9 January 2025

Complexity of the Incremental Algorithm

Assume there are n points.

1. Sort the points: ~~$O(n^2)$~~ $O(n \cdot \log n)$
   (example: MergeSort)
2. Loop over points: k from 4 to n — $n-3$ iterations
   Loop over existing hull edges — $\leq n$ edges
   } $O(n^2)$

Overall complexity: $O(n^2)$

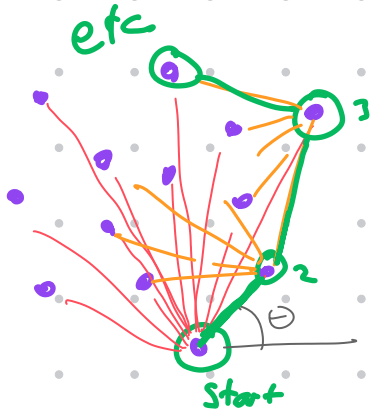
→ means runtime is bounded by $c \cdot n^2$ for some constant c and large n

---

GIFT-WRAPPING ALGORITHM

Complexity (n points)

$n \leftarrow$ 1. Start with the lowest point (if there is a tie, choose the rightmost).



$O(n^2)$ / $O(n \cdot h)$

2. Compute angles from starting point to all other points. The point with the smallest angle is the next hull point.
3. Repeat step (2), working around the convex hull until returning to the starting point.

(step 3: at most n times or exactly h times)

Let h be the number of vertices in the convex hull.

$h \leq n$

Gift Wrapping Algorithm

MATH 261 Computational Geometry

Input: a set S of n points in the plane, specified by xy-coordinates

Output: a list L of vertices of conv(S) in counterclockwise order

Algorithm:

1. Start with the lowest point in S (if there is a tie, choose the rightmost).

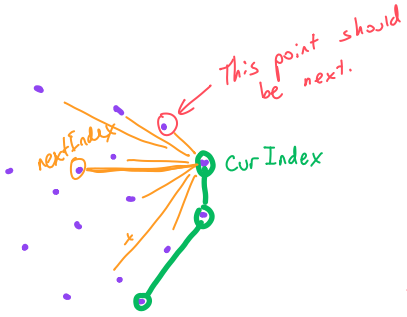
2. Draw lines from the starting point to all other points in S. The point whose line makes the largest angle with the negative horizontal axis is the next point on the hull.

3. Repeat #2, working counterclockwise around the hull. When you return to the starting point, you are done.

Answer the following questions:

1. Draw your own set of points. Work through the gift-wrapping algorithm by hand to find the convex hull of your points.

2. The algorithm above is phrased in terms of angles, but we've said that computing angles is computationally expensive and prone to numerical errors. How could you implement the algorithm using LeftOf queries instead?



Pseudocode:

Let curIndex be the index of the current point on the convex hull.

Let nextIndex = 1 if curIndex ≠ 1, otherwise nextIndex = 2.

→ For i from 2 to n:
  If i == nextIndex or i == curIndex: continue.
  If pts[nextIndex] is left of the line from pts[curIndex] to pts[i]:

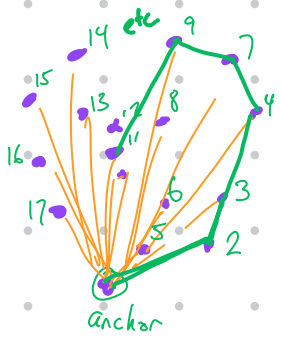
    Let nextIndex = i.

3. What is the computational complexity of the gift wrapping algorithm?

4. On the course website you will find an incomplete Mathematica implementation of the gift wrapping algorithm. Fill in the missing code to complete the implementation.

GRAHAM SCAN ALGORITHM

1. Choose the lowest point as an anchor.



2. Sort all other points by their angle from the anchor.

3. Construct convex hull following this order of points:

   REPEAT:
   - append the next point to the hull
   - remove any reflex vertices (right turns) that result.

