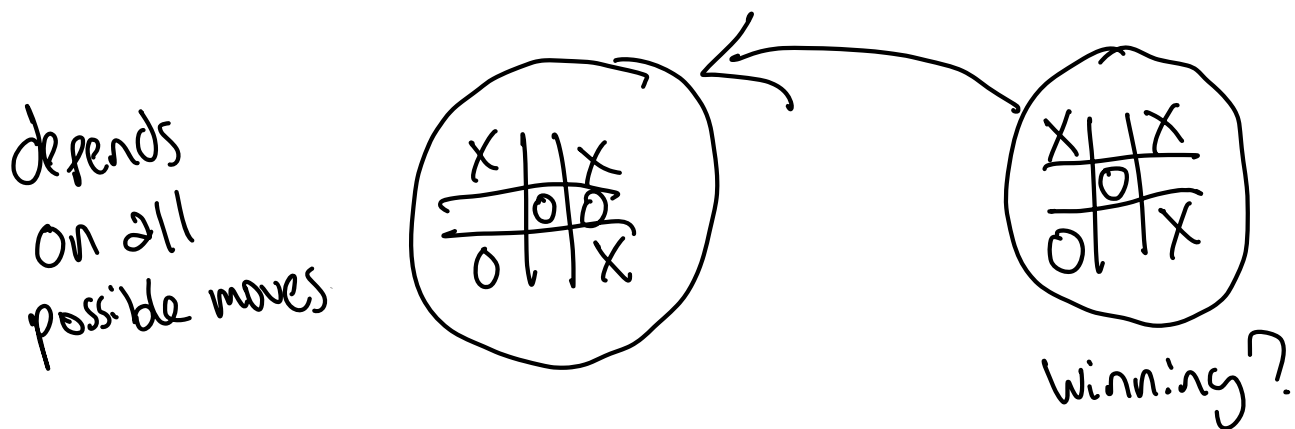


CS 331, Fall 2026

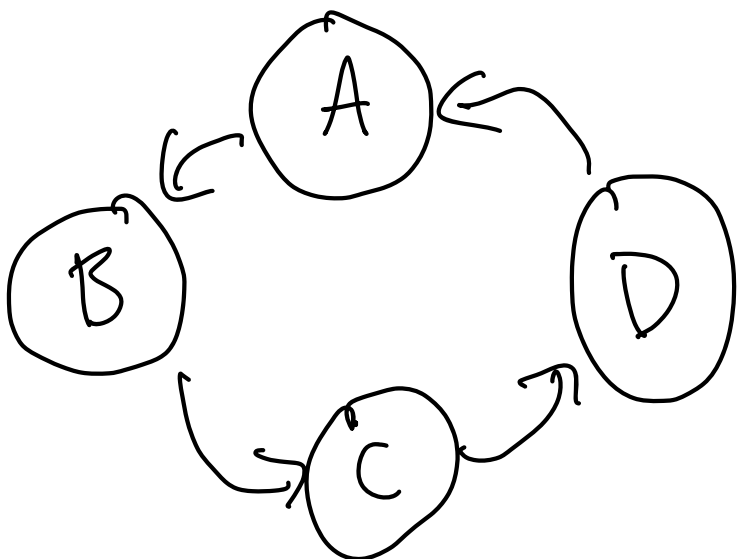
lecture 7 (9/16)

Today: - Graphs
- DAGs
- APSP

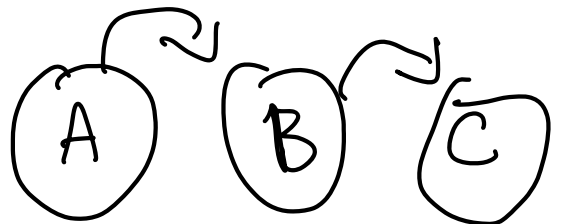
Motivation from last time:



What can get us in trouble?



e.g. chess



😊 e.g. tic-tac-toe

key diff: progress

Background: Graphs (Part I, Section 4)

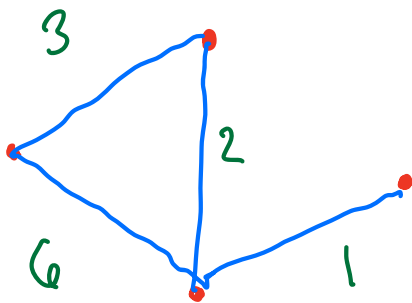
$$G = (V, E, w)$$

Vertices color: blue edges color: green weights

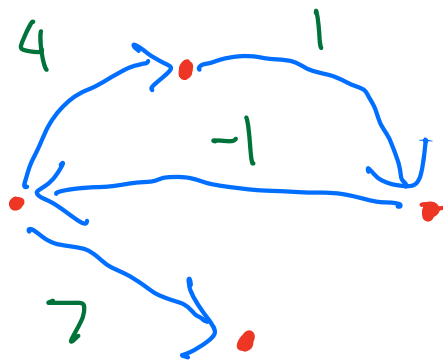
$n = |V|$
 $m = |E|$

Typically identify $V \equiv [n]$

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



undirected



directed

Edges: $e \in E$

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



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

$e \in E$

"tail"

"head"

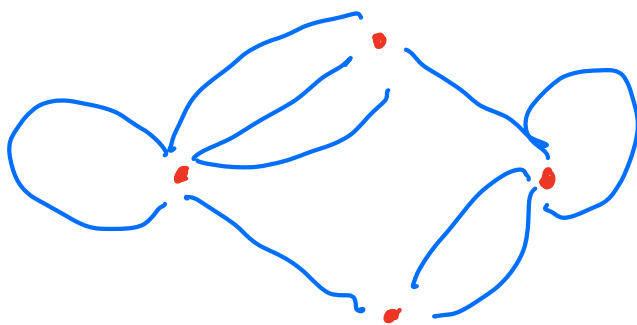
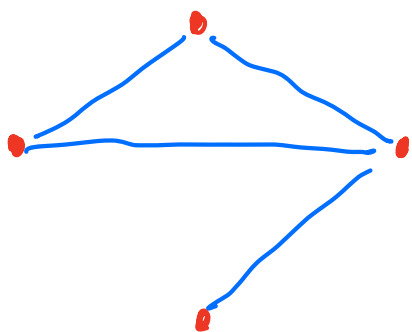


$w_e \in \mathbb{R}$

Quest: Simple

vs.

multigraph



All graphs assumed simple. no multi-edges self loops

$$\text{Hence, } m \leq n(n-1) = O(n^2)$$

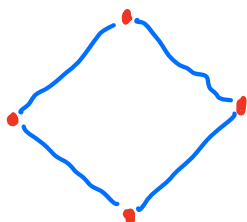
Special graphs

undirected

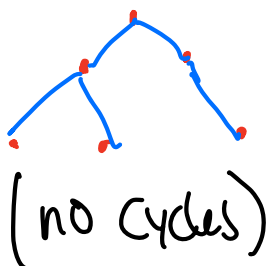
Paths



Cycles

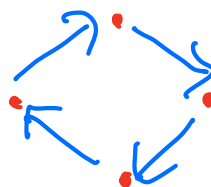


Trees



(no cycles)

directed



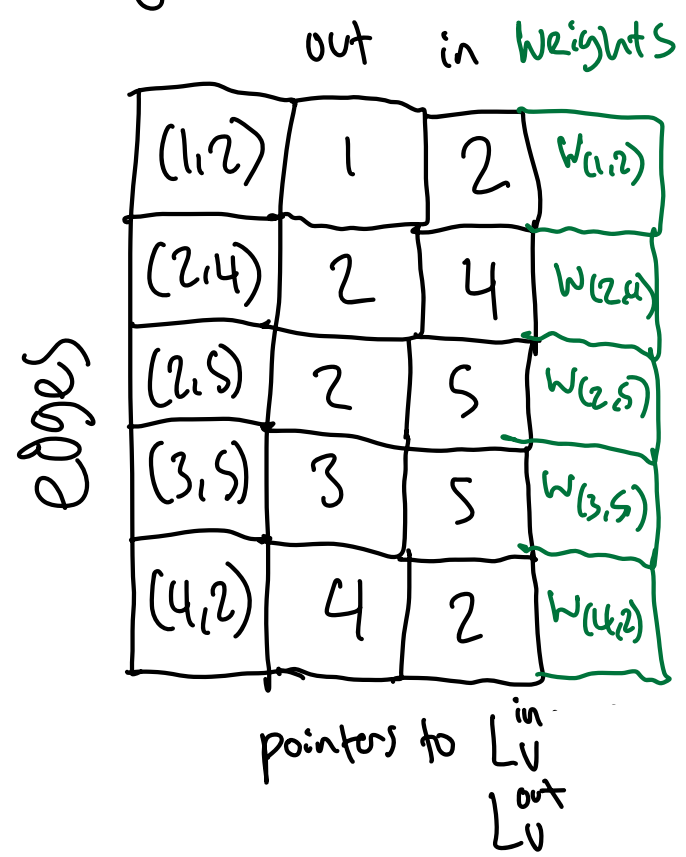
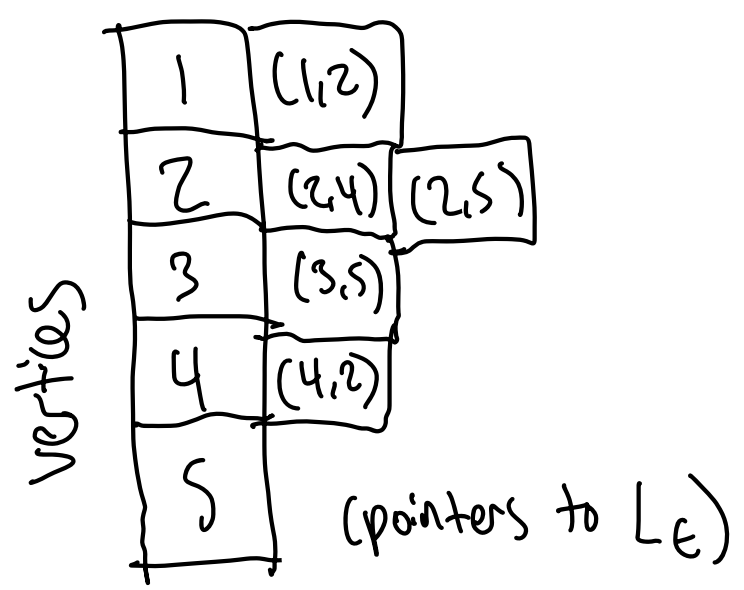
stay tuned...

Representing a graph

In this class: adjacency list model ($L_V^{\text{in}}, L_V^{\text{out}}, L_E$)

L_V^{out} : vertices know edges

L_E : edges know vertices



Space: $O(n + m)$

Primitives: • "pass over all edges"

runtime: $O(m)$

• "pass over neighbors of v "

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

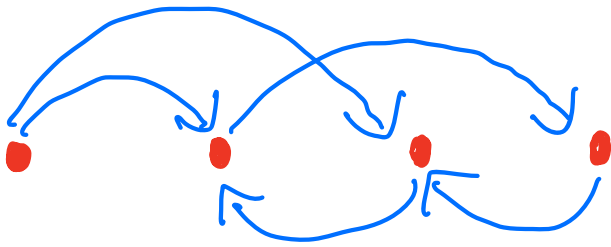
Directed acyclic graphs (Part III, Section 5.2)

Definition:

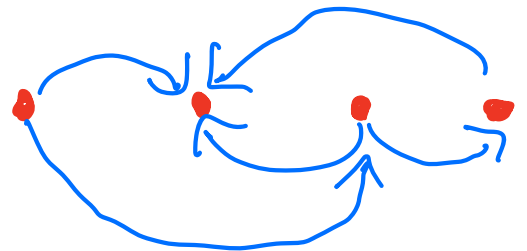
- Directed graph
- Acyclic

(duh)

Example



Not a DAG



DAG

It's not too obvious...

Can we make it easier to tell?

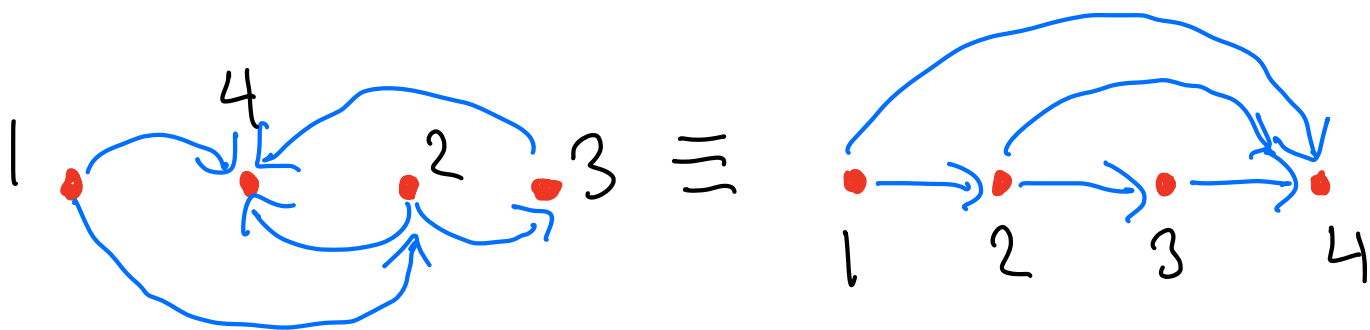
Yes! topological ordering

Topological ordering

Remove vertices $1, 2, \dots, n$

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

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



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

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

$\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad$

Contradiction!

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

runtime: $O(n + m)$

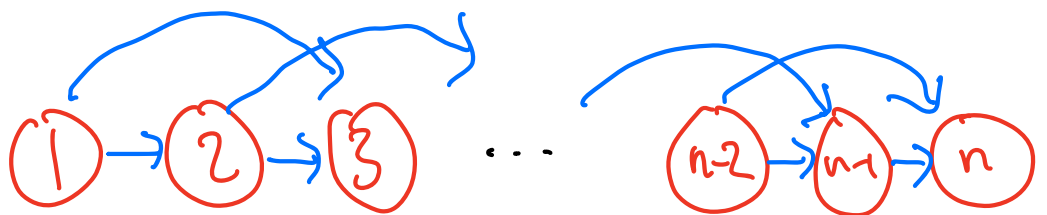
Consider DP algo.

Dependency graph must be DAG

- Vertices = subproblems
- edges: $u \rightarrow v$ iff $S[v] = \dots (S[u])$

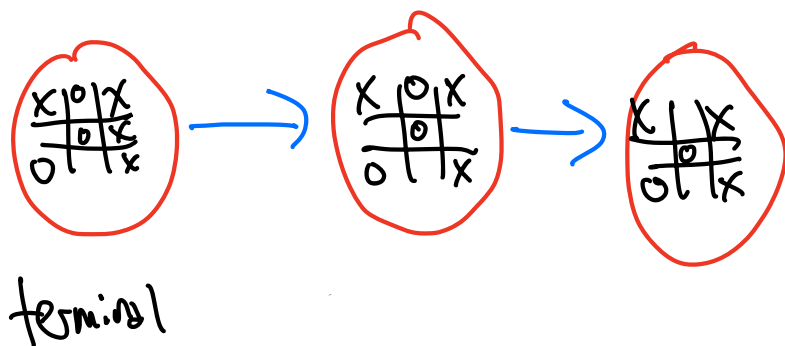
Example

Fibonacci:



Game theory

(reverse of
game graph)



Claim: evaluate in top order!

Proof: strong induction.

