Unit 1

Arrays

	Syllabus	Guidelines	Suggested number of lectures
1	Single and multi-dimensional arrays, analysis of insert, delete and search operations in arrays (both linear search and binary search), Implementing sparse matrices	Chapter 7, Section 7.1, 7.2, 7.3, 7.4 (upto pg. no. 253) **Additional Resource 3**	12
2	Applications of arrays to sorting: selection sort, insertion sort, bubble sort, comparison of sorting techniques via empirical studies.	Chapter 9, Section 9.1.1--9.1.3, **Ref 1**	
3	Introduction to Vectors	Chapter 6 Section 6.1, **Ref 2**	

References

1. Ref 1: . Drozdek, A., (2012), *Data Structures and algorithm in C++.* 3rd edition. Cengage Learning. Note: 4th edition is available. Ebook is 4th edition

2. Ref 2.: Goodrich, M., Tamassia, R., & Mount, D., (2011). *Data Structures and Algorithms Analysis in C+ +.* 2nd edition. Wiley.

3. Additional Resource 3: Sahni, S. (2011). Data Structures, Algorithms and applications in C++. 2ndEdition, Universities Press

4. Additional Resource 4: Tenenbaum, A. M., Augenstein, M. J., & Langsam Y., (2009), *Data Structures Using C and C++.* 2nd edition. PHI.

Note: Ref1, Additional resource etc. as per the LOCF syllabus for the paper.

ch-4 Arrays and matrices

'S. Sohni'

(4.1, 4.2, 4.3).

(w.e.f 2012.)

Row-major mapping.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17 ([2][5])

Column major mapping

0	3	6	9	12	15
1	4	7	10	13	16
2	5	8	11	14	17.

for 3 X 6 ~~mat~~ matrix.

when row-major order is used, the mapping function is,

$$map(i_1, i_2) = i_1 u_2 + i_2$$

where u_2 is the no. of columns in the array.

eg:- $map(2,5) = 2.6 + 5 = 17$

$map(1,3) = 1.6 + 3 = 9.$

Col. major order : $map(i_1, i_2) = i_1 + i_2 . u_1$ → no. of rows.

for 3-d Array.

eg:- a[3][2][4]

[0][0][0], [0][0][1], [0][0][2], [0][0][3], [0][1][0], [0][1][1], [0][1][2], (0)(1)(3)

[1][0][0], [1][0][1], [1][0][2], [1][0][3], [1][1][0], [1][1][1], [1][1][2], (1)(1)(3)

[2][0][0], [2][0][1], [2][0][2], [2][0][3], [2][1][0], [2][1][1], [2][1][2], (2)(1)(3)

$$map(i_1, i_2, i_3) = i_1 u_2 u_3 + i_2 u_3 + i_3$$

special matrices

A square matrix has same no. of rows & columns.
special form of square matrices are:-

1) Diagonal:- M is diagonal iff $M(i,j) = 0$ for $i \neq j$.

$$\begin{array}{c} \\ i \end{array} \begin{array}{c} \begin{array}{cccc} j & & & \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \end{array}$$

2) Tridiagonal :- M is tridigonal iff $M(i,j) = 0$ for $|i-j| > 1$

$$\begin{array}{c} \\ i \end{array} \begin{array}{c} \begin{array}{cccc} j & & & \end{array} \\ \begin{bmatrix} 2 & 1 & 0 & 0 \\ 3 & 1 & 3 & 0 \\ 0 & 5 & 2 & 7 \\ 0 & 0 & 9 & 0 \end{bmatrix} \end{array}$$

3) Lower Triangular :- M is lower-Triangular iff $M(i,j) = 0$ for $i < j$.

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 4 & 2 & 7 & 0 \end{bmatrix}$$

4) upper Triangular :- M is upper Triangular iff $M(i,j) = 0$ for $i > j$.

$$\begin{bmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 3 & 8 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

5) <u>symmetric</u> :- Matrix M is symmetric iff ~~M(i,j)=M(j,i)~~

$M(i,j) = M(j,i)$ for all i & j.

$$\begin{bmatrix} 2 & 4 & 6 & 0 \\ 4 & 1 & 9 & 5 \\ 6 & 9 & 4 & 7 \\ 0 & 5 & 7 & 0 \end{bmatrix}$$

<u>Sparse Matrices</u>.

