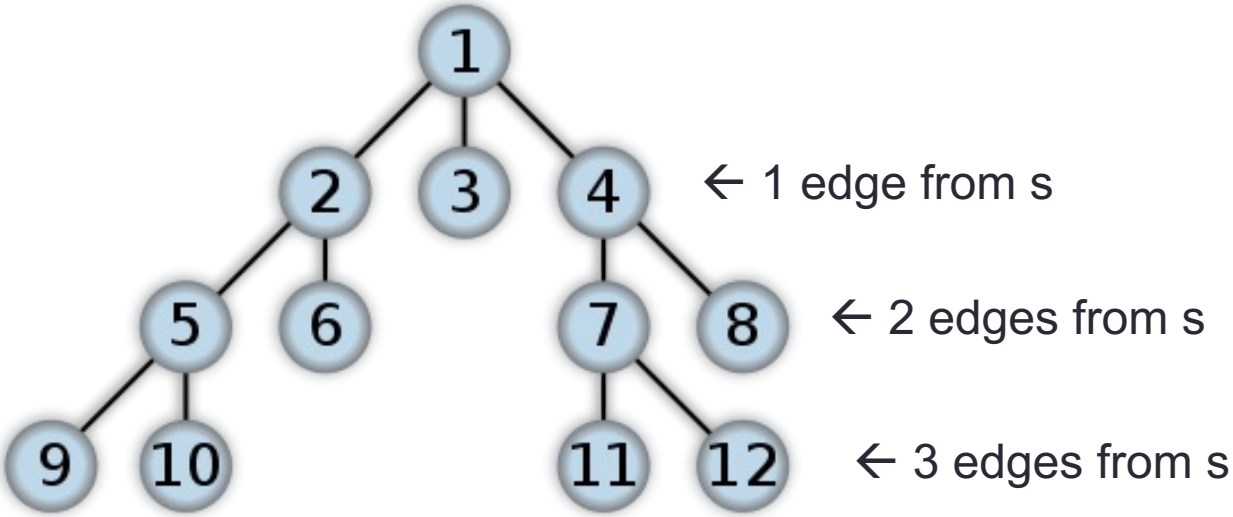
BREADTH FIRST SEARCH AND DEPTH FIRST SEARCH

CS340

Breadth First Search

- Input: A graph and a source vertex, s
- Send a **wave** out from s.
- First hits all vertices 1 edge from s.
- From there, hits all vertices 2 edges from s.
- Etc.
- Use FIFO queue Q to maintain wavefront.

