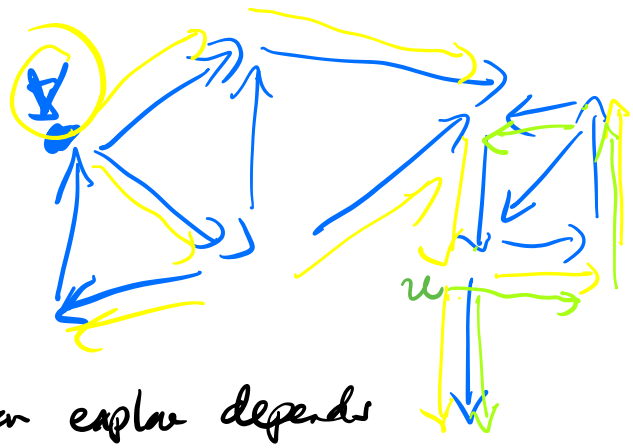
CS 3510

- Strongly Connected Components
  - BFS
    - Shortest paths

Directed paths

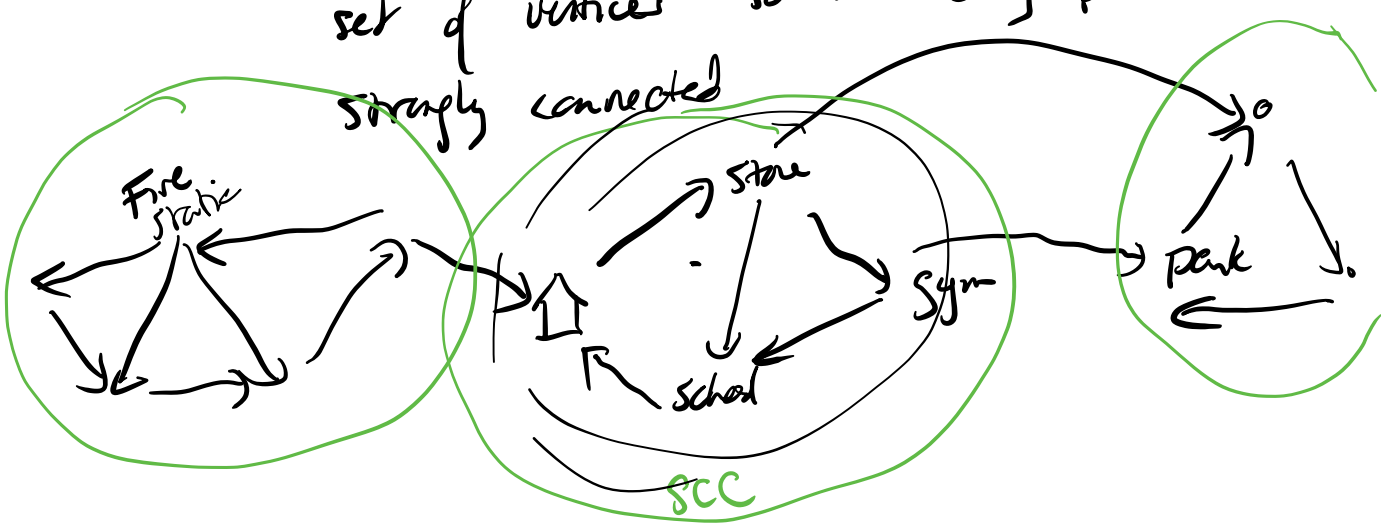
Start at v & do DFS



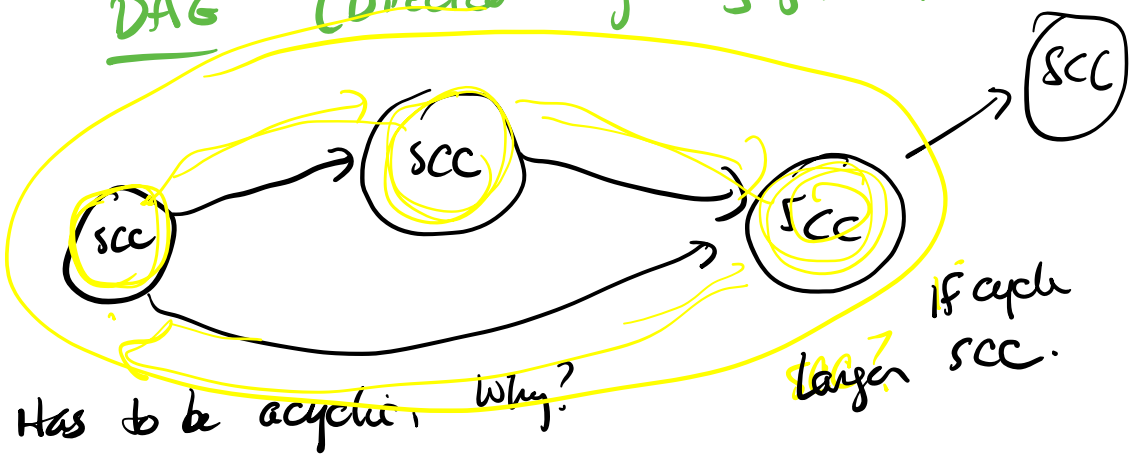
What is discovered from explore depends on where you start

Def: Vertices u & v are strongly connected if $\exists$ a path from u to v & from v to u.

Def: A strongly connected component is the set of vertices so that every pair is strongly connected

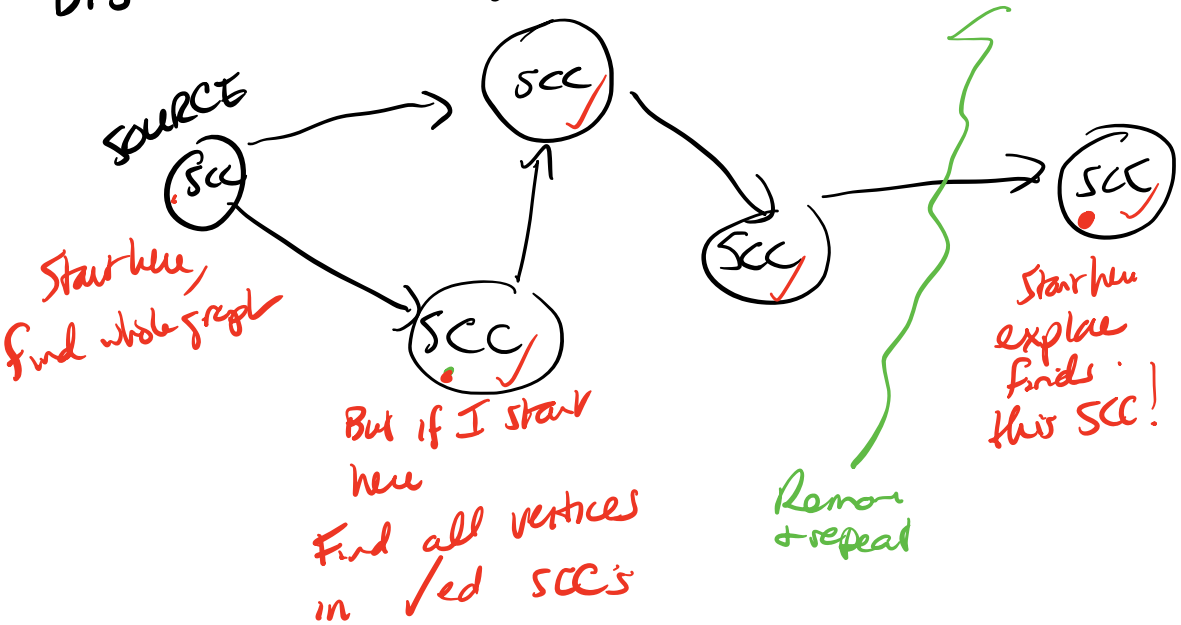


In general, any directed graph is a DAG (Directed acyclic graph) of SCC.



We can get this structure from DFS!

DFS: Start anywhere.

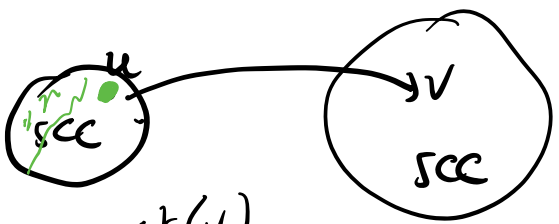


Property 1 : Explore from u stops when all vertices from u are reached

Property 2: The vertex with highest post # is in a source SCC

(Recall every DAG has $\geq$ source + $\geq$ sink)

Property 3': If



$post(u) > post(v)$

Pf:

Case 1: We visit u before v in DFS

If I see u, I will see v during same explore subroutine, regardless of where I started.

