Graphs!

About me

John Wright

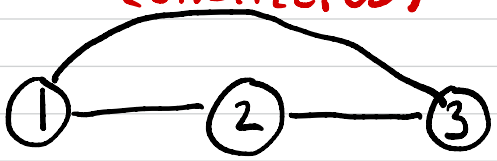
4th year at Berkeley

Office: Soda 683

Office hours: Tuesday 3-4pm

Graphs

(undirected)



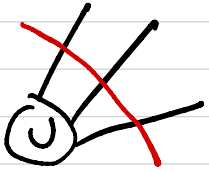
$G = (V, E)$

(directed)

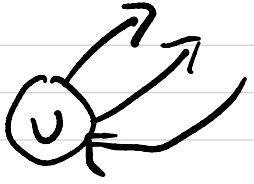
$(u,v) \in E$ if $u \to v$

Parameters: $n = |V|$

$m = |E|$ $\quad (m \leq n^2)$



$deg(u) = 3$



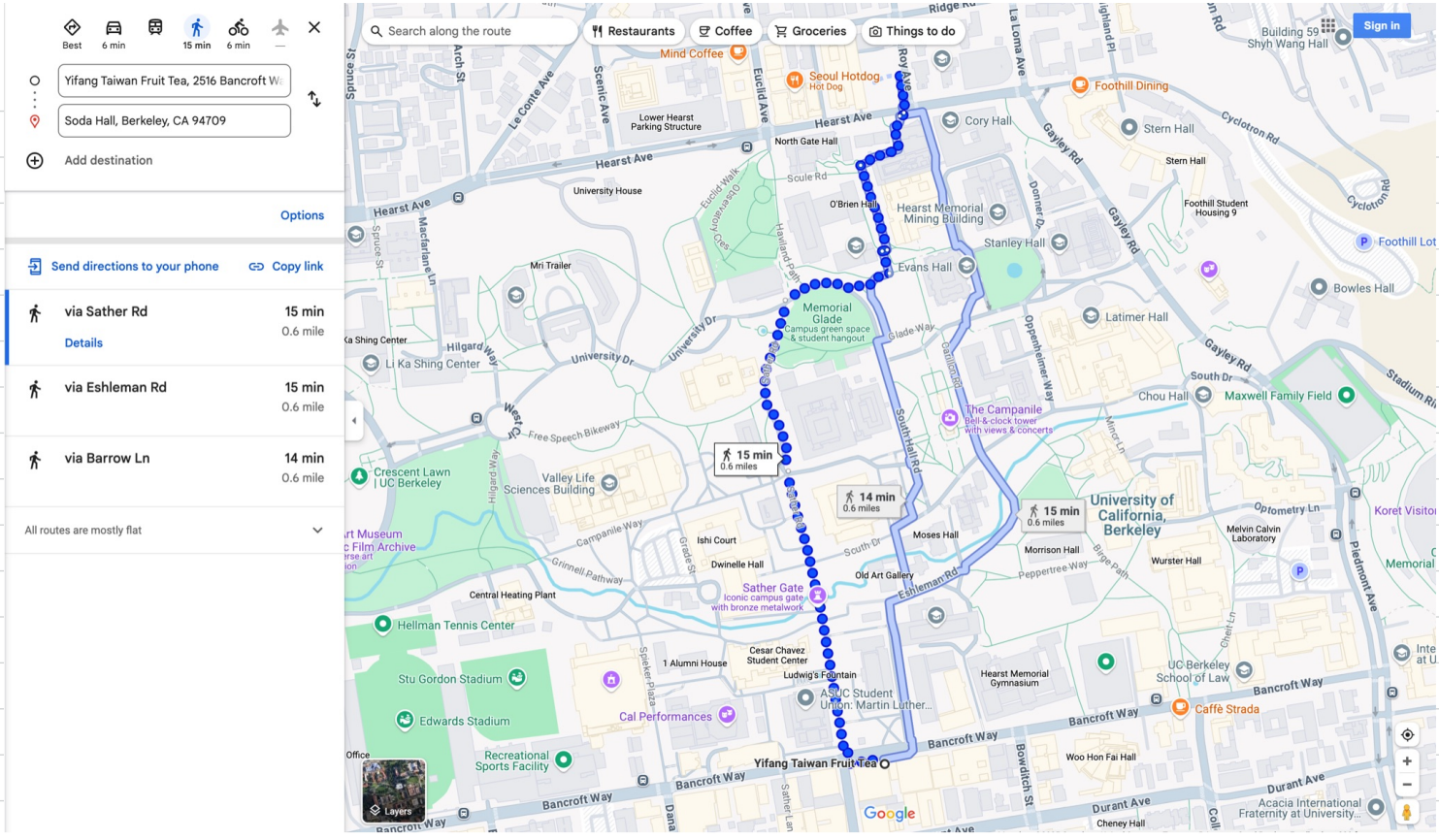
$in\text{-}deg(u) = 1$

$out\text{-}deg(u) = 2$

Facebook friend graph (undirected)