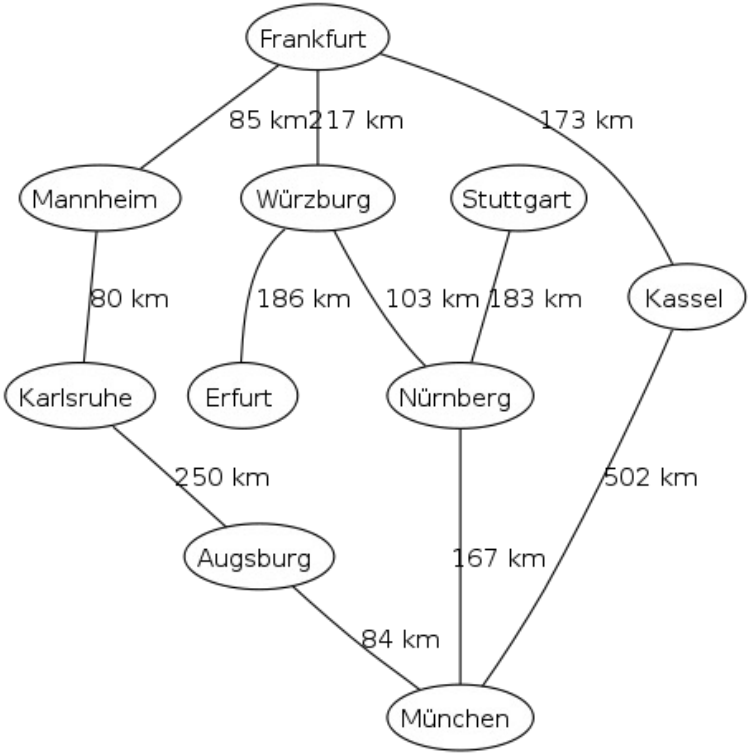
The "wave" of BFS

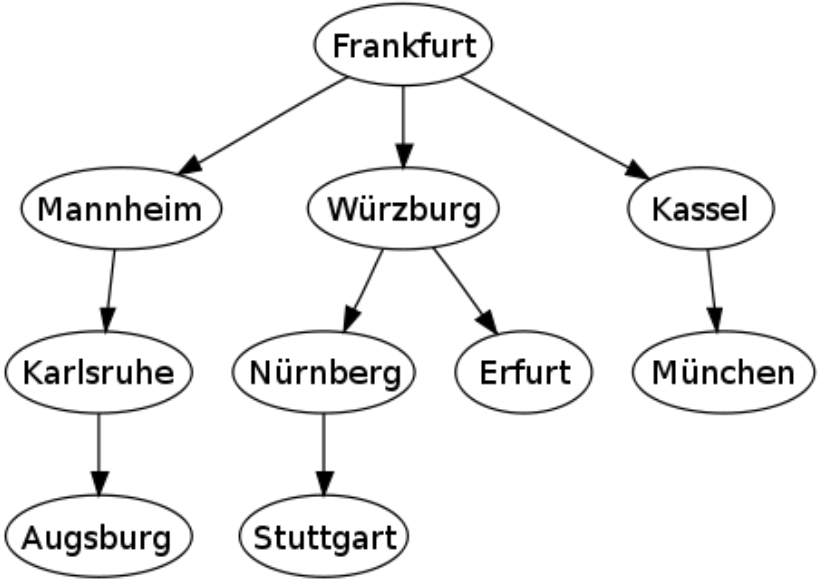


Breadth-First Search

- Discovers every vertex that is reachable from the source
- Computes distance (in edges) from source to each reachable vertex
- Can construct a breadth-first tree of reachable vertices
- BFS may not reach all vertices

