CSE 640:
Graph Mining and Management

Lecture 5 (Feb 21)

A. Erdem Sariyuce

A couple updates

- Will send feedback for proposal reports
  - All grades will be posted on AutoLab

- Homework 1 is out on Wed (2/23)
  - Due the following Wed (3/2)

Graph traversal

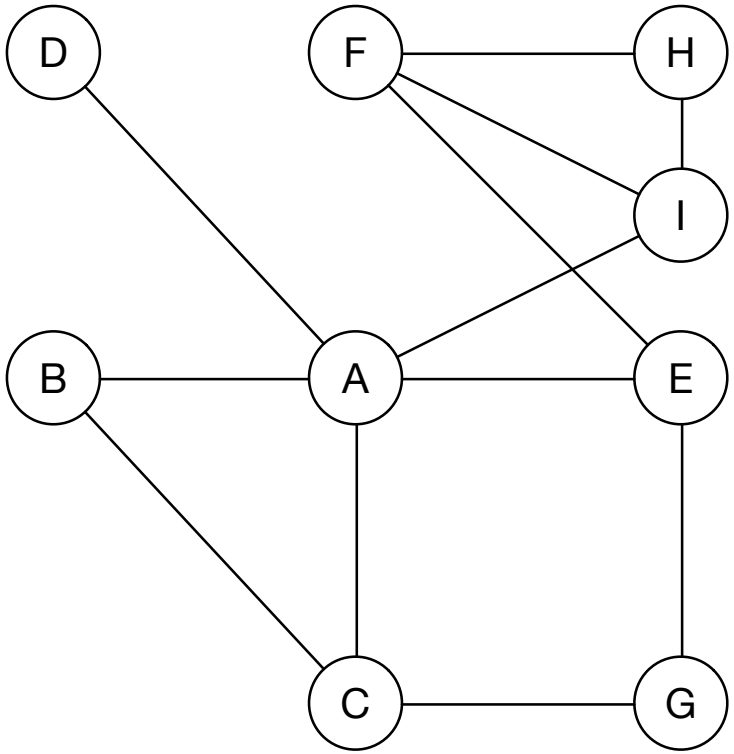
Fundamental building block

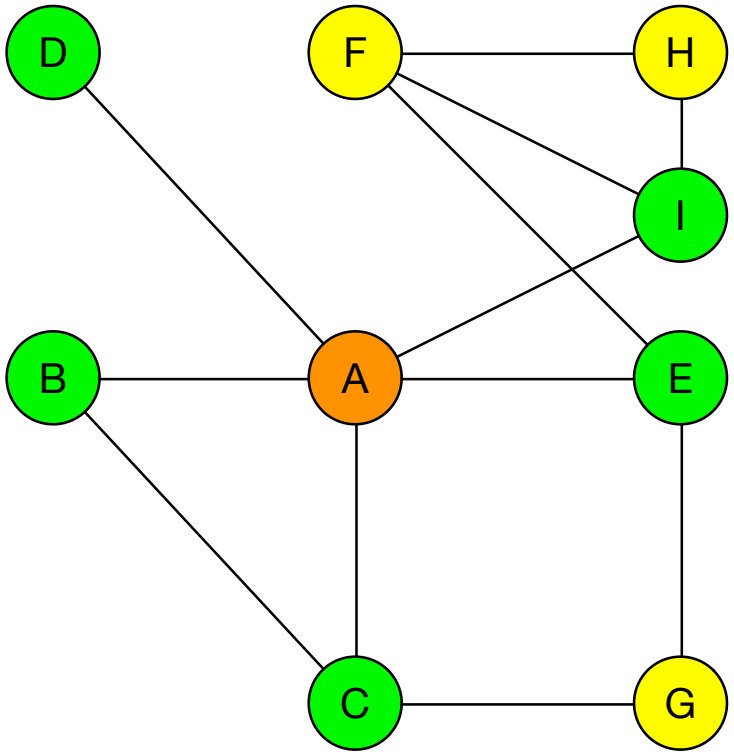
- Graph traversal is part of many important tasks
  - Connected components
  - Tree/Cycle detection
  - Articulation vertex finding
- Real-world applications
  - Peer-to-peer networks: Discover what's around
    - Broadcasting
  - Web crawlers
  - Notion of proximity
  - GPS navigation systems
  - Garbage collection

