

```

//code for the minimum spanning tree using kruskal algorithm through
openmp
#include <stdio.h>
#include <omp.h>

#define max 81
#define nodes 9

typedef struct edge
{
    int u, v, wt;
}edge;

typedef struct edgelist
{
    edge elist[max]; //for storing the all the egdes with their weigths
    int n; //for total number of edges in the graph
}edgelist;

//nodelist will give all the edges in the graph..
//spanlist will give the minimum spannnng edges for MST..
edgelist nodelist, spanlist;

//Graph taken form-->http://www.ggu.ac.in/download/Class-Notel3/ds%20lecture%20notes%20graph12.11.13.pdf<--Fig.9.21-Page 7
int graph[nodes][nodes]=
{{0,2,0,7,4,0,0,10,3},{2,0,2,0,7,4,0,0,6},{0,2,0,5,0,2,1,0,4},{7,0,5,0,0,0,6,5,1},{4,7,0,0,0,0,0,0,0},{0,0,2,0,0,0,0,0,0},{0,0,1,6,0,0,0,0,0},{10,0,0,5,0,0,0,0,0},{3,6,4,1,0,0,0,0,0}};

void edgelist_sort()
{
    //sorting the nodelsit
    int i, j;
    edge temp;

    for(i=0; i<nodelist.n; i++)
        for(j=i+1; j<nodelist.n; j++)
            if(nodelist.elist[i].wt > nodelist.elist[j].wt)
            {
                //performing swaping
                temp = nodelist.elist[i];
                nodelist.elist[i] = nodelist.elist[j];
                nodelist.elist[j] = temp;
            }
    //end of if
} //end of sort_edgelsit()

void join_sets(int edge_belongs[], int s1, int s2)
{
    int i=0;
    //will tell us if the copy of an egde exits in the nodelist..
    for(i=0; i<nodes; i++)
        if(edge_belongs[i]==s2)
            edge_belongs[i]=1;
} //end of union1()

int cost_calculation(int t)
{
    int i=0, cost=0;

```

```

        printf("\n*****\nThe Minimum Spanning Tree for thread %d\n", t);
        for(i=0; i<spanlist.n; i++)
        {
            printf("%d\t%d\t%d\n", spanlist.elist[i].u,
spanlist.elist[i].v, spanlist.elist[i].wt);
            cost+=spanlist.elist[i].wt;
        }
        //printing the total cost
        printf("\nThe Total Cost for the Graph is %d\n", cost);

        return cost;
    } //end of cost_calculation()

void kruskal(int thread)
{
    int i, j;
    int edge_belongs[nodes];
    int set1, set2;
    #pragma omp parallel sections num_threads(thread) default(none)
    shared(graph, nodelist, spanlist, edge_belongs, set1, set2) private(i,j)
    {
        //getting all the edges from the graph given
        #pragma omp section
        {
            nodelist.n=0;

            for(i=0; i<nodes; i++)
            {
                for(j=0; j<nodes; j++)
                {
                    if(graph[i][j]!=0)
                    {
                        nodelist.elist[nodelist.n].u = i;
                        nodelist.elist[nodelist.n].v = j;
                        nodelist.elist[nodelist.n].wt =
graph[i][j];

                        nodelist.n++;
                    } //end of if
                } //end of for-j
            } //end of for-i

            //sorting the egdes according to increasing order of
weight
            edgelist_sort();
        } //end of parallel section-1

        #pragma omp section
        {
            for(i=0; i<nodes; i++)
                edge_belongs[i] = i; //verties/node starts from 0
        } //end of parallel section-2

        #pragma omp section
        {
            spanlist.n = 0;

            for(i=0; i<nodelist.n; i++)
            {
                set1 = edge_belongs[nodelist.elist[i].u];

