## Merge sort :-

- Merge sort is a sorting algorithm that uses divide and conquer strategy
- Merge sort an input array with n elements consists of 3 steps.

① Divide

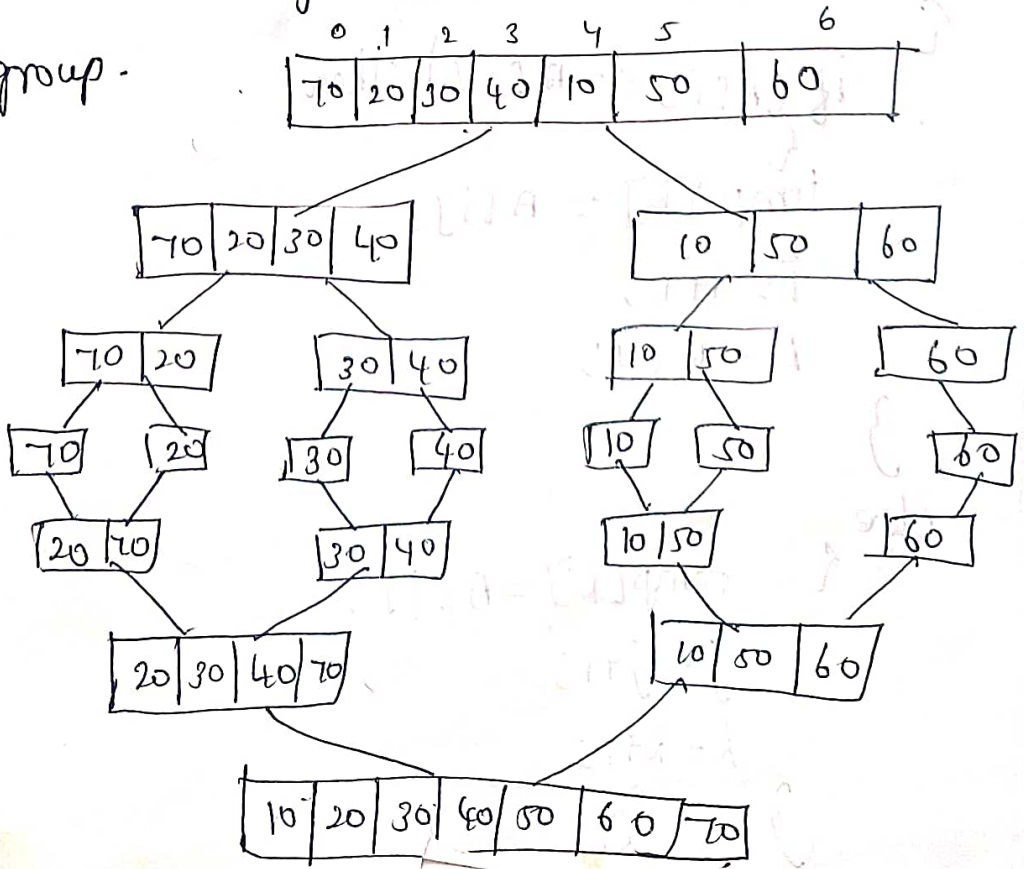
partition array into two sub-lists $S_1$ & $S_2$ with 'n/2' elements each.

② Conquer :-

sort sublists $S_1$ & $S_2$

③ Combine :-

Merge $S_1$ & $S_2$ into unique sorted group.



```
Algorithm mergesort(int A[0...n-1], low, high)
{
    if (low < high) then
    {
        mid ← (low+high)/2;
        mergesort (A, low, mid);
        mergesort (A, mid+1, high);
        combine(A, low, mid, high);
    }
}

Algorithm combine (A[0...n-1], low, mid, high)
{
    k = low;
    i = low;
    j = mid+1;
while (i<=mid && j<=high)
{
    if (A[i]<=A[j]) then
    {
        temp[k] = A[i];
        i = i+1;
        k = k+1;
    }
else
    {   temp[k] = A[j];
        j = j+1;
        k = k+1;
    }
}
```

```
while (i <= mid) do
{
    temp [k] = A[i]
    i = i+1;
    k = k+1;
}
while (j <= high) do
{
    temp[k] = A[j];
    j = j+1;
    k = k+1;
}
```

## Complexity Analysis of Mergesort

* Best case O(n log n)
* worst case O(n log n)
* Average case O(n log n)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 10 | 50 | 60 |

① low = 0 high = 6

low < high 0 < 6

mid = (0+6)/2

mid = 6/2 = 3

mid = 3

mergesort (A, 0, 3)

mergesort (A, 3+1, 6)

Combine (A, low, mid, high)

Mergesort (A, 0, 3)

mid = 0+3/2 = 1.5 = 1

Mergesort (A, 0, 1);

Mergesort (A, 2, 3);

Mergesort (A, 0, 1)

mid = (0+1)/2 = 0.5

Merg(A, 0, 0)

Mergesor(A-1, 1)

Mergesort (A, 0, 0)

low = 0 high = 0 mid = 0

k = 0

i = 0

j = 0+1 = 1

while (0<=0 && j<=1)

A[0] <= A[j]

temp[0] = A[i]

| 20 | |
|---|---|
| 0 | |