Algorithm Analysis and Design

Divide and Conquer strategy (Convex Hull Problem)

Lecture -17

Overview

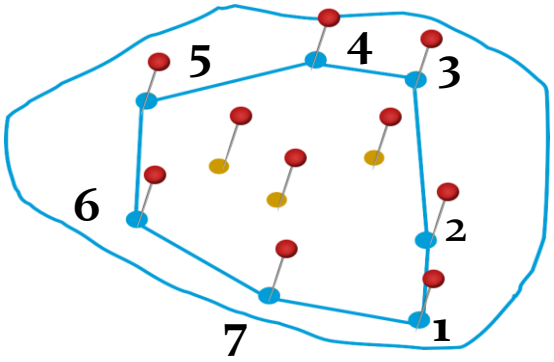
- *Learn the implementation techniques of "divide and conquer" in the context of the Convex Hull Problem with analysis.*

Convex Hull

- Given a set of pins on a pinboard, and a rubber band around them .
- How does the rubber band look when it snaps tight? Just imagine.

Convex Hull

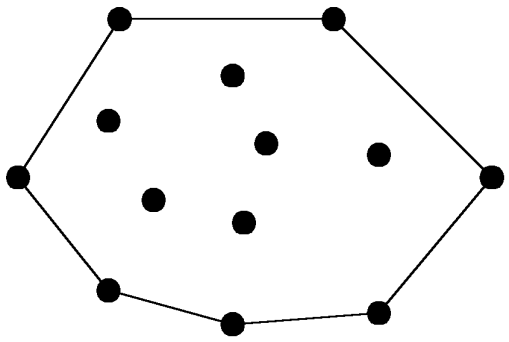
- Given a set of pins on a pinboard, and a rubber band around them .
- How does the rubber band look when it snaps tight? Just imagine.



- We represent the convex hull as the sequence of points on the convex hull polygon, in counter- clockwise order.

Convex Hull

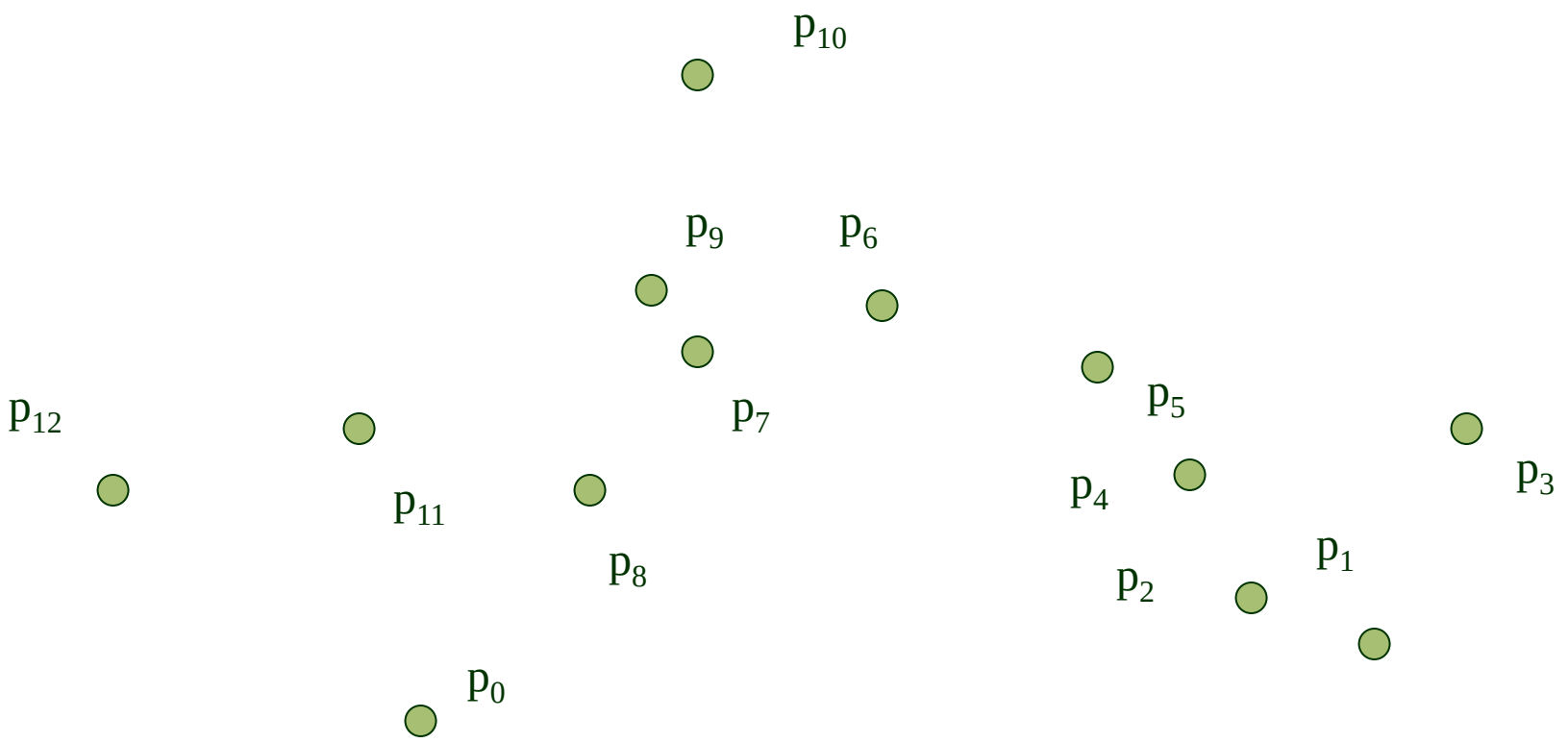
- Definition:
  - Informal: Convex hull of a set of points in plane is the shape taken by a rubber band stretched around the nails pounded into the plane at each point.
  - Formal: The convex hull of a set of planar points is the smallest convex polygon containing all of the points.



