1: Quicksort

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
#define maxsize 30000
#define NTIMES 5000
int partition(int a[],int low,int high)
{
    int p,i,j,temp;
    p=a[low];
    i=low+1;
    j=high;
    while(1)
    {
        while(a[i]<=p&&i<high)
            i++;
        while(a[j]>p&&j>=low)
            j--;
        if(i<j)
        {
            temp=a[j];
            a[j]=a[i];
            a[i]=temp;
        }
        else
        {
            temp=a[j];
            a[j]=a[low];
            a[low]=temp;
            return j;
        }
    }

}
void quicksort(int a[],int low,int high)
{
    int s;
    if(low<high)
    {
        s=partition(a,low,high);
        quicksort(a,low,s-1);
        quicksort(a,s+1,high);
    }
}
void main()
{
    int a[maxsize],n,i,k;
    clock_t start,end;
    double runtime=0;
    clrscr();
    printf("Enter the size of array: ");
    scanf("%d",&n);
    for(k=0;k<NTIMES;k++)
    {
        srand(1);
        for(i=0;i<n;i++)
            a[i]=rand();
        start=clock();
        quicksort(a,0,n-1);
        end=clock();
        runtime+=(end-start)/CLK_TCK;
    }
    runtime/=NTIMES;
    printf("Sorted elements are\n");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\nAvg runtime = %lfs",runtime);
    getch();
}
```

2: Mergesort

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
#define maxsize 30000
#define NTIMES 5000
void merge(int a[],int low,int mid,int high)
{
    int i,j,k;
    int b[maxsize];
    i=low;
    j=mid+1;
    k=low;
    while(i<=mid&&j<=high)
    {
        if(a[i]<=a[j])
            b[k++]=a[i++];
        else    b[k++]=a[j++];
    }
    while(i<=mid)
        b[k++]=a[i++];
    while(j<=high)
        b[k++]=a[j++];
    for(i=low;i<=high;i++)
        a[i]=b[i];
}
void mergesort(int a[],int low,int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        mergesort(a,low,mid);
        mergesort(a,mid+1,high);
        merge(a,low,mid,high);
    }
}
void main()
{
    int a[maxsize],n,i,k;
    clock_t start,end;
    double runtime=0;
    clrscr();
    printf("Enter the size of array: ");
    scanf("%d",&n);
    for(k=0;k<NTIMES;k++)
    {
        srand(1);
        for(i=0;i<n;i++)
            a[i]=rand();
        start=clock();
        mergesort(a,0,n-1);
        end=clock();
        runtime+=((end-start)/CLK_TCK);
    }
    runtime/=NTIMES;
    printf("Sorted elements are\n");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\nAvg runtime = %lfs",runtime);
    getch();
}
```

3A: BFS

```c
#include<stdio.h>
#include<conio.h>
int i,j,n,f=0,r=0,q[10],a[10][10],vis[10];
void bfs(int u)
{
	int v;
	vis[u]=1;
	q[r]=u;
	while(f<=r)
	{
		u=q[f++];
		for(v=1;v<=n;v++)
			if(a[u][v]&&!vis[v])
			{
				vis[v]=1;
				q[++r]=v;
			}
	}
}
void main()
{
	int src;
	clrscr();
	printf("\nEnter no.of vertices: ");
	scanf("%d",&n);
	printf("Enter adjacency matrix\n");
	for(i=1;i<=n;i++)
		for(j=1;j<=n;j++)
			scanf("%d",&a[i][j]);
	printf("Enter the source vertex: ");
	scanf("%d",&src);
	for(i=0;i<=n;i++)
		vis[i]=0;
	bfs(src);
	printf("From vertex %d, the vertices\n",src);
	for(i=1;i<=n;i++)
		if(vis[i])
			printf("%d is reachable\n",i);
	getch();
}
```

3B: DFS

```
#include<stdio.h>
#include<conio.h>
int n,a[10][10],vis[10];
void dfs(int u)
{
    int v;
    vis[u]=1;
    for(v=1;v<=n;v++)
        if(a[u][v]==1&&vis[v]==0)
            dfs(v);

}
void main()
{
    int i,j;
    clrscr();
    printf("\nEnter no.of vertices: ");
    scanf("%d",&n);
    printf("Enter adjacency matrix\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    for(i=1;i<=n;i++)
        vis[i]=0;
    for(i=1;i<=n;i++)
    {
        dfs(i);
        for(j=1;j<=n;j++)
        {
            if(vis[j]!=1)
            {
                printf("%d is not reachable from %d",j,i);
                printf("\nSo graph is not connected");
                getch();
                return;
            }
        }
        for(j=1;j<=n;j++)
            vis[j]=0;
    }
    printf("\nGraph is connected");
    getch();
}
```

