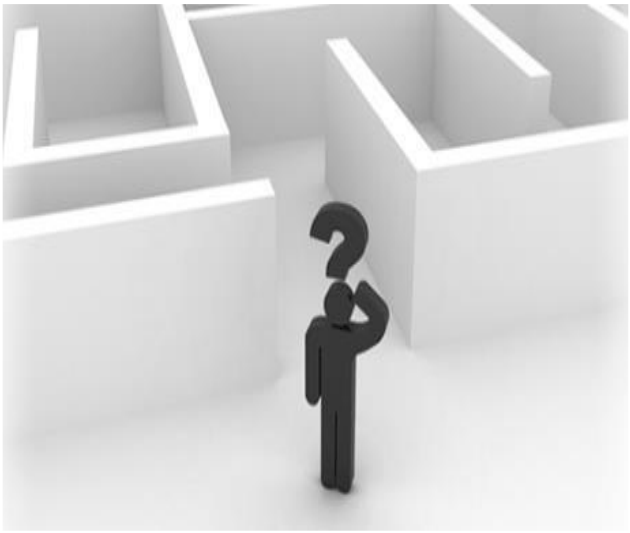
UNIT -2 – PART -1 - SORTING & SEARCHING



- Bubble sorting
- Insertion sorting
- Quick sorting
- Bucket sorting
- Merge sorting
- Selection sorting
- Shell sorting
- Basic searching technique
- Sequential searching
- Binary searching

- **The process of "looking up" a particular record in the data is called "searching".**
- **The process of ordering the records in a database is called "sorting".**

Sorting

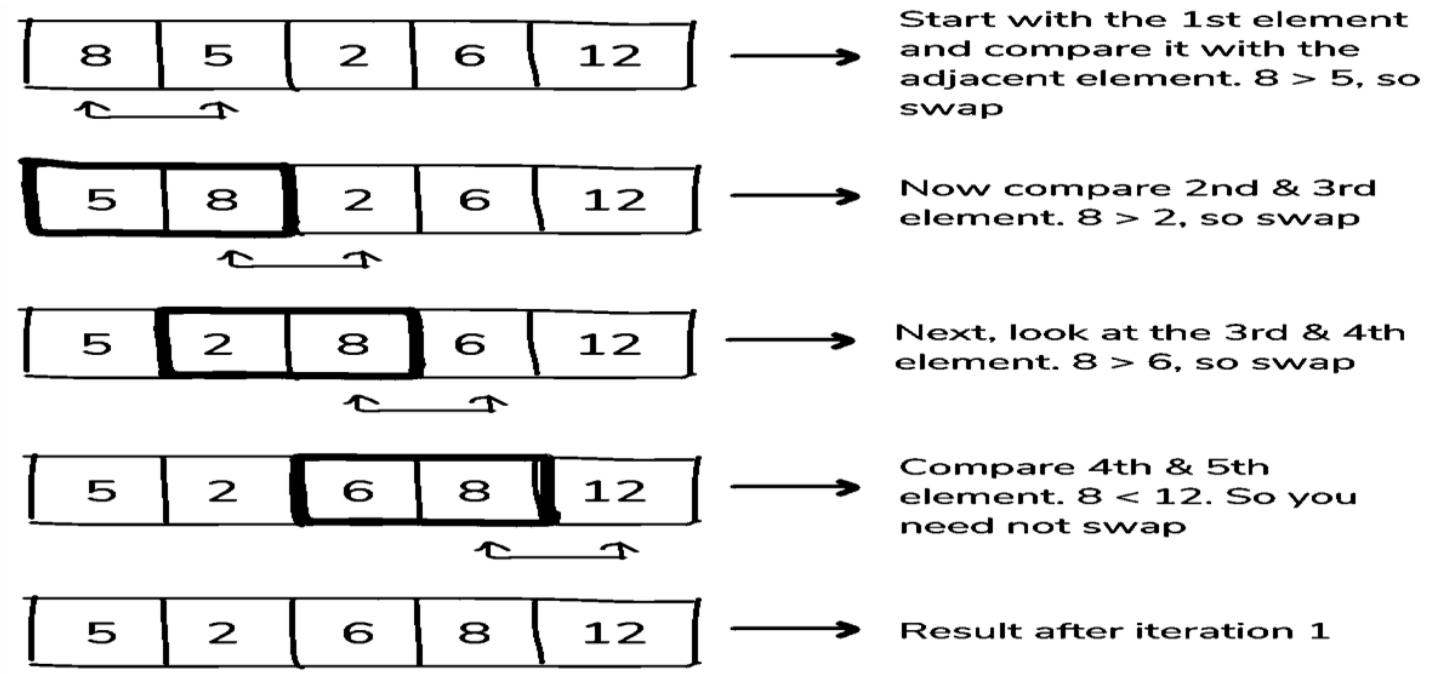
- The operation is the most common task performed by computers.
- Sorting is the process of arranging data information in some logical order.
- This logical order may be ascending or descending in case of numeric values.
- Various techniques are available to sort data depending on length of data, speed of sorting, number of swapping done during procedure of sorting etc .

