Divide and Conquer Algorithms (Practice Set-1)

(Courtesy Aditya Pancholi)

1. Consider the following code and answer the questions below

```
my_Search(arr, low, high, key)
            if low > high
                return False

            else
                mid = (low + high) / 2
                    if key == arr[mid]
                    return True

                else
                    found1 = my_Search(arr, low, mid-1, key)
                    found2 = my_Search(arr, mid + 1, high, key)
                    return found1 OR found2
```

   (a) What does the above function perform?

   (b) What is the best case and worst case input for the above function?

   (c) Write the recurrence relation for the above function.

   (d) What is the running time for the above function?

2. Consider the following code and answer the questions below

```
my_Search(arr, low, high, key)
            if low > high
                return False

            else
                mid = (low + high) / 2
                    if key == arr[mid]
                    return True

                else if key > arr[mid]
                    return my_Search(arr, mid + 1, high, key)

                else
                    return my_Search(arr, low, mid - 1, key)
```

   (a) What does the above function perform?

(b) What is the best case and worst case input for the above function?

(c) Write the recurrence relation for the above function.

(d) What is the running time for the above function?

(e) If the statement mid=(low+high)/2 is replaced by **mid=(low+high)/3**

1. What is the best case and worst case input for the above function?
2. Write the recurrence relation for the best case time complexity of the above function.
3. Write the recurrence relation for the worst case time complexity of the above function.
4. What is the running time for the above function?

(f) If the statement mid=(low+high)/2 is replaced by **mid=2*(low+high)/3**

1. What is the best case and worst case input for the above function?
2. Write the recurrence relation for the best case time complexity of the above function.
3. Write the recurrence relation for the worst case time complexity of the above function.
4. What is the running time for the above function?

(g) If the statements *return my_Search(arr, mid + 1, high, key)* and *return my_Search(arr, low, mid-1, key)* are replaced by **return my_Search(arr, mid , high, key)** and **return my_Search(arr, low, mid, key)** respectively, What effect will it have on the function?

3. Suppose we have an $\mathcal{O}(n)$ time algorithm to find median of n elements.

   (a) Can you modify Quick-Sort Algorithm to improve it's worst case performance?

   (b) How will it affect the best running time of the algorithm?

4. Given a sorted array $A[1, \ldots, n]$ such that elements may not be distinct. Modify binary search such that for a given *key*, it returns the index of the first occurrence of *key*.

5. Given a sorted array $A[1, \ldots, n]$ such that elements are distinct. The array is rotated around some random position.

   (a) Modify binary search such that for a given *key*, it returns the index where *key* is present (if it exists).
   For example if the input array is $[5, 8, 10, 11, -2, 0, 1, 2, 4]$ and $key = 10$, it returns the index where 10 is present, that is 3.

(b) Can your algorithm be further modified to handle the case where elements may not be distinct?

6. Given a sorted array $A[1, \ldots, n]$ such that elements may not be distinct. Give an $\mathcal{O}(lg(n))$ algorithm that returns the frequency of a given element key.

7. Given a sorted array $A[1, \ldots, n]$ such that elements are all distinct. Give an $\mathcal{O}(lg(n))$ algorithm that finds out if $\exists k$ such that $A[k] = k$. The algorithm returns k if such an index exists, else -1.

8. A ternary search is similar to binary search where instead of splitting the array into 2 equal parts, we split the array into 3 equal parts.
   (a) Write the algorithm to search a given key using Ternary Search.
   (b) Write the recurrence relation for the best case time complexity of the above function.
   (c) Write the recurrence relation for the worst case time complexity of the above function.
   (d) Solve the recurrence relation to calculate the running time of Ternary Search.
   (e) Compare the running time of Ternary Search to standard Binary Search.

9. You are given two sorted lists of size m and n respectively. Give an $\mathcal{O}(lg(m)+lg(n))$ algorithm for computing the k^{th} smallest element in the union of two lists.

10. For a given list L of size n, an element e is said to be **majority element** iff the frequency of e is strictly greater than $\frac{n}{2}$. Give a divide and conquer algorithm that returns the majority element for a given list (if one exists).
   (a) Write the recurrence relation for your algorithm.
   (b) What is the running time for your algorithm.

11. Given an array $A[1, \ldots, n]$, the goal is to find out a subarray A' of $A[]$ (Subarray is defined as a continuous part or section of an array) such that the sum of element of A' is maximum over all subarrays. Give a divide and conquer algorithm that returns the maximum sum subarray of a given array.
   (a) Write the recurrence relation for your algorithm.
   (b) What is the running time for your algorithm.