4: Floyd's Algorithm

```c
#include<stdio.h>
#include<conio.h>
int min(int a,int b)
{
	return (a<b)?a:b;
}
void floyds(int a[10][10],int n)
{
	int i,j,k;
	for(k=0;k<n;k++)
		for(i=0;i<n;i++)
			for(j=0;j<n;j++)
				a[i][j]=min(a[i][j],a[i][k]+a[k][j]);
}
void main()
{
	int cost[10][10],i,j,n;
	clrscr();
	printf("Enter size of cost matrix:");
	scanf("%d",&n);
	printf("\nEnter cost matrix\n");
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			scanf("%d",&cost[i][j]);
	floyds(cost,n);
	printf("\nAll pair shortest path is \n");
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
			printf("%d ",cost[i][j]);
		printf("\n");
	}
	getch();
}
```

5: Warshall's Algorithm

```
#include<stdio.h>
#include<conio.h>
void war(int a[10][10],int n)
{
    int i,j,k;
    for(k=0;k<n;k++)
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                a[i][j]=a[i][j]||a[i][k]&&a[k][j];
}
void main()
{
    int a[10][10],i,j,n;
    clrscr();
    printf("Enter size of adjacency matrix: ");
    scanf("%d",&n);
    printf("\nEnter adjacency matrix\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    war(a,n);
    printf("\nMatrix of transitive closure is \n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
    getch();
}
```

6: Horspool String Matching

```c
#include<stdio.h>
#include<conio.h>
#include<string.h>
int table[128];
void shifttable(char p[],int m)
{
    int i;
    for(i=0;i<128;i++)
        table[i]=m;
    for(i=0;i<m-1;i++)
        table[p[i]]=m-1-i;
}
int horspoolmatch(char p[],int m,char t[],int n)
{
    int i,k;
    shifttable(p,m);
    for(i=m-1;i<n;i+=table[t[i]])
    {
        k=0;
        while(k<m&&p[m-1-k]==t[i-k])
            k++;

        if(k==m)
            return i-m+1;
    }
    return -1;
}
void main()
{
    char t[100],p[50];
    int m,n,pos;
    clrscr();
    printf("\nEnter the text made up of alphabets of all possible 128 characters:\n");
    gets(t);
    printf("\nEnter the pattern: ");
    gets(p);
    n=strlen(t);
    m=strlen(p);
    pos=horspoolmatch(p,m,t,n);
    if(pos==-1)
        printf("Pattern not found");
    else
        printf("Pattern found at index %d",pos);
    getch();
}
```

7: Knapsack using Dynamic Programming

```c
#include<stdio.h>
#include<conio.h>
int weight[10],val[10],v[10][10];
int max(int a,int b)
{
    return (a>b)?a:b;
}
int knapsack(int n,int w)
{
    int i,j;
    for(i=0;i<=n;i++)
    {
        for(j=0;j<=w;j++)
        {
            if(i==0||j==0)
                v[i][j]=0;
            else if(j-weight[i]>=0)
                v[i][j]=max(v[i-1][j],val[i]+v[i-1][j-weight[i]]);
            else
                v[i][j]=v[i-1][j];
        }
    }
    return v[n][w];
}
void optimalsubset(int n,int w)
{
    int i,j;
    for(i=n,j=w;i>=1&&j>0;i--)
    {
        if(v[i][j]!=v[i-1][j])
        {
            printf("Item %d\n",i);
            j-=weight[i];
        }
    }
}

void main()
{
    int n,w,value,i,j;
    clrscr();
    printf("Enter the no. of items: ");
    scanf("%d",&n);
    printf("Enter the weights of each item\n");
    for(i=1;i<=n;i++)
        scanf("%d",&weight[i]);
    printf("Enter the values of each item\n");
    for(i=1;i<=n;i++)
        scanf("%d",&val[i]);
    printf("Enter the knapsack capacity: ");
    scanf("%d",&w);
    value=knapsack(n,w);
    printf("Solution of knapsack problem\n");
    for(i=0;i<=n;i++)
    {
        for(j=0;j<=w;j++)
            printf("%d\t",v[i][j]);
        printf("\n");
    }
    printf("\nThe maximum value is %d",value);
    printf("\nThe items of an optimal subset are\n");
    optimalsubset(n,w);
    getch();
}
```

8: Kruskal's Algorithm

```c
#include<stdio.h>
#include<conio.h>
int parent[10];
void main()
{
    int mincost=0,cost[10][10];
    int n,i,j,ne,min,a,b,u,v;
    clrscr();
    printf("Enter no. of vertices of graph: ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        parent[i]=0;
    printf("Enter cost matrix\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    ne=1;
    printf("The edges of minimal spanning tree are\n");
    while(ne<n)
    {
        for(i=1,min=999;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]<min)
                {
                    min=cost[i][j];
                    a=u=i;
                    b=v=j;
                }
        while(parent[u])
            u=parent[u];
        while(parent[v])
            v=parent[v];
        if(v!=u)
        {
            printf("\n%d.Edge (%d,%d) : %d",ne++,a,b,min);
            mincost+=min;
            parent[b]=a;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\nMincost= %d",mincost);
    getch();
}
```

