* Merge Sort Pseudo Code:-

```
mergeSort(A, l, u)
If (l < u)
    mid = l + (u - l)/2
    mergeSort(A, l, mid)
    mergeSort(A, mid+1, u)
    merge(A, l, mid, u)
merge(A, l, mid, u)
n1 = mid - l + 1
n2 = u - mid
L[n1], R[n2]
for i=0 to n1-1
    L[i] = A[i+l]
for i=0 to n2-1
    R[i] = A[mid+l+1]
i=0, j=0, k=l
while(i<n1 and j<n2)
    if(L[i] <= R[j])
        A[k] = L[i]
        i = i+1
    else
        A[k] = R[j]
        j = j+1
    k = k+1
while(i<n1)
    A[k] = L[i]
    i = i+1
    k = k+1
while(j<n2)
    A[k] = L[j]
    j = j+1
    k = k+1
```

→ Lets assume that Time taken by mergeSort function for n length array is T[n] and time taken by merge function ~~for n length~~ will be O(n) as traversal of Array 1 time.

⇒ $T(n) = 2T(n/2) + Q(n)$

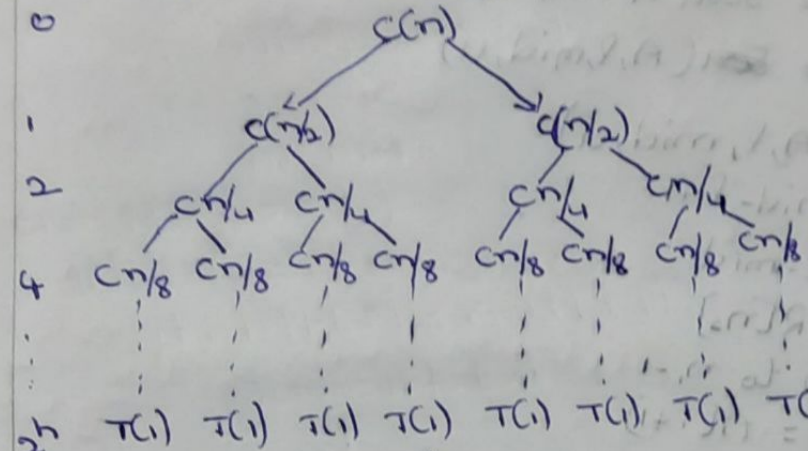
let $Q_n = C_n$

$T(n) = 2T(n/2) + C_n$ ⇒ recurrance relation

⇒ $T(n/2) = 2T(n/4) + C_n/2$ Here $[C_n = nk]$ k-Constant

$T(n) = 4T(n/4) + C_n + C_n$

⇒ Solving equation using recursive tree method.



⇒ Adding all nodes

$T(n) = 2^0 C(n) + 2^1 (C(n/2)) + 2^2 (Cn/4) + \dots + 2^h (T(1))$

$n = 2^h$

$h = \log_2 n$

$T(n) = Cn[1 + 1 + 1 + \dots + \log n \text{ times}] +$

$= \Theta nk \log n$

$T(n) = O(n \log n)$

merge Sort using divide an array in 3 subarray

⇒ mergesort(A, l, u)

```
If(l < u)
    Size = u - l + 1
    mid1 = l + (Size/3)
    mid2 = l + (Size x 2/3)
    mergeSort(A, l, mid1)
    mergeSort(A, mid1+1, mid2)
    mergeSort(A, mid2+1, u)
    merge(A, l, mid1, mid2, u)
```

```
merge(A, l, mid1, mid2, u)
    n1 = mid1 - l + 1
    n2 = mid2 - mid1
    n3 = u - mid2
    left[n1], mid[n2], right[n3]
    for i=0 to n1-1
    | left[i] = A[i+l]
    for i=0 to n2-1
    | mid[i] = A[i+mid1+1]
    for i=0 to n3-1
    | right[i] = A[i+mid2+1]
    i=0, j=0, k=0, m=l
While (i<n1 and j<n2 and k<n3)
    if (left[i] <= mid[j])
    | if (left[i] <= right[k])
    | | A[m] = left[i]
    | | i = i+1
    | | m = m+1
    | else
    | | A[m] = right[k]
    | | k = k+1
    | | m = m+1
    else
    | if (mid[j] <= right[k])
    | | A[m] = mid[j]
    | | j = j+1
    | | m = m+1
    | else
    | | A[m] = right[k]
    | | k = k+1
    | | m = m+1
```

