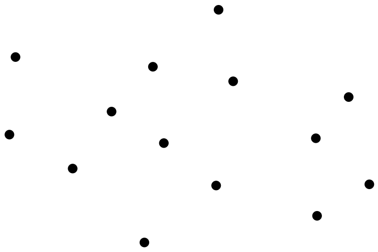
Convex Hulls: An Application of Stacks and Deques

Pat Morin
COMP2402/2002

Carleton University

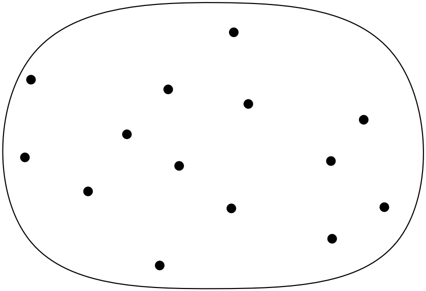
Convex Hulls

- Let $P = \{p_0, \ldots, p_{n-1}\}$ be a set of points in the plane
- The *convex hull* of S
  - the smallest convex set that contains S.



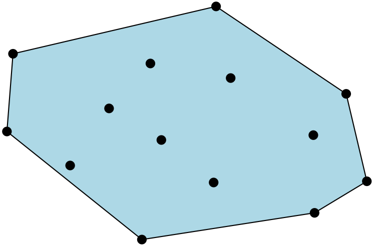
Convex Hulls

- Let $P = \{p_0, \ldots, p_{n-1}\}$ be a set of points in the plane
- The *convex hull* of S
  - the smallest convex set that contains S.
  - stretch a rubber band around S



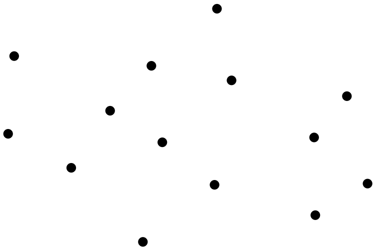
Convex Hulls

- Let $P = \{p_0, \ldots, p_{n-1}\}$ be a set of points in the plane
- The *convex hull* of S
  - the smallest convex set that contains S.
  - stretch a rubber band around S and let it go



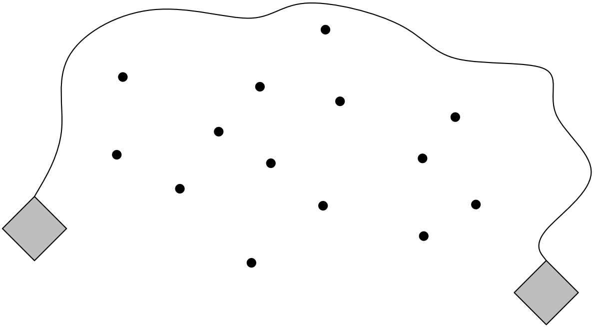
Upper and Lower Hulls

- The *upper hull* of S is a bit simpler



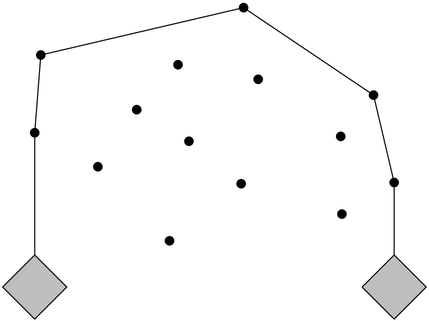
Upper and Lower Hulls

- The *upper hull* of S is a bit simpler
  - get a rope with weights on both ends



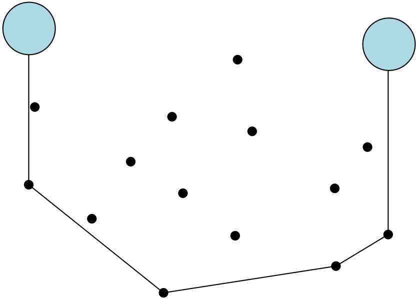
Upper and Lower Hulls

- The *upper hull* of S is a bit simpler
  - get a rope with weights on both ends and throw it over S



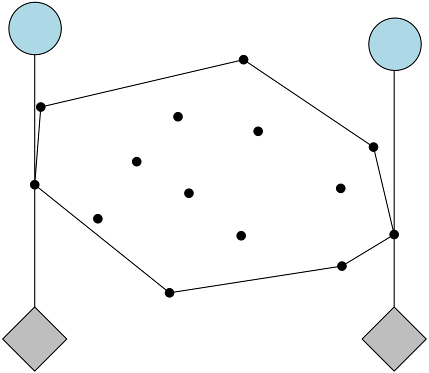
Upper and Lower Hulls

- The *upper hull* of S is a bit simpler
  - get a rope with weights on both ends and throw it over S
- For the *lower hull*, use helium-baloons



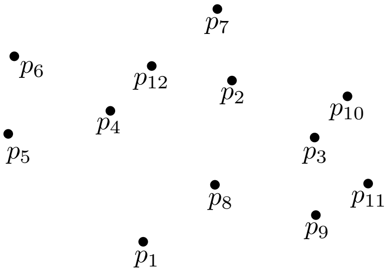
Upper and Lower Hulls

- The *upper hull* of S is a bit simpler
  - get a rope with weights on both ends and throw it over S
- For the *lower hull*, use helium-baloons
- Convex hull $=$ upper hull $+$ lower hull



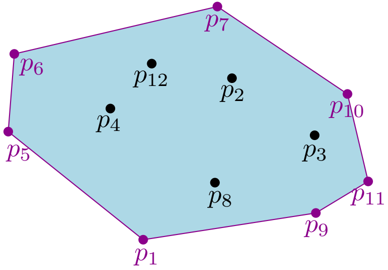
Summary so far

- Input: A set P of n points (unordered)



Summary so far

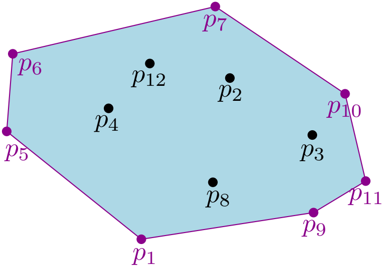
- Input: A set P of n points (unordered)
- Output: A list of the points of P on the convex hull — in clockwise order



$\langle p_5, p_6, p_7, p_{10}, p_{11}, p_9, p_1 \rangle$

Summary so far

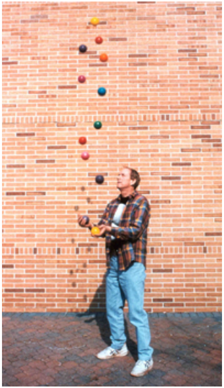
- Input: A set P of n points (unordered)
- Output: A list of the points of P on the convex hull — in clockwise order
- First compute the upper hull, then the lower hull



$\langle p_5, p_6, p_7, p_{10}, p_{11}, p_9, p_1 \rangle$

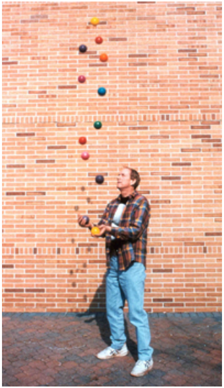
Graham's Scan

- Invented by Ron Graham in 1973



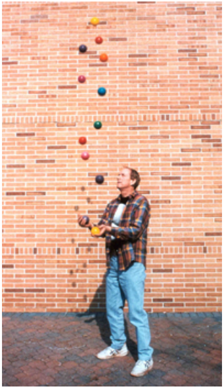
Graham's Scan

- Invented by Ron Graham in 1973
- Sorts the points by x-coordinate

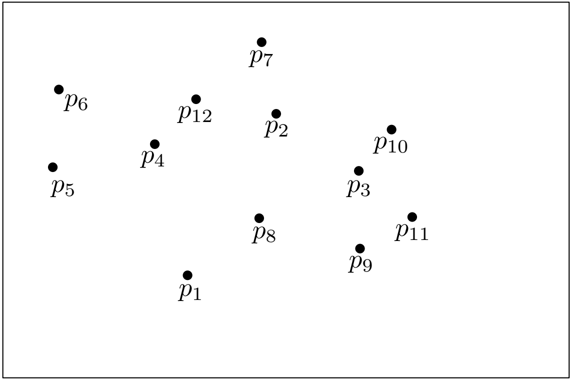


Graham's Scan

- Invented by Ron Graham in 1973
- Sorts the points by x-coordinate
- Constructs the upper hull incrementally using a stack

