1. The recurrence (as usual \( n \) is \( j - i + 1 \)) is

\[
T_{RM}(n) = \begin{cases} 
1 & \text{if } n = 1 \\
T_{RM}(\lfloor \frac{n}{2} \rfloor) + T_{RM}(\lceil \frac{n}{2} \rceil) + 4 & \text{if } n > 1
\end{cases}
\]

The 1 for the case of \( n = 1 \) comes from the observation that the only work done in this case is the comparison of \( i \) and \( j \). The 4 in the recursive case comes from the \( i < j \) test, the assignment to \( m \) (I guess it is debatable how many operations this corresponds to), the test \( \ell < r \), and the subsequent \textbf{return}.

Then using the Master theorem, we have \( k = 0 \) and \( c = 1 \). Hence running time is \( \Theta(n) \).

2. (a) We prove \( T(\hat{n}) = (\hat{n})^2(1 + \lg(\hat{n})) \) for all powers-of-2 by induction.

\textit{base case:} \( p = 0 \) and \( n = 1 \). Then \( T(1) = 1 \) by definition. Also \( 1^2(1 + \lg(1)) = 1^2(1 + 0) = 1 \). So true.

\textit{Induction hypothesis (IH):} \( T(\hat{n}) = (\hat{n})^2(1 + \lg(\hat{n})) \) for \( \hat{n} = 1, \ldots, 2^p \).

\textit{Induction step:} We must prove that under the (IH), that the claim also holds for \( \hat{n} = 2^{p+1} \).

We have \( p + 1 \geq 1 \), so \( 2^{p+1} \geq 2 \), so we can apply the recurrence to get

\[
T(2^{p+1}) = 4T([2^{p+1}/2]) + 2^{2(p+1)}
\]

\[
= 4T(2^p) + 2^{2(p+1)} \quad \text{(because } 2^{p+1}/2 = 2^p \in \mathbb{N})
\]

\[
= 4(2^p)^2(1 + \lg(2^p)) + 2^{2(p+1)} \quad \text{(by (IH))}
\]

\[
= 2^{2p+2}(1 + \lg(2^p)) + 2^{2(p+1)}
\]

\[
= 2^{2p+2}(1 + \lg(2^{p+1})) + 2^{2(p+1)} \quad \text{(by } \lg(2 \cdot 2^p) = \lg(2) + \lg(2^p) = 1 + \lg(2^p)\text{)}
\]

\[
= 2^{2p+2}(1 + \lg(2^{p+1})),
\]

as required.

Note that the first line is obtained by substituting \( \hat{n} = 2^{p+1} \) into the recurrence; the second line is by observing that \( [\cdot] \) is unnecessary as \( 2^{p+1}/2 = 2^p \) is an integer; the third line is due to substituting the (IH) for \( T(2^p) \), \( 2^p \) being \textit{strictly} smaller than \( 2^{p+1} \); the fourth and fifth lines come from applying multiplication and properties-of-logs directly; and the final line by rearranging terms.
(b) We can just prove $T(n) \leq T(n+1)$ for all $n \in \mathbb{N}$. Then we can use transitivity to observe that $T(j) \leq T(k)$ for all $j < k, j, k \in \mathbb{N}$. There are other ways, e.g. working explicitly with $n$ and $m$, but the (IH) would be slightly messier in wording - e.g., see slide 14 of lectures 2-3: where the (IH) is less tidy.

First we prove the base case.

Base case: $k = 1$. We have $T(1) = 1$; however $T(2) = 4 \cdot T(1) + 2^2 = 8$; clearly $T(1) < T(2)$.

Next we formulate our Induction Hypothesis.

Induction Hypothesis (IH): for every $k, 1 \leq k < n$, we have $T(k) < T(k+1)$.

