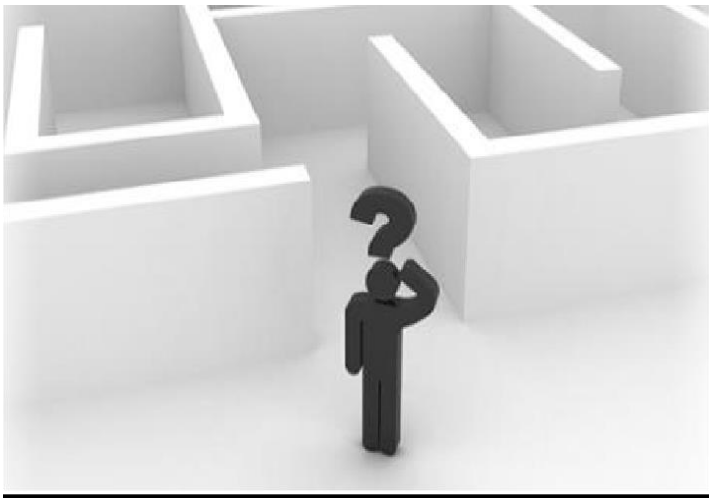
UNIT -2 – PART -1 - SORTING & SEARCHING



- 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

- Sorting is important!
- Things that would be much more difficult without sorting:
  - finding a phone number in the phone book
  - looking up a word in the dictionary
  - finding a book in the library
  - buying a cd/dvd
  - renting a video
  - buying groceries

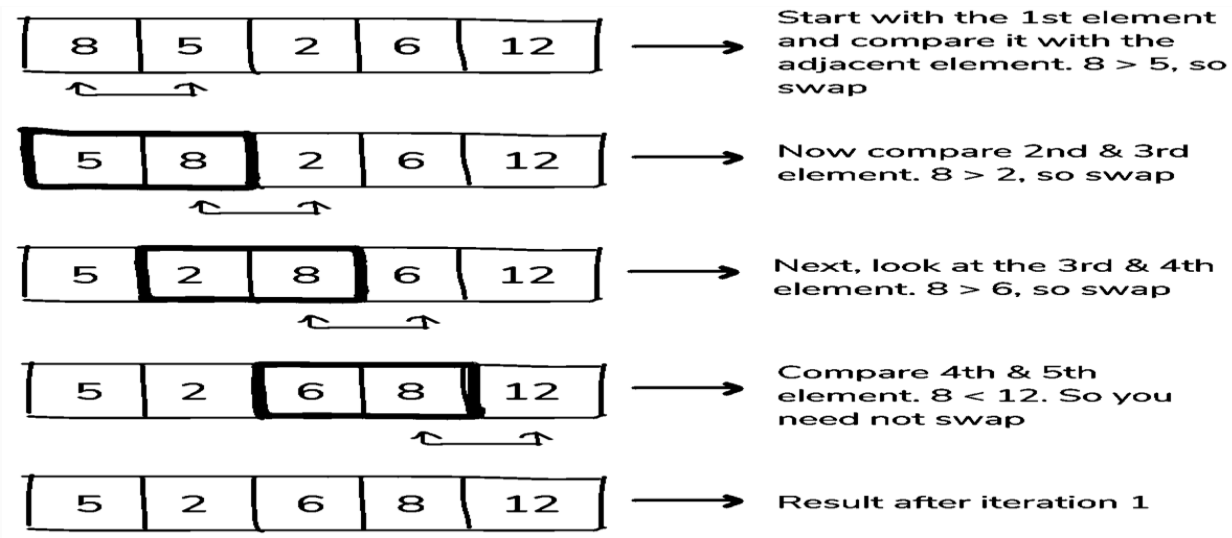
- The operation is the most common task performed by computers.
- Sorting is the process of arranging data information in some logical order.
- This logical order may be ascending or descending in case of numeric values.
- Various techniques are available to sort data depending on length of data, speed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

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 arrange N elements of array by placing the biggest element on proper position.



