Searching :- To search given value in the list of existing elements, called searching.

There are two types of searching

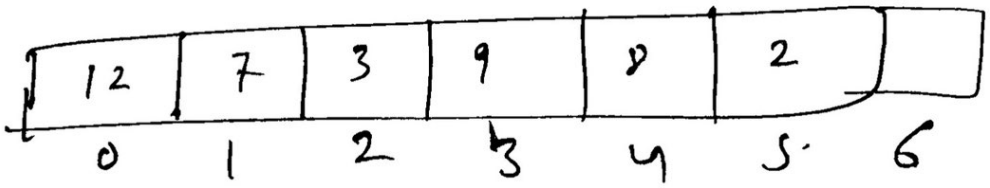
1) Linear (Sequential) search
2) Binary search

Linear Search :- ① To search given element one by one in the list of existing elements, ~~called linear~~ if element found then stop searching

2) Works on sorted & unsorted array both

3) It is ~~for~~ slow compare than Binary search

Array size : 7




search value 9

Found at index no 3

8 6

Not found

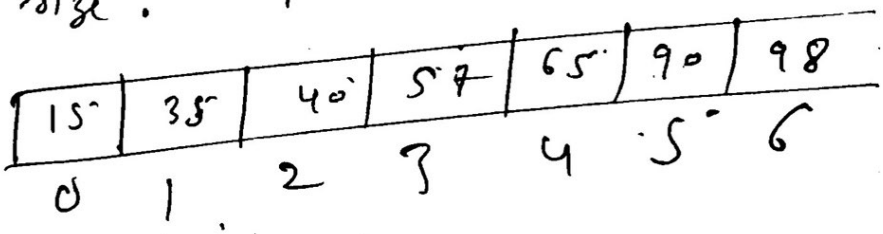
```
class Arr
{
static int LSearch (int a[],
                    int item)
{
  int p = -1;
  int i;
for ( i = 0; i < a.length; i++)
{
  if (a[i] == item)
    { p = i;
      break;
    }
}
return p;
}
}

class T
{
  p s v m (String K[])
  {
    Scanner ob = new Scanner (System.in);
      Sopl ("enter Array size");
```

```
int N = ob.nextInt();
int a[] = new int[N];
SoPL("enter values");
int i;
for(i=0; i<N; i++)
{
a[i] = ob.nextInt();
}
SoPL("enter search value");
int t = ob.nextLine();
int r = Arr.LSearch(a,t);
if(r==-1)
SoPL("element Not found");
else
SoPL("element found at index No "+r);
}
}
```

## Binary Search

1) It works on divide Conquer meth
2) It works only on Sorted Array
3) It is fast compare than Linear Search

Array size :- 7




Search value (57)

```
b = 0
e = 6
mid = (0+6)/2 => (3)

if (search value == Array [mid])
  { p = mid;
    break;
  }
else if (search value > Array[mid])
  { b = mid+1;
  }
else
  { e = mid-1;
```

```
class Arr
{
static in BSearch (in a [], in item)
{
 in p = -1, b = 0, mid;
 in l = a.length - 1;
while ( b <= l)
{
  mid = (b+l)/2;
  if (item == a [mid])
  {
     p = mid;
     break;
  }
  else if (item > a [mid])
       { b = mid+1;
       }
    else
       { l = mid-1;
       }
}
return p;
}
```

## INSERTION :-

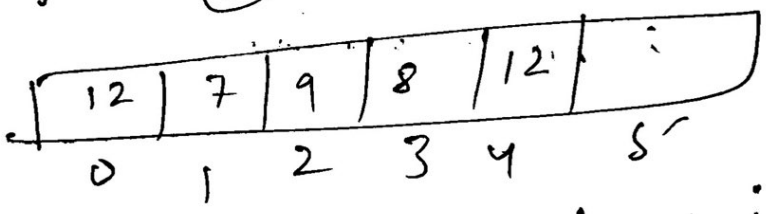
To insert a new element in the list of existing elements called insertion

1) insert At begining
2) Insert At last
3) insert At given index No

### Insert At beging

Actual Arr (6)

Array Siz : (5)




```
for ( i = a.length -1 ; i > 0 ; i--)
    { a[i] = a[i-1];
    }
    a[0] = new item;
```

