Merge Sort

Merge Sort

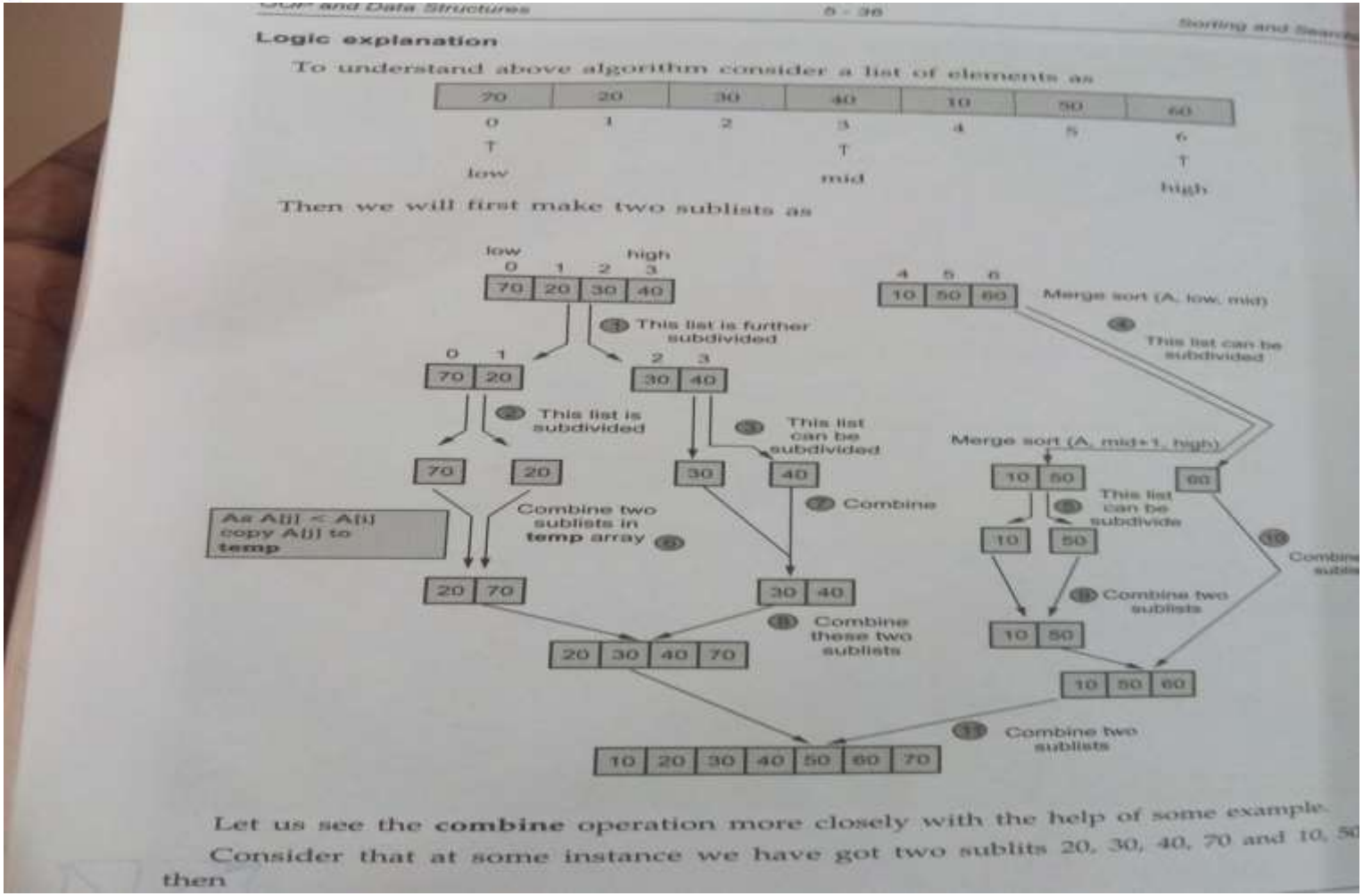
- Merge sort is external sorting technique which uses divide and conquer strategy
- Algorithm:

  Divide: partition array into two sub-list s1 and s2 with n/2 elements each.