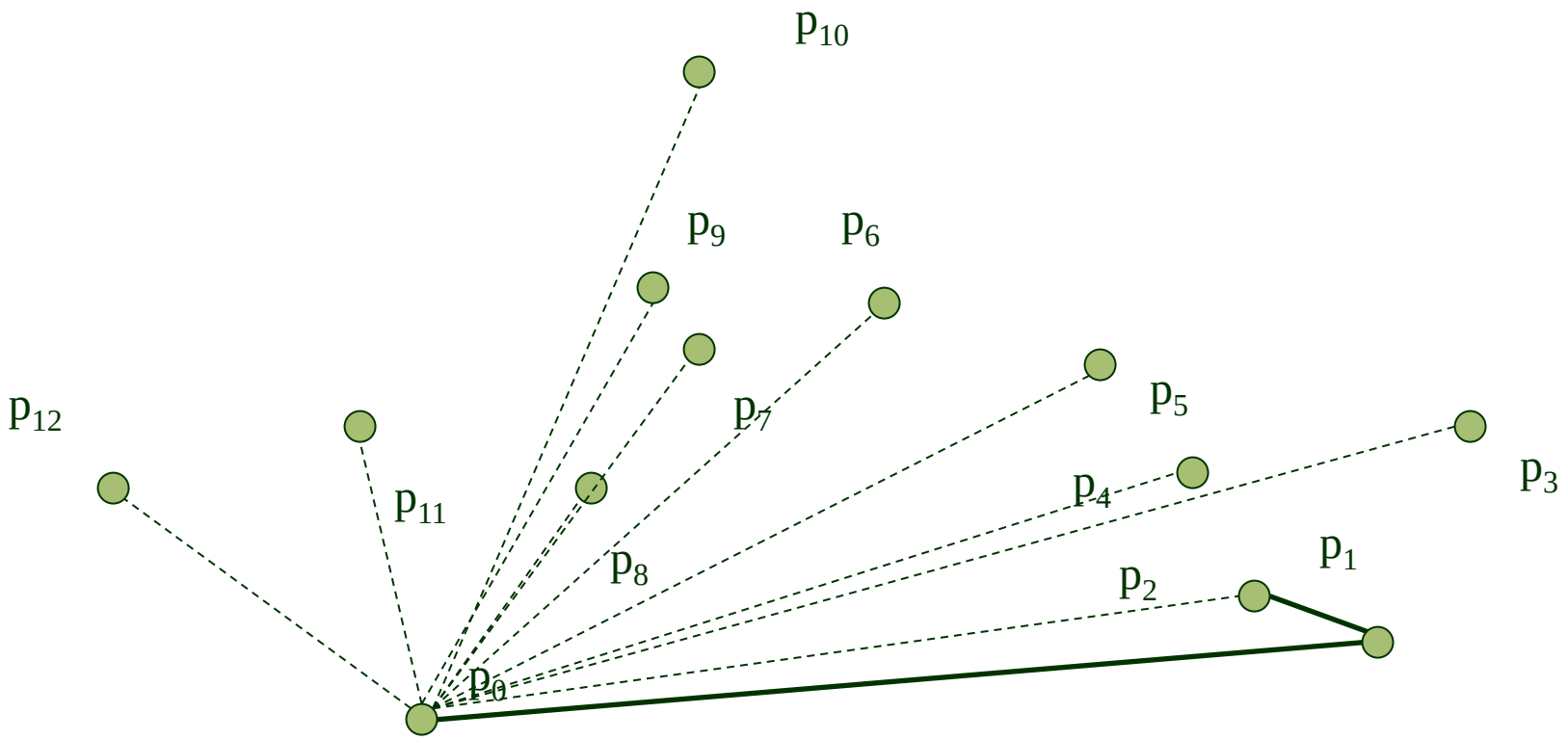
Graham Scan

- Concept:
  - Start at point guaranteed to be on the hull. (the point with the minimum y value)
  - Sort remaining points by polar angles of vertices relative to the first point.
  - Go through sorted points, keeping vertices of points that have left turns and dropping points that have right turns.

