17.2.1
BFS with distances and layers
**BFS with distances**

**BFS(s)**
Mark all vertices as unvisited; for each $v$ set $\text{dist}(v) = \infty$
Initialize search tree $T$ to be empty
Mark vertex $s$ as visited and set $\text{dist}(s) = 0$
set $Q$ to be the empty queue
**enqueue(s)**
**while** $Q$ is nonempty **do**
  $u = \text{dequeue}(Q)$
  **for** each vertex $v \in \text{Adj}(u) **do**$
    **if** $v$ is not visited **do**
      **if** $v$ is not visited **do**
        add edge $(u, v)$ to $T$
        Mark $v$ as visited, **enqueue(v)**
        and set $\text{dist}(v) = \text{dist}(u) + 1$
Properties of BFS: Undirected Graphs

Theorem

The following properties hold upon termination of BFS(s):

(A) Search tree contains exactly the set of vertices in the connected component of s.

(B) If $\text{dist}(u) < \text{dist}(v)$ then $u$ is visited before $v$.

(C) For every vertex $u$, $\text{dist}(u)$ is the length of a shortest path (in terms of number of edges) from $s$ to $u$.

(D) If $u, v$ are in connected component of $s$ and $e = \{u, v\}$ is an edge of $G$, then $|\text{dist}(u) - \text{dist}(v)| \leq 1$. 
Properties of **BFS**: Directed Graphs

### Theorem

The following properties hold upon termination of **BFS**(*s)*:

- **A.** The search tree contains exactly the set of vertices reachable from *s*
- **B.** If \( \text{dist}(u) < \text{dist}(v) \) then *u* is visited before *v*
- **C.** For every vertex *u*, \( \text{dist}(u) \) is indeed the length of shortest path from *s* to *u*
- **D.** If *u* is reachable from *s* and \( e = (u, v) \) is an edge of *G*, then \( \text{dist}(v) - \text{dist}(u) \leq 1 \).

*Not necessarily the case that \( \text{dist}(u) - \text{dist}(v) \leq 1 \).*
BFS with Layers

**BFSLayers**($s$):
Mark all vertices as unvisited and initialize $T$ to be empty
Mark $s$ as visited and set $L_0 = \{s\}$
$i = 0$
while $L_i$ is not empty do
    initialize $L_{i+1}$ to be an empty list
    for each $u$ in $L_i$ do
        for each edge $(u, v) \in \text{Adj}(u)$ do
            if $v$ is not visited
                mark $v$ as visited
                add $(u, v)$ to tree $T$
                add $v$ to $L_{i+1}$
    $i = i + 1$

Running time: $O(n + m)$
**BFS with Layers**

**BFS\text{Layers}(s):**
Mark all vertices as unvisited and initialize $T$ to be empty
Mark $s$ as visited and set $L_0 = \{s\}$

$i = 0$

while $L_i$ is not empty do
  initialize $L_{i+1}$ to be an empty list
  for each $u$ in $L_i$ do
    for each edge $(u, v) \in \text{Adj}(u)$ do
      if $v$ is not visited
        mark $v$ as visited
        add $(u, v)$ to tree $T$
        add $v$ to $L_{i+1}$
  
  $i = i + 1$

Running time: $O(n + m)$
Example
BFS with Layers: Properties

**Proposition**

The following properties hold on termination of $\text{BFS Layers}(s)$.

1. $\text{BFS Layers}(s)$ outputs a BFS tree
2. $L_i$ is the set of vertices at distance exactly $i$ from $s$
3. If $G$ is undirected, each edge $e = \{u, v\}$ is one of three types:
   - *tree* edge between two consecutive layers
   - *non-tree* forward/backward edge between two consecutive layers
   - *non-tree* cross-edge with both $u, v$ in same layer
4. $\implies$ Every edge in the graph is either between two vertices that are either (i) in the same layer, or (ii) in two consecutive layers.
Definition

A directed graph (also called a digraph) is \( G = (V, E) \), where

- \( V \) is a set of vertices or nodes,
- \( E \subseteq V \times V \) is set of ordered pairs of vertices called edges.
BFS with Layers: Properties

For directed graphs

**Proposition**

The following properties hold on termination of \( \text{BFSLayers}(s) \), if \( G \) is directed. For each edge \( e = (u, v) \) is one of four types:

1. a **tree** edge between consecutive layers, \( u \in L_i, v \in L_{i+1} \) for some \( i \geq 0 \)
2. a non-tree **forward** edge between consecutive layers
3. a non-tree **backward** edge
4. a **cross-edge** with both \( u, v \) in same layer
THE END

...

(for now)