Algorithm A takes $\log_2 n$ time. On one input, A takes $x$ time. How long will it take if I double the input size?

$x + 1$  \[ \square \] \quad 2x  \quad \square \quad 2^x  \quad \square \quad x^2  \quad \square

\[
T(1) = c \\
T(n) = 4T(n/2) + n
\]

$\Theta(n)$ \[ \square \]  $\Theta(n \log n)$ \[ \square \]  $\Theta(n^2)$ \[ \square \]  $\Theta(n^3)$ \[ \square \]  $\Theta(n^{\log_2 3})$ \[ \square \]  $\Theta(n^{\log_3 2})$ \[ \square \]  $\Theta(2^n)$ \[ \square \]  $\Theta(3^n)$ \[ \square \]

The running time of binary search is $O(n \log n)$.  \quad true  \[ \square \]  false  \[ \square \]

For a problem to satisfy the definition of NP, a "yes" answer must have a succinct justification.  \quad true  \[ \square \]  false  \[ \square \]

Deciding whether an input logic expression be made true by appropriate choice of input values.  \quad polynomial  \[ \square \]  exponential  \[ \square \]  in NP  \[ \square \]
(15 points) Check the (single) box that best characterizes each item.

Karatsuba’s integer multiplication algorithm

- Θ(\log n)
- Θ(n)
- Θ(n \log n)
- Θ(n^2)
- Θ(n^3)
- Θ(n^{\log_2 2})
- Θ(n^{\log_2 3})
- Θ(2^n)

\[ T(1) = d \]
\[ T(n) = 2T(n/4) + n \]

- Θ(\log n)
- Θ(\sqrt{n})
- Θ(n)
- Θ(n \log n)
- Θ(n^2)
- Θ(2^n)
- Θ(3^n)

The running time of merge is recursively defined by

\[ T(1) = d \]
\[ T(n) = 2T(n-1) + c \]

- \( T(n - 1) + c \)
- \( T(n - 1) + cn \)
- \( 2T(n - 1) + c \)
- \( 2T(n - 1) + cn \)

Circuit satisfiability can be solved in polynomial time.

- true
- false
- not known

For a problem to satisfy the definition of co-NP, a “no” answer must have a succinct justification.

- true
- false
(15 points) Check the (single) box that best characterizes each item.

<table>
<thead>
<tr>
<th>The running time of merge</th>
<th>Θ(log n)</th>
<th>Θ(n)</th>
<th>Θ(n log n)</th>
<th>Θ(n²)</th>
<th>Θ(n³)</th>
<th>Θ(n³ log 2)</th>
<th>Θ(n³ log 3)</th>
<th>Θ(n³ log 3 log 2)</th>
<th>Θ(2^n)</th>
<th>Θ(3^n)</th>
<th>Θ(2n³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(1) = d</td>
<td>Θ(n)</td>
<td>Θ(n log n)</td>
<td>Θ(n²)</td>
<td>Θ(n³)</td>
<td>Θ(n³ log 2)</td>
<td>Θ(n³ log 3)</td>
<td>Θ(n³ log 3 log 2)</td>
<td>Θ(2^n)</td>
<td>Θ(3^n)</td>
<td>Θ(2n³)</td>
<td></td>
</tr>
<tr>
<td>T(n) = T(n-1) + n</td>
<td>Θ(n³ log 2)</td>
<td>Θ(n³ log 3)</td>
<td>Θ(2^n)</td>
<td>Θ(3^n)</td>
<td>Θ(2n³)</td>
<td>Θ(3^n)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T(1) = d</td>
<td>Θ(n³ log 2)</td>
<td>Θ(n³ log 3)</td>
<td>Θ(2^n)</td>
<td>Θ(3^n)</td>
<td>Θ(2n³)</td>
<td>Θ(3^n)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The solution to the Tower of Hanoi puzzle with n disks requires Θ(2^n) steps. **true** **false** **not known**

The chromatic number of a graph with n nodes can be found in polynomial time. **true** **false** **not known**
(6 points) Fill in the missing bits of a recursive implementation of Merge, which merges two lists of integers sorted in increasing order. Use the functions first (first element), rest (everything after first element), and and cons (adds number to list).

Merge($L_1, L_2$: sorted lists of real numbers)

if ($L_1$ is empty and $L_2$ is empty) return emptylist
else if ($L_2$ is empty or first($L_1$) $<$ first($L_2$))

else

(9 points) Check the (single) box that best characterizes each item.

$T(1) = d$
$T(n) = 3T(n/3) + c$

<table>
<thead>
<tr>
<th>$\Theta(n)$</th>
<th>$\Theta(n \log n)$</th>
<th>$\Theta(n^2)$</th>
<th>$\Theta(n^3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(n^{\log_2 3})$</td>
<td>$\Theta(n^{\log_3 2})$</td>
<td>$\Theta(2^n)$</td>
<td>$\Theta(3^n)$</td>
</tr>
</tbody>
</table>

The Towers of Hanoi puzzle requires exponential time.

true □  false □  not known □

Finding the chromatic number of a graph with $n$ nodes requires $\Theta(2^n)$ time.

true □  false □  not known □
(15 points) Check the (single) box that best characterizes each item.