Then we finish exploring v's SCC before we postvisit u.

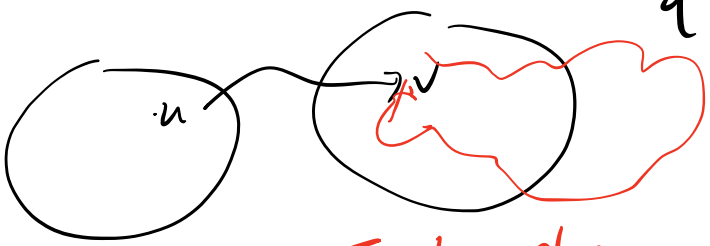
(Alt: u is on the stack + all of v's SCC is popped of stack before u is popped off)

Case 2: We visit v before u

If we visit v, we visit v's SCC

all SCCs reachable from v.

But this cannot include u's SCC
or we have a cycle (in metagraph of SCCs)



Finish explore
before exploring + reaching u

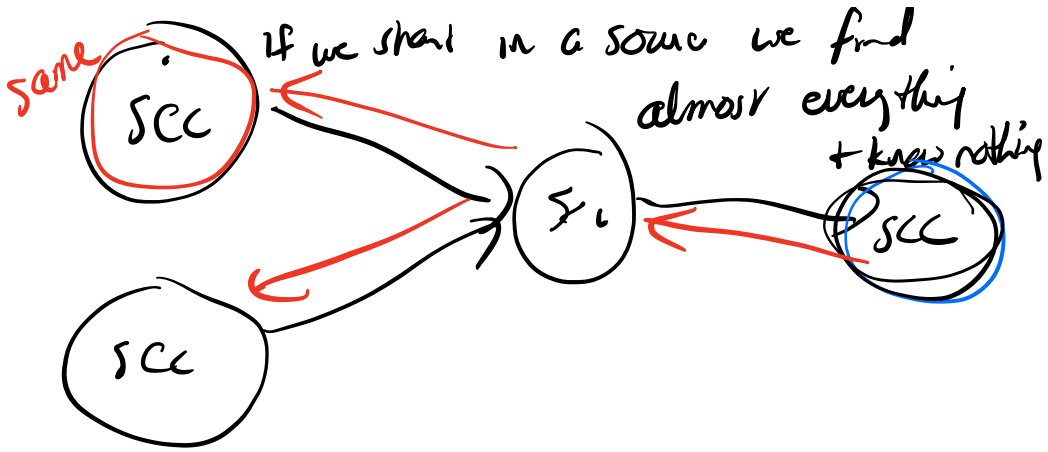
So v gets a post # before u
gets a previsit # + therefore
postV(u) > postV(v)

