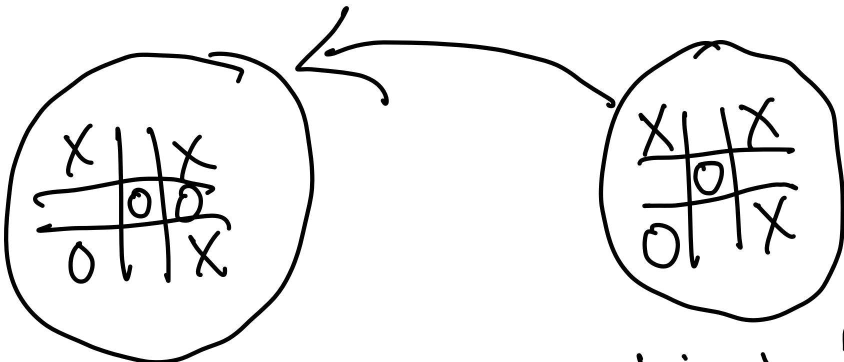
CS 331, Fall 2025
Lecture 7 (9/17)

Today:
- Graphs
- DAGs
- APSP

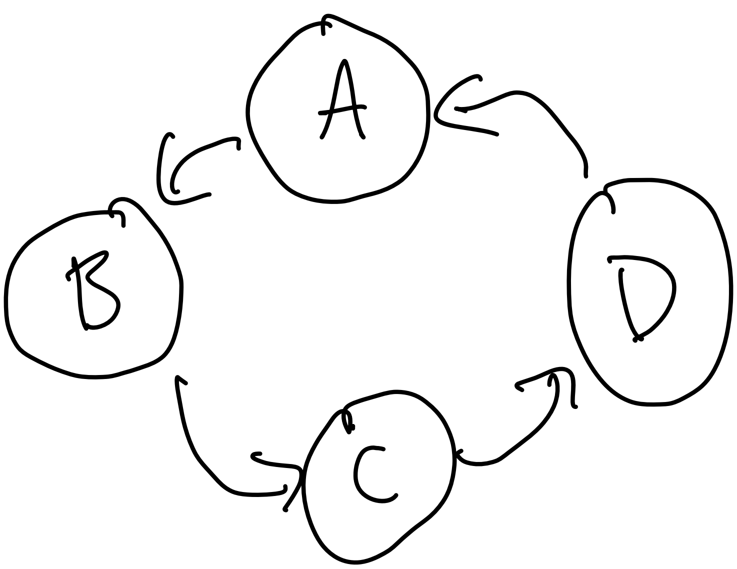
Motivation from last time:

depends on all possible moves

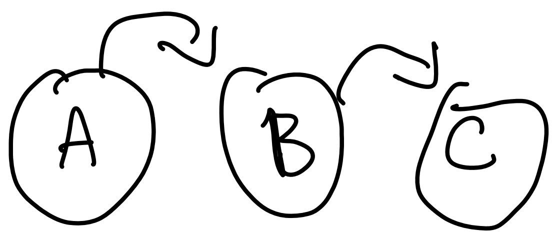


Winning?

What can get us in trouble?



e.g. chess



e.g. tic-tac-toe

key diff: <u>progress</u>

Background: Graphs (Part I, Section 4)

$G = (V, E, w)$

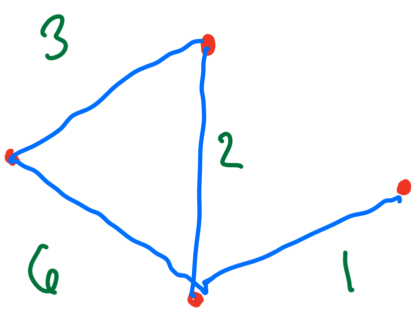
vertices edges weights

$n = |V|$

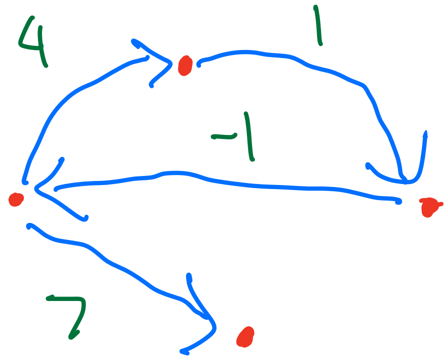
$m = |E|$

Typically identify $V \equiv [n]$

Vertex names: $1, 2, \ldots, n$



undirected



directed

Edges: $e \in E$ $e \in E$

$(i,j) \equiv (j,i)$

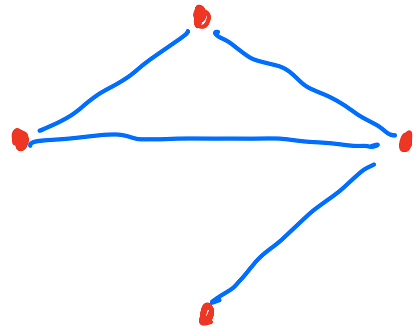
"tail" "head"