- It is the simplest sorting algorithm techniques.
- In this technique, we continually compare two adjacent items (elements) from the list. If the first element is larger than the second one, then the position of the elements are interchanged (swap) otherwise not changed and then the after sorting is completed. This process is used frequently until no swaps are needed.
- It is also known as "**comparison sort**" because it continually compares two adjacent elements from the list.

Advantages & Disadvantages of Bubble sort

- One of the primary advantages of the bubble sort is that it is comparatively easy to write and learn.

- It is also comparatively easy to understand in terms of sorting algorithm.
- Unfortunately, the bubble sort is also a relatively slow algorithm, taking $O(n^2)$ to complete sorting and therefore, should not be used on large tables.

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

Index: 0 1 2 3 4

8	7	5	6	2

EG:-

step 1: **8 7** 5 6 2
7 **8 5** 6 2
7 5 6 **8** 2
5 7 6 2 **8**

Step 2: 7 5 6 2
5 7 6 2
5 6 7 2
5 6 2 **7**

Step 3:

5 6 2
5 2 **6**

Step 4:

5 2
2 **5**

Program for Bubble sort

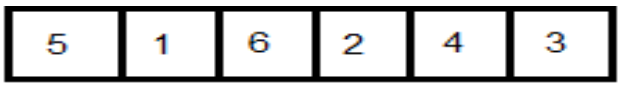
```
#include<stdio.h>
#include<conio.h>
void bubble_sort(int[],int);
void main()
{
int a[30],n;
int i;
clrscr();
printf("enter size of array:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
        printf("\nenter element %d:",i+1);
        scanf("%d",&a[i]);
}
bubble_sort(a,n);
getch();
}
void bubble_sort(int a[],int num)
{
        int i,j,temp;
        for(i=0;i<num;i++)
        {
                for(j=0;j<num-1;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
printf("\n sorted list:\n");
for(i=0;i<num;i++)
{
        printf("%d\n",a[i]);
}
}
```

SR.NO.	QUESTION	ANSWER
1	WHAT IS AN EXTERNAL SORTING ALGORITHM?	ALGORITHM THAT USES TAPE OR DISK DURING THE SORT
2	WHAT IS AN INTERNAL SORTING ALGORITHM?	ALGORITHM THAT USES MAIN MEMORY DURING THE SORT
3	WHAT IS THE WORST CASE COMPLEXITY OF BUBBLE SORT?	O(N2)
4	WHAT IS THE AVERAGE CASE COMPLEXITY OF BUBBLE SORT?	O(N2)
5	THE GIVEN ARRAY IS ARR = {1, 2, 4, 3}. BUBBLE SORT IS USED TO SORT THE ARRAY ELEMENTS. HOW MANY ITERATIONS WILL BE DONE TO SORT THE ARRAY?	4
6	WHAT IS THE BEST CASE EFFICIENCY OF BUBBLE SORT IN THE IMPROVISED VERSION?	O(N)

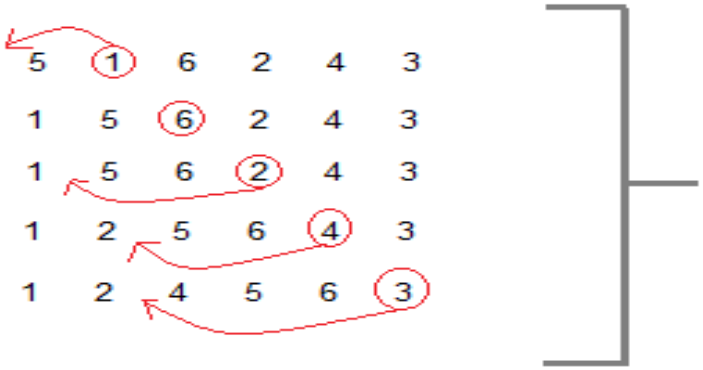
Insertion sort

Insertion sort arrange N elements of array by inserting particular items in a particular such a way that the items are in sorted order.

It is very simple and efficient algorithms for the smallestlists.



Lets take this Array.



(Always we start with the second element as key.)

As we can see here, in insertion sort, we pick up a key, and compares it with elemnts ahead of it, and puts the key in the right place

5 has nothing before it.

