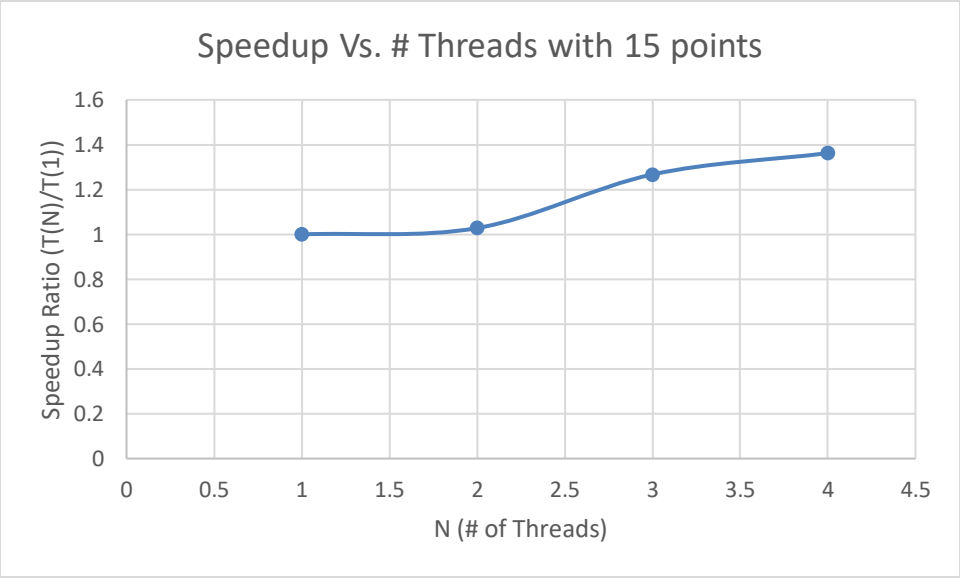
Q2

The algorithm running sequentially works as follows: it looks through all edges and finds direct common pairs of points between the two points on such edges (essentially a path of size 2). It checks if there is exactly 1 edge that an edge created by such pairs would intersect, and that the edge being inspected is the one that's being intersected. The angles between the pair of points and their respective vectors connecting them to the edges points must also have an added angle greater than PI radians. If so, this edge is "flipable" and will be replaced by the edge connected by the pair of points. When this occurs, the algorithm must relook at all edges of the graph that now includes the new edge and excludes the old one. If no flip is to be done, then the algorithm checks out a new edge. This goes on until no flips can occur.

The parallel implementation is quite similar, however threads are delegated different edges based on when which thread gets to an edge first. If an edge is locked, it moves onto the next open edge and examines it. Because multiple threads are interacting with similar edges (as they must consider attached edges that may even be locked by other threads), some sort of synchronization is required. The flipping portion is the most critical part of the algorithm, as edges are removed and added to the graph. Thus locking this action is necessary. Whenever an edge is deemed "flipable", it first checks if an edge associated to it had previously been removed while being locked out of this action. If it had been removed, it moves on to look at the next edge in the graph. If all associated edges still exist, it flips the edge and is set to relook at the entire graph.



This graph displays the speedup from 1 thread to up to 4 threads. There is a slight increase in performance as the number of threads increases. This makes sense due the fact that while a thread that has flipped an edge re-examines all edges, the other threads modify the edges ahead of the list. This way the next iteration of a threads traversal through the graph will have less flips to perform as they may have already been caught. Additionally, these tests were run on a quadcore processor. Thus, the threads are able to run on their own CPU's which would reduce the amount of overhead from time slicing.