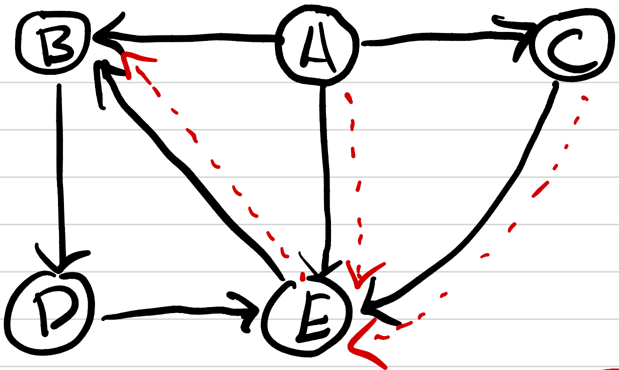
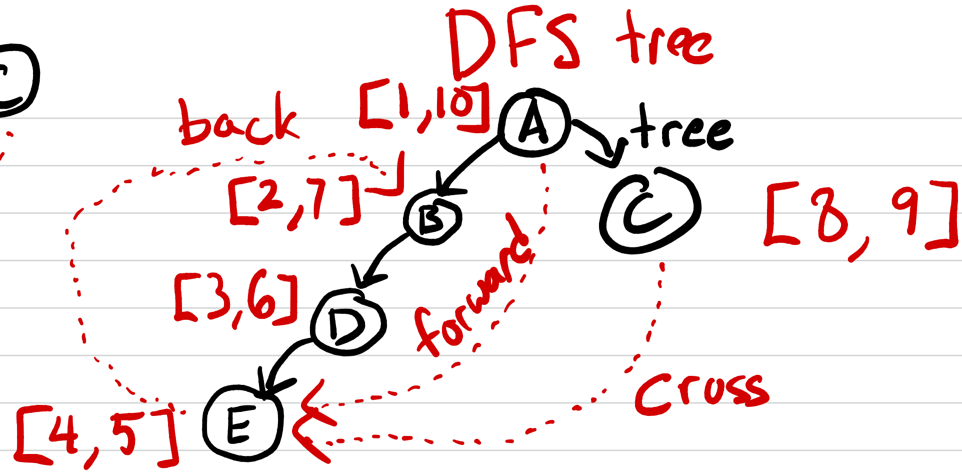
Directed graphs






```
explore(G, u)
  visited[u] = true
  pre[u] = clock
  clock = clock+1
  for v s.t. (u,v) ∈ E
    is visited[v] = false
        explore(G,v)
  post[u] = clock
  clock = clock+1
```

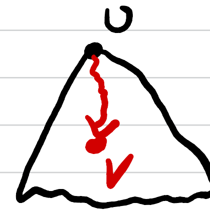
```
dfs(G)
  boolean array visited[n]
     (init to all 0)
  clock = 1
  int array pre[n], post[n]
  for all v ∈ V
    if visited[v] = false
      explore(G,v)
```

Classifying edges using pre & post #'s

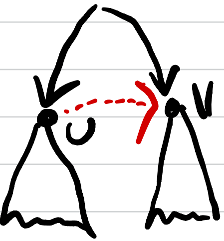
Say $(u,v) \in E$

Clock: $\overset{\text{pre}}{[_u} \quad \overset{\text{pre}}{[_v} \quad \overset{\text{post}}{]_v} \quad \overset{\text{post}}{]_u}$