```

```

        set2 = edge_belongs[nodelist.elist[i].v];

        if(set1 != set2)//avoiding the same node i.e 1,2
and 2,1(taking only one)
        {
            spanlist.elist[spanlist.n] =
nodelist.elist[i];
            spanlist.n++;
            join_sets(edge_belongs, set1, set2);
        }//end of if
    }//end of for-i
} //end of parallel section-3
} //end of the parallel code
} //end of kruskal()

int main()
{
    int i=0, j=0, total_cost=0;
    float total_time=0.0;
    //taking 20 threads.
    int threads[] = {1, 2, 4, 6, 8, 10, 12, 14, 18, 22, 26, 30, 34, 38,
42, 46, 50, 54, 58, 62};

    //printing the adjacent matrix.
    printf("\n*****KRUSKAL'S ALGOTITHM FOR MINIMUM SPANNING TREE
USING OPENMP*****");
    printf("\nThe number of verties taken for the graph is 9.\n(Vertie
starts from 0 to
8)\n*****\n");

    printf("\nThe Adjacent Matrix --\n");
    for(i=0; i<nodes; i++)
    {
        for(j=0; j<nodes; j++)
            printf("%d\t", graph[i][j]);
        printf("\n");
    }

    //running threads
    printf("\n*****\n");
    for(i=0; i<20; i++)
    {
        float start=omp_get_wtime();

        kruskal(threads[i]);

        float end=omp_get_wtime();
        float time=end-start;

        total_time+=time;
        //printing the egdes of MST along with their weigths
        total_cost+=cost_calculation(threads[i]);
        printf("%d Thread Takes\t%f Time\n", threads[i],time);
    }

    printf("\n*****\n
*****\n\nThe Average Cost of MST for 20 threads is %d\n\nThe Average
Time taken by 20 threads is %f", total_cost/20, total_time/20);
    printf("\n*****XXXXXX*****");

```

```

    return 0;
} //end of main()

```

```

*****
*****

```

//the output of the above program...

*****KRUSKAL'S ALGOTITHM FOR MINIMUM SPANNING TREE USING OPENMP*****

The number of verties taken for the graph is 9.

(Vertie starts from 0 to 8)

```

*****

```

The Adjacent Matrix --

0	2	0	7	4	0	0	10	3
2	0	2	0	7	4	0	0	6
0	2	0	5	0	2	1	0	4
7	0	5	0	0	0	6	5	1
4	7	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	1	6	0	0	0	0	0
10	0	0	5	0	0	0	0	0
3	6	4	1	0	0	0	0	0

```

*****

```

```

*****

```

The Minimum Spanning Tree for thread 1

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	2	2
2	1	2
8	0	3
0	8	3
0	4	4
2	8	4
1	5	4
8	2	4
4	0	4
7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134

1 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 2

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
0	8	3
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5
2	5	2
0	1	2
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	0	2
4	0	4
4	1	7
5	2	2
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
2	1	2

The Total Cost for the Graph is 139

2 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 4

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
1	0	2
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5
2	5	2
0	1	2
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5

0	8	3
4	0	4
4	1	7
0	8	3
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
2	1	2

The Total Cost for the Graph is 140
 4 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 6

0	1	2
1	0	2
0	4	4
0	7	10
0	3	7
0	7	10
0	7	10
1	4	7
0	3	7
1	4	7
1	8	6
2	3	5
2	3	5
0	4	4
2	8	4
3	0	7
1	8	6
3	6	6
3	7	5
3	6	6
3	2	5
4	1	7
3	7	5
2	3	5
6	3	6
7	0	10
3	6	6
1	8	6
8	1	6
6	3	6
3	6	6

The Total Cost for the Graph is 188
 6 Thread Takes 0.000488 Time

The Minimum Spanning Tree for thread 8

The Total Cost for the Graph is 0
 8 Thread Takes 0.002441 Time

The Minimum Spanning Tree for thread 10

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
1	0	2
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5
2	5	2
2	6	1
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	0	2
4	0	4
4	1	7
5	2	2
6	2	1
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
2	1	2

The Total Cost for the Graph is 136

10 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 12

0	1	2
0	3	7
0	4	4
0	7	10
0	3	7
0	3	7
0	3	7
1	4	7
0	3	7
1	8	6
1	8	6
2	3	5
2	3	5
0	4	4
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
2	3	5
3	2	5
4	1	7

3	6	6
2	3	5
6	3	6
7	0	10
3	6	6
1	8	6
8	1	6
6	3	6
3	6	6

The Total Cost for the Graph is 185
 12 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 14

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
1	8	7
0	7	10
1	4	7
0	3	7
1	4	7
1	8	6
2	3	5
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	5	4
4	0	4
4	1	7
0	4	4
2	8	4
6	3	6
7	0	10
7	3	5
1	5	4
8	1	6
8	2	4
4	0	4