An $m \times n$ matrix is said to be sparse if "many" of its elements are zero.

A matrix that is not sparse is dense.

eg:- Diagonal & Tridigonal matrices are sparse.
each has $O(n)$ nonzero terms & $O(n^2)$ zero terms.

Triangular matrix (lower & upper) has —

atleast $\frac{n(n-1)}{2}$ zero terms &

atmost $\frac{n(n+1)}{2}$ non-zero terms.

The no. of non-zero terms in SPARSE matrices need to be less than $n^2/3$ & in some cases $n^2/5$.
In this context, triangular matrices are dense.

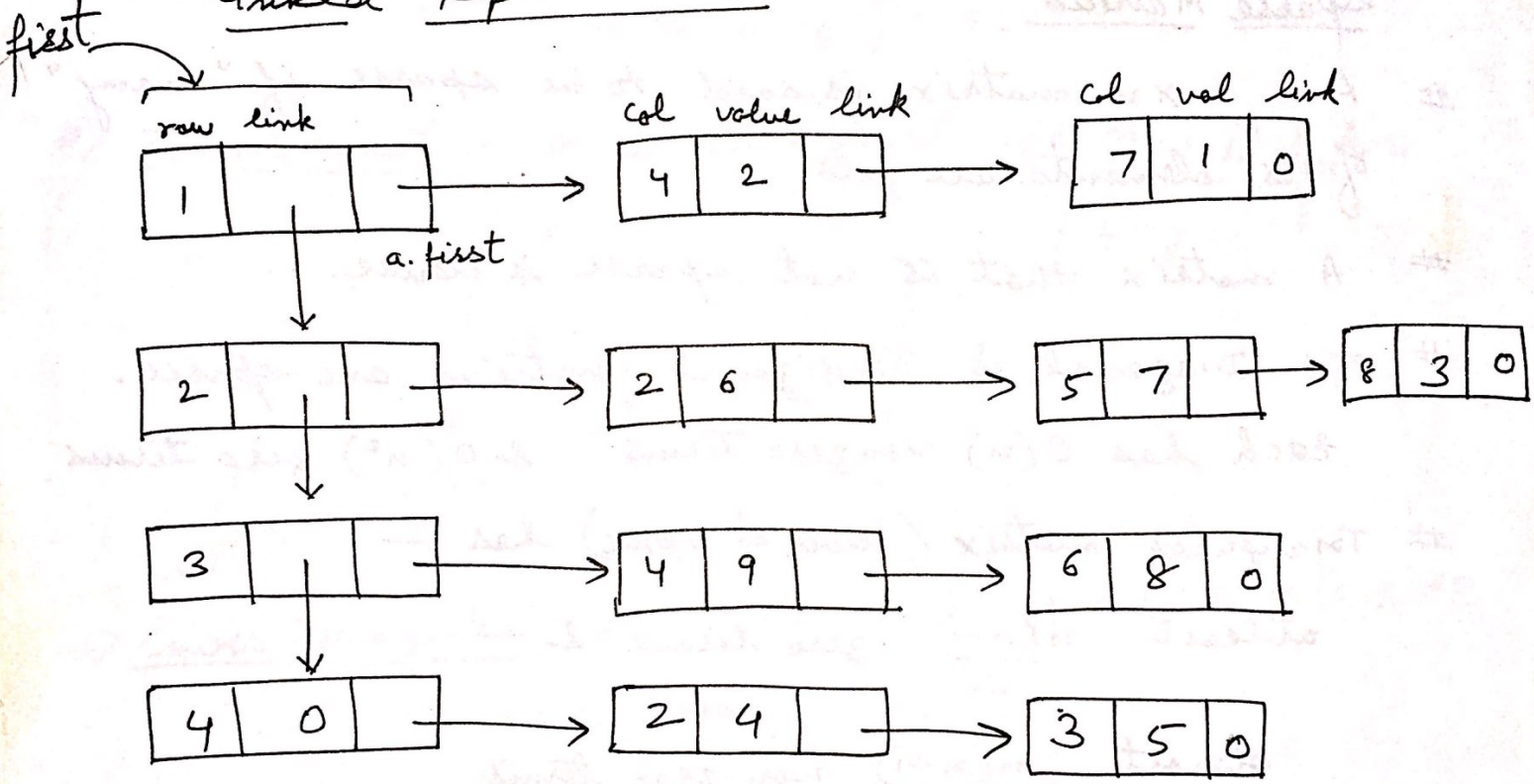
eg:- 4 × 8 sparse matrix

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 6 & 0 & 0 & 7 & 0 & 0 & 3 \\ 0 & 0 & 0 & 9 & 0 & 8 & 0 & 0 \\ 0 & 4 & 5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