$[_u\ [_v\ ]_v\ ]_u$: tree, forward

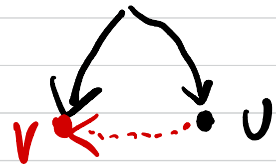


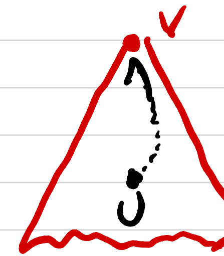
$[_u\ ]_u\ [_v\ ]_v$: impossible



$[\ ]_v\ [\ ]_u$: cross

$[_v\ [_u\ ]_u\ ]_v$: back



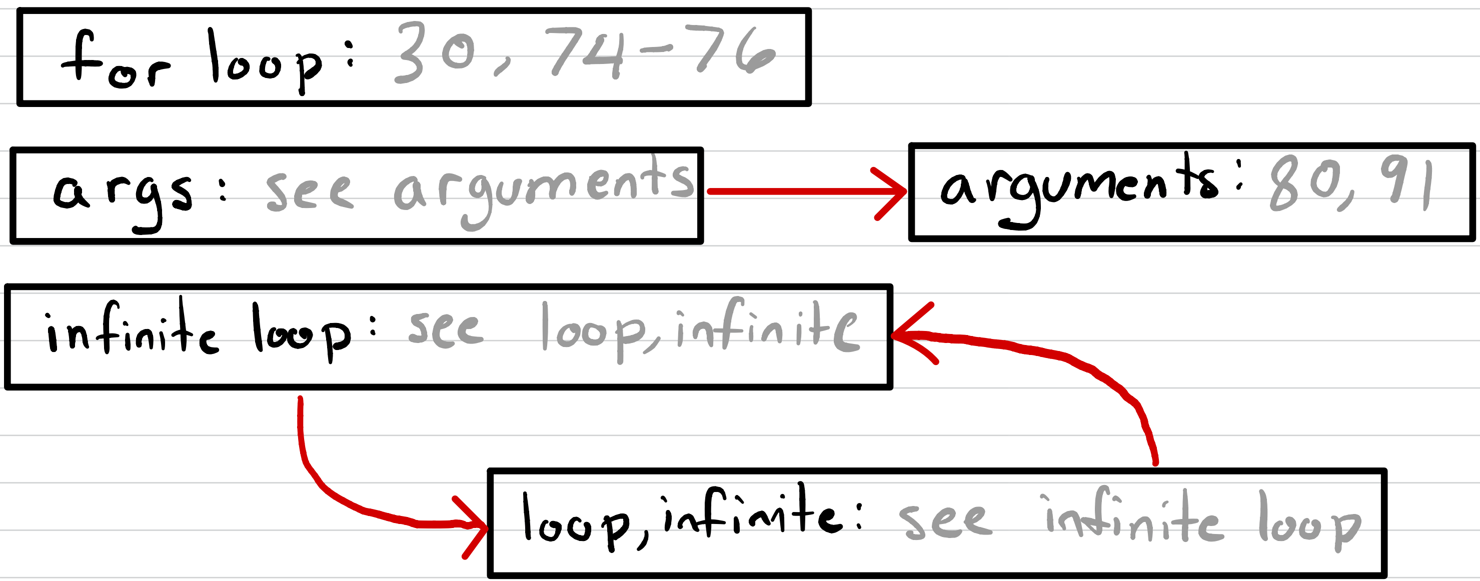


$[_u\ [_v\ ]_u\ ]_v$: impossible

Fact: (u,v) is back edge $\Leftrightarrow$ $post(u) < post(v)$

Application #1: Cycle detection

Book index:

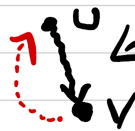


Q: Does my graph have a cycle?

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 no back edges.

Pf: (1) If back edge $\Rightarrow$ G is not a DAG.



cycle!