Breadth-First Search starting at Frankfort



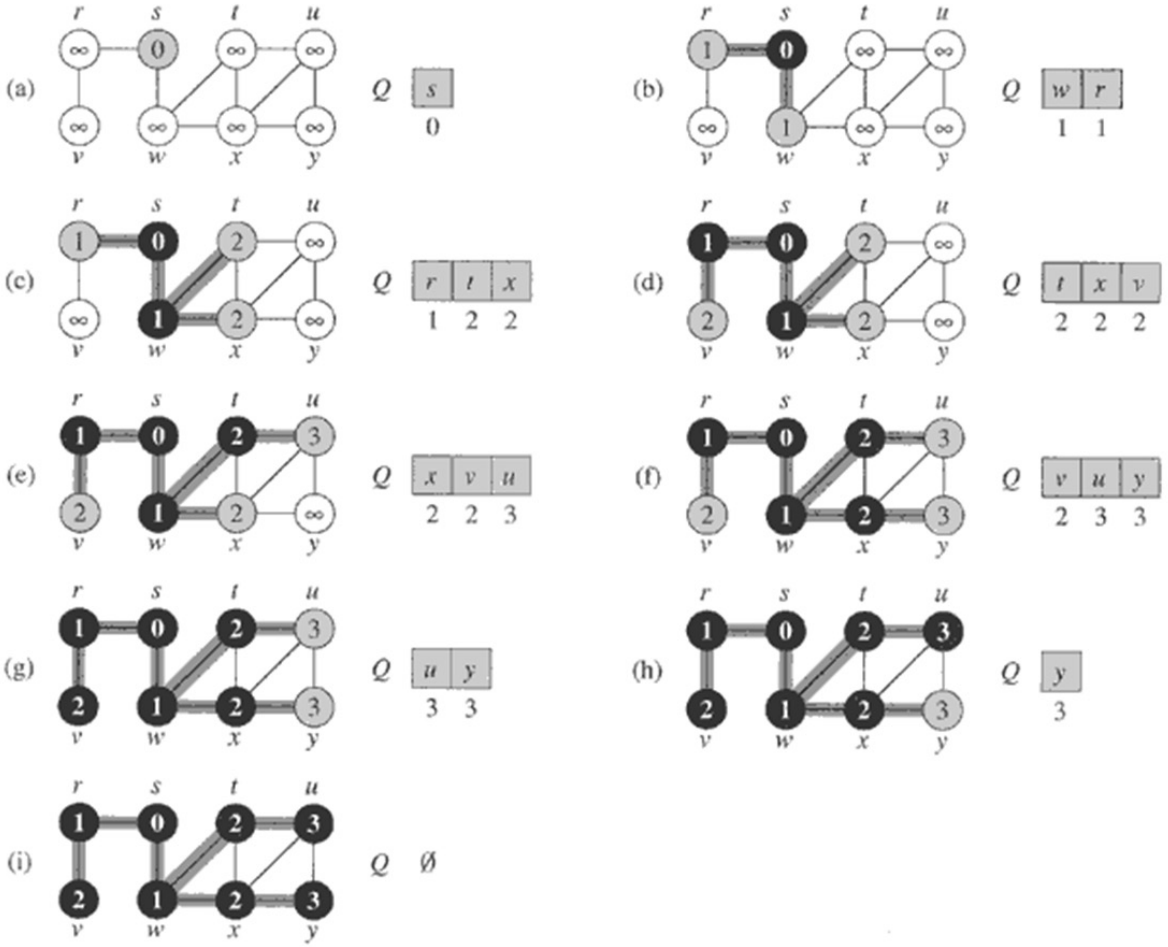


BFS

- All vertices start out white
- When a vertex is discovered (and put into the queue) it is colored gray. Gray vertices have not yet had their adjacency lists fully examined.
- When a vertex is removed from the queue it is colored black
- The queue consists only of gray vertices

```
BFS(G, s)
 1  for each vertex u ∈ V[G] − {s}
 2       do color[u] ← WHITE
 3          d[u] ← ∞
 4          π[u] ← NIL
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       do u ← DEQUEUE(Q)
12          for each v ∈ Adj[u]
13               do if color[v] = WHITE
14                    then color[v] ← GRAY
15                         d[v] ← d[u] + 1
16                         π[v] ← u
17                         ENQUEUE(Q, v)
18          color[u] ← BLACK
```

BFS

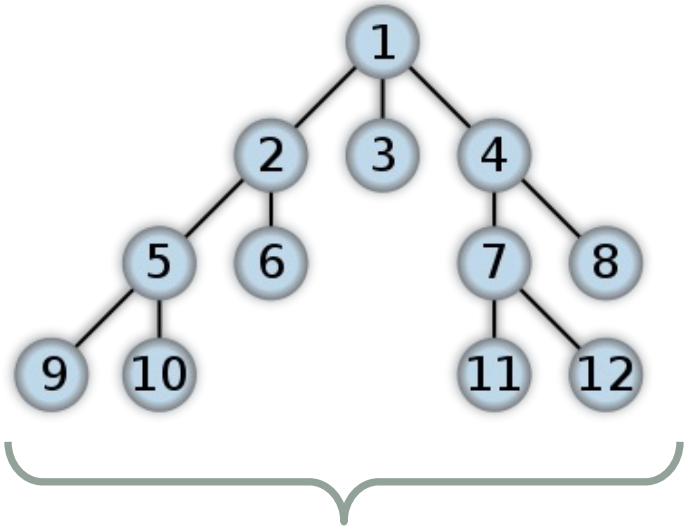

(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
r s t u
v w x y
Q
s
0
w r
1 1
r t x
1 2 2
t x v
2 2 2
x v u
2 2 3
v u y
2 3 3
u y
3 3
y
3
Ø