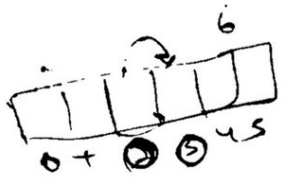
Insert At La

```
a[a.length-1] = new item
```

---

## Insert At Pos

```
for (i = a.length-1; i > pos; i--)
{
    a[i] = a[i-1];
}
a[pos] = new item
```



```
Class Arr
{
Static void insertAtBeg (int a[],
                         int item)
{
  int i;
for( i = a.length-1 ; i > 0 ; i--)
   { a[i] = a[i-1];
   }
   a[0] = item;
}
Static void insertAtLast (int a[],
                          int item)
{
   a[a.length-1] = item;
}
Static void insertAtPos (int a[],
                         int Pos, int item)
{
   int i;
   for (i = a.length-1 ; i>Pos ; i--)
      { a[i] = a[i-1];
      } a[Pos] = item;
}
```

```
class T
{
    Scanner ob = new Scanner (System.in);
    SOPL (" Enter Size of array");
    int N = ob.nextInt();
    int arr[] = new Int [N+1];
    SOPL (" Enter values");
    for (int i=0 ; i < N ; i++)
        arr[i] = ob.nextInt;
```

```
    SOPL ("enter New value that you want
                to insert");
    int t = ob.nextInt();
      Arr.insertAtBeg (a, t);
    SOPL ("New Array");
      for (int b:a)
        sop (b + " ");
```

```
    SOPL ("enter New value");
    int t = ob.nextInt();
      Arr.insertAtLast (a, t);
    SOPL ("New Array");
    for (int b : a)
      Sop (b + " ");
```

```
SOPL("Enter New Value");
in. t = obj.nextInt();
SOPL("Enter Position");
in pos = obj.nextInt();
    Arr.insertAtPos(a, pos, t);
    SOPL("New Array");
    for (in b : a)
      SOP(b + " ");
}}
```

# Deletion in an array

by index

1) Delete At beginning
2) Delet At given Position

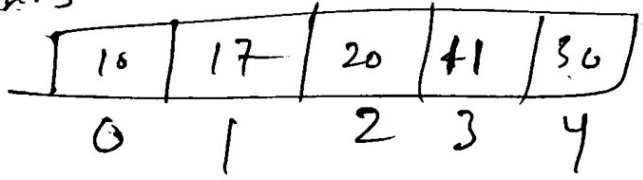
by given value

---

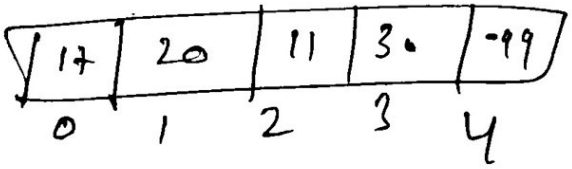
## by index

a) Delet at beginig

Array size: 5

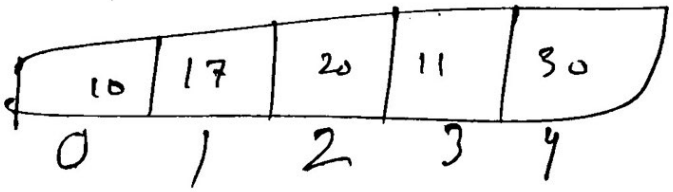



Down ward Shifting

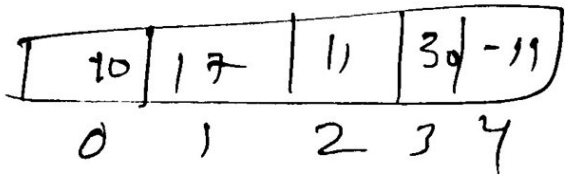



---

## Delet At Position




Pos = 2



```
class Arr
{
  void delAtbeg (int a[])
  {
    int i;
    for (i = 0; i < a.length - 1; i++)
    {
      a[i] = a[i+1];
    }
    a[a.length - 1] = -99;
  }
void delAtPos (int a[], int Pos)
  {
    for (i = Pos; i < a.length - 1; i++)
    {
      a[i] = a[i+1];
    }
    a[a.length - 1] = -99;
  }
}
```

```
class T
{
p s v m (String K[])
{
Scanner ob = new Scanner (System.in);
    SOPL ("enter Array size");
    int N = ob.nextInt();
int a[] = new int[N];
    SOPL ("enter values");
  for (i=0; i<N; i++)
  {
   a[i] = ob.nextInt();
  }
```

```
Arr.delAtBeg (int);
SOPL ("New Array");
for (i=0; i<a.length-1; i++)
    SOPL (a[i]);
```

```
SOPL("Enter Position from
        Where You want to Delete");
int Pos = ob.nextInt();
Arr.delAtPos(a, Pos);
SOPL("New Array");
for(i=0; i<a.length-1; i++)
    SOPL(a[i]);
}
}
```

