15-499: Parallel Algorithms **Lecturer:** Guy Blelloch
Topic: Computational Geometry **Date:** March 17, 2009
Scribe: Lawrence Jesper

17.1 Convex Hull

17.1.1 Problem Basics

Input: A set of points in 2D space
Output: Intuitively it is the elastic band around all points in the input set. Formally, it is a list of points x_0, x_1, ..., x_{n-1} such that connecting x_i to x_{i+1}, $0 \leq i \leq n-2$, and x_{n-1} to x_0 creates a convex figure that contains every input point.
Motivation: This problem arises in engineering where modeling an items exact dimensions would be too expenisve/inefficient. Instead the item is modeled as its simpler convex hull. You are guaranteed that if convex hulls do not intersect, the items themselves do not intersect.

17.1.2 Quick Hull: Divide and Conquer Algorithm

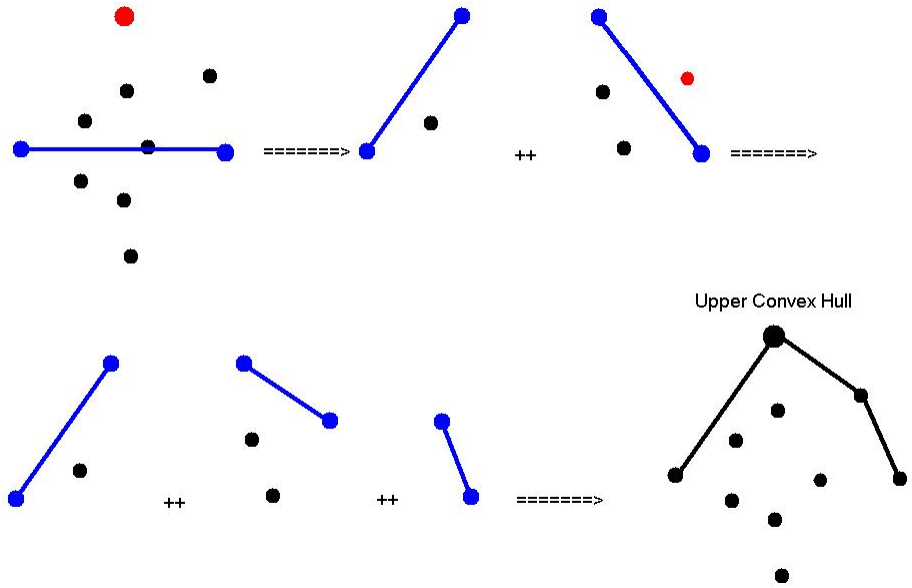
Upper Hull Simplification: We will simplify the problem by designing an algorithm that finds the upper hull from the left most point to the right most point. Clearly, an algorithm that finds the upper hull can then be used to find the lower hull (and therefore the entire hull) by chaning which direction is up. This algorithm is similar to quick sort although no randomization is involved.

The algorithm:

$P =$ input points
$l =$ left most point in P
$r =$ right most point in P

QuickHull(P,l,r) =
if $|P| = 0$ then return []
else
$\bar{D} =$ line from l to r
$P' = p \in P$ s.t. p above $\bar{D}$
$pivot = (x_p, y_p)$ s.t. (x_p, y_p) is the farthest point from $\bar{D}$ (this point is guaranteed to be on the hull)
$L = \{(x, y) \in P | x < x_p\}$
$R = \{(x, y) \in P | x > x_p\}$
return QuickHull(L,l,$pivot$) ++ [$pivot$] ++ Quickhull(R,$pivot$,r)

Algorithm Picture:



Runtime Analysis:
Worst case: Each round splits off one point resulting in n rounds. Each round is $O(n)$ work for determining the pivot, and $O(n)$ work for packing into L and R. The rounds are $O(log\ n)$ depth for determining the pivot (or $O(loglog\ n)$ using the algorithm from HW4) and $O(log\ n)$ depth for packing into L and R.
$W(n) = W(n-1) + W(0) + O(n) = O(n^2)$.
$D(n) = max(D(n-1), D(0)) + O(log\ n) = n * log(n)$.

Worst Case:
All nodes sorted into left side each iteration

Best case: (only 1 node in the set)
Work: $O(n)$
Depth: $O(log\ n)$

Alternate Algorithm: Instead of choosing the pivot as the element farthest from $\bar{D}$, first find the median element by x-coordinate, m. If m is on the convex hull then use m as the pivot. If not, use the "bridge" that spans m. The "bridge" is defined as the two elements farthest from $\bar{D}$ (and thus guaranteed to be on the convex hull) whose line segment crosses the line perpendicular to $\bar{D}$ through m (see picture). The call then puts both points of the bridge in the return list. Work: O(n). Depth: O(log n). Proof not given.

Bridge (green) over Median (blue)