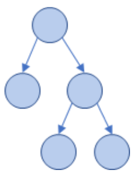
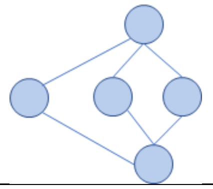
#38: Graph Traversals

Graph Traversal

Objective: Visit every vertex and every edge in the graph.
Purpose: Search for interesting sub-structures in the graph.

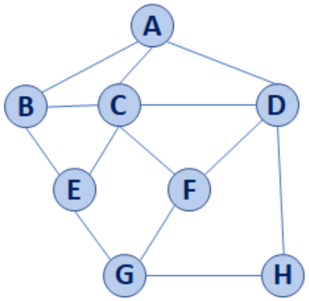
We've seen traversal before – this is different:

BST	Graph
	

BFS Graph Traversal:

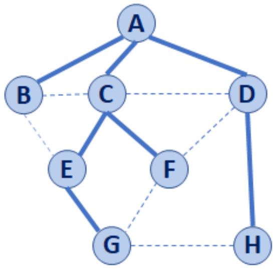
Pseudocode for BFS

```
BFS(G):
  foreach (Vertex v : G.vertices()):
setPred(v, NULL)
setDist(v, -1)

  foreach (Edge e : G.edges()):
    setLabel(e, UNEXPLORED)

  foreach (Vertex v : G.vertices()):
    if getDist(v) == -1:
       BFS(G, v)

BFS(G, v):
  Queue q
  setDist(v, 0)
  q.enqueue(v)

  while !q.empty():
    v = q.dequeue()

  foreach (Vertex w : G.adjacent(v)):
    if( getDist(w) == -1):
      setLabel((v, w), DISCOVERY)
      setPred(w, v)
      setDist(w, v + 1)
      q.enqueue(w)
    else:
      setLabel((v, w), CROSS)
```



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?

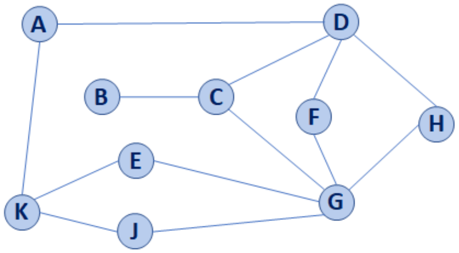
6. What does a cross edge tell us about its endpoints?

7. What structure is made from discovery edges in **G**?

Big Ideas: Utility of a BFS Traversal

Obs. 1: 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:

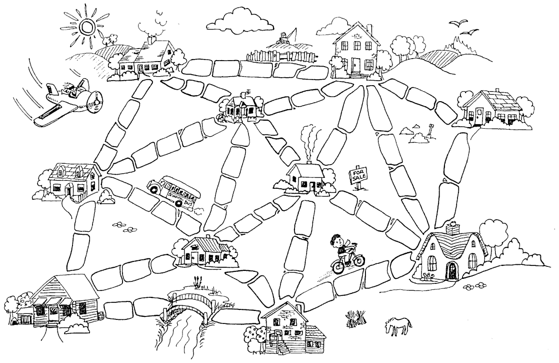
1.

2.

Modifying BFS to create DFS

```

DFS(G):
  foreach (Vertex v : G.vertices()):
    setPred(v, NULL)
    setDist(v, -1)

  foreach (Edge e : G.edges()):
    setLabel(e, UNEXPLORED)

  foreach (Vertex v : G.vertices()):
    if getDist(v) == -1:
      DFS(G, v)

DFS(G, v):

foreach (Vertex w : G.adjacent(v)):
   if( getDist(w) == -1):
      setLabel((v, w), DISCOVERY)
      setPred(w, v)
      setDist(w, v + 1)
      DFS(G, w)
   else:
      setLabel((v, w), BACK)
```

Minimum Spanning Tree



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