Types of sorting techniques

- **1. Bubble sort**
- **2. Insertion sort**
- **3. Quick sort**
- **4. Bucket sort**
- **5. Merge sort**
- **6. Selection sort**
- **7. Shell sort**

❖ Bubble sort

Bubble sort is a simple sorting algorithm. It repeatedly swaps adjacent elements if they are in the wrong order until the array is sorted. In this algorithm, the largest element "bubbles up" to the end of the array in each iteration. Bubble sort is inefficient for large data sets, but it is useful for educational purposes and small data sets. In this article, we will implement the bubble sort algorithm in C programming language.



It is also known as "comparison sort" because it continually compares two adjacent elements from the list.

- Bubble sort is a simple sorting algorithm.
- It works by repeatedly swapping adjacent elements if they are in the wrong order.
- The algorithm sorts the array in ascending or descending order.
- It has a time complexity of $O(n^2)$ in the worst case, where n is the size of the array.

Advantages:

- Bubble sort is easy to understand and implement.
- It requires minimal additional memory space to perform the sorting.

Disadvantages:

- It is not efficient for large data sets because of its time complexity.
- It has poor performance compared to other sorting algorithms, such as quicksort and mergesort.

Algorithm for Bubble sort

▪ Let “a” be an array of n numbers. “temp” is a temporary variable for swapping the position of the numbers.

Step 1: Input n numbers for an array “a”

Step 2: Initialize i=0 and repeat through step 4 if(i<n)

Step 3: Initialize j=0 and repeat through step 4 if(j<n-1)

Step 4: if(a[j]>a[j+1])

```
temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;
```

Step 5: Display the sorted numbers of array a

Step 6: Exit

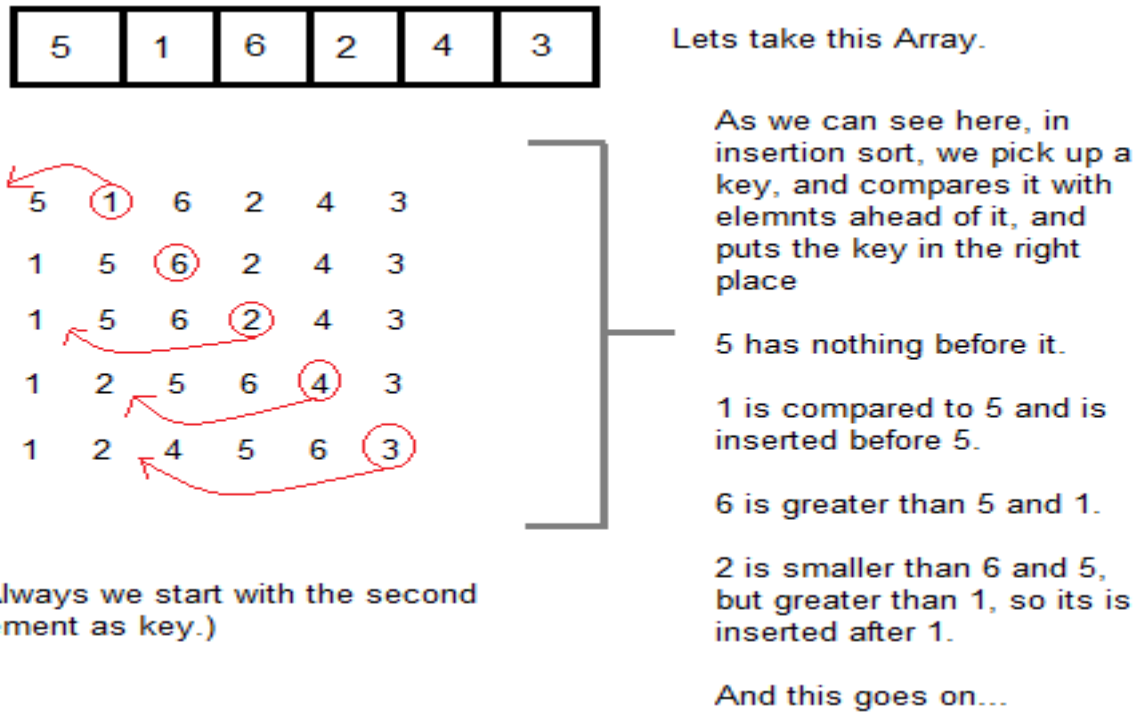
Program for Bubble sort

```
void main()
{
	int a[100],temp,i,j,n,;
	clrscr();
	printf("\nEnter array size:");
	scanf("%d",&n);

	for(i=0;i<n;i++) {

			printf("\nEnter array elements a[%d]:",i);
			scanf("%d",&a[i]);
				}
	for(i=0;i<n;i++)
		{
			for(j=0;j<n-1;j++)
			{
				if(a[j]>a[j+1])
				{
					temp=a[j];
					a[j]=a[j+1];
					a[j+1]=temp;
				}
			}
			printf("\nAfter pass %d elements are:",i+1);
			for(k=0;k<n;k++)
			printf("\t%d",a[k]);
			printf("\n");
		}
	printf("\nAfter sorting array elements are:");
for(i=0;i<n;i++)
	{
printf("\n%d",a[i]);
} getch(); }
```