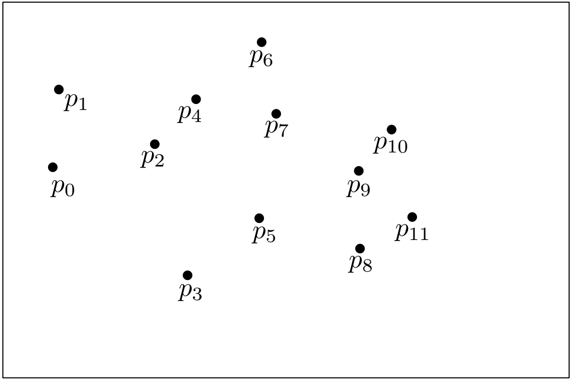


Graham's Scan



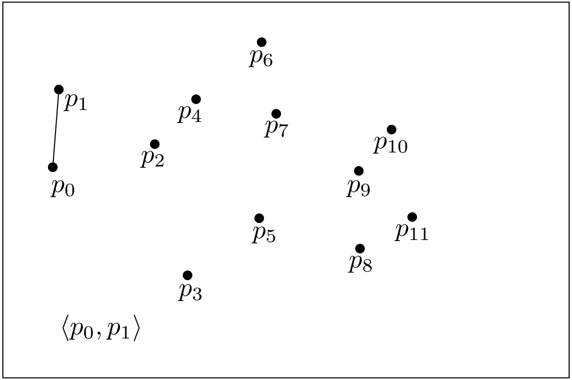
Graham's Scan

1. Sort the points by x-coordinate



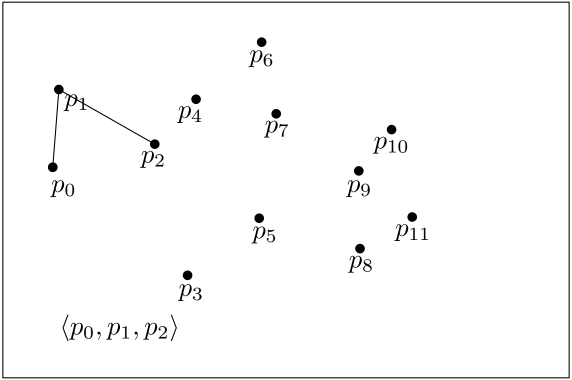
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$



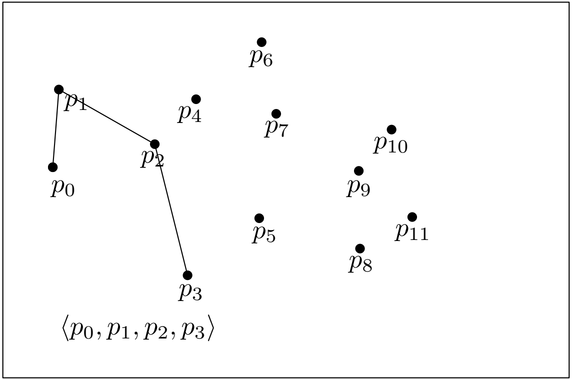
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



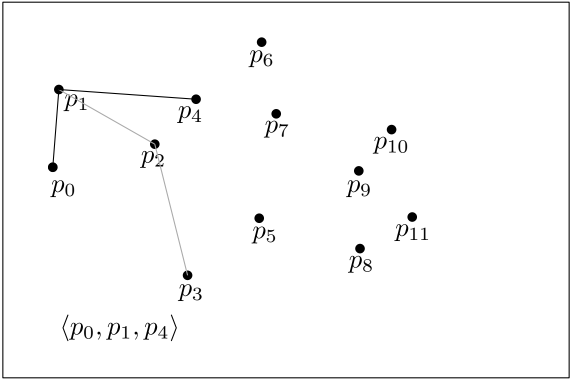
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