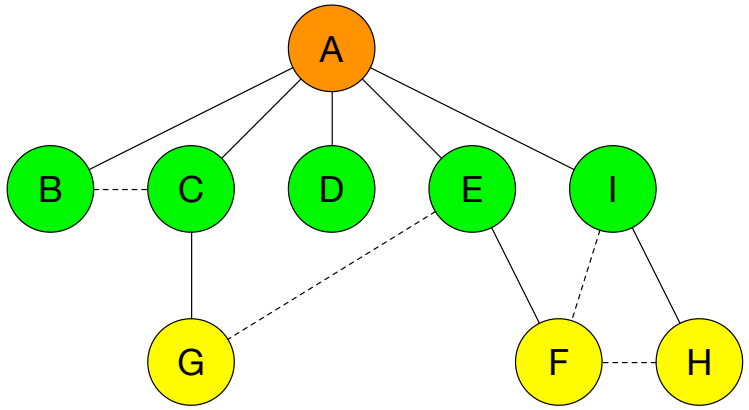
Today

- Breadth First Search (BFS)
- Depth First Search (DFS)
- How to leverage characteristics of real-world networks?
- Applications of DFS
  - Detecting cycles
  - Topological sort
  - Finding SCCs

Breadth-first search







How do you implement BFS?

- Input: Graph G (n=|V|, m=|E|) and vertex u
- Output: Levels of vertices, parents of vertices

- Level of u is 0, parent of u is -1
- Initially k=0 (current level)

How do you implement BFS?

- Input: Graph G (n=|V|, m=|E|) and vertex u
- Output: Levels of vertices, parents of vertices

- Level of u is 0, parent of u is -1
- Initially k=0 (current level)
- Repeat the following
  - Find all vertices with level k and put them in set N
    - Stop if no vertex found
  - For each vertex v in N
    - Check each of its neighbors
    - If no level assigned yet, set its level as k+1 and its parent as v
  - Increment k

How do you implement BFS?

- Input: Graph G (n=|V|, m=|E|) and vertex u
- Output: Levels of vertices, parents of vertices

- Level of u is 0, parent of u is -1
- Initially k=0 (current level)
- Repeat the following
  - Find all vertices with level k and put them in set N
    - Stop if no vertex found
  - For each vertex v in N
    - Check each of its neighbors
    - If no level assigned yet, set its level as k+1 and its parent as v
  - Increment k

Complexity?
- Initialize level, parent arrays: O(n)
- Scanning level information: O(n)
  - r levels: O(rn)
- Checking neighbors: O(m)

- Total: O(n+rn+m)
- Worst case
  - r is n (chain)
  - O ($m + n^2$)

Can we do better?

- Aim for the most expensive part
  - Scanning level information; O(rn)
  - For real-world networks, what's the expected value of r?

- We can store the vertices in the next level
  - So, one array for the current level vertices, another for the next
  - Combine them, just make first come-first serve
    - That's called queue

