Lecture 22

COL 351: Analysis and Design of Algorithms

Lecture 22

$\boxed{\begin{array}{c}\text{Transitivity Relation} \\ \text{If } i \to j \ \& \ j \to k \text{ then } i \to k\end{array}}$

This relation holds for Reachability property

Matrix Product / All-Pairs Shortest Path / Transitive Closure

(Divide and Conquer strategy)

Transitive Closure

Given: Directed graph $G = (V, E)$.

Find: For each $v \in V$, the vertices reachable from v in graph G.

Naive Way: Run BFS / DFS algorithm from every vertex

- Run time: $O(mn) = O(n^3)$

Better than n^3 algo?

* Floyd Warshall Algo (DP)
  $O(n^3)$

All Pairs Shortest Path

Given: Directed / undirected graph $G = (V, E)$.

Find: The shortest path (or distances) between all vertex pairs.

— **Dijkstra's algorithm from every vertex**

- Run time: $O(mn + n^2 \log n)$

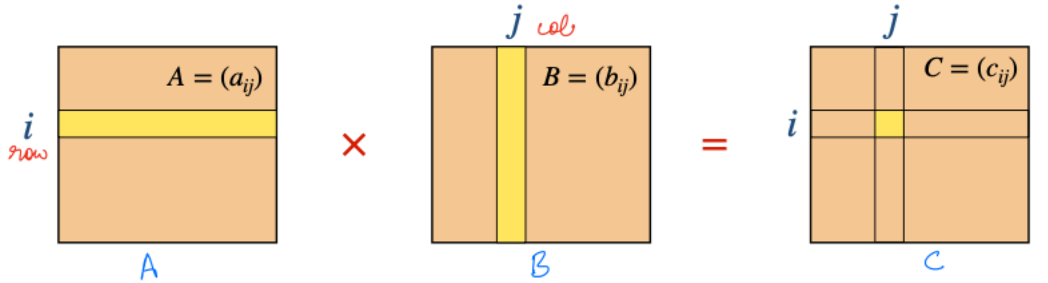
— **Floyd-Warshall algorithm (Lec 14)**

- Run time: $O(n^3)$

$O(n^3)$

Better than n^3 algo?

Matrix Multiplication of $n \times n$ Square Matrices



$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Run time = $O(n^3)$

Better than n^3 algo?

Matrix Multiplication

(Divide and Conquer strategy)

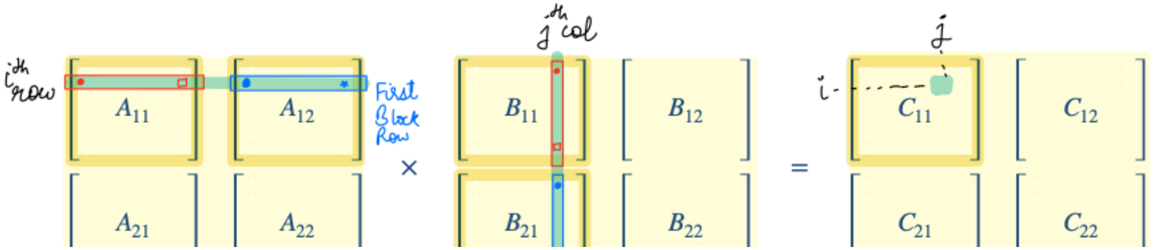
Product of 2x2 integer matrices

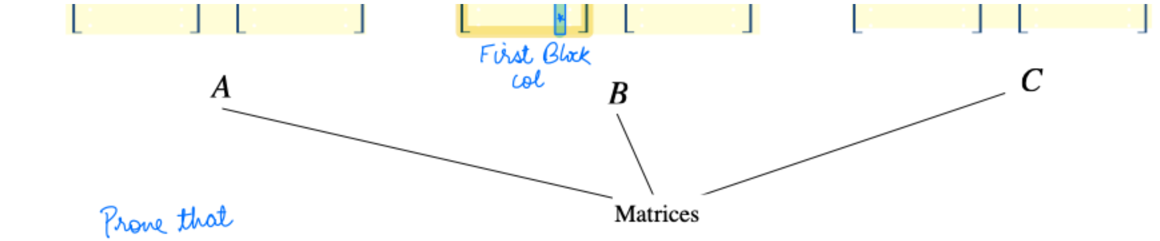
$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

8 integer multiplications
4 integer additions

$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$
$$c_{12} = \dots a_{11} b_{12} + a_{12} b_{22}$$
$$c_{21} = \dots a_{21} b_{11} + a_{22} b_{21}$$
$$c_{22} = \dots a_{21} b_{12} + a_{22} b_{22}$$

Product of Block-matrices





Prove that

$$\underbrace{C_{11}}_{(i,j)} = \underbrace{A_{11}B_{11}}_{(i,j)^{th}} + \underbrace{A_{12}B_{21}}_{(ij)^{th}}$$

Product of Block-matrices

i^{th} row, j^{th} column

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{array}{c|c} & j \\ \hline i & \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \end{array}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = \ldots A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = \ldots A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = \ldots A_{21}B_{12} + A_{22}B_{22}$$

$$T(n) = 8T(n/2) + O(n^2)$$

H.W. $T(n) = ?$

Strassen's Divide & Conquer Algorithm

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} = A_{11}B_{11} + A_{12}B_{21} & C_{12} = A_{11}B_{12} + A_{12}B_{22} \\ C_{21} = A_{21}B_{11} + A_{22}B_{21} & C_{22} = A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

