Breadth

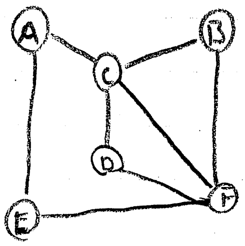
Alg 11/21/17

Breadth-First → Look at all neighbors. Look wide before deep

- Explores all options at equal rate
- Will find the shortest path
  (since all path potentials progress at same rate)

BFS Traversal → Enqueue start, then neighbors, and neighbors of those ...
Also, standard is to mark node as "Checked"
when enqueueing to avoid double enqueueing

- Generally a style for order of checking (e.g. alphabetical)
- Keep traversing until queue is empty



☆ Queue is FIFO

D → Enq (C, F) → C → Enq (A, B) → F → Enq(E)
→ A → B → E

```
Node* BFS (Node* n, T value) {
   n → Checked = true;
   Queue q;
   q. enq(n);
   While ( ! q.isEmpty) {
      For ( Node* neighbor : n → neighbors) {
          if(! neighbor → checked) {
             neighbor → checked = true;
             q. enq(neighbor)
          }
      }
      if(n → value == value)
         return n;
      n = q. deq();
   }
   return null;
}
```

Alg

What time does this run in?

- Outer loop O(vertices)
- Inner loop O(edges)

Multiply? No eventhough nested

$\hookrightarrow \Theta(|V| + 2|E|)$

What if directional (or disconnected) graph?

→ Starting point matters, may not find all neighbors.

- Sol^{n} is to have list all nodes to verify if all have been reached, and jump to any that aren't

$\Theta(|V| + |E|)$

. h → Contract/Declarations only (Exception is for "friend")

. cpp → Strictly implementation