Advanced Algorithms

Geometric Algorithm

Graham Scan Algorithm

The ***convex hull*** of a set *Q* of points is the smallest convex polygon *P* for which each point in *Q* is either on the boundary of *P* or in its interior. We denote the convex hull of *Q* by CH(*Q*).

Intuitively, we can think of each point in *Q* as being a nail sticking out from a board. The convex hull is then the shape formed by a tight rubber band that surrounds all the nails.

Graham's scan use a technique called "rotational sweep," processing vertices in the order of the polar angles they form with a reference vertex.

Graham Scan Algorithm

Graham's scan solves the convex-hull problem by maintaining a stack *S* of candidate points.

Each point of the input set *Q* is pushed once onto the stack, and the points that are not vertices of CH(*Q*) are eventually popped from the stack.

When the algorithm terminates, stack *S* contains exactly the vertices of CH(*Q*), in counterclockwise order of their appearance on the boundary.

Graham Scan Algorithm

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 $\langle p_1, p_2, . . . , p_m\rangle$ be the remaining points in Q, sorted by polar angle in counterclockwise order around p_0 (if more than point has the same angle, remove all but the one that is farthest from p_0)

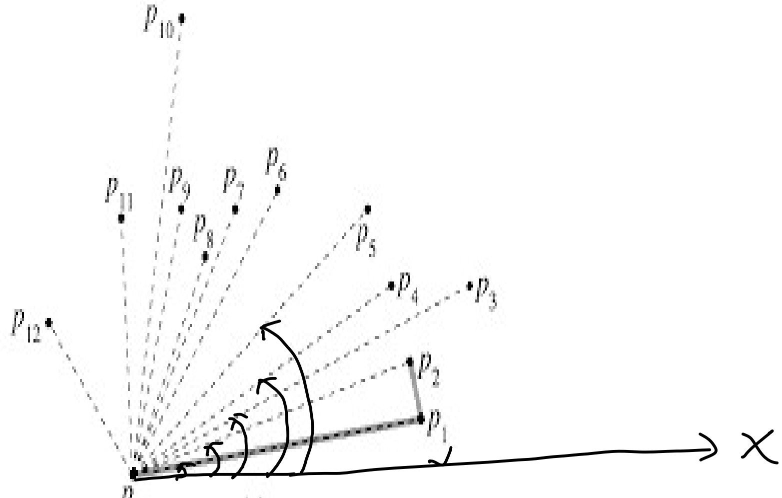
3 $top[S] \leftarrow 0$

4 PUSH(p_0, S)

5 PUSH(p_1, S)

6 PUSH(p_2, S)

Graham Scan Algorithm

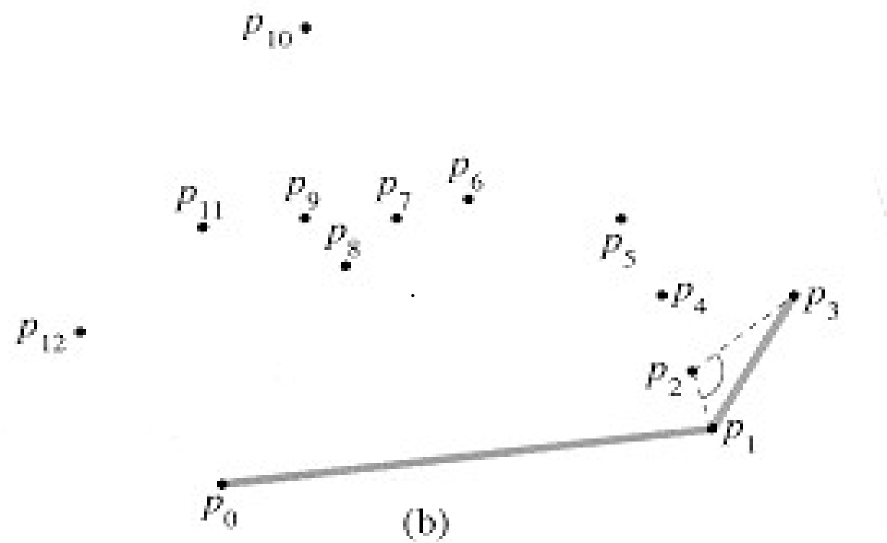


(a)