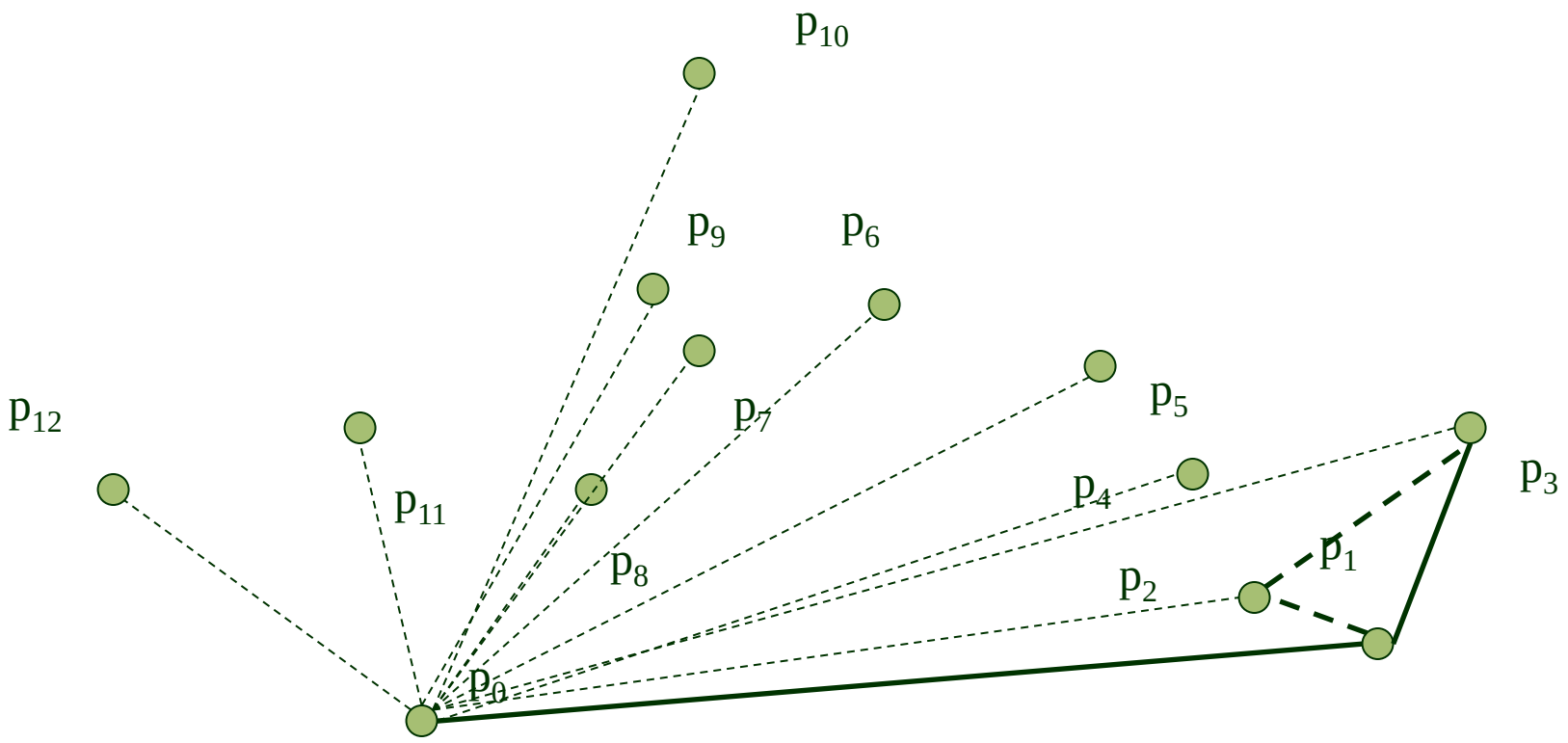
Graham Scan - Example



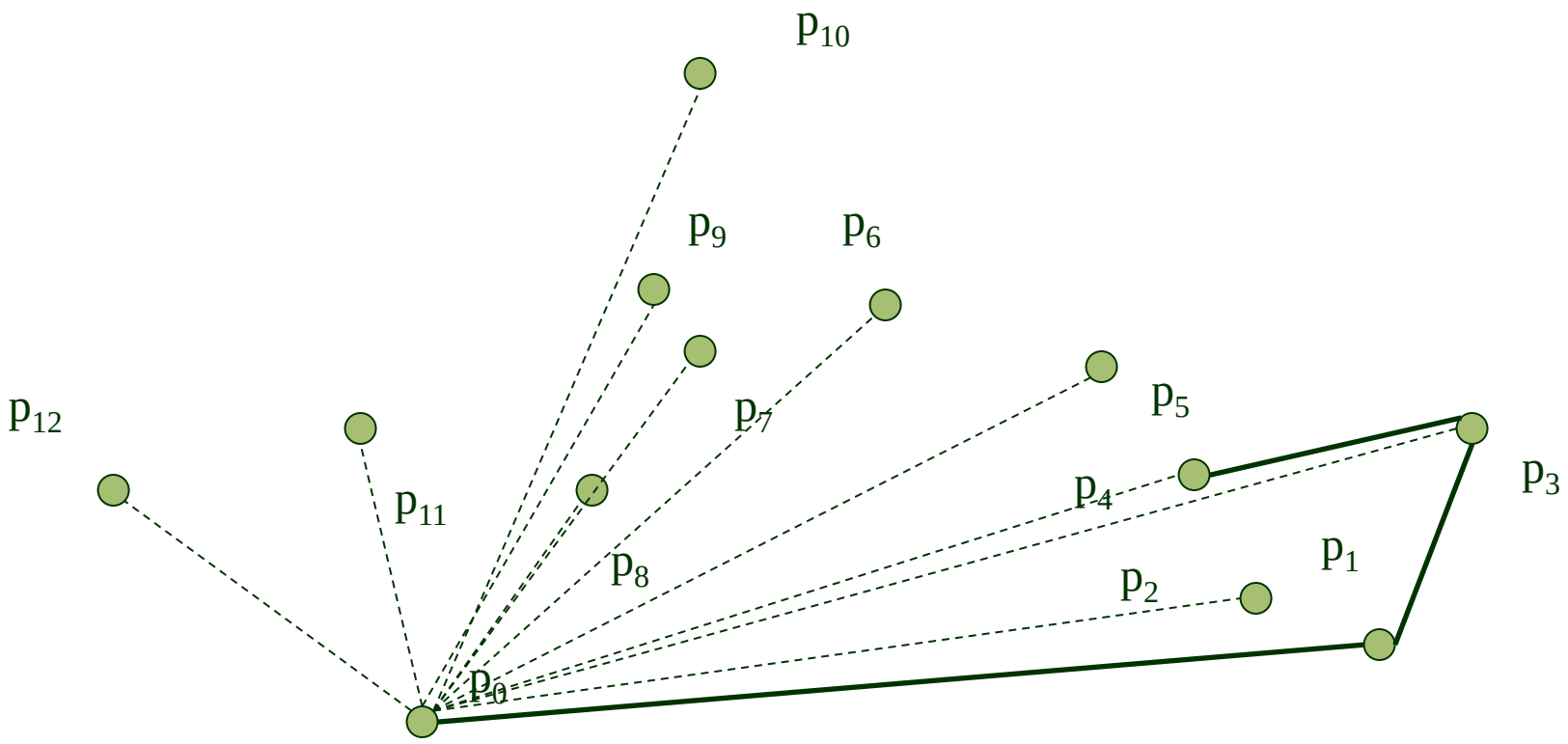
Graham Scan - Example



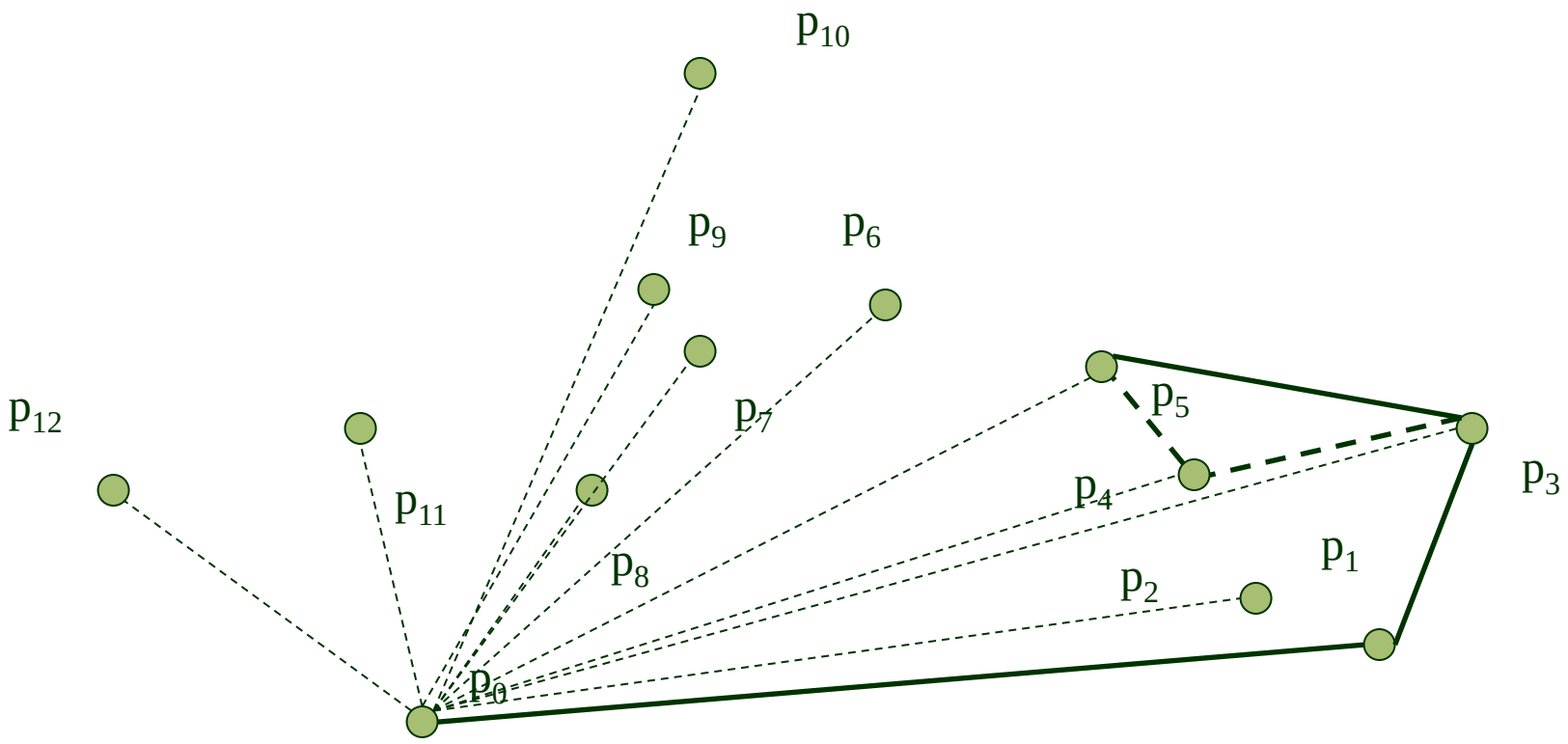
Graham Scan - Example



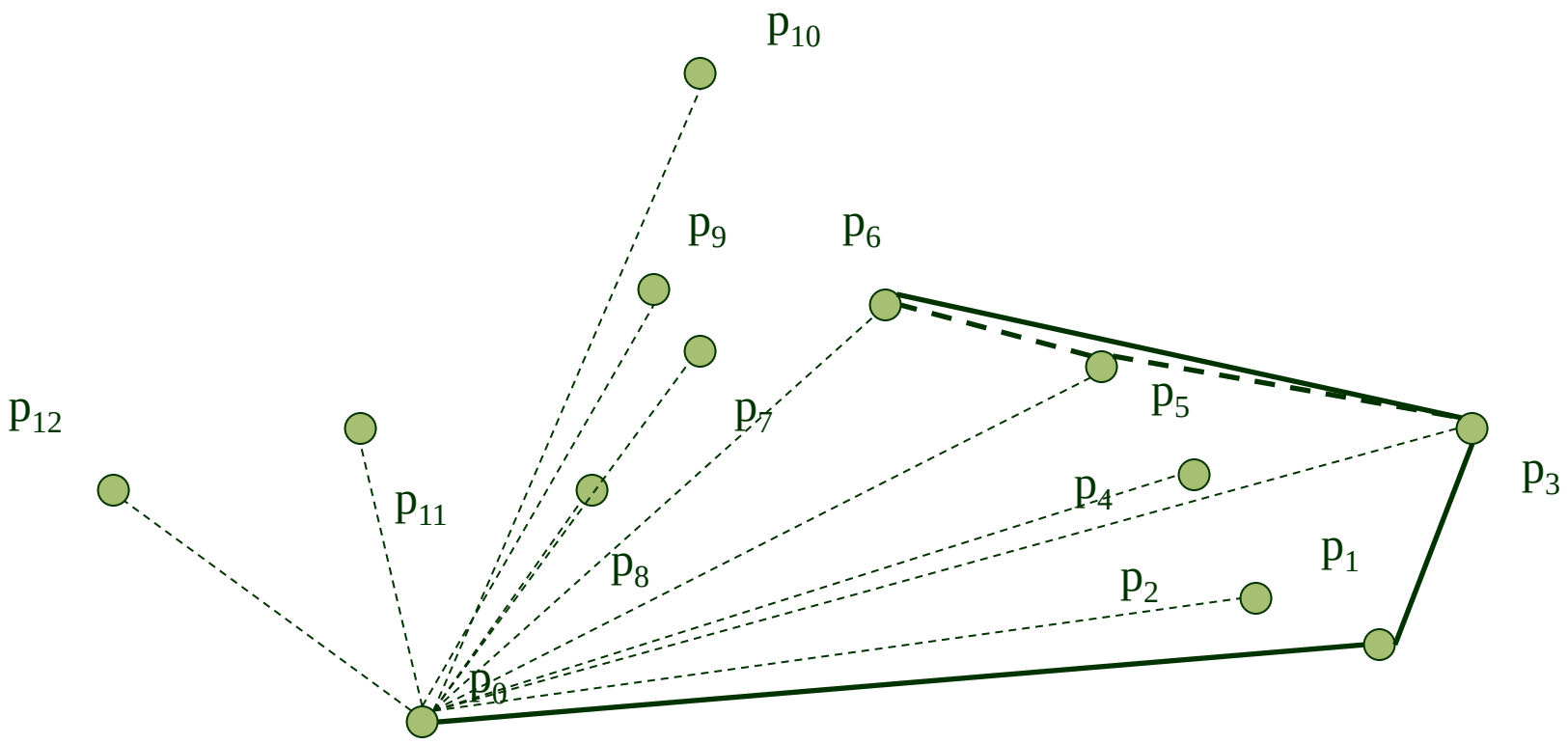
Graham Scan - Example



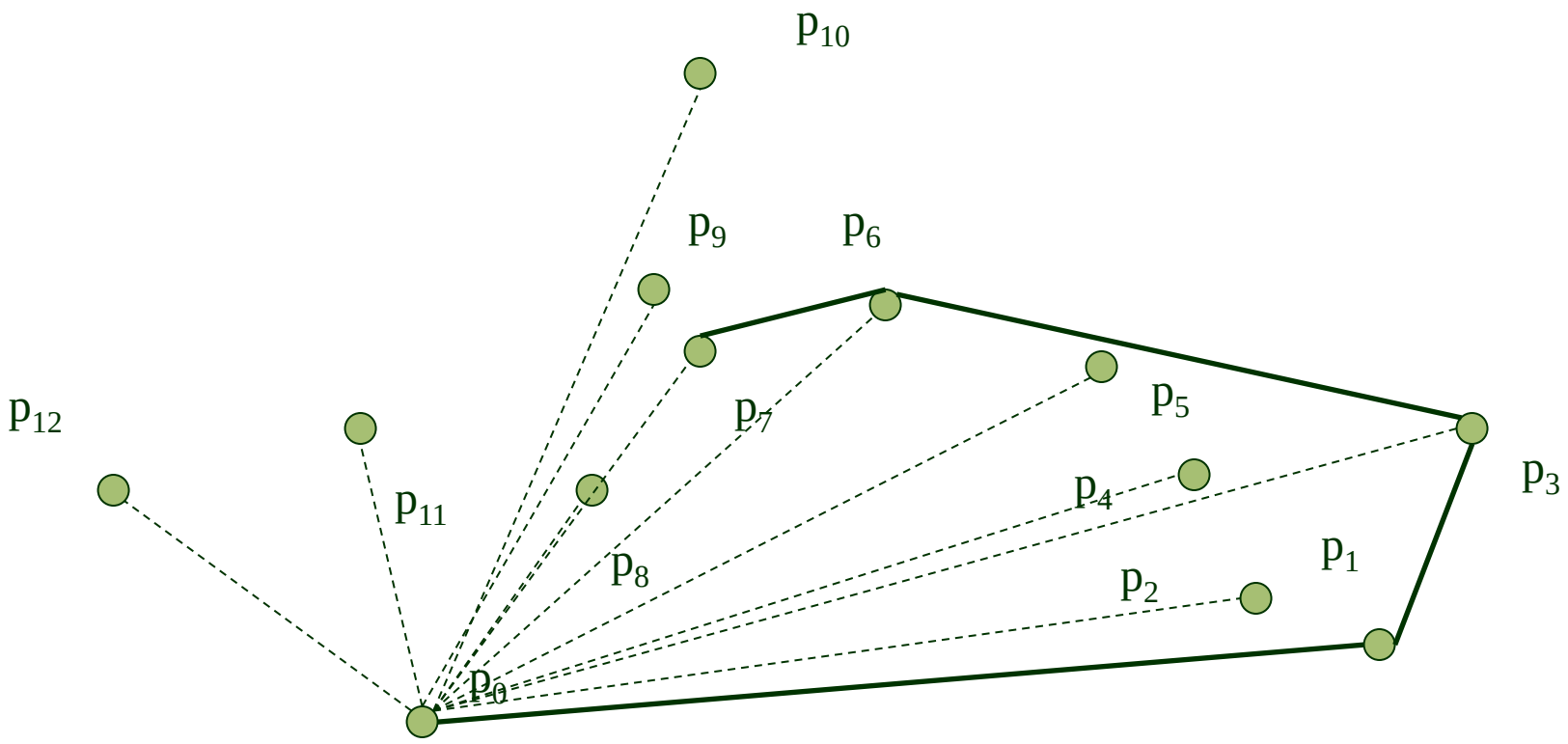
Graham Scan - Example



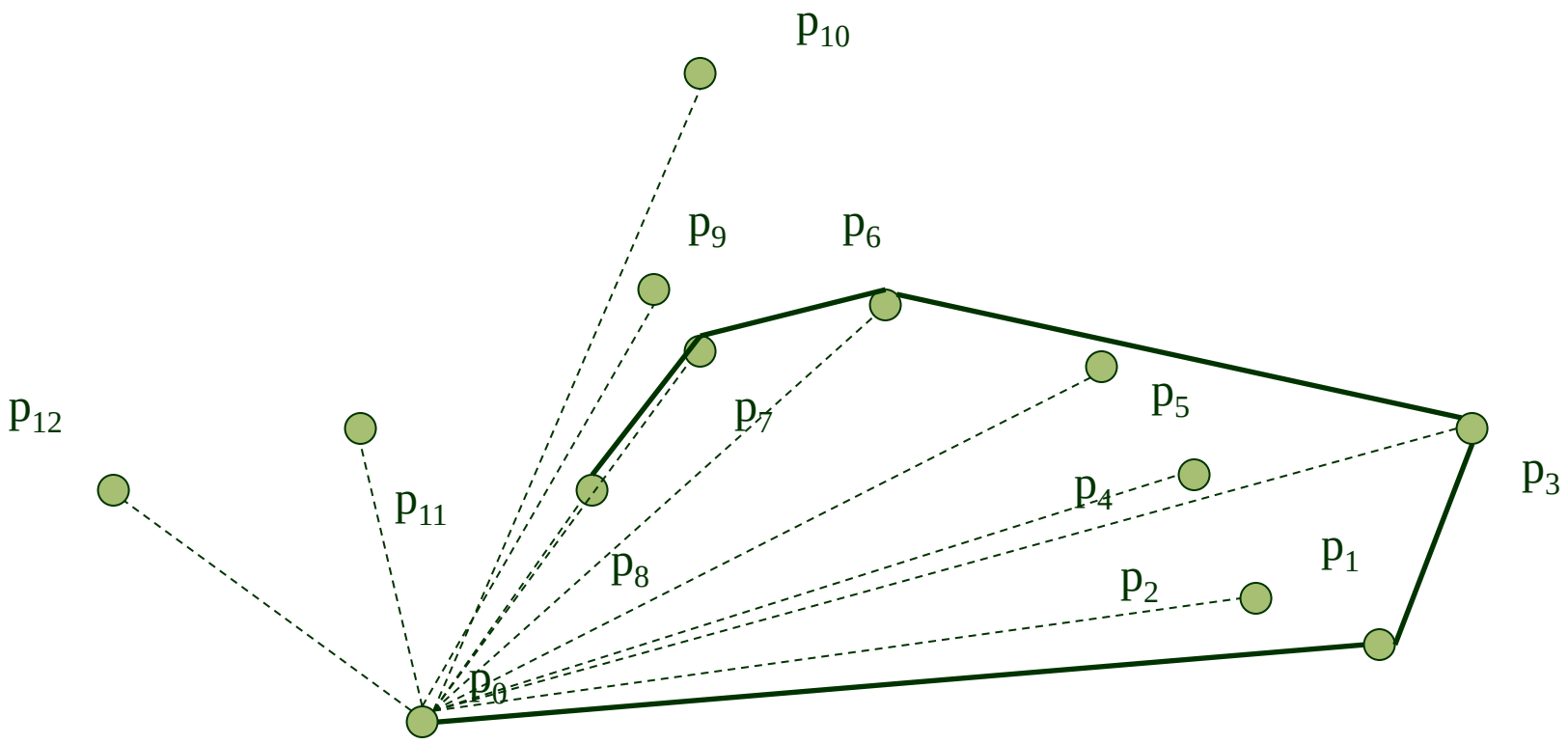
Graham Scan - Example



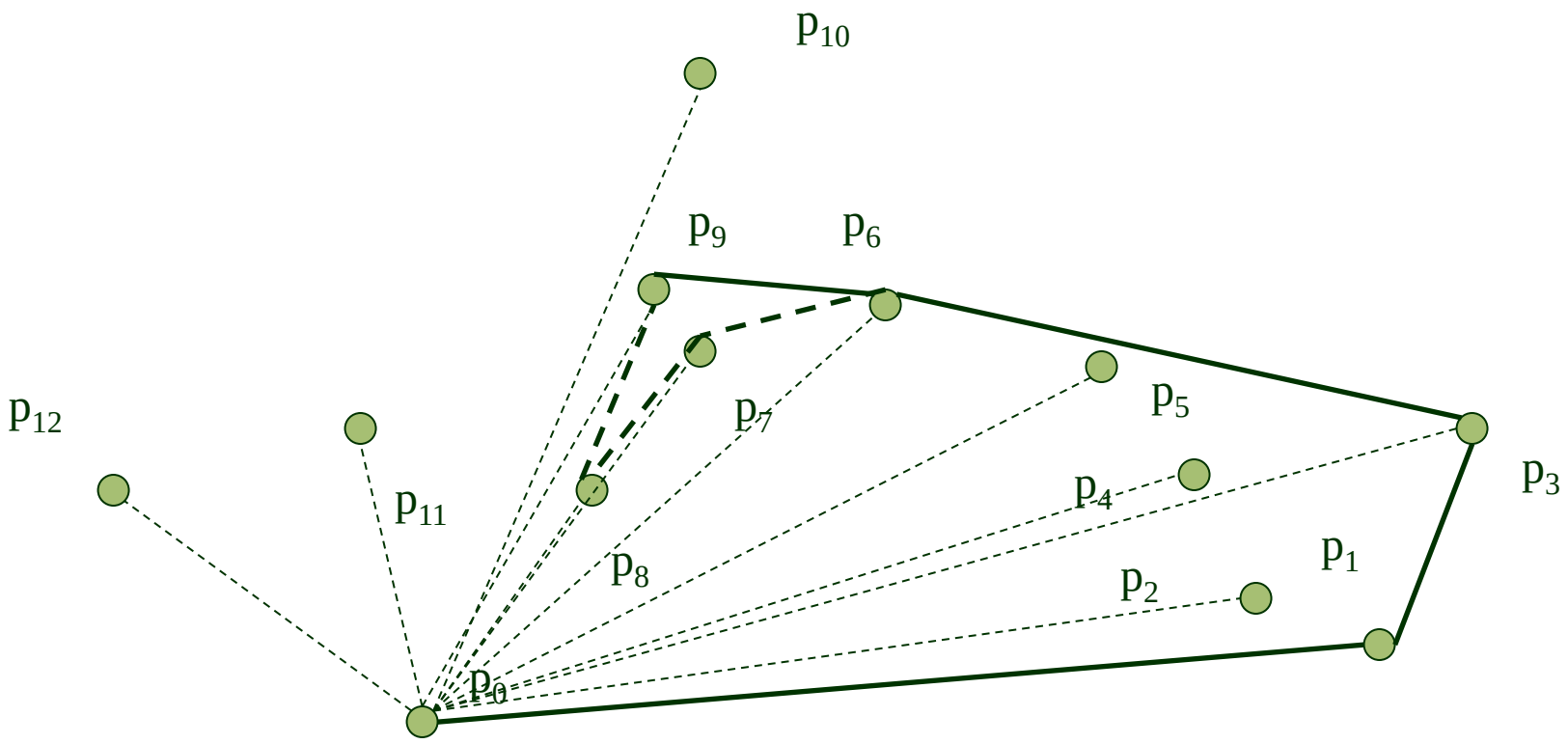