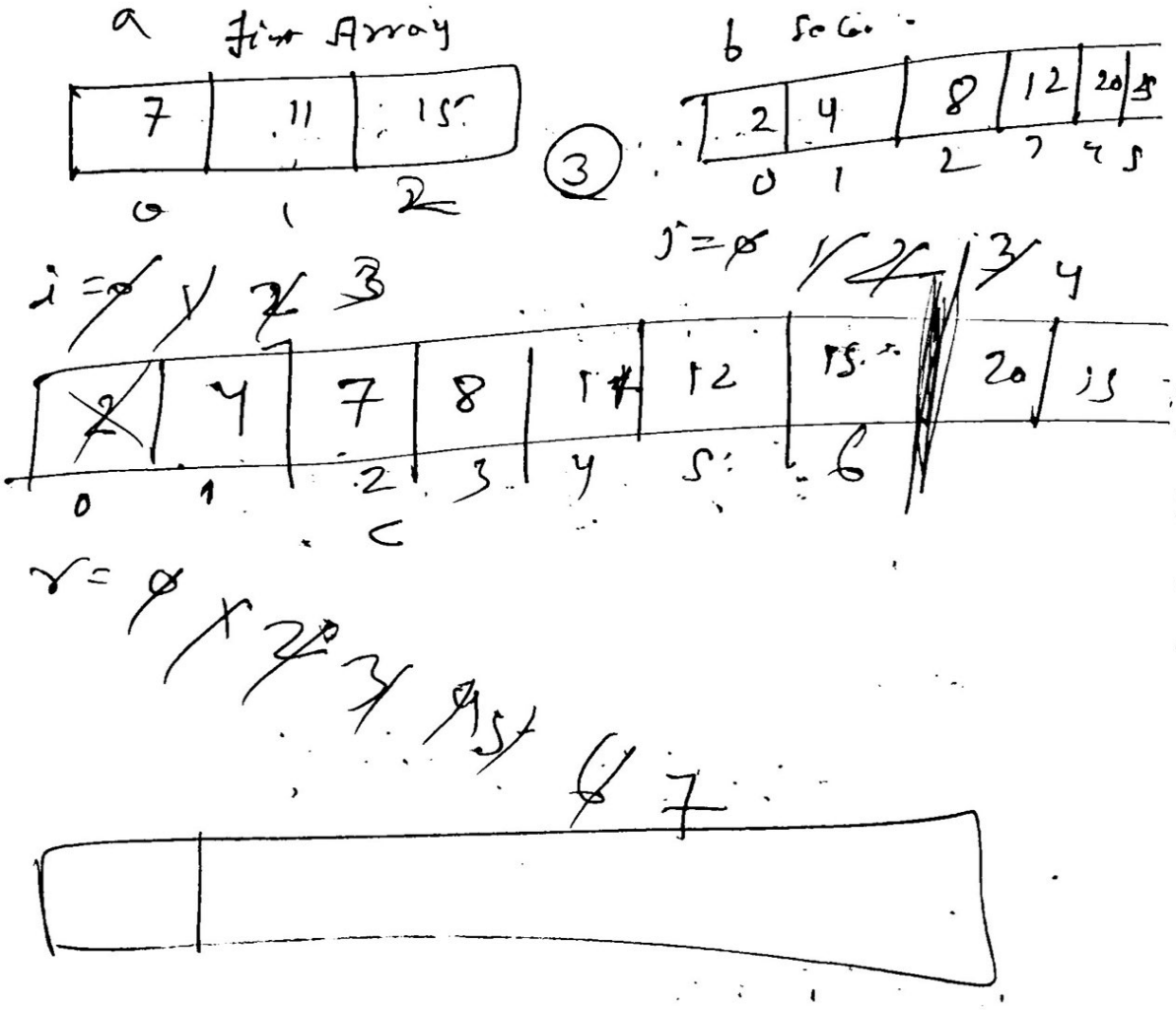
```
Class Avv
{
static int Position_Return (int a[], int item)
    { int Pos = -1;
      for (int i=0; i<a.length; i++)
       {
          if (a[i] == item)
              Pos = i;
       }
      return Pos;
    }

static void delete_at_index (int a[], int Pos)
 {

      for (int i=pos; i<a.length-1; i++)
       {
          a[i] = a[i+1];
       }

      a[a.length-1] = -99;
 }

}

Class main
{
      int a[] = {10,20,30,40};
   int p=avv.Position_return (30);
    if (P==-1)
       SOPL ("Element Not found");
    else
    {
       avv.deleteat_index (a, P);
       SOPL("Element deleted");
}   }
```