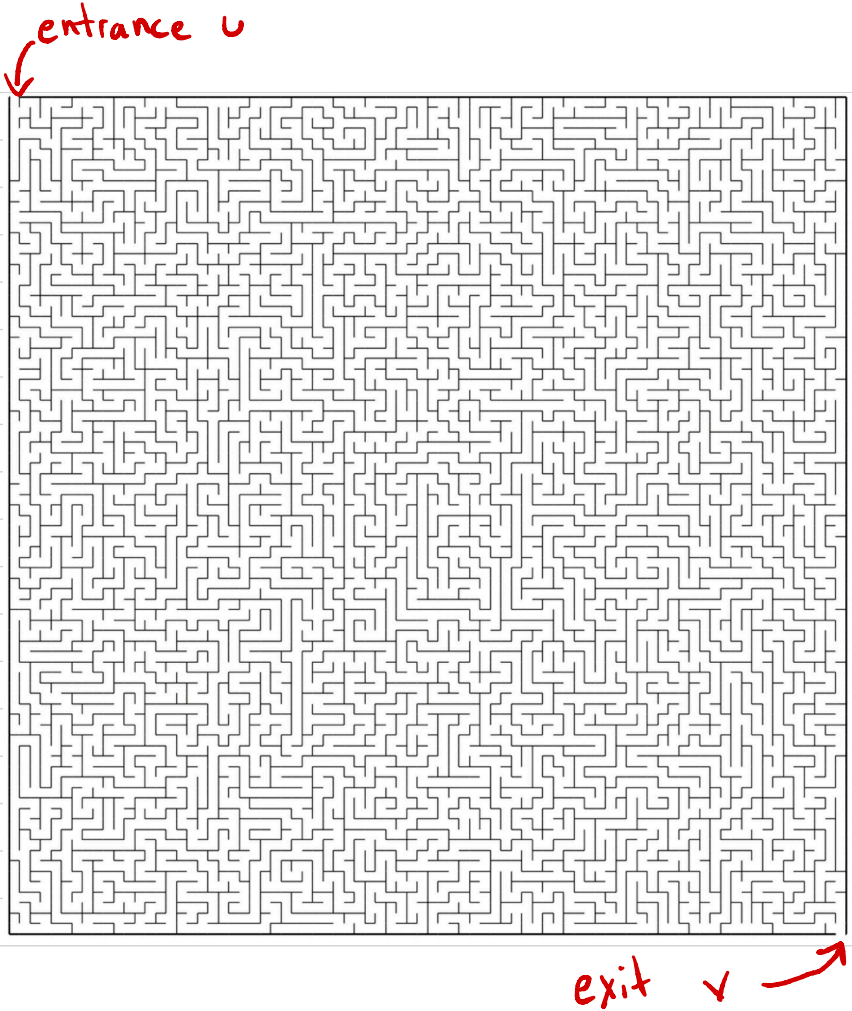
- $n = 2.91$ billion users
- avg user has 338 friends
  $m =$ 500 billion edges

Question: Who are my friends?



Google Maps

Q: Shortest path to boba?



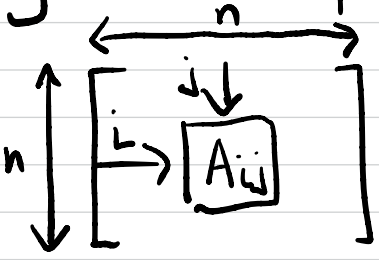
Maze solving

Q: Is u connected to v?

