CPSC 420 Lecture 3 : Today's announcements:

- HW1 available on Gradescope, due Jan 19, 23:59
- Examlet 1 on Jan 27 in class.
- Reading: Chan's Algorithm [Wikipedia]
- Reading: Voronoi Diagrams [Computational Geometry: Algorithms and Applications 3rd Edition pg 147]

Today's Plan

- Convex hulls
  - Optimal algorithm?
  - Chan's Algorithm

Graham's Scan Run Time

1. Finding p_1 takes $\underline{\quad O(n) \quad}$ time
2. Sorting by angle takes $\underline{\ O(n \log n)\ }$ time
3. Put $p_1 p_2 p_3$ on a stack S takes $\underline{\quad O(1) \quad}$ time
4.
```
for i = 4 to n
   while not LeftTurn(S[top-1], S[top], p_i)
         pop(S)
   push p_i onto S
return S
```

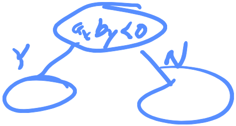
   One iteration of for-loop causes $< n$ pops $\Rightarrow$ $\underline{\quad O(n^2) \quad}$ time

   But, over all iterations, #pushes $< n$ and #pops $<$ #pushes

   So total time taken by for-loop is $\underline{\quad O(n) \quad}$

   Graham's Scan runtime: $O(n \log n)$

Faster Convex Hull?



Is this the fastest algorithm for Convex Hull?

How powerful is our computer?

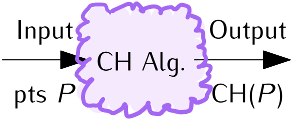
- It can multiply, add, subtract, compare two real numbers in one step.
- It cannot wrap a string around n objects in linear time.

This is called an Algebraic Decision Tree model of computation.

Is this the fastest algorithm using an Algebraic Decision Tree model computer for Convex Hull?

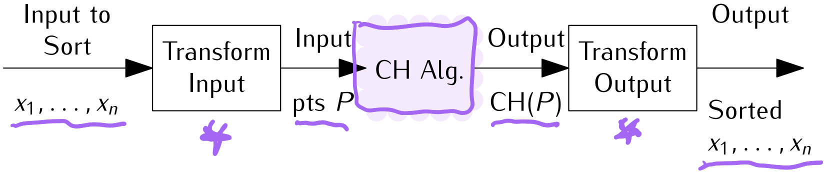
Faster Convex Hull?

Suppose there exists a really fast Convex Hull algorithm.



Faster Convex Hull?

Suppose there exists a really fast Convex Hull algorithm.



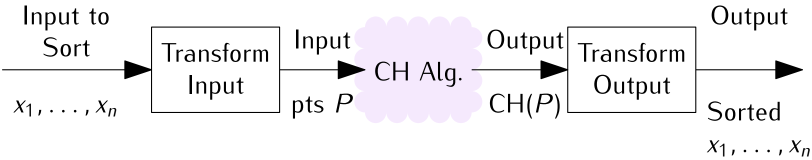
Create an alg. that transforms an input to the sorting problem into an input to CH problem...
Make this really fast.

let CH Alg. do all the hard work...

and then transform CH output into the answer to the sorting problem.
Make this really fast.

Faster Convex Hull?

Suppose there exists a really fast Convex Hull algorithm.

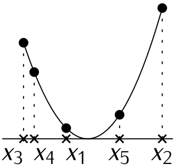


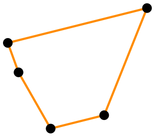
Create an alg. that transforms an input to the sorting problem into an input to CH problem... Make this really fast.

let CH Alg. do all the hard work...

and then transform CH output into the answer to the sorting problem. Make this really fast.

for all i:
$x_i \to (x_i, x_i^2)$





Find hull pt w/ min x-coord
Output x-coords in ccw order

Transform Input

Transform Output

Why did we do this?

We already have good algorithms for sorting. Why make this complicated sorting algorithm?

