# PARALLEL AND CONCURRENT PROGRAMMING

# PRELIMINARY REPORT

**JILLELA DEEPAK REDDY - CS18BTECH11016**

**ABBURI VENKATA SAI MAHESH - CS18BTECH11001**

November 29, 2020

---

**This document is generated by LaTeX**

---

## 1 TT-Lock:-

TT-Lock (Time Tampering lock) is a lock mechanism with FCFS property based on the CPU time they are requesting to access critical section of a specific program. Each thread writes it's time of request for critical section that is time stamp in a priority queue with earlier time stamp is prior to later time stamp. We are being precise in FCFS by considering time stamps because we believe that there are always programs where nano seconds of execution time and the corresponding relative order of execution of threads effects the expected output in an unexpected way.In this lock execution mechanism, the threads will be given access to critical section based on the order of time stamps corresponding to relative threads in the priority queue. We are using priority queue because there may be a situation where the threads may request the access to critical section earlier but failed to write into the queue subsequently, So we intended to use priority queue which ensures that the threads with earlier time stamps are given more priority when compared with the threads with the later time stamps in turn ensuring the FCFS property.

## 2 PARALLELIZATION IN MST:-

We present new parallel prim algorithm for finding Minimum Spanning Tree(MST). We use cut property of graph to grow tress simultaneously in multiple threads and a merging mechanism when threads collide. We assign the adjacency matrix in 1D fashion in such a way that each thread will be getting $\frac{n \times n}{P}$ part of the array for P processors. Each processor chooses a root node and grows tree in similar fashion of serial Prim approach and when the tree finds a nearest node that doesn't belong to any other tree it can add the node, whereas if the node belongs to another tree then it must stop growing and start with a new root. For Example, when a collision occurs between two threads i and j (i < j), thread j merges with i. Thread i continues to grow the tree from there and thread j picks another node randomly and grows a new tree. In the end, we get different connected components(which are trees) and some isolated vertices. No two trees share a vertex because merging was avoided. Now Find-Min step of Boruvka Algorithm is used to shrink each of the components into a super node.