Show your work! Give the best result that you can give!

1. Suppose you have 3 algorithms $A_1$, $A_2$, $A_3$ all for the same problem. Assume, all input can be divided into 4 groups, say, $X_1, X_2, X_3, X_4$ and you happen to know performance, in terms of running time, of these algorithms over all possible inputs of size $n$. for each group.

<table>
<thead>
<tr>
<th>Group</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$5n^4 + n^3$</td>
<td>$200n^4 + n \log n$</td>
<td>$0.1n^3 + 2^n$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$100n^4$</td>
<td>$0.001n^2$</td>
<td>$n^2 + 90000n + 5n \log n$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$0.05n^3 \log n$</td>
<td>$100n^4 \log n$</td>
<td>$125n^2 + n^3 \log n$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$n^5 + 10^{10} \log n$</td>
<td>$500n^2 + 100n \log n$</td>
<td>$n \log n + 0.01n^4$</td>
</tr>
</tbody>
</table>

Give performance of $A_1$, $A_2$, $A_3$ in terms of $O()$, $\Omega()$, $\Theta()$ notation. 3 pts

<table>
<thead>
<tr>
<th>Group</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>$n^4$</td>
<td>$n^4$</td>
<td>$2^n$</td>
<td>$n^5$</td>
</tr>
<tr>
<td>$A_2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>$A_3$</td>
<td>$n^4 \log n$</td>
<td>$n^4 \log n$</td>
<td>$n^4 \log n$</td>
<td>$n^4 \log n$</td>
</tr>
</tbody>
</table>

2. Assume $X$ is an algebraic object whose multiplication is relatively expensive. We want to compute $X^n$ for various $n$.

- Assume you can store as many $X^j$ as you wish, how would you compute $X^n$ for $n=71, 127, 512$. Indicate which $X^j$ you store and how do you compute desired $X^n$'s? (Show which multiplications) 3 pts

$n=71 = 64 + 4 + 1 = 2^6 + 2^2 + 1 \implies X^{67} = X^{64} * X^4 * X^2 * X$ Thus you need 6 +3 multiplication

$n=127 = 64 + 32 + 16 + 8 + 4 + 2 + 1 \implies x^{126} = X^{64} * X^{32} * X^{16} * X^8 * X^4 * X^2 * X$. Thus you need 6+6=12 multiplications.

$n=512 = 2^9$. You need 9 multiplications of series $X^{2^i} = X^{2^{i-1}} * X^{2^{i-1}}$ for $i=1, 2, k=9$
• **B** Given \(b(0), b(1), b(k)\), binary expansion of \(n\), with \(b(k)=1\), how would you compute \(X^n\) without storing all intermediate results. Write an algorithm/pseudo-code!  

Let \(P\) be product \(X^n\), and \(T\) be the term \(X^{2^i}\). The following algorithm will compute \(X^n\)

\[
P=1 \quad T=X
\]

\[
\text{for } i=0 \text{ to } k
\]
\[
\text{if } b(i)=1 \text{ then } P=P \times T \fi
\]
\[
T = T \times T
\]
\[\text{endfor}\]

3. **Fibonacci**

Given \(F(n+2) = F(n+1) + F(n)\) with \(F(0) = 0, F(1) = 1\) you can compute a) as a recursion function, b) as an array using \(n\) addition, c) using matrix multiplication via

\[
A^1 = A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad A^n = \begin{bmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{bmatrix}
\]

Assume multiplication of two 2x2 matrix costs \(\alpha\) additions!  

• Assume \(n = 2^k\) and \(\alpha = 2^t\). Compute cost of \(F(n)\) using b) and c) in terms of additions. Write an equation/inequality to determine when matrix multiplication becomes cheaper.

we can compute \(F(n)\) via b) in \(n\) additions and via matrix multiplication \(k\alpha\) for \(n = 2^k\).

Thus we look for smallest \(k\) such that

\[
k\alpha \leq n \quad \text{OR} \quad 2^k \times 2^t \leq n
\]

• For \(k = 2^t\) determine first \(l\) such that matrix multiplication is cheaper for \(t=10\)

\[
2^{10} \times 2^t \leq 2^{2^t}
\]

clearly for \(l=3\) the above inequality is not satisfied but for \(l=4\) it is satisfied for \(2^{10} \times 2^4 = 2^{14} < 2^{16}\)

• For \(\alpha = 100\) compute cost of computing \(F(n)\) for \(n=1536\) using b) and c)

\(n=1536 = 1024 + 512 = 2^{10} + 2^9\). We compute \(A^n = A^{1024} \times A^{512} = \) via 10+1 matrix multiplication.

From \(A^n\) we read \(F(n)\). Thus \(11\alpha = 1100\) additions will suffice to compute \(F(n)\) via c).