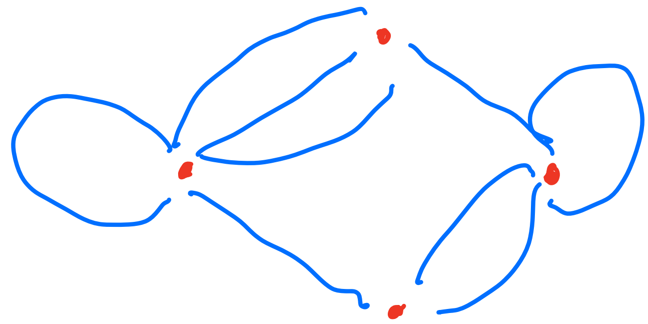
i  j

i  j

Weights: $w_e \in \mathbb{R}$ $w_e \in \mathbb{R}$

Caveat: simple vs. multigraph

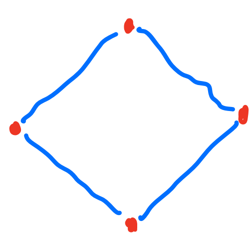
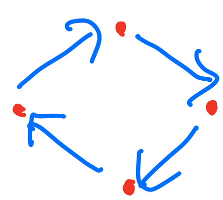
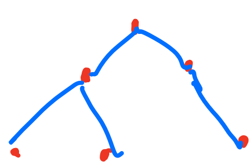




All graphs assumed <u>simple</u>. no multi-edges / self loops

Hence, $m \le n(n-1) = O(n^2)$

Special graphs

	undirected	directed
Paths		
cycles		
trees	 (no cycles)	stay tuned...

Representing a graph

In this class: <u>adjacency list</u> model $(L_V^{in}, L_V^{out}, L_E)$

L_V^{out}: vertices know edges

L_E: edges know vertices

vertices

1	(1,2)	
2	(2,4)	(2,5)
3	(3,5)	
4	(4,2)	
5		

(pointers to L_E)

edges

	out	in	weights
(1,2)	1	2	$w_{(1,2)}$
(2,4)	2	4	$w_{(2,4)}$
(2,5)	2	5	$w_{(2,5)}$
(3,5)	3	5	$w_{(3,5)}$
(4,2)	4	2	$w_{(4,2)}$

pointers to L_V^{in} L_V^{out}

Space: $O(n + m)$

Primitives:

- "pass over all edges"

  runtime: $O(m)$

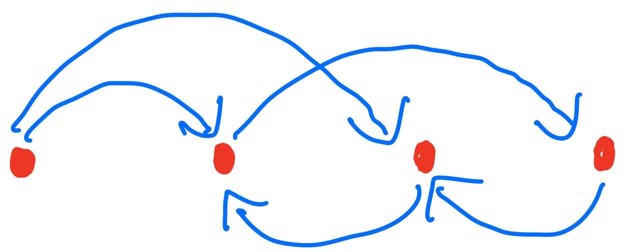
- "pass over neighbors of v"

  runtime: $O(|\text{neighbors of } v|)$

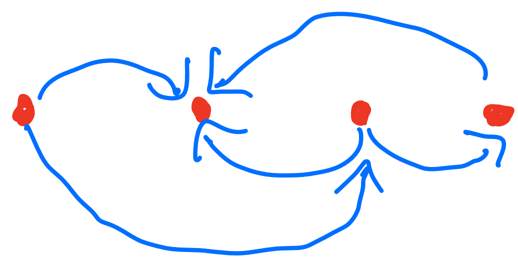
Directed acyclic graphs (Part III, Section 5.2)

Definition: • Directed graph
(duh) • Acyclic

Example



Not a DAG



DAG

It's not too obvious...

Can we make it easier to tell?