Induction step: Based on the (IH) for all $k < n$, we will show $T(n) \leq T(n+1)$ also. Note we must have $n \geq 2$ (else we'd be in the base case), so the recursive step of the recurrence applies to both $T(n)$ and also $T(n+1)$. We can write

$$T(n) = 4T\left(\left\lfloor \frac{n}{2} \right\rfloor \right) + n^2$$

$$T(n+1) = 4T\left(\left\lfloor \frac{n+1}{2} \right\rfloor \right) + (n+1)^2$$

Now observe that either

$$\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor \text{ or } \left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor + 1.$$ 

In the first case ($n$ even, $\left\lfloor \frac{n+1}{2} \right\rfloor = \left\lfloor \frac{n}{2} \right\rfloor$), we have $4T(\left\lfloor \frac{n+1}{2} \right\rfloor) = 4T(\left\lfloor \frac{n}{2} \right\rfloor)$.

In the second case ($n$ odd) the (IH) can be applied to $\left\lfloor \frac{n}{2} \right\rfloor$ because $\left\lfloor \frac{n}{2} \right\rfloor \leq n$ (this is true always when $n \geq 2$). Hence the (IH) tells us that $4T(\left\lfloor \frac{n}{2} \right\rfloor) < 4T(\left\lfloor \frac{n+1}{2} \right\rfloor)$.

We get $4T(\left\lfloor \frac{n}{2} \right\rfloor) \leq 4T(\left\lfloor \frac{n+1}{2} \right\rfloor)$ in either case.

Also $n^2 < (n+1)^2$. Combining these two facts, we get that overall $T(n) < T(n+1)$ (i.e., given the (IH), the claim holds for $n$ also)

By induction, we have $T(n) < T(n+1)$ for all $n \in \mathbb{N}$.

(*) Note we really needed a recurrence with $= \text{ and with explicit constants (no O, no } \Theta \text{) to prove the strictly increasing. This is because we substituted the } T \text{ on the right-hand side and the left-hand side of the claim } T(j) < T(k)$.

(c) Now consider an arbitrary $n \in \mathbb{N}$. Let $p$ be the greatest integer such that $2^p \leq n$ (note we are then guaranteed $2^p > n/2$).

By (a), $T(2^p) = (2^p)^2(1 + \log(2^p))$. By (b), we know that $T(n) \geq T(2^p)$.

By above $2^p > n/2$. Hence we have

$$T(n) \geq T(2^p) = (2^p)^2(1 + \log(2^p))$$

$$> (n/2)^2(1 + \log(n/2))$$

$$= (n^2/4)(\log(n)).$$
This gives $\Omega(n^2 \lg(n))$ for $n_0 = 1$ and $c = 1/4$.

3. Use Strassen’s algorithm to compute the matrix product

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}.$$ 

Just set up the P1-P7 equations on the board, multiply them out, then evaluate $C_{11}, C_{12}, C_{21}, C_{22}$. You’ll need to have lecture 4 (or the book) along with you.

4. Describe an algorithm for efficiently multiplying a $(p \times q)$ matrix with a $(q \times r)$ matrix, where $p, q, r$ are arbitrary positive integers. The running time should be $\Theta(n^{\lg(7)})$, where $n = \max\{p, q, r\}$.

**Answer:**
Let $A$ be the $p \times q$ matrix, and $B$ be the $q \times r$ matrix. We round up the matrices to become $n \times n$ matrices $A', B'$, keeping $A$ in the top lhs of $A'$ (and similarly $B$ in the top lhs of $B'$). All the entries outside the top-left $p \times q$ of $A'$ are 0 and similarly for entries outside the top-left $q \times r$ of $B'$.

We call Strassen($A', B'$) and then extract the top-left hand $p \times r$ matrix.

For this algorithm it’s clear that the runtime is $\Theta(n^{\lg(7)})$ (because that is the running time of Strassen on $n \times n$ matrices, and because the “extra work” in mapping to-and-from $n \times n$ matrices is only $O(n^2)$).

**Observation:** A tangential issue wrt this algorithm is that for this general “rectangular” case it is NOT clear that this “reduce to Strassen” algorithm is often a good strategy. Suppose wlog that $p = \max\{p, q, r\}$. Then the naive matrix multiplication algorithm is $\Theta(pqr)$. Our asymptotic running-time from “reduce to Strassen” is only better if $qr \geq p^{\lg(7)} - 1 \sim p^{1.8}$, which is not necessarily the case in the “rectangular” setting.