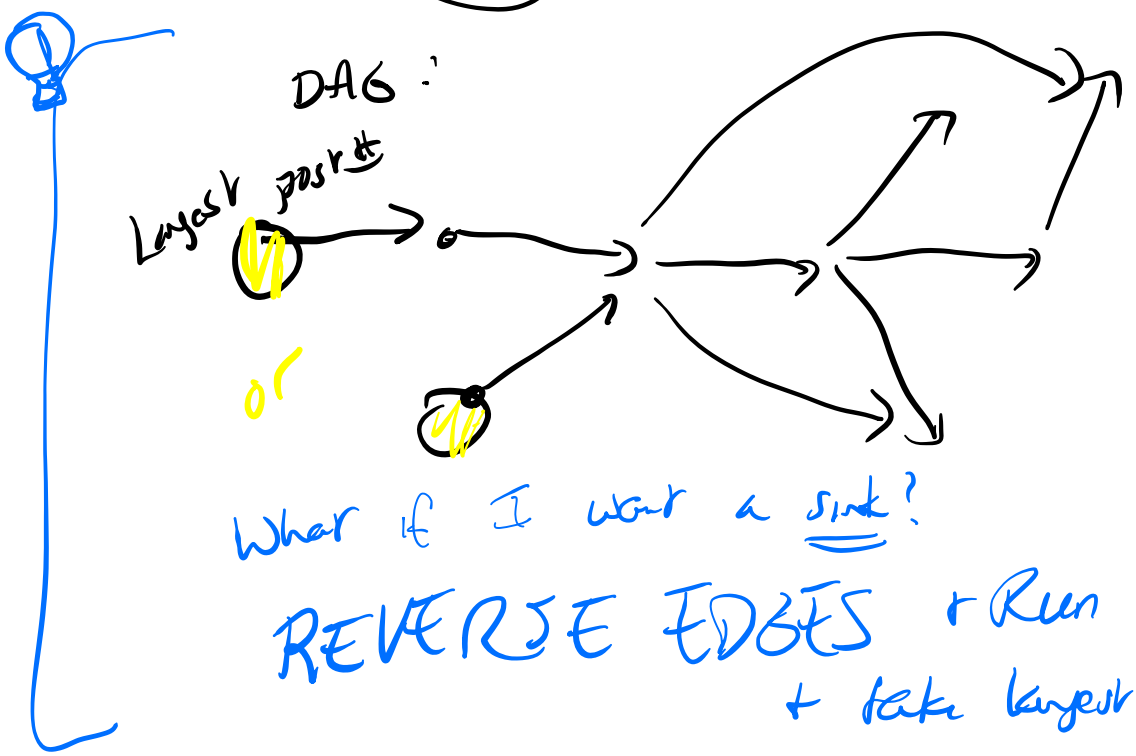
Obs

1. So, to find DAG of SCCs, we want
to start in a sink SCC
because we find that SCC + nothing
else + we can remove it + repeat

2. We know the vertex with highest postV#
is in a source SCC.

??





What if I want a sink?

REVERSE EDGES + Run DFS
+ take largest post#

1. Reverse edges to get G^R
2. Run DFS on G^R
3. Vertex with largest post# is in a source SCC of G^R
   + this is a sink SCC of G.

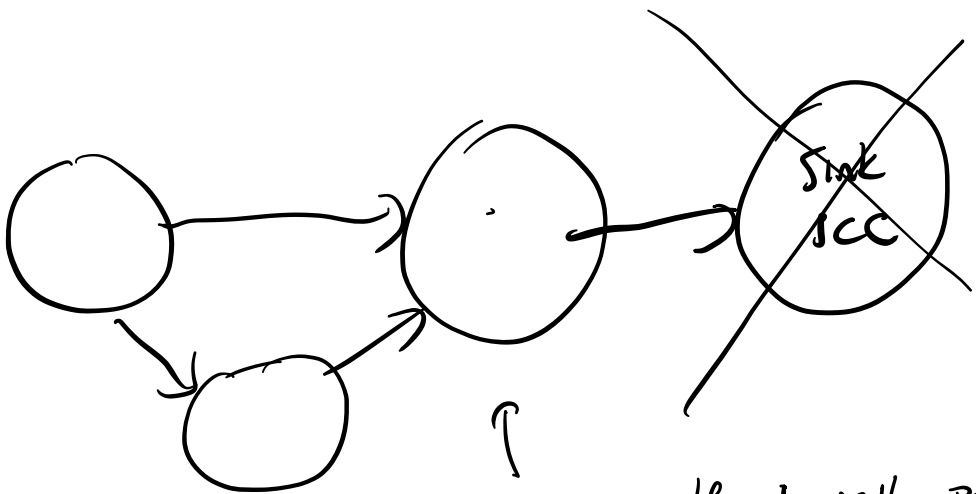
Alg:

- Run DFS on G^R ← Just to get post #s
- Explore from vertex in G with largest post #

That is a sink SCC.

Remove it.

Repeat.



vertex with largest post # in remaining vertices (from original DFS of G^R)

is now in a sink SCC of what's remaining!

Total running time?

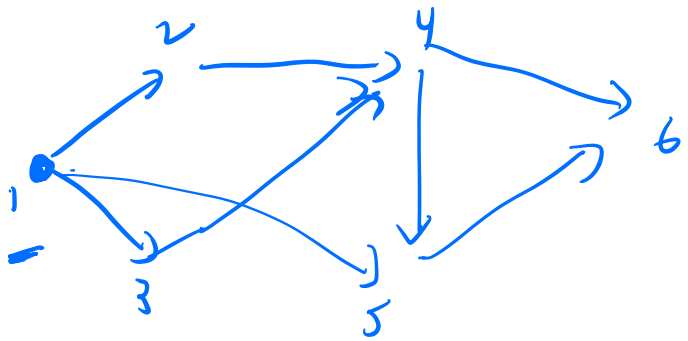
DFS on G^R $O(n+m)$

DFS on G $O(n+m)$

Very fast!