its representation

a[]	0	1	2	3	4	5	6	7	8
row	1	1	2	2	2	3	3	4	4
col	4	7	2	5	8	4	6	2	3
value	2	1	6	7	3	9	8	4	5

Linked Representation



Diagonal Triangular matrix (using 1-D Array) eg $\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$

```
# include <iostream.h>
# include <conio.h>
# include <process.h>
class diagonal
{
    int * d;
    int n;
    public:
        diagonal (int size = 10)
        {
            n = size;
            if (size != 0)
                d = new int [n];
            else
                d = 0 ;
        }
        ~ diagonal()
        {
            delete [ ] d;
        }
        void store (int x, int i, int j);
        int retreive (int i, int j);
        void show ();
};
void diagonal :: store (int x, int i, int j)
{
    if ( i<1 || i>n || j<1 || j>n)
        { cout << " out of Bound.";
          exit(0);
        }
    else If (i>j && x != 0)
            cout << " must be zero ";
    else If (i == j)
            d[i] = x;
}
```

a[0] = 2
a[1] = 1
a[2] = 4
a[3] = 6.

```
int diagonal :: retrieve (int i, int j)
{
    If (i<1 || j<1 || i>n || j>n)
    {
        cout << "out of Bound.";
        exit(0);
    }
    else If (i==j)
        return d[i];
    else
        return 0;
}
void diagonal :: show()
{
    for (int i=1 ; i<=n ; i++)
    {
        for ( int j= 1; j<=n; j++)
        {
            If (i != j)
                cout << " 0 " << " \t ";
            else
                cout << d[i] << " \t ";
        }
        cout << " \n ";
    }
}
void main ()
{
    clrscr();
    int m1 = 0, m2 = 0, y = 0, i = 0, j = 0;
    cout << " \n Enter no. of rows & columns in diagonal matrix \n ";
    cin >> m1 >> m2;
    If ( m1 != m2)
    {
        cout << " Invalid size Entered ";
        exit(0);
    }
    diagonal d1 (m1);
```

```
cout << "\n Enter the elements of diagonal matrix \n";
for ( i=1, j=1 ; i <= m1, j <= m1 ; i++, j++)
  {
      cin >> y;
      d1. store ( y, i, j);
  }
cout << "\n The diagonal matrix is: \n";
d1. show ();
cout << "\n Enter the location for retrieval \n";
 cin >> i >> j;
 int res = d1. retrieve (i, j);
 cout << "\n Retrieved value is : " << res;
 cout << "\n";
}
```

output :-

```
Enter the no. of rows & columns in Diagonal matrix
  3
  3
Enter elements of diagonal matrix
 1
 2
 3
The diagonal matrix is :
  1    0    0
  0    2    0
  0    0    3
Enter the location for retrieval
2
2
Retrieved value is : 2
```

Lower Triangular Matrix (using 1-d Array)

```
# include <iostream.h>
# include <conio.h>
# include <process.h>
class lower
{
    int * a;
    int n;
    public:
        lower (int s=10)
        { n=s;
          if (s!=0)
              a= new int [n*(n+1)/2];
          else
              a=0;
        }
        ~lower()
          { delete [] a; }
        void store (int, int, int);
        int retrieve (int, int);
        void show();
};

void lower::store (int x, int i, int j)
{
    int k=0;
    If ( i<1 || i>n || j<1 || j>n)
    {
        cout << "out of Bound"; exit(0); }
    else If (i>=j)
     {
        k = (i * (i-1)/2 + j - 1);
        a[k] = x;
     }
    else If (x!=0)
            cout << "\n must be zero.";
}
```

3×3

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}_{3\times 3}$$

a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
a[5] = 6.