9: Prim's Algorithm

```c
#include<stdio.h>
#include<conio.h>
int visited[10];
void main()
{
    int mincost=0,cost[10][10];
    int n,i,j,ne,min,a,b;
    clrscr();
    printf("Enter no. of vertices of graph: ");
    scanf("%d",&n);
    printf("Enter cost matrix\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    for(i=2;i<=n;i++)
        visited[i]=0;
    visited[1]=1;
    ne=1;
    printf("The edges of minimal spanning tree are\n");
    while(ne<n)
    {
        for(i=1,min=999;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]<min&&visited[i])
                {
                    min=cost[i][j];
                    a=i;
                    b=j;
                }

        if(!visited[a]||!visited[b])
        {
            printf("%d.Edge (%d,%d) : %d\n",ne++,a,b,min);
            mincost+=min;
            visited[b]=1;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\nMincost= %d",mincost);
    getch();
}
```

10: Dijkstra's Algorithm

```c
#include<stdio.h>
#include<conio.h>
void dijkstra(int n,int v,int cost[10][10],int dist[10])
{
    int i,k,count,visited[10],min;
    for(i=1;i<=n;i++)
    {
        visited[i]=0;
        dist[i]=cost[v][i];
    }
    visited[v]=1;
    dist[v]=0;
    count=2;
    while(count<n)
    {
        min=999;
        for(i=1;i<=n;i++)
        {
            if(dist[i]<min&&!visited[i])
            {
                min=dist[i];
                k=i;
            }
        }
        visited[k]=1;
        count++;
        for(i=1;i<=n;i++)
            if(!visited[i])
                if(dist[i]>dist[k]+cost[k][i])
                    dist[i]=dist[k]+cost[k][i];


    }
}
void main()
{
    int n,v,i,j,cost[10][10],dist[10];
    clrscr();
    printf("Enter no. of vertices of graph: ");
    scanf("%d",&n);
    printf("Enter cost matrix\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    printf("\nEnter source vertex: ");
    scanf("%d",&v);
    dijkstra(n,v,cost,dist);
    printf("Shortest paths are\n");
    for(i=1;i<=n;i++)
        if(i!=v)
            printf("%d->%d\tcost=%d\n",v,i,dist[i]);
    getch();
}
```

11: Sum of Subset

```
#include<stdio.h>
#include<conio.h>
int x[10],a[10],sum,n,i,j,flag=0,count=0;
void sos(int s,int k,int r)
{
    x[k]=1;
    if(s+a[k]==sum)
    {
        printf("\nSubset %d: ",++count);
        flag=1;
        for(i=0;i<=k;i++)
            if(x[i])
                printf("%3d",a[i]);
    }
    else if(s+a[k]+a[k+1]<=sum)
        sos(s+a[k],k+1,r-a[k]);

    if((s+r-a[k]>=sum)&&(s+a[k+1]<=sum))
    {
        x[k]=0;
        sos(s,k+1,r-a[k]);
    }
}
void main()
{
    int r=0,s,temp;
    clrscr();
    printf("Enter no. of elements: ");
    scanf("%d",&n);
    printf("Enter the elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n;i++)
        for(j=0;j<n-i-1;j++)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
    printf("After sorting, elements are: \n");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\nEnter the required sum: ");
    scanf("%d",&sum);
    for(i=0;i<n;i++)
    {
        x[i]=0;
        r+=a[i];
    }
    if(r<sum)
    {
        printf("\nNo subset possible");
        flag=1;
    }
    else
        sos(0,0,r);
    if(flag==0)
        printf("\nNo subset possible");
    getch();
}
```

12: N-Queens

```c
#include<stdio.h>
#include<conio.h>
int x[10];
int place(int k,int i)
{
    int j;
    for(j=1;j<k;j++)
        if(x[j]==i||(abs(x[j]-i)==abs(j-k)))
            return 0;
    return 1;
}
void nqueens(int k,int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        if(place(k,i))
        {
            x[k]=i;
            if(k==n)
            {
                printf("\nSolution is\n");
                for(j=1;j<=n;j++)
                    printf("%2d",x[j]);
            }
            else
                nqueens(k+1,n);
        }
    }
}
void main()
{
    int n;
    clrscr();
    printf("Enter no. of queens: ");
    scanf("%d",&n);
    if(n==0||n==2||n==3)
        printf("No solutions\n");
    else
        nqueens(1,n);
    getch();
}
```