$O(n+m)$

BFS



Explore from 1. (root)
Then explore from everyth reachable from 1 (root)

We use a queue Q

Queue evolves

1

2 3 5

3 5 4

5 4

4 6

6

procedure bfs (G, s)

Input is $G = (V, E)$ (directed or undir)

Vertex $s \in V$

Output all vertices u reachable from s,

dist(u) is a set of distances

for all $u \in V$, dist$(u) = \infty$ initialize

dist$(s) = 0$

$Q = \{s\}$

while Q is not empty:

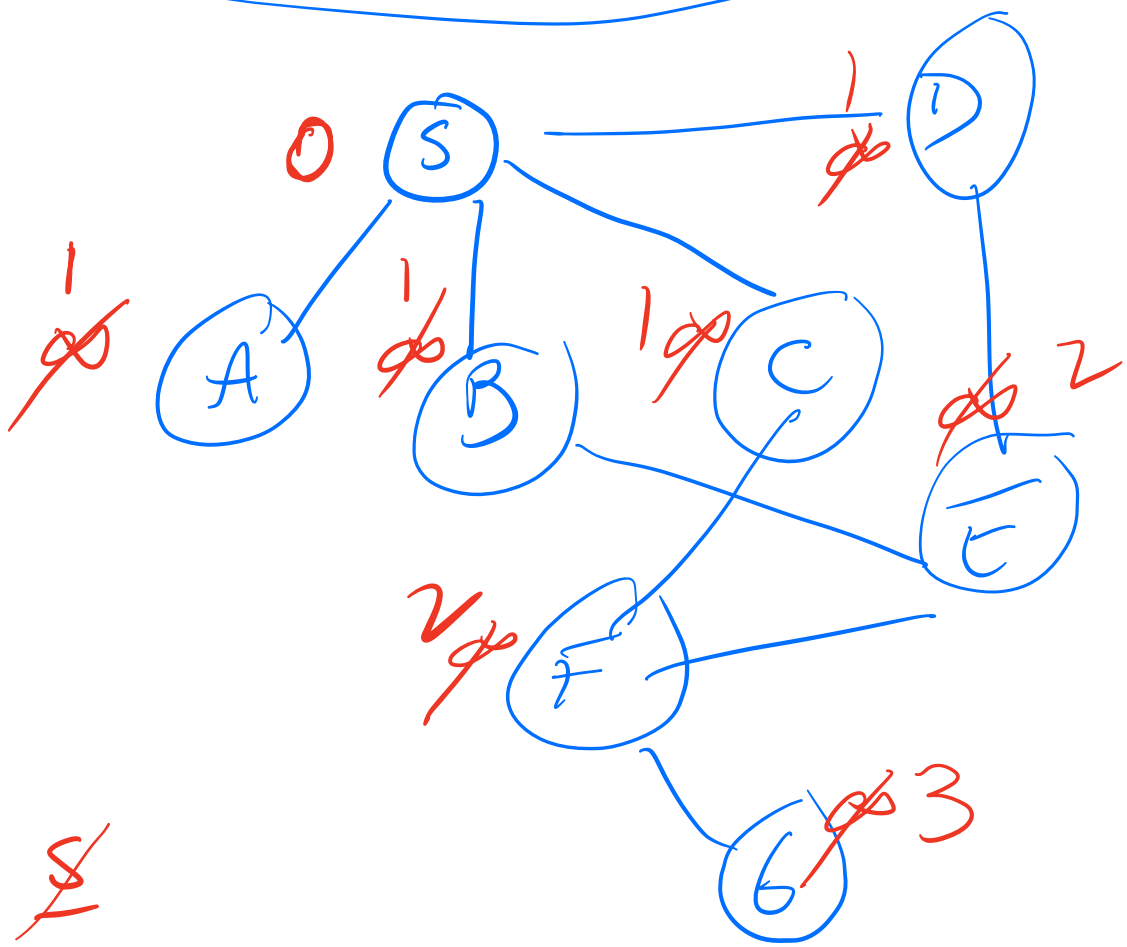
$u =$ eject(Q)

