11/9/24 LAB-10 Krushals Algorithm

```
#include<stdio.h>
#define INF 9999
#define MAX 10

int find(int parent[], int i){
    while(parent[i]!=0){
        i=parent[i];
    }
    return i;
}

void unionSets(int parent[], int u, int v){
    parent[v]=u;
}

void kruskal(int c[MAX][MAX], int n){
    int parent[MAX], ne=0;
    int mincost=0;

for(int i=0; i<n; i++){
        parent[i]=0;
    }
pf("Edges in the min spanning tree
    are:\n");
while(ne<n-1){
    int min=INF;
    int a=-1, b=-1, u=-1, v=-1;
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        if(c[i][j]<min){
            min=c[i][j];
```

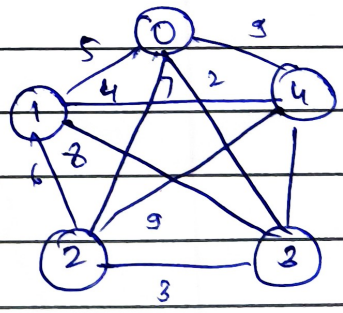
```
        a=u=i;
        b=v=j;
      }
    }
  }
  u= find(parent, u);
  v= find(parent, v);
  if(u!=v){
      pf("%d - %d\n", a, b, min);
      unionsets(parent, u, v);
      ne++;
      mincost += min;
    }
    c[a][b] = c[b][a] = INF;
  }
  pf("Minimum cost = %d\n", mincost);
}
```

Output:

Enter the number of nodes: 5.

Enter the cost matrix:

0	5	7	2	9
5	0	6	8	4
7	6	0	3	9
2	8	3	0	4
9	4	9	4	0



Edges in the min spanning tree are:

0 - 3 ; 2

2 - 3 ; 3

1 - 4 ; 4

3 - 4 ; 4

Min cost: 13.

11/7/24.

LAB-10 Djikstra's Algorithm

```
#include<stdio.h>
#define INF 9999
# define MAX 10
void dijkstra(int c[MAX][MAX], int n, int src){
   int dist[MAX], vis[MAX], count, min, u;
     for(int j=0; j<n; j++){
           dist[j] = c[src][j];
              vis[j]=0;
     }
     dist[src]=0;
     vis[src]=1;
     count=1;
     while(count!=n){
          min=INF;
          for(int j=0; j<n; j++) {
              if(dist[j]<min && vis[j]!=1){
                  min=dist[j];
                  u=j;
              }
          }
      vis[u]=1;
      count++;
     for(int j=0; j<n; j++){
          if(dist[j]<min && vis[j]!=1){
               min=dist[j];
                  u=j;
          }
     }
        vis[u]=1;
        count++;
        for(int j=0; j<n; j++) {
              if(min+ c[u][j] <
                  dist[j] && vis[j]!=1){
```

```
dist[j] =
  }
 }
}

printf(" S
for(int j
   p

 }
 }
```

Output:
Enter
Enter
0 1
3 0 C
1
7
6
Enter
1 -
1 -
1

```
        dist[j] = min + c[u][j];
    }
}

printf("Shortest ~~dist~~ distances are: \n");
for(int j=0; j<n; j++) {
    pf("From %d to %d : %d \n", src, j,
        dist[j]);
}
}
```

Output:

Enter no. of nodes = 6

Enter the cost matrix:

0	1	3	6	2	8
3	0	9	1	5	2
1	2	0	5	6	2
7	4	3	0	8	3
6	3	6	8	1	0

Enter the source node: 1

1 - 0 : 3

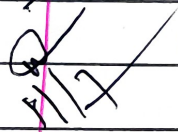
1 - 1 : 0

1 - 2 : 4

1 - 3 : 1

1 - 4 : 3

1 - 5 : 2



Graph: