Aditya Shirode, Rhythm Shah, Shrenuj Gandhi

5. [5 points] *Applying analysis of divide and conquer to a concrete problem.*

The dominant operations in Strassen's matrix multiplication algorithm, particularly if the numbers being manipulated are multiple precision (take up more than one machine word), are scalar multiplications and scalar additions. Assuming that multiplication takes twice as long as addition and using addition as a unit of time, a recurrence for time is

$$\begin{aligned} T(n) &= 7T(n/2) + 18(n/2)^2 && \text{note: no scalar multiplications are done} \\ & && \text{before or after any recursive calls} \\ T(1) &= 2 && \text{accounts for all scalar multiplications} \end{aligned}$$

Strassen's algorithm, even if you ignore bookkeeping overhead, is not a good choice for small matrices.

Your job is to figure out the exact value of n at which it makes sense to stop recursing and use the ordinary matrix multiplication algorithm instead. The number of scalar operations for the ordinary algorithm is n^3 multiplications and n^3 additions. Once you've set up an equation to solve, you can use a sophisticated calculator, MatLab, or any program that works (I wrote a simple Python program.)

Answer:

To determine the value of n (order of matrix) after which we should use ordinary algorithm, we compare the running time of the two algorithms.
This inequality would be of the form T'(n) ≤ T(n),
where T(n) is running time of Strassen's algorithm,
and T'(n) is running time of ordinary matrix multiplication .
T(n) i.e. running time of Strassen's algorithm can be calculated using tree/levels method.

Assume $n = 2^k$.

Level	instances	instance size	cost/instance	total cost
0	1	n	$18(\frac{n}{2})^2$	$18(\frac{n}{2})^2$
1	7	$n/2$	$18(\frac{n}{4})^2$	$7 * 18(\frac{n}{4})^2$
2	7^2	$n/4$	$18(\frac{n}{8})^2$	$7^2 * 18(\frac{n}{8})^2$
.	.	.	.	.
i	7^i	$n/2^i$	$18(\frac{n}{2^{i+1}})^2$	$7^i * 18(\frac{n}{2^{i+1}})^2$
⋮	⋮	⋮	⋮	⋮
k	7^k	1	2	$2 * 7^k$

$$\begin{aligned} \text{T(n)} &= 2 * 7^k + \sum_{i=0}^{k-1} 7^i * 18(\tfrac{n}{2^{i+1}})^2 \\ &= 2 * 7^{lgn} + 18n^2 \sum_{i=0}^{k-1} 7^i(\tfrac{1}{2^{2i+2}}) && \text{as } k = lgn \\ &= 2 * 7^{lgn} + 18n^2 \sum_{i=0}^{k-1} 7^i(\tfrac{1}{4*4^i}) && \text{as } 2^{a+b} = 2^a * 2^b \\ &= 2n^{lg7} + \tfrac{18n^2}{4} \sum_{i=0}^{k-1} (\tfrac{7}{4})^i && \text{as } a^{log_b n} = n^{log_b a} \\ &= 2n^{lg7} + \tfrac{18n^2}{4} \sum_{i=0}^{k-1} (\tfrac{7}{4})^i && \text{as } a^{log_b n} = n^{log_b a} \end{aligned}$$

$= 2n^{lg7} + \frac{18n^2}{4} * (\frac{(\frac{7}{4})^k - 1}{(\frac{7}{4}) - 1}$ as $\sum x^i = \frac{x^k - 1}{x - 1}$

$= 2n^{lg7} + 6n^2(\frac{7}{4})^k - 6n^2$

$= 2n^{lg7} + 6n^2(\frac{7}{4})^l gn - 6n^2$ as $k = lgn$

$= 2n^{lg7} + 6n^2(n^{lg\frac{7}{4}}) - 6n^2$ as $a^{log_b n} = n^{log_b a}$

$= 2n^{lg7} + 6n^2(n^{lg7-2}) - 6n^2$ as $lg(a\underline{)} = lg(a) - lg(b)$

$= 2n^{lg7} + \frac{6n^2(n^{lg7})}{n^2} - 6n^2$

$= 8n^{lg7} - 6n^2$

T'(n) i.e. running time for ordinary matrix multiplication can be calculated as follows:
T'(n) = $2n^3 + n^3 = 3n^3$

Substituting the values in the inequality, we get
$3n^3 \geq 8n^{lg7} - 6n^2$
=> $n \geq$ 151.942 (approx value) (used Wolfram Alpha Engine)
But our assumption in calculating T(n) is that $n = 2^k$
=> $n \geq 256$

Hence at n=256, we should stop recursing and use ordinary matrix multiplication.