for all edges $(u, v) \in E$

if dist$(v) = \infty$

inject(Q, v)

dist(v) = dist$(u) + 1$



S

A

A B

A B C

A B C D

B C D

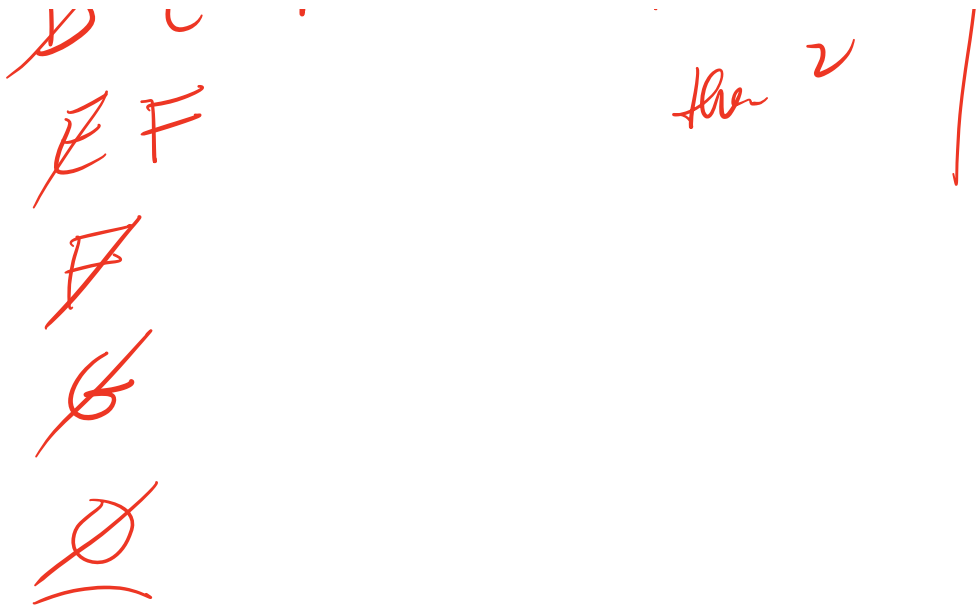
D E

F F

S

1

the B



BFS gives shortest distances from s.

