1 Merge sort

Write the recurrence for the following code and solve it using tree method.

Algorithm 1

```
Require: left < right; right < length(arr)
function mergeSort(int arr[], int left, int right)
    int middle = (left+right)/2
    mergeSort(arr,left,middle);
    mergeSort(arr, middle+1, right);
    merge(arr, left, middle, right);
end function
```

Algorithm 2

```
function merge(int arr[], int left, int middle, int right)
    Divide arr[] into two arrays: Left[] and Right[] //O(n)
    int n1 = middle - left + 1 //Size of Left array
    int n2 = right - middle //Size of Right array
    int i = 0, j = 0, k = left //Initial index variables
    while i < n1 AND j < n2 do
        if Left[i] $\leq$ Right[j] then
            arr[k] = Left[i]
            i++
        else
            arr[k] = Right[j]
            j++
        end if
        k++
    end while
    while i < n1 do
        arr[k] = Left[i]
        i++
        k++
    end while
    while j < n2 do
        arr[k] = Right[j]
        j++
        k++
    end while
end function
```

2 Tree method

Approximate upper and lower asymptotic bound of the following recurrences using tree method.

$$T(n) = T(n-1) + n \tag{1}$$

$$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + T(\frac{n}{6}) + n \tag{2}$$

$$T(n) = T(\frac{n}{2}) + n^2 \tag{3}$$

$$T(n) = T(n-1) + T(n-2) + 1 \tag{4}$$

3 Substitution method*

Guess and then prove the upper asymptotic bounds of the following recurrences using substitution method.

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + 1; T(1) = 0 \tag{5}$$

$$T(n) = T(\left\lceil \frac{n}{2} \right\rceil) + 1; T(1) = 0 \tag{6}$$

$$T(n) = 2 \cdot T(\left\lfloor \sqrt{n} \right\rfloor) + lg(n); T(1) = 1 \tag{7}$$