SSSP on DAGs

SSSP = Single source shortest paths

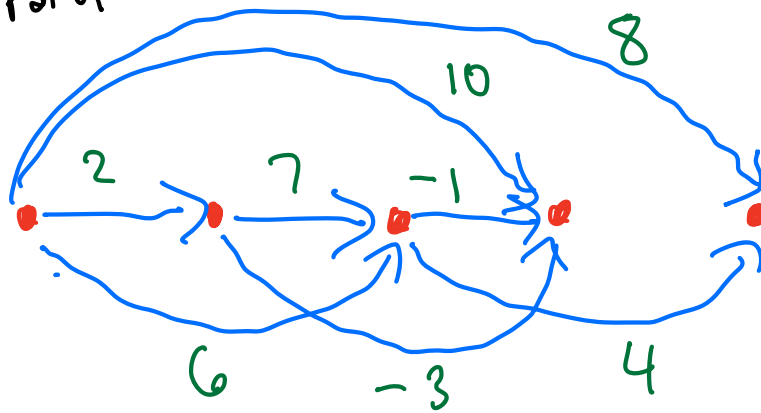
Input: DAG (V, E, w)

V in top order

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

$\min_{\substack{P \subseteq E \\ P \text{ is } 1 \rightarrow j \text{ path}}} \sum_{e \in P} w_e$ (think of as lengths)

Example



$S(j)$ = Shortest $1 \rightarrow j$ path

$S(1) = 0$ $S(2) = 2$ $S(3) = 6$

$S(4) = -1$ $S(5) = 8$

DP recursion:

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

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
- DAGs
- Multidimensional DP
- Divide-and-Conquer

Let's begin.

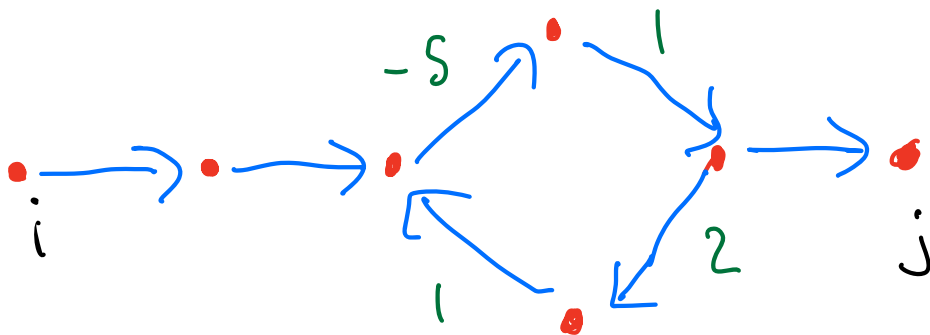
Input: $G = (V, E, w)$

Warning:
maybe not DAG.

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

$$\min_{\substack{P \subseteq E \\ P \text{ is } i \rightarrow j \text{ path}}} \sum_{e \in P} w_e \quad \left(\begin{array}{l} \text{unreachable} \\ = \infty \end{array} \right)$$

Assumption: no negative-weight cycles



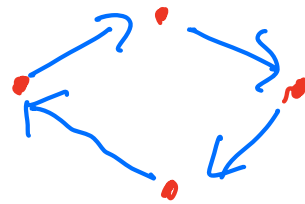
(revisit in Part V)

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

$$= O(n^3)$$

We'll get this for all graphs.

Issue: progress notion?



Idea 1: length of path

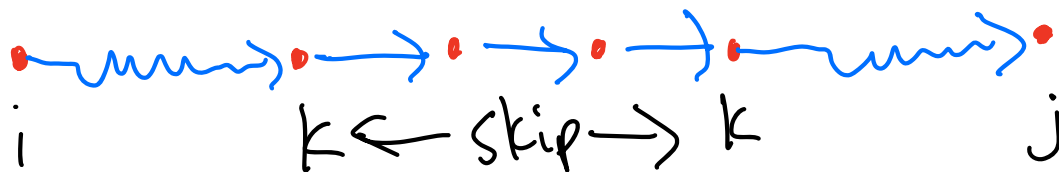
Try: $S(i)(j)(\ell)$

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

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

= shortest $i \rightarrow j$ path!

DP recursion: guess the last edge.