The Total Cost for the Graph is 148
 14 Thread Takes 0.000488 Time

The Minimum Spanning Tree for thread 18

0	1	2
0	3	7
0	4	4
0	7	10
0	4	4
0	4	4
0	8	3
1	4	7

1	5	4
1	8	6
0	4	4
2	3	5
1	5	4
0	8	3
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
0	8	3
4	0	4
4	1	7
0	8	3
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
0	8	3

The Total Cost for the Graph is 150
 18 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 22

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	2	2
2	1	2
8	0	3
0	8	3
0	4	4
2	8	4
1	5	4
8	2	4
4	0	4
7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134
22 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 26

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	2	2
2	1	2
8	0	3
0	8	3
0	4	4
2	8	4
1	5	4
8	2	4
4	0	4
7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134
26 Thread Takes 0.000488 Time

The Minimum Spanning Tree for thread 30

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

7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134
 30 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 34

2	6	1
3	8	1
1	2	2
0	3	7
0	7	10
0	7	10
0	3	7
1	4	7
0	3	7
1	8	6
1	8	6
2	3	5
1	5	4
0	4	4
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	5	4
4	0	4
4	1	7
0	4	4
2	8	4
6	3	6
7	0	10
7	3	5
1	5	4
8	1	6
8	2	4
4	0	4

The Total Cost for the Graph is 166
 34 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 38

2	6	1
3	8	1
6	2	1

8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	2	2
2	1	2
8	0	3
0	8	3
0	4	4
2	8	4
1	5	4
8	2	4
4	0	4
7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134
 38 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 42

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
1	0	2
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5
2	5	2
0	1	2
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	0	2
4	0	4
4	1	7
5	2	2
1	2	2
6	3	6
7	0	10

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

The Total Cost for the Graph is 138
 42 Thread Takes 0.000488 Time

The Minimum Spanning Tree for thread 46

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
1	0	2
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5
2	5	2
0	1	2
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	0	2
4	0	4
4	1	7
5	2	2
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
2	1	2

The Total Cost for the Graph is 138
 46 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 50

0	1	2
0	3	7
0	4	4
0	7	10
0	8	3
1	0	2
1	2	2
1	4	7
1	5	4
1	8	6
2	1	2
2	3	5

2	5	2
0	1	2
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	0	2
4	0	4
4	1	7
5	2	2
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
2	1	2

The Total Cost for the Graph is 138
 50 Thread Takes 0.000000 Time

 The Minimum Spanning Tree for thread 54

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	2	2
2	1	2
8	0	3
0	8	3
0	4	4
2	8	4
1	5	4
8	2	4
4	0	4
7	3	5
3	2	5
3	7	5
2	3	5
1	8	6
8	1	6
6	3	6
3	6	6
3	0	7
1	4	7
0	3	7
4	1	7
7	0	10
0	7	10

The Total Cost for the Graph is 134
 54 Thread Takes 0.000488 Time

The Minimum Spanning Tree for thread 58

2	6	1
3	8	1
6	2	1
8	3	1
2	5	2
0	1	2
1	0	2
5	2	2
1	8	6
0	7	10
0	3	7
1	4	7
0	3	7
1	4	7
0	3	7
3	0	7
1	8	6
3	6	6
1	8	6
3	6	6
1	8	6
4	1	7
3	6	6
1	8	6
6	3	6
7	0	10
3	6	6
1	8	6
8	1	6
6	3	6
2	3	5

The Total Cost for the Graph is 164

58 Thread Takes 0.000000 Time

The Minimum Spanning Tree for thread 62

2	6	1
3	8	1
6	2	1
0	8	3
0	7	10
0	3	7
0	4	4
1	4	7
1	5	4
1	8	6
0	4	4
2	3	5
1	5	4
0	8	3
2	8	4
3	0	7
3	2	5
3	6	6
3	7	5
1	5	4
4	0	4

4	1	7
0	8	3
1	2	2
6	3	6
7	0	10
7	3	5
8	0	3
8	1	6
8	2	4
0	8	3

The Total Cost for the Graph is 144
62 Thread Takes 0.000488 Time

The Average Cost of MST for 20 threads is 138
The Average Time taken by 20 threads is 0.000269
*****XXXXXX*****