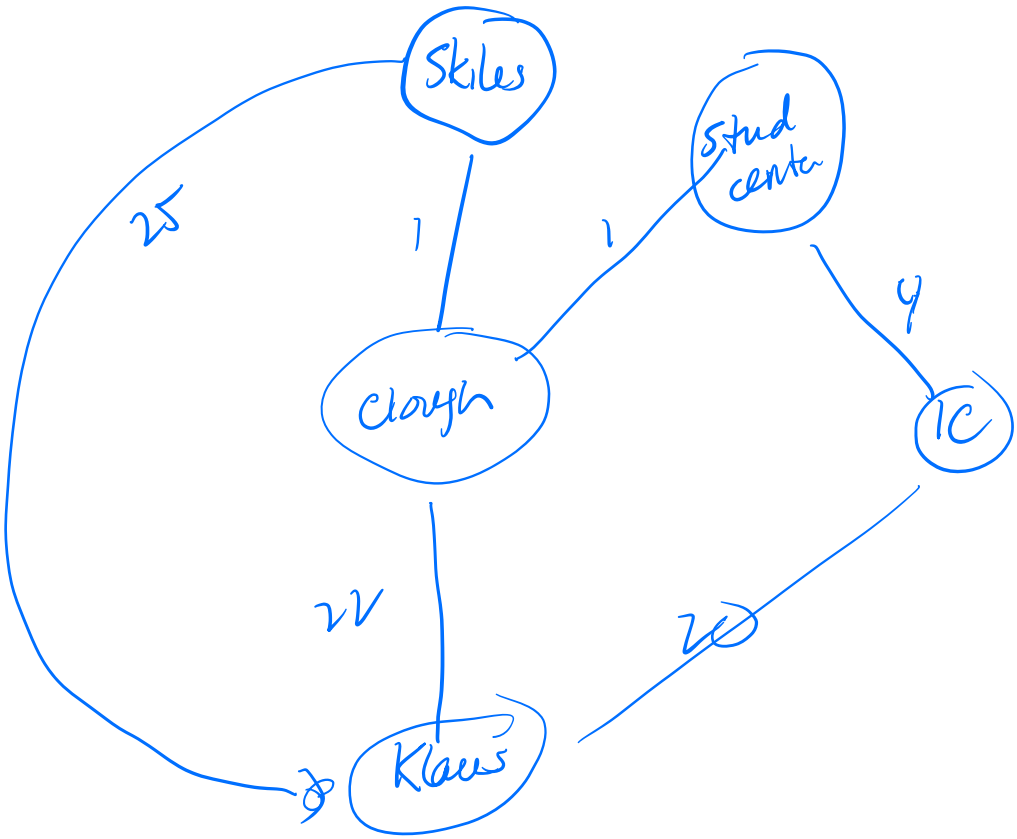
Fewest hops

If dist(u) = 4 ⇒

there is a path with 4 hops

& there is not a shorter one.

Weighted graphs



What is the best way to get from X to Y?

w(u,v) is the weight of edge

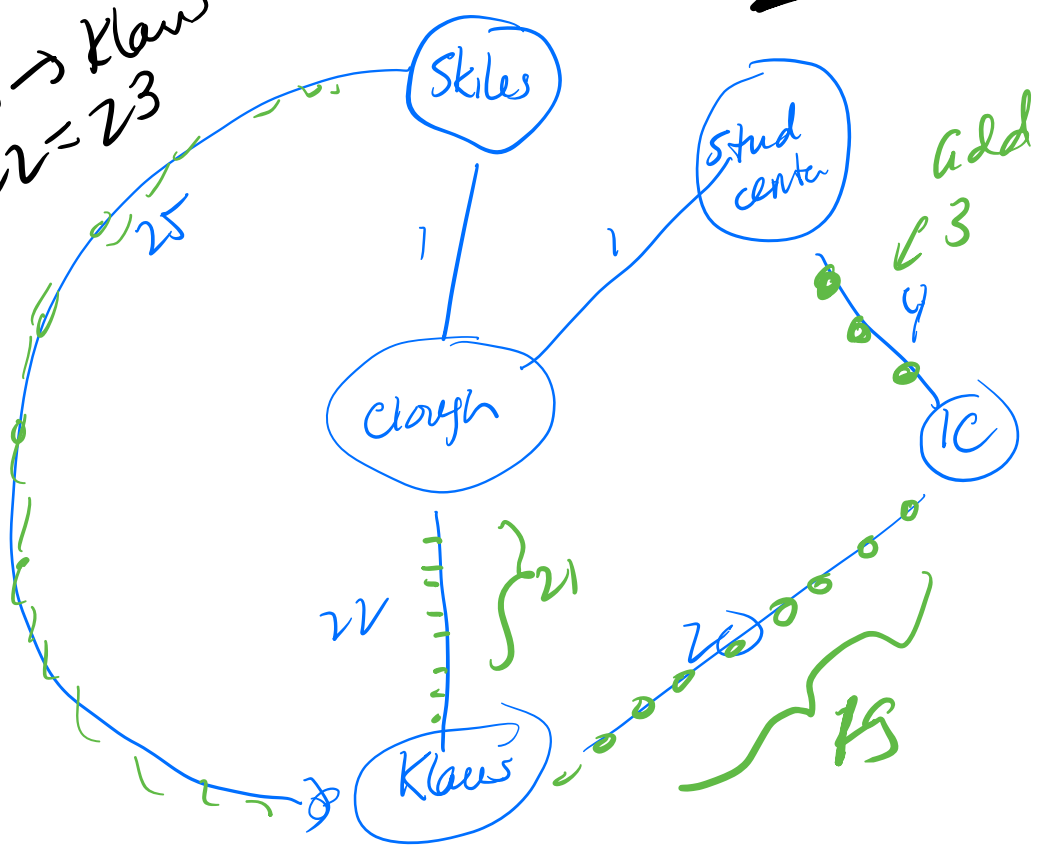
between u + v

Assume $w(u,v) \geq 0$, $\forall u, v$

(nonnegative weights)

Dist between u + v is min cost path (Σ wts on edges)

Skiles → Klaw
1+22=23



Can I use BFS to find shortest distances (from Skiles to all else)?

Add dummy vertices

BFS on this new graph gives shortest distances!

Instead: Find next interesting event (not dummy vtxs)