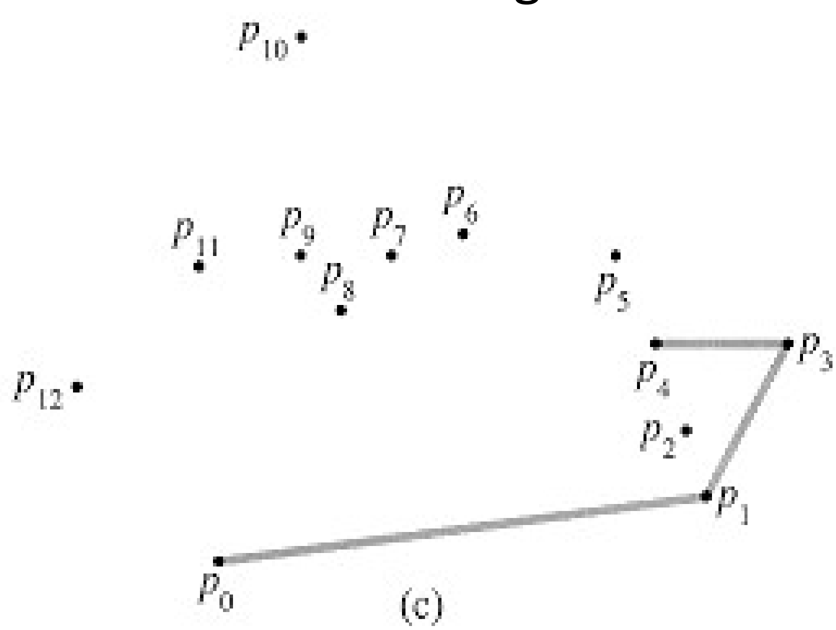
Graham Scan Algorithm

```
for i ← 3 to m
        do while the angle formed by points NEXT-TO-TOP(S),
TOP(S), and p_i makes a nonleft turn
                do POP(S)
            PUSH(S, p_i)
return S
```

