Lab 1 kruskal's

```
#include<stdio.h>
#include<conio.h>
int ne=1,min_cost=0;
void main()
{
int n,i,j,min,a,u,b,v,cost[20][20],parent[20];

printf("Enter the no. of vertices:");
scanf("%d",&n);
printf("\nEnter the cost matrix:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&cost[i][j]);
for(i=1;i<=n;i++)
parent[i]=0;
printf("\nThe edges of spanning tree are\n");
while(ne<n)
{
min=999;
for(i=1;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(u!=v)
{
printf("Edge %d\t(%d->%d)=%d\n",ne++,a,b,min);
min_cost=min_cost+min;
parent[v]=u;
}
cost[a][b]=cost[a][b]=999;
}
printf("\nMinimum cost=%d\n",min_cost);

}
```

Lab 2 Prims

```
#include<stdio.h>
int ne=1,min_cost=0;
void main()
{
int n,i,j,min,cost[20][20],a,u,b,v,source,visited[20];
printf("Enter the no. of nodes:");
scanf("%d",&n);
printf("Enter the cost matrix:\n");
```

```
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
}
}

for(i=1;i<=n;i++)
visited[i]=0;
printf("Enter the root node:");
scanf("%d",&source);
visited[source]=1;

printf("\nMinimum cost spanning tree is\n");
while(ne<n)
{
min=999;
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
if(cost[i][j]<min)
if(visited[i]==0)
continue;

else
{
min=cost[i][j];
 a=u=i;
b=v=j;

}
}
}

if(visited[u]==0||visited[v]==0)
{
printf("\nEdge %d\t(%d->%d)=%d\n",ne++,a,b,min); min_cost=min_cost+min;
visited[b]=1;
}
cost[a][b]=cost[b][a]=999;
}
printf("\nMinimum cost=%d\n",min_cost); getch();
}
```

Lab 3a floyds

```
#include<stdio.h>
#include<conio.h>
#define INF 999
int min(int a,int b)
{
```

```
    return(a<b)?a:b;
}
void floyd(int p[][10],int n)
{
    int i,j,k;
    for(k=1; k<=n; k++)
        for(i=1; i<=n; i++)
            for(j=1; j<=n; j++)
                p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
}
void main()
{
    int a[10][10],n,i,j;
    printf("\nEnter the n value:");
    scanf("%d",&n);
    printf("\nEnter the graph data:\n");
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            scanf("%d",&a[i][j]);
    floyd(a,n);
    printf("\nShortest path matrix\n");
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }

}
```

Lab 3b Warshal's

```
#include<stdio.h>
void warshall(int[10][10],int);

void main()
{
int a[10][10],i,j,n;
printf("Enter the number of nodes:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("The adjacency matirx is:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
warshall(a,n);
}

void warshall(int p[10][10],int n)
{
```

```
int i,j,k;
for(k=1;k<=n;k++)
{
for(j=1;j<=n;j++)
{
for(i=1;i<=n;i++)
{
if((p[i][j]==0) && (p[i][k]==1) && (p[k][j]==1))
{
p[i][j]=1;
}
}
}
}
printf("\nThe path matrix is:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t",p[i][j]);
}
printf("\n");
}
}
```

Lab 4 Dijkstra

```
#include<stdio.h>
void dij(int,int [20][20],int [20],int [20],int);

void main()
{
int i,j,n,visited[20],source,cost[20][20],d[20];
printf("Enter no. of vertices: ");
scanf("%d",&n);
printf("Enter the cost adjacency matrix\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
}
}
printf("\nEnter the source node: ");
scanf("%d",&source);
dij(source,cost,visited,d,n);
 for(i=1;i<=n;i++)
if(i!=source)
printf("\nShortest path from %d to %d is %d",source,i,d[i]);

}

void dij(int source,int cost[20][20],int visited[20],int d[20],int n)
{
int i,j,min,u,w;
for(i=1;i<=n;i++)
{
```

```
visited[i]=0;
d[i]=cost[source][i];
}
visited[source]=1;
d[source]=0;
for(j=2;j<=n;j++)
{
min=999;
for(i=1;i<=n;i++)
{
if(!visited[i])
{
if(d[i]<min)
{

min=d[i];
 u=i;
}
}
}     //for i
visited[u]=1;

for(w=1;w<=n;w++)
{
if(cost[u][w]!=999 && visited[w]==0)
{
 if(d[w]>cost[u][w]+d[u])
d[w]=cost[u][w]+d[u];
}
} //for w
} // for j
}
```