Graham Scan - Example



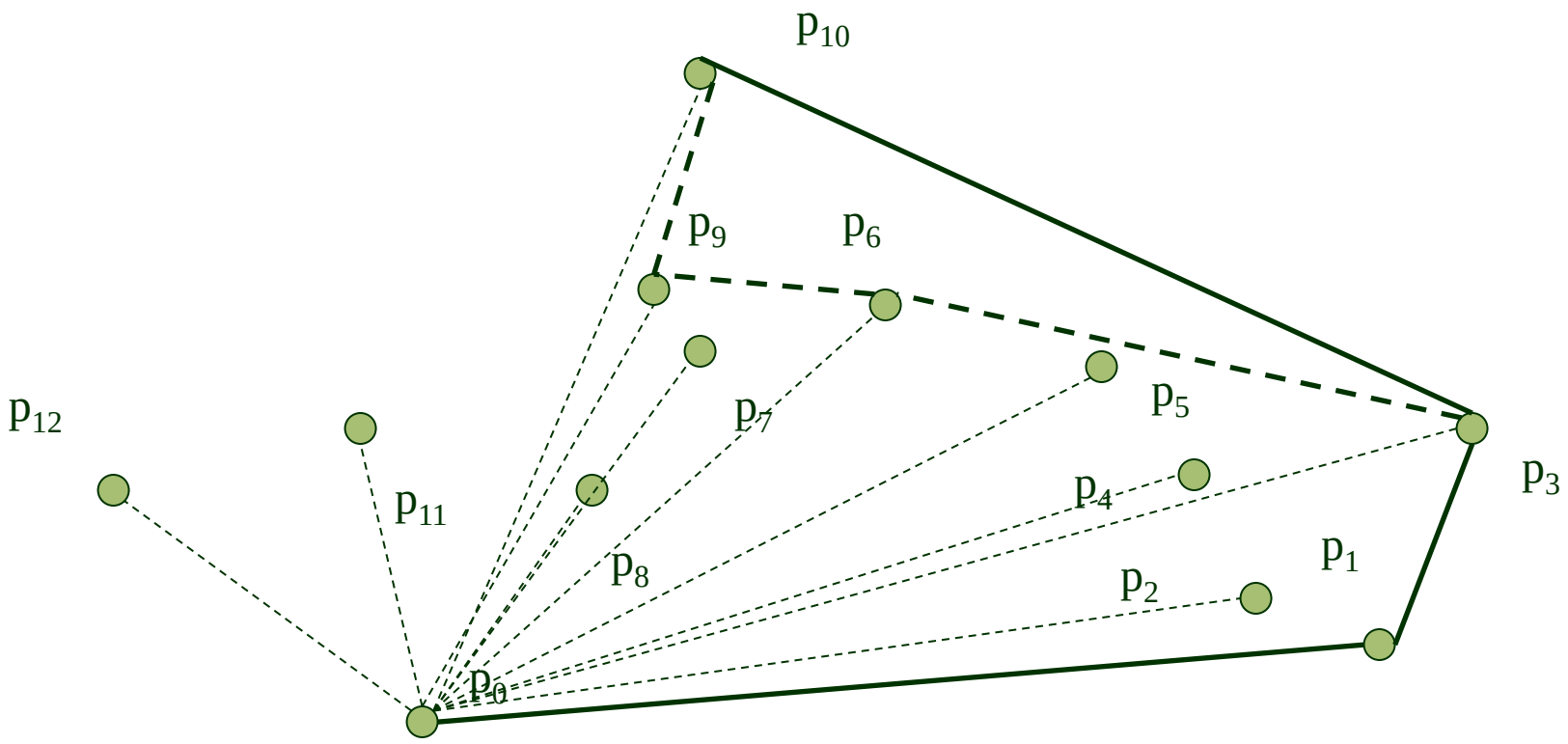
Graham Scan - Example



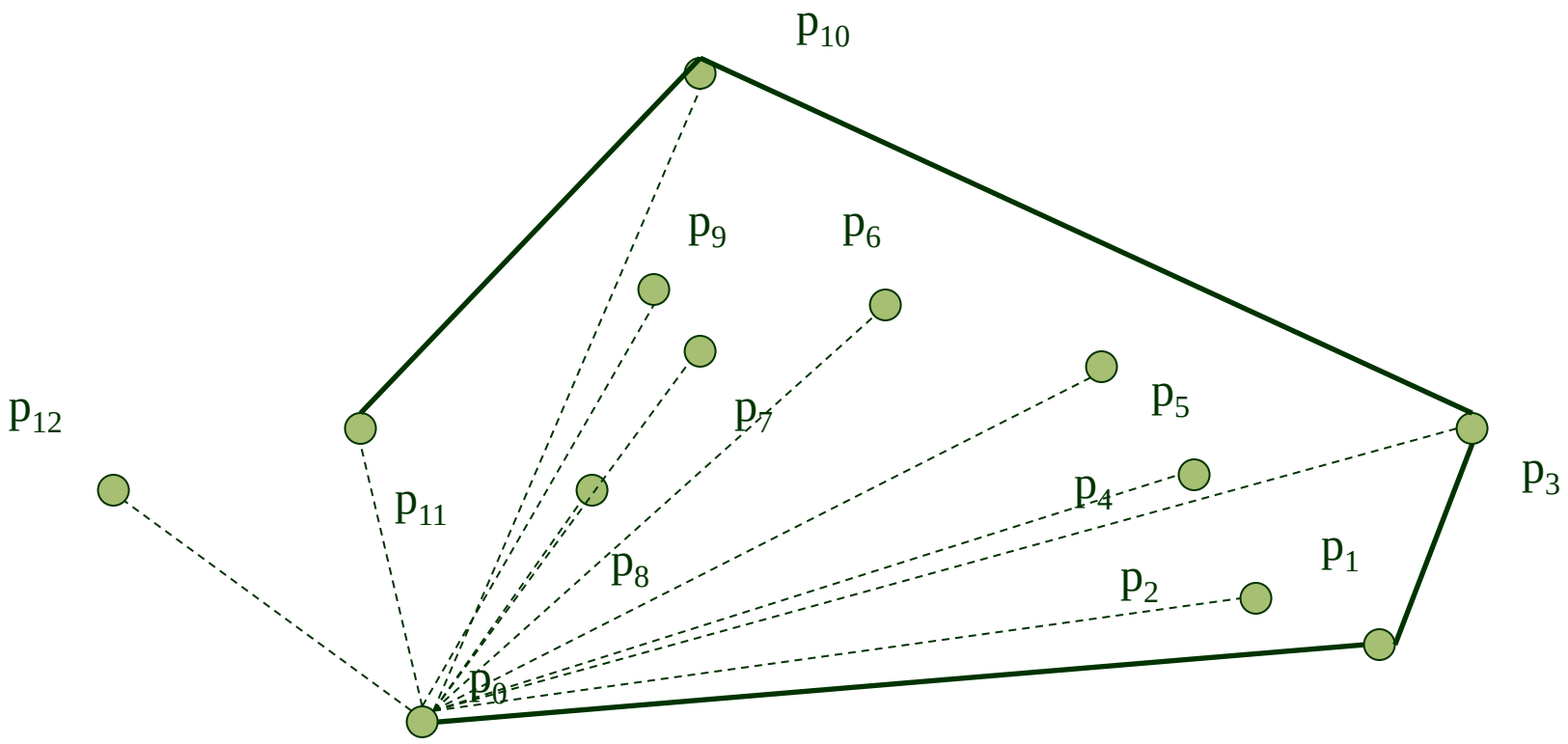
Graham Scan - Example



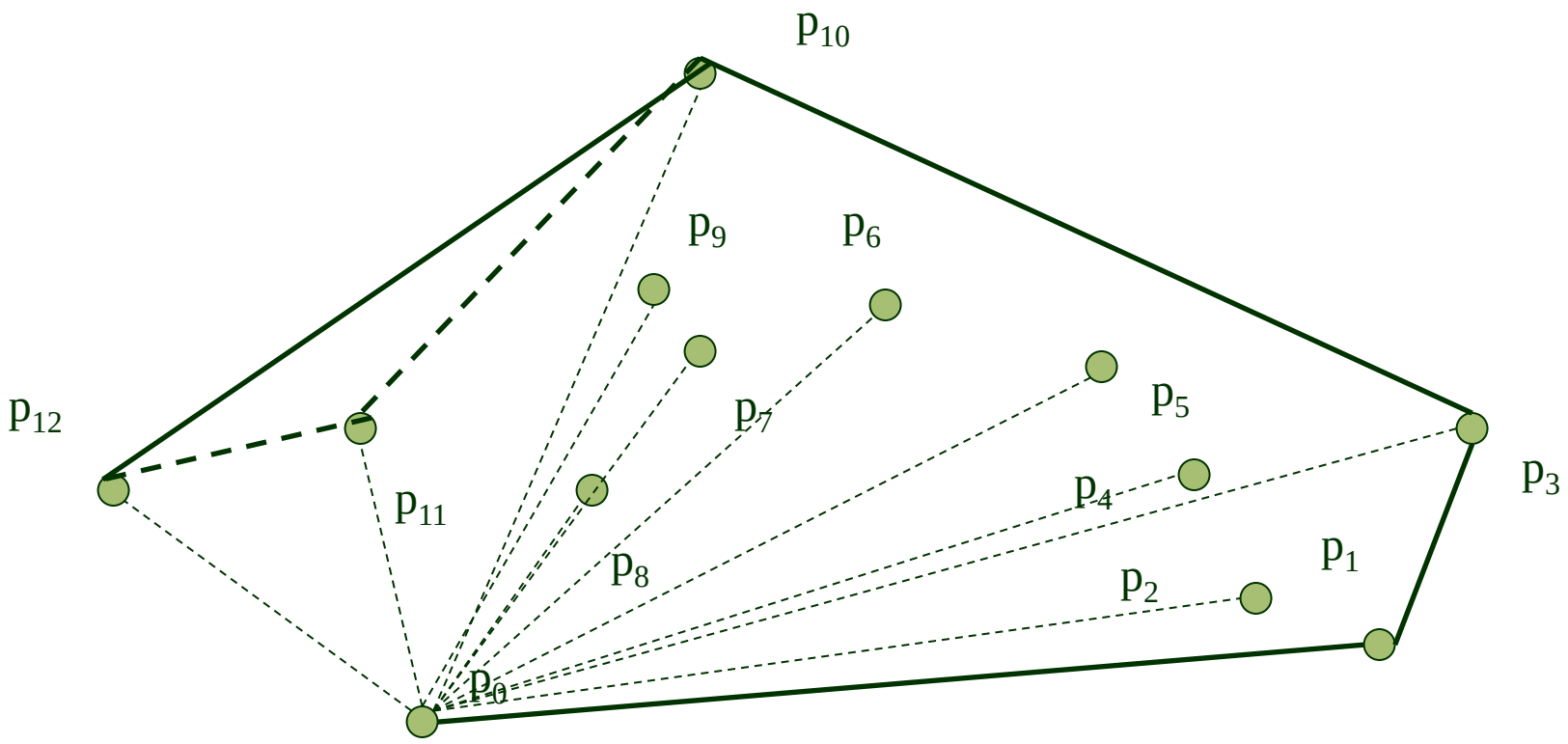
Graham Scan - Example



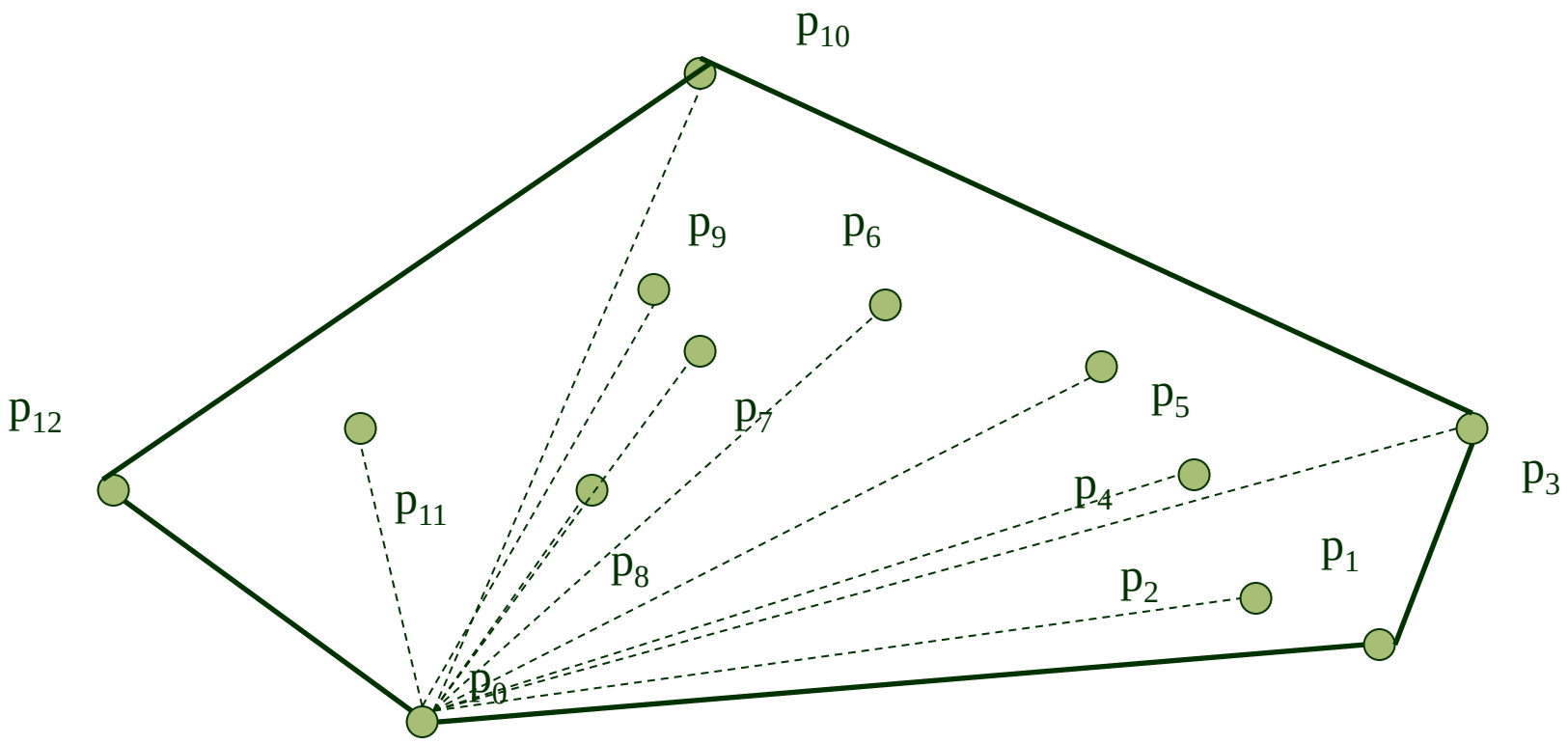
Graham Scan - Example



Graham Scan - Example



Graham Scan - Example



Graham Scan - Algorithm

```
GRAHAM-SCAN(Q)
 1  let p_0 be the point in Q with the minimum y-coordinate,
