Week 3 Quiz

Question #1

10

Question #2

C = [5, 7, 7, 10, 12]

Question #3

True. As referred from CLRS p. 207:

Assume a decision tree of height h with l reachable leaves corresponding to a comparison sort on n elements.

$\because$ Each of the $n!$ permutations appear as one or more leaves,

$\therefore n! \leq l.$

$\because$ A binary tree of height h has no more than 2^h leaves,

$\therefore n! \leq l \leq 2^h,$

$\therefore \log n! \leq \log l \leq \log 2^h,$

$\therefore \log n! \leq h.$

As for the lower bound of comparison sorts $\log n!$,

$\because \log n! = \log 1 + \cdots + \log \frac{n}{2} + \cdots + \log n \geq \log \frac{n}{2} + \cdots + \log n,$

$\therefore \log n! \geq \log \frac{n}{2} + \log(\frac{n}{2} + 1) + \cdots + \log n \geq \log \frac{n}{2} + \log \frac{n}{2} + \cdots + \log \frac{n}{2},$

$\therefore \log n! \geq \frac{n}{2} \cdot \log \frac{n}{2},$

$\therefore$ The lower bound of comparison sorts is $\Omega(n \cdot \log n)$.

Question #4

True, when it is the best and average case for quick select. Pseudocodes

are as following:

```
QUICK-SELECT(arr, k):
    # Randomly choose a pivot from the array.
    pivot = random.choice(arr)
    # Split the array into three parts, which are lists of numbers lower than the pivot,
    # equal to the pivot, and higher than the pivot.
    low_nums = [element for element in arr if element < pivot]
    pivots = [element for element in arr if element = pivot]
    high_nums = [element for element in arr if element > pivot]
    if k < len(low_numbers):
        return QUICK-SELECT(low_nums, k)
    elif k < len(low_numbers) + len(pivots):
    # When k is larger than the length of smaller numbers but smaller than the sum of
    # length of lower and equal numbers, then obviously the k-th number is in the pivot
    # list.
        return pivot[0]
    else:
        return QUICK-SELECT(high_nums, k – len(low_nums) – len(pivots))
```

And the time complexity of above quick select algorithm is $O(n)$, since on average we split the array into two halves, and each time we would recurse one list, based on the length of those lists. Therefore, we know that:

$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \cdots \leq 2n$$

Thus, the upper bound of above algorithm is $O(2n) = O(n)$.