**Objective:**

We want to analyse if there is a difference in running the Floyd-Warshall algorithm using single thread and multi threads.

The pseudocode for only one threads was given and discussed in class. Putting it in the report seemed redundant.

**Pseudocode for multithreads:**

We have two data structures:

1.graph which is the matrix that contains the graph from user

2.Dist which is the matrix that contains the shortest distance between each node

Each instance (node) inherits two characteristics: i //current node

k// all other nodes

**About printgraph()**

1.The program starts by asking the user for number of nodes and number of vertices

We need to validate number of nodes and vertices

It increments the number of numOfNodes based on the very first input from user

It increments the number of numOfVertices based on the second input from user

2. It creates the initial graph based on user input which consists of 0,actual weight, and INF

3.It also asks the user for the edge, and the associated edge

4.We dynamically allocate an array for each vertex. Note that vertex[2] contains the weight

5. We add the weight of each edge to the graph matrix

**About worker()**

**From slides:**

Each thread either reads or writes

Uses readCountLock for reading

Uses semphoreRW for writing

*sem\_wait- Locks a semaphore and returns 0. If the semaphore value is zero 0, the calling process gets blocked. Returns -1 if unsuccessful (deadlock, interrupt etc..)*

*sem\_post- It increments the value of the semaphore and wakes up a blocked process waiting on the semaphore, if any*

**Pseudocode**

Each thread must read first

They acquire the readlock

Stop any other threads from writing

sem\_wait(&semphoreRW);

Release lock

Condition : if (dist[i][k] != INF && dist[k][j] != INF && dist[i][k] + dist[k][j] < dist[i][j])

Start reading

If no threads are writing, write

Release lock

Stop thread from writing

Do: dist[i][j] = dist[i][k] + dist[k][j];

