Assignment 2

Parallel Computing(CS 309)

RUCHIR MEHTA
Roll No: 180001044

Date: 12-04-2021

Problem Statement

Write a multi-core program for the “all-pairs shortest paths” problem. The input is a weighted graph with no negative cycles and the expected output are the lengths of the shortest paths between all pairs of vertices (where the length of a path is a sum of weights along the edges that the path consists of). Write a sequential program (no OpenMP directives at all) as well. Compare the running time of the sequential program with the running time of the multi-core program and compute the speedup achieved.

1. for different number of cores and different number of threads per core, and
2. for a different number of vertices and a different number of edges.

Code files for serial and parallel implementation: [Attached]

Main Code Snippet:

```
for (int k = 0; k < V; k++)
{
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            if (graph[i][k] + graph[k][j] < graph[i][j])
                graph[i][j] = graph[i][k] + graph[k][j];
        }
    }
}
```

Serial Execution

```
for (int k = 0; k < V; k++)
{
    #pragma omp parallel for
    for (int i = 0; i < V; i++)
    {
        #pragma omp parallel for
        for (int j = 0; j < V; j++)
        {
            graph[i][j] = min(graph[i][k] + graph[k][j],graph[i][j]);
        }
    }
}
```

Parallel Execution

The approach used is Multithreading using OpenMP

Testing sample input

```
10 10
1 5 1
1 2 2
1 3 3
2 4 4
5 3 5
5 1 6
3 4 7
4 5 8
5 6 9
6 7 10
```

Graph input

```
0  INF  INF  INF  INF  INF  INF  INF  INF  INF
INF  0  2  3  6  1  10  20  INF  INF
INF  18  0  17  4  12  21  31  INF  INF
INF  21  23  0  7  15  24  34  INF  INF
INF  14  16  13  0  8  17  27  INF  INF
INF  6  8  5  12  0  9  19  INF  INF
INF  INF  INF  INF  INF  INF  0  10  INF  INF
INF  INF  INF  INF  INF  INF  INF  0  INF  INF
INF  INF  INF  INF  INF  INF  INF  INF  0  INF
INF  INF  INF  INF  INF  INF  INF  INF  INF  0
```

All Pair Shortest Paths

Time Analysis(Time in seconds):

Test Case1:

```
Number of nodes: 10
Number of edges: 10
Time taken by Serial code: 3.6853e-05
```

```
Number of nodes: 10
Number of edges: 10
Time taken by:
1 threads: 9.0422e-05 with Speedup: 0.407567
2 threads: 0.000276262 with Speedup: 0.133399
3 threads: 0.000170046 with Speedup: 0.216724
4 threads: 0.000156158 with Speedup: 0.235998
5 threads: 0.000181116 with Speedup: 0.203477
6 threads: 7.2514e-05 with Speedup: 0.508219
7 threads: 0.000308223 with Speedup: 0.119566
8 threads: 0.01443 with Speedup: 0.00255392
9 threads: 0.00815816 with Speedup: 0.00451732
10 threads: 0.00132186 with Speedup: 0.0278797
```

Test Case2:

```
Number of nodes: 1000
Number of edges: 600000
Time taken by Serial code: 3.76041
```

```
Number of nodes: 1000
Number of edges: 600000
Time taken by:
1 threads: 5.78999 with Speedup: 0.649468
2 threads: 3.51029 with Speedup: 1.07125
3 threads: 2.52886 with Speedup: 1.487
4 threads: 2.23024 with Speedup: 1.6861
5 threads: 3.05391 with Speedup: 1.23134
6 threads: 2.65786 with Speedup: 1.41483
7 threads: 2.51371 with Speedup: 1.49596
8 threads: 2.57142 with Speedup: 1.46239
9 threads: 2.92001 with Speedup: 1.28781
10 threads: 2.94187 with Speedup: 1.27824
```

Test Case3:

```
Number of nodes: 500
Number of edges: 3500
Time taken by Serial code: 0.568052
```

```
Number of nodes: 500
Number of edges: 3500
Time taken by:
1 threads: 0.664688 with Speedup: 0.854614
2 threads: 0.393961 with Speedup: 1.4419
3 threads: 0.366304 with Speedup: 1.55077
4 threads: 0.311979 with Speedup: 1.8208
5 threads: 0.392449 with Speedup: 1.44746
6 threads: 0.321295 with Speedup: 1.76801
7 threads: 0.560332 with Speedup: 1.01378
8 threads: 0.497999 with Speedup: 1.14067
9 threads: 0.401543 with Speedup: 1.41467
10 threads: 0.383011 with Speedup: 1.48312
```

Observations:

When the number of nodes and edges are very less then the speedup is less than one as the time gets wasted in thread creation. But as the problem size increases, we see a good amount of speedup in the parallel algorithm. For test case 2, the speedup is highest with 4 threads. For test case 3, the speedup is highest with 4 threads.