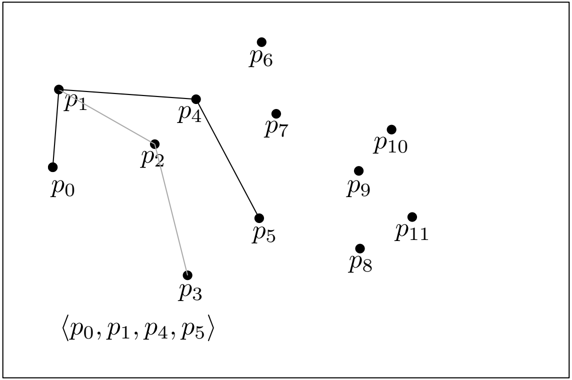
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



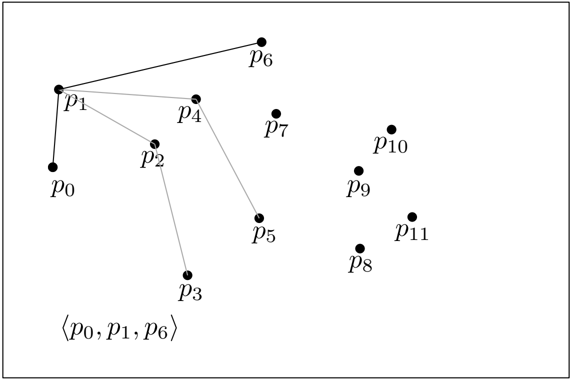
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



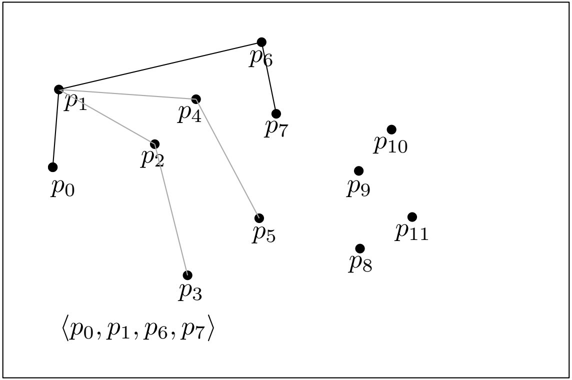
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



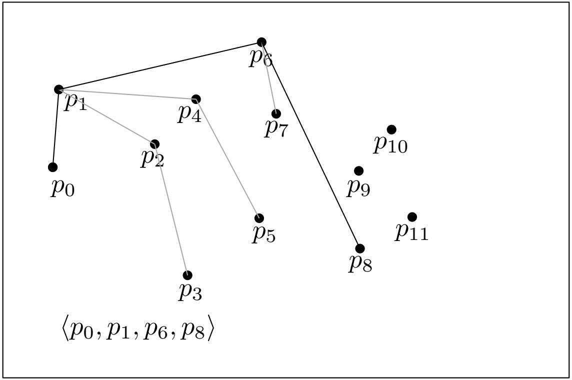
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



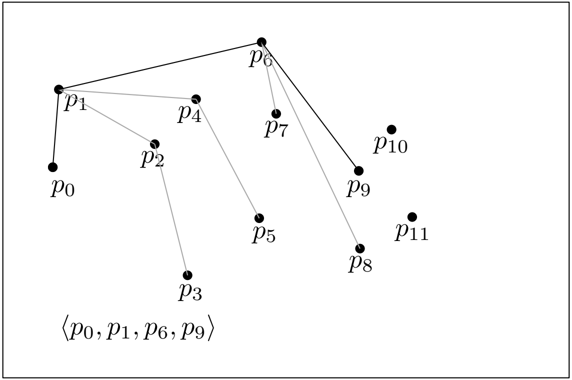
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