Graham Scan Algorithm



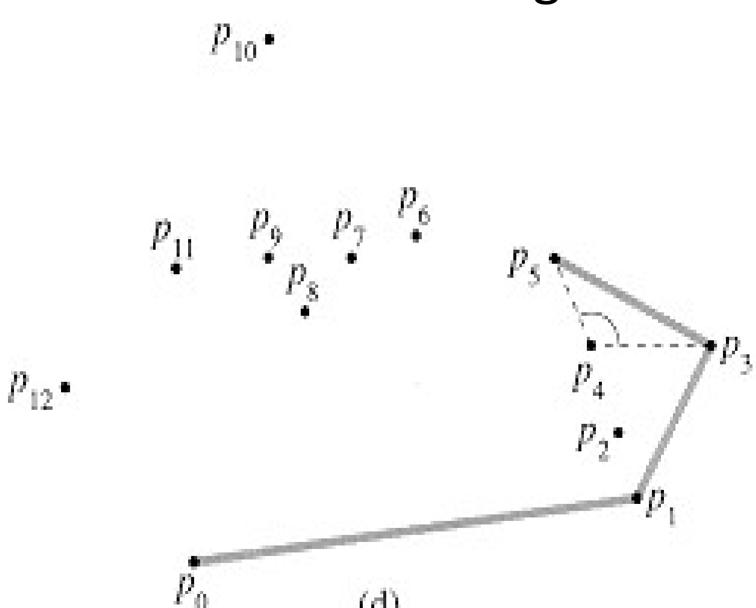
Graham Scan Algorithm



Graham Scan Algorithm

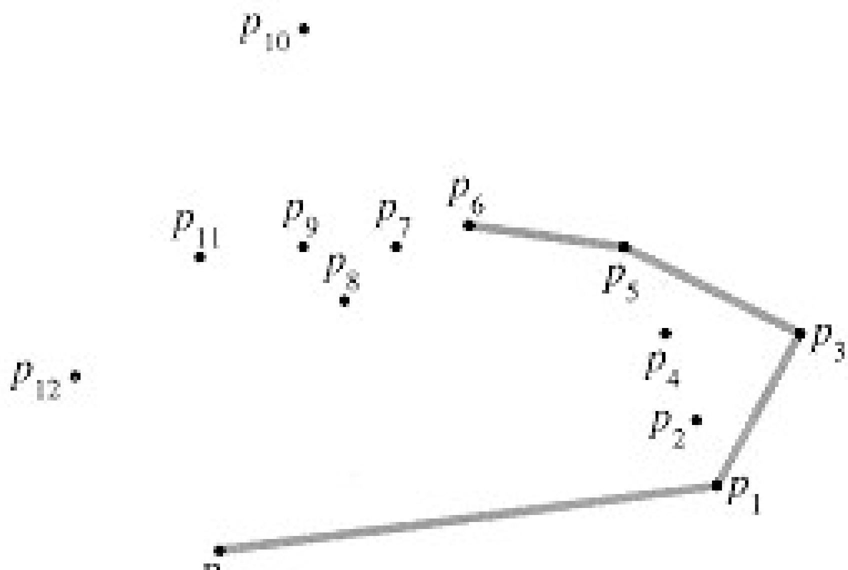
```
for i ← 3 to m
        do while the angle formed by points NEXT-TO-TOP(S),
TOP(S), and p_i makes a nonleft turn
                do POP(S)
            PUSH(S, p_i)
return S
```

Graham Scan Algorithm



(d)

Graham Scan Algorithm

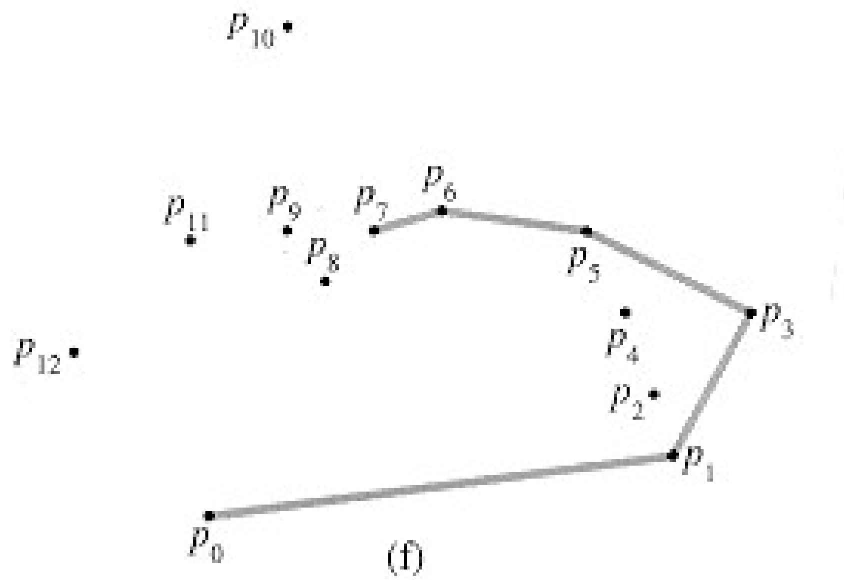


(e)

