• Q1(18 points): Asymptotic Notations

(a)(10 points) Try to show $n^2 \frac{2}{3} - 10n - 10 = \Theta(n^2)$ using the basic definition of $\Theta$ notation. That is, to find positive constants $c_1, c_2, n_0$ such that $c_1 n^2 \leq \frac{2}{3} n^2 - 10n - 10 \leq c_2 n^2, \forall n \geq n_0$.

(b)(4 points) Which one of the following is true?

1. $2^{2n} = \Theta(2^n)$
2. $n + \Omega(n) = \Omega(n)$
3. If $f(n) = O(g(n))$ and both $f(n)$ and $g(n)$ are asymptotically positive, then $f(n) + g(n) = O(f(n))$
4. $f(n) = \Omega(g(n))$ implies $f(n) = \omega(g(n))$

(c)(4 points) Which one of the following sorting algorithms is not possible to have the running time recurrence equation $T(n) = T(n - 1) + n$?

1. Insertion sort
2. Selection sort
3. Merge sort
4. Quick sort
Q2 (15 points): Recursive procedures

The Insertion Sort algorithm can be interpreted as follows:

**Divide:** Divide the array $A$ into left and right subarrays, with size $n - 1$ and 1, respectively.

**Conquer:** Recursively sort the left subarray.

**Combine:** Insert the element in the right subarray into the sorted left subarray.

(a) (10 points) Please write a recursive pseudocode for this algorithm.

```plaintext
 Insert-Sort(A, p, r) // A: the input array
                    // p: starting index, r: ending index
{

```

(b) (5 points) Please write down the average case recurrence equation for the Insertion Sort, assuming the insertion process required to search half of the left subarray in average.
• Q3(30 points): Solving Recurrences

(a) (10 points) Try to use the substitution method to show that $n^2$ is $T(n)$’s upper bound for a recurrence $T(n) = T(n/3) + T(2n/3) + 1$. (suppose that $T(1) = 1$)

(b) (10 points) Try to use a recursion tree to solve the recurrence $T(n) = T(n/6) + T(5n/6) + n$ (please derive a tight bound).
(c)(10 points) Please use the Master method to solve the recurrence $T(n) = T(n - 2) + 1$
(please derive the tight bound). You need to transfer the equation to an appropriate
form and then solve it by the Master method.

• Q4(22 points): Sorting

(a)(6 points) For a given input array $A : <6, 4_a, 3, 1, 4_b, 4_c, 9, 7, 5>$, what is the sequence
of numbers in $A$ after calling Build-Max-Heap($A$)?
(b)(6 points) For a given input array \( A : < 6, 4_a, 3, 1, 4_b, 4_c, 9, 7, 5 > \), what is the sequence of numbers in \( A \) after the first partition (by call \( \text{Partition}(A, 1, 9) \))? 

(c)(10 points) For a Max-Heap, if we would like to delete an arbitrary element stored in the heap, please write a procedure for it?

\[
\text{MaxHeap-Delete}(A, i) \quad // A: \text{input array which stores a max heap.} \\
\quad // i: \text{the element stored in } A[i] \text{ will be deleted} \\
\quad // \text{from the heap.}
\]
• **Q5 (15 points): Running Time**

Please indicate the running time of each sorting algorithm in the table below.

<table>
<thead>
<tr>
<th></th>
<th>Bubble-Sort</th>
<th>Insertion-sort</th>
<th>Merge-Sort</th>
<th>Heap-Sort</th>
<th>Quick-Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best-case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average-case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worst-case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **Q6 (extra credit 10 points) Divide and Conquer**

If we would like to multiply two \( n \times n \) matrices \( A \) and \( B \), where \( n \) is power of 2.

A divide-and-conquer technique to solve this problem is as follows:

We can divide each of \( A \) and \( B \) into four \( (n/2) \times (n/2) \) matrices and express the product of \( A \) and \( B \) in terms of these \( (n/2) \times (n/2) \) matrices as:

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

where

\[
C_{11} = A_{11}B_{11} + A_{12}B_{21} \\
C_{12} = A_{11}B_{12} + A_{12}B_{22} \\
C_{21} = A_{21}B_{11} + A_{22}B_{21} \\
C_{22} = A_{21}B_{12} + A_{22}B_{22}
\]

Let \( T(n) \) be the running time for this multiplication algorithm (\( T(n) \) denotes the number of scalar operations). Please write down the recurrence for this matrix-multiplication approach.