Description:

C implementation of Kruskal's algorithm to build a Minimum Spanning Tree from a cost adjacency matrix. Uses a simple union–find to avoid cycles, selects the cheapest edges, prints MST edges and total minimum cost, and reports CPU time.

Source Code:

```
#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <time.h>


int i, j, k, a, b, u, v, n, ne = 1;

int min, mincost = 0, cost[9][9], parent[9];


int find(int);

int uni(int, int);


void main()

{

   clock_t start,end;

   double cpu_time_used;

   start=clock();

printf("Kruskal's algorithm in C\n");

printf("========================\n");


printf("Enter the no. of vertices:\n");
```

```
scanf("%d", &n);

printf("\nEnter the cost adjacency 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("The edges of Minimum Cost 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;
```

```
}
}
}

u = find(u);
v = find(v);

if (uni(u, v))
{
printf("%d edge (%d,%d) =%d\n", ne++, a, b, min);
mincost += min;

}

cost[a][b] = cost[b][a] = 999;
}

printf("\nMinimum cost = %d\n", mincost);
end=clock();
cpu_time_used=((double)(end-start))/CLOCKS_PER_SEC;
printf("Execution time:%f",cpu_time_used);
getch();
}

int find(int i)
{
```

```
while (parent[i])

i = parent[i];

return i;

}


int uni(int i, int j)

{

if (i != j)

{

parent[j] = i;

return 1;

}


return 0;

}
```

Output:

```
Kruskal's algorithm in C
========================
Enter the no. of vertices (max 8):
2

Enter the cost adjacency matrix (0 for no edge):
4
6
8
10
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) = 6

Minimum cost = 6
Execution time: 0.000031 seconds
```