Convex Hull
Given a set \( P \) of points in 2D, their convex hull is the smallest convex polygon that contains all points of \( P \).

Convex Hull
The problem: Given a set \( P \) of points in 2D, describe an algorithm to compute their convex hull.

Convex Hull
Output: list of points on the CH (in boundary order)

Convex Hull
A polygon \( P \) is convex if for any \( p, q \) in \( P \), the segment \( pq \) lies entirely in \( P \).

Convex Hull
- One of the first problems studied in CG
- Many solutions
  - simple, elegant, intuitive, expose techniques
- Lots of applications
  - robotics
  - path planning
  - partitioning problems
  - shape recognition
  - separation problems
Path planning: find (shortest) collision-free path from start to end

Applications

- Start
- End
- Obstacle
- It can be shown that the path follows CH(obstacle) and shortest path s to t is the shorter of the upper path and lower path

Shape analysis, matching, recognition
- Approximate objects by their CH

Applications

- Find the two points in P that are farthest away

Applications

- Partitioning problems
  - Does there exist a line separating two objects?

Applications

- Yes
- No
Find the two points in P that are farthest away.

Applications

Outline

- Properties of CH
- Algorithms for computing the CH (P)
  - Brute-force
  - Gift wrapping (or: Jarvis march)
  - Quickhull
  - Graham scan
  - Andrew’s monotone chain
  - Incremental
  - Divide-and-conquer
- Can we do better?
  - Lower bound

Convex Hull Properties

Convexity: algebraic view

- Segment pq = set of all points of the form c_1p + c_2q with c_1, c_2 in [0,1], c_1+c_2=1
- A convex combination of points p_1, p_2, ..., p_k is a point of the form
  c_1 p_1 + c_2 p_2 + ... + c_k p_k with c_i in [0,1], c_1+c_2+...+c_k=1
- Example: a triangle consists of all convex combinations of its 3 vertices
- With this notation, the convex hull CH(P) = all convex combinations of points in P

Extreme points

- A point p is extreme if there exists a line l through p, such that all the other points of P are on the same side of l (or on l)
Extreme points

- A point \( p \) is extreme if there exists a line \( l \) through \( p \), such that all the other points of \( P \) are on the same side of \( l \) (and not on \( l \)).

Claim: If a point is on the CH if and only if (iff) it is extreme.

CH Variants

- Several types of convex hull output are conceivable:
  - all points on the convex hull in arbitrary order
  - all points on the convex hull in boundary order
  - only non-collinear points in arbitrary order
  - only non-collinear points in boundary order

- It may seem that computing in boundary order is harder
  - we’ll see that identifying the extreme points is \( \Omega(n \log n) \)
  - so sorting is not dominant
Interior points

- A point $p$ is not on the CH if and only if $p$ is contained in the interior of a triangle formed by three other points of $P$ (or in interior of a segment formed by two points).

Extreme edges

- An edge $(p_i, p_j)$ is extreme if all the other points of $P$ are on one side of it (or on).

Claim: A pair of points $(p_i, p_j)$ form an edge on the CH iff edge $(p_i, p_j)$ is extreme.
Extreme edges

- An edge \((p_i, p_j)\) is extreme if all the other points of \(P\) are on one side of it (or on)
- Claim: A pair of points \((p_i, p_j)\) form an edge on the CH iff edge \((p_i, p_j)\) is extreme.

Claim: A pair of points \((p_i, p_j)\) form an edge on the CH iff edge \((p_i, p_j)\) is extreme.

CH by finding extreme edges

Brute force

- Algorithm (input \(P\))
  - for all distinct pairs \((p_i, p_j)\)
    - check if edge \((p_i, p_j)\) is extreme
- Analysis?

Gift wrapping (1970)

- Observations
  - CH consists of extreme edges
  - each edge shares a vertex with next edge
- Idea: use an edge to find the next one
- How to find an extreme edge to start from?
- Given an extreme edge, how to find the next one?
Can you think of some points that are guaranteed to be in CH?

- **Claim**
  - point with minimum x-coordinate is extreme
  - point with maximum x-coordinate is extreme
  - point with minimum y-coordinate is extreme
  - point with maximum y-coordinate is extreme

- **Proof**
  - Start from point with smallest x-coordinate
    - if more than one, pick right most
  - Find first edge. HOW?
Gift wrapping (1970)

- Start from point with smallest x-coordinate
  - If more than one, pick rightmost

// Find first edge

- for each point q (q ≠ p)
  - Compute ccw-angle of q wrt p
  - Let p' = point with smallest angle
  - Output (p, p') as CH edge
  - What next?