Representing graphs on computers

$V = \{1, \ldots, n\}$

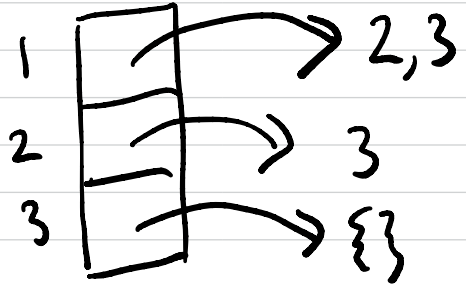
(1) adjacency matrix representation



$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{o.w.} \end{cases}$$

(2) adjacency list representation

(unordered)



(linked list)

		Adjacency matrix	Adjacency list
	size		
time to:	answer is $(u,v) \in E$?		
	enumerate u's neighbors		

Facebook graph w/ adacency matrix

size $n^2 = (2.91 \text{ billion})^2$ space

$= 8.5 \times 10^{18} = 1$ million TB

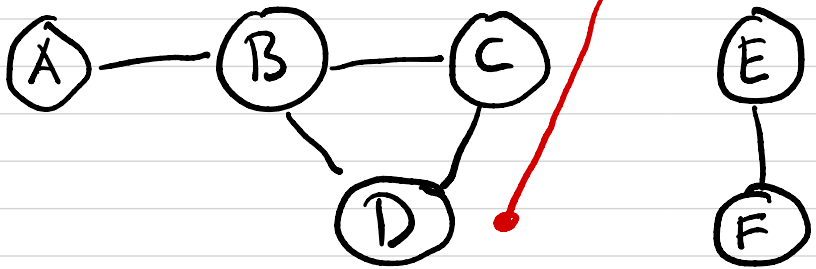
computing friends list = 2.91 billion steps

Graph Exploration

Useful for:

1. Is there a path from u to v?
2. Is G connected?
3. What are G's connected components?

Connectivity for undirected graphs (∃ path from u to v?)




```
explore(G,u)
  visited[u] = true

  for v s.t. (u,v) ∈ E
      if visited[v] = false
          explore(G,v)
```

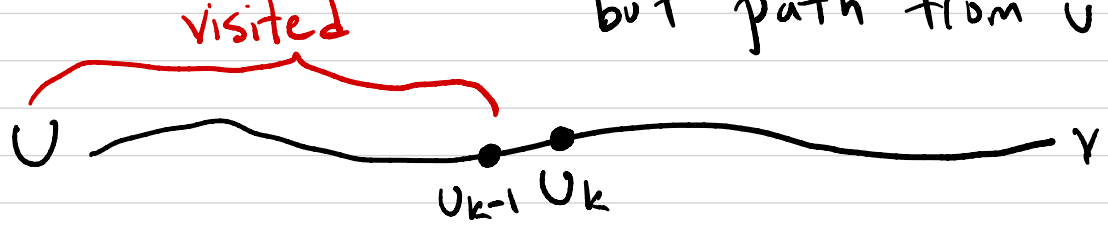
boolean array visited[n]
(init to all 0's)

Property: explore(G, u) visits exactly the vertices v s.t. G has path from u to v

Pf: 1. v explored $\Rightarrow$ path from u to v

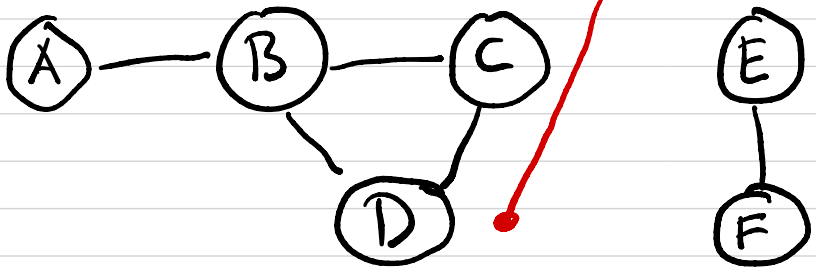
2. path from u to v $\Rightarrow$ explore v

Assume false, so v not explored but path from u to v



Should call explore(G, u_k). Contradiction!

Connectivity for undirected graphs (∃ path from u to v?)



