Merge Sort

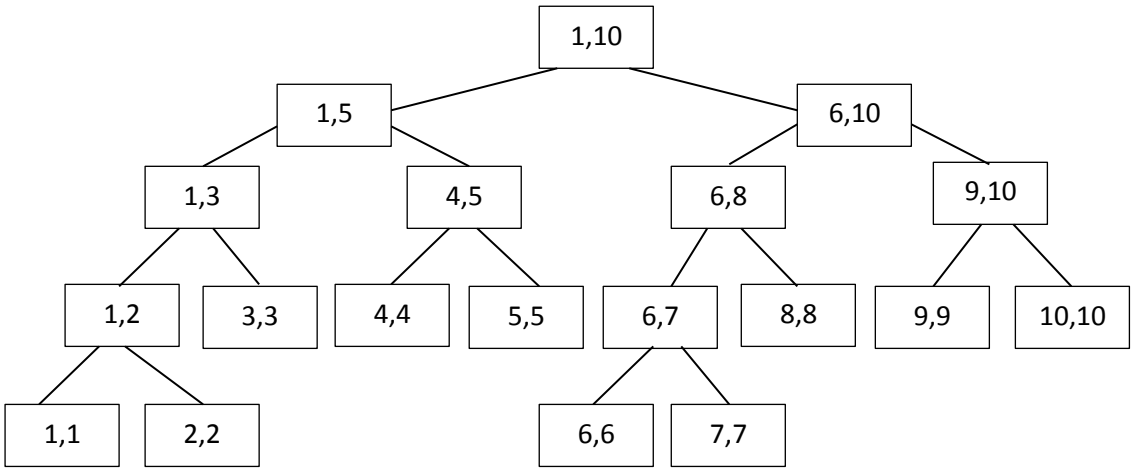
The second example for using divide and conquer strategy is the merge sort problem. The idea: to sort an array a[p..q] , we can divide it into two subarrays, a[p..m] and a[m+1.. q], sort each of the two subarrays recursively, and then merge the two sorted subarrays to produce one sorted array.

```
Function MergeSort(a( ), p , q)
Begin
    If p < q then
        m= (p+q) /2
        MergeSort( a , p , m)
        MergeSort( a , m+1 , q)
        Merge(a , p , m , q)
    Endif
End
Function Merge( a( ) , p, m , q)
Begin
    h=p , j=m+1  , i = p
    while( h< = m) and (j <= q) do
        if a(h) < a(j) then
            b(i)= a(h)
            h=h+1
        else
            b(i)= a(j)
            j=j+1
        endif
```

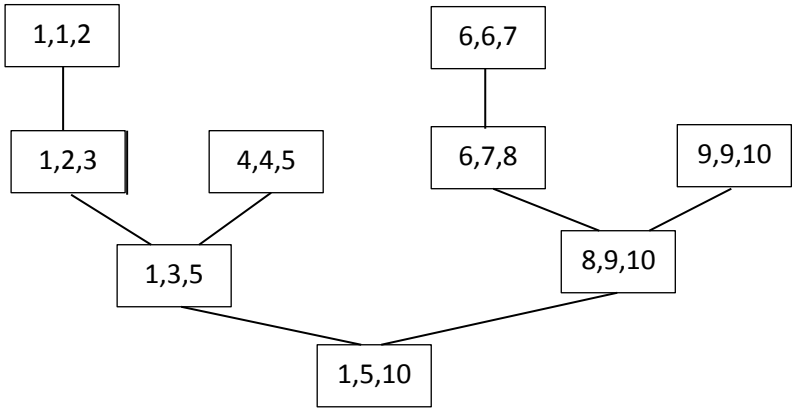
```
            i=i+1
        endwhile
        if h>m  then
            for k=j to q
                b(i)= a(k)
                i=i+1
            endfor
        else
            for k=h to m
                b(i)= a(k)
                i=i+1
            endfor
        endif
        for k=1 to q
            a(k)=b(k)
endfor
```

End

The recursion tree of the MergeSort procedure when n=10 is shown as below. Each node represent recursion described into the current values of p and q.

