In this lecture we:

- Gave an example of how to use features of input space to proof computation lower bounds
- Introduced another convex hull algorithm: the Chan’s algorithm

1 Algebraic Decision Tree

1.1 Linear Decision Tree Example

Lemma 1. Any linear decision tree that computes $F$ has height at least $\lceil \log_3(\sum_{t \in \text{outputs}} \#F_t) \rceil$.

Theorem 1. Any linear decision tree that computes Element Uniqueness has height $\Omega(n \log n)$, where $n$ is the number of elements.

Proof Idea: are $x = (1, 2, 3, \ldots, n)$ and $y = (1, 3, 2, \ldots, n)$ in the same connected component? (NO).

let $x$ be a vector of $n$ unique numbers and $y \neq x$ be an permutation of $x$. There must be indices $i$ and $j$ such that $x_i < x_j$ and $y_i > y_j$. Any continuous “path” from $x$ to $y$ must contain a point $z = z_j$ by intermediate value theorem. $z$ is a NO input, so $x$ and $y$ are not in the same connected component (notice both $x$ and $y$ are YES inputs).

Therefore, none of the $n!$ permutations of $x$ are in the same connected component, and thus $\#F_{\text{yes}} \geq n!$.

Side: What is a “path”?

Let $p$ be a path from $x$ to $y$. $p : [0, 1] \rightarrow \mathbb{R}^n$, $p(0) = x$, $p(1) = y$. Let $q(t) = p(t)i - p(t)j$ (the different between ith and jth coordinate in point $p(t)$). Because $x_i < x_j$ and $y_i > y_j$, we have $q(0) > 0$ and $q(1) < 0$. $q$ is a continuous function, so there must exist a $0 < t < 1$ such that $q(t) = 0$. ($p(t)$ is the $z$ in the argument above.)

1.2 General Algebraic Decision Tree

A dth order algebraic decision tree is like a linear decision tree but functions in internal nodes are dth order polynomials.

Lemma 2. The number of connected components of input space that can reach the same leaf is less than or equal to $d(2d - 1)^{n+h-1}$ in the tree with height $h$.

For example, under this model, the Element Uniqueness has $\Omega(d(2d - 1)^{n+h-1})$ connected components that can reach the same leaf. We have proved that the number of YES connected components is at least $n!$. Notice that the tree can have at most $3^h$ leaves, which implies the inequality:
3^h d(2d + 1)^{n + h - 1} \geq n! \Rightarrow h \geq n \log n \text{ for some constant } d. \text{ Therefore the Element Uniqueness takes } 
\Omega(n \log n) \text{ time under the linear decision tree model.}

2 Chan’s Algorithm for Convex Hull

2.1 Guess given

Given \( n \) points \( P \) and a guess \( h \) for the number of hull vertices.

1. Divide points into \( n/h \) groups of size \( h \) (except the last one).
2. Do graham scan on each group
3. Find the lowest point \( p_0 \)
4. Do giftwrapping (Jarvis March) for \( h \) steps: find the right tangent from \( p_i \) to each group hull (with a binary search); let \( p_{i+1} \) be the rightmost of these tangent points; if \( p_{i+1} = p_0 \), output the hull. Increment \( i \).
5. output “\( h \) is too small”

2.2 Analysis

The slowest steps are 2 and 4. Step 2 takes \( O(n/h \times \log h) = O(n \log h) \). In step 4, finding the right tangent in a group hull takes \( O(\log h) \), and finding the rightmost one of all groups takes \( O(n/h \times \log h) \), so step 4 takes \( O(n \log h) \). The algorithm takes \( O(n \log h) \) time, which indicates it is an output sensitive algorithm, just like Javis March.

2.3 Generation of guesses

We use an aggressive guess sequence: 4, 16, 256, \ldots, \( 2^{2^t} \).

Let \( h^* \) be the true number of hull vertices. The time complexity is

\[
\sum_{h = 2^t \text{ until } h \geq h^*} \Omega(n \log h) = \sum_{t=1}^{\lceil \log \log h^* \rceil} O(n2^t) = O(n \sum_{t=1}^{\lceil \log \log h^* \rceil} 2^t) \approx O(n \log h^*)
\]