All-pairs shortest paths

- Directed graph $G=(V,E)$, weight function $w: E \to \mathbb{R}$, $|V| = n$
- Goal: Create $n \times n$ matrix of shortest path distances $\delta(i,j)$
- Bellman-Ford: Run once per vertex as source: $O(V^2 E)$, this is $O(n^4)$ on dense graph ($E = \Omega(V^2)$).

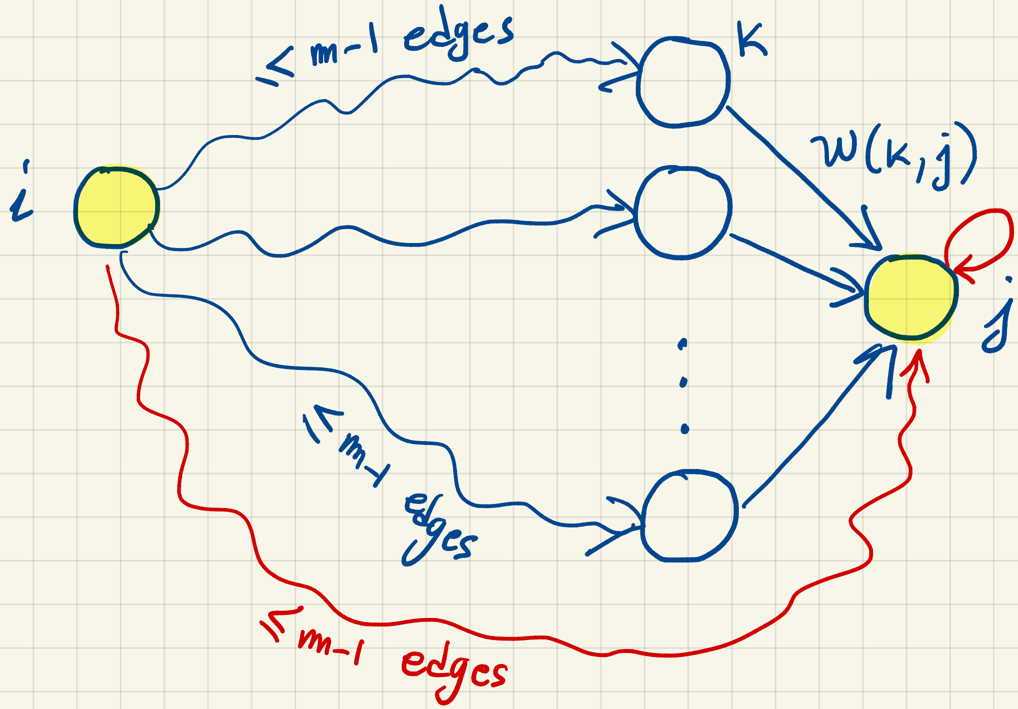
- Consider the adjacency matrix representation.
  - $n \times n$ matrix W where $W_{ij} = w(i,j)$
  - assume $W_{ii} = 0$ (No negative weight cycle $\Rightarrow$ that's shortest distance from i to i)

Dynamic Programming formulations:

Let $D_{ij}^{(m)}$ = weight of shortest path from i to j that uses at most m edges (length of path $\leq m$)

Then $D_{ij}^{(0)} = \begin{cases} 0 & i=j \\ \infty & i \neq j \end{cases}$

How can we write $D_{ij}^{(m)}$?



$$D_{ij}^{(m)} = \min_k \left[\underbrace{D_{ik}^{(m-1)} + w(k,j)}_{\text{when } k=j \text{ this is } D_{ij}^{(m-1)}} \right]$$

for $m \leftarrow 1$ to $n-1$

  do for $i \leftarrow 1$ to n

    do for $j \leftarrow 1$ to n

      do $D_{ij}^{(m)} \leftarrow \min\limits_{k=1..n} \left[D_{ik}^{(m-1)} + w(k,j) \right]$

return $D^{(m)}$

$D \leftarrow D^{(0)}$
for $m \leftarrow 1$ to $n-1$
    do $D' \leftarrow \infty$ (or $D' \leftarrow D$)
        for $i \leftarrow 1$ to n
            do for $j \leftarrow 1$ to n
                do for $k \leftarrow 1$ to n
                    do if $D'_{ij} > D_{ik} + w(k,j)$
                        then $D'_{ij} \leftarrow D_{ik} + w(k,j)$
        $D \leftarrow D'$