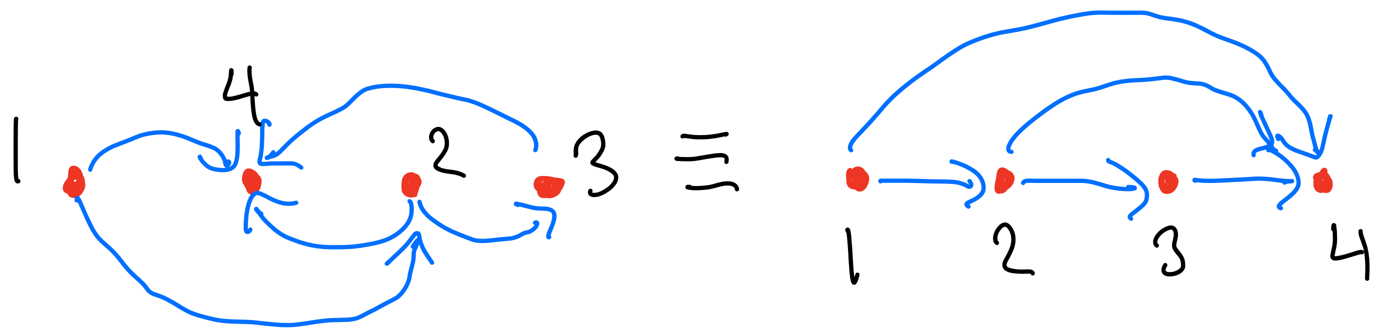
Yes! topological ordering

Topological ordering

Rename vertices $1, 2, \ldots, n$

All edges $e = (i, j)$, $i < j$

Claim: $\exists$ top order $\iff$ DAG



Proof ($\Rightarrow$): Suppose top order + cycle

$(i_1, i_2), (i_2, i_3) \ldots (i_{k-1}, i_k), (i_k, i_1)$

$<$ $<$ $<$ Contradiction!

Proof ($\Leftarrow$): We'll see in Part V

runtime: $O(n + m)$

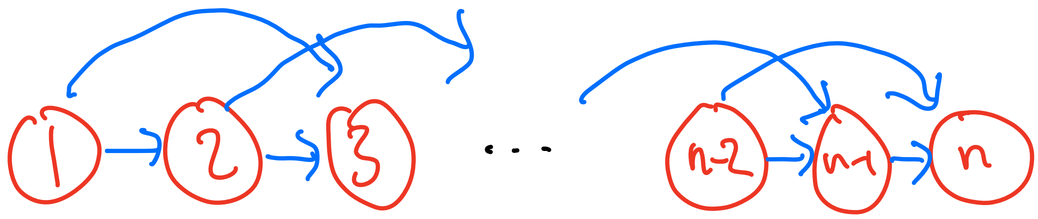
Consider DP algo.

Dependency graph must be DAG

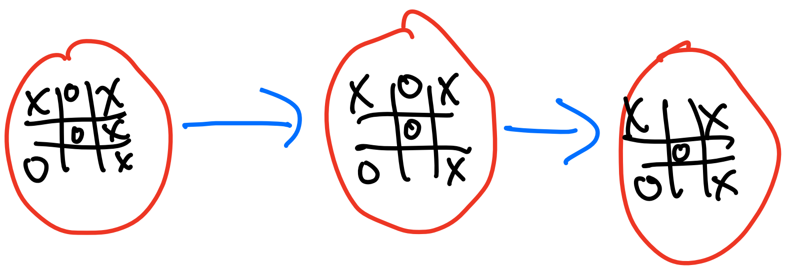
- vertices = subproblems
- edges: $u \to v$ iff $S[v] = \ldots(S[u])$

$\boxed{\text{Example}}$

Fibonacci:



Game theory (reverse of game graph)



terminal

Claim: evaluate in top order!

Proof: strong induction.

SSSP on DAGs

SSSP = single source shortest paths

Input: DAG (V, E, w)

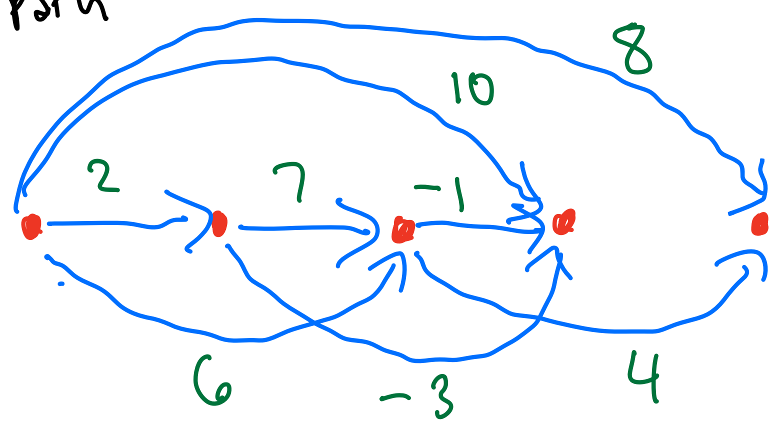
V in top order

Output: Shortest $(1, j)$ path $\forall j \in [n]$

$$\min_{\substack{P \subseteq E \\ P \text{ is } 1 \to j \text{ path}}} \sum_{e \in P} w_e$$