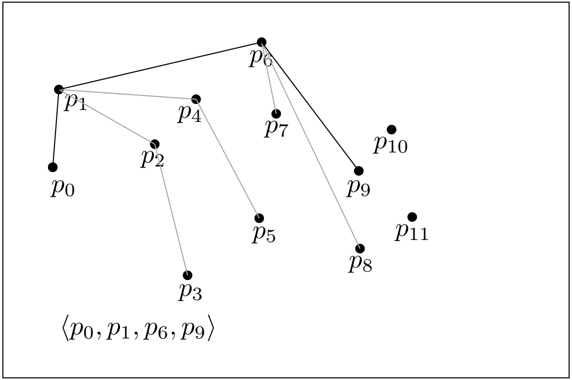
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



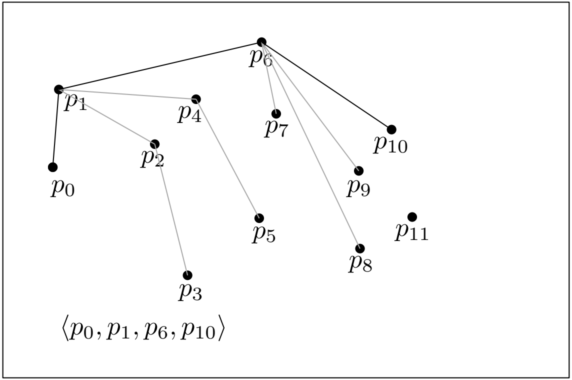
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1\rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



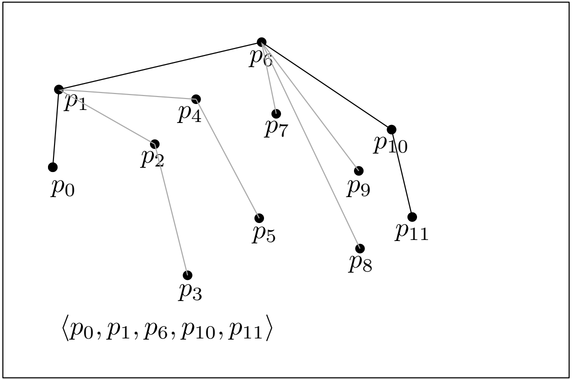
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



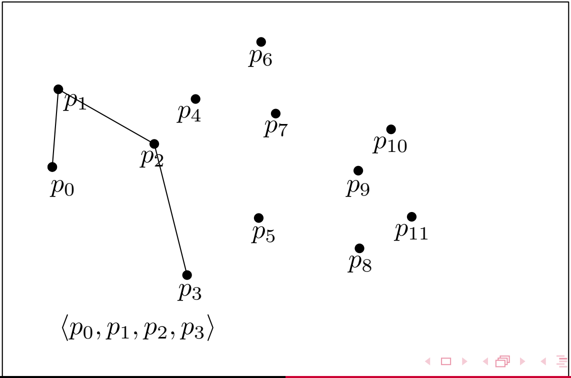
Graham's Scan

1. Sort the points by x-coordinate
2. Create a stack s containing $\langle p_0, p_1 \rangle$
3. For $i = 3$ to n do
   - add p_i to the convex hull of $p_1, \ldots, p_{i-1}$



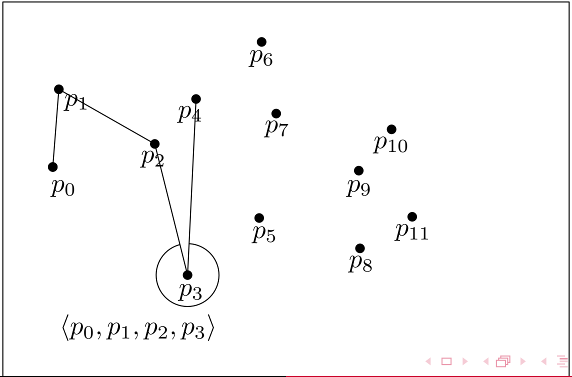
Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