6 if n = 3

i=1, j=1
k = (1*(1-1)/2 + 1 - 1) = 0

i=1, j=2 ×

i=3, j=1
k = (3*(3-1)/2 + 1 - 1)
= 3 + 1 - 1 = 3

i=3, j=2
k = 3*(3-1)/2 + 2 - 1 = 4

```
int lower :: retrieve (int i, int j)
{
    if( i<1 || j<1 || i>n || j>n)
    {
        cout <<" Out of Bound.";
        exit(0);
    }
    If( i< j)
        return 0;
    else
    {
        int k = a[ i * (i-1)/2 + j-1];
        return (k);
    }
}
void lower :: show()
{  int i, j;
   for (i=1; i<=n ; i++)
   {
      for (j=1; j<=n; j++)
      {
          If (i< j)
              cout <<"0"<<"\t";
          else
          {
              int k = i * (i-1)/2 + j-1;
              cout << a [k] <<"\t";
          }
      }
      cout <<"\n";
   }
}
```

```
void main()
{
    int m1=0, m2=0, X=0, i=0, j=0; char ch;
    cout<<"\n Enter the no. of rows & col. in lower Triangular
        matrix \n";
    cin >> m1 >> m2;
    if ( m1 != m2)
    {   cout << " Invalid size";
        exit(0);
    }

    lower obl(m1);
    do
    {
     cout<<"\n enter the element, row, col. of LT Matrix \n:";
     cin>> X >> i >> j;
     obl. store (X, i, j);
    cout << "\n Enter ur choice"; cin >> ch;
    }while(ch==='y' || ch==='y');
cout<<"\n lower Triangular matrix is: \n";
    obl. show();
    cout<<"\n Enter the location for retrieval \n";
    cin>> i >>j;
    int res = obl. retrieve (i,j);
    cout<<"\n Retrieved value is: "<<res;
    cout << "\n";
}
```

Output :-

Enter the no. of Rows & Col. in LTM.

3
3

Enter the elements of LTM row, col

1 → ele
1 → Row No
1 → col No
want to cont y
2 → ele
2 → row No
1 → col No

Lower Triangular matrix is

1	0	0
2	3	0
4	5	6

Enter the location for retrieval

3
3

Retrieved value is : 6.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
a[5] = 6

Program to implement upper Triangular matrix (using 1-D Array)

```
#include <iostream.h>
#include <conio.h>
#include <process.h>
class upper
{
   int * a;
   int n;
  public:
        upper(int s=0)
        {
          n=s;
          if(s!=0)
                a=new int[s*(s+1)/2];
          else
                a=0;
        }
        ~upper()
          { delete[] a;}
        void store(int, int, int);
        int retrieve(int, int);
        void show();
};
void upper::store(int x, int i, int j)
{
    int k=0;
    if(i<1 || i>n || j<1 || j>n)
    { cout<<"OUT OF BOUND";
      exit(0);
    }
    else if(i<=j)
    {
        k=(n*(n+1)/2)-(j*(j-1)/2)+i-j;
        a[k]=x;
    }
    else if(x!=0)
          cout<<"\n must be zero";
}
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}_{3\times 3}$$

$n=3$

$i=1, j=2$

$k=\left(3*\frac{(3+1)}{2}\right)-\left(2*\frac{(2-1)}{2}\right)+1-2$

$= 6-1+1-2=4.$

ele	Row, col	Index
1	i=1, j=1	Pos = 6
2	1, 2	Pos = 4
3	1, 3	Pos = 1
4	2, 2	Pos = 5
5	2, 3	Pos = 2
6	3, 3	Pos = 3

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

a[1] = 3
a[2] = 5
a[3] = 6
a[4] = 2
a[5] = 4
a[6] = 1

```
int upper :: retrieve (int i, int j)
{
    if ( i<1 || j<1 || i>n || j>n)
    {
        cout << " OUT OF BOUND ";
        exit(0);
    }
    if (i>j)
        return 0;
    else
    {
        int k = a[(n * (n+1)/2) - (j * (j-1)/2) + i - j];
        return (k);
    }
}
void upper :: show()
{
    int i, j;
    for ( i=1 ; i<=n ; i++)
    {
        for (j=1 ; j<=n ; j++)
        {
            if (i>j)
                cout << " 0 " << " \t ";
            else
            {
                int k = (n * (n+1) /2) - (j * (j-1) /2) + i - j;
                cout << a[k] << " \t ";
            }
        }
        cout << " \n ";
    }
}
```

