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

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?

$$x + 1 \quad \sqrt{\ } \quad 2x \quad \square \quad 2^x \quad \square \quad x^2 \quad \square$$

$$T(1) = c$$
$$T(n) = 4T(n/2) + n$$

$$\Theta(n) \quad \square \quad \Theta(n \log n) \quad \square \quad \Theta(n^2) \quad \sqrt{\ } \quad \Theta(n^3) \quad \square$$

$$\Theta(n^{\log_2 2}) \quad \square \quad \Theta(n^{\log_2 3}) \quad \square \quad \Theta(2^n) \quad \square \quad \Theta(3^n) \quad \square$$

The running time of binary search is $O(n \log n)$.  

true $\sqrt{\ }$  false $\square$

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

true $\sqrt{\ }$  false $\square$

Deciding whether an input logic expression be made true by appropriate choice of input values.

polynomial $\square$  exponential $\square$  in NP $\sqrt{\ }$
(15 points) Check the (single) box that best characterizes each item.

**Karatsuba’s integer multiplication algorithm**

\[
\begin{array}{cccc}
\Theta(\log n) & \Theta(n) & \Theta(n \log n) & \Theta(n^2) \\
\Theta(n^3) & \Theta(n^{\log_3 2}) & \Theta(n^{\log_2 3}) & \sqrt{} \\
\end{array}
\]

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

\[
\begin{array}{cccc}
\Theta(\log n) & \Theta(\sqrt{n}) & \Theta(n) & \sqrt{} \\
\Theta(n^2) & \Theta(n^3) & \Theta(2^n) & \Theta(3^n) \\
\end{array}
\]

The running time of merge is recursively defined by

\[
T(1) = d \quad \text{and} \quad T(n) =
\]

\[
\begin{array}{cccc}
T(n-1) + c & \sqrt{} & T(n-1) + cn & \\
2T(n-1) + c & \sqrt{} & 2T(n-1) + cn & \\
\end{array}
\]

Circuit satisfiability can be solved in polynomial time.

\[
\begin{array}{cccc}
\text{true} & \text{false} & \text{not known} & \sqrt{} \\
\end{array}
\]

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

\[
\begin{array}{cccc}
\text{true} & \sqrt{} & \text{false} & \\
\end{array}
\]
(15 points) Check the (single) box that best characterizes each item.

The running time of merge:

- $\Theta(\log n)$
- $\Theta(n)$
- $\sqrt{\Theta}$
- $\Theta(n \log n)$
- $\Theta(n^2)$
- $\Theta(n^3)$
- $\Theta(n^{\log_3 2})$
- $\Theta(n^{\log_2 3})$
- $\Theta(2^n)$
- $\Theta(3^n)$
- $\Theta(2^n \log 3)$
- $\Theta(2^n \log 2)$
- $\Theta(2^n)$
- $\Theta(3^n)$

$T(1) = d$
$T(n) = T(n-1) + n$

- $\Theta(n)$
- $\Theta(n \log n)$
- $\Theta(n^2)$
- $\sqrt{\Theta}$
- $\Theta(n^3)$
- $\Theta(n^{\log_3 2})$
- $\Theta(n^{\log_2 3})$
- $\Theta(2^n)$
- $\Theta(3^n)$

$T(1) = d$
$T(n) = 2T(n/3) + d$

- $\Theta(n)$
- $\Theta(n \log n)$
- $\Theta(n^2)$
- $\sqrt{\Theta}$
- $\Theta(n^3)$
- $\Theta(n^{\log_3 2})$
- $\Theta(n^{\log_2 3})$
- $\Theta(2^n)$
- $\Theta(3^n)$

The solution to the Tower of Hanoi puzzle with $n$ disks requires $\Theta(2^n)$ 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($L_2$))

Solution: return cons(first($L_1$), merge(rest($L_1$), $L_2$))

else

Solution: return cons(first($L_2$), merge($L_1$, rest($L_2$)))