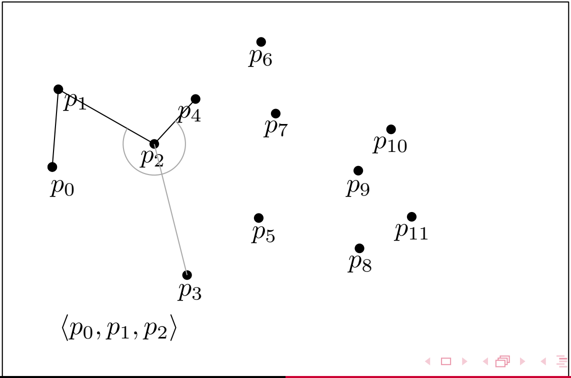
Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



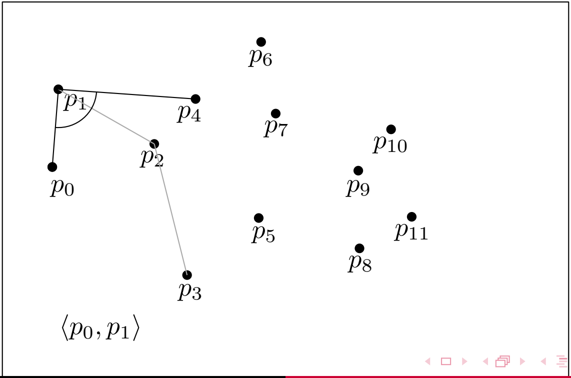
Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



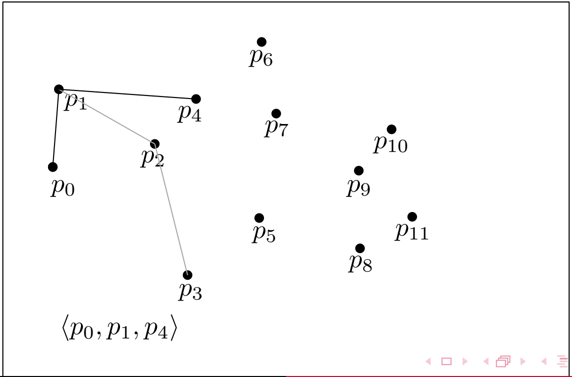
Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



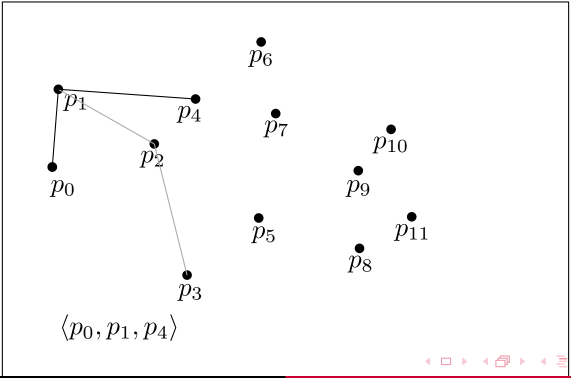
Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



Adding p_i

- Suppose `s` contains the upper hull of $p_0, \ldots, p_{i-1}$
- We want to compute upper hull of $p_0, \ldots, p_i$
  1. while `s.get(s.size()-2)`, `s.get(s.size()-1)`, and p_i form a left turn
     - `s.remove(s.size()-1)` (pop)
  2. `s.add(`p_i`)`



Graham's Scan – in Java

```
public static List<Point2D>
      grahamScan(List<Point2D> p) {
   Collections.sort(p, new XComparator());
   List<Point2D> s = new ArrayList<Point2D>();
   s.add(p.get(0)); s.add(p.get(1));
   for (int i = 2; i < p.size(); i++) {
      Point2D pi = p.get(i);
      while (s.size() >= 2
            && leftTurn(s.get(s.size()-2),
                        s.get(s.size()-1),
                        pi)) {
         s.remove(s.size());    // pop
      }
      s.add(pi);
   }
   return s;
}
```

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values
  - each iteration does some `pop`/`remove` operations
    - how many?