```
void main()
{
    int m1=0, m2=0, x=0, i=0, j=0;
    int ch;
    cout << "\n Enter the no. of rows & columns in the upper
                Triangular matrix \n";
    cin >> m1 >> m2;
    if (m1 != m2)
     { cout << "Invalid size entered";
       exit(0);
     }
    upper obj(m1);
    do{                                    rows, colums
       cout << "\n Enter the elements of upper Triangular matrix\n";
       cin >> x >> i >> j;
       obj.store(x, i, j);
       cout << "\n Enter ur choice";
     }while (ch == 'y' || ch == 'Y');
    cout << "\n The upper Triangular matrix is : \n";
     obj.show();
     cout << "\n Enter the location for retrieval \n";
     cin >> i >> j;
     int res = obj.retrieve(i,j);
     cout << "\n Retrieved value is : " << res;
     cout << "\n";
}
```

output :-

Enter the no. of rows & col in the upper Triangular matrix

4
4

Enter the elements of upper triangular matrix (row, columns)

1, 1, 1
2, 1, 2
3, 1, 3
4, 1, 4
5 :
6 :
7 :
8 :
9, 3, 4
10, 4, 4.

// a[1] = 4
a[2] = 7
// a[3] = 9
a[4] = 10
// a[5] = 3
a[6] = 6
a[7] = 8
a[8] = 2
a[9] = 5
a[10] = 1

The upper Triangular matrix is :

1	2	3	4
0	5	6	7
0	0	8	9
0	0	0	10

Enter the location for retrieval

4
4

Retrieved value is : 10

Implement symmetric matrix (using 1-D Array).

```
#include <iostream.h>
#include <conio.h>
#include <process.h>

class sym_matrix
{
    int * a;
    int size;
    public:
        sym_matrix(int n=0)
        {
            size = n;
            if (n! =0)
                a= new int [n * (n+1)/2];
            else
                a=0;
        }
    ~sym_matrix()
    { delete [] a; }
    void store (int, int, int);
    int retrieve (int, int);
    void display();
};
Void sym_matrix :: store (int x, int i; int j)
{
    int k = 0;
    if (i<1 || i> size || j<1 || j > size)
    { cout << " out of Bound ";
      exit(0);
    }
    If (i<j) return;
    k = i * (i-1)/2 + j-1 ; cout<<" array index Position : "<<k;
    a[k] = x
}
```

```
int sym_mat :: retrieve (int i, int j)
{
   If ( i<1  ||  j <1   ||  j >size || i > size)
    {   cout <<" OUT OF BOUND";
        exit(0);
    }
    If (i < j)
     {
        int temp = i;
        i = j;
        j = temp ;
     }
    return (a[ i * (i-1) /2 + j -1]);
}
void sym_mat :: display()
{
    for (int i=1 ; i<= size ; i++)
       {
          for ( int j = 1; j<= size ; j++)
             {
                if (i<j)
                     cout << a[j * (j-1)/2 + i -1] << "  ";
                else
                     cout << a[ i * (i-1)/2 + j - 1] << "  ";

             }
          cout <<" \n";
       }
}
```

```
Void main()
{
    clrscr();
    int m=0, n=0, y=0, i=0, j=0;
    cout<<" Enter the no. of rows & col. in the symmetric matrix";
    cin>>m>>n;
    if (m!=n)
    {  cout<<"Invalid size entered ";
       exit(0);
    }
    sym_matrix obj(m);
    cout<<"\n Enter the lower triangular ~~matrix~~ elements of
          sym. matrix \n";
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=i; j++)
          {
              cin>>y;
              obj.store(y,i,j);
          }
    }
    cout<<"\n Symmetric matrix is : \n";
    obj.display();
    cout<<"\n Enter the location for retrieval \n";
    cin>>i>>j;
    cout<<"\n Retrieved value is : "<<obj.retrieve(i,j)
                                    <<"\n";
    getch();
}
```

output :-

Enter No. of rows & columns in symmetric matrix

3
3

Enter elements of symmetric matrix

1

array index position : 0

2

array index Pos : 1

3

" " " : 2

4

" " " : 3

5

" " " : 4

6 " " " : 5

a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
a[5] = 6.

The symmetric matrix is

1	2	4
2	3	5
4	5	6

Enter the location for retrieval

2
3

Retrieved value is : 5

Date

(1) Analysis of Algorithms.

Bubble Sort