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

$T(1) = d$  \hspace{1cm} $T(n) = 3T(n/3) + c$

$\Theta(n)$  \hspace{1cm} $\Theta(n \log n)$  \hspace{1cm} $\Theta(n^2)$  \hspace{1cm} $\Theta(n^3)$  \hspace{1cm} $\Theta(n \log_2 n)$  \hspace{1cm} $\Theta(n \log_2 3)$  \hspace{1cm} $\Theta(2^n)$  \hspace{1cm} $\Theta(3^n)$

The Towers of Hanoi puzzle requires exponential time. \hspace{1cm} true  \hspace{1cm} false  \hspace{1cm} not known

Finding the chromatic number of a graph with $n$ nodes requires $\Theta(2^n)$ time. \hspace{1cm} true  \hspace{1cm} false  \hspace{1cm} 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 $T(n) =$

- $4T(n/2) + cn$ false
- $2T(n/2) + cn$ true
- $3T(n/2) + cn$ false

$T(1) = d$
$T(n) = 2T(n - 1) + c$
$\Theta(n)$ false
$\Theta(n \log n)$ false
$\Theta(n^{\log_3 2})$ false
$\Theta(n \log_2 3)$ false
$\Theta(n^2)$ false
$\Theta(2^n)$ true
$\Theta(3^n)$ false

The running time of the Towers of Hanoi solver is recursively defined by $T(1) = d$ and $T(n) =$

- $2T(n - 1) + c$ true
- $2T(n/2) + c$ false
- $2T(n/2) + cn$ false

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></td>
<td>(\Theta(n^3))</td>
<td>(\Theta(2^n))</td>
<td>(\Theta(3^n))</td>
</tr>
</tbody>
</table>

\[
T(1) = d \\
T(n) = 3T(n/2) + d
\]

<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_3 2}))</td>
<td>(\Theta(n^{\log_2 3}))</td>
<td>(\Theta(2^n))</td>
<td>(\Theta(3^n))</td>
</tr>
</tbody>
</table>

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

<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_3 2}))</td>
<td>(\Theta(n^{\log_2 3}))</td>
<td>(\Theta(2^n))</td>
<td>(\Theta(3^n))</td>
</tr>
</tbody>
</table>

Problems in class NP require exponential time

true \[\square\]  false \[\square\]  not known \[\sqrt{\square}\]

The Marker Making problem can be solved in polynomial time.

true \[\square\]  false \[\square\]  not known \[\sqrt{\square}\]
(15 points) Check the (single) box that best characterizes each item.

\[ T(1) = d \]
\[ T(n) = T(n/3) + c \]

- \( \Theta(n^2) \)
- \( \Theta(n^3) \)
- \( \Theta(2^n) \)
- \( \Theta(3^n) \)

Dividing a linked list in half

- \( \Theta(1) \)
- \( \Theta(n) \)
- \( \sqrt{n} \)
- \( \Theta(n \log n) \)

The running time of the Towers of Hanoi solver is recursively defined by

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

Producing all parses for a sentence.

- polynomial
- exponential
- in NP

The Travelling Salesman Problem

- polynomial
- exponential
- in NP
(15 points) Check the (single) box that best characterizes each item.

\[
\begin{array}{c}
T(1) = d \\
T(n) = T(n/2) + n
\end{array}
\]

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

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?

- \(x + 2\)  
- \(2x\)  
- \(2^x\)  
- \(x^2\)  

\[
\begin{array}{c}
T(1) = d \\
T(n) = 3T(n/2) + n
\end{array}
\]

- \(\Theta(n)\)  
- \(\Theta(n \log n)\)  
- \(\Theta(n^2)\)  
- \(\Theta(n^3)\)  
- \(\Theta(n \log_2 n)\)  
- \(\Theta(n \log_3 n)\)  
- \(\Theta(2^n)\)  
- \(\Theta(3^n)\)  

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

- true  
- false  
- not known  

The Travelling Salesman problem can be solved in polynomial time.

- true  
- false  
- not known