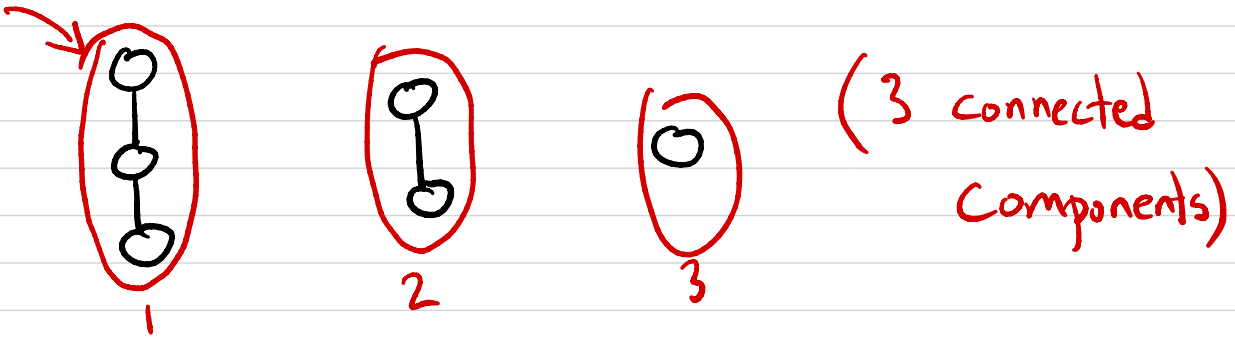

```
explore(G,u)
  visited[u] = true

  for v s.t. (u,v) ∈ E
      if visited[v] = false
          explore(G,v)
```

```
dfs(G)
  boolean array visited[n]
      (init to all 0's)

  for v ∈ V
      if visited[v] = false
          explore(G,v)
```

Computing G's connected components (undirected)



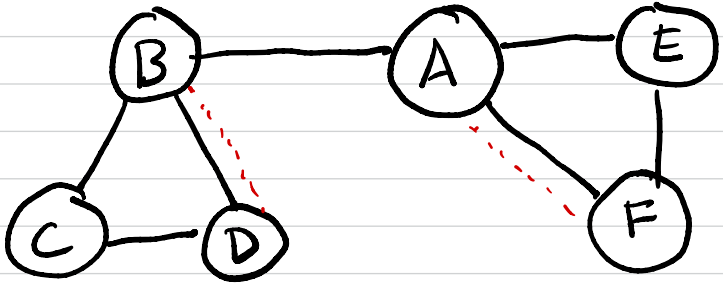
(3 connected components)

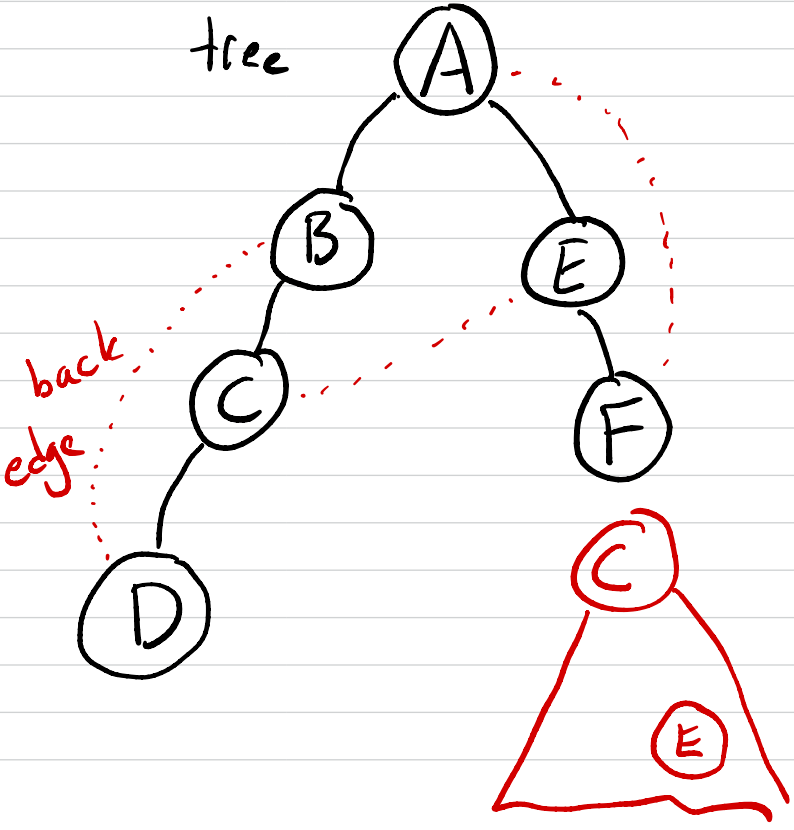
```
explore(G,v)
  visited[v] = true
  ccnum[v] = count

  for w s.t. (v,w) ∈ E
      if visited[w] = false
          explore(G,w)
```

```
dfs(G)
  boolean array visited[n]
      (init to all 0's)
  count = 1
  int array ccnum[n]
  for v ∈ V
      if visited[v] = false
          explore(G,v)
          count = count + 1
```