❖ Insertion sorting



- It is very simple and efficient algorithms for the smallest lists.
- Its mechanism is very simple just take elements from the list one by one and insert them in their correct position into a new sorted list.
- The name inserting sorting means that sorting is occurred by inserting a particular element at proper position.

Algorithm for Insertion sort

Let "a" be an array of n numbers. "temp" is a temporary variable for swapping the position of the numbers. "pos" is the control variable to hold the position of each pass.

Step 1: Input n numbers for an array "a"

Step 2: Initialize i=0 and repeat through step 4 if(i<n-1)

temp=a[i]

j=i-1

Step 3: Repeat the step 3 if(temp<a[j] and (j>=0))

a[j+1]=a[j]

j=j-1

Step 4: a[j]=temp

Step 5: Exit

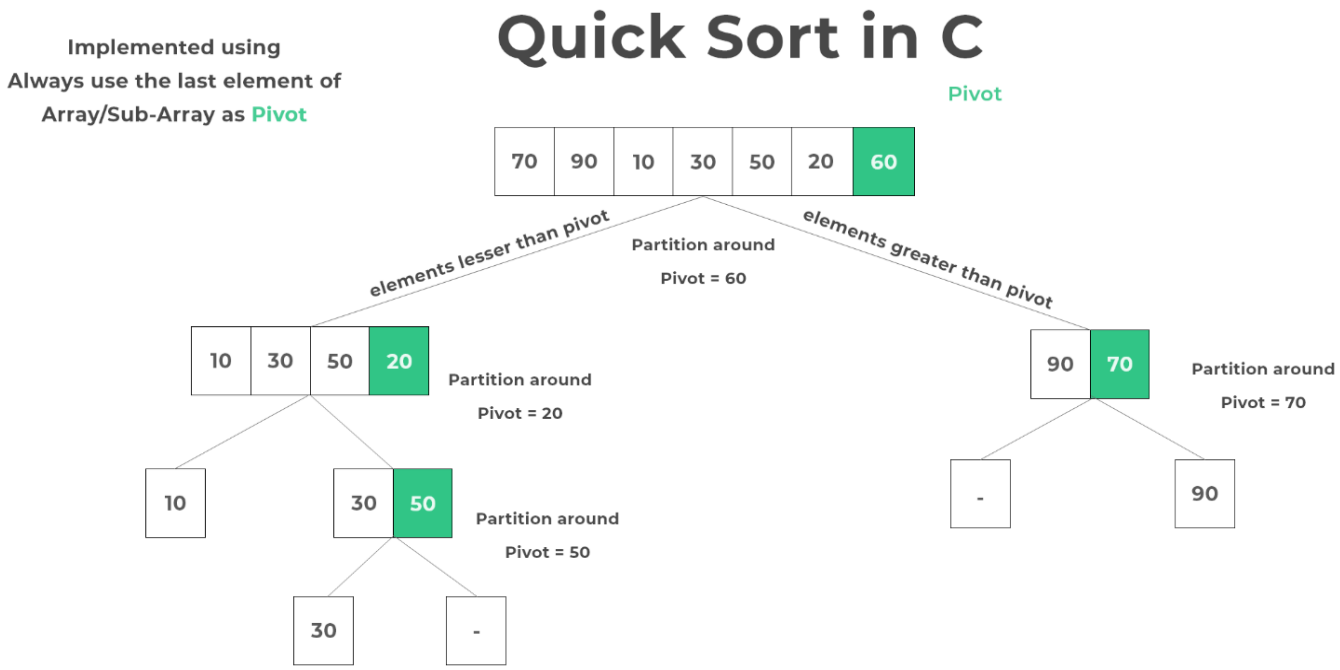
Program for insertion sort

```
void main()
{
	int a[100],n,i,j,k,tmp,;
	clrscr();
	printf("\nEnter array size:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter array element a[%d]:",i);
		scanf("%d",&a[i]);
	}
	printf("\n\nUnsorted array");
	for(i=0;i<n;i++)
	{
		printf("\t%d",a[i]);
	}
	for(i=1;i<n;i++)
		{
	tmp=a[i];
	for(j=i-1;j>=0;j--)
		{
			if(tmp<a[j])
			{
				a[j+1]=a[j];
				a[j]=tmp;
			}
		}
```

```
printf("\nPass %d,element inserted at proper place:%d",i,tmp);
for(k=0;k<n;k++)
printf("\t%d",a[k]);
printf("\n");
}
printf("\n\nSorted array");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
```

