```
    1.    Heap Max Algo  / Heap Min Algo  (both Algo same)
Int A(1:n)
int i,j,item;
j=n, i=[n/2], item=a(n)
        while(i > 0 && A[i] < item)  //for creating MAX HEAP
        //while(i>0 && A[i] >item)   //for creating MIN HEAP
                A[j] = A[i];
                j = i;
                i = i/2;
        repeat
        A[j] = item;
End INSERT
        For i=2 to n, do
                Call INSERT ( A, i)
        repeat
```

```
    2.     Heapify/Adjust Max/Min Algo
Int n,i;
        for(int i=n/2;i>=1;i--)
                ADJUST(i,n);
        repeat
end HEAPIFY
  ADJUST
        int I,j,n
        int j;
        j = 2 * i;
        int item = A[i];
        while(j<=n)
                if(j<n && A[j] > A[j+1])   //MIN HEAP
                        j = j+1;
                end if
                if(item < A[j])  //MIN HEAP
                        break;
                else
                        A[j/2] = A[j];
                        j = 2 * j;
                end if
        repeat
        A[j/2] = item;
End ADJUST-MAX
```

```
    3.   Max/Min Given array algo
MAXMIN(p,q,max,min)
        if(p==q)
                max=min=A[p];
        else
        {
                if(p==q-1),then
                        if(A[p]>A[q])
                                max=A[p];min=A[q];
                        else
                                max=A[q];min=A[p];
                          end if
                        else
                                int m=(p+q)/2;

        MAXMIN(p,m,fmax,fmin);

        MAXMIN(m+1,q,hmax,hmin);
                                if(fmax>hmax)
                                        max=fmax;
                                else
                                        max=hmax;
                                if(fmin<hmin)
                                        min=fmin;
                                else
                                        min=hmin;
                        end if
        end if
end MAXMIN
```

```
    4.   UNION & FIND Operation Algo
        int i,j,x;
                            int x=PAR[i]+PAR[j];
                            if(PAR[i]<PAR[j])
                            {
                                    PAR[j]=i;
                                    PAR[i]=x;
                            }
                            else
                            {
                                    PAR[i]=j;
                                    PAR[j]=x;
                               End if
                            End UNION
   FIND
        int i,j,k

        int j=i; //first a root of tree
        while(PAR[j]>0)
                j=PAR[j];
           repeat
        k=i
        while k = j do //collapse nodes from k to root
j
                temp=PAR[k];
                PAR[k]=j;
                K=temp;
           repeat
        return(j)
end FIND
```

```
    5.   BINary SeaRCH
        int low,high,mid,j,n;
        low=1;high=n;
        while(low<=high)
                mid=(low+high)/2;
                if(x < A[mid])
                        high=mid-1;
                else
                        if(x > A[mid])
                                low=mid+1;
                        else
                        {
                                j=mid;
                                return;
                        end case
                repeat

                j=o;

        end BINary SeaRCH
```

```
    6.   HEAPSORT Max/Min
        Int (A,n)
        HEAPIFY(A,n);
        for(int i = n; i>=2; i--)
                int temp = A[1];
                A[1] = A[i];
                A[i] = temp;
                ADJUST(1,i-1);
        repeat
end HEAP_SORT
```

```
    7.   MERGE_SORT(int A,low,high)
      if(low!=high)
        int mid=(low+high)/2;
        MERGE_SORT(low,mid);
        MERGE_SORT(mid+1,high);
        MERGE(low,mid,high);
      End if
End MERGE_SORT
```

```
    8.   Quick sort
        If p,q,then
         J=q+1;
                Partition (low, j)
                QUICK_SORT(arr,low,j-1);
                QUICK_SORT(arr,j+1,high);
        End if
End QUICK_SORT
```

```
    9.   STRASSEN_MATRIX

        int P = (A[1][1] + A[2][2]) * (B[1][1] +
B[2][2]) ;
        int Q = (A[2][1] + A[2][2]) * B[1][1];
        int R = A[1][1] * (B[1][2] - B[2][2]);
        int S = A[2][2] * (B[2][1] - B[1][1]);
        int T = (A[1][1] + A[1][2]) * B[2][2];
        int U = (A[2][1] - A[1][1]) * (B[1][1] +
B[1][2]);
        int V = (A[1][2] - A[2][2]) * (B[2][1] +
B[2][2]);

        C[1][1] = P + S - T + V;
        C[1][2] = R + T;
        C[2][1] = Q + S;
        C[2][2] = P + R - Q + U;
```

```
    10.  KNAPSACK
        Int I,n;
        for(int i=1; i<=n; i++)
           if(W[i] > Cu)
                        break;
           else
                        X[i] = 1;
                        Cu = Cu - W[i];
           End if
        repeat
        if(i <= n)
                X[i] = Cu/W[i];
        End if
End GREEDY_KNAPSACK
```

```
    11.  SSSP Algo
        int num, *S=new int[n+1];
        for(int i=1;i<=n;i++)
                S[i]=0;
                DIST[i]=COST[v][i];
        S[v]=1;
        DIST[v]=0;
        for(num=2;num<=n-1;num++)
        int u=0;
        float min=9999;
        for(int w=1;w<=n;w++)
                        if(S[w]==0 && DIST[w]<min)
                                min=DIST[w];
                                u=w;
                S[u]=1
                for(w=1;w<=n;w++)
                        if(S[w]==0)

        DIST[w]=MIN(DIST[w],DIST[u]+COST[u][w]);
                End if
        repeat
End SHORTEST_PATH
```

```
    12.  PRIMS
        int j,min=9999,k,l;
        for(int i=1;i<=n;i++)
                for(j=i;j<=n;j++)
                        if(COST[i][j]< min)
                                min=COST[i][j];
                                k=i;l=j;
        //======
        mincost=COST[k][l];
        T[1][1]=k;
        T[1][2]=l;
        for(i=1;i<=n;i++)
                if(COST[i][k] < COST[i][l])
                        NEAR[i]=k;
                else
                        NEAR[i]=l;
        End if
        NEAR[k]=0;NEAR[l]=0;
        repeat
        for(i=2;i<=n-1;i++)
                //---find j such that .....
