```
void MergeSort(int low, int high)
// a[low : high] is a global array to be sorted.
// Small(P) is true if there is only one element to
// sort. In this case the list is already sorted.
{
    if (low < high) { // If there are more than one element
       // Divide P into subproblems.
          // Find where to split the set.
             int mid = (low + high)/2;
       // Solve the subproblems.
          MergeSort(low, mid);
          MergeSort(mid + 1, high);
       // Combine the solutions.
          Merge(low, mid, high);
    }
}
```

Program 3.8 Merge sort

```
void Merge(int low, int mid, int high)
// a[low:high] is a global array containing two sorted
// subsets in a[low:mid] and in a[mid+1:high]. The goal
// is to merge these two sets into a single set residing
// in a[low:high]. b[] is an auxiliary global array.
{
    int h = low, i = low, j = mid+1, k;
    while ((h <= mid) && (j <= high)) {
       if (a[h] <= a[j]) { b[i] = a[h]; h++; }
       else { b[i] = a[j]; j++; } i++;
    }
    if (h > mid) for (k=j; k<=high; k++) {
                     b[i] = a[k]; i++;
                 }
    else for (k=h; k<=mid; k++) {
            b[i] = a[k]; i++;
         }
    for (k=low; k<=high; k++) a[k] = b[k];
}
```

Program 3.9 Merging two sorted subarrays using auxiliary storage

Computing Time for Merge Sort

• If the time for the merging operation is proportional to n,

then :-

The computing time for mergesort is :-

$$T(n) = \begin{cases} a & , n=1 \\ 2T(n/2) + cn & , n>1 \end{cases}$$

[a] a constant

[c] a constant

MergeSort (low, mid)
MergeSort (mid+1, high)

Merge
(Merge time $\propto n$)
$\Rightarrow$ Merge time $= cn$

$$T(n) = 2\,T(n/2) + cn$$

$$= 2\left[2\,T\left(\frac{n}{4}\right) + \frac{cn}{2}\right] + cn$$

$$T(n) = 2\,T(n/2) + cn$$

$$T(n/2) = 2\,T\left[\frac{n/2}{2}\right] + \frac{cn}{2}$$

$$\therefore T(n/2) = 2T\left[\frac{n}{4}\right] + \frac{cn}{2}$$

$$= 2\left[2\,T(n/4) + \frac{cn}{2}\right] + cn$$

$$= 4\,T(n/4) + \cancel{2} \times \frac{cn}{\cancel{2}} + cn$$

$$= 4\,T(n/4) + cn + cn$$

$$= 4T(n/4) + 2cn$$

$$\therefore T(n) = 4\,T(n/4) + 2cn$$

$\Rightarrow$

4 9 T n/s

$$T(n) = 4\,T\left(\frac{n}{4}\right) + 2cn$$

Put $\boxed{n = 2^k}$ Put $\Rightarrow k = 2$

$2^k = 2^2 = 4 = n$

$\Rightarrow n = 4$

$k = 2$

$$\Rightarrow T(n) = 4T(n/4) + 2cn$$

$$\vdots$$

$$T(n) = 2^k T(1) + kcn$$

$\boxed{2^k = 2^2 = 4}$

Keep n as it is multiplying here to study relation with "n"

$$T(n) = 2^k . T(1) + kcn$$

$\boxed{T(1) = a}$

$$T(n) = 2^k . a + kcn$$

$$\boxed{T(n) = a\,2^k + kcn}$$

Now:- $2^k = n \Longrightarrow$ Apply $\log_2$ on both sides

$\Rightarrow \log_2 2^k = \log_2 n \Rightarrow k \log_2 2 = \log_2 n$

$\Rightarrow k \times 1 = \log_2 n$

$\therefore \boxed{k = \log_2 n}$ and we have $\boxed{2^k = n}$

$T(n) = a(n) + \log n\, cn$

$T(n) = an + cn \log n \Rightarrow O(n \log n)$