
18
19    while !q.empty():
20      v = q.dequeue()
21      foreach (Vertex w : G.adjacent(v)):
22        if getLabel(w) == UNEXPLORED:
23          setLabel(v, w, DISCOVERY)
24          setLabel(w, VISITED)
25          q.enqueue(w)
26        elseif getLabel(v, w) == UNEXPLORED:
27          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. **Programming Exam C is different than usual schedule:**
Exam: Sunday, Dec 2 – Tuesday, Dec 4
2. lab_dict due on Tuesday, Nov. 27; lab_ml this week!
3. MP6 due tonight; MP7 released tomorrow
4. Daily POTDs resume today!