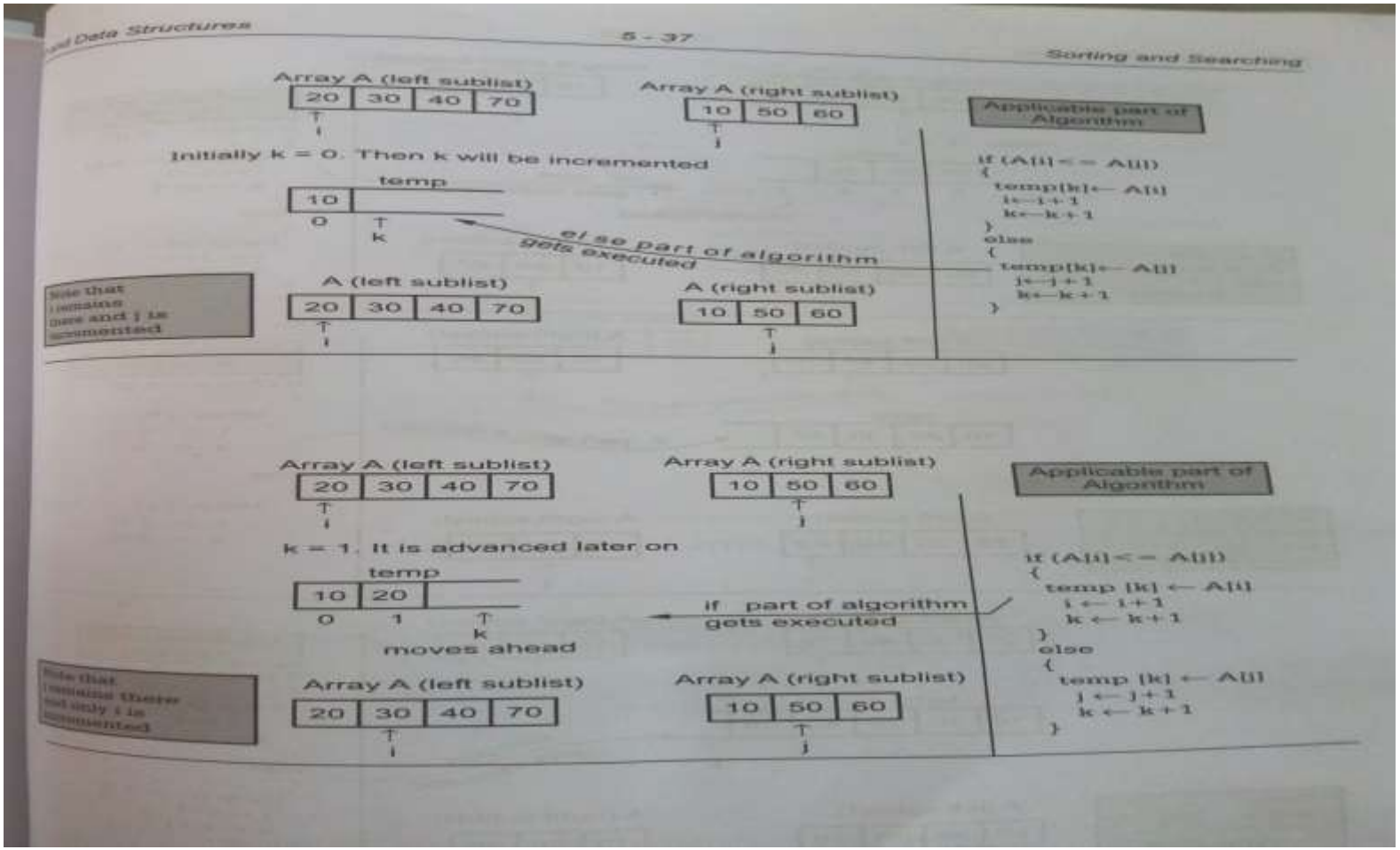
  Conquer: Sort sub-list s1 and sub-list s2