          or the leftmost such point in case of a tie
 2  let ⟨p_1, p_2, ..., p_m⟩ be the remaining points in Q,
          sorted by polar angle in counterclockwise order around p_0
          (if more than one point has the same angle, remove all but
          the one that is farthest from p_0)
 3  PUSH(p_0, S)
 4  PUSH(p_1, S)
 5  PUSH(p_2, S)
 6  for i ← 3 to m
 7       do while the angle formed by points NEXT-TO-TOP(S), TOP(S),
                    and p_i makes a nonleft turn
 8            do POP(S)
 9          PUSH(p_i, S)
10  return S
```

Graham Scan - Algorithm

- Time complexity of Graham's scan:
  - $O(n \log n)$ time required to sort of angles in step 2.
  - $O(n)$ time required for visiting n points.(step 6 to step 9)

  Hence we can write the complexity of Graham's scan as:

$$T(n) = \mathrm{O}(n \lg n) + \mathrm{O}(n) = \mathrm{O}(n \lg n)$$

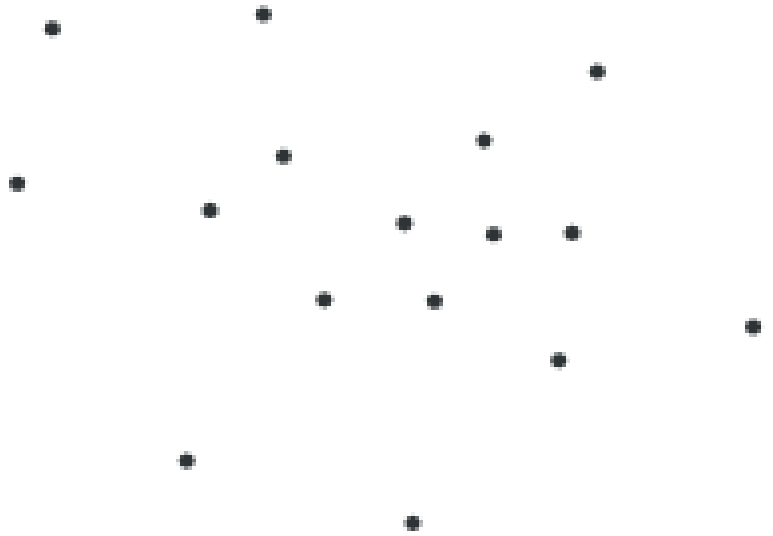