❖ Quick sort

- It is widely used sorting techniques which uses divide and conquer (also known as partition exchange sort)mechanism.
- The quick sort algorithm works by partioning the array to be sorted. And each partition is internally sorted recursively.
- In the quick sort mechanism, first of all we have to select middle element form the list and is known as pivot element.
- After that, the sort is divides the list into two sub lists.
- First list contains the elements that are less than the pivot elements and a second list contains elements that are greater than pivot elements.
- The pivot or pivot element is the element of a matrix, or an array, which is selected first by an algorithm (e.g. Gaussian elimination, simplex algorithm, etc.), to do certain calculations.



Algorithm for Quick sort

Let "a" be an array of n numbers. "temp" is a temporary variable for swapping the position of the numbers.

Step 1: initialize low=first, high=last
pivot=(low+high)/2
Step 2 repeat this step till low,=high
while(a[low]<pivot)
low++;
while(a[high]>pivot)
high--;
if(low<=high)
temp=a[low]
a[low]=a[high]
a[high]=temp
Step 3: if(first<high)
quicksort(a,first,high)
Step 4: if(low<last)
quicksort(a,low,last)
Step 5: Exit

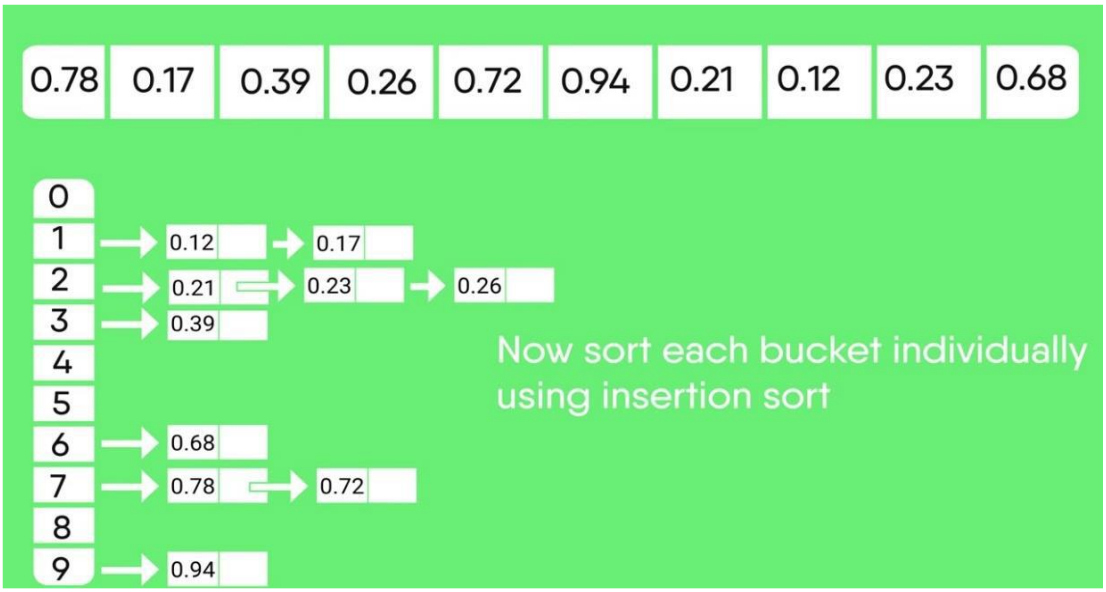
Program for quick sort

```
#include<stdio.h>
#include<conio.h>
void quicksort(int [],int,int);//function declaration
void main()
{
int a[100],n,i;
clrscr();
printf("\nenter array size:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\n enter array elements a[%d]",i);
scanf("%d",&a[i]);
}
quicksort(a,0,n-1);//function calling
for(i=0;i<n;i++)
{
printf("\n sorted array elements %d",a[i]);
}
getch();
        }
void quicksort(int a[], int first, int last)//function definition
{
int low, high, temp, pivot, i;
low=first;
high=last;
pivot=(low+high)/2;
while(low<=high)
{
while(a[low]<a[pivot])
low++;
```

```
while(a[high]>a[pivot])
high--;
if(low<=high)
{
temp=a[low];
a[low]=a[high];
a[high]=temp;
low++;
high--;
}
}
if(first<high)
quicksort(a,first,high);
if(low<last)
        quicksort(a,low,last);
        }
```

