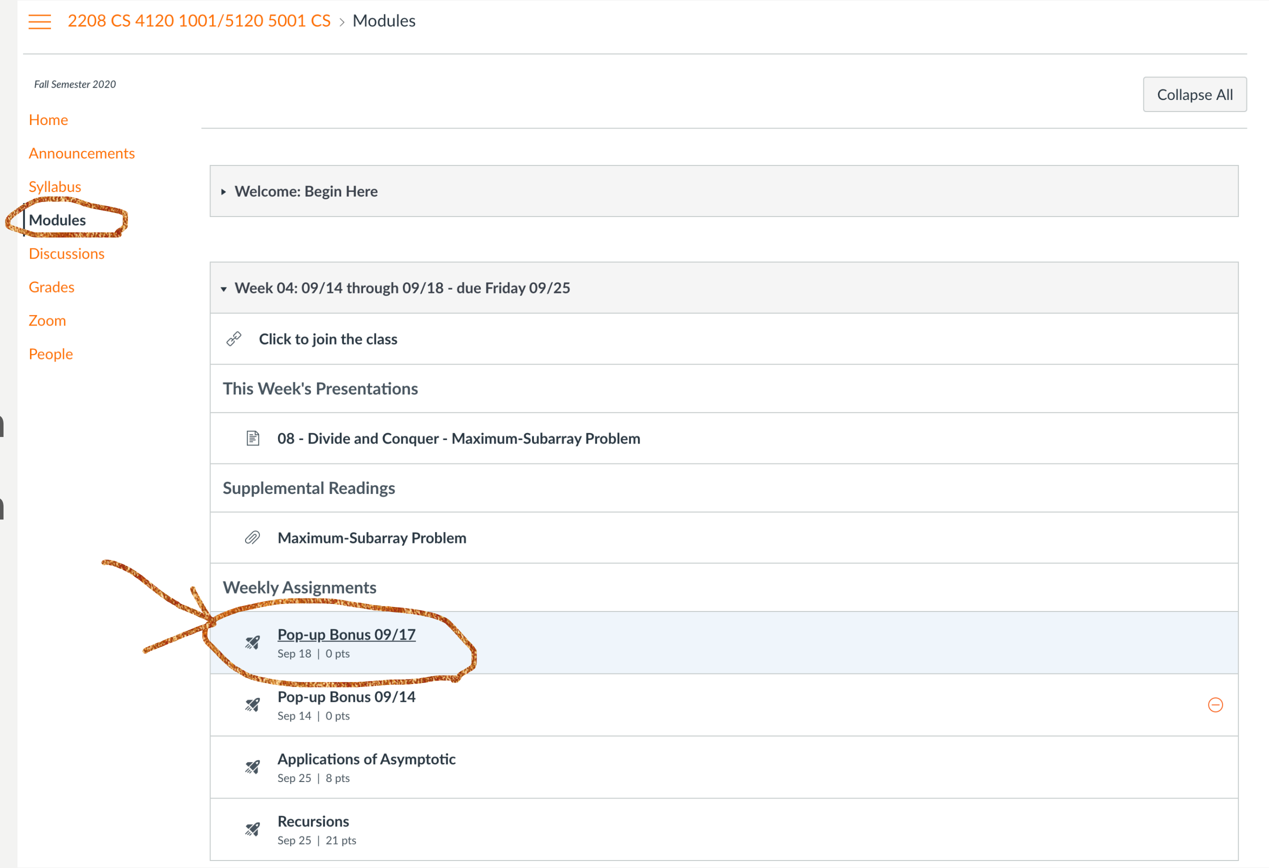
DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

STRASSEN'S ALGORITHM

[BONUS] 5-MIN CHALLENGE

- Pop-up 6 bonus points.
- One attempt
- Closes 5 minutes after class starts
  - Section 1001/5001 closes @ 2:35pm
  - Section 1002/5002 closes @ 3:35pm



AGENDA

- Review matrix multiplication
- Strassen's algorithm

SQUARE MATRIX MULTIPLICATION
2-BY-2

- Given two **2**-by-**2** matrices. Compute their dot product $C = A \cdot B$.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = A \cdot B = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

  - Entry C_{ij} is calculated as $\sum_{k=1}^{\mathbf{2}} A_{ik} B_{kj}$
  - To calculate one entry of the resulting matrix.
    - ___________ additions and ___________ multiplications are involved.