1 is compared to 5 and is inserted before 5.

6 is greater than 5 and 1.

2 is smaller than 6 and 5, but greater than 1, so its is inserted after 1.

And this goes on...

- 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.
- Step 1: Input n numbers for an array "a".
  Step 2: Initialize i=1 and repeat through step 4 if(i<n)
  Step 3: Repeat the step 3 (j=i;j>=1;j--)

```
if(a[j-1]>a[j])
    temp=a[j-1]
    a[j-1]=a[j]
```

Step 4: a[j]=temp
Step 5: Exit.

EG:

7	6	3	4	1

7 6 3 4 1
6 **7 3** 4 1
6 3 7 4 1
3 6 **7 4** 1
3 **6 4** 7 1
3 4 6 **7 1**
3 4 **6 1** 7
3 **4 1** 6 7
3 1 4 6 7
1 3 4 6 7

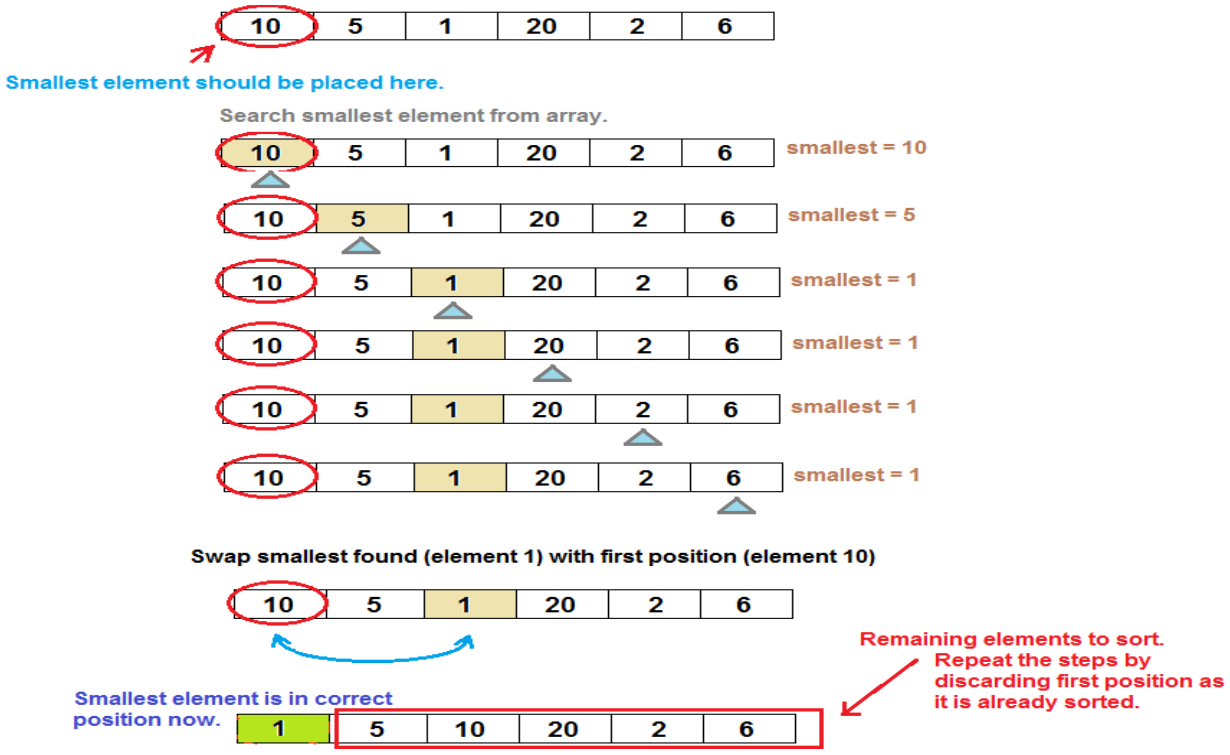
Program for insertion sort

```
#include<stdio.h>
#include<conio.h>
void ins_sort(int[],int);
void main()
{
        int *a,size,i;
        clrscr();
        printf("enter limit of elements");
        scanf("%d",&size);
        for(i=0;i<size;i++)
        {
                printf("enter element %d:",i);
                scanf("%d",a+i);
        }
        ins_sort(a,size);
        getch();
}
void ins_sort(int a[],int n)
{
        int i,j,temp;
        for(i=1;i<n;i++)//1<5
        {
                for(j=i;j>=1;j--) //j=1;1>=1;
                {
                        if(a[j-1]>a[j]) //a[1-1=0]>a[1]
                        {
                                temp=a[j-1]; //a[0]
                                a[j-1]=a[j]; //a[0]= a[1]
                                a[j]=temp; //a[1]=a[0]
                        }
                  }
          }
                printf("sorted list:\n");
        for(i=0;i<n;i++)
        {
```

```
            printf("%d \n",a[i]);
        }
}
```

