

## Graph Traversal – BFS

### 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.

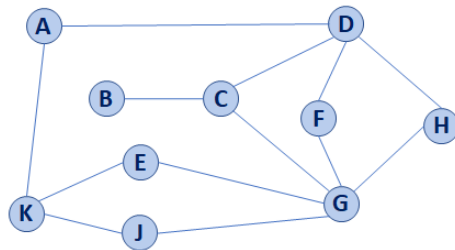
**Obs. 3:** In BFS,  $d$  provides the shortest distance to every vertex.

**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

**Idea:** Traverse deep into the graph quickly, visiting more distant nodes before neighbors.

Two types of edges:



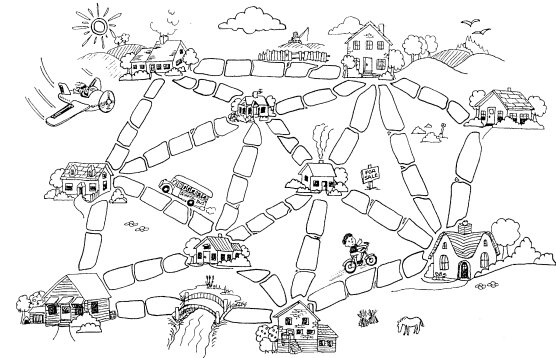
### Modifying BFS to create DFS

```

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

A **Spanning Tree** on a connected graph  $G$  is a subgraph,  $G'$ , such that:

1. Every vertex in  $G$  is in  $G'$  and
2.  $G'$  is connected with the minimum number of edges

This construction will always create a new graph that is a \_\_\_\_\_ (connected, acyclic graph) that spans  $G$ .

A **Minimum Spanning Tree** is a spanning tree with the **minimal total edge weights** among all spanning trees.

- Every edge must have a weight
  - The weights are unconstrained, except they must be additive (*eg: can be negative, can be non-integers*)
- Output of a MST algorithm produces  $G'$ :
  - $G'$  is a spanning graph of  $G$
  - $G'$  is a tree

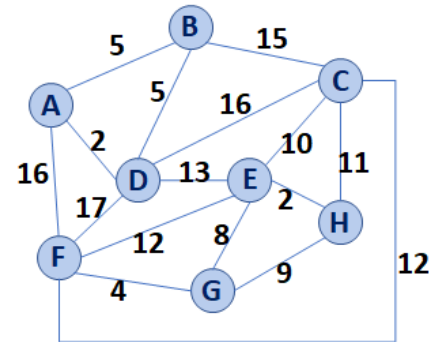
$G'$  has a minimal total weight among all spanning trees. There may be multiple minimum spanning trees, but they will have the same total weight.

**Pseudocode for Kruskal's MST Algorithm**

```

1  KruskalMST(G) :
2  DisjointSets forest
3  foreach (Vertex v : G) :
4      forest.makeSet(v)
5
6  PriorityQueue Q // min edge weight
7  foreach (Edge e : G) :
8      Q.insert(e)
9
10 Graph T = (V, {})
11
12 while |T.edges()| < n-1:
13     Vertex (u, v) = Q.removeMin()
14     if forest.find(u) != forest.find(v):
15         T.addEdge(u, v)
16         forest.union( forest.find(u) ,
17                     forest.find(v) )
18
19 return T
    
```

**Kruskal's Algorithm**



(A, D)
(E, H)
(F, G)
(A, B)
(B, D)
(G, E)
(G, H)
(E, C)
(C, H)
(E, F)
(F, C)
(D, E)
(B, C)
(C, D)
(A, F)
(D, F)

**CS 225 – Things To Be Doing:**

1. Programming Exam C: Thursday, April 18 – Sunday, April 21
2. **MP7 Released: Part 1 due April 22<sup>nd</sup>!**
3. lab\_ml this week in lab
4. Daily POTDs are ongoing for +1 point /problem