Design:

In this design we have 3 cases for m and n (m=number of threads ,n=number of nodes):

case1: m>n

Here we assigned task like each thread will visit only one node that is only min(n,m) threads will visit each node. So rest n-m threads will only be constructed but they won't have work to do as the all the nodes in the graph are visited by the previous threads.

Case2:m==n

here also we assigned the task of discovering each node to one thread.

Case 3:m<n

Here we assigned tasks such that (m-1) threads will discover floor(n/m) nodes and the last thread will discover (n/m+n%m) nodes.

So we implemented BFS using threads where each thread task is to push children of a node into the queue and then next thread will visit the nodes and then again push their children to the queue.

Functions:

do_work() function handles the multi-threaded BFS logic by checking the child is visited or not, wheather the queue is empty or not.

worker() function is used to assign the work to the threads

Synchronization:

For synchronization we used mutex lock of mutex library in cpp