Graham Scan Algorithm

MATH 261 Computational Geometry

Input: a set S of n points in the plane, specified by xy-coordinates

Output: a list L of vertices of conv(S) in counterclockwise order

Algorithm:

O(n)
O(n·log(n))
O(n)

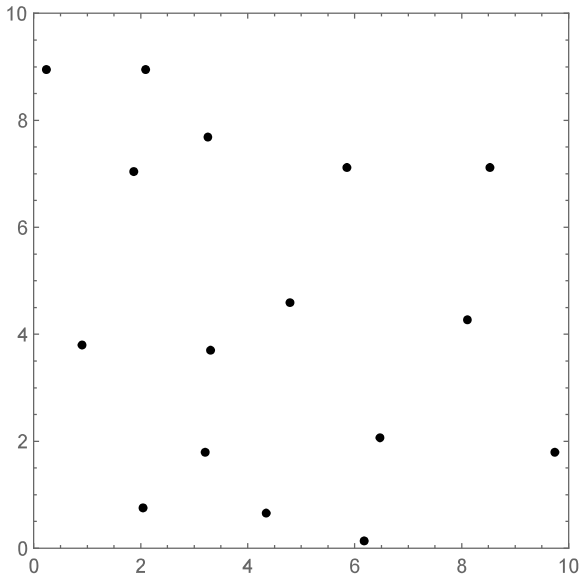
```
anchor = point with lowest y-coordinate
sorted = other points, sorted by their angle from anchor

hull = {anchor}
for i = 1 to length(sorted):
  append sorted[i] to hull
  while the next-to-last vertex of hull forms a right turn:
    remove the next-to-last vertex from hull

return hull
```

Answer the following questions:

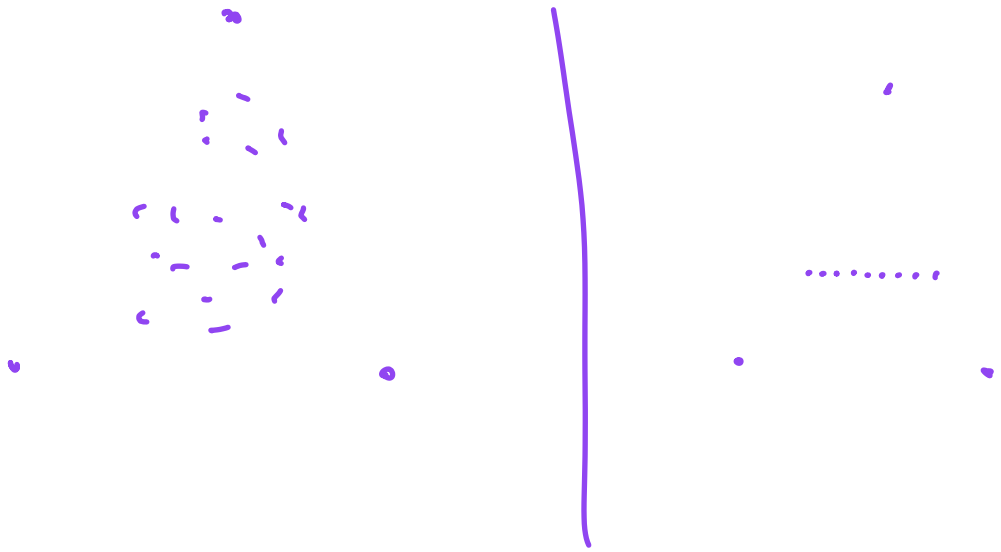
1. For the following configuration of points, how many points does the algorithm add to the convex hull, only to remove them later? Indicate the order in which these points are added and removed.



2. What is the computational complexity of the Graham Scan algorithm?

O(n·log(n))

3. What configuration of points would cause worst-case runtime for the Graham Scan algorithm?



4. What degenerate configurations of points could cause problems for a naive implementation of the Graham Scan algorithm? How would you avoid such problems?

Collinear points