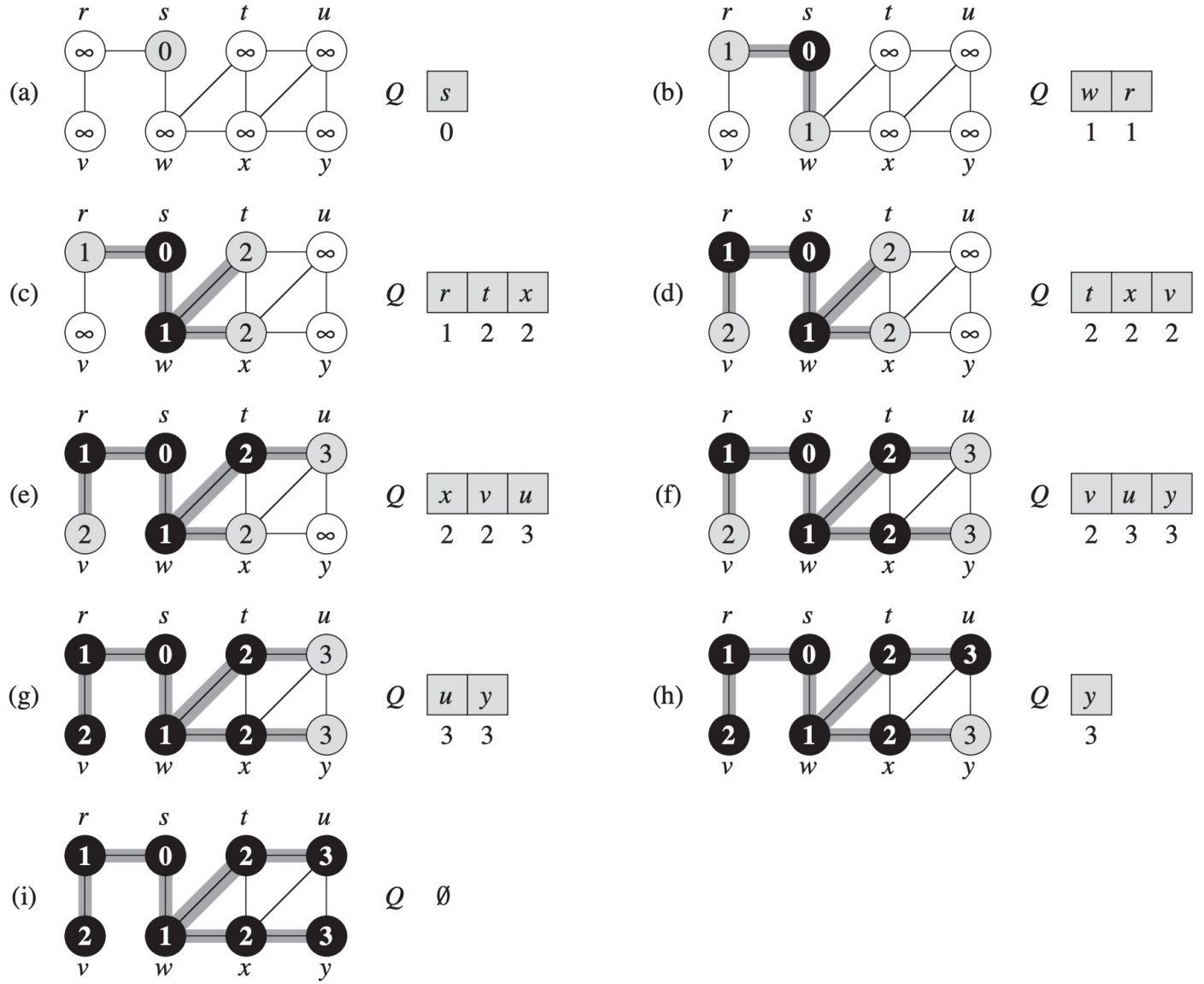
  - So, complexity becomes O(m)