Gift wrapping (1970)

- Start from point p with smallest x-coordinate
  - If more than one, pick rightmost

// Find first edge

- for each point q (q ≠ p)
  - Compute ccw-angle of q wrt p
  - Let p' = point with smallest angle
  - Output (p, p') as CH edge
  - Repeat from p'

Gift wrapping (1970)

- p₀ = point with smallest x-coordinate (if more than one, pick rightmost)
  - p = p₀
  - Repeat
    - for each point q (q ≠ p)
      - Compute ccw-angle of q wrt p
      - Let p' = point with smallest angle
      - Output (p, p') as CH edge
      - p = p'
    - Until p = p₀ (Until it discovers first point again)
Gift wrapping (1970)

• $p_0 =$ point with smallest $x$-coordinate (if more than one, pick right most)
• $p = p_0$
• repeat
  • for each point $q$ ($q \neq p$)
    • compute ccw-angle of $q$ wrt $p$
    • let $p' =$ point with smallest angle
  • output $(p, p')$ as CH edge
  • $p = p'$
• until $p = p_0$

Analysis

• for each vertex on the CH, it takes $O(n)$
• overall $O(nk)$, where $k$ is the size of the CH(P)

Summary

• Gift wrapping
  • Runs in $O(nk)$ time, where $k$ is the size of the CH(P)
  • this is efficient if $k$ is small
  • for $k = O(n)$, gift wrapping takes $O(n^2)$
  • Faster algorithms are known
  • Gift wrapping extends easily to 3D and for many years was the primary algorithm for 3D

Quickhull

upper hull
lower hull
left most point
right most point
Quickhull (late 1970s)

• Similar to Quicksort (in some way)

• Idea: start with 2 extreme points

CH consists of upper hull (CH of P_1) and lower hull (CH of P_2)
Quickhull (late 1970s)

• We’ll find the CH of $P_1$ and CH of $P_2$ separately

$P_1$

$P_2$

Quickhull (late 1970s)

• First let’s focus on $P_1$

Quickhull (late 1970s)

• For all points $p$ in $P_1$: compute $\text{dist}(p, ab)$

Quickhull (late 1970s)

• Find the point $c$ with largest distance (i.e. furthest away from $ab$)

Quickhull (late 1970s)

• Claim: $c$ must be an extreme point (and thus on the CH of $P_1$)
  • Proof: ?

Quickhull (late 1970s)

• Claim: $c$ must be an extreme point (and thus on the CH)
  • Proof: ?

let’s ignore collinear points for now
Quickhull (late 1970s)

- Discard all points inside triangle abc

67

Quickhull (late 1970s)

- Discard all points inside triangle abc

68

Quickhull (late 1970s)

- Recurse on the points left of ac and right of bc

69

Quickhull (late 1970s)

- Compute CH of P similarly

70

Quickhull (late 1970s)

- Quickhull (P)
  - find a, b
  - partition P into P1, P2
  - return Quickhull(a, b, P1) + Quickhull(b, a, P2)

71

Quickhull (late 1970s)

- Quickhull(a, b, P)
  - P is a set of points all on the left of ab
  - if P empty return emptyset
  - for each point p in P, compute its distance to ab
  - let c = point with max distance
  - let P1 = points to the left of ac
  - let P2 = points to the left of cb
  - return Quickhull(a, P1) + c + Quickhull(c, b, P2)

72
Quickhull (late 1970s)

- Analysis:
  - Write a recurrence relation for its running time
  - What is the best case running time? Show an example
  - What is the worst case running time? Show an example

- More exercises:
  - Argue that Quickhull's average complexity is $O(n)$ on points that are uniformly distributed.

Graham scan

- In late 60s an application at Bell Labs required the hull of 10,000 points, for which a quadratic algorithm was too slow
- Graham developed his simple algorithm which runs in $O(n \log n)$
  - one sort plus a linear pass

points on the boundary are in radial order around p

Graham scan (late 1960s)

- Idea: start from a point p interior to the hull

Graham scan (late 1960s)

64
Graham scan (late 1960s)

- Idea: start from a point p interior to the hull
  order all points by their ccw angle wrt p

Graham scan (late 1960s)

- Idea: start from a point p interior to the hull
  order all points by their ccw angle wrt p

Graham scan (late 1960s)

- Idea: start from a point p interior to the hull
  order all points by their ccw angle wrt p

79

80

81

Graham scan (late 1960s)

- Idea: traverse the points in this order a, b, c, d, e, f, g,...

82

83

84
Graham scan (late 1960s)

Now we read point c: what do we do with it?

\[ S = (b, a) \]

**Invariant:**
we maintain \( S \) as the CH of the points traversed so far

Is \( (c, b, a) \) convex?