The DFS tree

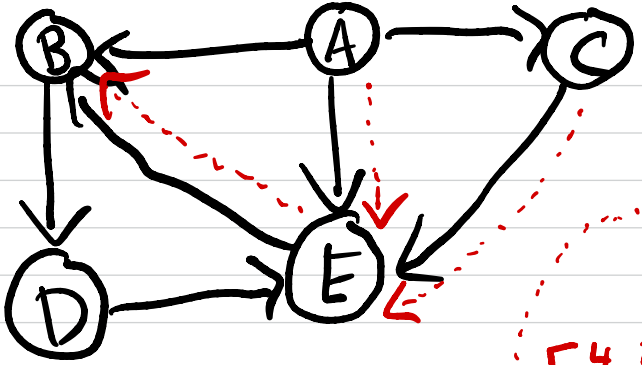


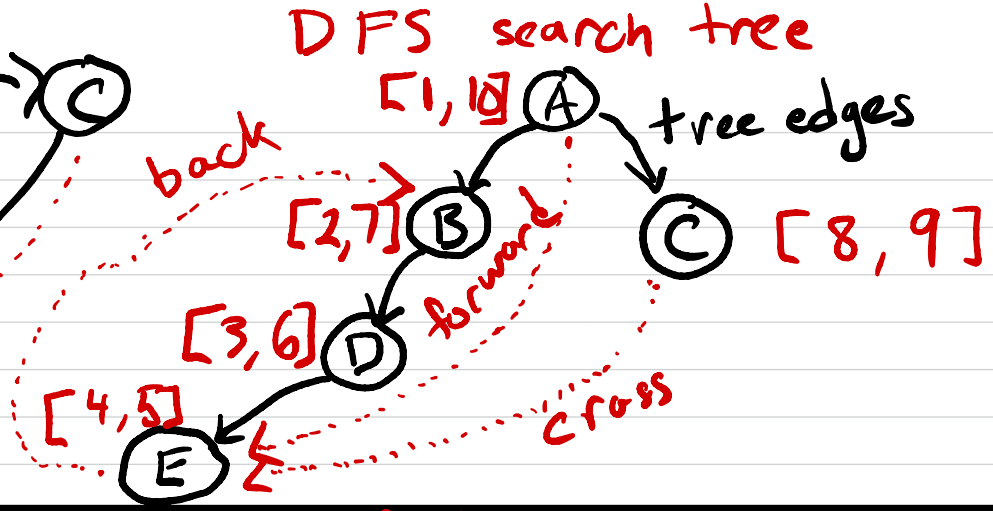


Runtime of DFS

Only call explore(G,v) exactly once, for each v

Runtime of explore(G,v):






```
explore(G,v)
  visited[v] = true

  pre[v] = clock
  clock = clock+1

  for w s.t. (v,w) ∈ E
      if visited[w] = false
          explore(G,w)
  post[v] = clock
  clock = clock+1
```

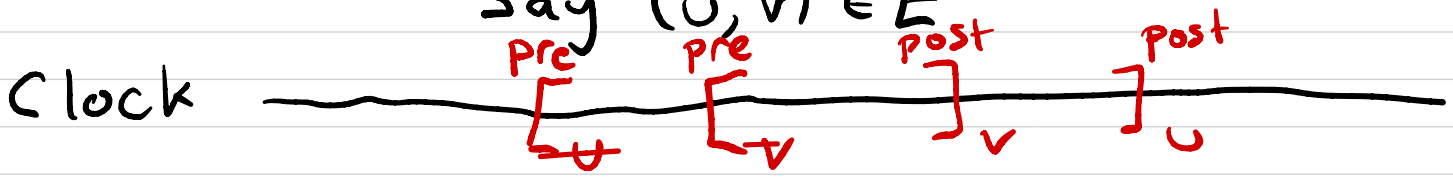
```
dfs(G)
  boolean array visited[n]
      (init to all 0's)
  clock = 1

  int array pre[n], post[n]

  for v ∈ V
      if visited[v] = false
          explore(G,v)
```

