1. Merge Sort

```
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

int main()
{
    int a[30],n,i;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    mergesort(a,0,n-1);
    for(i=0;i<n;i++)
    printf("%d ",a[i]);
    return 0;   }

void mergesort(int a[],int i,int j)
{
   int mid;
   if(i<j)  {
      mid=(i+j)/2;
      mergesort(a,i,mid);
      mergesort(a,mid+1,j);
      merge(a,i,mid,mid+1,j);   }
}

void merge(int a[],int i1,int j1,int i2,int j2)
{
   int temp[50];
   int i,j,k;
   i=i1;
   j=i2;
   k=0;

   while(i<=j1 && j<=j2)  {
            if(a[i]<a[j])
     temp[k++]=a[i++];
   else
      temp[k++]=a[j++];  }
           while(i<=j1)
    temp[k++]=a[i++];
    while(j<=j2)
    temp[k++]=a[j++];
      for(i=i1,j=0;i<=j2;i++,j++)
      a[i]=temp[j];
}
```

2. Matrix Chain Multiplication

```
#include<stdio.h>
#include<limits.h>
int MatrixChainMultiplication(int p[], int n)
{
  int m[n][n];
  int i, j, k, L, q;
  for (i=1; i<n; i++)
  m[i][i] = 0;
  for (L=2; L<n; L++)
  {
   for (i=1; i<n-L+1; i++)
        {
          j = i+L-1;
     m[i][j] = INT_MAX;
    for (k=i; k<=j-1; k++)
        {
          q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
          if (q < m[i][j])
          {
            m[i][j] = q; } } } }
      return m[1][n-1];
}

int main()
{
  int n,i;
  scanf("%d",&n);
  n++;
  int arr[n];
  for(i=0;i<n;i++)
  scanf("%d",&arr[i]);
  int size = sizeof(arr)/sizeof(arr[0]);
  printf("%d ", MatrixChainMultiplication(arr, size));
}
```

3. knapsack 0/1

```
#include <stdio.h>
int max(int a, int b)
{
   return (a > b) ? a : b;
}

int knapSack(int W, int wt[], int val[], int n)
{
   if (n == 0 || W == 0)
      return 0;

   if (wt[n - 1] > W)
      return knapSack(W, wt, val, n - 1);
else
return max(val[n-1]+ knapSack(W-wt[n-1],wt, val, n-1),knapSack(W, wt, val, n -1));
}

int main()
{
  int n,W;
   scanf("%d",&n);
   scanf("%d",&W);
   int val[n],wt[n];
   int i;
   for(i=0;i<n;i++)
      scanf("%d",&val[i]);
    for(i=0;i<n;i++)
      scanf("%d",&wt[i]);
   printf("%d", knapSack(W, wt, val, n));
}
```

4. OBST

```
#include<stdio.h>
#define MAX 10
int main()
{
   int w[MAX][MAX], c[MAX][MAX],  p[MAX], q[MAX];
   int  min, min1, n;
   int i,j,k,b;
   scanf("%d",&n);
   for(i=1; i <= n; i++)
   {
     scanf("%d",&p[i]);
           }
           for(i=0; i <= n; i++)
           {
            scanf("%d",&q[i]);
           }
            for(i=0; i <= n; i++)
            {
              for(j=0; j <= n; j++)
              {
                if(i == j)
                       {
                         w[i][j] = q[i];
                         c[i][j] = 0;
                        } } }

for(b=0; b < n; b++)
{
  for(i=0,j=b+1; j < n+1 && i < n+1; j++,i++)
  {
    if(i!=j && i < j)
           {
             w[i][j] = p[j] + q[j] + w[i][j-1];
             min = 30000;
             for(k = i+1; k <= j; k++)
             {
               min1 = c[i][k-1] + c[k][j] + w[i][j];
                      if(min > min1)
                      {
                        min = min1;
                      } }
              c[i][j] = min;
           } } }
printf("%d\n",c[0][n]);
}
```

5. Sum of Subsets

```
#include <stdio.h>
#include<stdlib.h>
int d;
void sum(int,int,int[]);
int main() {
   int n,w[100],i;
   scanf("%d%d",&n,&d);
   for(i=1;i<=n;i++)
   scanf("%d",&w[i]);
   sum(n,d,w);
}

void sum(int n,int d,int w[])
{
   int x[100],s,k,i;
   for(i=1;i<=n;i++)
   x[i]=0,s=0,k=1,x[k]=1;
   while(1)
         {
    if(k <= n && x[k]==1) {
    if(s+w[k] == d) {
          for(i=1;i<=n;i++) {
     if(x[i]==1)
     printf("%d ",w[i]);
          }
           printf("\n");
      x[k]=0;
         }
          else if(s+w[k] < d)
    s+=w[k];
    else {
      x[k]=0;
                  } }
                   else{
           k--;
    while(k>0 && x[k]==0)
    k--;
    if(k<=0){
       break;
                   }
     x[k]=0;
     s=s-w[k];  }
     k=k+1;
     x[k]=1;
 } }
```