```
for (i = n-1 ; i >= 1 ; i--)
{
    for (j = 1 ; j <= i ; j++)
    {
        If (a[j] >= a[j+1])
        {
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
}
```

eg :-

Index	elements	I ST iter	II nd iter	III rd iter	IV th iter
1	50	40	40	30	25
2	40	50	30	25	30 ↱
3	60	30	25	40 ↱	40 ↱
4	30	25	50 ↱	50 ↱	50 ↱
5	25	60 ↱	60 ↱	60 ↱	60 ↱

steps :-
1) start with 1 ST element.
2) compare subsequent elements & push the largest one to the back.

Analysis :-
1). no. of times inner for loop get executed = i
2). The outer loop is executed for $(n-1)$ down to 1.
3). no. of operations done by outer for loop is :-

$$(n-1) + (n-2) + (n-3) + \dots\dots + 2 + 1.$$

$$= \frac{n(n+1)}{2} - n = \frac{n^2 - n}{2} = O(n^2).$$

Best case : $\Theta(n^2)$
worst case : $\Theta(n^2)$
Avg case : $\Theta(n^2)$.

Date _____

(2). Selection Sort

```
for (i=1 ; i <= n-1 ; i++)
{
    min = i;
    for (j = i+1 ; j <= n ; j++)
    {
        If (a[j] <= a[min])
            min = j;
    }
    If (i != min)
        swap a[i] && a[min]
}
```

Eg. n = 5

index	elts.	I st iter	II nd iter	III rd iter	IV th iter.
1	50	25	25	25	25
2	40	40	30	30	30
3	60	60	60	40	40
4	30	30	40	60	50
5	25	50	50	50	60

steps :-

1) consider 1st elt as the smallest element.
2) compare it with rest of the array.
3) find min elt from rest of the array & swap it with min. elt. chosen.

Analysis:-

1). No. of times inner for loop executes = $n - i$

2). At the worst case, 'If' statement will do one swap operation every time. Therefore, total no. of operations done in inner loop are $n - i + 1$.

3). outer loop is executed for $i = 1$ to $n-1$ times.

$$\therefore (n-1+1) + (n-2+1) + (n-3+1) + \dots + (n-(n-1)+1)$$
$$= n + (n-1) + (n-2) + \dots + 2.$$
$$= \frac{n(n+1)}{2} - 1 = \frac{n^2+n-2}{2} = O(n^2)$$

Best case $= \Theta(n^2)$
Worst case $= O(n^2)$
Avg. case $= \Theta(n^2)$.

Insertion sort lies b/w $\Omega(n)$ to $O(n^2)$

(3). Date

Insertion Sort.