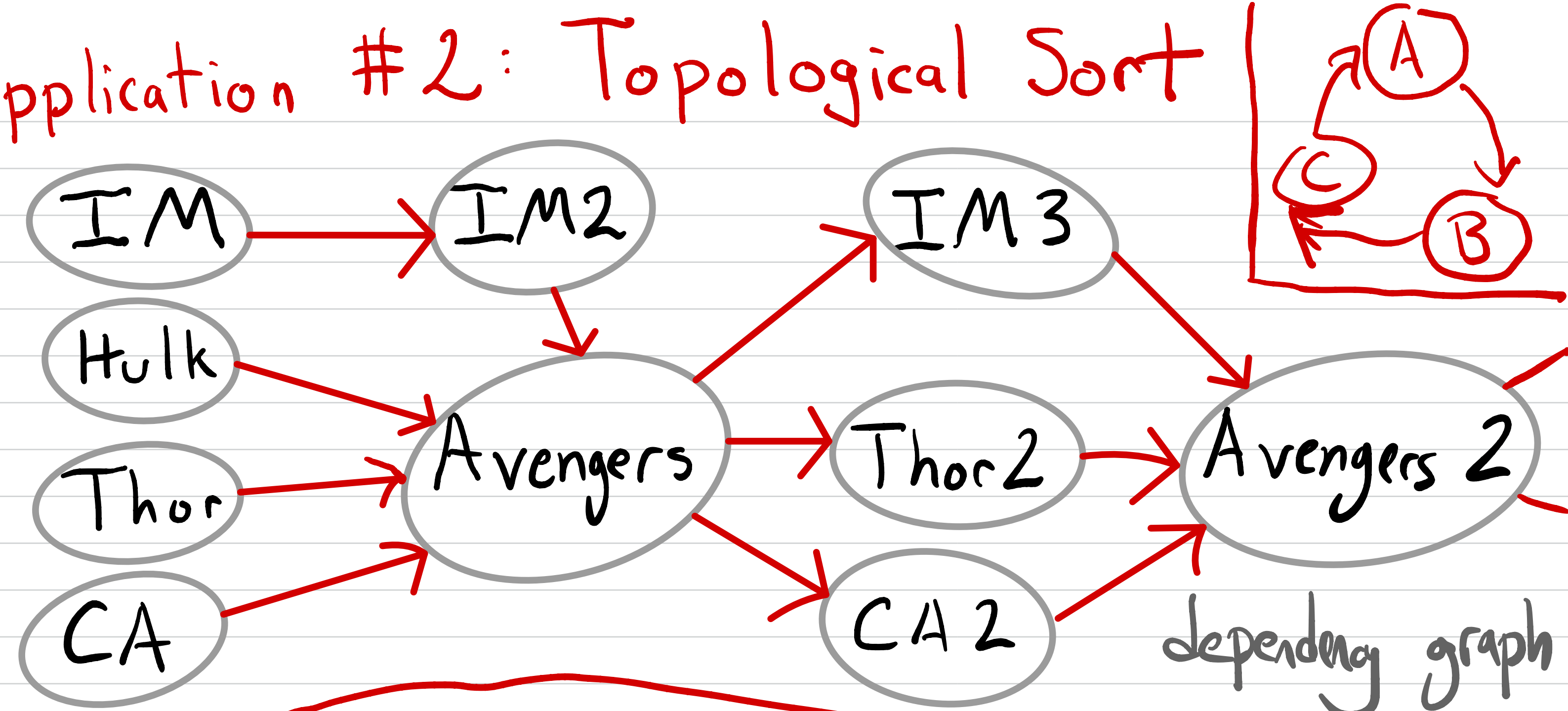
(2) G is not a DAG $\Rightarrow$ back edge

$$v_1 \to v_2 \to v_3 \to \cdots \to v_k$$

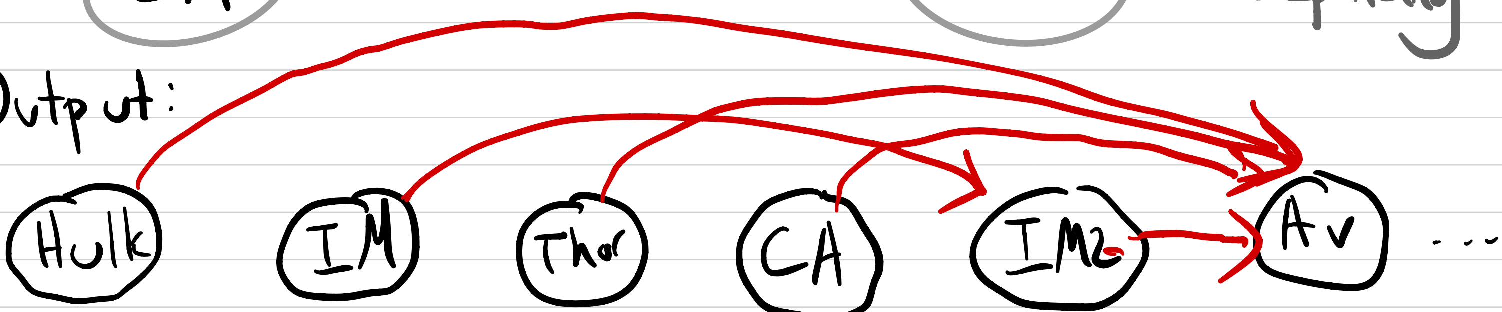
back edge!