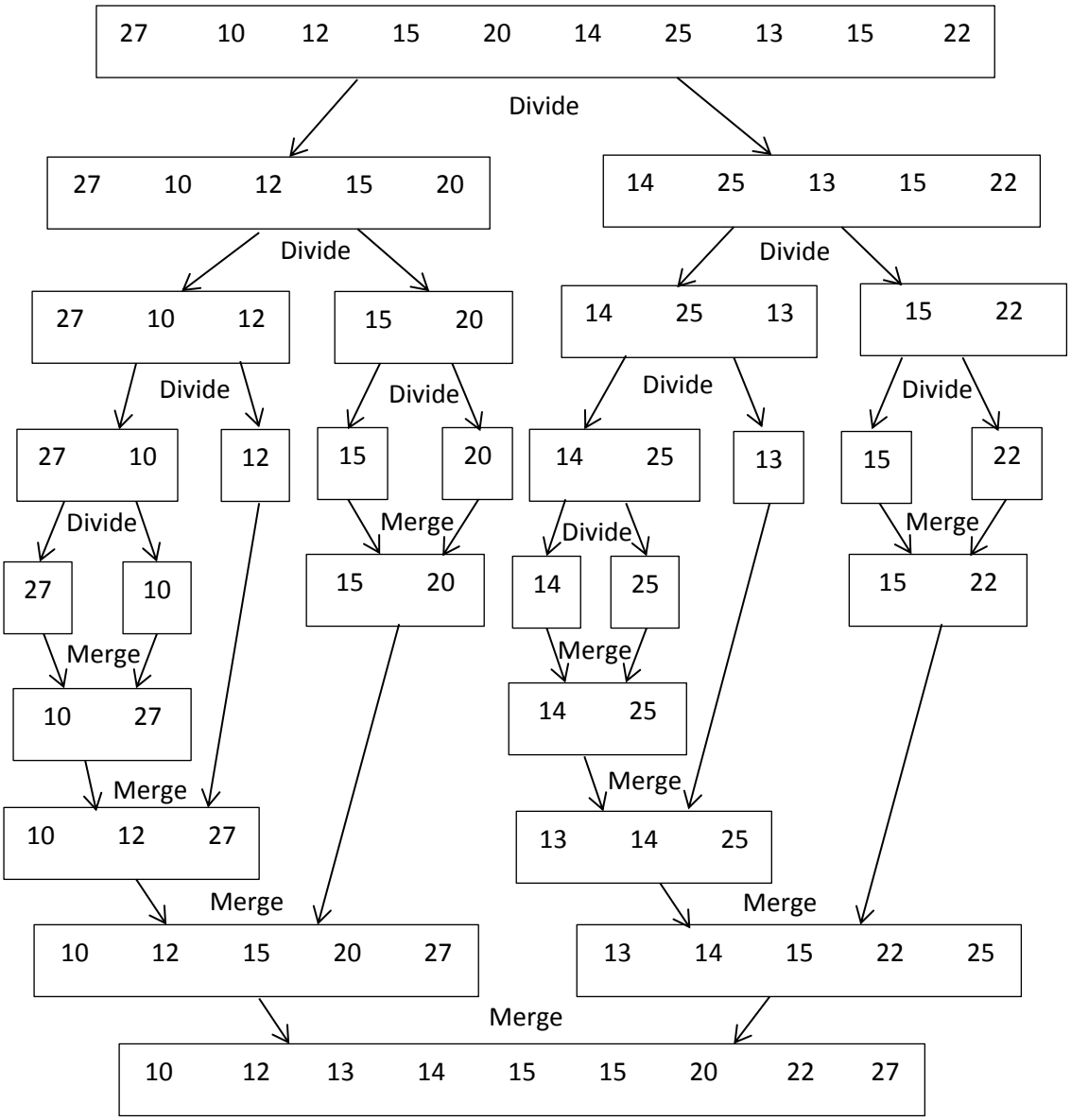


The recursion tree of Merge procedure when n=10 is shown as below. Each node contains of values of p, m , and q.



Example: Sort the following list using Merge Sort algorithm ?

A= [27, 10, 12, 15, 20,14, 25, 13, 15, 22]



Note that, the terminal condition occurs when an array of size 1 is reached; at that time, the merging begin.

Time Complexities:

$$T_{MergeSort}(n) = \begin{cases} a & \text{if } n = 1 \\ 2 * T_{MergeSort}(n/2) + c * n & \text{if } n > 1 \end{cases}$$

$T_{MergeSort}(n) = 2* T_{MergeSort}(n/2)+c*n$

$=2*[2* T_{MergeSort}(n/2^2)+c*n/2]+c*n$

$=2^2* T_{MergeSort}(n/2^2)+c*n+c*n$

$=2^2* T_{MergeSort}(n/2^2)+2c*n$

$=2^2*[2* T_{MergeSort}(n/2^3)+ c* n/2^2] +2c*n$

$=2^3* T_{MergeSort}(n/2^3)+ c* n+2c*n$

$=2^3* T_{MergeSort}(n/2^3)+ 3c*n$

$=2^k* T_{MergeSort}(n/2^k)+ kc*n$

Suppose $n= 2^k$, $k = \log n$

$T_{MergeSort}(n) = an+ cn \log n$

$T_{MergeSort}(n) = O(n \log n)$

Notes:

1.The best, average and worst case of time complexity for MergeSort algorithm is the same O(n log n)

2.We can improve the Merge Sort algorithm by using Insertion Sort when the size of the list become small where this will decrease the recursion depth and enhance the use of the stack space. The following code explain this:

```
Begin

If (q – p + 1) < 20 Then Insertion Sort (a, p, q)

Else   If p < q then

            m= (p+q) /2
            MergeSort( a , p , m)
            MergeSort( a , m+1 , q)
            Merge(a , p , m , q)
       Endif
End
```

This improvements staying the time complexities O(n log n).