CS420+500: Advanced Algorithm Design and Analysis

Lectures: Jan 11 + Jan 13, 2017

Prof. Will Evans *Scribe: Omar AlOmeir*

In this lecture we:

- Did a quick group exercise
- Discussed reduction of a Convex Hull algorithm into a sorting algorithm to find a lower bound for Convex Hull algorithms
- Discussed the Element Uniqueness problem
- Discussed Linear Decision Trees

Handouts (posted on webpage):

- Group-work exercises (Wednesday Jan 11) (Not posted yet!)

Reading:

- Otfried Cheong's Algebraic Decision Tree Notes (Posted on webpage)

1 Reduction of Convex Hull algorithm

Is Graham Scan the fastest possible Convex Hull algorithm? How to show that every Convex Hull algorithm takes $\Omega(n \log n)$ *on worst-case inputs of size* n*?*

To answer those questions, suppose there exists a Convex Hull algorithm that's really fast. We assume our algortihm solves the Convex Hull problem correctly but we do not know how it works. The algorithm is shown in figure 1.

We can use this algorithm to build a really fast sorting box, as shown in figure 2.

We have two elements to our sorting box, input transformer I and output transformer O. I transforms input from an unsorted list or array (input of sorting algorithm) $x_1, x_2 ... x_n$ into a set of

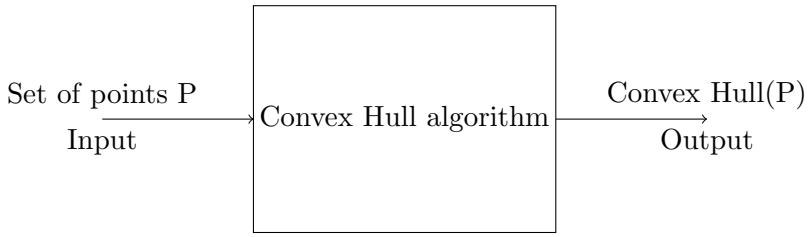

Figure 1: The (really!) fast Convex Hull algorithm is treated as a black box with input and output.

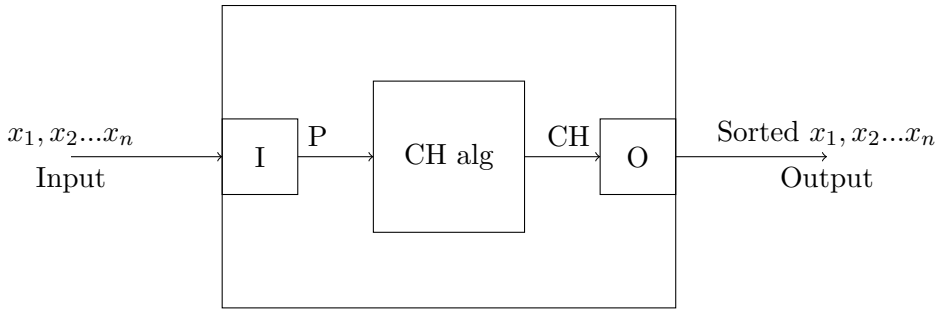

Figure 2: The Convex Hull algorithm reduced into a sorting algorithm with all the necessary components.

points P (input for the Convex Hull algorithm). O transforms output of the Convex Hull algorithm (set of points P) into sorted order list $x_1, x_2 ... x_n$ (output of sorting algorithm). These I and O transformers must be really fast ($\leq o(n\ log\ n)$). They must not be a bottleneck to the sorting algorithm.

Intuition for I and O:
I: Every point x_i is mapped to (x_i, x_i^2). Time complexity $O(n)$.
O: Find the smallest x-coordinate point then report Convex Hull points from it in counter-clockwise order. Time complexity $O(n)$.

To transform single values from the array $x_1, x_2 ... x_n$ into inputs for the Convex Hull algorithm (in the form of (x, y) points, the values in the array need a y-value. Giving the y value as x_i^2 creates a parabola and the Convex Hull will include all the points on the parabola. Reporting the points from smallest to largest x-coordinates will give a sorted list of x values. The time complexities of I and O show that they will not be a bottleneck.