❖Bucket sort

- Bucket sort is a sorting method that can be used to sort a list of numbers by its base.
- If we want to sort list of English words where base is 26, then 26 buckets is used to sort the words.
- To sort array of decimal numbers where base is 10 we need 10 buckets and it can be numbered as 0,1,2,3,4,5,6,7,8,9.
- On the basis of the largest number's digit that many passes are required.
- Mechanism includes comparison of the first position of digit with the digit of bucket and place it. (recursive)



Algorithm for Bucket sort

Step 1: Input n number of elements in array a.
Step 2: Find out the largest element and the digit of the largest element
Step 3: Initialize i=1 and repeat steps 4 and 5 until (i<digitcnt)
Step 4: Initialize the buckets j=0 and repeat the steps (a) until (j<n)
(a)Compare i the position of each element of the array with

the bucket number and place it int the corresponding bucket.
Step 5: Read the elements of the bucket from 0th bucket to 9th bucket and from first position to higher one to generate new array a.
Step 6: Display the sorted array a
Step 7: Exit

Program for Bucket sort

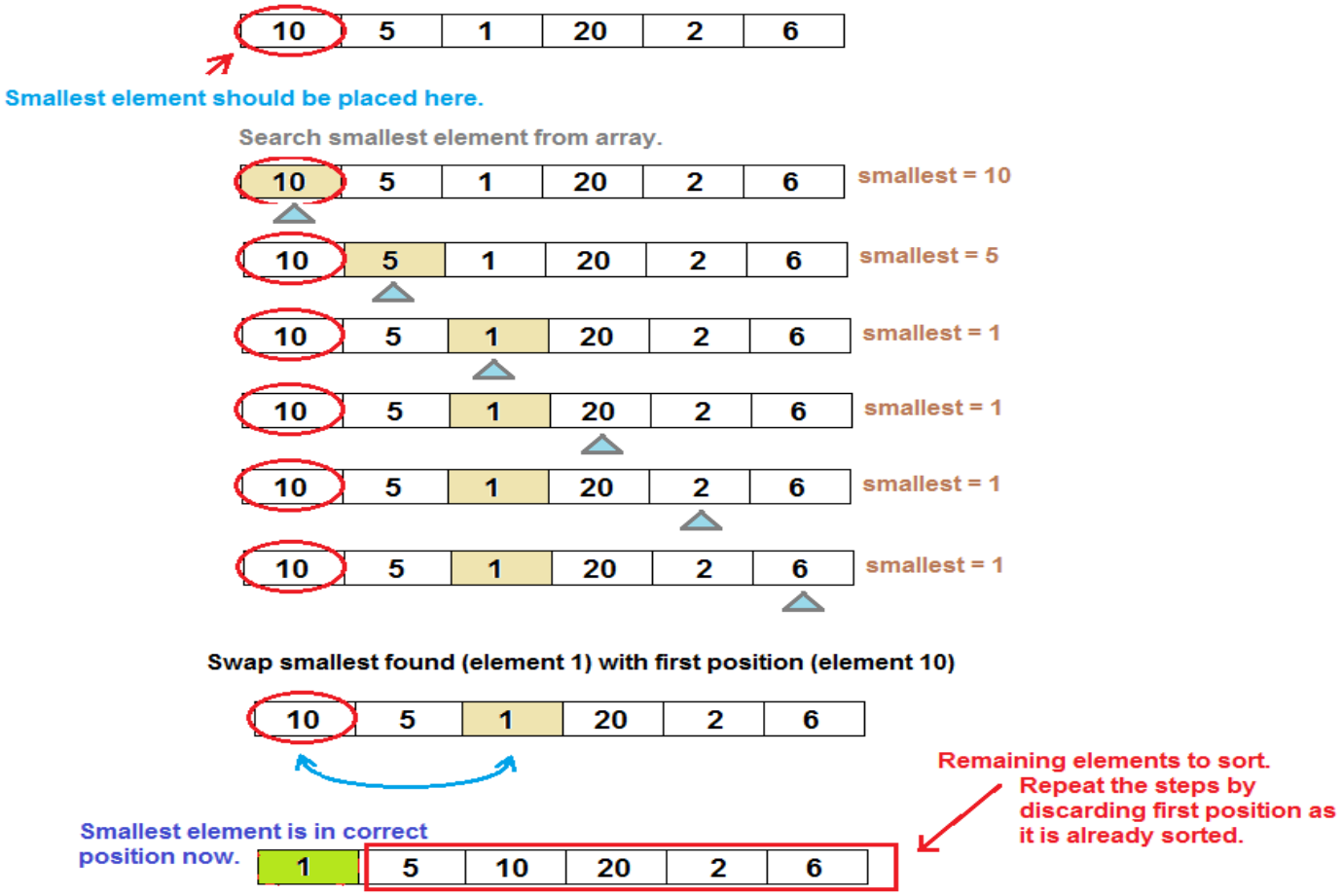
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,lrg,i,j,k,p,digcnt,divsr,r;
int bucktcnt[10],buckt[10][10];
clrscr();
printf("\nEnter array size:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter array elements a[%d]:",i);
scanf("%d",&a[i]);
}
i=0;
lrg=a[i];
while(i<n)/* find the largest element in array */
{
if(a[i]>lrg)
lrg=a[i];
i++;
}
/*count the no of digit in the largest no*/
digcnt=0;
while(lrg>0)
```