SQUARE MATRIX MULTIPLICATION
3-BY-3

- Given two **3**-by-**3** matrices. Compute their dot product $C = A \cdot B$.

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix}, C = A \cdot B = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix}$$

  - Entry C_{ij} is calculated as $\sum_{k=1}^{3} A_{ik} B_{kj}$
  - To calculate one entry C_{ij}.
    - ___________ additions and ___________ multiplications are involved.

SQUARE MATRIX MULTIPLICATION n-BY-n

- Given two $\boldsymbol{n}$-by-$\boldsymbol{n}$ matrices. Compute their dot product $C = A \cdot B$.

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1\boldsymbol{n}} \\ \vdots & \ddots & \vdots \\ A_{\boldsymbol{n}1} & \cdots & A_{\boldsymbol{nn}} \end{pmatrix}, B = \begin{pmatrix} B_{11} & \cdots & B_{1\boldsymbol{n}} \\ \vdots & \ddots & \vdots \\ B_{\boldsymbol{n}1} & \cdots & B_{\boldsymbol{nn}} \end{pmatrix}, C = A \cdot B = \begin{pmatrix} C_{11} & \cdots & C_{1\boldsymbol{n}} \\ \vdots & \ddots & \vdots \\ C_{\boldsymbol{n}1} & \cdots & C_{\boldsymbol{nn}} \end{pmatrix}$$

  - Entry C_{ij} is calculated as $\sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \boldsymbol{A_{ik} B_{kj}}$
  - To calculate **one** entry C_{ij}.
    - ___________ additions and ___________ multiplications are involved.

SQUARE MATRIX MULTIPLICATION GENERALIZATION

- Compute $C = A \cdot B$, where A and B are n-by-n matrices.
  - To compute one entry of C,

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

    - $\underline{\quad n-1 \quad}$ additions and $\underline{\quad n \quad}$ multiplications are involved.
  - There are $\underline{\quad n^2 \quad}$ entries in C.
  - There are total $\underline{\quad n^2(n-1) \quad}$ additions and $\underline{n^2 \cdot n = n^3 \quad}$ multiplications being involved in computing $C = A \cdot B$.

TRADITIONAL SQUARE-MATRIX-MULTIPLICATION

- Complete the **Cost** and **Time** columns.
  - If you prefer Σ, you may use the following notation for the **Time** column.
    - $\Sigma 1 + 1$ for line 3
    - $\Sigma(\Sigma 1 + 1)$ for line 4
    - $\Sigma\Sigma 1$ for line 5
    - $\Sigma\Sigma(\Sigma 1 + 1)$for line 6
    - $\Sigma\Sigma\Sigma 1$ for line 7

SQUARE-MATRIX-MULTIPLY (A, B)		Cost	Time
1	$n = A.rows$	$\Theta(1)$	1
2	Let C be a new $n \times n$ matrix	$\Theta(1)$	1
3	**for** $i = 1$ **to** n	$\Theta(1)$	$n + 1$
4	**for** $j = 1$ **to** n	$\Theta(1)$	$n(n + 1)$
5	$c_{ij} = 0$	$\Theta(1)$	$n \cdot \boldsymbol{n}$
6	**for** $k = 1$ **to** n	$\Theta(1)$	$n \cdot \boldsymbol{n} \cdot (\boldsymbol{n} + \boldsymbol{1})$
7	$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$	$\Theta(1)$	$n \cdot \boldsymbol{n} \cdot \boldsymbol{n}$
8	**return** C	$\Theta(1)$	1