                min=9999;
                for(k=1;k<=n;k++)
                        if(NEAR[k]!=0 &&
COST[k][NEAR[k]] < min)
                                j=k;

        min=COST[k][NEAR[k]];
                // add next edge in the Tree
                T[i][1]=j;
                T[i][2]=NEAR[j];
                mincost=mincost+COST[j][NEAR[j]];
                NEAR[j]=0;
                        for(k=1;k<=n;k++)
                        if(NEAR[k]!=0 && COST[k][j]
< COST[k][NEAR[k]])
                                NEAR[k]=j;
                End if
        repaet
        if(mincost>=9999)
                cout<<"\nNo spanning Tree";
        end if
end PRIM
```

```
    13.  KRUSKAL
        int u,v,j,k;
        //crate edge list
        CREATE_ED_LIST();
        SORT_ED_LIST();
        int i=0;
        mincost=0;
        int ptr=1;
        while(i<=n-1 && i<=noe)
                u= EDGE[ptr][1];
                v= EDGE[ptr][2];
                j=F(u);
                k=F(v);
                if(j!=k)
                        i=i+1;
                        T[i][1]=u;
                        T[i][2]=v;
                        mincost=mincost+COST[u][v];
                        U(j,k);
                End if
                ptr=ptr+1;
        repeat
        if(i < n-1)
                cout<<"\nNo spanning Tree";
        end if
end KRUSKAL
```

```
    14.  All PAIR SHORTEST PATH
        for(int i=1;i<=n;i++)
                for(int j=1;j<=n;j++)
                        A[i][j]=COST[i][j];
        repeat
        for(int k=1;k<=n;k++)
                for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)

        A[i][j]=MIN(A[i][j],A[i][k]+A[k][j]);
        repeat
        repeat
END ALL_PATHS
```

```
    15.  LCS
        for(int i=m;i>=1;i--) //rows
                for(int j=n;j>=1;j--)//column
                        if(B[i][j]=='\\')
                                top=top+1;
                                stk[top]=str1[i];
                                i=i-1;
                        else
                                if(B[i][j]=='|')
                                        i=i-1;
                end if
        for(i=top;i>=1;i--)
                cout<<stk[i]<<" ";
        repeat
end LCS_PRINT
```

```
    16.  DFS_S(int v)
        int STK[10],top=0;
        int u=v;
        VISITED[v]=1;
        do
                cout<<u<<" ";
                for(int w=1;w<=n;w++)
                      // adj      &&    not visited
                        if(A[u][w]==1 &&
VISITED[w]==0)
                                top=top+1;
                                STK[top]=w;
                                VISITED[w]=1;
                if(top==0)
                        break;
                else
                        u=STK[top];
                        top=top-1;
                end if
        repeat while(1);
End DFS
::BFS_Q(int v)
        int QUE[10],rear=0,front=0;
        int u=v;
        VISITED[v]=1;
        do
                cout<<u<<" ";
                for(int w=1;w<=n;w++)
                     // adj      &&    not visited
                        if(A[u][w]==1 &&
VISITED[w]==0)
                                rear=rear+1;
                                QUE[rear]=w;
                                if(front==0)
                                        front=1;
                                VISITED[w]=1;
                End if
                if(front==0) // Q empty
                        break;
                else
                        u=QUE[front];
                        if(front==rear)
                                front=0;
                        else
                                front=front+1;
                end if
        repeat while(1);
        End BFS
```

```
    17.  NQUEEN(int n)
        int k=1;
        X[k]=0;
        //--- false - tried all possible solutions ---
        while(k>0)
                X[k]=X[k]+1;
                //----queen is in board && not
attacking
                while( X[k]<=n && !PLACE(k) )
                        X[k]=X[k]+1;
                repeat
                if(X[k]<=n) // queen is placed
                        if(k==n)//last queen
                                count++;
                                GET_RESULT();
                        else // next queen
                                k=k+1;
                                X[k]=0;
                Ed if
                else // back track
                        k = k-1;
                repeat
        End N-QUEENS
```