```
{
digcnt++;
lrg=lrg/10;
}
i=1;
divsr=1;
while(i<=digcnt)
{
j=0;
while(j<10)
{
bucktcnt[j]=0;
j++;
}
j=0;
while(j<n)
{
r=(a[j]/divsr)%10;
buckt[r][bucktcnt[r]]=a[j];
bucktcnt[r]++;
j++;
}
/*collect all elements in order*/
j=0;
p=0;
while(j<10)
{
k=0;
while(k<bucktcnt[j])
{
a[p]=buckt[j][k];
p++;
k++;
}
```

```
j++;
}
i++;
divsr=divsr*10;
}
printf("\nSorted Elements:");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
```

❖Selection sort

Selection Sort algorithm is a simple sorting algorithm which specially is an in-place comparison sorts. It is a technique to arrange the data in proper order.
This type of sorting is called “Selection sort” because it works by repeatedly selecting smallest element.
If we want to sort array in increasing order(i.e smallest element at the beginning of the array and the largest element at the end.) then find the minimum element and place it in the first position (recursion).



Algorithm for Selection sort

Step 1: Input n number of elements in array a.
Step 2: Initialize i TO 0 (i=0)
Step 3: Repeat through step 8 while i<n-1(i=0,1,2….,n-1)
Step 4: Initialize min TO i (min=i)
Step 5: Initialize j=i+1
Step 6: Repeat through j=j+1 while j<n(j=i+1,i+2,….)
If(a[j]<a[min])
min = j (min=j)
Step 7: if(min!=i)
temp = a[i]
a[i] = a[min]
a[min] = temp

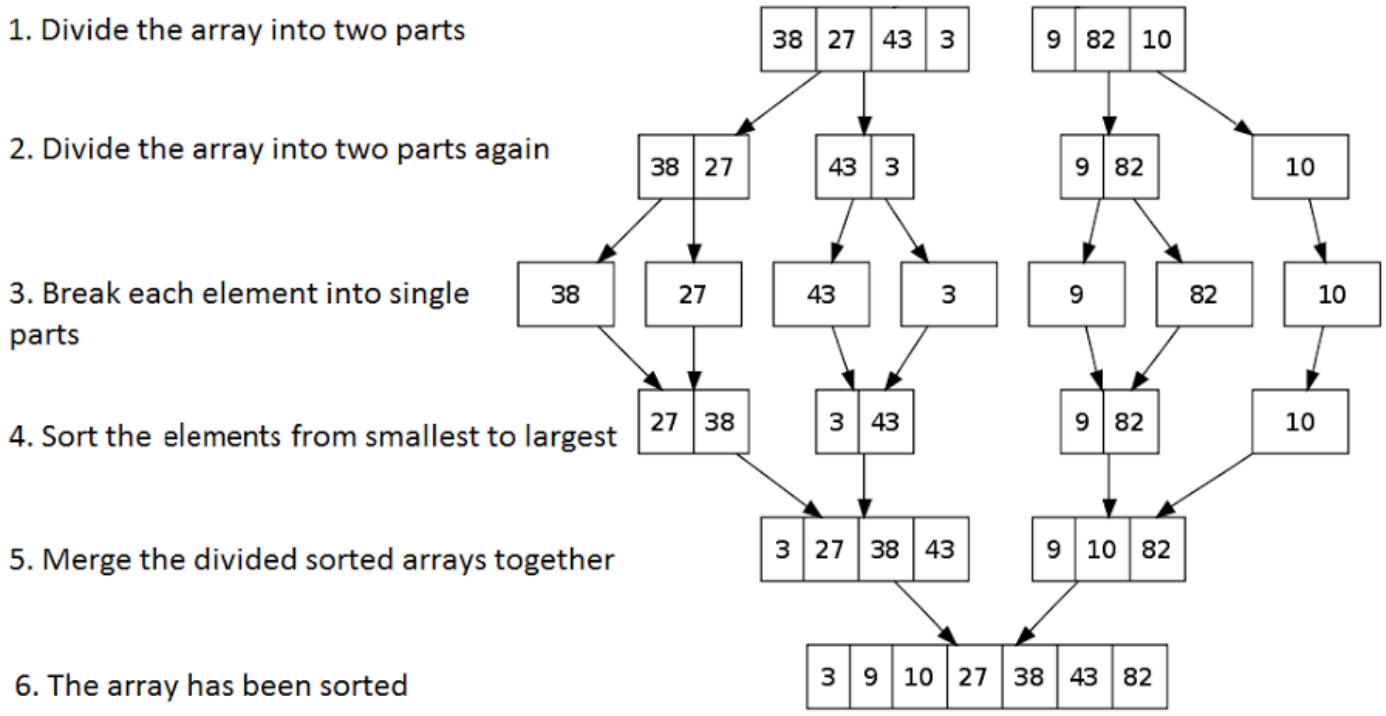
Step 8: i=i+1
Step 9: Exit

Program for Selection sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,i,j,temp,min;
clrscr();
printf("\nEnter array size:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter array elements a[%d]:",i);
scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)
{
min=i;
for(j=i+1;j<n;j++)
{
if(a[j]<a[min])
min=j;
}
if(min!=i)
{
temp=a[i];
a[i]=a[min];
a[min]=temp;
}
}
printf("\n\nSorted array:");
for(i=0;i<n;i++)
```

```
{
printf("\t%d",a[i]);
}
getch();
}
```