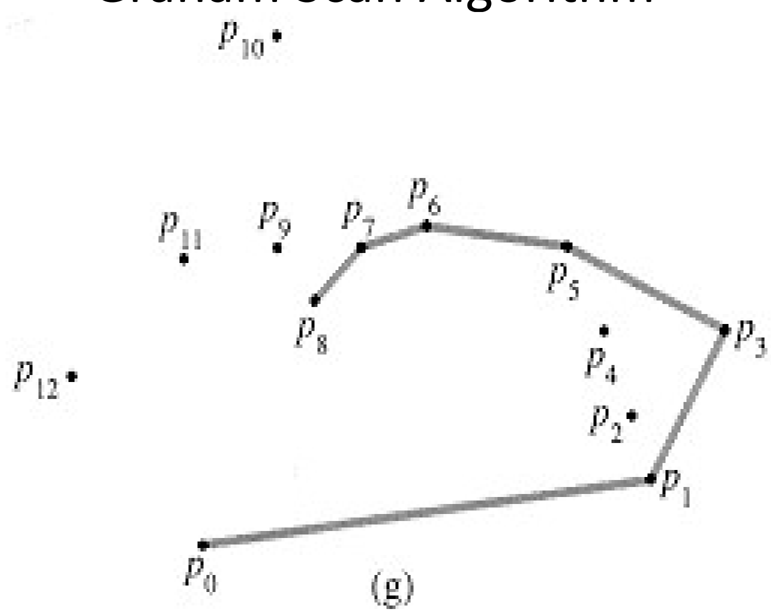
Graham Scan Algorithm

for $i \leftarrow 3$ **to** m
  do while the angle formed by points NEXT-TO-TOP(S), TOP(S), and p_i makes a nonleft turn
    do POP(S)
  PUSH(S, p_i)
return S

Graham Scan Algorithm



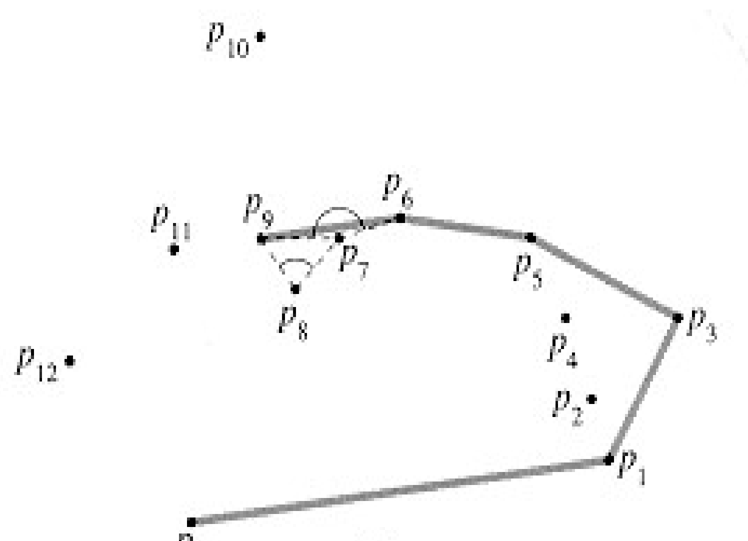
Graham Scan Algorithm



Graham Scan Algorithm

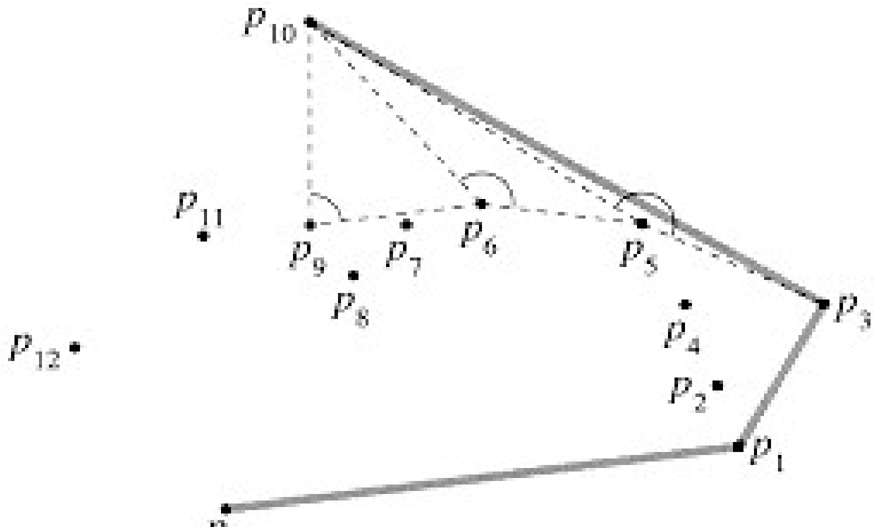
```
for i ← 3 to m
        do while the angle formed by points NEXT-TO-TOP(S),
TOP(S), and p_i makes a nonleft turn
                do POP(S)
            PUSH(S, p_i)
return S
```