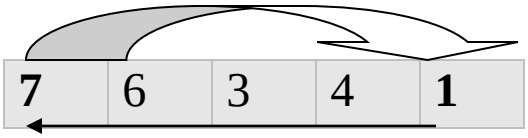
Selection sort

Selection sort arrange N elements of array by placing the smallest Item in proper position incase of ascending order arrangement.



- 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).

Example:



7	6	3	4	**1**

1	**6**	**3**	4	**7**

1	**3**	**6**	4	7

1	3	4	6	7

How to work:

Step 1: Begin
Step 2 : find the smallest element of array.
Step 3: swape the smallest number with the 1st number in list.
Step 4: repeat for the rest.
Step 5: swape again with newly founds elements.
Step 6 : keep repeating.
Step 7 : done sorting.
Step 8 : Exit.

15	28	17	12	18	9	**6**

➔ Find smallest then → swap

6	**28**	17	12	18	**9**	15

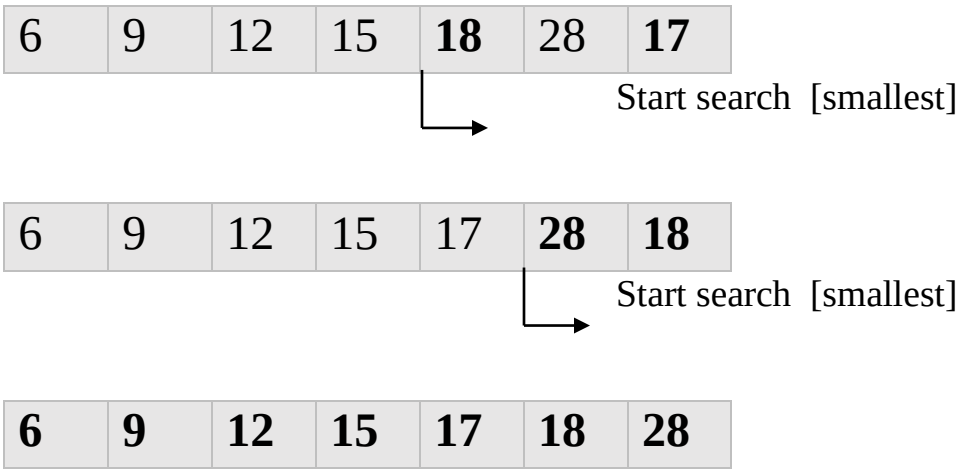
Start search [smallest]

6	9	**17**	**12**	18	28	15

Start search [smallest]

6	9	12	**17**	18	28	**15**

Start search [smallest]



1.Program for Selection sort :

```
#include<stdio.h>
#include<conio.h>
void sel(int[])
void main()
{
int a[5],i;
clrscr();
printf("\nEnter 5 array element:");
for(i=0;i<5;i++)
{
        Printf(“enter element %d”,i);
        scanf(“%d”,&a[i]);
        }
        sel(a);
        getch();
        }

        void sel(int a[])
        {
                int i,j,temp;
        for(i=0;i<=5;i++)
        {
```

```
            for(j=i;j<5;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
        }
        }
}
        printf("\n\nSorted array:");
         for(i=0;i<5;i++)
        {
        printf("\t%d",a[i]);
        }
```

