The n-th Fibonacci number can be computed by divide and conquer method of computing x^n, where x is the matrix $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$.

Then the n^2-th Fibonacci number F_{n^2} can be computed in $O(\log n)$ time.

T F

答案正确：2 分 创建提问

For the recurrence equation $T(N) = 9T(N/3) + N^2 logN$, we obtain $T(N) = O(N^2 logN)$ according to the Master Theorem.

T F

答案正确：2 分 创建提问

Givien two $n \times n$ matrices A and B, the time complexity of the simple matrix multiplication $C = A \cdot B$ is $O(n^3)$. Now let's consider the following Divide and Conquer idea:

Divide each matrix into four $\frac{n}{2} \times \frac{n}{2}$ submatrics as follows:

$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \cdot \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$

We recursively calculate each block of C as $C_1 = A_1 \cdot B_1 + A_2 \cdot B_3$ and so on. This can reduce the time complexity of the simple calculation.

Givien two $n \times n$ matrices A and B, the time complexity of the simple matrix multiplication $C = A \cdot B$ is $O(n^3)$. Now let's consider the following Divide and Conquer idea:

Divide each matrix into four $\frac{n}{2} \times \frac{n}{2}$ submatrics as follows:

$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \cdot \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$

We recursively calculate each block of C as $C_1 = A_1 \cdot B_1 + A_2 \cdot B_3$ and so on. This can reduce the time complexity of the simple calculation.

F

Which of the asymptotic upper bound for the following recursive $T(n)$ is correct?

A. $T(n) = 2T(n/2) + n\log^2 n$. Then $T(n) = O(n\log^2 n)$.

B. $T(n) = T(n^{1/3}) + T(n^{2/3}) + \log n$. Then $T(n) = O(\log n \log\log n)$

C. $T(n) = 3T(n/2) + n$. Then $T(n) = O(n)$.

D. $T(n) = 2T(\sqrt{n}) + \log n$. Then $T(n) = O(\log n)$.

答案正确：3 分 创建提问

To solve a problem with input size N by divide and conquer, algorithm A divides the problem into 6 subproblems with size $N/2$ and the time recurrences is $T(N) = 6T(N/2) + \Theta(N^2)$.

Now we attempt to design another algorithm B dividing the problem into a subproblems with size $N/4$ and the time recurrences is $T(N) = aT(N/4) + \Theta(N^2)$.

In order to beat algorithm A, what is the largest integer value of a for which algorithm B would be asymptotically faster than algorithm A?

A. 12

B. 18

C. 24

D. 36

答案正确：3 分 创建提问

How many of the following sorting methods use(s) Divide and Conquer algorithm?

- Heap Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Selection Sort
- Shell Sort

A. 2

B. 3

C. 4

D. 5

答案正确：3 分　创建提问

\

Givien two $n \times n$ matrices A and B. Let's consider the following Divide and Conquer idea to do matrix multiplication $C = A \cdot B$.

Divide each matrix into four $\frac{n}{2} \times \frac{n}{2}$ submatrics as follows:

$$\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \cdot \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$$

We define $P_1, P_2, \cdots, P_7$ as follows:

$P_1 = A_1 \cdot (B_2 - B_4)$

$P_2 = (A_1 + A_2) \cdot B_4$

$P_3 = (A_3 + A_4) \cdot B_1$

$P_4 = A_4 \cdot (B_3 - B_1)$

$P_5 = (A_1 + A_4) \cdot (B_1 + B_4)$

$P_6 = (A_2 - A_4) \cdot (B_3 + B_4)$

$P_7 = (A_1 - A_3) \cdot (B_1 + B_2)$

Here all the matrix multiplications are done **recursively**. Then each part of C can be calculated by simple additions and subtractions among $P_1, P_2, \cdot \cdot$

Which of the following is the closest to the actual time complexity?

A. $O(n^2 \log_2 n)$

B. $O(n^e)$

C. $O(n^{\log_2 7})$

D. $O(n^3)$

答案错误：0 分　创建提问