**YES!**

85

Graham scan (late 1960s)

Now we read point c: what do we do with it?

\[ S = (c, b, a) \]

Is \( (c, b, a) \) convex?

**YES!**

86

Graham scan (late 1960s)

Now we read point c: what do we do with it?

\[ S = (c, b, a) \]

Is \( (c, b, a) \) convex?

**YES!**

87

Graham scan (late 1960s)

Now we read point c: if \( (c+S) \) stays convex: add c to S

\[ S = (c, b, a) \]

Is \( (c, b, a) \) convex?

**YES!**

88

Graham scan (late 1960s)

Now we read point d:

\[ S = (c, b, a) \]

Is \( c \) left of \( ab? \)

89

Graham scan (late 1960s)

Now we read point d:

\[ S = (c, b, a) \]

89

Graham scan (late 1960s)

Now we read point c:

\[ S = (c, b, a) \]
Graham scan (late 1960s)

Now we read point d: is d left of bc? NO

\[ S = (c, b, a) \]

\[ \text{Invariant: } \text{we maintain } S \text{ as the CH of the points traversed so far} \]

Graham scan (late 1960s)

Now we read point d: is d left of bc? NO

(p can't add d, because (d,c,b,a) not convex)

\[ S = (d, c, b, a) \]

\[ \text{Invariant: } \text{we maintain } S \text{ as the CH of the points traversed so far} \]

Graham scan (late 1960s)

Now we read point d: is d left of bc? NO

Now we read point d: is d left of ab? YES \implies insert d in S

\[ S = (d, b, a) \]

\[ \text{Invariant: } \text{we maintain } S \text{ as the CH of the points traversed so far} \]
In general...

We read next point q:

• let $b = \text{head}(S)$, $a = \text{next}(b)$
• if $q$ is left of $ab$: add $q$ to $S$

$S = (b, a, \ldots)$

Graham scan (late 1960s)

In general...

We read next point q:

• let $b = \text{head}(S)$, $a = \text{next}(b)$
• if $q$ is right of $ab$: pop $b$, repeat until $q$ is left of $ab$, then add $q$ to $S$

$S = (q, b, a, \ldots)$

Graham scan (late 1960s)

• Find interior point $p$
• Sort all other points ccw around $p_0$, and call them $p_1$, $p_2$, $p_3$, …, $p_n$ in this order
• Initialize stack $S = (p_0, p_1)$
• for i=3 to n-1 do
  • if $p_i$ is left of (second(S), first(S)):
    • push $p_i$ on S
  • else
    • do
    • pop S
    • while $p_i$ is right of (second(S), first(S))
    • push $p_i$ on S

Graham scan: ANALYSIS

• Find interior point $p$
• Sort all other points ccw around $p_0$, ...
• Initialize stack $S = (p_0, p_1)$
• for i=3 to n-1 do
  • if $p_i$ is left of (second(S), first(S)):
    • push $p_i$ on S
  • else
    • do
    • pop S
    • while $p_i$ is right of (second(S), first(S))
    • push $p_i$ on S
Graham scan: ANALYSIS

1. Find interior point $p_0$
2. Sort all other points ccw around $p_0$
3. Initialize stack $S = (p_2, p_1)$
4. for $i = 3$ to $n-1$ do
   - if $p_i$ is left of (second(S), first(S))
     - push $p_i$ on $S$
   - else
     - do
     - pop $S$
     - while $p_i$ is right of (second(S), first(S))
     - push $p_i$ on $S$

$O(n)$ (we’ll think of it later)

1. $O(n)$ (we’ll think of it later)

103

Graham scan: ANALYSIS

1. Find interior point $p_0$
2. Sort all other points ccw around $p_0$
3. Initialize stack $S = (p_2, p_1)$
4. for $i = 3$ to $n-1$ do
   - if $p_i$ is left of (second(S), first(S))
     - push $p_i$ on $S$
   - else
     - do
     - pop $S$
     - while $p_i$ is right of (second(S), first(S))
     - push $p_i$ on $S$

$O(n \lg n)$

104

Graham scan: Details

1. How to find an interior point?
   - A simplification is to pick $p_0$ as the lowest point

105

Graham scan: Details

1. How to find an interior point?

106

How long does this take?

107

How long does this take?

108
Graham scan: Details

- How to find an interior point?
- A simplification is to pick \( p_0 \) as the lowest point
- Initialize stack \( S = (p_1, p_0) \)
  (both are on \( CH \) and \( S \) will always contain at least 2 points)

Graham scan: Details

- Handling collinear-ities

Speeding up Graham scan

What happens when you run on this input?

How can you fix it?
Speeding up Graham scan

![Diagram of points]