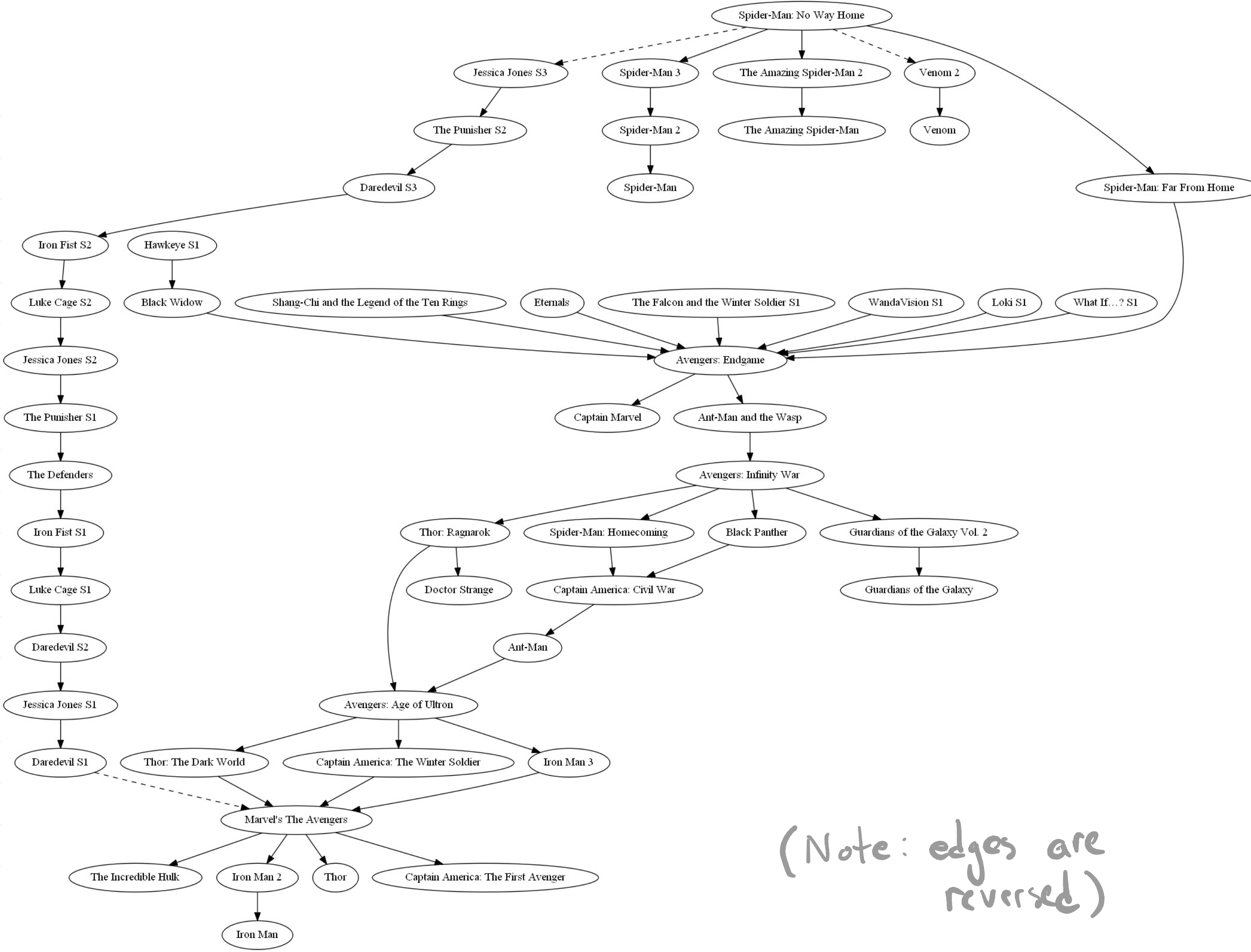
Cycle detection alg: Run DFS.
Output "DAG" if no back edges
($\forall (u,v) \in E$, check $post(u) > post(v)$)