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values
  - each iteration does some `pop`/`remove` operations
    - how many?
  - each iteration does 1 `push`/`add` operation
    - $O(1)$ per iteration $= O(n)$ overall

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values
  - each iteration does some `pop`/`remove` operations
    - how many?
  - each iteration does 1 `push`/`add` operation
    - $O(1)$ per iteration $= O(n)$ overall
- Total: $O(n \log n) + O(n) + O(\text{num. pop operations})$

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values
  - each iteration does some `pop`/`remove` operations
    - how many?
  - each iteration does 1 `push`/`add` operation
    - $O(1)$ per iteration $= O(n)$ overall
- Total: $O(n \log n) + O(n) \quad + O(n)$

Analysis of Graham's Scan

- Graham's Scan first sorts the data
  - can be done $O(n \log n)$ time (see COMP3804)
- Creates a stack and pushes two values
  - takes $O(1)$ time
- A `for` loop over $n - 2 = O(n)$ values
  - each iteration does some `pop`/`remove` operations
    - how many?
  - each iteration does 1 `push`/`add` operation
    - $O(1)$ per iteration $= O(n)$ overall
- Total: $O(n \log n) + O(n) \quad + O(n) \; = O(n \log n)$

Summary

- **Theorem:** Given a collection P of n points in the plane, Graham's Scan can compute their upper hull in $O(n \log n)$ time.

Summary

- **Theorem:** Given a collection P of n points in the plane, Graham's Scan can compute their upper hull in $O(n \log n)$ time.
- With two passes, we can compute the upper hull and lower hull and attach them together to get the convex hull:

Summary

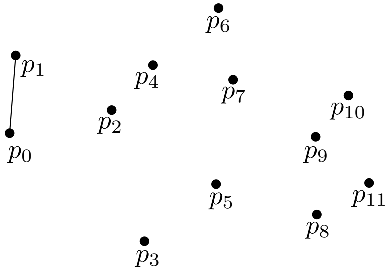
- **Theorem:** Given a collection P of n points in the plane, Graham's Scan can compute their upper hull in $O(n \log n)$ time.
- With two passes, we can compute the upper hull and lower hull and attach them together to get the convex hull:
- **Theorem:** Given a collection P of n points in the plane, two applications of Graham's Scan can compute their convex hull in $O(n \log n)$ time.

Summary

- **Theorem:** Given a collection P of n points in the plane, Graham's Scan can compute their upper hull in $O(n \log n)$ time.
- With two passes, we can compute the upper hull and lower hull and attach them together to get the convex hull:
- **Theorem:** Given a collection P of n points in the plane, two applications of Graham's Scan can compute their convex hull in $O(n \log n)$ time.
- By using a Deque, we only need one pass

Graham's Scan with a deque

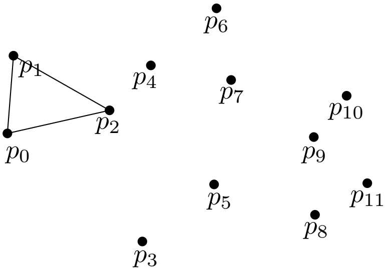
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_1, p_0, p_1 \rangle$

Graham's Scan with a deque

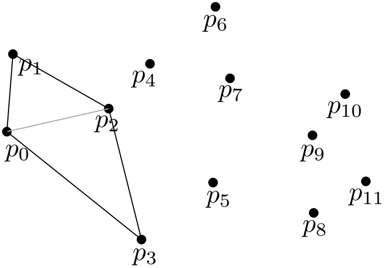
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_2, p_0, p_1, p_2 \rangle$

Graham's Scan with a deque

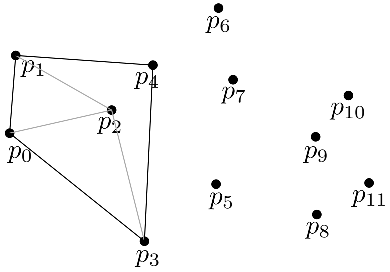
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_3, p_0, p_1, p_2, p_3 \rangle$

