1 Merge sort

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

Algorithm 1

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

Algorithm 2

1: function MERGE(int arr[], int left, int middle, int right)
2:     Divide arr[] into two arrays: Left[] and Right[] //O(n)
3:     int n1 = middle - left + 1 //Size of Left array
4:     int n2 = right - middle //Size of Right array
5:     int i = 0, j = 0, k = left //Initial index variables
6:     while i < n1 AND j < n2 do
7:         if Left[i] ≤ Right[j] then
8:             arr[k] = Left[i]
9:             i++
10:        else
11:            arr[k] = Right[j]
12:            j++
13:        end if
14:        k++
15:    end while
16:    while i < n1 do
17:        arr[k] = Left[i]
18:        i++
19:        k++
20:    end while
21:    while j < n2 do
22:        arr[k] = Right[j]
23:        j++
24:        k++
25:    end while
26: end function
2 Tree method

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

\[ T(n) = T(n - 1) + n \] \hspace{1cm} (1)

\[ T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + T(\frac{n}{6}) + n \] \hspace{1cm} (2)

\[ T(n) = T(\frac{n}{2}) + n^2 \] \hspace{1cm} (3)

\[ T(n) = T(n - 1) + T(n - 2) + 1 \] \hspace{1cm} (4)

3 Substitution method*

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

\[ T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + 1; T(1) = 0 \] \hspace{1cm} (5)

\[ T(n) = T(\left\lceil \frac{n}{2} \right\rceil) + 1; T(1) = 0 \] \hspace{1cm} (6)

\[ T(n) = 2 \cdot T(\left\lfloor \sqrt{n} \right\rfloor) + \log(n); T(1) = 1 \] \hspace{1cm} (7)