Data Structures and Algorithms — CSCI 230
Chapter 7 — Review Solutions

1. The second version quick sort discussed in class uses the median-of-three technique to find the pivot and does not continue sorting when the length of the sublist (the difference between `Right` and `Left` is less than `Cutoff`). After this algorithm completes, insertion sort is run over the resulting array to bring the values into their final sorted position. What is the worst-case time required by insertion sort in this case? Justify your answer.

   Solution: $O(kn)$, where k is the size of `Cutoff`. The reason is that during each iteration i of insertion sort, at most k values must be examined shifted before the the proper sorted location (relative to the first $i-1$ locations) for what was in the ith location of the array.

2. Here is a slightly different version of the Quick Select algorithm, which is much closer to the original version of Quick Sort. At the end of the function (including all recursive calls), the desired value will be in the kth location of the array.

```
Q_Select_One ( int A[ ], const int k, const int Left, const int Right )
{
    if( Left < Right ) {
        int Pivot = A [Left];
        unsigned int i = Left, j = Right + 1;

        for( ; ; ) {
            while( A[ ++i ] < Pivot && i<Right );
            while( A[ --j ] > Pivot );
            if( i < j )
                Swap( A[ i ], A[ j ] );
            else
                break;
        }
        Swap( A[ j ], A[ Left ] ); // Move pivot to sorted position.

        if( k < j )       Q_Select_One( A, k, Left, j-1 );
        else if( k > j )  Q_Select_One( A, k, j+1, Right );
    }
}
```

   (a) Assume the original call is made to this function with

```
Q_Select_One( A, 3, 0, 8)
```

with the following contents of A

0	1	2	3	4	5	6	7	8
14	24	8	16	32	71	26	25	12

Show the contents of A near the end of the code, just before entering the conditional to check if a recursive call should be made. What recursive call, if any, is made?

Solution: Here is the array.

0	1	2	3	4	5	6	7	8
8	12	14	16	32	71	26	25	24

The recursive call is `Q_Select_One( A, 3, 3, 8)`. Note that this call is made, even though the 16 in position 3 is the correct value.

(b) What is the worst-case number of comparisons in `Q_Select_One` as a function of both n and k, where n is the value of `Right-Left+1` in the first call? When does this occur?

Solution: The worst-case occurs when A is already sorted into increasing order. In this case, the first call will make $n-1$ comparisons and recursively consider the interval from 1 to $n-1$. Each of the k subsequent calls will each involve one fewer comparisons and, except when `Left==k`, will make a recursive call involving an interval that is one location smaller. This gives $k+1$ calls and $O(n)$ comparisons per call, yielding $O(nk)$ comparisons. More precisely, the number of comparisons is

$$\begin{aligned}\sum_{i=0}^{k}(n-i-1) &= (k+1)\,n \;-\; \sum_{i=0}^{k} i \;-\; (k+1)] \\ &= (k+1)\,n \;-\; k(k+1)/2 \;-\; (k+1)] \\ &= O(k\,n) \qquad \text{since } k \le n\end{aligned}$$

3. Solve the Quick Sort recursive function

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

to yield a non-recursive form when $k = 1$. Assume that $T(0) = T(1) = 0$ and $T(2) = 2$. You may assume that n is even.

Solution: In this special case,

$$T(n) = T(1) + T(n-2) + n = T(n-2) + n.$$

Hence,

$$\begin{aligned}
T(n) &= T(n-2) + n \\
&= T(n-4) + (n-2) + n \\
&= T(n-6) + (n-4) + (n-2) + n \\
&= \ldots \\
&= 2 + 4 + \ldots + (n-4) + (n-2) + n \\
&= \sum_{i=1}^{n/2} 2i = 2\frac{n/2(n/2+1)}{2} = n^2/4 + n/2
\end{aligned}$$