Graham's Scan with a deque

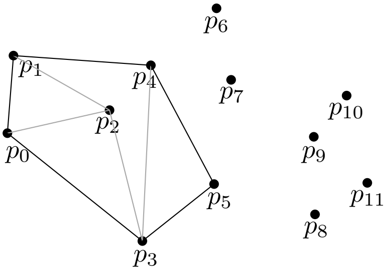
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_4, p_3, p_0, p_1, p_4 \rangle$

Graham's Scan with a deque

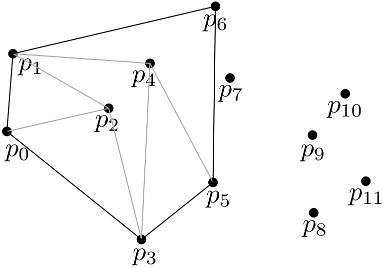
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_5, p_3, p_0, p_1, p_4, p_5 \rangle$

Graham's Scan with a deque

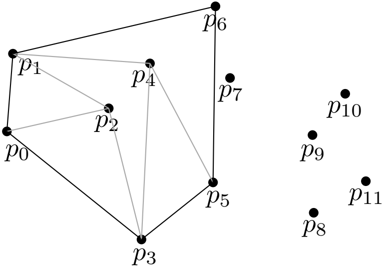
- Graham's Scan can compute the convex hull in one-pass using a deque



$\langle p_6, p_5, p_3, p_0, p_1, p_6 \rangle$

Graham's Scan with a deque

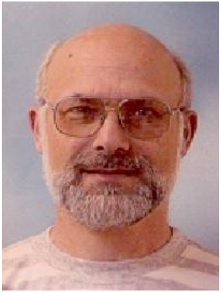
- Graham's Scan can compute the convex hull in one-pass using a deque

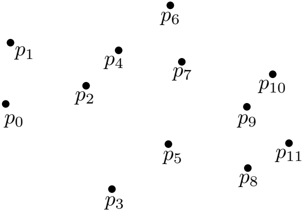


$\langle p_6, p_5, p_3, p_0, p_1, p_6 \rangle \ \ldots$

Melkman's Algorithm

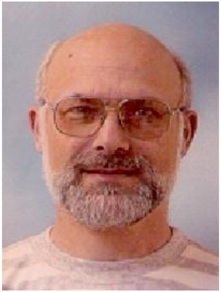
- Graham's Scan starts by sorting the the points by x-coordinate

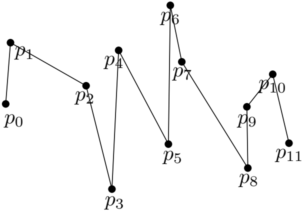




Melkman's Algorithm

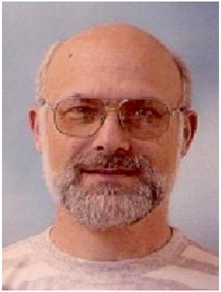
- Graham's Scan starts by sorting the the points by x-coordinate
  - This means that $p_0, \ldots, p_{n-1}$ becomes a non-self-intersecting path
  - If points are already sorted then Graham's Scan takes $O(n)$ time

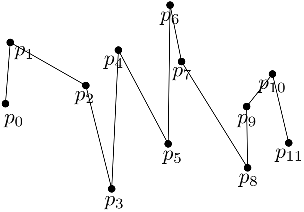




Melkman's Algorithm

- Graham's Scan starts by sorting the the points by x-coordinate
  - This means that $p_0, \ldots, p_{n-1}$ becomes a non-self-intersecting path
  - If points are already sorted then Graham's Scan takes $O(n)$ time
- Melkman's Algorithm:





Melkman's Algorithm

- Graham's Scan starts by sorting the the points by x-coordinate
  - This means that $p_0, \ldots, p_{n-1}$ becomes a non-self-intersecting path
  - If points are already sorted then Graham's Scan takes $O(n)$ time
- Melkman's Algorithm:
  - Works for any non-self-intersecting path $p_0, \ldots, p_{n-1}$