Running time

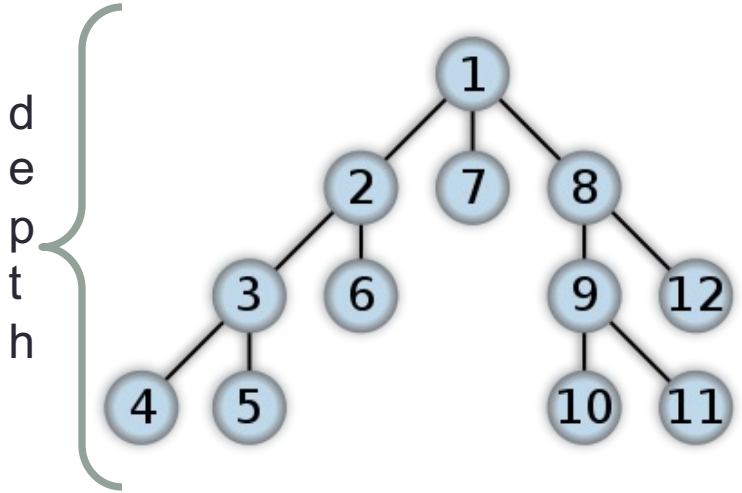
- Time O(V + E)
- V contribution: Initialization and while loop iterations in which every vertex is enqueued at most once.
- O(V+E): Every vertex u is also dequeued at most once in the main while loop. For each u, the inner for loop iterates through all the *neighbors* v of u.
  - How many *total* times does line 12 execute?
  - Recall: $2E = \sum_{v \in V} degree(v)$

Depth first search

Search "deeper" whenever possible



breadth



*example shows discovery times

Depth first search

- Input: G = (V,E), directed or undirected.
  No source vertex is given!
- Output: **2 timestamps** on each vertex:
  - v.d **discovery time**
  - v.f **finishing time**
- These will be useful for other algorithms later on.
- Can also compute v.π

Depth first search

- Will methodically explore every edge.
  - Start over from different vertices as necessary.
- As soon as we discover a vertex, explore from it.
  - Unlike BFS, which puts a vertex on a queue so that we explore from it later.
- DFS may repeat from multiple source nodes
  - Unlike BFS, result is a forest of DFS trees