The running time of Karatsuba’s algorithm is recursively defined by $T(1) = d$ and $T(n) = 4T(n/2) + cn$  
$2T(n/2) + cn$  
$3T(n/2) + cn$

The running time of the Towers of Hanoi solver is recursively defined by $T(1) = d$ and $T(n) = 2T(n - 1) + c$  
$2T(n/2) + cn$  
$2T(n/2) + cn$  

For a problem to satisfy the definition of co-NP, a “yes” answer must have a succinct justification.  
true  
false

The Towers of Hanoi puzzle can be solved in polynomial time.  
true  
false  
not known
(15 points) Check the (single) box that best characterizes each item.

<table>
<thead>
<tr>
<th>Adding element to start of array (array gets longer)</th>
<th>( \Theta(1) )</th>
<th>( \Theta(\log n) )</th>
<th>( \Theta(n) )</th>
<th>( \Theta(n \log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Theta(n^2) )</td>
<td>( \Theta(n^3) )</td>
<td>( \Theta(2^n) )</td>
<td>( \Theta(3^n) )</td>
<td></td>
</tr>
</tbody>
</table>

| \( T(1) = d \) | \( T(n) = 3T(n/2) + d \) | \( \Theta(n) \) | \( \Theta(n \log n) \) | \( \Theta(n^2) \) | \( \Theta(n^3) \) |
|---|---|---|---|---|
| \( \Theta(n \log n) \) | \( \Theta(n \log^3 n) \) | \( \Theta(2^n) \) | \( \Theta(3^n) \) |

| \( T(1) = c \) | \( T(n) = 2T(n/2) + n^2 \) | \( \Theta(n) \) | \( \Theta(n \log n) \) | \( \Theta(n^2) \) | \( \Theta(n^3) \) |
|---|---|---|---|---|
| \( \Theta(n \log n) \) | \( \Theta(n \log^3 n) \) | \( \Theta(2^n) \) | \( \Theta(3^n) \) |

Problems in class NP require exponential time.  

- true  
- false  
- not known

The Marker Making problem can be solved in polynomial time.  

- true  
- false  
- not known
(15 points) Check the (single) box that best characterizes each item.

<table>
<thead>
<tr>
<th>Item</th>
<th>$T(1) = d$</th>
<th>$T(n) = T(n/3) + c$</th>
<th>$T(n) = T(n/2) + c$</th>
<th>$2T(n/2) + cn$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dividing a linked list in half</td>
<td>$\Theta(1)$</td>
<td>$\Theta(n^2)$</td>
<td>$\Theta(\log n)$</td>
<td>$\Theta(n)$</td>
</tr>
<tr>
<td>The running time of the Towers of Hanoi solver</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Producing all parses for a sentence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Travelling Salesman Problem</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(15 points) Check the (single) box that best characterizes each item.

<table>
<thead>
<tr>
<th>Item</th>
<th>( T(1) = d )</th>
<th>( T(n) = T(n/2) + n )</th>
<th>( \Theta(\log n) )</th>
<th>( \Theta(\sqrt{n}) )</th>
<th>( \Theta(n) )</th>
<th>( \Theta(n \log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(1) = d )</td>
<td></td>
<td></td>
<td>( \Theta(n^2) )</td>
<td>( \Theta(n^3) )</td>
<td>( \Theta(2^n) )</td>
<td>( \Theta(3^n) )</td>
</tr>
<tr>
<td>( T(n) = T(n/2) + n )</td>
<td></td>
<td></td>
<td>( \Theta(\log n) )</td>
<td>( \Theta(\sqrt{n}) )</td>
<td>( \Theta(n \log n) )</td>
<td>( \Theta(n^2) )</td>
</tr>
<tr>
<td>Algorithm A takes ( 2^n ) time. On one input, A takes ( x ) time. How long will it take if I add one to the input size?</td>
<td></td>
<td>( x + 2 )</td>
<td>( 2x )</td>
<td>( 2^x )</td>
<td>( x^2 )</td>
<td></td>
</tr>
</tbody>
</table>

Problems in class P (as in P vs. NP) require exponential time

<table>
<thead>
<tr>
<th>Item</th>
<th>( T(1) = d )</th>
<th>( T(n) = 3T(n/2) + n )</th>
<th>( \Theta(n) )</th>
<th>( \Theta(n \log n) )</th>
<th>( \Theta(n^2) )</th>
<th>( \Theta(n^3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(1) = d )</td>
<td></td>
<td></td>
<td>( \Theta(n) )</td>
<td>( \Theta(n \log n) )</td>
<td>( \Theta(n^2) )</td>
<td>( \Theta(n^3) )</td>
</tr>
<tr>
<td>( T(n) = 3T(n/2) + n )</td>
<td></td>
<td></td>
<td>( \Theta(n \log^2 3) )</td>
<td>( \Theta(n \log^2 3) )</td>
<td>( \Theta(2^n) )</td>
<td>( \Theta(3^n) )</td>
</tr>
</tbody>
</table>

The Travelling Salesman problem can be solved in polynomial time.

<table>
<thead>
<tr>
<th>Item</th>
<th>( T(1) = d )</th>
<th>( T(n) = 3T(n/2) + n )</th>
<th>( \Theta(n) )</th>
<th>( \Theta(n \log n) )</th>
<th>( \Theta(n^2) )</th>
<th>( \Theta(n^3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects and unique problems in class P (as in P vs. NP) require exponential time</td>
<td></td>
<td></td>
<td>( \Theta(n) )</td>
<td>( \Theta(n \log n) )</td>
<td>( \Theta(n^2) )</td>
<td>( \Theta(n^3) )</td>
</tr>
</tbody>
</table>