(think of as lengths)

Example



$S[j]$ = shortest $1 \to j$ path

$S[1] = 0$ $S[2] = 2$ $S[3] = 6$

$S[4] = -1$ $S[5] = 8$

DP recursion:

$$S[j] = \min_{(i,j)\in E} S[i] + w(i,j)$$

Evaluate in top order.

Runtime: $O(m+n)$ using L_V^{in}

All-Pairs Shortest Paths (Part III, Section 5.3)

The final boss. Our tools:

- Prefix-based DP
- Multidimensional DP
- DAGs
- Divide-and-Conquer

Let's begin.

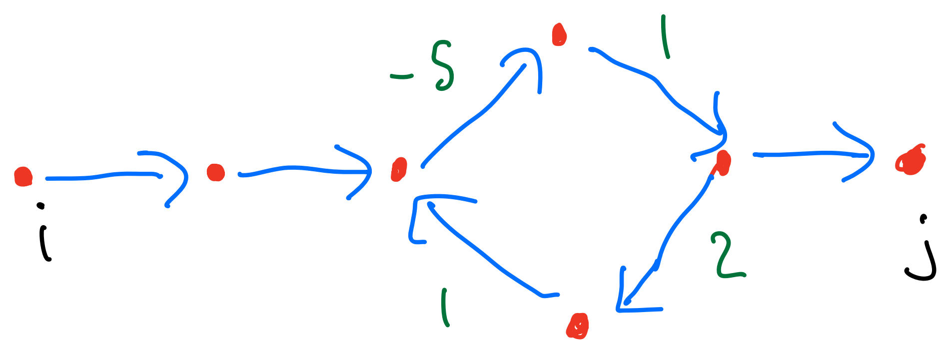
Input: $G = (V, E, w)$

Warning: maybe not DAG.

Output: $\forall (i,j) \in V \times V$, $i \neq j$

$$\min_{\substack{P \subseteq E \\ P \text{ is } i \to j \text{ path}}} \sum_{e \in P} w_e \qquad \begin{pmatrix} \text{unreachable} \\ = \infty \end{pmatrix}$$

Assumption: no negative-weight cycles



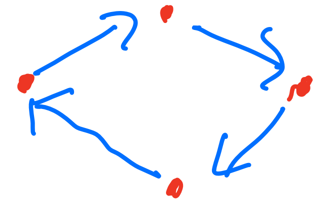
(revisit in Part V)

Baseline: $O(m+n) \times O(n)$ (DAGs)

$= O(n^3)$

we'll get this for all graphs.

Issue: progress notion?



Idea 1: <u>length of path</u>

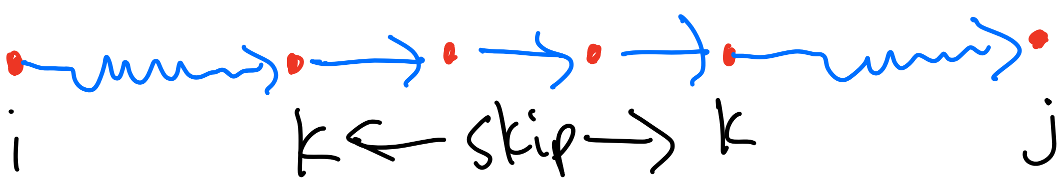
Try: $S[i][j][\ell]$

$=$ shortest $i \to j$ path w/ $\leq \ell$ edges

Lemma: $\exists$ shortest $i \to j$ path w/ $\leq n-1$ edges

Proof: Suppose k appears twice.

Shortcut! Remove all cycles



At end, $\leq n$ vertices

$\Rightarrow \leq n-1$ edges

Punchline:

$S[i][j](n-1)$

= shortest $i \to j$ path w/ $\leq n-1$ edges

= shortest $i \to j$ path!

DP recursion: guess the last edge.