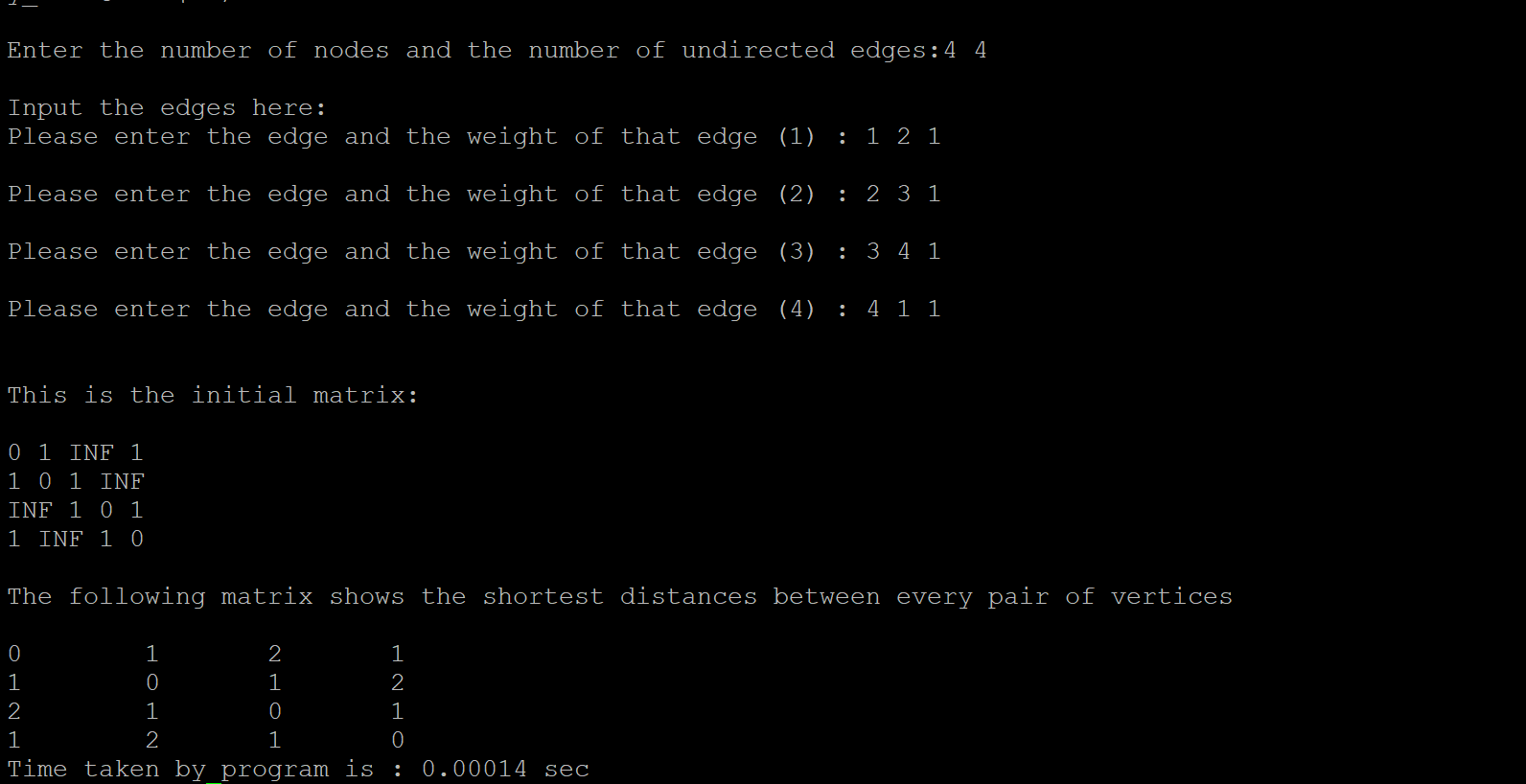
Write the result

Else //do nothing

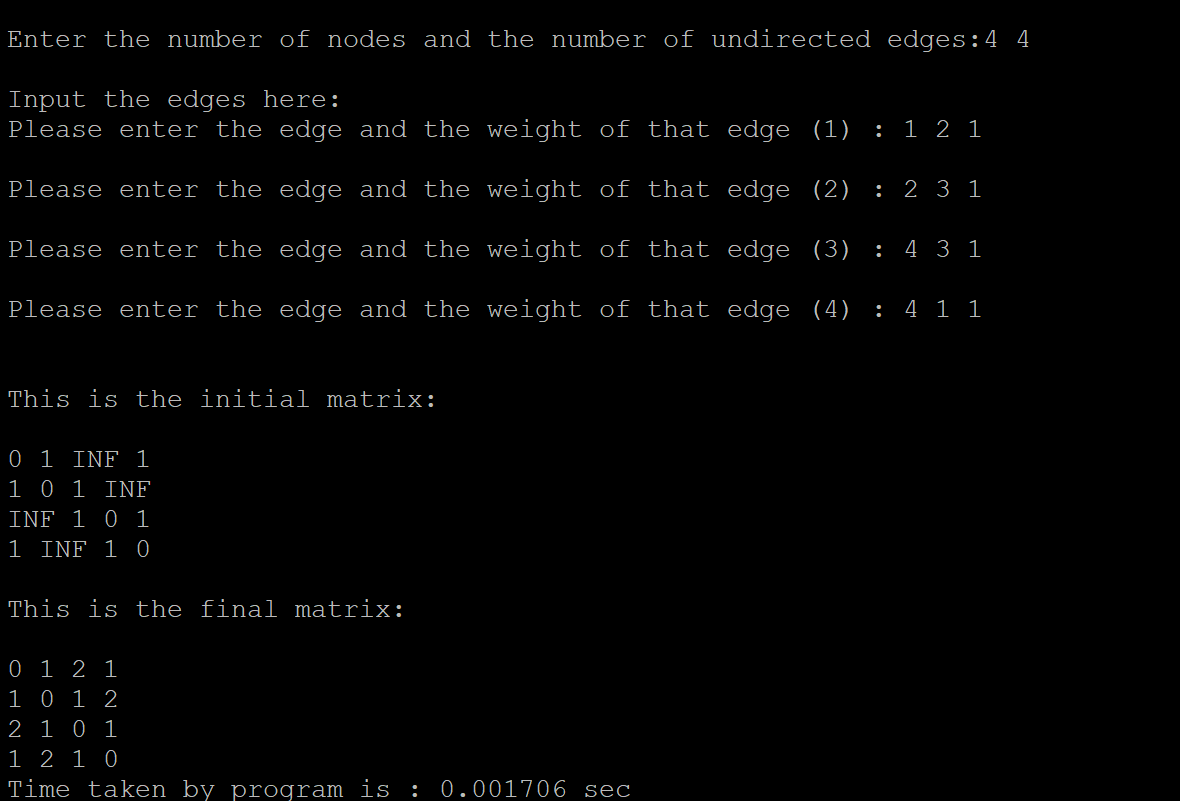
Start reading

If no threads are writing, write

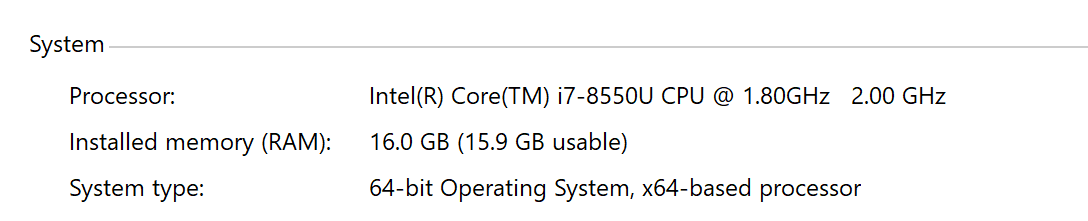
Release lock

**Result using one thread**

**Result using multiple threads:**



**Architecture of machine:**



**Graph for running with 4,10,100 and 1000 vertices using single and multiple threads.**

My computer could not run the program for 1000 vertices using multiple threads.

When I tested for 500 threads, it was still not working. It seems that the limit on the number of thread is about 500 threads.

**Inferences/Conclusion:**

From my results, I conclude that the program using only one thread was able to handle large amount of inputs while the multi-threads program was not.

The time taken for the single thread program was always less than the multithreads one.

**References:**

1.Lecture Slides

2.GeekForGeeks in general

3.Tutorial point in general

4.Followed some basic C++ format from but NOT pseudocode or code from:<https://github.com/Trav996/Multithreading-and-Synchronization/blob/master/threads.cpp>