Application #2: Topological Sort



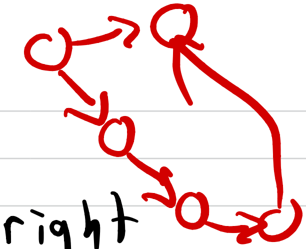
Output:





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 $v_1, \dots, v_n$
s.t. all edges go left $\longrightarrow$ right

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

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

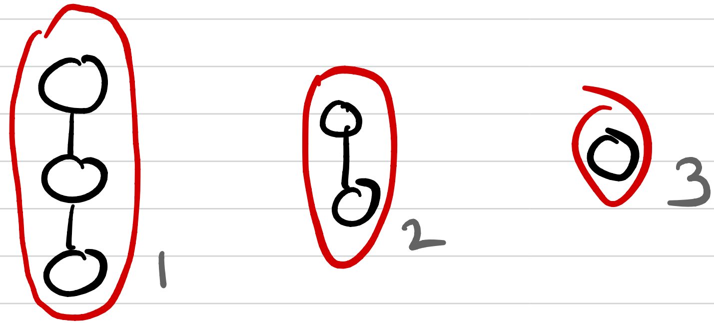
Alg: Run DFS on G.
Sort vertices from highest to lowest post #.

Output:

post 12	post 11	post 9	
u	v	w	...

Application #3: Connected components

Undirected:



Directed:

connected?

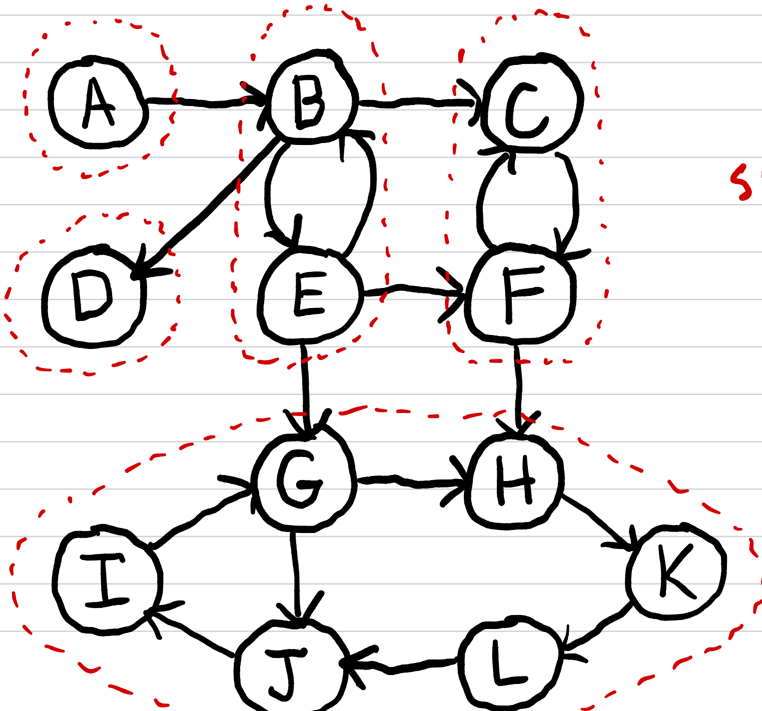
Questions:

1. How do we define connected components in digraphs?
2. How do we compute them?

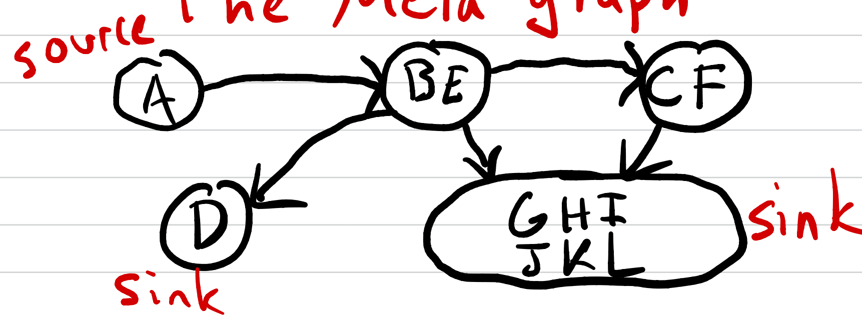
Strongly Connected Components (SCCs)

Def: Vertices u and v are **strongly connected** if there is a path from u to v and v to u

Claim: $u \sim v$ is an equivalence relation
(i) reflexive
(ii) symmetric
(iii) transitive



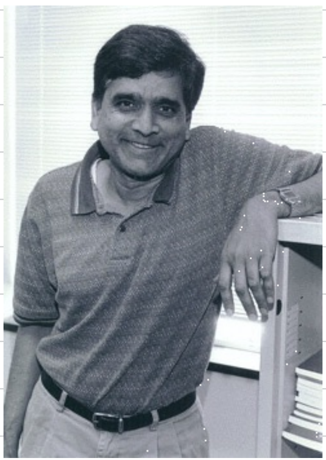
The Meta-graph



Claim: The metagraph is a DAG.

Today: Algorithm to compute SCCs

Kosaraju

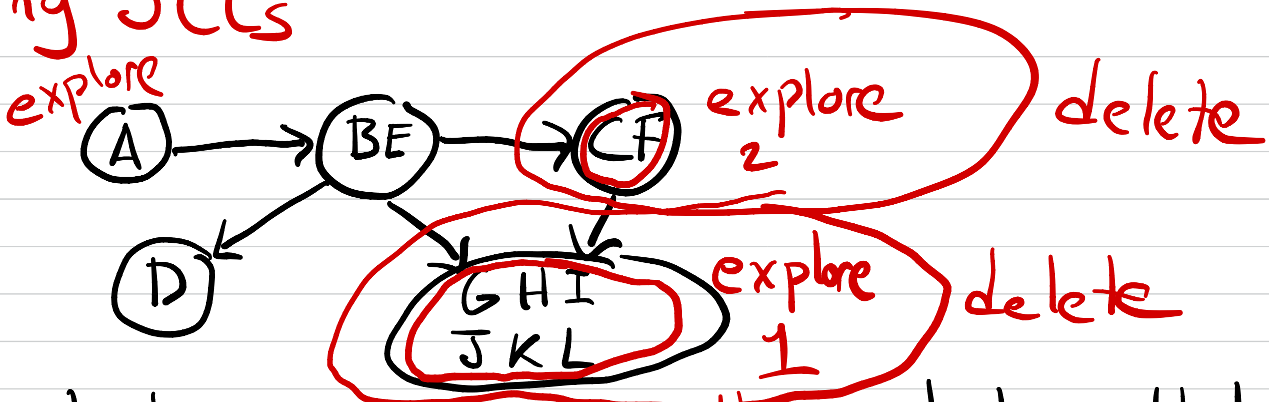


1978

Sharir

1981

Computing SCCs



Suppose we had a magic algorithm which could tell us a vertex u in a sink SCC

If we explore starting at u,
we explore all vertices in u's SCC.

Magic algorithm: DFS! (with a twist)