6. N-Queens

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int a[30],count=0;

int place(int pos) {
  for(int i=1;i<pos;i++) {
    if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))
    return 0; }
     return 1;
}

void print_sol(int n) {
  int i,j; count++;
  for(i=1;i<=n;i++)  {
    for(j=1;j<=n;j++)  {
      if(a[i]==j)
      printf("%d ",a[i]);
      }  }
     printf("\n");
}

void queen(int n)  {
   int k=1;
   a[k]=0;
   while(k!=0)  {
     a[k]=a[k]+1;
     while((a[k]<=n)&&!place(k))
     a[k]++;
     if(a[k]<=n)  {
       if(k==n)
       print_sol(n);
       else  {
          k++;
          a[k]=0;
          }  }
          else k--;
    }  }

int main()  {
   int n;
   scanf("%d",&n);
   queen(n);
}
```

7. JOB SEQUENCING WITH DEADLINE

```
#include <stdio.h>
int main()
{
    int n,i,k;
    scanf("%d",&n);
    int p[100],d[100];
    for(i=0;i<n;i++)  {
    scanf("%d",&p[i]); }
     for(i=0;i<n;i++) {
     scanf("%d",&d[i]);
          }
      int ft=0;
      int md=0;
      for(i=0;i<n;i++) {
      if(d[i]>md) {
      md=d[i];
            }  }
     int tl[n];
     for(i=1;i<=md;i++)  {
     tl[i]=-1;
           }
      for(i=1;i<n;i++) {
        k=d[i-1];
        while(k>=1)  {
        if(tl[k]==-1)
       {
         tl[k]=i-1;
         ft=ft+1;
         break;
                  }
         k--;
          }
      if(md==ft)  {
       break;
            } }
    int s=0;
          for(i=1;i<=ft;i++)
          {s=s+p[tl[i]];
 }
   printf("%d",s);
}
```

8. Dijkstra: Shortest Reach 2

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{
    int t,n,e,i,j;
    scanf("%d",&t);
    for(i=0;i<t;i++)
    {
        scanf("%d%d",&n,&e);
        int s,k,mat[n+1][n+1],d[n+1],x,y,r,v[n+1];
        for(j=1;j<=n;j++)
        {
            d[j] = 999999;
            v[j] = 0;
            for(k=1;k<=n;k++)
            {
                mat[j][k] = 999999;
            }
        }
        for(j=0;j<e;j++)
        {
            scanf("%d%d%d",&x,&y,&r);
            if(mat[x][y] > r)
            {
                mat[x][y] = r;
                mat[y][x] = r;
            }
        }
        scanf("%d",&s);
        d[s] = 0;
        int p,min,mind=s;
        v[mind] = 1;
        for(j=1;j<=n;j++)
        {
            for(k=1;k<=n;k++)
            {
                if(mat[mind][k] != 999999)
                {
                    p = d[mind] + mat[mind][k];
                    if(d[k] > p)
```

```
                    {
                        d[k]= p;
                    }
                }
            }
            min = 9999999;
            for(k=1;k<=n;k++)
            {
                if(d[k] < min && v[k]==0)
                {
                    min = d[k];
                    mind = k;
                }
            }
            v[mind] = 1;
        }
        for(j=1;j<=n;j++)
        {
            if(d[j]!= 0)
            {
                if(d[j] == 999999)
                {
                    printf("-1 ");
                }
                else
                {
                    printf("%d ",d[j]);
                }

            }
        }
        printf("\n");
    }
  return 0;
}
```

9. Prim's (MST) : Special Subtree

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>

struct edge{
int x, y, l;
};

int main(){
int n,e,i,j;
  scanf("%d%d",&n,&e);
  struct edge a[e],t;

  for(i=0;i<e;i++)
  {
    scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].l);
    }
for(i=0;i<e-1;i++)
{
  for(j=i+1;j<e;j++)
  {
    if(a[i].l > a[j].l)
    {
      t = a[i];
      a[i] = a[j];
      a[j] = t;
      }
    }
  }

int sum=0,s,vn[3002]={0},vf,vi[n],vil,xz,yz;
scanf("%d",&s);
vi[0] = s;
vil = 1;
while(vil<n)
{
  for(i=0;i<e;i++)
  {
    xz = 0;
    yz = 0;
    if(vn[a[i].x] == 0 || vn[a[i].y] == 0)
```