---

2. 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)
{
```

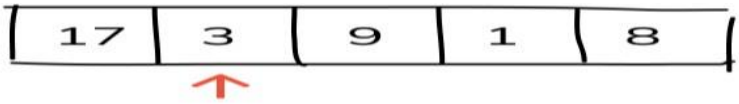
```
temp=a[i];
 a[i]=a[min];
a[min]=temp;
}

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

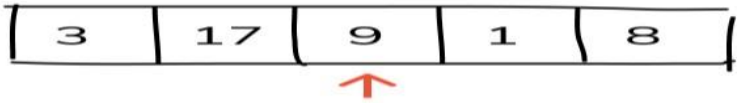
Quick sort:

Shell short:

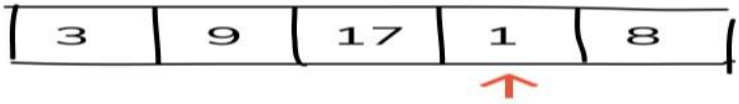
Comparsions:
3 < 17 ? Yes, so swap



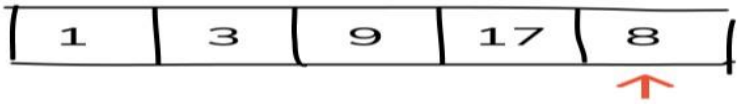
Comparsions:
9 < 17 ? Yes, so swap
9 < 3 ? No



Comparsions:
1 < 17 ? Yes, so swap
1 < 9 ? Yes, so swap
1 < 3 ? Yes, so swap



Comparsions:
8 < 17 ? Yes, so swap
8 < 9 ? Yes, so swap
8 < 3 ? No



Remaining comparision are not required as we know for sure that elements on the left han side of 3 are less than 3

- Shell sort is introduced to improve the efficiency of simpleinsertion sort.
- Shell sort is also called **diminishing increment sort**
- Based on insertion sort algoritham.
- Compare elements that are distant apart rather than adjacent.
- Spacing between elements is known as Gap/Interval.
- In every pass, Gap is reduced by 1 till we reach last pass when gap is 1.

Important Point/ Formula:-

Gap=Floor(N/2), where N= no of elements in array.

Example:-

0	1	2	3	4	5	6	7	8	9
19	**63**	**2**	**6**	**7**	**18**	**60**	**16**	**9**	**4**

N=10

Gap(pass1)=floor(n/2)

=Floor(10/2)

=Floor(5) , So gap_1 =5

a[0]=a[5] , a[1]=a[6], a[2]=a[7], a[3]=a[8], a[4]=a[9]

now,

+5compare

Gap(pass1)=floor(gap/2)

=Floor(5/2)

=Floor(2.5),So gap_2 =2

0	1	2	3	4	5	6	7	8	9
18	60	**2**	6	**4**	19	**63**	16	**9**	7

2	6	**4**	7	**9**	16	**18**	19	**63**	60

Gap/2

2/2=1

2	**4**	**6**	**7**	**9**	**16**	**18**	**19**	**60**	**63**

Program For shell sort:

```
#include <stdio.h>
#include<conio.h>
void shell_sort(int a[], int num);
void main()
 {
   int i, n, x[100];
   printf("Enter the size of elements: ");
   scanf("%d", &n);
   printf("Enter elements:\n");
   for (i = 0; i < n; i++)
    {
                scanf("%d", &x[i]);
   }
   shell_sort(x, n);
printf("Sorted list:\n");
   for (i = 0; i < n; i++)
   {
          printf("%d\t", x[i]);
   }
 getch();
}
```

```
void shell_sort(int a[], int num)
{
   int i, j, temp, k;
   for (i = num / 2; i > 0; i = i / 2)
    {
              for (j = i; j < num; j++)
             {
                  for (k = j - i; k >= 0; k = k - i)
                   {
                                        if (a[k + i] >= a[k])
                                           break;
                                        else
                                        {
                                           temp = a[k];
                                           a[k] = a[k + i];
                                           a[k + i] = temp;
                                        }
                            }
               }
       }
}
```

Algorithm for Shell sort:

shell (a, n) where, a=represent the list of a(0) to (n-1) n=represent number of element in the list.