Graham Scan Algorithm



(h)

Graham Scan Algorithm

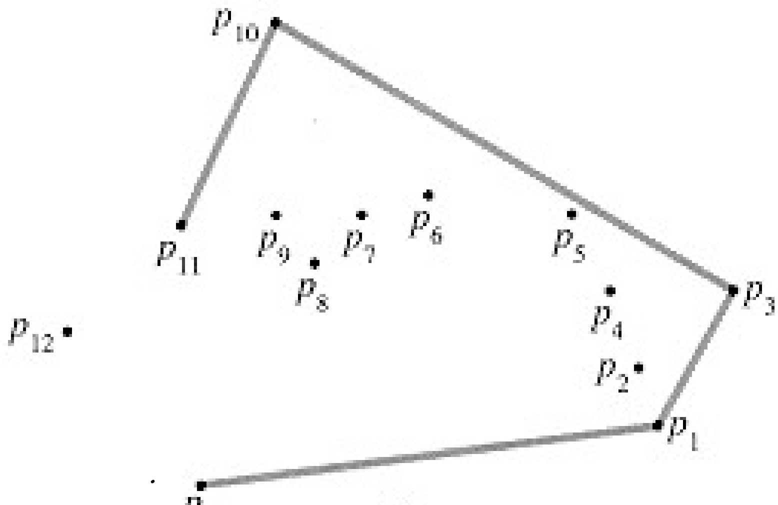


(1)

Graham Scan Algorithm

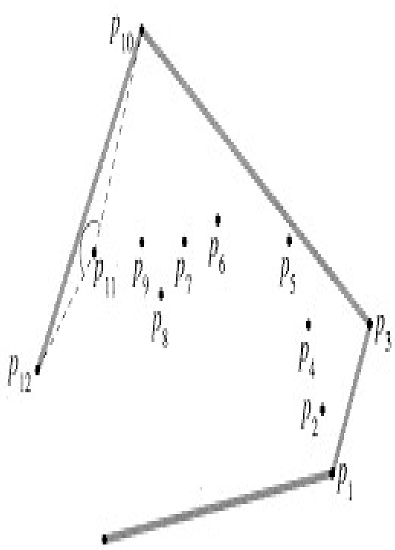
```
for i ← 3 to m
    do while the angle formed by points NEXT-TO-TOP(S),
TOP(S), and p_i makes a nonleft turn
        do POP(S)
    PUSH(S, p_i)
return S
```

Graham Scan Algorithm

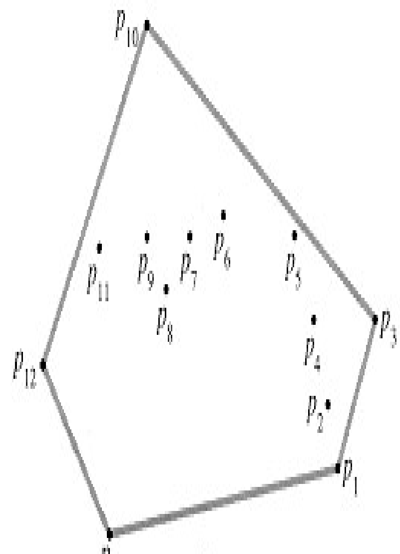


(j)

Graham Scan Algorithm



(k)