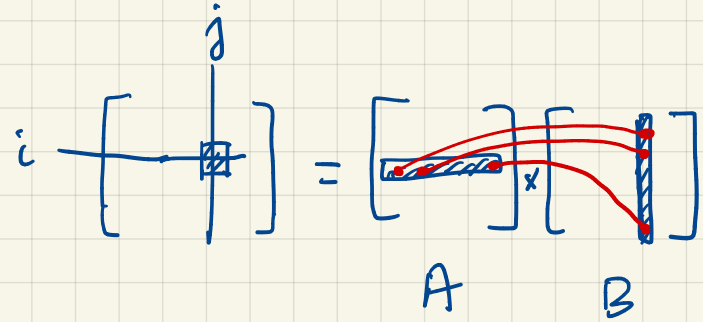
Using D' for the "new" D

$$\delta(i,j) = D_{ij}^{(n-1)} = D_{ij}^{(n)} = D_{ij}^{(n+1)} = \ldots \quad (\text{no} < 0 \text{ weight cycles})$$

Time: $O(n^4)$

Space: $O(n^2)$

Matrix multiplication



$C = A \times B$, $n \times n$ matrices

$$c_{ij} = \sum_k A_{ik} \cdot B_{kj}$$

This can be done in $O(n^3)$ time

Replace: $+ \rightarrow \min$

$\cdot \rightarrow +$

gives: $C_{ij} = \min_k [A_{ik} + B_{kj}]$ $(C = A \text{"x"} B)$

So $D^{(m)} = D^{(m-1)} \text{"x"} W$

$$D^{(0)} = \begin{bmatrix} 0 & & \infty \\ & 0 & \\ \infty & \ddots & \\ & & 0 \end{bmatrix}$$

is identity for "x".

$$D^{(1)} = D^{(0)} W = W$$

$$D^{(2)} = D^{(1)} W = W^2$$

$$\vdots$$

$$D^{(n-1)} = D^{(n-2)} W = W^{n-1}$$

So we have $\Theta(n)$ "multiplications", each requires $\Theta(n^3)$ time $\Rightarrow$ $\Theta(n^4)$ time.

Not better than before, but we can do matrix multiplication using repeated squaring!

Compute:

$$\underbrace{W,\ W^2,\ W^4,\ W^8,\ \ldots,\ W^{2^{\lceil \lg(n-1) \rceil}}}_{\Theta(\lg n) \text{ squarings}}$$

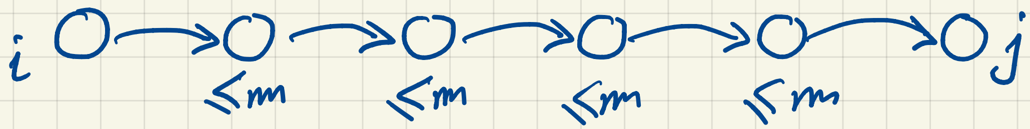
(ok to overshoot since product does not change after converging)

Time: $\Theta(n^3 \log n)$.

Floyd-Warshall : A faster DP.

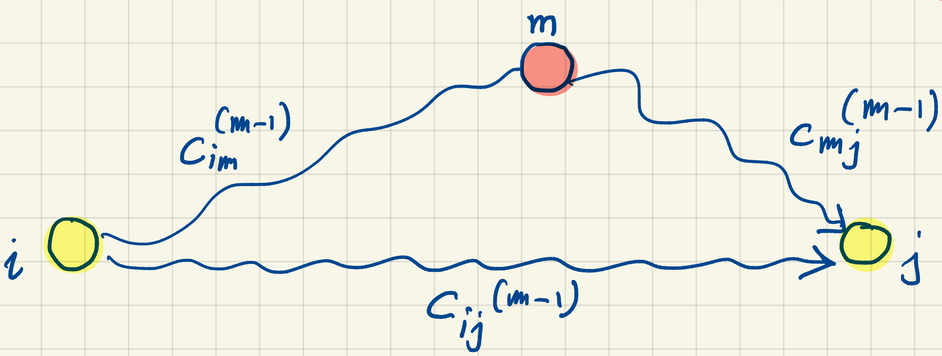
let $c_{ij}^{(m)}$ = weight of shortest path from i to j

with intermediate vertices in $\{1, 2, \dots, m\}$



Then $\delta(i,j) = c_{ij}^{(n)}$. $c_{ij}^{(0)} = W_{ij}$ (no intermediate vertices)