Step1:- [initialize] a[0]=0

Step2:- repeat through step -3 to for i=num/2.j=i,k=j-i.....n-1

Step3:- if (a[k + i] >= a[k])

break;

else

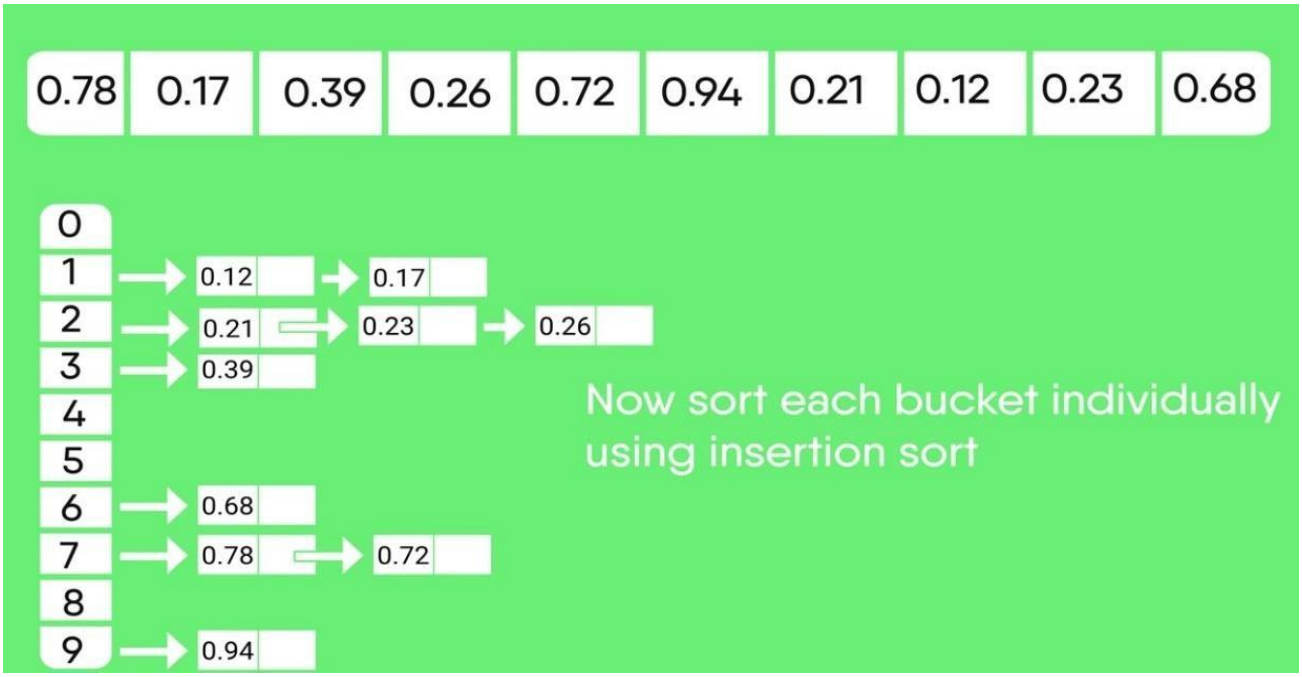
temp = a[k];

a[k] = a[k + i];

a[k + i] = temp;