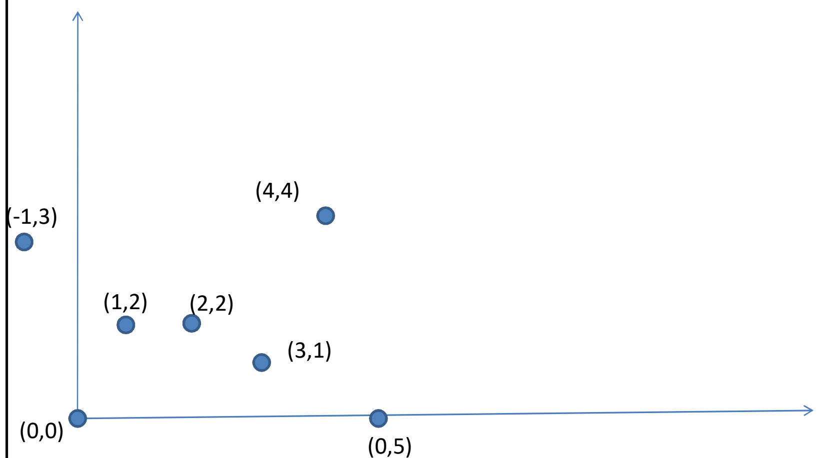


(l)

Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

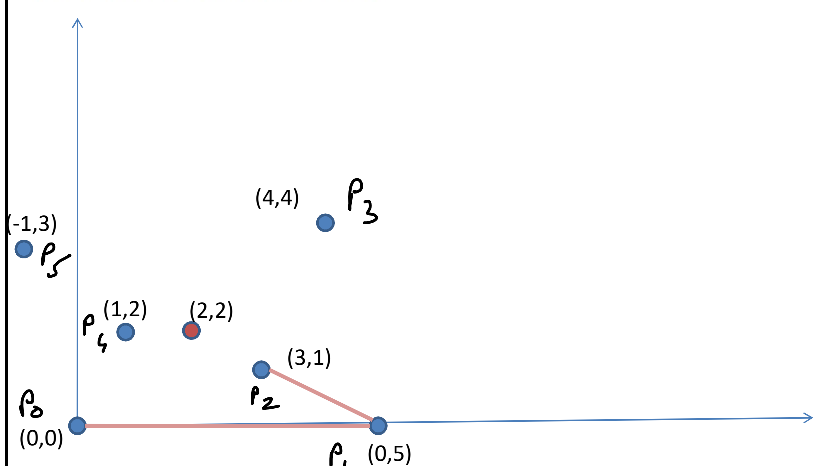
{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

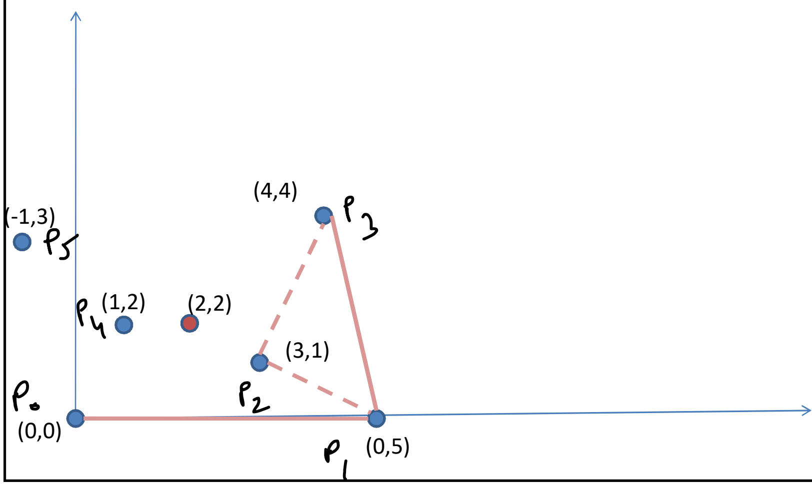
{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

