Solution #3

Problem 1

Use the $O(n)$-time procedures $Select$ and $Partition$ of Lecture Notes #7 to partition $A$ into two halves. Then, partition each half into two halves. The following is the algorithm.

procedure $FourPartition(A,n)$

\[
\begin{align*}
\text{split}0 &:= Select(A, 1, n, n/2); \\
\text{Partition}(A, 1, n, \text{split}0); \\
\text{split}1 &:= Select(A, 1, n/2, n/4); \\
\text{Partition}(A, 1, n/2, \text{split}1); \\
\text{split}2 &:= Select(A, n/2 + 1, n, n/4); \\
\text{Partition}(A, n/2 + 1, n, \text{split}2)
\end{align*}
\]

Problem 2

We use the $O(n)$-time $Select$ algorithm to find the median of a (sub)array as the pivot element to achieve balanced partitioning. Then, the total time for the $Quicksort$ is

\[ T(n) = 2T\left(\frac{n}{2}\right) + O(n), \]

where $O(n)$ is the time for finding the pivot element and partitioning $n$ elements into two halves, and $T\left(\frac{n}{2}\right)$ is the time for recursively sort one half of the elements. Solving this recurrence relation, we obtain $T(n) = O(n \log n)$ (case 2 of the Master Theorem).

Problem 3

First, we know that

\[ C_{1,2} = A_{1,1} \cdot B_{1,2} + A_{1,2} \cdot B_{2,2} \quad (1) \]

In the Strassen’s Method,
\[ C_{1,2} = M_1 + M_2 + M_5 + M_6 \]  

(2)

where

\[ M_1 = (A_{2,1} + A_{2,2} - A_{1,1}) \cdot (B_{2,2} - B_{1,2} + B_{1,1}) \]
\[ = A_{2,1}B_{2,2} - A_{2,1}B_{1,2} + A_{2,1}B_{1,1} + A_{2,2}B_{2,2} - A_{2,2}B_{1,2} + A_{2,2}B_{1,1} \]
\[ - A_{1,1}B_{2,2} + A_{1,1}B_{1,2} - A_{1,1}B_{1,1} \]

\[ M_2 = A_{1,1}B_{1,1} \]

\[ M_5 = (A_{2,1} + A_{2,2}) \cdot (B_{1,2} - B_{1,1}) \]
\[ = A_{2,1}B_{1,2} - A_{2,1}B_{1,1} + A_{2,2}B_{1,2} - A_{2,2}B_{1,1} \]

\[ M_6 = (A_{1,2} - A_{2,1} + A_{1,1} - A_{2,2}) \cdot B_{2,2} \]
\[ = A_{1,2}B_{2,2} - A_{2,1}B_{2,2} + A_{1,1}B_{2,2} - A_{2,2}B_{2,2} \]

By substituting into Equation (2) and cleaning up, we find that Equations (1) and (2) are the same.

**Brain Teaser Problem**

First, sort the array. Then, for each \( A[i] \), search value \((val - A[i])\) in \( A \).

**procedure** Check\( (A, n, val) \)

\{
    Sort\( (A, n) \); /* sort \( A \) into increasing order */
    for \( i = 1 \) to \( n - 1 \) do
    \{
        \( c := val - A[i] \);
        if Search\( (A, n, c) \) then return true
    \}
    return false
\}

Procedure Search\( (A, n, c) \) searches array \( A \) and returns true if \( c \) is the value of one of the elements from \( A[i+1] \) to \( A[n] \); otherwise, false is returned. Search is carried out by binary search.

Time Complexity: Sorting takes \( O(n \log n) \) time. The for loop has \( n - 1 \) iterations, each taking \( O(\log n) \) time (because of binary search). Thus, the total time is \( O(n \log n) \).