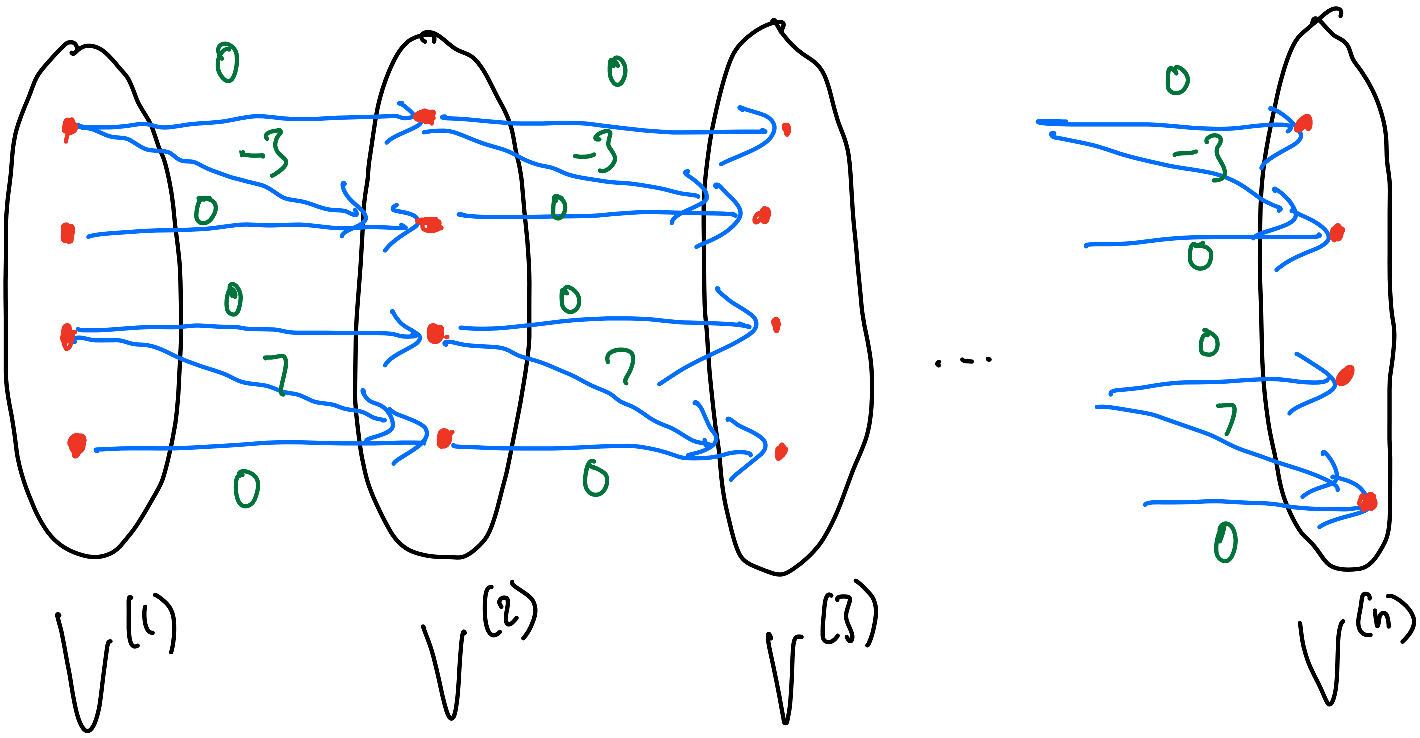
$$S[i][j](\ell) = \min_{(k,j) \in E} S[i][k](\ell-1) + w(k,j)$$

Base cases: $S[i][j](1) = \begin{cases} w(i,j) & (i,j) \in E \\ \infty & \text{else} \end{cases}$

Order: one "slice" ℓ at a time.

Runtime: $O(n^3) \times O(n) = O(n^4)$

Intuition: "layered graph"



n copies of each vertex $i^{(1)}, i^{(2)}, \ldots, i^{(n)}$

0-weight "shortcut edges" between $i^{(\ell)} \to i^{(\ell+1)}$

Copies of E between $\underbrace{V^{(\ell)}}_{\text{tails}} \times \underbrace{V^{(\ell+1)}}_{\text{heads}}$

Claim 1: DAG. (Proof: layer = progress)

