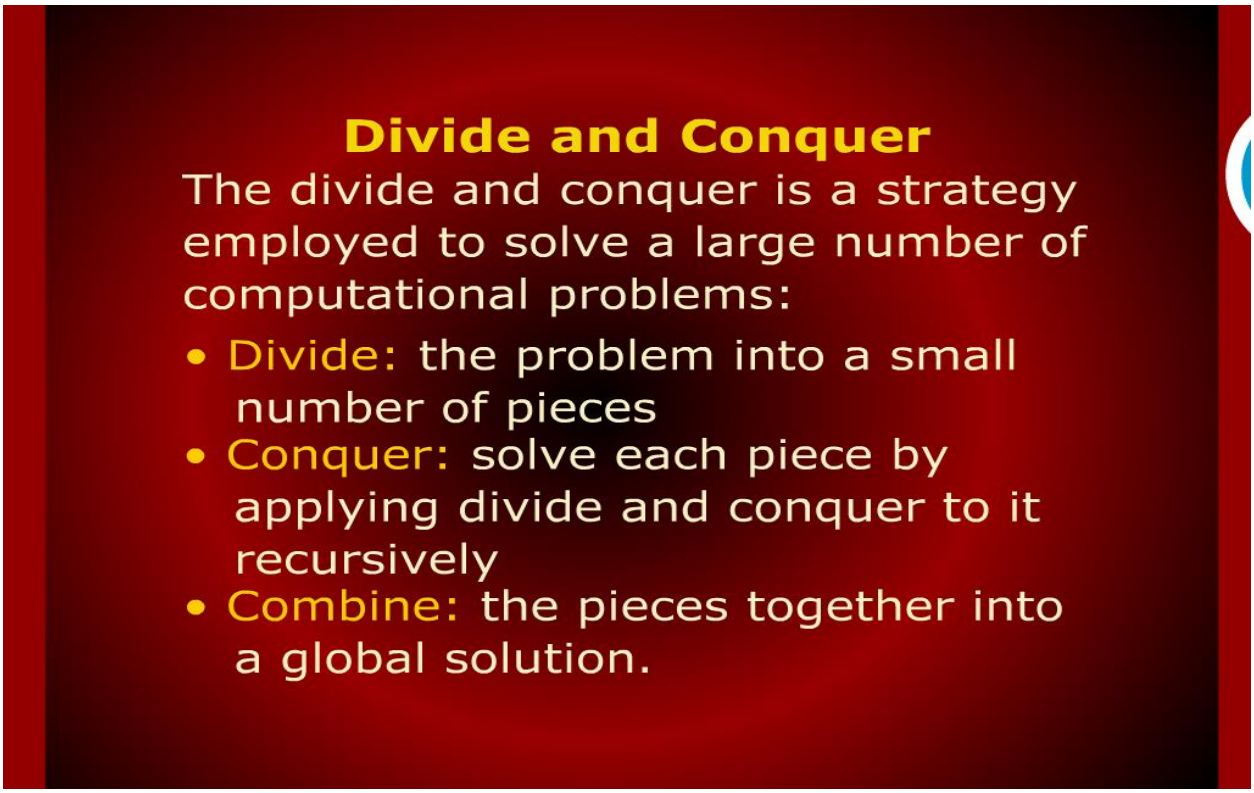
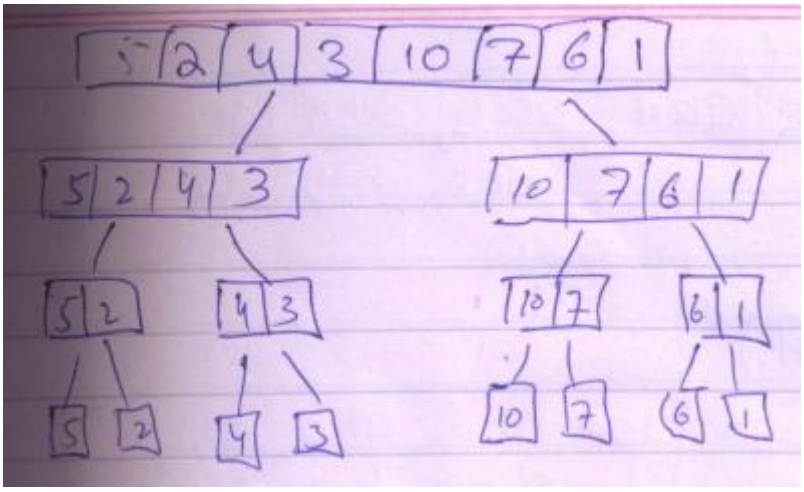
- **Divide and Conquer Algorithm.**



- **MERGE SORT:**

Which uses a recursive technique known as Divide-and-Conquer.



Merge Sort

- Divide and conquer strategy is applicable in a huge number of computational problems.
- The first example of divide and conquer algorithm we will discuss is a simple and efficient sorting procedure called *Merge Sort.*

Merge Sort

- We are given a sequence of n numbers A,which we will assume are stored in an array A[1..n].
- The objective is to output a permutation of this sequence sorted in increasing order.
- This is normally done by permuting the elements within the array A.