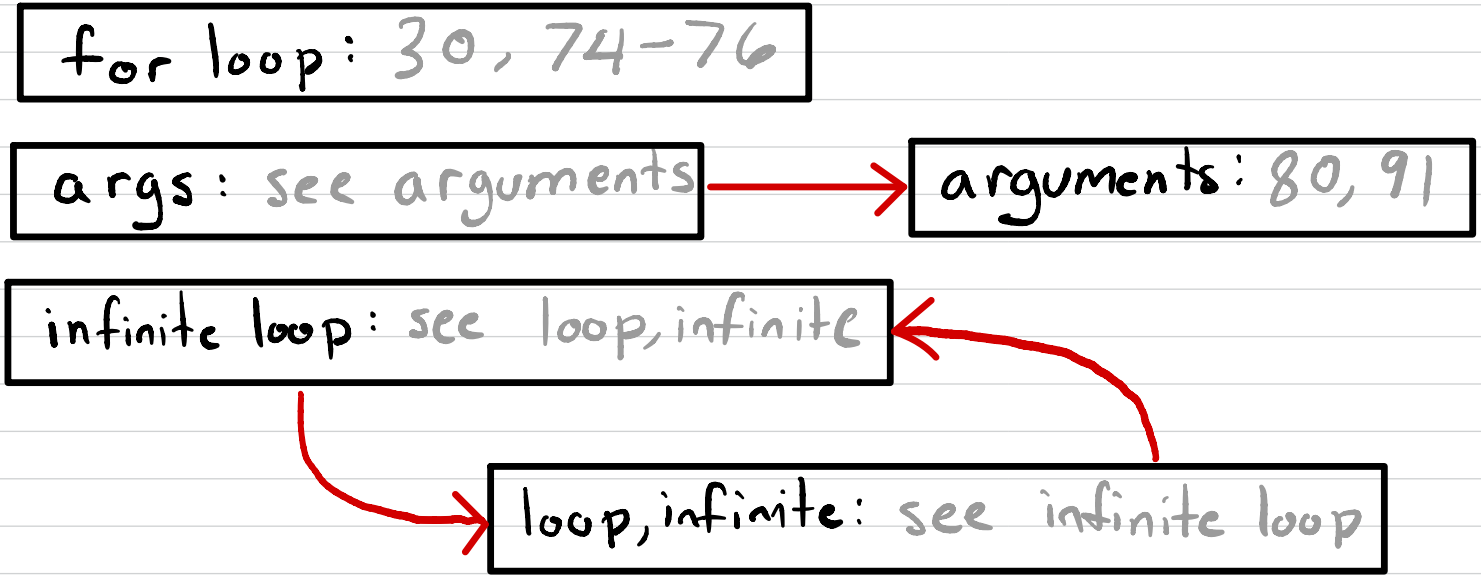
Classifying edges using pre & post #'s

Say $(u, v) \in E$



Application #1: Cycle detection

Book index:

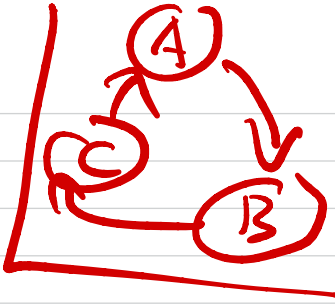


Q: Does my graph have a cycle?

Def: A directed acyclic graph (DAG) is a directed graph w/ no cycles.

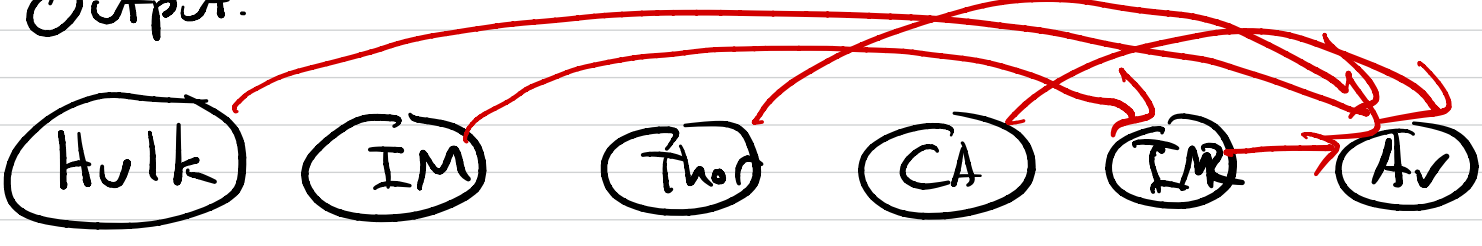
Claim: Suppose we run DFS on G. Then G is a DAG iff

