CS69001 Computing Laboratory – I

Assignment No: C4

Date: 06–November–2019

Implement the parallel level-by-level breadth-first-search (BFS) algorithm of Assignment C3 using POSIX threads. The master thread creates P worker threads which run the BFS algorithm in parallel with appropriate synchronization and mutual exclusion. The master thread acts as the coordinator.

Global variables

All shared data are to be declared as global variables. This includes the $N \times N$ adjacency matrix of the graph, the *visited* array, the queue Q of vertices, the two ends F and B of Q, and the $P \times 2$ chunk-definition array. In this assignment, you do not need the variable n_{done}, because the barrier keeps track of the count. The mutexes and the barrier may also be defined globally.

Mutexes

A mutex M_Q is used for the mutually exclusive write access to Q. Moreover, P^2 mutexes M_V will guard the access to the *visited* array indices. All these mutexes are initialized to the unlocked state.

Barriers

Use a single barrier L at all synchronization points. This barrier should be so initialized that all of the $P+1$ threads (the master thread and P worker threads) must participate in a wait call on it for synchronization. The same barrier L is reused on every occasion when a synchronization is needed.

Synchronization

The tasks of the master thread and the worker threads would proceed as follows.

Master thread	i-th worker thread
Generate a random adjacency matrix for the graph. Print the graph. Enqueue vertex 0 to Q, and set $F = B = 0$. Initialize the *visited* array (only 0 is visited). Initialize M_Q, M_V, and L.	
Create P worker threads.	
Repeat until BFS stops:	Repeat until BFS stops:
Divide the interval $[F, B]$ into P chunks. Write the chunk boundaries in C. Wait on the barrier L.	Wait on the barrier L.
Wait on the barrier L.	Read the assigned chunk from $C[i]$. Store new BFS links in local memory. Lock the mutex M_Q. /* Critical section */ Enqueue by copying from local memory. Unlock the mutex M_Q. Wait on the barrier L.
Wait for all worker threads to exit.	Exit.
Destroy the mutexes and the barrier. Exit.	

Sample output: A verbose sample output file is separately linked from the lab web site.

Submit a single C/C++ source file. Do not use global/static variables other than those shared by the threads.
Do not use STL queues or vectors. The shared arrays *visited* and Q are to be managed by you.