❖Merge sort



It is widely used sorting technique which uses divide & conquer mechanism.

- In this type of method the problems which can be broken into smaller problems, solve the smallest problems and then merge them to get the final answer. It means that we continue dividing till only one element is left.
- In this techniques, our large list is broken down into smaller lists and then after merge together.
- It is the process of combining two or more sorted array into third sorted array.

- When we are using these techniques different lists available are:

1. First divide the list into half
2. Then sort the left half
3. Then sort the right half
4. Then merge the two sorted halves into one sorted list.

Algorithm for Merge sort

Mergesort(n,list1,m,list2);
n:- represent number of elements in first list
list1:- represent list of elements (first list)
m:- represent number of elements in second list

list2: represent list of elements (second list)
Step 1: first, the array is divided into two parts. i.e. mid is determined between low index and high index.
midsort(low, high);
low: low index
high: high index
mid=(low+high)/2
step 2: then first part(from low to mid) and second part (mid+1 to high) are sorted by calling the function midsort
midsort(low,mid);
midsort(mid+1,high)
step 3: then, above two sorted parts are merged by calling mergesort function
step 4: initialize i low, j mid+1, k high(low)
step 5: repeat this step till i<=mid and j<=high
if(a[i]>=a[j])
temp[k]=a[j];
k++;
j++;
else
temp[k]=a[i];
k++;

i++
step 6: repeat this step till i<=mid.
temp[k]=a[i];
k++;
i++;
step 7: repeat this step till j<=high
temp[k]=a[j];
k++;
j++;
step 8: repeat this step till i<=high
for(i=low;i<=high;i++) copying final array to a
a[i]=temp[i]
step 9: STOP