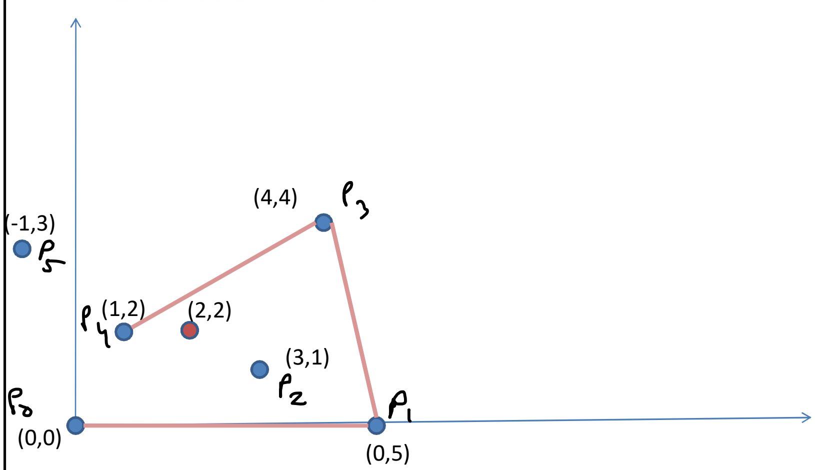
{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

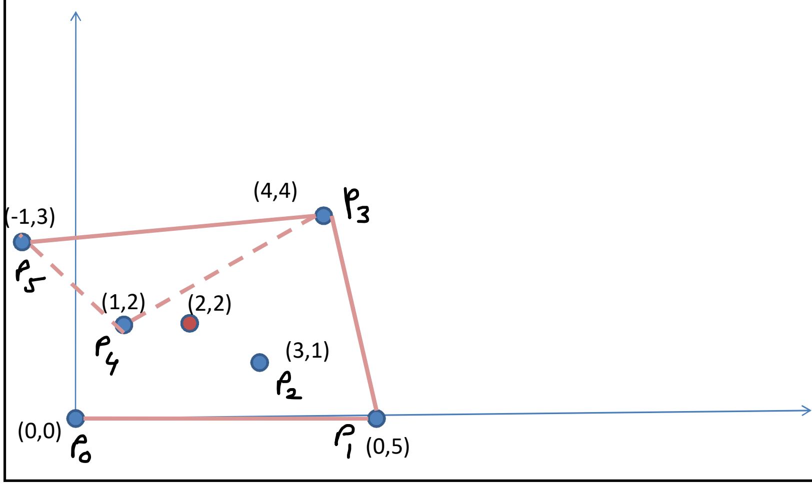
{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

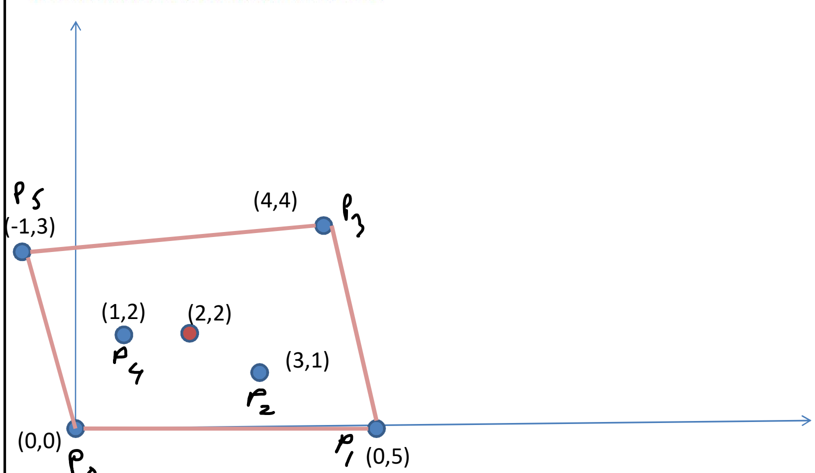
{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Gram Scan Algorithm

Explain working of Graham scan algorithm **step by step** and generate Convex Hull for given set of points.