  Combine: Merge s1 and s2 into unique sorted group

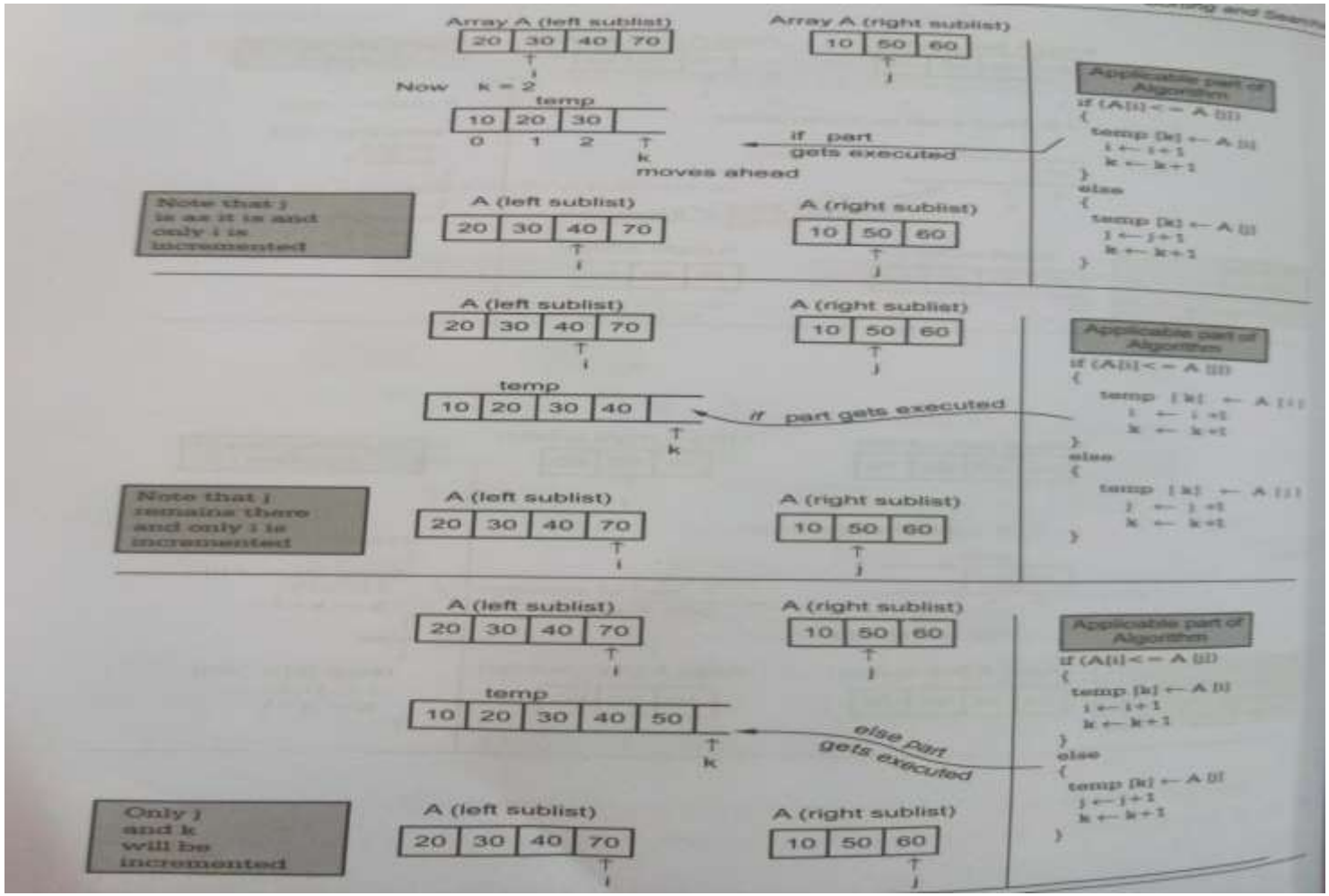
Merge Sort


Logic explanation
To understand above algorithm consider a list of elements as
70 20 30 40 10 50 60
0 1 2 3 4 5 6
low
mid
high
Then we will first make two sublists as
This list is further subdivided
This list is subdivided
This list can be subdivided
Combine
Combine two sublists in temp array
As A[i] < A[j] copy A[i] to temp
Combine these two sublists
Merge sort (A, low, mid)
This list can be subdivided
Merge sort (A, mid+1, high)
This list can be subdivide
Combine two sublists
Combine two sublists
Let us see the combine operation more closely with the help of some example.
Consider that at some instance we have got two sublits 20, 30, 40, 70 and 10, 50
then