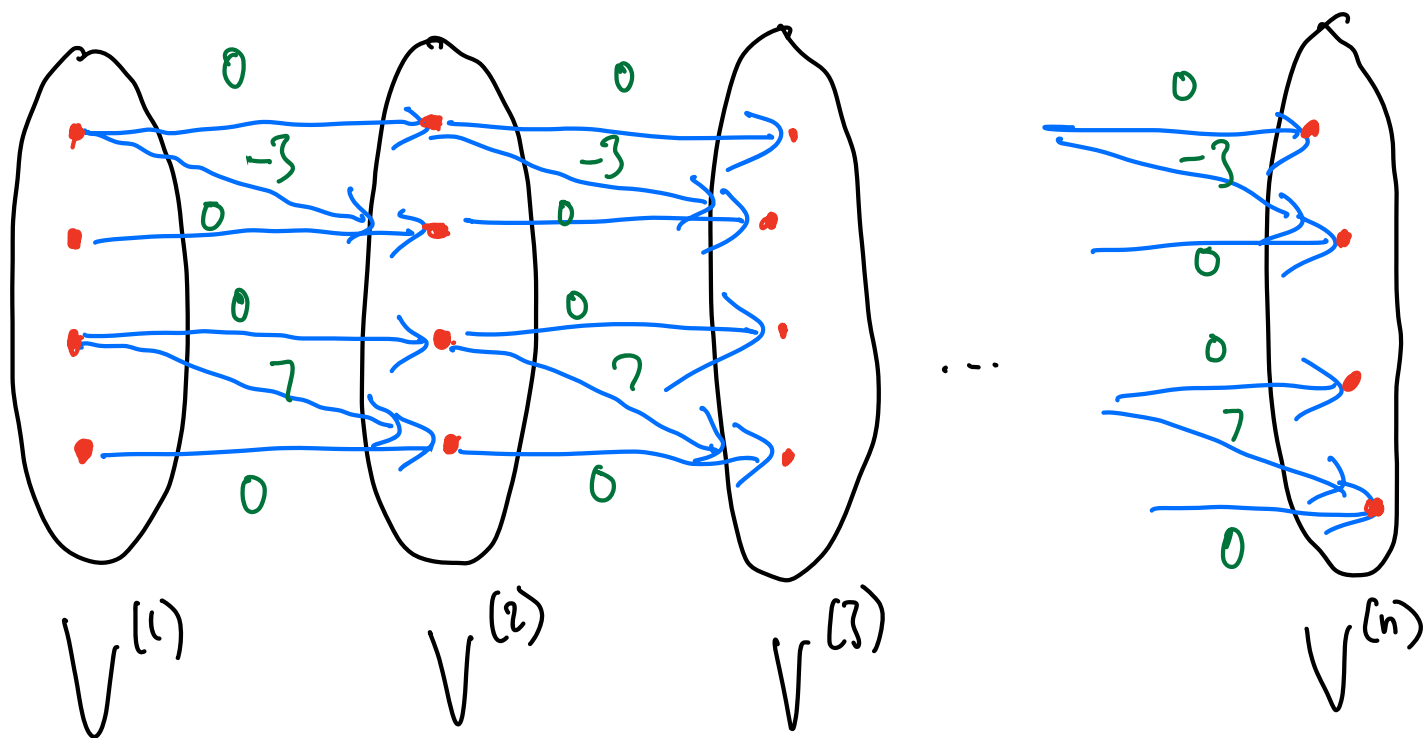
$$S(i, j, l) = \min_{(k, j) \in E} S(i, k, l-1) + W(k, j)$$

Base cases: $S(i, j, 1) = W(i, j)$ if $(i, j) \in E$
 ∞ else

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 $i^{(1)}, i^{(2)}, \dots, i^{(n)}$

0-weight "short cut edges" between $i^{(l)} \rightarrow i^{(l+1)}$

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

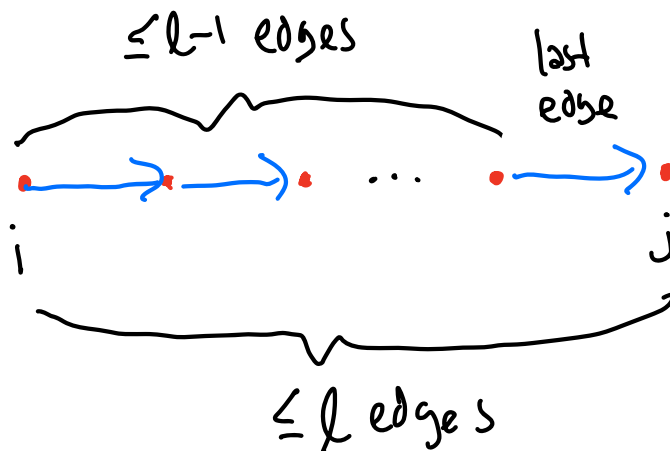
Claim 1: DAG. (Proof: layer = progress)

