Today's class: All Pair reachability in $o(n^2)$ time in fully dynamic setting in DAGs. with $o(n)$ query time.

Static All-Pairs Reachability

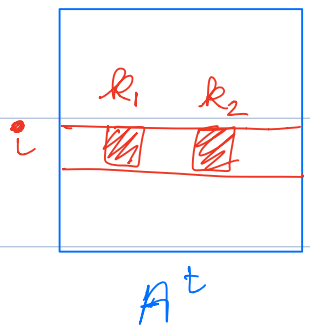
$$A^t(x,y) = \begin{cases} \neq 0 & \text{if } \exists\ x \text{ to } y \text{ path in } G \text{ of length at most } t. \\ 0 & \text{o/w.} \end{cases}$$

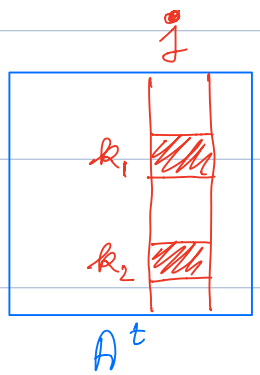
What is A^1 ?? $(Adj + I)$

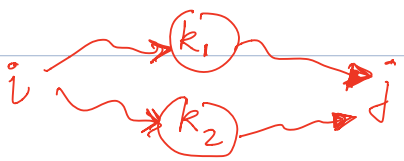
How to compute A^{2t} from A^t ?

Sol^n : Take A^t, make nonzero entries of it 1.

Output $(A^t) * (A^t)$.







Claim: Time to find A^n from A^1 is $O(n^w \log n)$, as we do $\log n$ multiplications to reach A^n from A^1.

Theorem: Time to multiply $n \times B$ & $B \times n$ matrix is $O\left(\frac{n^2}{B^{2-\omega}}\right)$ ← matrix mult coeff

$\omega \leq 2.38$

Proof:

$$\begin{bmatrix} M_1 \\ B\times B \\ M_2 \\ B\times B \\ M_3 \\ B\times B \end{bmatrix}_{n\times B} \quad \begin{bmatrix} \tilde{M}_1 & \tilde{M}_2 & \tilde{M}_3 \\ B\times B & B\times B & B\times B \end{bmatrix}_{B\times n}$$

(assume $\frac{n}{B}$ = integer)

$$= \begin{bmatrix} M_1\tilde{M}_1 & M_1\tilde{M}_2 & M_1\tilde{M}_3 \\ M_2\tilde{M}_1 & M_2\tilde{M}_2 & M_2\tilde{M}_3 \\ M_3\tilde{M}_1 & M_3\tilde{M}_2 & M_3\tilde{M}_3 \end{bmatrix}_{n\times n}$$

So time $= \left(\frac{n}{B}\right)^2 \cdot O(B^\omega) = O\left(\frac{n^2}{B^{2-\omega}}\right)$

Theorem: Let $P_1 P_2 \cdots P_B$ be B colums of size $n \times 1$ and $Q_1, Q_2 \cdots, Q_B$ are B rows of size $1 \times n$, then

$$\sum_{j=1}^{B} (P_j \cdot Q_j) = [P_1 \cdots P_B] \cdot [Q_1 \cdots Q_B]$$

$$\sum_{j=1}^{B} \left(P_j \cdot Q_j \right) = \left(P_1 \cdots P_B \right)_{n \times B} \begin{pmatrix} Q_1 \\ \vdots \\ Q_B \end{pmatrix}_{B \times n}$$

Proof Hint:

It suffices to show that $(i,k)^{th}$ entry of $\sum_{j=1}^{B} (P_j \cdot Q_j)$ is same as $(i,k)^{th}$ entry of $([P_1 \cdots P_B] \cdot [Q_1 \cdots Q_B])$.

———— x ———— x ————

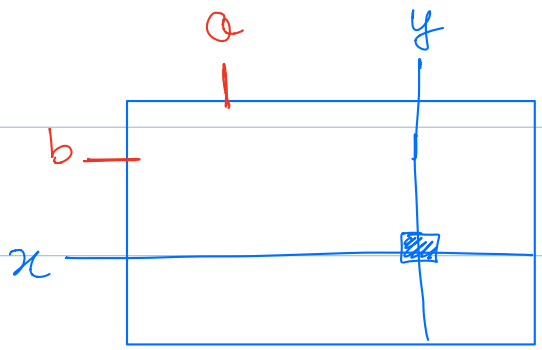
Ques: Can we get better than n^2 update time at cost of higher query time?

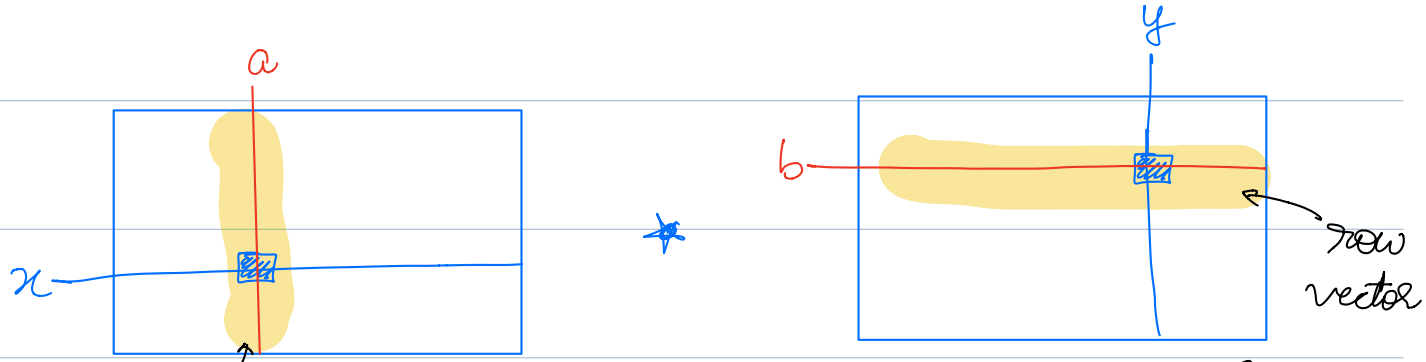
(We will update $\tilde{C}$ in lazy manner).