Merge Sort

Here is how the merge sort algorithm works:

Merge Sort

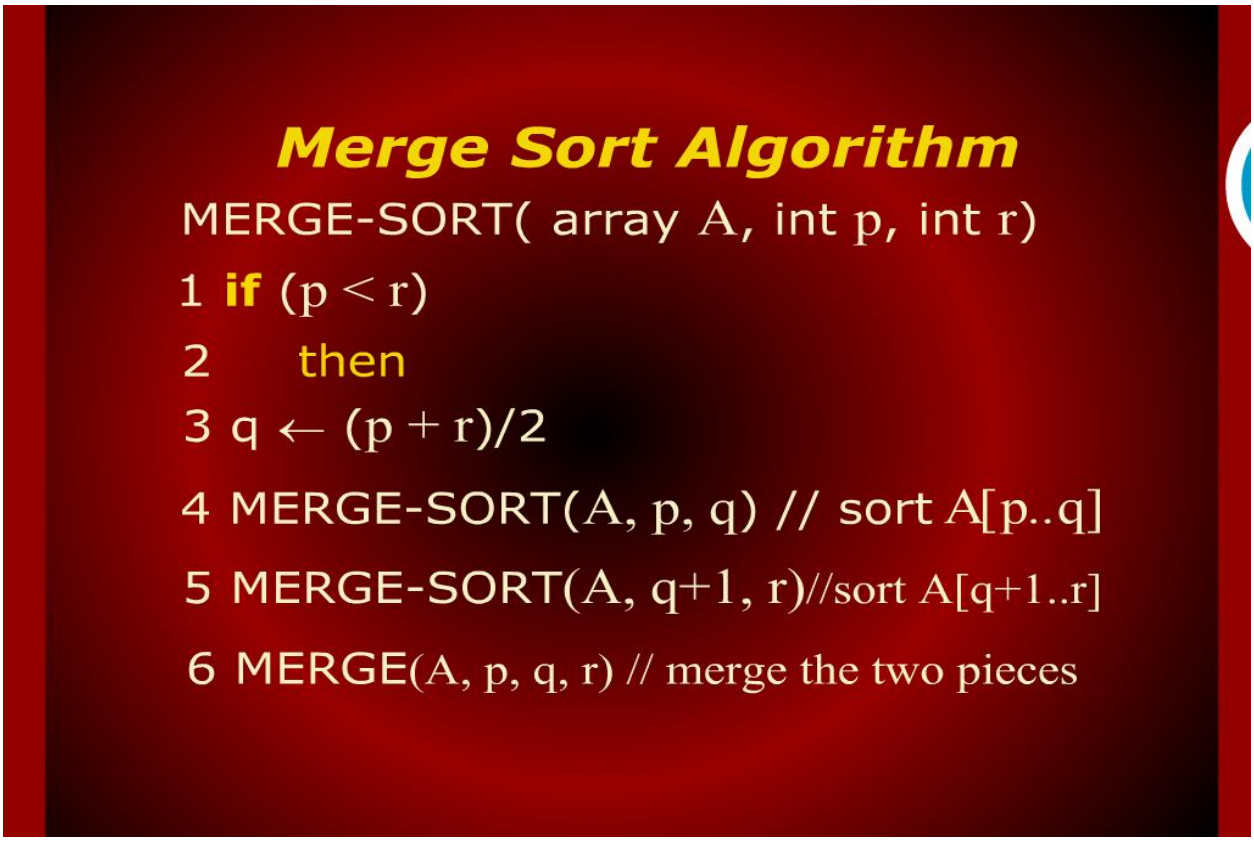
- Divide: split A down the middle into two subsequences, each of size roughly n/2
- Conquer: sort each subsequence by calling merge sort recursively on each.
- Combine: merge the two sorted subsequences into a single sorted list.

Merge Sort

- The dividing process ends when we have split the subsequences down to a single item.
- A sequence of length one is trivially sorted.
- The key operation is the combine stage which merges together two sorted lists into a single sorted list.

Merge Sort

- A sequence of length one is trivially sorted.
- The key operation is the combine stage which merges together two sorted lists into a single sorted list.
- Fortunately, the combining process is quite easy to implement.



Where p is beginning index of array list A and r is ending index of array list A

Merge Sort Algorithm

```
MERGE( array A, int p, int q, int r)
1  int B[p..r]; int i ← k ← p; int j ← q + 1
2    while (i ≤ q) and ( j ≤ r)
3    do if (A[i] ≤ A[ j])
4          then B[k++ ] ← A[i++ ]
5          else B[k++ ] ← A[ j++ ]
6    while (i ≤ q)
7    do B[k++ ] ← A[i++ ]
8    while ( j ≤ r)
```

```
9    do B[k++ ] ← A[ j++ ]
10   for i ← p to r
11   do A[i] ← B[i]
```