TRADITIONAL SQUARE-MATRIX-MULTIPLICATION

- The running time function $T(n)$
  $=$
- The bound of $T(n) =$
  - **For an asymptotically positive polynomial $p(n)$ of degree n, we have $p(n) = \Theta(n^d)$.**

SQUARE-MATRIX-MULTIPLY (A, B)	Cost	Time
$n = A.rows$	$\Theta(1)$	1
Let C be a new $n \times n$ matrix	$\Theta(1)$	1
for $i = 1$ **to** n	$\Theta(1)$	$n + 1$
for $j = 1$ **to** n	$\Theta(1)$	$n(n + 1)$
$c_{ij} = 0$	$\Theta(1)$	$n \cdot n$
for $k = 1$ **to** n	$\Theta(1)$	$n \cdot n \cdot (n + 1)$
$c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$	$\Theta(1)$	$n \cdot n \cdot n$
return C	$\Theta(1)$	1

SQUARE-MATRIX-MULTIPLICATION
A DIVIDE-AND-CONQUER APPROACH

- **Divide** the input $n \times n$ matrices into four $\frac{n}{2} \times \frac{n}{2}$ sub-matrices.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

SQUARE-MATRIX-MULTIPLICATION
A DIVIDE-AND-CONQUER APPROACH

- **Conquer** $\frac{n}{2} \times \frac{n}{2}$ sub-matrix multiplication.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

  – Following the matrix-mult. rules

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

$$\begin{matrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{matrix}$$

SQUARE-MATRIX-MULTIPLICATION
A DIVIDE-AND-CONQUER APPROACH

- **Combine** the resulting $\frac{n}{2} \times \frac{n}{2}$ sub-matrices into ONE matrix.
  - Put the sub-matrix at the corresponding position in the original matrix.

$$C = \begin{pmatrix} \boxed{C_{11}} & \boxed{C_{12}} \\ \boxed{C_{21}} & \boxed{C_{22}} \end{pmatrix}$$

DIVIDE-AND-CONQUER SQUARE-MATRIX-MULTIPLICATION

- Complete the **Cost** and **Time** columns.

SQUARE-MATRIX-MULTIPLY-RECURSIVE (A, B)	Cost	Time
$n = A.rows$	$\Theta(1)$	1
Let C be a new $n{\times}n$ matrix	$\Theta(1)$	1
if $n == 1$	$\Theta(1)$	base
$c_{11} = a_{11} \cdot b_{11}$	$\Theta(1)$	base
else partition A, B, and C in 4 equal parts	$\Theta(1)$	1
C_{11} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{12}, B_{21})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
C_{12} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{12}, B_{22})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
C_{21} = SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{21}, B_{11}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{22}, B_{21})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
C_{22} = SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{21}, B_{12}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{22}, B_{22})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
return C	$\Theta(1)$	1

DIVIDE-AND-CONQUER SQUARE-MATRIX-MULTIPLICATION

- The running time function $T(n)$ $=$

SQUARE-MATRIX-MULTIPLY-RECURSIVE (A, B)		Cost	Time
1	$n = A.rows$	$\Theta(1)$	1
2	Let C be a new $n \times n$ matrix	$\Theta(1)$	1
	if $n == 1$	$\Theta(1)$	base
	$c_{11} = a_{11} \cdot b_{11}$	$\Theta(1)$	base
	else partition A, B, and C in 4 equal parts	$\Theta(1)$	1
	C_{11} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{11}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{12}, B_{21})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
3	C_{12} = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_{11}, B_{12}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{12}, B_{22})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
4	C_{21} = SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{21}, B_{11}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{22}, B_{21})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
5	C_{22} = SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{21}, B_{12}) + SQUARE-MATRIX-MULTIPLY-RECURSIVE (A_{22}, B_{22})	$2T\left(\frac{n}{2}\right) + \Theta(n^2)$	1
8	**return** C	$\Theta(1)$	1

SQUARE-MATRIX-MULTIPLY-RECURSIVE

- Tolerate sloppiness
- The recursive running time of the SQUARE-MATRIX-MULTIPLY-RECURSIVE algorithm is

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 8T\left(\frac{n}{2}\right) + \Theta(n^2) & \text{if } n > 1 \end{cases}$$

- The function $T(n) = \Theta(n^3)$.