Application #2: Topological Sort



dependency graph

Output:




Spider-Man: No Way Home
Jessica Jones S3
Spider-Man 3
The Amazing Spider-Man 2
Venom 2
The Punisher S2
Spider-Man 2
The Amazing Spider-Man
Venom
Daredevil S3
Spider-Man
Spider-Man: Far From Home
Iron Fist S2
Hawkeye S1
Luke Cage S2
Black Widow
Shang-Chi and the Legend of the Ten Rings
Eternals
The Falcon and the Winter Soldier S1
WandaVision S1
Loki S1
What If…? S1
Jessica Jones S2
Avengers: Endgame
The Punisher S1
Captain Marvel
Ant-Man and the Wasp
The Defenders
Avengers: Infinity War
Iron Fist S1
Thor: Ragnarok
Spider-Man: Homecoming
Black Panther
Guardians of the Galaxy Vol. 2
Luke Cage S1
Doctor Strange
Captain America: Civil War
Guardians of the Galaxy
Daredevil S2
Ant-Man
Jessica Jones S1
Avengers: Age of Ultron
Daredevil S1
Thor: The Dark World
Captain America: The Winter Soldier
Iron Man 3
Marvel's The Avengers
The Incredible Hulk
Iron Man 2
Thor
Captain America: The First Avenger
Iron Man
(Note: edges are reversed)

Topological sort

Input: DAG $G=(V,E)$

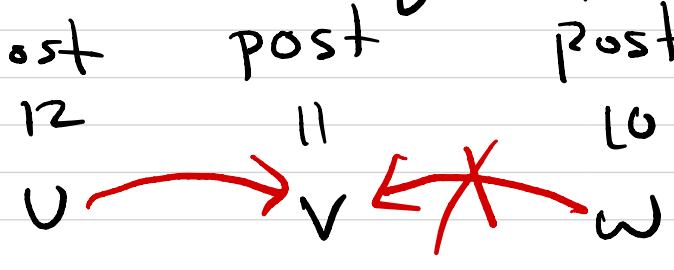
Output: Ordering of vertices $v_1, \ldots, v_n$ s.t. all edges go left $\rightarrow$ right

Claim: Suppose we DFS on G. Then for all $(u,v) \in E$, $post(u) > post(v)$.

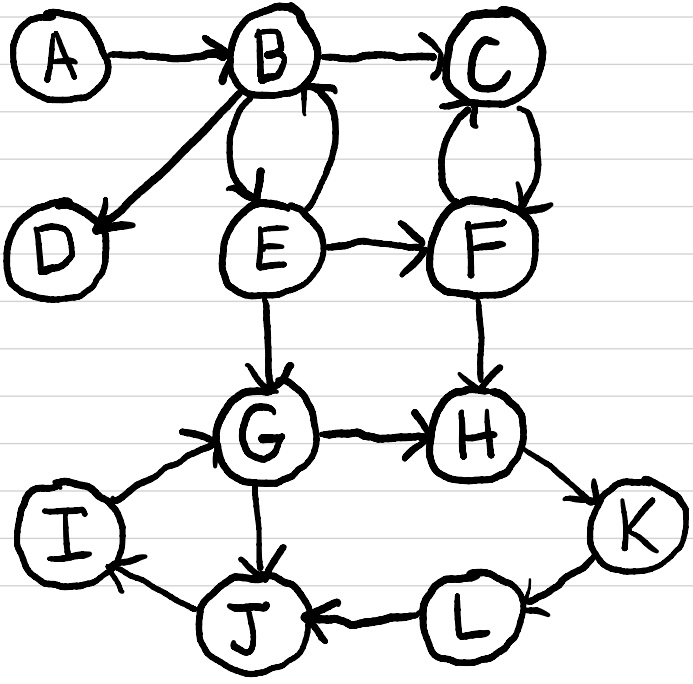
Pf: G is a DAG $\Rightarrow$ no back edges $\Rightarrow$ $post(u) > post(v)$ for all edges. □

Alg: Run DFS on G. Sort vertices from highest post to lowest.

Output:



Application #3: Connected components in digraphs


A
B
C
D
E
F
G
H
I
K
J
L