$O(n^2)$ to $O(m)$ if you store the frontier

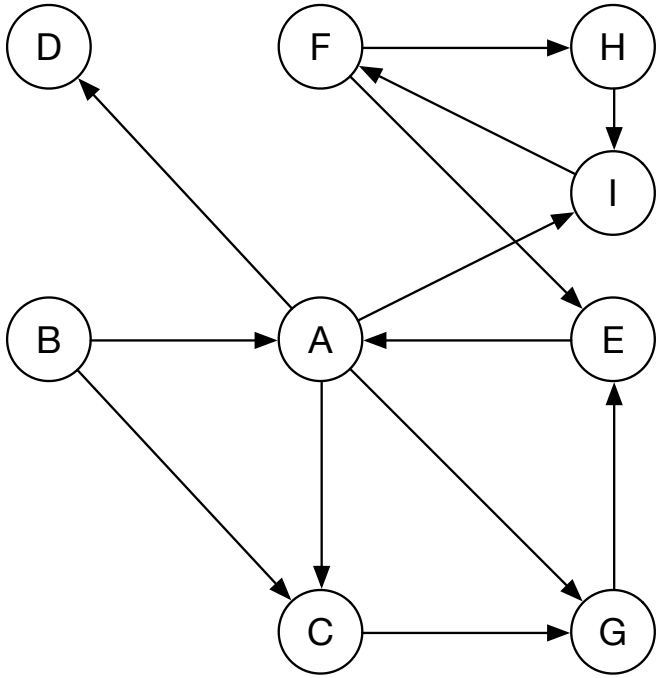
```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color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

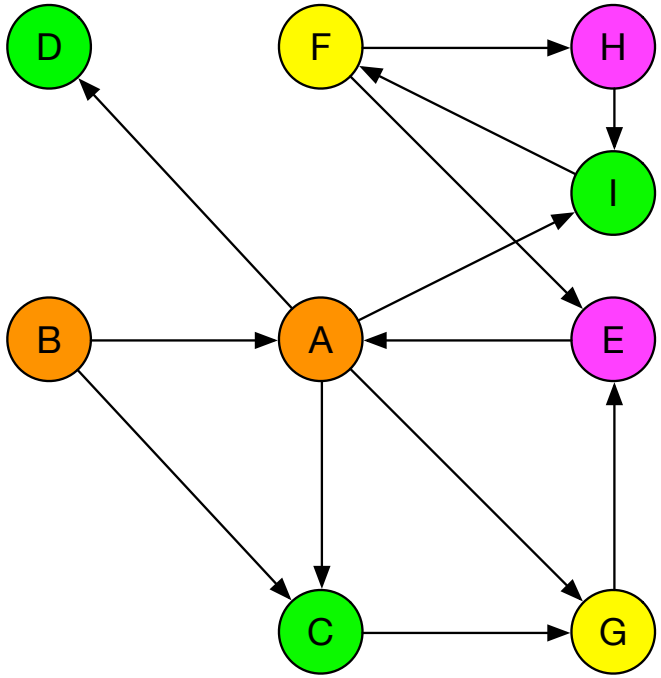
$O(n^2)$ to $O(m)$ if you store the frontier

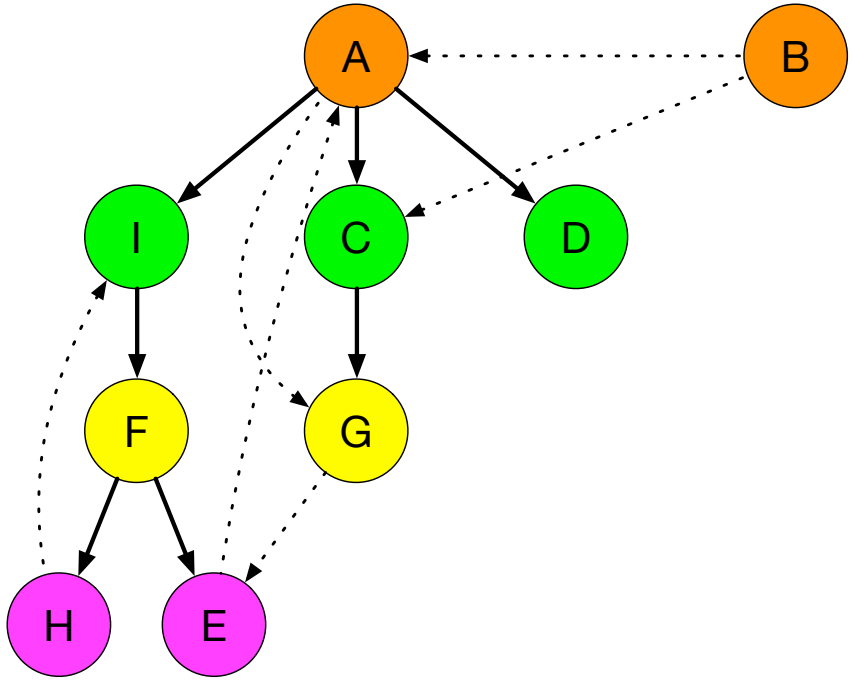
```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color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