STRASSEN'S ALGORITHM SQUARE-MATRIX-MULTIPLICATION

- A **divide**-and-**conquer** approach
- Trade one submatrix multiplication for several new additions.

STRASSEN'S ALGORITHM
SQUARE-MATRIX-MULTIPLICATION

- **Divide** the input $n{\times}n$ matrices into four $\frac{n}{2}\times\frac{n}{2}$ sub-matrices.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

STRASSEN'S ALGORITHM
SQUARE-MATRIX-MULTIPLICATION

- **Conquer** the subproblems by
  - **Step 1**: Creating 10 more matrices S_i.

$$
\begin{aligned}
S_1 &= B_{12} - B_{22}, & S_6 &= B_{11} + B_{22}, \\
S_2 &= A_{11} + A_{12}, & S_7 &= A_{12} - A_{22}, \\
S_3 &= A_{21} + A_{22}, & S_8 &= B_{21} + B_{22}, \\
S_4 &= B_{21} - B_{11}, & S_9 &= A_{11} - A_{21}, \\
S_5 &= A_{11} + A_{22}, & S_{10} &= B_{11} + B_{12}.
\end{aligned}
$$

    - Each matrix S_i is either a matrix addition or a subtraction on two $\frac{n}{2} \times \frac{n}{2}$ matrices.
    - The running time of this step is $\mathbf{10(n/2)^2 = 10 \cdot (n^2/4\,)}$ and bounded by $\mathbf{\Theta(n^2)}$.

STRASSEN'S ALGORITHM
SQUARE-MATRIX-MULTIPLICATION

- **Conquer** the subproblems by
  - **Step 3**: Adds and subtracts the P_i to construct the four $\frac{n}{2}$-by-$\frac{n}{2}$ submatrices of the product C as shown on the right.

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

    - For **Step 3,** there are __$\mathbf{8}$__ additions/subtractions on $\frac{n}{2}$-by-$\frac{n}{2}$ matrices, with the running time of each addition/subtraction being tightly bounded by __$\boldsymbol{\Theta(n^2)}$__.
    - We can say that the asymptotic tight bound of this step is __$\mathbf{8 \cdot \Theta(n^2) = \Theta(n^2)}$__.

STRASSEN'S ALGORITHM
SQUARE-MATRIX-MULTIPLICATION

- **Combine** $C_{11}, C_{12}, C_{21}, C_{22}$ as one output.

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{n-1} & c_n \\ c_{21} & c_{22} & \cdots & c_{2,n-1} & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n-1,1} & c_{n-1,2} & \cdots & c_{n-1,n-1} & c_{n-1,n} \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n-1} & c_{nn} \end{pmatrix}$$

C_{11} C_{12}

C_{21} C_{22}

STRASSEN'S ALGORITHM
TIME COMPLEXITY

- The recursive running time of Strassen's algorithm
  - Divide: Partitioning. $\Rightarrow \boldsymbol{\Theta(1)}$.
  - Conquer
    - Create 10 matrices S_i by adding/subtracting the submatrices. $\Rightarrow \boldsymbol{\Theta(n^2)}$.
    - Recursively multiply seven times to create matrices P_i. $\Rightarrow \boldsymbol{7T(n/2) + \Theta(n^2)}$.
    - Adding and subtracting P_i to get the submatrices of C. $\Rightarrow \boldsymbol{\Theta(n^2)}$
- Running time $T(n) = \begin{cases} 1, & n = 1 \\ 7T\left(\frac{n}{2}\right) + \Theta(n^2), & n > 1 \end{cases}$. We shall see $T(n) = \Theta\left(n^{\log 7}\right) < \boldsymbol{\Theta(n^3)}$.

NEXT UP
SOLVING RECURRENCE

REFERENCE