# Kruskal

```
/* Roll no. : 2002
Batch : E-10
*/

#include <iostream>
#include<stdlib.h>
#include "graph.h"
using namespace std;

int main()
{
	graph g;

	do
		{
			int ch;
			cout<<"\tMENU\n";

			cout<<"\t\t1.Enter Adjecency matrix\n\t\t2.Display Adjecency
matrix\n\t\t3.Kruskal\n\t\t"
				"4.Display minimum spanning tree\n\t\tExit\n\n";
			cout<<"Enter Choice : ";
			cin>>ch;
			cout<<endl;

			switch(ch)
			{
			case 1:		g.getdata();
					break;

			case 2:		g.print_adj_matrix();
					break;

			case 3:		g.kruskal();
					break;

			case 4:	g.print_spanning_tree();
					break;

			case 7:		exit(0);
					break;

			}
		}
		while(1);

	return 0;

}
```

**graph.h**

```
/*
 * graph.h
 *
 *  Created on: 12-Mar-2018
 *      Author: e2002
 */

#ifndef GRAPH_H_
#define GRAPH_H_

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

class graph
{
	int n,span_number,edge_number;
	int G[100][100];
	edge e[100];
	edge data[100];

public:
	graph(){n=0;span_number=0;edge_number=0;}
	void getdata();
	void print_adj_matrix();

	void sort();
	void print_spanning_tree();
	void kruskal();


};

#endif /* GRAPH_H_ */
```

**graph.cpp**

```
/*
 * graph.cpp
 *
 *  Created on: 12-Mar-2018
 *      Author: e2002
 */

#include "graph.h"
#include<iostream>
#include<string>
using namespace std;

void graph:: getdata()
{
	cout<<"Enter number of nodes : ";
	cin>>n;
	cout<<endl<<endl;

	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			cin>>G[i][j];
		}
	}
}

void graph:: print_adj_matrix()
{
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
			cout<<G[i][j]<<" ";

		cout<<endl;
	}
}

void graph:: sort()
{
	for(int i=0;i<edge_number-1;i++)
	{
		for(int j=0;j<edge_number-i-1;j++)
		{
			if(e[j].w > e[j+1].w)
			{
				edge temp=e[j];
				e[j] = e[j+1];
				e[j+1]=temp;
			}
```

```
                }
        }
}

void graph:: print_spanning_tree()
{
        int cost=0;

        for(int i=0;i<span_number;i++)
        {
                cout<<data[i].u<<" "<<data[i].v<<" "<<data[i].w<<endl;
                cost+=data[i].w;
        }

        cout<<"Cost of Spanning tree is : "<<cost<<endl;
}

int find(int arr[],int n)
{
        return arr[n];
}

void uni(int arr[],int v1,int v2,int n)
{
        for(int i=0;i<n;i++)
        {
                if(arr[i]==v2)
                        arr[i]=v1;
        }
}

void graph:: kruskal()
{
        int check[100];
        edge span;

        for(int i=1;i<n;i++)
        {
                for(int j=0;j<i;j++)
                {
                        if(G[i][j] != 0)
                        {
                                e[edge_number].u=i;
                                e[edge_number].v=j;
                                e[edge_number].w=G[i][j];

                                edge_number++;
                        }
                }
        }                                                       // edges inserted

        sort();                                         //sorted
```

```
	for(int i=0;i<edge_number;i++)		//check initialized
		check[i]=i;


	int v1,v2;
	for(int i=0;i<edge_number;i++)
	{
		v1=find(check,e[i].u);		//check cycle
		v2=find(check,e[i].v);

		if(v1 != v2)
		{
			data[span_number]=e[i];
			span_number++;

			uni(check,v1,v2,n);
		}
	}


}
```