3 (3)

## Merging

To merge two existing list and make a New List

To merge two sorted array in sorted way



```
class Arr
{
static void merge(int a[], int b[],
                  int c[])
{
  int i=0, j=0, r=0;
while(i<a.length && j<b.length)
{
  if(a[i]<=b[j])
  {
    c[r++] = a[i++];
  }
else
  { c[r++] = b[j++];
} }
while(i<a.length)
  { c[r++] = b[i++];
}
```

```
while (j < b.length)
  {
    c[r++] = b[j++];
  }
}
}

class T
{ p s v m (String K[])
  {
    int a [] = {7, 12, 15, 30};
    int b [] = {2, 3, 5, 6, 8, 11,
                18, 19, 40, 80};
    int c [] = new int [a.length + b.length];

    Arr.merge (a, b, c);
    sop("New Array");
    for (int d : c)
      sop (d + " ");
  }
```

Sorting :- Sorting means arrange elements in a specific order either Ascending (A TO Z) or descending (Z TO A)

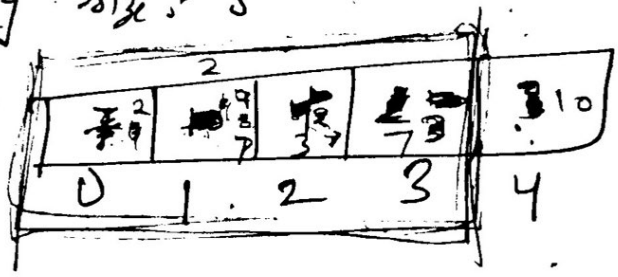
There are many types of Sorting.

1) Bubble Sort ✓
2) Selection " ✓
3) Insertion " ✓
4) Quick "
5) Merge "
6) Heap "

---

Bubble Sort :-

Array size :- 5



Pair 1

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 7 | 7 | 7 | 7 | 7 |
| 1 | 10 | 4 | 4 | 4 | 4 |
| 2 | 4 | 7 | 10 | 2 | 2 |
| 3 | 2 | 2 | 2 | 10 | 3 |
| 4 | 3 | 3 | 3 | 3 | 10 |

Pair 2

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 7 | 4 | 4 | 4 |
| 1 | 4 | 7 | 2 | 2 |
| 2 | 2 | 2 | 7 | 3 |
| 3 | 3 | 3 | 3 | 7 |

4 → 10

Pair 3

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 4 | 2 | 2 |
| 1 | 2 | 4 | 3 |
| 2 | 3 | 3 | 4 |

7
10

Pair 4

| | 0 | 1 |
|---|---|---|
| 0 | | |
| 1 | | |

×

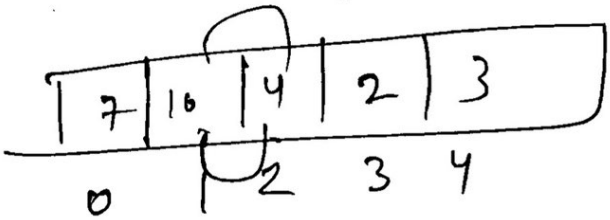
7 10 4 2 3

4 2 3 10

4 2 3 7 10

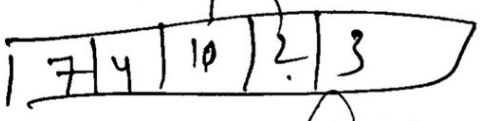
2 3 4 7 10

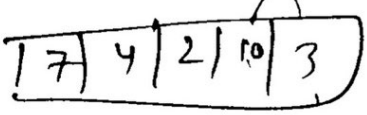
```
int i, j;
for (i=0; i<a.length-1; i++)
{
  for(j=0; j<a.length-1-i; j++)
  {
    if(a[j]>a[j+1])
    {
      int t = a[j];
      a[j] = a[j+1];
      a[j+1] = t;
    }
  }
}
```