It suffices to compute 7 product matrices

$$P_1 = (A_{11}) \times (B_{12} - B_{22})$$
$$P_2 = (A_{11} + A_{12}) \times (B_{22})$$
$$P_3 = (A_{21} + A_{22}) \times (B_{11})$$
$$P_4 = (A_{22}) \times (B_{21} - B_{11})$$
$$P_5 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$
$$P_6 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$
$$P_7 = (A_{11} - A_{21}) \times (B_{11} + B_{12})$$

$$C_{11} = (P_4 + P_5 + P_6 - P_2)$$
$$C_{12} = (P_1 + P_2)$$
$$C_{21} = (P_3 + P_4)$$
$$C_{22} = (P_1 + P_5 - P_3 - P_7)$$

Goal $\quad T(n) = 7T(n/2) + O(n^2)$

Recurrence relation in Strassen's Algorithm for matrix-product is
$T(n) = 7\ T(n/2) + d\ n^2$

$$a^{\log_2(b)} = b^{\log_2(a)}$$

$$T(n) \leq 7\,T\left(\frac{n}{2}\right) + dn^2$$

$$\leq 7\left(7\,T\left(\frac{n}{4}\right) + \frac{dn^2}{4}\right) + dn^2 = 7^2\,T\left(\frac{n}{4}\right) + dn^2\left(1 + \frac{7}{4}\right)$$

$$\leq 7^2\left(7\,T\left(\frac{n}{8}\right) + \frac{dn^2}{16}\right) + dn^2\left(1 + \frac{7}{4}\right)$$

$$\leq 7^3\,T\left(\frac{n}{8}\right) + dn^2\left(1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2\right)$$

$$\vdots$$

$$\leq 7^i\,T\left(\frac{n}{2^i}\right) + dn^2\left(1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2 + \dots + \left(\frac{7}{4}\right)^i\right)$$

Put $i = \log_2 n$

$$\leq 7^i\,T\left(\frac{n}{2^i}\right) + \frac{7}{3}dn^2\left(\frac{7}{4}\right)^i$$

$$\leq 7^{\log_2 n} + \frac{7}{3}dn^2\left(\frac{7}{4}\right)^{\log_2 n} = O\left(n^{\log_2 7} + n^2\, n^{\log_2 7/4}\right) = O\left(n^{\log_2 7}\right)$$

Matrix Multiplication Algorithms

Strassen - $O(n^{2.81})$

$\vdots$

Coppersmith and Winograd - $O(n^{2.376})$

Coppersmith and Winograd - $O(n$ $)$

Alman and Williams - $O(n^{2.373})$

Notation "ω" is used to denote the "smallest" constant such that two $n \times n$ square matrices can be multiplies in $O(n^{\omega})$ time.

$\omega \geqslant 2$

Transitive Closure

Given: Directed graph $G = (V, E)$.

Find: For each $v \in V$, the vertices reachable from v in graph G.

A is an adjacency-matrix of G

$$\text{if } A_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is edge in } G \\ 0 & \text{otherwise} \end{cases}$$

T is an transitive-closure-matrix of G

$$\text{if } T_{ij} = \begin{cases} 1 & \text{there is a path from } (i) \text{ to } (j) \text{ in } G \\ 0 & \text{otherwise} \end{cases}$$

Lemma 1: Let A be adjacency matrix of a graph. Then,

$(A^k)_{ij} > 0$ iff there is a walk of length exactly 'k' from (i) to (j).

If $\exists$ a walk from i to j of size k

$\boxed{\text{Hyp}(k)}$

If $(A^k)_{ij} > 0$

$\Rightarrow (A^k)_{ij} > 0$

$\Rightarrow \exists$ a walk from i to j of size k

H.W.

(Need to use fact that we are dealing with matrices of non-negative entries)

Assume Hyp(k-1) holds.

Now suppose for some i, j, $(A^k)_{ij} > 0$.

$\Rightarrow \exists x \in [1,n]$ s.t. $(A^{k-1})_{ix}, (A)_{xj} > 0$

By Hyp(k-1), $\exists$ a walk of size (k-1) from i to x and also (x, j) is an edge

$\Rightarrow \exists$ a walk of size k from i to j.

Lemma 1: Let A be adjacency matrix of a graph. Then,

$(A^k)_{ij} > 0$ iff there is a walk of length exactly 'k' from (i) to (j).

Lemma 2: Let A be adjacency matrix of a graph. Then,

$((I+A)^k)_{ij} > 0$ iff there is a walk of length at most 'k' from (i) to (j).

Proof Sketch:

$$(I+A)^k = \underbrace{I}_{A^0} + ({}^kC_1)A + ({}^kC_2)A^2 + \cdots + A^k$$

$((I+A)^k)_{ij} > 0 \iff \exists x \in [0,k]$ s.t. $(A^x)_{ij} > 0 \iff \exists x \in [0,k]$ s.t. there is a walk of size "x" from i to j

$\Updownarrow$

$\exists$ a path from i to j of size $\le k$

Transitive Closure

Transitive-Closure(A)

$M = I + A$;

$n = size(A)$;

```
n = size(A);

For i = 1 to ⌈log_2 n⌉:
    M = M^2
    Replace non-zero entries in M by 1;

Return M;
```

Result: The transitive closure of graph with n vertices is computable in $O(n^\omega \log n)$ time.