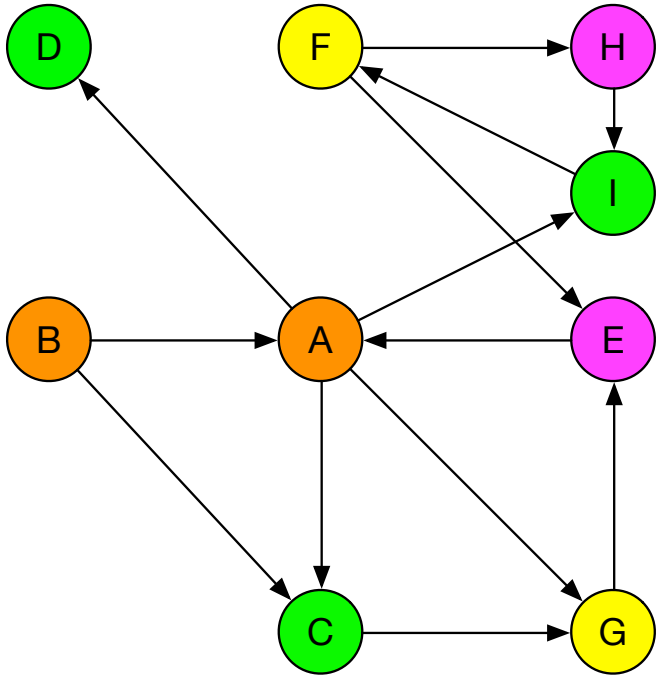
Depth-first search

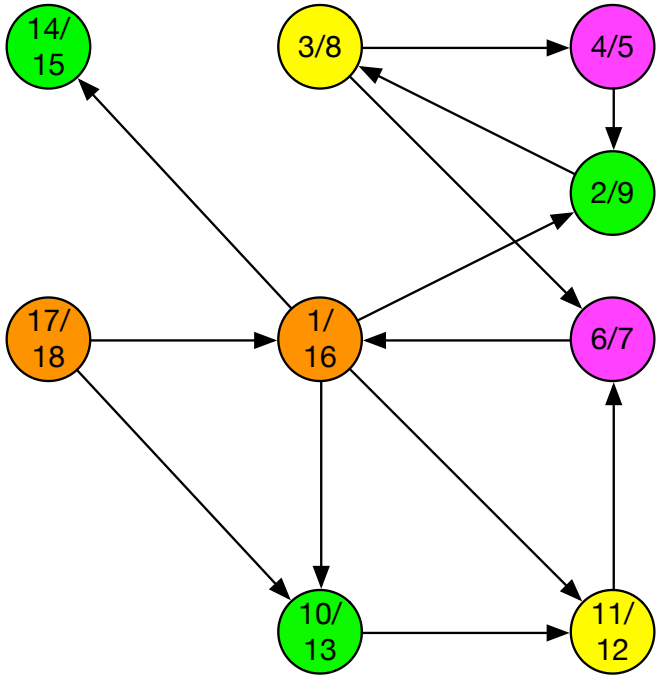






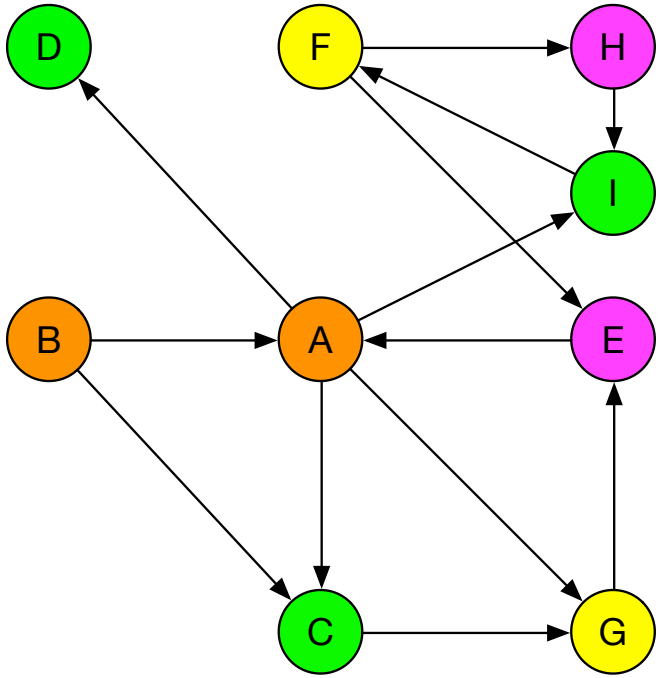
Depth-first search