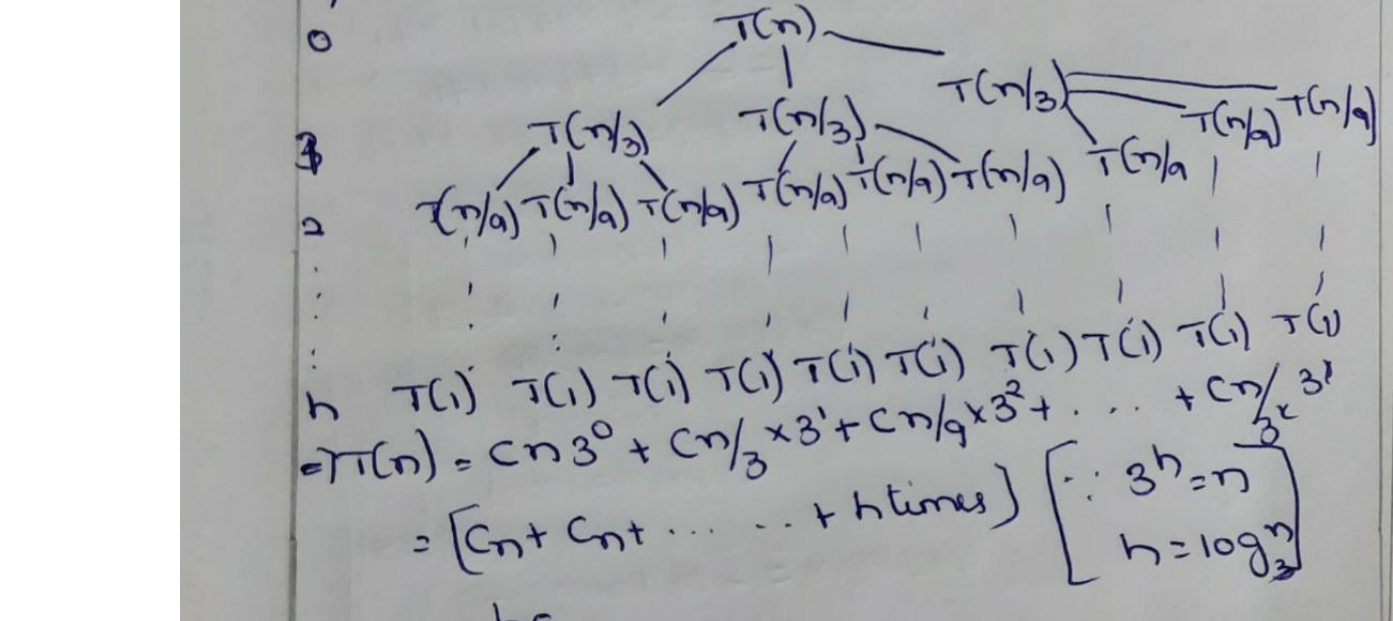
```
if (k >= n3)
  while (i < n1 and j < n2)
    if (left[i] <= mid[j])
      A[m] = left[i]
      i = i+1
      m = m+1
    else
      A[m] = mid[j]
      j = j+1
      m = m+1
else if (j >= n2)
  while (i < n1 and k < n3)
    if (left[i] < right[k])
      A[m] = left[i]
      i = i+1
      m = m+1
    else
      A[m] = right[k]
      k = k+1
      m = m+1
else if (i >= n1)
  while (j < n2 and k < n3)
    if (mid[j] < right[k])
      A[m] = mid[j]
      j = j+1
      m = m+1
    else
      A[m] = right[k]
      k = k+1
      m = m+1
while (i < n1)
  A[m] = left[i]
  i = i+1
  m = m+1
```

```
while(j < n2)
    A[m] = mid[j]
    j = j+1
    m = m+1
while( K < n3)
    A[m] = right[K]
    K = K+1
    m = m+1
```

→ AS merge function will take maximum iteration over n size array
→ Time taken by that function will be $\Theta(n)$.
→ $T(n) = 3T(n/3) + \Theta(n)$ let $\Theta(n) = cn = kn$

$T(n) = 3(T(n/3) + kn$

→ $T(n/3) = 3T(n/9) + kn/3$
→ We will Compute time by recursive tree method.



$\therefore T(n) = cn3^0 + cn/3 \times 3^1 + cn/9 \times 3^2 + \ldots + cn/3^h 3^h$

$= [cn + cn + \ldots + h \text{ times}]$ $\left[\because 3^h = n,\ h = \log_3 n\right]$

$= nhc$

$= c \times n \log_3 n$

$\therefore T(n) = O(n \log n)$