Claim 2: $i \rightarrow j$ shortest path
 $\equiv i^{(1)} \rightarrow j^{(n)}$ shortest path

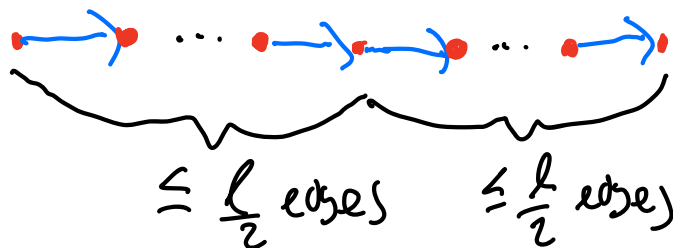
But wait, there's more!

Idea 2: Divide-and-conquer

Before:



Better:



$SC[i][j][l]$

= shortest $i \rightarrow j$ path w/ $\leq 2^l$ edges

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

$$S(i, j, l) = \min_{k \in V} S(i, k, l-1) + S(k, j, l-1)$$

"guess the midpoint"

Same base case, but less problems.

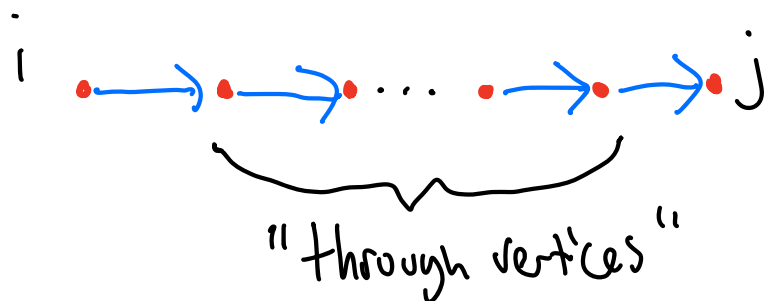
Runtime: $O(n^2 \log(n)) \times O(n) = O(n^3 \log(n))$

...

Idea 3: Prefixes (Floyd-Warshall)

Motivation: less cases in DP recursion

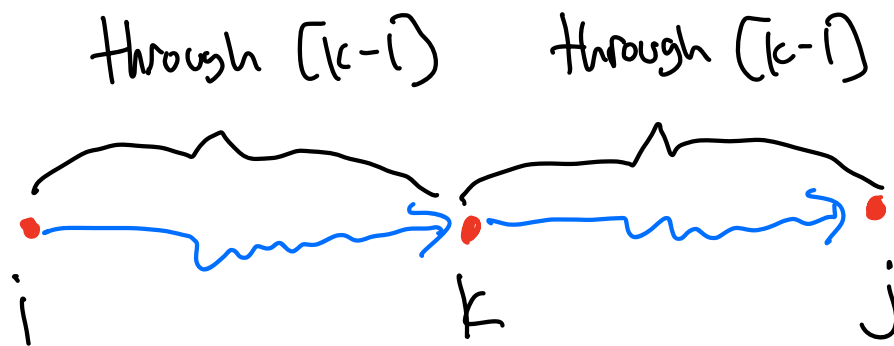
$S(i, j, k)$ = shortest $i \rightarrow j$ path
through vertices $\subseteq [k]$



Observation: Can't duplicate a
through vertex.

Case 1: no k .

Case 2: divide-and-conquer



$$S(i)(j)(k) = \min \left(S(i)(j)(k-1), \right. \\ \left. S(i)(k)(k-1) + S(k)(j)(k-1) \right)$$

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

DP done ☺