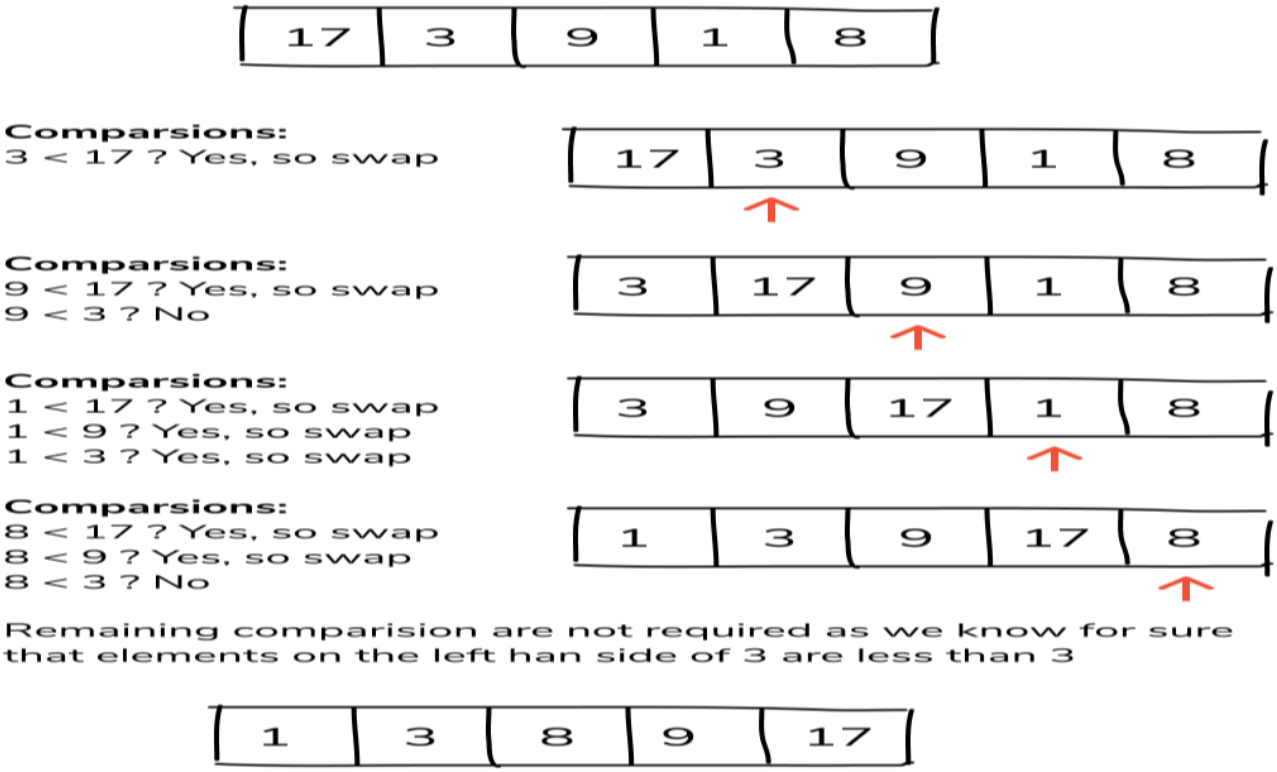
Program for Merge sort

```
#include<stdio.h>
#include<conio.h>
int a[100];
void m_sort(int,int);
void merge_sort(int,int,int);
void main()
{
int n,i;
clrscr();
printf("\nEnter array size:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter array elements a[%d]:",i);
scanf("%d",&a[i]);
}
printf("\n\nUnsorted array");
for(i=0;i<n;i++)
```

```
{
printf("\t%d",a[i]);
}
m_sort(0,n-1);
printf("\n\nSorted array");
for(i=0;i<n;i++)
{
printf("\t%d",a[i]);
}
getch();
}
void m_sort(int low,int high)
{
int mid;
if(low!=high)
{
mid=(low+high)/2;
m_sort(low,mid);
m_sort(mid+1,high);
merge_sort(low,mid,high);
}
}
void merge_sort(int low,int mid,int high)
{
int i,j,k,temp[100];
i=low;
j=mid+1;
k=low;
do
{
if(a[i]>=a[j])
temp[k++]=a[j++];
else
temp[k++]=a[i++];
```

```
}while((i<=mid)&&(j<=high));
while(i<=mid)
temp[k++]=a[i++];
while(j<=high)
temp[k++]=a[j++];
for(i=low;i<=high;i++)
{
a[i]=temp[i];
}
}
```

❖Shell sort



Shell sort is introduced to improve the efficiency of simple insertion sort.

- Shell sort is also called diminishing increment sort

Algorithm for Shell sort

Step 1: Input n number of elements in array a.
Step 2: Initialize i= 0 and repeat through step 6 if(i<x)
Step 3: span=incr[i]

Step 4: Initialize j=span and repeat through step 6 if (j<n)
temp=a[j]
Step 5: Initialize k=j-span and repeat through step 5 if(k>=0) and (temp<a[k])
a[k+span]=a[k]
Step 6: a[k+span]=temp
Step 7: Exit

Program for Shell sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100],n,i,num,k,j;
clrscr();
printf("\nEnter array size:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter array elements a[%d]:",i);
scanf("%d",&a[i]);
}
printf("\nEnter maximum number (odd value):");
scanf("%d",&num);
while(num>=1)
{
for(j=num;j<n;j++)
{
k=a[j];
for(i=j-num;i>=0 && k<a[i];i=i-num)
a[i+num]=a[i];
a[i+num]=k;
}
```

```
printf("\nIncrement=%d\n",num);
for(i=0;i<n;i++)
printf("%d\t",a[i]);
printf("\n");
num=num-2;
}
printf("\nSorted array");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
getch();
}
```