Divide and Conquer (Quickhull)

- QuickHull uses a Divide and Conquer approach similar to the Quick Sort algorithm.
- Benchmarks showed it is quite fast in most average cases.
- Recursive nature allows a fast and yet clean implementation.

Divide and Conquer (Quickhull)

Initial Input

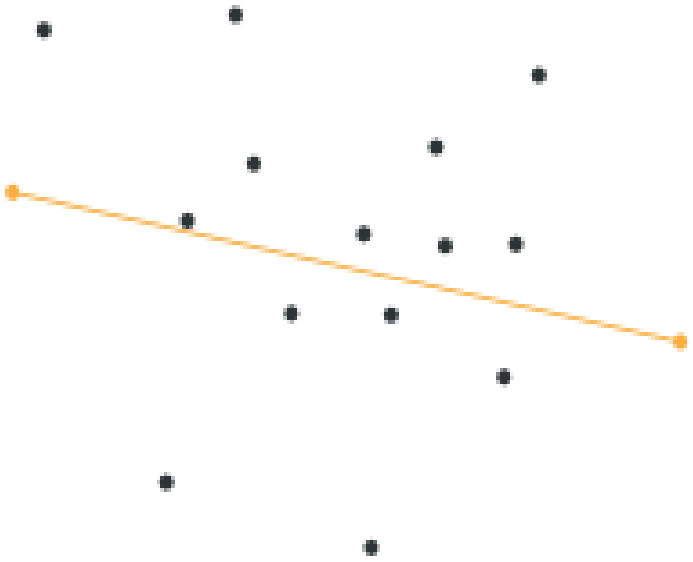
- The initial input to the algorithm is an arbitrary set of points as shown in the figure.