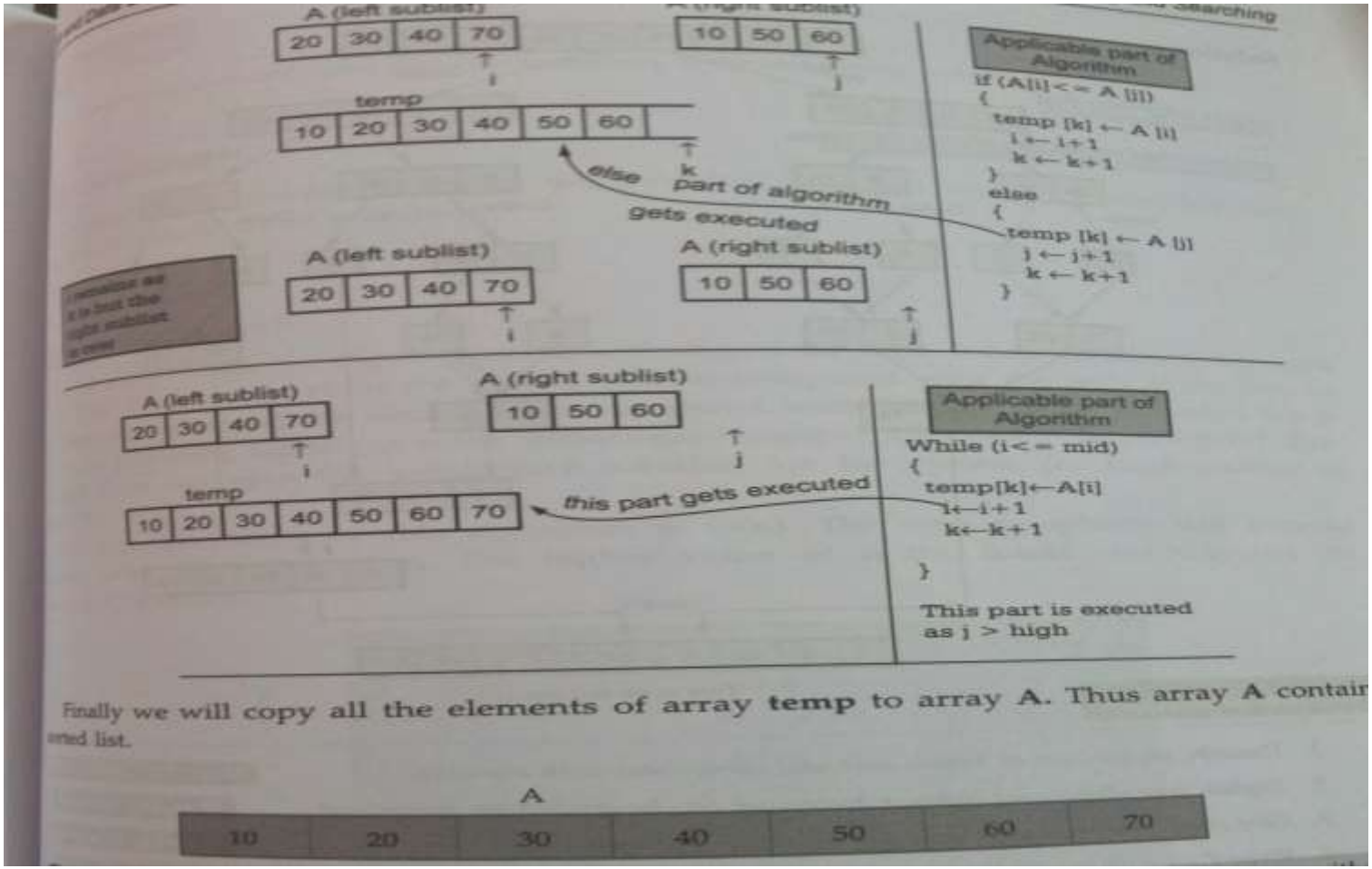
Merge Sort


Data Structures
5 - 37
Sorting and Searching
Array A (left sublist)
20 30 40 70
Array A (right sublist)
10 50 60
Applicable part of Algorithm
Initially k = 0. Then k will be incremented
temp
10
0 k
else part of algorithm gets executed
if (A[i] <= A[j])
{
temp[k]<- A[i]
i<-i+1
k<-k+1
}
else
{
temp[k]<- A[j]
j<-j+1
k<-k+1
}
Note that i remains there and j is incremented
A (left sublist)
20 30 40 70
A (right sublist)
10 50 60
Array A (left sublist)
20 30 40 70
Array A (right sublist)
10 50 60
Applicable part of Algorithm
k = 1. It is advanced later on
temp
10 20
0 1 k
moves ahead
if part of algorithm gets executed
if (A[i] <= A[j])
{
temp [k] ← A[i]
i ← i+1
k ← k+1
}
else
{
temp [k] ← A[j]
j ← j+1
k ← k+1
}
Note that i remains there but only j is incremented
Array A (left sublist)
20 30 40 70
Array A (right sublist)
10 50 60

Logic Explanation



Logic Explaination



EC6301-OOPS AND DATA STRUCTURES

Merge Sort

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
int n;
void main()
{
    int I,low,high,
    int A[10];
    void mergesort(int A[10],int low,int high);
    void display(int A[10];
