1. (12 points) For each of the following statements, determine whether it is correct or not. You should explain why for your answers (True or False).

(1). \[ n^{1/\log n} = O(4^{\log n}). \] \[ T, \quad n^{1/\log n} = 2 = \Theta(1) = O(4^{\log n}) = O(n^2) \]

(2). Sorting 6 elements with a comparison sort requires at least 15 comparisons in the average case. (\log 6 = 2.5) \[ F, \quad 2^9 < 6! < 2^{10}, \text{ so at least 10 comparisons in the worst case.} \rightarrow \text{average case must be equal or smaller} \]

(3). A set of n integers in the range \( \{1, 2, \ldots, n\} \) can be sorted by RADIX-SORT in \( O(n) \) time by running QUICK-SORT on each bit of the binary representation. \[ F, \quad \text{QUICK-SORT is not a stable sorting algorithm or its running time is } \Theta(n \log n). \]

2. (16 points) Student A has implemented an algorithm with time complexity \( O(2^n) \) in the text book. However, when he analyzed and measured his program, he got the time function \( f(n) \) is \( \left[ \frac{\log \log n}{n} \right] \). What happened? Please explain your answer.

Ans: \( \left[ \log \log n \right] \) ! = \( O(n^k) \), i.e., polynomially bounded. Some points could be described (1) \( O(2^n) \) is not the tightest bound for this algorithm, (2) the measured \( n \) is not big enough, (3) …

3. (9 points) Please write three recurrence functions which could be solved by Master method and they also match 3 different cases in Master method respectively. Finally, solve them.

Ans: (don’t forget the extra testing for case 3)

4. (5 points) Please explain, in your words, why the complexity of building a max heap is \( O(n) \) while we apply a \( O(\text{heap height}) \) time complexity Max-Heapify algorithm to build it?

Ans: Max heap is a kind of binary search tree. It cost \( O(\log n) \) to apply Max-Heapify algorithm. There are \( n \) element to build max-heap. Therefore, the time complexity is : \( O(n \log n) \). But it doesn’t tight enough.

The number of nodes at height \( h \leq \frac{n}{2^{h+1}} \). Therefore, the total cost of heapify is

\[
\sum_{h=0}^{\left\lfloor \log n \right\rfloor} \frac{n}{2^{h+1}} = O(n \sum_{h=0}^{\left\lfloor \log n \right\rfloor} \frac{1}{2^h}) = O(n \sum_{h=0}^{\infty} \frac{1}{2^h}) = O(n)
\]

5. (15 points) Write an algorithm to find the predecessor of an input node \( x \) in a binary search tree. (Note: you can use TREE_MINIMUM(\( x \)) and TREE_MAXIMUM(\( x \)) directly.)

Ans:

Tree_predecessor
if \( \text{left}[x] \) \( \neq \) null
    then return \( \text{TREE_MAXIMUM(left[x])} \)

\( y \leftarrow \text{parent}[x] \)
while \( y \) \( \neq \) null and \( x=\text{left}[y] \)
    do \( x \leftarrow y \)
    \( y \leftarrow \text{parent}[y] \)
return \( y \)
6. (19 points) (1) (10 points) Please describe the optimal substructures for the COIN-CHANGE problem. Please also calculate how many different subproblems (2 points) overall and how many choices (2 points) we have in determining which subproblem(s) to use in an optimal solution. The COIN-CHANGE problem: An amount of money $M$, and an array of $d$ denominations $c = (c_1, c_2, ..., c_d)$, in a decreasing order of value ($c_1 > c_2 > ... > c_d$). Please find a list of $d$ integers $i_1, i_2, ..., i_d$ such that $c_1i_1 + c_2i_2 + ... + c_di_d = M$ and $i_1 + i_2 + ... + i_d$ is minimal. (2) (5 points) Show your answer to find the optimal solution for the COIN-CHANGE problem with $c = (1, 3, 7)$ and $M = 9$.