```
for ( i = 2 ; i <= n ; i++)
  {
    v = a[i];
    j = i-1;
    while ( (j >= 1) && v <= a[j])
      {
        a[j+1] = a[j];
        j = j-1;
      }
    a[j+1] = v.
  }
```

Best case = $\Omega(n)$
Worst case = $O(n^2)$
Avg case = $O(n^2)$

n = 6

index / eg.	elts	I st iter	II nd iter	III rd iter	IV th iter	V th iter
1	5	2	2	2	1	1
2	2	5	4	4	2	2
3	4	4	5	5	4	3
4	6	6	6	6	5	4
5	1	1	1	1	6	5
6	3	3	3	3	3	6.

steps :-
1) start with 2nd elt & compare with above elt until u find a elt is less than elt chosen.
2). write the elt on the position u find & shift all elts right (down).

The array is partially sorted after each iteration.
In the worst case, eg:- 50 40 30 20. no. of comparisons

	comparison	
will be I st iter -	1	→ 2 elts
II nd iter -	2	→ 3 elts
III rd iter -	3	→ 4 els.
	n-1	→ n elts

OR. $(n-1) + (n-2) + \dots + 2 + 1$
$= \frac{n(n+1)}{2} - n = O(n^2)$.

Analysis:-
1). No. of times inner loop while get executed $i-1$
2). outer loop executed for $i = 2$ to n times.
3). ∴ Total no. of operations done are :-

$$(2-1) + (3-1) + (4-1) + \dots + (n-1-1) + (n-1)$$
$$= 1 + 2 + 3 + \dots + (n-1) = O(n^2).$$

Insertion sort :- running time $\neq \Omega(n^2)$ [Because it can be $\Omega(n)$ in the Best case, $O(n)$ in the Best case].

:- running time = $\Omega(n^2)$ or $O(n^2)$ (min. no. of comparisons will be n^2 in worst case).

Analysis.

	cost	times
`for ( i = 2; i <= n; i++)`	c_1	n
`{`		
`v = a[i];`	c_2	$n-1$
`j = i-1;`	c_4	$n-1$
`while (j >= 1 && v <= a[j])`	c_5	$\sum_{i=2}^{n} t_i$
`{`		
`a[j+1] = a[j];`	c_6	$\sum_{i=2}^{n} t_i - 1$
`j = j-1;`	c_7	$\sum_{i=2}^{n} t_i - 1$
`}`		
`a[j+1] = v;`	c_8	$n-1$
`}`		

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{i=2}^{n} t_i + c_6 \sum_{i=2}^{n} (t_i - 1) + c_7 \sum_{i=2}^{n} (t_i - 1) + c_8 (n-1).$$

Best case :- In Best case, while loop is not entered as $v \leq a[j]$ will be false, so the line get executed for $(n-1)$ times as other for loop lines :-

$$c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$
$$= (c_1 + c_2 + c_4 + c_5 + c_8) n - (c_2 + c_4 + c_5 + c_8).$$
$$= an + b.$$
$$= \Omega(n)$$

Worst Case :- In worst case, we must compare each element $a[i]$ with each element in the entire sorted subarray $a[1 \ldots i-1]$ & so $t_i = i$ for $i = 2, 3, \ldots n$. Also,

$$\sum_{i=2}^{n} i = \frac{n(n+1)}{2} - 1 \quad \text{and} \quad \sum_{i=2}^{n} (i-1) = \frac{n(n-1)}{2}$$

$$\therefore c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n+1)}{2}\right) + c_8(n-1).$$

$$\Rightarrow (\quad) n^2 + (\quad) n + (\quad) \Rightarrow an^2 + bn + c.$$

(4). Date

Linear Search

To find which element [if any] of the given array a[1..n] equals the target element.

```
for (i = 1; i <= n; i++)
    {
        If (target == a[i])
            {
                cout << "position:" << i;
                break;
            }
    }
```

Analysis:

Best case :- O(1)

Worst case :- O(n)

Average case :- O(n)

Date

Binary Search

To find which element (if any) of the given array a[1..n] equals the target element.

```
1. Begin
2. set left = 1, right = n
3. while (left <= right) do
   {  m = floor((left + right)/2)
      if (target == a[m] end with o/p as m.
      if (target < a[m]) then right = m-1
      if (target > a[m]) then left = m+1
   }
```

Analysis:-

Time to divide the array into 2 subarrays + 1 checking.

$T(n) = T(\frac{n}{2}) + 1$

$T(\frac{n}{2}) = T(\frac{n}{4}) + 1$

$T(\frac{n}{4}) = T(\frac{n}{8}) + 1$

⋮

$T(4) = T(2) + 1$

$T(2) = T(1) + 1.$

Backtrack.

$T(n) = T(1) + \log n$

$T(n) = 1 + \log n$

$= O(\log n).$

Eg. $T(8) = T(4) + 1$

$T(4) = T(2) + 1$

$T(2) = T(1) + 1$

Backtrack,

$T(8) = T(1) + 3.$

$= T(1) + \log_2 8 \rightarrow n$

$= 1 + \log 8.$

$T(n)$ → $T(\frac{n}{2})$ → $T(\frac{n}{4})$ → ⋮ → $T(1)$

$2^3 = 8$

taking log,

$\log_3 9 = 2.$

node size $= \frac{n}{2^i} = 1$

$n = 2^i$

Take log, $\log n = i \log 2 \therefore \boxed{i = \log n}$

Height of Tree = $\log_2 n$