Why did we do this?

We already have good algorithms for sorting. Why make this complicated sorting algorithm?

Because we know Sorting complexity is $\Omega(n \log n)$.
(i.e. fastest alg. for sorting takes $\geq cn \log n$ steps for large n)

We've just constructed a sorting algorithm that takes time

Input trans cost

$$T(n) = T_I(n) + \boxed{T_{CH}(n)} + T_O(n)$$

output trans cost

We know $T(n) \geq cn \log n$ (sorting complexity).
We know $T_I(n) \leq c_I n$ and $T_O(n) \leq c_O n$ for some constants c_I and c_O (from the input and output transformations).
That means (since $T_{CH}(n) = T(n) - T_I(n) - T_O(n)$)

$$T_{CH}(n) \geq cn \log n - c_I n - c_O n \in \Omega(n \log n).$$

This holds for **any** convex hull algorithm, so the complexity of convex hull is $\Omega(n \log n)$.

What about Jarvis?

Jarvis March takes time $O(nh)$.
For small enough h, this isn't $\Omega(n \log n)$.

Our lower bound only cared about one measure of the input: the number of points n.

With two measures, number of points n and size of output convex hull h, we might find a better algorithm...

Chan's Algorithm

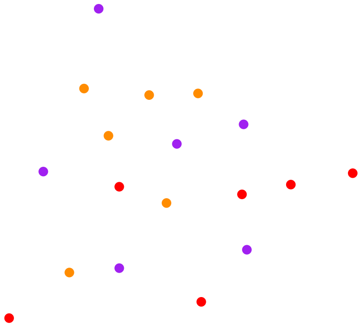
Given n points P and a guess g for the number of hull points...

1. Divide P into n/g groups of g points
2. Use Graham's Scan to find the convex hull of each group in $O(g \log g)$ time per group

Chan's Algorithm

Given n points P and a guess g for the number of hull points...

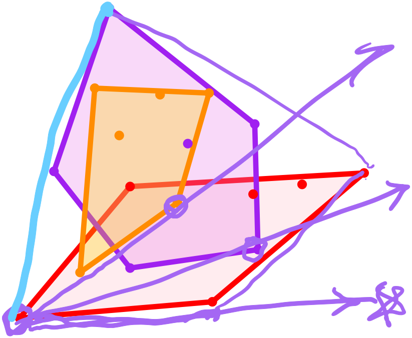
1. Divide P into n/g groups of g points
2. Use Graham's Scan to find the convex hull of each group in $O(g \log g)$ time per group



Chan's Algorithm

Given n points P and a guess g for the number of hull points...

1. Divide P into n/g groups of g points
2. Use Graham's Scan to find the convex hull of each group in $O(g \log g)$ time per group



Chan's Algorithm

Given n points P and a guess g for the number of hull points...

1. Divide P into n/g groups of g points
2. Use Graham's Scan to find the convex hull of each group in $O(g \log g)$ time per group
3. Find the lowest point p_0
4. Gift-wrap (Jarvis March) these convex hulls for g wrap steps. To find the next hull point p_{i+1}
   4.1 find the right-tangent from p_i to each group hull in $O(\log g)$ time per group
   4.2 p_{i+1} is rightmost-by-tangent-angle of these tangent points
   4.3 If $p_{i+1} = p_0$ output hull
5. Output "g is too small!"

Total time: $O(n \log g)$.

How to generate guesses

Start with $g = 4$ then $g = 16$ then $g = 256$ ...

$$g = 2^{2^t} \text{ on the } t^{\text{th}} \text{ try.}$$

Total run time (until $g \geq$ hull size h):

$$\sum_{t=1}^{\lceil \lg \lg h \rceil} O(n \log(2^{2^t})) = \sum_{t=1}^{\lceil \lg \lg h \rceil} O(n 2^t) = O(n \sum_{t=1}^{\lceil \lg \lg h \rceil} 2^t) = O(n \lg h)$$