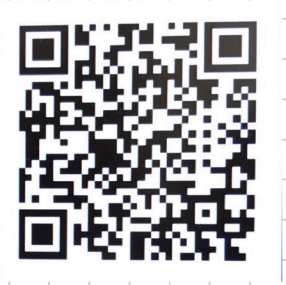
CS 170

Lecture 6

- use iClicker

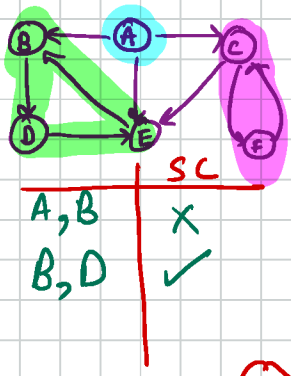
Sanjam Garg.

join.iclicker.com/RGWR

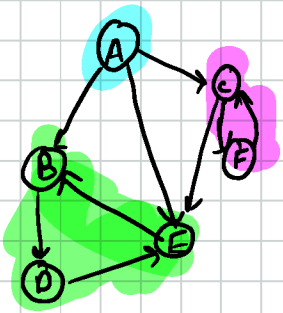


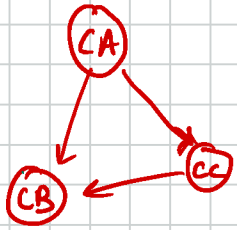
Today: Connectivity in directed graphs.

Refine: (SC) u is strongly-connected to v
if $\exists$ a path from u to v
& $\exists$ a path from v to u



Fact: Every Graph G can be decomposed as a DAG of strongly connected components (SCCs)



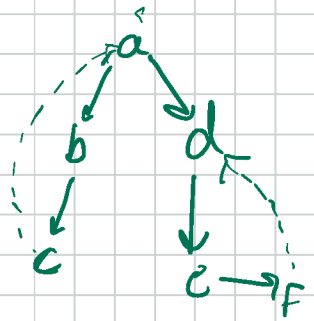


Decomposing Directed Graphs

Input : Directed Graph $G = (V, E)$

Output : Decompose G into DAG of SCCs.

smallest post #?



Intuition: ① Find a vertex v in the sink SCC

② Run explore(v)

③ Remove the visited vertices and recurse.

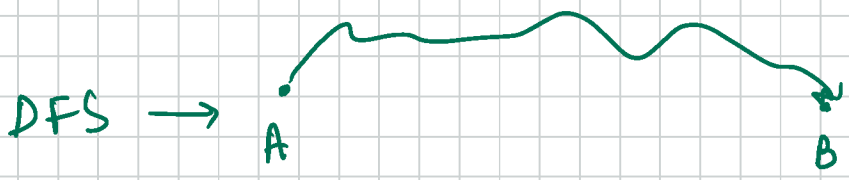
Recall: In a DFS traversal, vertex with the highest post number is in a source SCC ?

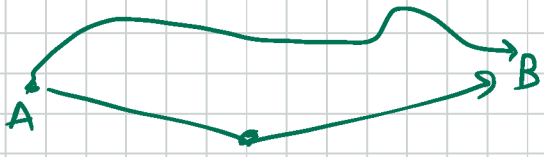
Idea : $G \longrightarrow G_R$

1) $G_R \leftarrow G$ with edges reversed $\longrightarrow$ linear
2) $post_R[v]$ be the post times in the execution DFS(G_R) $\rightarrow O(n+m)$
3) DFS on G (a second time) in decreasing $post_R[v]$ order. $\rightarrow O(n+m)$

```
a) visited = F   ∀ v∈V
b) count = 0     cc num[1...n] = ⊥
   if not visited[v]
       explore [v]
       count = (count+1)
```

```
Explore(v)
visited[v] = T
ccnum[v] = count
∀ (v, u) ∈ E
    if not visited[u]
        explore (u)
```



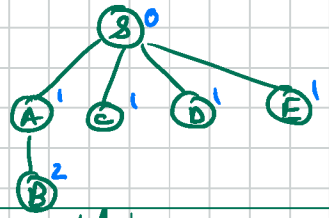


Breadth first search

Input: Graph $G=(V,E)$, directed or undirected
vertex $s \in V$

Output: $\forall u$ reachable from s, $dist[u]$ is set to the distance from s to u

```
∀ u ∈ V
    dist[u] = ∞
dist(s) = 0
Q = [s]  (queue initialized to s)
while Q is non-empty
    u = eject(Q)
    ∀ (u,v) ∈ E
        if dist(v) = ∞ then inject(Q,v)
                        dist[v] = dist[u] + 1
```



order of visitation	Queue
	[s]
s	[A, C, D, E]
A	[C, D, E, B]
C	[D, E, B]
D	[E, B]
E	[B]
B	[ ]

$$\sum_v 1 + deg(v)$$

$$= O(n+m)$$

Correctness

$\forall d = 0, 1, \ldots$ there exist a moment

(i) all nodes at distance $\leq d$ from s have their dist set correctly

(ii) all other nodes have dist set to ∞

(iii) the queue contains exactly the nodes at distance d.

$Q = [\ldots]$

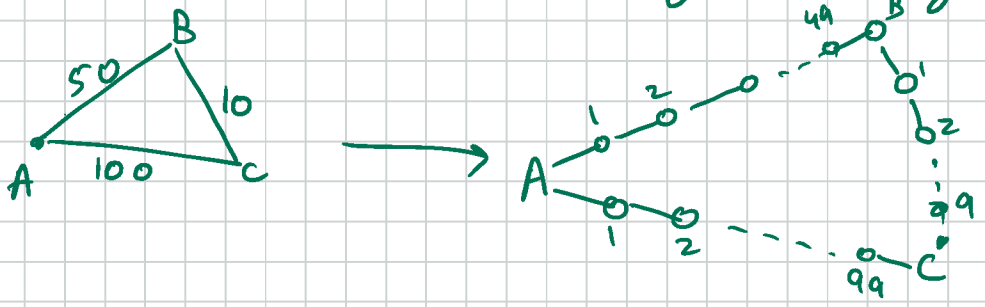
Base : $d = 0$

IS: $d \rightarrow d+1$

G with $\{l_e : e \in E\}$ $\quad l(e) \quad l(u,v)$

$G \longrightarrow G^{\text{extended}}$

→ each edge in G is replaced by l_e edges of weight 1 each.



Idea: Priority queue → elements are processed based on the priority
↳ (element, key)

① Insert: add a new element to the priority queue.

② Decrease-key: decreases the key of a particular element to a given value.

③ Delete-min: Return the element with the lowest key value & delete it

④ Make-queue: Builds a queue given a set $\{(e, key)\}$
↳ can potentially be faster than insertion one by one.