Divide and Conquer (Quickhull)

First Two Points on the Convex Hull

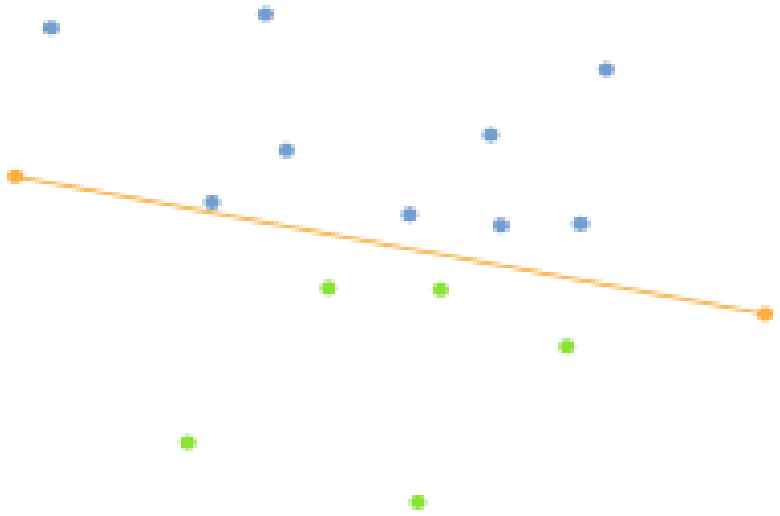
- Starting with the given set of points the first operation done is the calculation of the two maximal points on the horizontal axis. i.e.



Divide and Conquer (Quickhull)

Recursively Divide

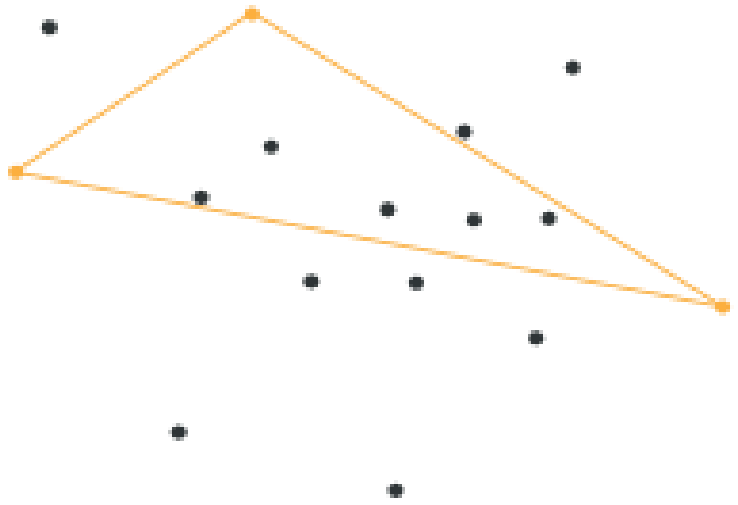
- Next the line formed by these two points is usedto divide the set into two different parts.
- Everything left from this line is considered onepart, everything right of it is considered another one.
- Both of these partsare processed recursively.



Divide and Conquer (Quickhull)

Max Distance Search

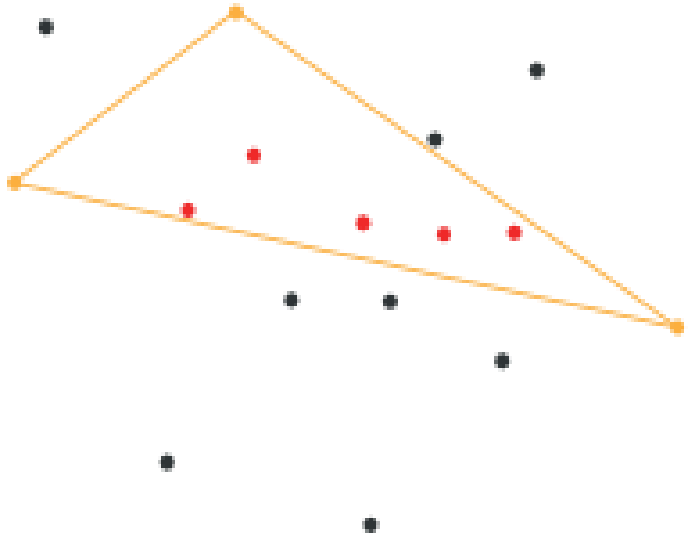
- To determine the next point on the convex hull a search for the point with the greatest distance from the dividing line is done.
- This point, together with the line start and end point forms a triangle.



Divide and Conquer (Quickhull)

Point Exclusion

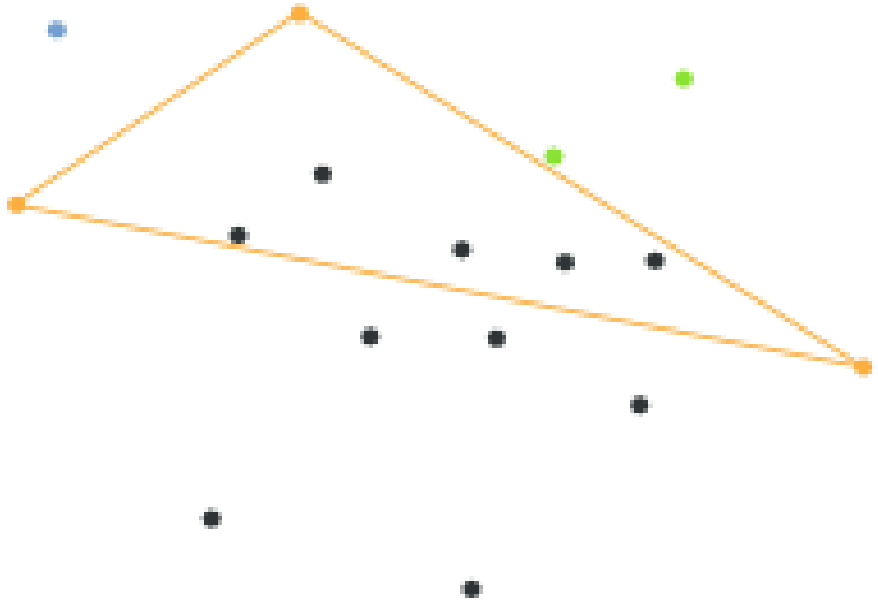
- All points inside this triangle can not be part of the convex hull polygon, as they are obviously lying in the convex hull of the three selected points.
- Therefore these points can be ignored for every further processing step.



Divide and Conquer (Quickhull)

Recursively Divide

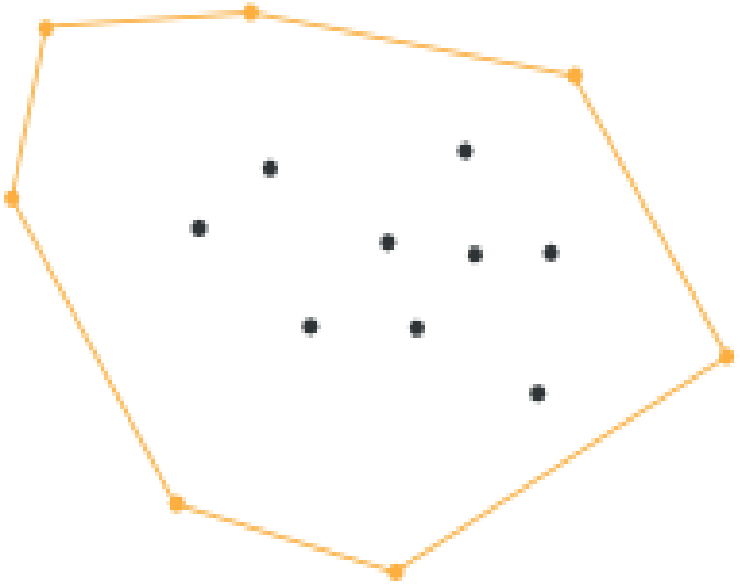
- Having this in mind the recursive processing can take place again.
- Everything right of the triangle is used as one subset, everything left of it as another one.



Divide and Conquer (Quickhull)

Abort Condition

- At some point the recursively processed point subset does only contain the start and end point of the dividing line.
- If this is case this line has to be a segment of the searched hull polygon and the recursion can come to an end.



Divide and Conquer (Quickhull)

Time Complexity of Quickhull

- The running time of Quickhull, as with QuickSort, depends on how evenly the points are split at each stage.
- If we assume that the points are ``evenly'' distributed, the running time will solve to $O(n \log n)$.
- if the splits are not balanced, then the running time can easily increase to $O(n^2)$.


Thank U