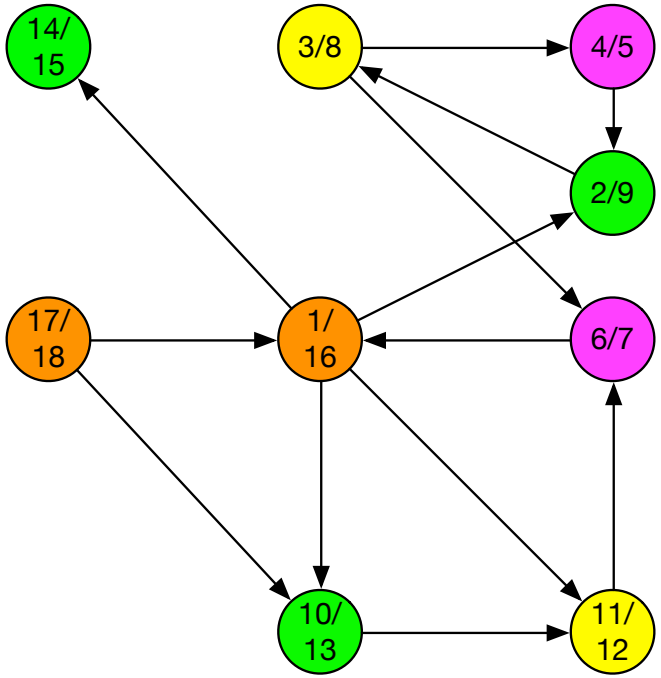


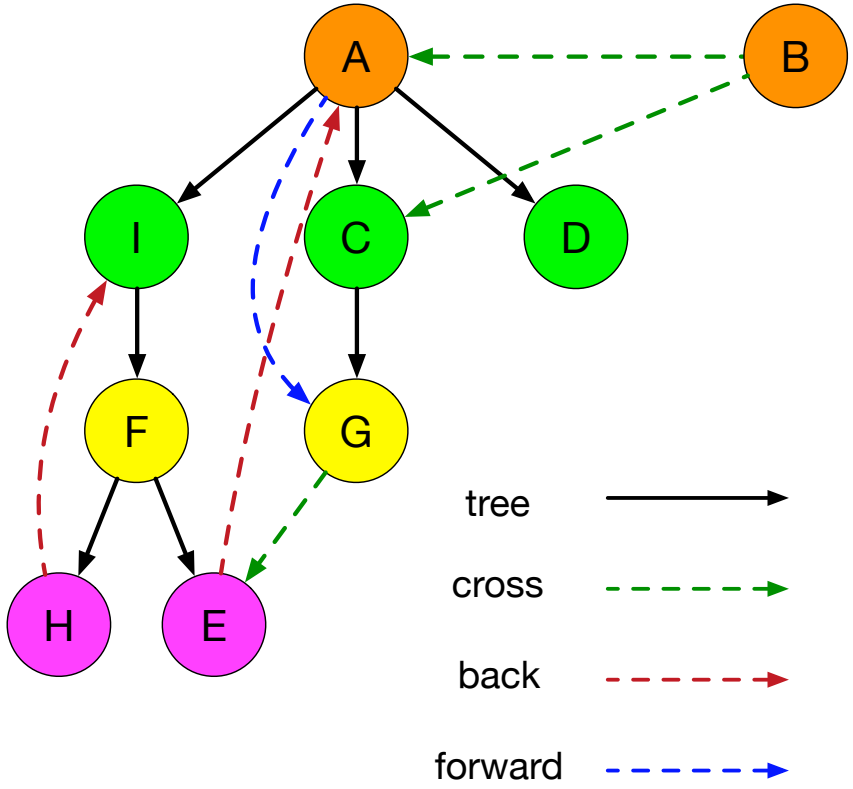


- Starting and finishing times

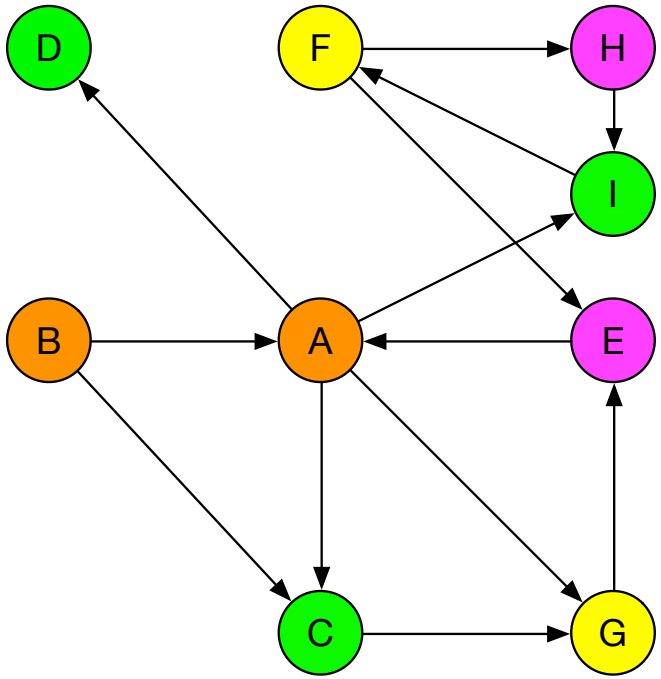
Depth-first search