Lab 5 Topological

```
#include<stdio.h>
void findindegree(int [10][10],int[10],int);
void topological(int,int [10][10]);
void main()
{
int a[10][10],i,j,n;
printf("Enter the number of nodes:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\nThe adjacency matirx is:\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
topological(n,a);
}
```

```
void findindegree(int a[10][10],int indegree[10],int n)
{
int i,j,sum;
for(j=1;j<=n;j++)
{
sum=0;
for(i=1;i<=n;i++)
{
sum=sum+a[i][j];
}
indegree[j]=sum;
}
}

void topological(int n,int a[10][10])
{
int k,top,t[100],i,stack[20],u,v,indegree[20];
k=1;
top=-1;
 findindegree(a,indegree,n);
for(i=1;i<=n;i++)
{
if(indegree[i]==0)
{
stack[++top]=i;
}
}
while(top!=-1)
{
u=stack[top--];
t[k++]=u;
for(v=1;v<=n;v++)
{
if(a[u][v]==1)
{
indegree[v]--;
if(indegree[v]==0)
{
stack[++top]=v;
}
}
}
}
printf("\nTopological sequence is\n");
for(i=1;i<=n;i++)
printf("%d\t",t[i]);
}
```

Lab 6 Knapsack

```
#include<stdio.h>
#define MAX 50

int p[MAX],w[MAX],n;
int knapsack(int,int);
int max(int,int);
```

```
void main()
{
int m,i,optsoln;
printf("Enter no. of objects: ");
scanf("%d",&n);
printf("\nEnter the weights:\n");
for(i=1;i<=n;i++)
scanf("%d",&w[i]);
printf("\nEnter the profits:\n");
for(i=1;i<=n;i++)
scanf("%d",&p[i]);
printf("\nEnter the knapsack capacity:");
scanf("%d",&m);
optsoln=knapsack(1,m);

printf("\nThe optimal soluntion is:%d",optsoln);
}

int knapsack(int i,int m)
{
if(i==n)
return (w[n]>m) ? 0 : p[n];
if(w[i]>m)
return knapsack(i+1,m);
return max(knapsack(i+1,m),knapsack(i+1,m-w[i])+p[i]);
}

int max(int a,int b)
{

if(a>b)
return a;
else
 return b;
}
```

Lab 7 Greedy

```
#include<stdio.h>

int main()

{

     float weight[50],profit[50],ratio[50],Totalvalue,temp,capacity,amount;

     int n,i,j;

     printf("Enter the number of items :");

     scanf("%d",&n);

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

    {

        printf("Enter Weight and Profit for item[%d] :\n",i);

        scanf("%f %f", &weight[i], &profit[i]);

    }
```

```
    printf("Enter the capacity of knapsack :\n");
    scanf("%f",&capacity);
     for(i=0;i<n;i++)
         ratio[i]=profit[i]/weight[i];
    for (i = 0; i < n; i++)
      for (j = i + 1; j < n; j++)
         if (ratio[i] < ratio[j])
        {
            temp = ratio[j];
            ratio[j] = ratio[i];
            ratio[i] = temp;
            temp = weight[j];
            weight[j] = weight[i];
            weight[i] = temp;
            temp = profit[j];
            profit[j] = profit[i];
            profit[i] = temp;
         }
printf("Knapsack problems using Greedy Algorithm:\n");
    for (i = 0; i < n; i++)
    {
     if (weight[i] > capacity)
         break;
      else
     {
         Totalvalue = Totalvalue + profit[i];
         capacity = capacity - weight[i];
      }
    }
      if (i < n)
      Totalvalue = Totalvalue + (ratio[i]*capacity);
    printf("\nThe maximum value is :%f\n",Totalvalue);
    return 0;
```

```
}
```

Lab 8 subset

```
#include<stdio.h>
void subset(int,int,int);
int x[10],w[10],d,count=0;

void main()
{
int i,n,sum=0;
printf("Enter the no. of elements: ");
scanf("%d",&n);
printf("\nEnter the elements in ascending order:\n");
for(i=0;i<n;i++)
scanf("%d",&w[i]);
printf("\nEnter the sum: ");
scanf("%d",&d);
for(i=0;i<n;i++)
sum=sum+w[i];
if(sum<d)
{
printf("No solution\n");
return;
}
subset(0,0,sum);
if(count==0)
{
printf("No solution\n");
return;
}
getch();
}

void subset(int cs,int k,int r)
{
int i; x[k]=1;
if(cs+w[k]==d)
{
printf("\n\nSubset %d\n",++count);
for(i=0;i<=k;i++)
if(x[i]==1)
printf("%d\t",w[i]);
}
else
 if(cs+w[k]+w[k+1]<=d)
subset(cs+w[k],k+1,r-w[k]);

if(cs+r-w[k]>=d && cs+w[k]<=d)
{
x[k]=0;
subset(cs,k+1,r-w[k]);
}
}
```