{{0,0},{3,1},{4,4},{1,2},{0,5},{2,2},{-1,3}}



Time Complexity

the running time of `GRAHAM-SCAN` is $O(n \lg n)$, where $n = |Q|$.

Line 1 takes (n) time.

Line 2 takes $O(n \lg n)$ time, using merge sort or heapsort to sort the polar angles and the cross-product method to compare angles. (Removing all but the farthest point with the same polar angle can be done in a total of $O(n)$ time.)

Lines 3-6 take $O(1)$ time.

Because $m <=$ $n - 1$, the **for** loop of lines 7-10 is executed at most $n - 3$ times. Since `PUSH` takes $O(1)$ time, each iteration takes $O(1)$ time exclusive of the time spent in the **while** loop of lines 8-9, and thus overall the **for** loop takes $O(n)$ time exclusive of the nested **while** loop.

Time Complexity

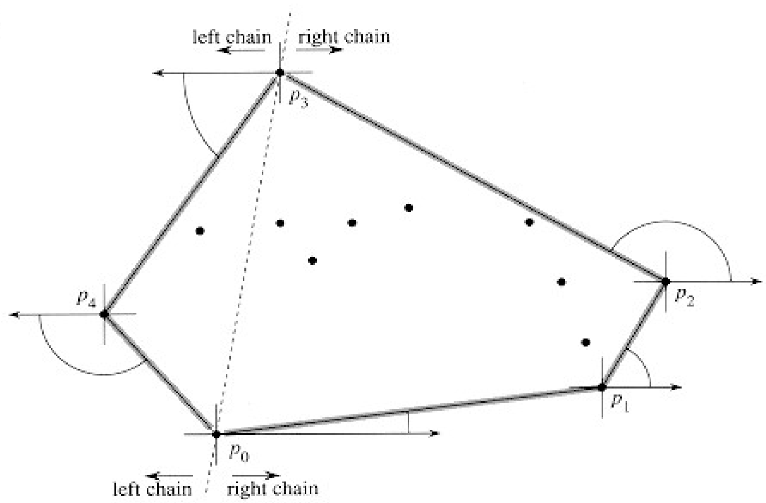
We use the aggregate method to show that the **while** loop takes $O(n)$ time overall.

For i = O, 1, . . . , m, each point p_i is pushed onto stack S exactly once.

At least three points--p_0, p_1, and p_m--are never popped from the stack, so that in fact at most $m - 2$ `POP` operations are performed in total. Each iteration of the **while** loop performs one `POP`, and so there are at most $m - 2$ iterations of the **while** loop altogether.

Since the test in line 8 takes $O(1)$ time, each call of `POP` takes $O(1)$ time, and $m <= n - 1$, the total time taken by the **while** loop is $O(n)$. Thus, the running time of `GRAHAM-SCAN` is $O(n \lg n)$.

Jarvis's march Approach



Jarvis's march Approach

Jarvis's march builds a sequence $H = \langle p_0, p_1, \ldots, p_h-1 \rangle$ of the vertices of CH(Q).

We start with p_0. The next convex hull vertex p_1 has the least polar angle with respect to p_0. (In case of ties, we choose the point farthest from p_0.)

Similarly, p_2 has the least polar angle with respect to p_l, and so on. When we reach the highest vertex, say p_k (breaking ties by choosing the farthest such vertex), we have constructed the ***right chain*** of CH(Q).

To construct the ***left chain***, we start at p_k and choose p_k+l as the point with the least polar angle with respect to p_k, but *from the negative x-axis*. We continue on, forming the left chain by taking polar angles from the negative x-axis, until we come back to our original vertex p_0.

Jarvis’s march Approach

Jarvis's march computes the convex hull of a set Q of points by a technique known as ***package wrapping*** (or ***gift wrapping***). The algorithm runs in time $O(nh)$, where h is the number of vertices of $CH(Q)$.

When h is $O(\lg n)$, Jarvis's march is asymptotically faster than Graham's scan.