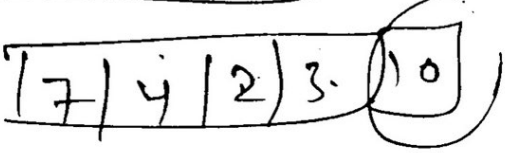



i
0

j
0
1
2
3




i
1
1

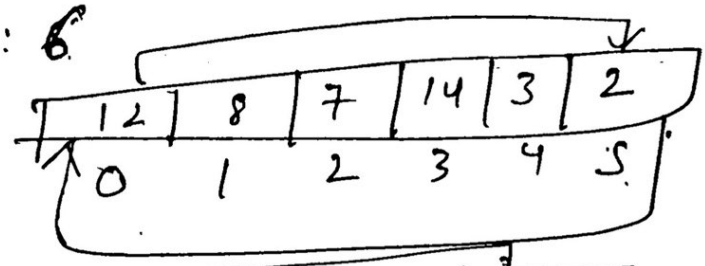
j
0
1
2

```
class Arr
{
.....
static void BSort(int a[])
{
    int i,j;
for(i=0; i<a.length-1; i++)
{
  for(j=0; j<a.length-1-i; j++)
  {
    if(a[j]>a[j+1])
    {
  int t=a[j];
  a[j]=a[j+1];
  a[j+1]=t;
    }
  }
}
}
```

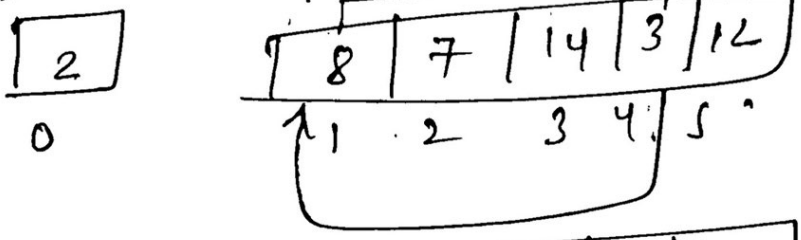
```
class T
{
p . s  v  m (String K[])
{

iw  a[] = { 10, 4, 7, 5, 4, 2, 3};
SOPL ("Original Array");
for (iw b: a)
 sop(b + " ");
Arr.BSort (a);
SOPL ("Sorted Array");
for (iw b: a)
 System.out.prn (b + " ");

}

}
```

Selection Sort :-

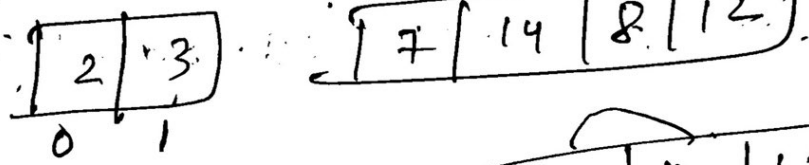
Array size : 6



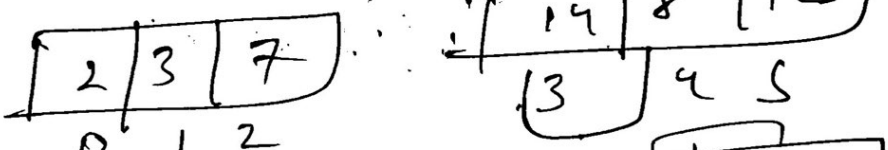

Pass 1



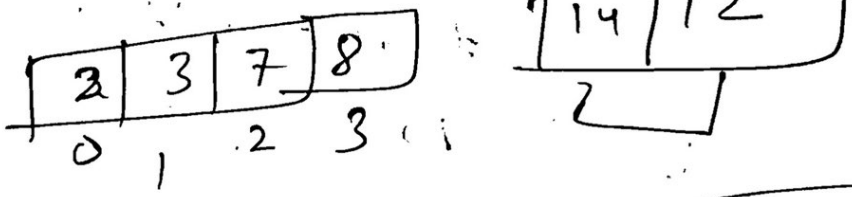

Pass 2




Pass 3




Pass 4




Pass