Lab 9 selsort

```
#include<stdio.h>
#include<time.h>
#define max 50000

void main()
{
int a[max],i,n;
clock_t start,end;
double time_taken;

printf("Enter the value of n:");
scanf("%d",&n);

for ( i = 0; i < n; i++)
{
    a[i] = rand() % 1000;
}

printf("\nThe array elements before\n");
for ( i = 0; i < n; i++)
{
    printf("%d\t", a[i]);
}

start = clock();
selectionsort(a, n);
end = clock();

time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

printf("\nElements of the array after sorting are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);

printf("\nTime taken:%f", time_taken);
}

void selectionsort(int a[],int n)
{
int temp,min,i,j;
for(i=0;i<=n-2;i++)
{
      min=i;
      for(j=i+1;j<=n-1;j++)
      {
            if(a[j]<a[min])
            {
                  min=j;
            }
      }
      temp=a[i];
      a[i]=a[min];
      a[min]=temp;
}
}
```

Lab 10 quicksort

```
#include<stdio.h>
#include<time.h>
#define max 50000

void  quicksort(int a[],int low, int high);
int partition(int a[],int low, int high);
void  interchange(int a[], int i, int j);

void main()
{
int a[max],i,n;
clock_t start,end;
double time_taken;

printf("Enter the value of n:");
scanf("%d",&n);

for ( i = 0; i < n; i++)
{
    a[i] = rand() % 1000;
}

printf("\nThe array elements before\n");
for ( i = 0; i < n; i++)
{
    printf("%d\t", a[i]);
}

start = clock();
quicksort(a,0,n-1);
end = clock();

time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

printf("\nElements of the array after sorting are:\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);

printf("\nTime taken:%f", time_taken);
}

void  quicksort(int a[],int low, int high)
{
int  j;
if(low < high)
{
j=partition(a,low,high);  // partition array into parts
quicksort(a,low,j-1);  // sort left part of array
quicksort(a,j+1,high);  // sort right part of array
}
}

int partition(int a[],int low, int high)
{
int  pivot, i, j,temp;
pivot=a[low];  // first element is pivot element
```

```
i=low;
j=high;

while(i <= j)
{
while(a[i] <= pivot)
i++;
while(a[j] > pivot)
j--;
if(i < j)
{
temp = a[i];
a[i] = a[j];
a[j] = temp;  //swap the contents
}
}
a[low] = a[j];
a[j] = pivot;
return j;
}
```

11. Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<time.h>
#define max 50000

void mergesort(int a[], int low, int high);
void merge(int a[], int low, int mid, int high);

void main()
{
        int a[max],i,n;
        clock_t start,end;
        double time_taken;

        printf("Enter the value of n:");
        scanf("%d",&n);

        for ( i = 0; i < n; i++)
        {
                a[i] = rand() % 1000;
        }

        printf("\nThe array elements before\n");
        for ( i = 0; i < n; i++)
        {
                printf("%d\t", a[i]);
        }
```

```
	start = clock();
	mergesort(a,0,n-1);
	end = clock();

	time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

	printf("\nElements of the array after sorting are:\n");
	for(i=0;i<n;i++)
	printf("%d\t",a[i]);

	printf("\nTime taken:%f", time_taken);
}


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 merge(int a[], int low, int mid, int high)
{
	int i, j, h, k, b[100000];
	h=low; i=low; j=mid+1;

	while((h<=mid) && (j<=high))
	{
		if(a[h] < a[j])
			b[i++] = a[h++];
		else
			b[i++] = a[j++];
	}

	for(k=j; k<=high; k++)
	{
		b[i] = a[k];
		i = i+1;
	}

	for(k=h; k<=mid; k++)
	{
		b[i] = a[k];
		i = i+1;
	}
```

```
        for(k=low; k<= high; k++)
                a[k] = b[k];
}
```

Lab 12 queen

```
#include<stdio.h>
void nqueens(int); int place(int[],int);
void printsolution(int,int[]);

void main()
{
int n;
printf("Enter the no.of queens: ");
scanf("%d",&n);
nqueens(n);
}

void nqueens(int n)
{
int x[10],count=0,k=1;
x[k]=0;
while(k!=0)
{
x[k]=x[k]+1;

while(x[k]<=n&&(!place(x,k)))
x[k]=x[k]+1;
if(x[k]<=n)
{
if(k==n)
{
 count++;
printf("\nSolution %d\n",count);
printsolution(n,x);
}
else
{
 k++;
x[k]=0;
 }
}
else
{

k--; //backtracking
}
 }

return;
}
int place(int x[],int k)
{
int i;
for(i=1;i<k;i++)
if(x[i]==x[k]||(abs(x[i]-x[k]))==abs(i-k)) return 0;
return 1;
```

```
}

void printsolution(int n,int x[])
{
int i,j;
char c[10][10];

for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
c[i][j]='X';

}

for(i=1;i<=n;i++)
c[i][x[i]]='Q';

for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("%c\t",c[i][j]);
}
printf("\n");
}
}
```