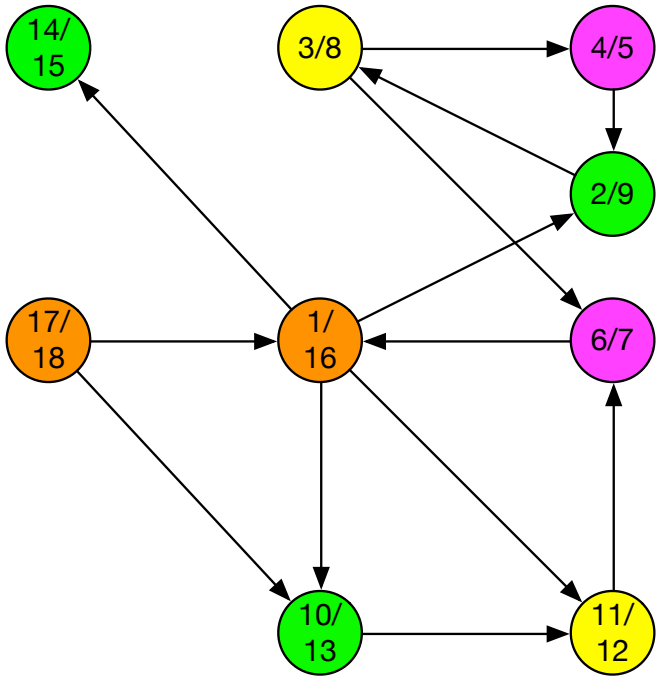


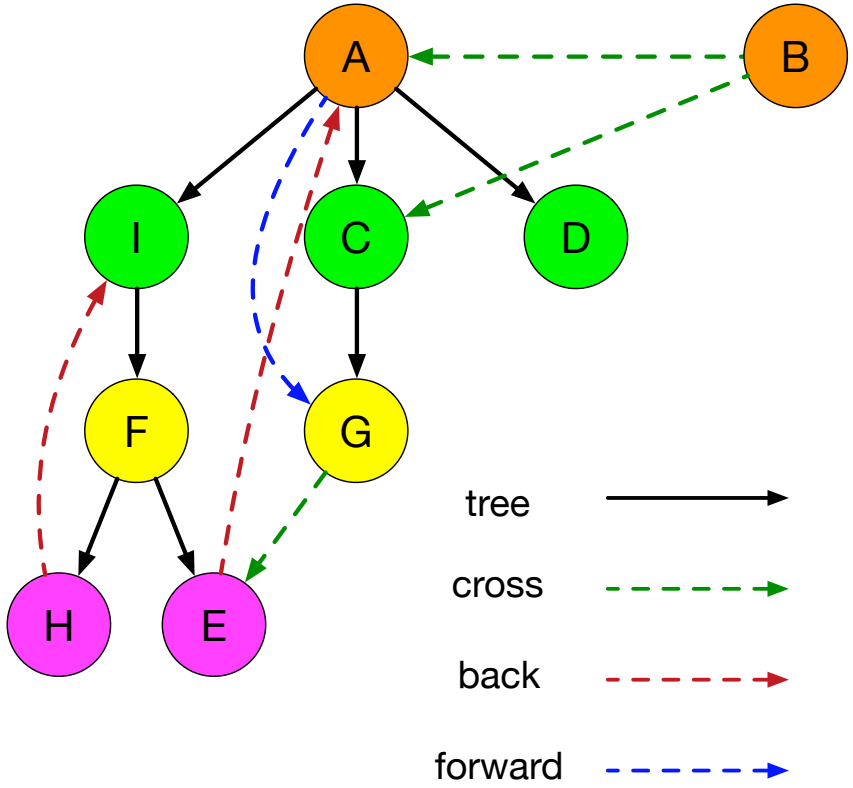




Depth-first search







- What types of edges can be observed in BFS?