    clrscr();
    printf("enter the length of list:");
```

Merge Sort

```
    scanf(“%d”,&n);
    printf(“\n enter the elements:”);
    for(i=0;i<n;i++)
    scanf(“%d”,&A[i]);
    low=0;
    high=n-1;
    mergesort(A,low,high);
    Display(A);
    getch();
}
```

Merge Sort

```
void mergesort(int A[10],int low,int high)
{
    int mid;
    void combine(int A[10],int low, int mid,int high);
    if(low<high)
    {
        mid=(low+high)/2;
        mergesort(A,low,mid);
        mergesort(A,mid+1,high);
        combine(A,low,mid,high);
    }
}
```

Merge Sort

```
void combine(int A[10],int low, int mid,int high)
{
        int I,j,k;
        int temp;
        k=low;
        i=low;
        j=mid+1;
        while(i<=mid && j<=high)
        {
        if(A[i]<=A[j])
        {
            temp[k]=A[j]; i++; k++; }
```

Merge Sort

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

Merge Sort

```
    while(j<=high)
    {
        temp[k]=A[j];
        j++;
        k++;
    }
    for(k=low;k<=high;k++)
    A[k]=temp[k];
}   void display(int A[10)
    {   int i;
        printf("\n the sorted array is");
        for(i=0;i<n;i++)printf("%d\t",A[i]); }
```