How can we write $c_{ij}^{(m)}$? (shortest path either includes m or doesn't)



$$c_{ij}^{(m)} = \min\left[c_{ij}^{(m-1)},\ c_{im}^{(m-1)} + c_{mj}^{(m-1)}\right]$$

Floyd-Warshall

Not trivial but superscripts can be dropped!

for $m \leftarrow 1$ to n
  do for $i \leftarrow 1$ to n
    do for $j \leftarrow 1$ to n
      do if $c_{ij} > c_{im} + c_{mj}$
        then $c_{ij} \leftarrow c_{im} + c_{mj}$

here, $c_{ij}^{(m)}$ is implicitly $c_{ij}^{(m-1)}$

The advantage is that we don't check all intermediate vertices as before. Time is $\Theta(n^3)$.

Space is $\Theta(n^2)$

Section 25.3: Johnson's alg. $O(V^2 \log V + VE)$

Floyd-Warshall for Transitive Closure:

The transitive closure G^* of G:

$(i,j) \in G^*$ iff $\exists$ path from i to j in G

Solution:
- Use adjacency matrix with elements in $\{0,1\}$ (no need for actual weight)
- Use Floyd-Warshall alg. replacing

$$\min \longrightarrow \text{OR}$$
$$+ \longrightarrow \text{AND}$$

$$C_{ij} \leftarrow C_{ij} \text{ OR } (C_{im} \text{ AND } C_{mj})$$

Linear programming with constraints of the form

$$x_j - x_i \leqslant b_k$$

Example: Find x_1, x_2, x_3 such that:

$$x_1 - x_2 \leqslant 3$$
$$x_2 - x_3 \leqslant -2$$
$$x_1 - x_3 \leqslant 2$$

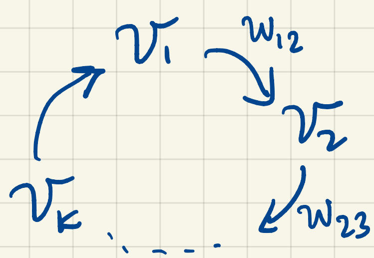
Solution: $x_1 = 3$, $x_2 = 0$, $x_3 = 2$

Goal: Find x_i that satisfy constraint or determine that there is no solution.

Construct graph: Add vertex for each of the n variables.
Add edge for each of the m constraints.

$$v_i \xrightarrow{b_k} v_j \qquad\qquad x_j - x_i \leq b_k$$

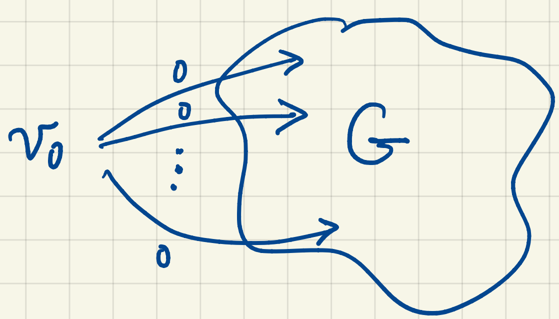
- Negative weight cycle $\Rightarrow$ No solution.



Suppose solution:

$$\begin{aligned}
\cancel{x_2} - \cancel{x_1} &\leq w_{12} \\
\cancel{x_3} - \cancel{x_2} &\leq w_{23} \\
&\vdots \\
\cancel{x_k} - x_{k-1} &\leq w_{k-1\,k} \\
\cancel{x_1} - \cancel{x_k} &\leq w_{k1} \\
\hline
0 &\leq \text{negative (contradiction)}
\end{aligned}$$

- No negative weight cycle $\Rightarrow$ Solution exists



$x_i = \delta(v_0, v_i)$ is a solution.

Proof: Triangular Inequality:

$$\delta(v_0, v_j) \leq \delta(v_0, v_i) + w(i,j)$$

$$x_j - x_i \leq w(i,j).$$

Bellman-Ford can be used and its running time would be

$O(VE)$ where $V = n+1$

$E = n + m$

So $O(n+1)(n+m) = O(n^2 + nm)$