```
    {
      for(j=0;j<vil;j++)
      {
        if(a[i].x == vi[j])
        {
          xz++;
          }

        if(a[i].y == vi[j])
        {
          yz++;
          }
        }

      if((xz==0 && yz!=0) || (yz==0 && xz!=0))
      {
        sum = sum + a[i].l;
        if(xz==0)
        {
          vi[vil] = a[i].x;
          vil++;
          vn[a[i].x]++;
          }
        else
        {
          vi[vil] = a[i].y;
          vil++;
          vn[a[i].y]++;
          }
        i = -1;  } } }
  }

  printf("%d\n",sum);
  return 0;
}
```

Sample Input 0

```
5 6
1 2 3
1 3 4
4 2 6
5 2 2
2 3 5
3 5 7
1
```

Sample Output 0

```
15
```

9. PRIM'S ALGORITHM

```
#include<stdio.h>
#include<stdlib.h>
#define infinity 9999
#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;
int prims();

int main()
{
  int i,j,total_cost;
  scanf("%d",&n);

  for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  scanf("%d",&G[i][j]);
  total_cost=prims();
  printf("%d",total_cost);
  return 0;
}

int prims()
{
    int cost[MAX][MAX];
    int u,v,min_distance,distance[MAX],from[MAX];
    int visited[MAX],no_of_edges,i,min_cost,j;
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
          {
           if(G[i][j]==0)
      cost[i][j]=infinity;
    else
      cost[i][j]=G[i][j];
      spanning[i][j]=0;
         }
    distance[0]=0;
    visited[0]=1;

          for(i=1;i<n;i++)
          {
           distance[i]=cost[0][i];
```

```
 from[i]=0;
 visited[i]=0;
    }
     min_cost=0;
  no_of_edges=n-1;
  while(no_of_edges>0)
      {
       min_distance=infinity;
   for(i=1;i<n;i++)
   if(visited[i]==0&&distance[i]<min_distance)
            {
             v=i;
             min_distance=distance[i];
            }
             u=from[v];
    spanning[u][v]=distance[v];
    spanning[v][u]=distance[v];
    no_of_edges--;
    visited[v]=1;

   for(i=1;i<n;i++)
   if(visited[i]==0&&cost[i][v]<distance[i])
            {
    distance[i]=cost[i][v];
    from[i]=v;
            }
   min_cost=min_cost+cost[u][v];
            }
 return(min_cost);
}
```

Sample Input 0

```
6
0 4 4 0 0 0
4 0 2 0 0 0
4 2 0 3 2 4
0 0 3 0 0 3
0 0 2 0 0 3
0 0 4 3 3 0
```

Sample Output 0

```
14
```

10. Kruskal (MST): Really Special Subtree

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

typedef struct  {
   int u, v, m, w;
} Arco;

int menorArco( Arco * arcos, int M ) {
   int menor, i;
   menor = -1;
   for (i = 0; i < M; i++ ) {
      if ( arcos[i].m == 0 ) {
         if ( menor == -1 )
            menor = i;
         else
            if ( arcos[i].w < arcos[menor].w  )
               menor = i;
            else
               if ( arcos[i].w == arcos[menor].w )
                  if ( arcos[i].u + arcos[i].w + arcos[i].v <
                       arcos[menor].u + arcos[menor].w + arcos[menor].v )
                     menor = i;
      }
   }
   return menor;
}

void unirComponentes( int * vertices, int N, int c1, int c2 ) {
   int i;
   for ( i = 0; i < N; i++ )
      if ( vertices[i] == c2 )
         vertices[i] = c1;
}

int main() {
   int N, M, S, vs, vl, d;
   int **graph;
   Arco *arcos;
   int *vertices;
```

```
 int i, j, k;
scanf("%d%d",&N, &M);
graph = (int **)malloc(N*sizeof(int *));
for ( i = 0; i < N; i++ )
    graph[i] = (int *)malloc( N*sizeof(int) );

for ( i = 0; i < N; i++ )
    for ( j = 0; j < N; j++ )
       graph[i][j] = -1;

for ( i = 0; i < M; i++ ) {
    scanf("%d%d%d", &vs, &vl, &d );
    vs--;vl--;
    if ( vl > vs ) {
        int t = vs; vs = vl; vl = t;
    }
    if ( graph[vs][vl] == -1 || graph[vs][vl] > d )
        graph[vs][vl] = d;
}

M = 0;
for ( i = 0; i < N; i++ )
    for ( j = 0; j < N; j++ )
       if ( graph[i][j] >= 0 )
          M++;

arcos = (Arco *)malloc( M*sizeof(Arco) );
for ( k = i = 0; i < N; i++ )
    for ( j = 0; j < N; j++ )
        if ( graph[i][j] >= 0 ) {
            arcos[k].u = i;
            arcos[k].v = j;
            arcos[k].w = graph[i][j];
            arcos[k].m = 0;
            k++;
        }

for ( i = 0; i < N; i++ )
    free( graph[i] );
free( graph );

vertices = (int *)malloc( N*sizeof(int) );
for ( i = 0; i < N; i++ )
```

```
        vertices[i] = i;

    scanf("%d", &S);
    S--;

    int menor, u, v;
    while ( (menor = menorArco(arcos,M)) >= 0 ) {
        u = arcos[menor].u;
        v = arcos[menor].v;
        if ( vertices[u] != vertices[v] ) {
            unirComponentes( vertices, N, vertices[u], vertices[v]);
            arcos[menor].m = 1;
        }
        else
            arcos[menor].m = -1;
    }

    long R;
    R = 0;
    for ( i = 0; i < M; i++ )
        if ( arcos[i].m == 1 )
            R = R + arcos[i].w;

    printf("%ld",R);


    return 0;
}
```