Algorithm 2:

Recall $C_t(x,y) = C_{t-1}(x,y) \pm C_{t-1}(x,a) \cdot C_{t-1}(b,y)$

$\pm$: addition / deletion of (a,b)





column vector $\left(C^{col}_{t-1}(a)\right)$

$\left(C^{row}_{t-1}(b)\right)$

OBSERVATION:

Take product of $C_{t-1}^{col}(a)$ & $C_{t-1}^{row}(b)$.

The $(x,y)^{th}$ entry here is the term to be $C_{t-1}(x,a) \cdot C_{t-1}(b,y)$ added or subtracted.

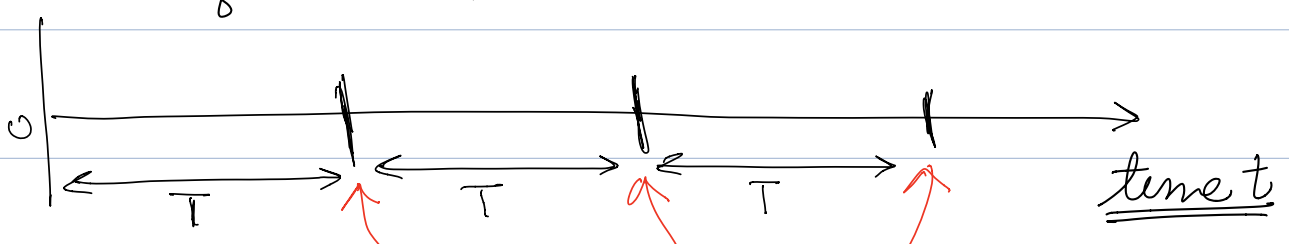
Fact: $C_t = C_{t-1} \pm \underbrace{C_{t-1}^{col}(a)}_{\text{column vector}} \cdot \underbrace{C_{t-1}^{row}(b)}_{\text{row vector}}$

$\pm$: add^n / del^n of edge (a, b)

So, if $(a_1 b_1)\ (a_2 b_2)\ \cdots\ (a_T b_T)$ are updates then,

$$C_T = C_0 + \underbrace{\left[\pm C_0^{col}(a_1) \pm C_1^{col}(a_2) \cdots \pm C_{T-1}^{col}(a_T)\right]}_{n \times T} \cdot \underbrace{\begin{bmatrix} C_0^{row}(b_1) \\ C_1^{row}(b_2) \\ \vdots \\ C_{T-1}^{row}(b_T) \end{bmatrix}}_{T \times n}$$

Batch of T updates:



Update the matrix "$\tilde{C}$" only after batch of T edge updates.

How to answer queries ??

Suppose we have "C_0", during a batch of T updates we explicitly maintain:

$$C_0^{col}(q_1) \cdots C_{T-1}^{col}(q_T) \quad \& \quad C_0^{row}(b_1) \cdots C_{T-1}^{row}(b_T)$$

After $K \leq T$ edge updates if query (x, y) comes then

$$C_K(x,y) = C_0(x,y) + \sum_{i=1}^{K} \pm C_{i-1}^{col}(a_i)[x] \cdot C_{i-1}^{row}(b_i)[y]$$

(x, a_i) $\qquad$ (b_i, y)

So, query time is $O(K)$

$= O(T)$

- Now the question is how to maintain $C_0^{col}(q_1) \dots C_{T-1}^{col}(q_T)$ & $C_0^{row}(b_1) \dots C_{T-1}^{row}(b_T)$.

- After k updates, we would have computed $C_0^{col}(q_1) \dots C_{k-1}^{col}(q_k)$ & $C_0^{row}(b_1) \dots C_{k-1}^{row}(b_k)$.

- Now at $(k+1)^{th}$ edge update, i.e. (q_{k+1}, b_{k+1}) added/removed we need to find $C_k^{col}(q_{k+1})$ & $C_k^{row}(b_{k+1})$.

i.e. $\forall v$, $C_k(v, q_{k+1})$ & $C_k(b_{k+1}, v)$.

As there are '2n' entries to update we can just spent $O(T)$ time for all 2n values.

Total time for this task after each edge update $= O(Tn)$.

Now amortized update time is

$$O(nT) + \frac{O\left(\frac{n^2}{T^{2-\omega}}\right)}{T}$$

$$= O\left(nT + \frac{n^2}{T^{3-\omega}}\right)$$

$$nT = \frac{n^2}{T^{3-\omega}} \Rightarrow T = n^{\frac{1}{4-\omega}}$$

$$= O\left(n^{1+\frac{1}{4-\omega}}\right) = O(n^{1.616}).$$

Query time $= O(T) = O(n^{0.616})$.