Parallel Programming
Assignment 1

Krunal Shah (2015EE10476)

1 Parallel convex hull

1.1 Algorithm

The quick hull algorithm is implemented. The algorithm proceeds as follows

1. Find the leftmost and the rightmost points and divide the rest of the points into two sets belonging to either side of the line formed by the two points. The observation is that the leftmost and the rightmost points must belong to the convex hull.
2. We will now call our method recursively on both these sets.
3. For the two points and one of the set, we find the point from the set farthest from the line connecting the two points, this point must belong to the convex hull.
4. So we get a triangle formed by the farthest point and the two points of the line. The points inside the triangle cannot belong to the convex hull and so we recursively call our methods for the two lines involving the farthest point and the regions created by these lines sans the triangular region which amounts to two calls.
5. We keep on calling the methods recursively till no more points are left.

1.2 Implementation

1. The two dimensional vector *pts* stores all the black points as their coordinates.
2. The global variable *counter* keeps a track of the number of threads currently in use and does not let the number of threads exceed *NUM_THREADS*.
3. The method *findhull* takes three arguments, (int a, int b, vector set_pts) where the vector is the set of points lying on one side of the vector joining *a* and *b*.
4. It first finds the point farthest from the vector in the set and adds it to the hull if not already added and then constructs two set of points from *set_pts* which are used to call the recursive methods with *(a,maxpt)* and *(maxpt,b)* as the line.
5. To not let counter exceed *NUM_THREADS*, *findhull* spawns a new thread for it's recursive executions only if the value of *counter* is less than *NUM_THREADS*.

References

[1] C. Bradford Barber, David P. Dobkin, Hannu Huhdanpaa *The Quickhull algorithm for Convex Hulls*. Journal ACM Transactions on Mathematical Software (TOMS) Volume 22 Issue 4, Dec. 1996 Pages 469-483