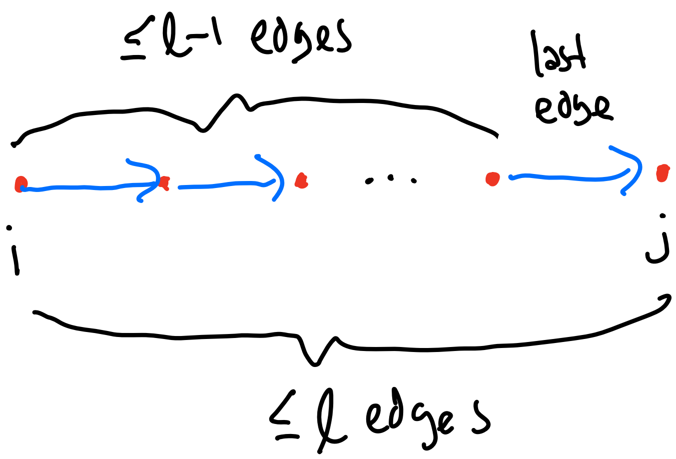
Claim 2: $i \to j$ shortest path

$\equiv i^{(1)} \to j^{(n)}$ shortest path

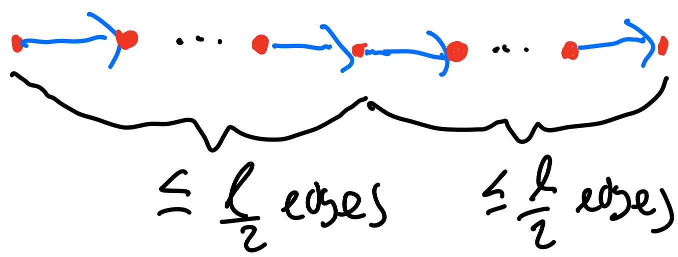
But wait, there's more!

Idea 2: <u>Divide-and-conquer</u>

Before:



Better:



$S[i][j][\ell]$

= shortest $i \to j$ path w/ $\leq 2^{\ell}$ edges

Only $O(n^2 \log(n))$ such problems!

$$S[i][j][\ell] = \min_{k \in V} S[i][k][\ell-1] + S[k][j][\ell-1]$$

"guess the midpoint"

Same base case, but less problems.

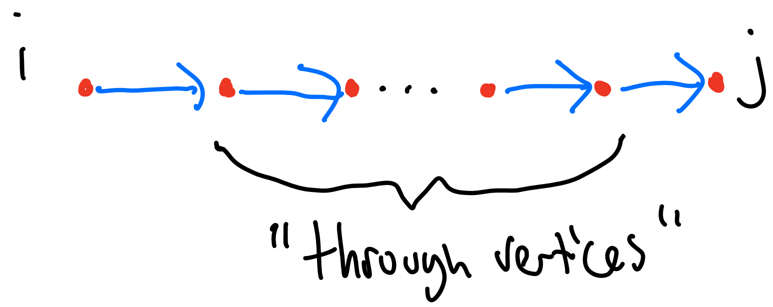
Runtime: $O(n^2 \log(n)) \times O(n) = O(n^3 \log(n))$

. . .

Idea 3: <u>Prefixes</u> (Floyd-Warshall)

Motivation: less cases in DP recursion

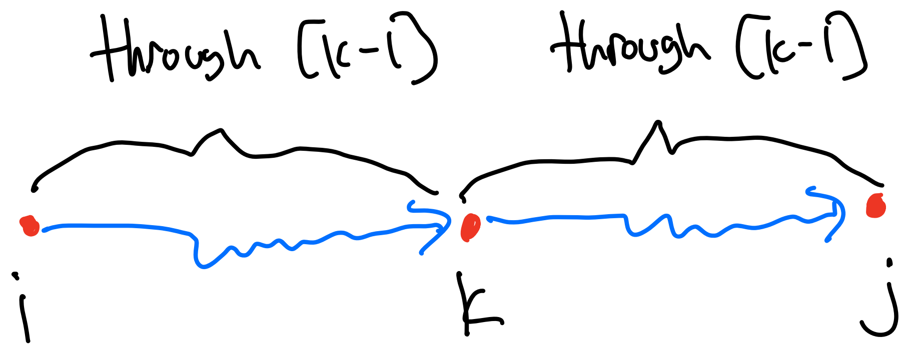
$S[i][j][k]$ = shortest $i \to j$ path through vertices $\subseteq [k]$



Observation: Can't duplicate a through vertex.

Case 1: no k.

Case 2: divide-and-conquer



$$S[i][j][k] = \min\Big(S[i][j][k-1],\ S[i][k][k-1] + S[k][j][k-1]\Big)$$

Runtime: $O(n^3) \times O(1) = O(n^3)$

DP done ☺