10. KRUSHAL'S MINIMUM SPANNING TREE

```
#include <stdio.h>

#define MAX 30

typedef struct edge {
  int u, v, w;
} edge;

typedef struct edge_list {
  edge data[MAX];
  int n;
} edge_list;

edge_list elist;

int Graph[MAX][MAX], n;
edge_list spanlist;

void kruskalAlgo();
int find(int belongs[], int vertexno);
void applyUnion(int belongs[], int c1, int c2);
void sort();
void print();

// Applying Krushkal Algo
void kruskalAlgo() {
  int belongs[MAX], i, j, cno1, cno2;
  elist.n = 0;

  for (i = 1; i < n; i++)
    for (j = 0; j < i; j++) {
      if (Graph[i][j] != 0) {
        elist.data[elist.n].u = i;
        elist.data[elist.n].v = j;
        elist.data[elist.n].w = Graph[i][j];
        elist.n++;
      }
    }

  sort();

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

```
    belongs[i] = i;

  spanlist.n = 0;

  for (i = 0; i < elist.n; i++) {
    cno1 = find(belongs, elist.data[i].u);
    cno2 = find(belongs, elist.data[i].v);

    if (cno1 != cno2) {
      spanlist.data[spanlist.n] = elist.data[i];
      spanlist.n = spanlist.n + 1;
      applyUnion(belongs, cno1, cno2);
    }
  }
}

int find(int belongs[], int vertexno) {
  return (belongs[vertexno]);
}

void applyUnion(int belongs[], int c1, int c2) {
  int i;

  for (i = 0; i < n; i++)
    if (belongs[i] == c2)
      belongs[i] = c1;
}

// Sorting algo
void sort() {
  int i, j;
  edge temp;

  for (i = 1; i < elist.n; i++)
    for (j = 0; j < elist.n - 1; j++)
      if (elist.data[j].w > elist.data[j + 1].w) {
        temp = elist.data[j];
        elist.data[j] = elist.data[j + 1];
        elist.data[j + 1] = temp;
      }
}

// Printing the result
```

```
void print() {
  int i, cost = 0;

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

    cost = cost + spanlist.data[i].w;
  }

  printf("%d",cost);
}

int main() {


  n = 6;

  Graph[0][0] = 0;
  Graph[0][1] = 4;
  Graph[0][2] = 4;
  Graph[0][3] = 0;
  Graph[0][4] = 0;
  Graph[0][5] = 0;


  Graph[1][0] = 4;
  Graph[1][1] = 0;
  Graph[1][2] = 2;
  Graph[1][3] = 0;
  Graph[1][4] = 0;
  Graph[1][5] = 0;


  Graph[2][0] = 4;
  Graph[2][1] = 2;
  Graph[2][2] = 0;
  Graph[2][3] = 3;
  Graph[2][4] = 2;
  Graph[2][5] = 4;


  Graph[3][0] = 0;
  Graph[3][1] = 0;
  Graph[3][2] = 3;
```

```
  Graph[3][3] = 0;
  Graph[3][4] = 0;
  Graph[3][5] = 3;


  Graph[4][0] = 0;
  Graph[4][1] = 0;
  Graph[4][2] = 2;
  Graph[4][3] = 0;
  Graph[4][4] = 0;
  Graph[4][5] = 3;


  Graph[5][0] = 0;
  Graph[5][1] = 0;
  Graph[5][2] = 4;
  Graph[5][3] = 3;
  Graph[5][4] = 3;
  Graph[5][5] = 0;


  kruskalAlgo();
  print();
}
```

Sample Input 0

```
6
0 4 4 0 0 0
4 0 2 0 0 0
4 2 0 3 2 4
0 0 3 0 0 3
0 0 2 0 0 3
0 0 4 3 3 0
```

Sample Output 0

```
14
```