Step4:- Exit

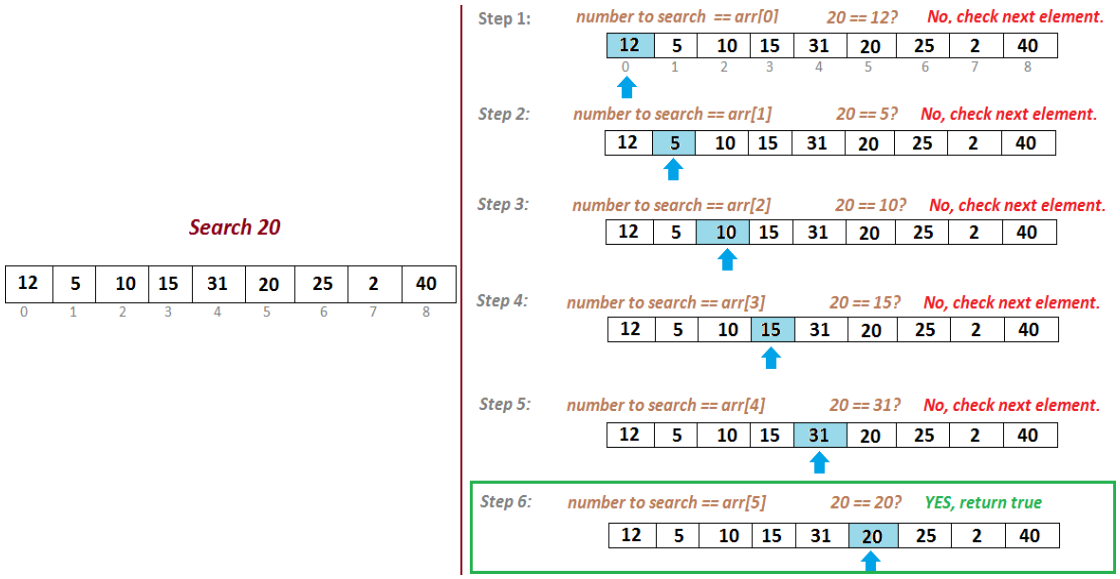
Bucket sort:



- Bucket sort is a sorting method that can be used to sort a listof 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 need10 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 passesare required.
- Mechanism includes comparison of the first position of digitwith the digit of bucket and place it. (recursive)

SEARCHING METHOD

- Traversing an array to locate a particular item
- It requires the user to specify the target item
- If the target item is found, its index is returned
- If the target item is NOT found, -1 is returned
- Two searching algorithms
  - Linear Search (works with any array)
  - Binary Search (works if the searched array is sorted)



1. Sequential searching

- To search (locate, find) an element from the unsorted array list weare using this simplest technique.
- It simply traverses from top to bottom in the array and searches forthe key value from the list and displays output as well. It is called sequential searching method.

Algorithm for sequential searching

- a represents array
- n represent number
- of elements in arrayele represents element to be searched

Step 1: Initialize flag=0

Step 2: Initialize i=0 and repeat till if(i<n)
Get array elements (&a[i])

Step 3: Repeat step 4 for i=0,1,2….n-1

Step 4: if(a[i]==ele)
Output "successful searching"
flag=1

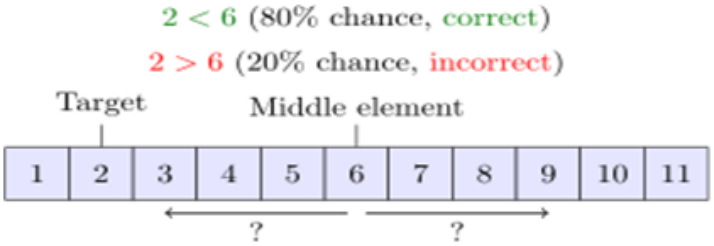
Step 5: if flag==0
Output "unsuccessful search"

Step 6: STOP

Program for sequential searching

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[100],n,ele,i,flag=0;
        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 the element you want to search:");
        scanf("%d",&ele);
        for(i=0;i<n;i++)
        {
                if(a[i]==ele)
                {
                        printf("\nYour element %d is at position %d",ele,i);
                        flag=1;
                }
        }
        if(flag==0)
                printf("\nNo such element avaialable in array");
        getch();
}
```

2. Binary searching



Goal: Search for the index of the target value so that there is a probability of p that the index is correct. p is specified before the search.

- The main characteristics of binary search is that it works only on sorted array lists and so it becomes easy to find any information very fast.

Algorithm for binary searching

a ➡ represents array
n ➡ represent number of elements in array
ele ➡ represents element to be searched

Step 1: Initialize flag=0,start:
Step 2: Initialize i=0 and repeat through step 3 till if(i<n)
Get array elements (&a[i])
Step 3:if(i>0)
if(a[i-1]>a[i])
break;
goto start; //label
Step 4: Repeat step 4 for i=0,1,2….n-1
Step 5:if(a[i]==ele)
Output "successful searching"
flag=1
Step 6: if flag==0
Output "unsuccessful search"
Step 7: STOP

Program for binary searching