To analyze run-time:
Let $T_{CH}(n)$ = run time of fastest Convex Hull algorithm
Let $T_{sort}(n)$ = run time of fastest sorting algorithm
$T_{sort}(n) \in \Omega(n \log n)$

We know the following:

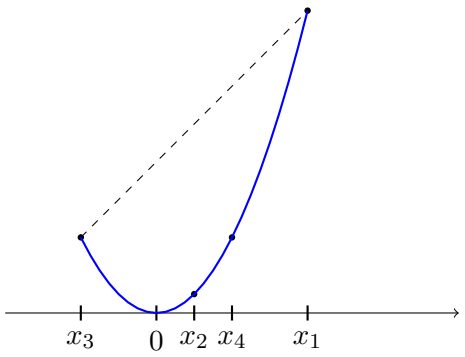

Figure 3: Points in $P : x_1, x_2 ... x_n$ mapped to (x_i, x_i^2). All points in P, since they lie on a parabola, are vertices of the convex hull of P.

1. $T_{sort}(n) \leq T_{CH}(n) + O(n) + O(n)$
2. $T_{sort}(n) \geq c.nlogn$

From 1, and 2 we can deduce that $T_{CH}(n) + O(n) + O(n) \ \geq c.nlogn$
Thus $T_{CH}(n) \geq c.nlogn - \alpha n \in \Omega(nlogn)$

There are two issues that come up with this reduction of Convex Hull algorithm into sorting:

1. Finding unordered Convex Hull vertices can possibly be done a lot faster
2. We know $T_{sort}(n) \in O(nlogn)$ only in a comparison based computational model. Convex Hulls cannot be computed in such a model (lower bound is infinity for the Convex Hull algorithm in this model). The reduction to sorting uses at least the transformation from x_i to (x_i, x_i^2), this transformation requires an algebraic operation (multiplication) that cannot be performed in the comparison based model. So we cannot tell whether Graham Scan is a good algorithm or not.

To address these issues we will we can show that Element Uniqueness, a simpler problem than sorting, can be solved using an algorithm for unordered (or even counting) Convex Hulls. This algorithm takes $\Omega(nlogn)$ time on a more powerful model of computation: The Algebraic Decision Tree model.

2 Element Uniqueness problem

Given $x_1, x_2...x_n$
Output No if $x_i = x_j \ for \ i \neq j$
Yes otherwise

3 Linear decision trees (Algebraic decision trees of degree 1)

A linear decision tree is a ternary tree model of computation. It has a fixed input of size n. Every internal node has a vector label of real numbers $< a_0, a_1...a_n >$ and 3 child edges labeled -, 0, +.

Figure 4 shows an example of a linear decision tree with n = 2.

To use a linear decision tree with input $x_1, x_2...x_n$, start at the root with label $a_0, a_1...a_n$. Evaluate $a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n$ and follow the child edge with the sign of the result. Repeat until you reach a leaf.

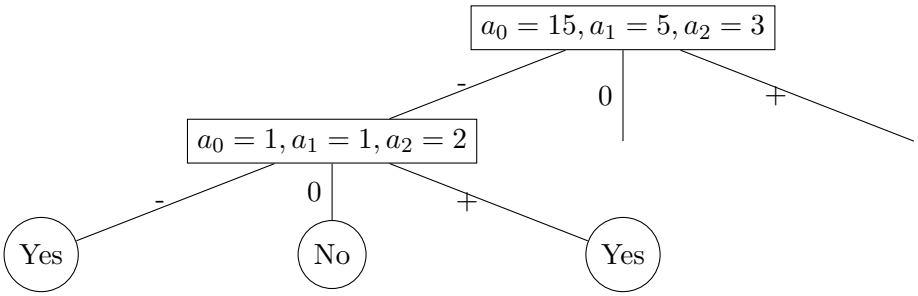

Figure 4: A linear decision tree with n = 2. The leaves have yes/no answers as output for element uniqueness.