Depth first search

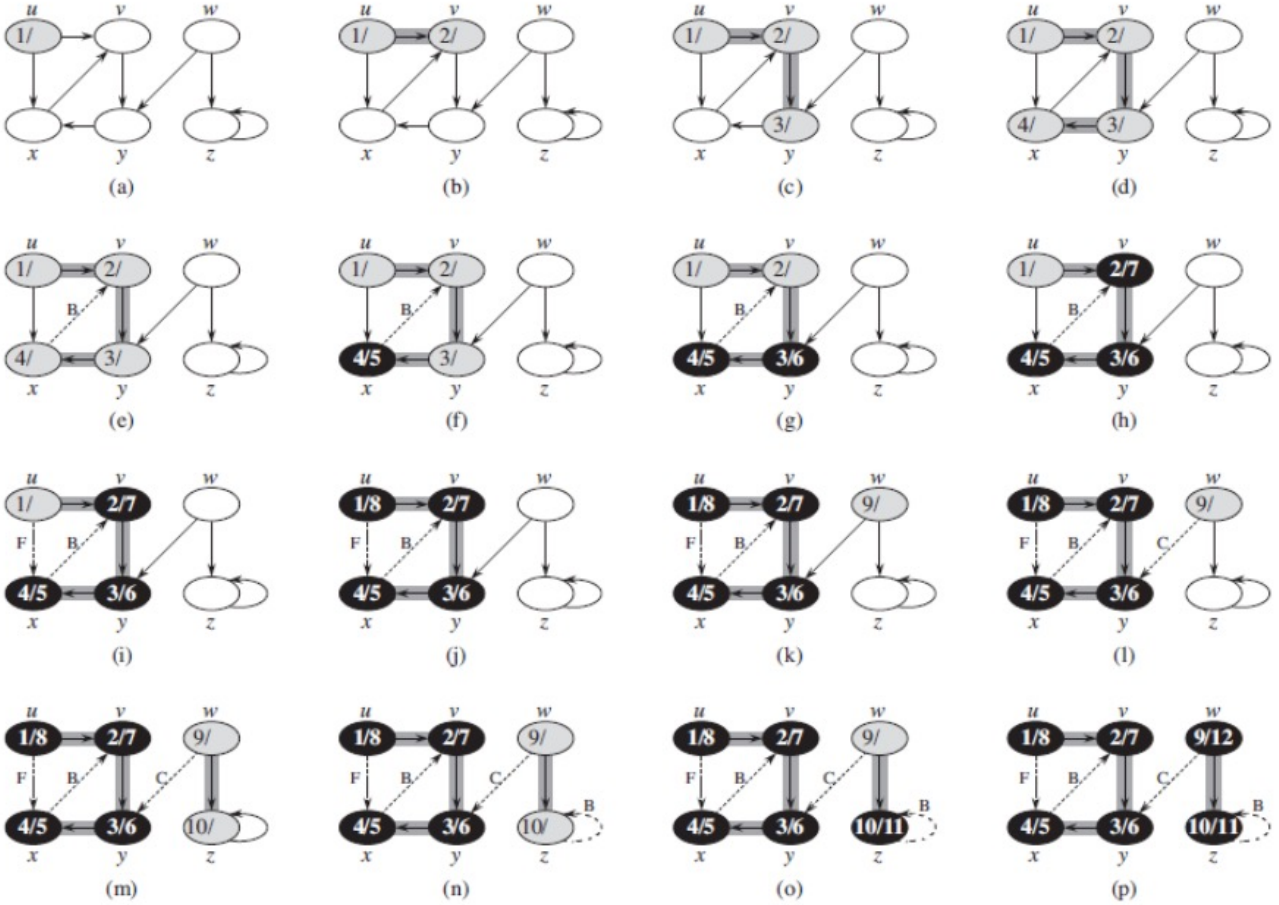
- As DFS progresses, every vertex has a color:
  - WHITE = undiscovered
  - GRAY = discovered, but not finished (not done exploring from it)
  - BLACK = finished (have found everything reachable from it)
- Discovery and finishing times:
  - Unique integers from 1 to 2|V|
  - For all v, v.d < v.f
- In other words, $1 \leq v.d < v.f \leq 2|V|$

DFS, recursive version

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
 1  time = time + 1          // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]    // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK          // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

DFS



Time complexity

- The procedure DFS-VISIT is called exactly once for each vertex $v \in V$, since the vertex u on which DFS-VISIT is invoked must be white and the first thing DFS-VISIT does is paint vertex u gray.
- During an execution of DFS-VISIT, the loop on lines 4–7 executes *Adj*[E] times = $\Theta(E)$.
- The running time of DFS is therefore $\Theta(V + E)$
- Notice that BFS was $O(V + E)$ because it was not certain that every vertex would be visited.

Classification of edges

- **Tree** edge: in the depth-first forest. Found by exploring (u, v).
- **Back** edge: (u,v), where u is a descendant of v.
- **Forward** edge: (u,v), where v is a descendant of u, but not a tree edge.
- **Cross** edge: any other edge. Can go between vertices in same depth-first tree or in different depth-first trees.

Classification of edges

- When we first explore an edge (u,v), the color of
- vertex v tells us something about the edge:
- 1. WHITE indicates a tree edge,
- 2. GRAY indicates a back edge, and
- 3. BLACK indicates a forward or cross edge.

Classification of edges

- Forward and cross edges never occur in a depth-first search of an undirected graph.
- In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.