```
#include <stdio.h>
#include <conio.h>

void main() {
   int a[100], n, i, ele, flag = 0;
   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]);
      if(i > 0) {
         if(a[i - 1] > a[i]) {
            printf("\nnumber should be greater than the previous value");
            getch();
            break;
         }
      }
   }
   printf("\nEnter element to be searched :");
   scanf("%d", &ele);
   for(i = 0; i < n; i++) {
      if(a[i] == ele) {
         printf("\nYour element %d is at position %d", ele, i);
         flag = 1; // Corrected line
      }
   }
   if(flag == 0)
      printf("\nElement not found");
   getch();
}
```

How to work Binary Search:-

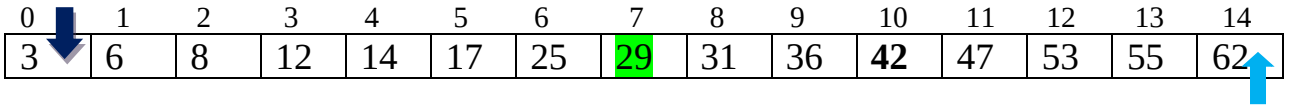
1.Define start & end.

2.find the middle element = [start+end]/2

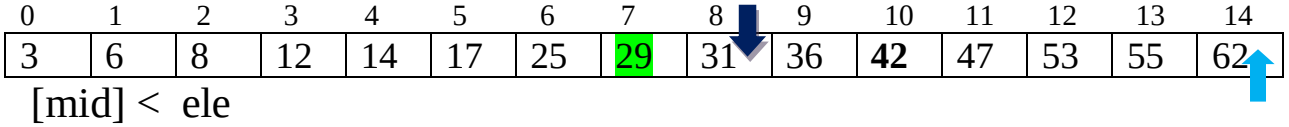
3.if a[mid] < ele
So, search to the right of the middle element , like
Start= mid+1 , end = end.

4.if a[mid] > ele
So, search to the left of the middle element , like
Start= start , end = mid-1

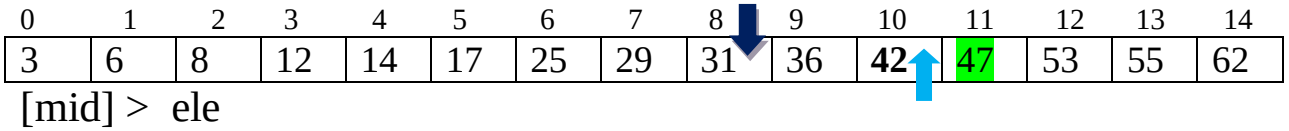
Search 42



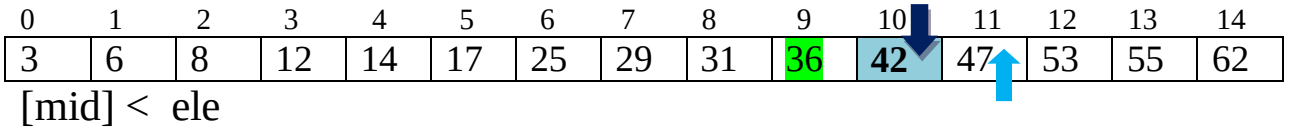
Middle= [start+end]/2 0+14 / 2 = **7**



[mid] < ele
Start= mid+1 =7+1 = **8** end= **14**
8+14 / 2 **= 11**



[mid] > ele
Start = **8** end = mid-1 11-1 = 10
8 + 10 / 2 = **9**



[mid] < ele
Start= mid+1 =9+1 = **10** end= **11**
10+11 / 2 **= 10.5 = 10**

So 10 is floor value consider