\[
\text{minNumCoins}(M) = \min \begin{cases} 
\text{minNumCoins}(M-c_1) + 1 \\
\text{minNumCoins}(M-c_2) + 1 \\
\vdots \\
\text{minNumCoins}(M-c_d) + 1 
\end{cases}
\]

**Ans:** (1) #subproblems = O(M), #choice to determining which subproblems = d

In this problem, we need to use Dynamic Programming (DP) algorithm to calculate the minimum changed coins.

\[
\text{minNumCoins}(9) = \begin{cases} 
\text{minNumCoins}(9-7) + 1 \\
\text{minNumCoins}(9-3) + 1 \\
\text{minNumCoins}(9-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(9-7) = \begin{cases} 
\text{minNumCoins}(2-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(2-1) = \begin{cases} 
\text{minNumCoins}(1-1) + 1 \text{ (return, 3 coins is the best.)} 
\end{cases}
\]

\[
\text{minNumCoins}(9-3) = \begin{cases} 
\text{minNumCoins}(6-3) + 1 \\
\text{minNumCoins}(6-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(6-3) = \begin{cases} 
\text{minNumCoins}(3-3) + 1 \text{ (return, 3 coins is the best.)} \\
\text{minNumCoins}(3-1) + 1 \text{ (return, more than 3 coins, not answer.)} 
\end{cases}
\]

\[
\text{minNumCoins}(9-1) = \begin{cases} 
\text{minNumCoins}(8-7) + 1 \text{ (return, 3 coins is the best.)} \\
\text{minNumCoins}(8-3) + 1 \\
\text{minNumCoins}(8-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(8-3) = \begin{cases} 
\text{minNumCoins}(5-3) + 1 \\
\text{minNumCoins}(5-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(5-3) = \begin{cases} 
\text{minNumCoins}(2-1) + 1 \text{ (return, more than 3 coins, not answer.)} 
\end{cases}
\]

\[
\text{minNumCoins}(5-1) = \begin{cases} 
\text{minNumCoins}(4-3) + 1 \text{ (return, more than 3 coins, not answer.)} \\
\text{minNumCoins}(4-1) + 1 
\end{cases}
\]

\[
\text{minNumCoins}(8-1) = \begin{cases} 
\text{minNumCoins}(7-7) + 1 \text{ (return, 3 coins is the best.)} \\
\text{minNumCoins}(7-3) + 1 \text{ (return, more than 3 coins, not answer.)} \\
\text{minNumCoins}(7-1) + 1 \text{ (return, more than 3 coins, not answer.)} 
\end{cases}
\]
So the answer is 3.

Another solution

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Num or coins</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Num or coins</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Num or coins</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|

7. (24 points) Please fill in the following blanks. (2 points for each sub-problems of Bucket Sort and Counting Sort)

<table>
<thead>
<tr>
<th></th>
<th>Merge Sort</th>
<th>Quick Sort</th>
<th>Bucket Sort (auxiliary sorting: insertion sort)</th>
<th>Counting Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort in place (T/F)</td>
<td>(1) F</td>
<td>(5) T</td>
<td>(9) F</td>
<td>(13) F</td>
</tr>
<tr>
<td>stable(T/F)</td>
<td>(2) T</td>
<td>(6) F</td>
<td>(10) T</td>
<td>(14) T</td>
</tr>
<tr>
<td>worst case time</td>
<td>(3) $\Theta(n \lg n)$</td>
<td>(7) $O(n^2)$</td>
<td>(11) $\Theta(n^2)$</td>
<td>(15) $\Theta(k), k=\omega(n)$</td>
</tr>
<tr>
<td>best case time</td>
<td>(4) $\Theta(n \lg n)$</td>
<td>(8) $\Theta(n \lg n)$</td>
<td>(12) $\Theta(n)$</td>
<td>(16) $\Theta(n)$</td>
</tr>
</tbody>
</table>

* You should consider the **tightest asymptotic notations** in the running time analysis.