17.2.1
BFS with distances and layers

BFS with distances

```
BFS(s)
    Mark all vertices as unvisited; for each v set dist(v) = ∞
    Initialize search tree T to be empty
    Mark vertex s as visited and set dist(s) = 0
    set Q to be the empty queue
    enqueue(s)
    while Q is nonempty do
        u = dequeue(Q)
        for each vertex v ∈ Adj(u) do
            if v is not visited do
                add edge (u, v) to T
                Mark v as visited, enqueue(v)
                and set dist(v) = 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* $\mathrm{dist}(u) < \mathrm{dist}(v)$ *then* u *is visited before* v.
- (C) *For every vertex* u, $\mathrm{dist}(u)$ *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* $|\mathrm{dist}(u) - \mathrm{dist}(v)| \leq 1$.

Properties of BFS: Directed Graphs

Theorem

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

1. *The search tree contains exactly the set of vertices reachable from* s
2. *If* $\mathrm{dist}(u) < \mathrm{dist}(v)$ *then* u *is visited before* v
3. *For every vertex* u, $\mathrm{dist}(u)$ *is indeed the length of shortest path from* s *to* u
4. *If* u *is reachable from* s *and* $e = (u, v)$ *is an edge of* G, *then* $\mathrm{dist}(v) - \mathrm{dist}(u) \leq 1$.
   Not necessarily the case that $\mathrm{dist}(u) - \mathrm{dist}(v) \leq 1$.

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) ∈ 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
```

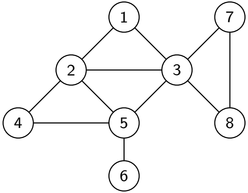
Running time: $O(n + m)$

BFS with Layers

```
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) ∈ 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



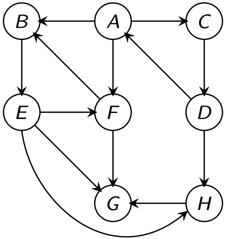
BFS with Layers: Properties

Proposition

The following properties hold on termination of **BFSLayers**(s).

1. **BFSLayers**(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:*
    1. ***tree*** *edge between two consecutive layers*
    2. *non-tree* ***forward***/***backward*** *edge between two consecutive layers*
    3. *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.*

Example



BFS with Layers: Properties

For directed graphs

Proposition

The following properties hold on termination of **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)