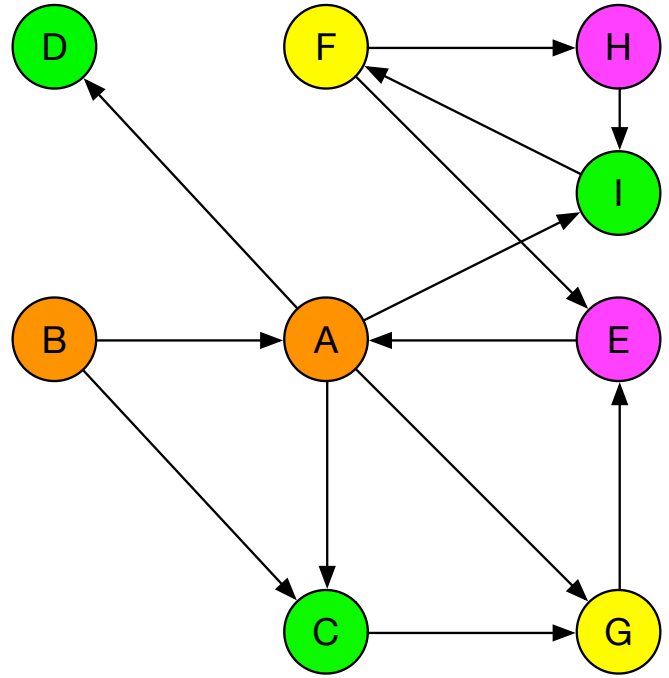
How to implement?

- Notice the repetitive pattern
  - Going deep
- Recursion
- Define function DFS

- DFS(G, u, levels, k)
  - Return if levels[u] is assigned
  - Set levels[u] = k
  - For all vertices v which are neighbor to u
    - DFS (G, v, levels, k+1)

Recursion is a stack

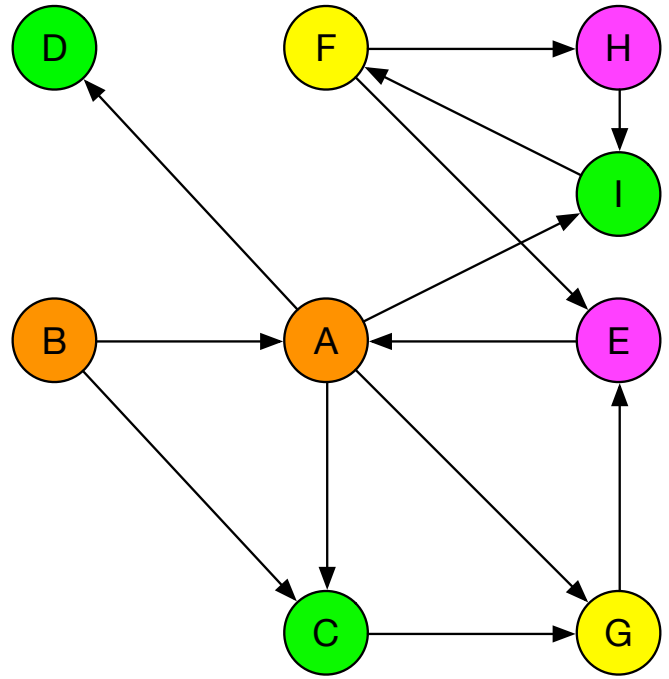
- First in Last Out (FILO)
- Any recursion can be performed by a stack
  - Indeed, CPU literally does that!
- Start from vertex A



- Complexity?

Recursion is a stack

- First in Last Out (FILO)
- Any recursion can be performed by a stack
  - Indeed, CPU literally does that!
- Start from vertex A



- Complexity?
  - O(m)

BFS vs. DFS

- Which ones sounds more natural?

- Which one seems to be easier to be parallelized ?
  - Watch for race conditions

- Queue vs. stack

Sometimes you need multiple BFSs

- Coarse-level parallelism
  - Each BFS by a processing unit

- Betweenness centrality
- Closeness centrality
- One-to-all shortest paths

- Next week!

How to adapt your algorithm for real-world data?

- BFS is more popular (Graph500)

How to adapt your algorithm for real-world data?

- BFS is more popular (Graph500)
- What characteristic of real-world networks can be used to optimize BFS?
  - Sparseness?
  - Diameter?

How to adapt your algorithm for real-world data?

- BFS is more popular (Graph500)
- What characteristic of real-world networks can be used to optimize BFS?
  - Sparseness?
  - Diameter?

- Direction-Optimizing Breadth-First Search, SC 2012
  - By Scott Beamer, Krste Asanovic, and David Patterson
  - http://www.scottbeamer.net/pubs/beamer-sc2012.pdf