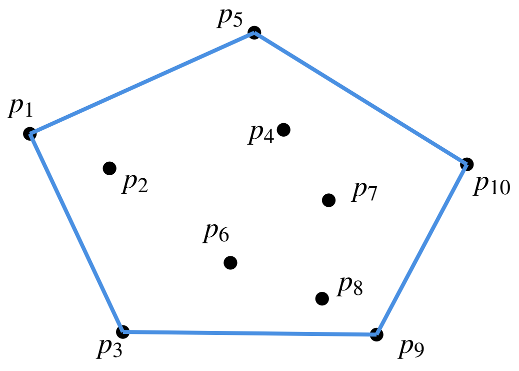
Convex Hull

Definitions

Definition 1 (Convex Hull)**.** Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of points on 2D plane, each of which is represented as $p_i = (x_i, y_i)$. The *convex hull* of P, denoted as $CH(P)$, is defined as the smallest, convex polygon that includes all points in P.

A convex hull is usually represented as the list of vertices (which is subset of P) of the polygon in counter-clockwise order. Such list is circular, and it's equivalent to shifting any number of vertices. If a point $p \in P$ is one of the vertices of the convex hull of P, we usually say that p is *on* the convex hull, and denote it as $p \in CH(P)$.



$$CH(P) = (p_1, p_3, p_9, p_{10}, p_5)$$

Figure 1: A set of points $P = \{p_1, p_2, \cdots, p_{10}\}$ and its convex hull.

For most computational geometry problems, we always assume that the data (points, lines, etc) given to us are in *general* situation. For example, we usually assume no three (or more) lines go through the same point, no three (or more) points are colinear, and no four points are on the same circle, etc.

Property 1. Let $p \in P$. Then $p \in CH(P)$ if and only if there exists a line l such that p is on l, and that all other points, i.e. $P \setminus \{p\}$, are on the same side of l.

The above property gives us a way to determine if a point is on the convex hull, which will be used in the following algorithm (e.g., Graham-Scan).

Property 2. For any set of points P and another point q, we have $CH(P \cup \{q\}) = CH(CH(P) \cup \{q\})$.

The above property gives a way to gradually contruct convex hull. Suppose we already know the convex hull of P, i.e., $CH(P)$. Now we want to calculate the convex hull of $P \cup \{q\}$. We only need to consider these points in $CH(P)$ and q: those points that are inside of the convex hull of P can be safely discarded.

Graham-Scan Algorithm

The idea of Graham-Scan is to gradually construct the convex hull. The first step of this algorithm is the identification of the lowest point in P, i.e., the point with smallest y-coordinate, denoted as p_*. The second step is to sort all others points in counter-clockwise order w.r.t. p_*. Specifically, for any point $p \in P$, the measure we use in sorting is the *angle* defined by points p, p_* and $(+\infty, y\text{-coordinate of } p_*)$. We define such angle for p_* is 0. All points are then sorted in ascending order by their angles. After sorting, for the sake of simplicity, we rename all points in P in this order, i.e., rename p_* as p_1, the point with second smallest angle as p_2, and so on. These two steps can be regarded as *preprocessing* steps of the Graham-Scan algorithm (i.e., first 3 lines of Algorithm Graham-Scan); the resulting of it is a sorted list of points P. See Figure 2.

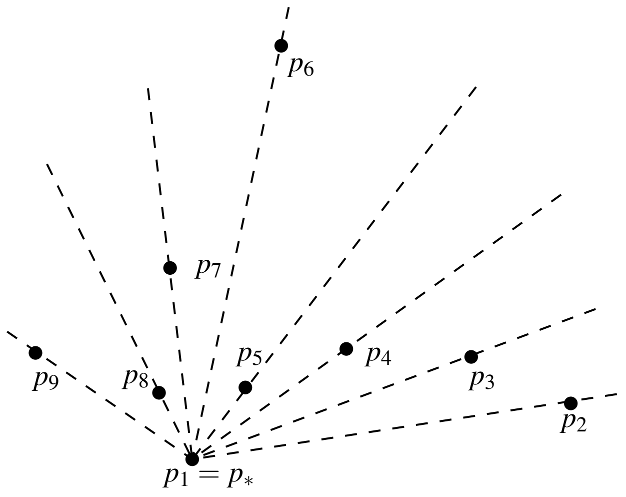


Figure 2: Preprocessing steps of Graham-Scan.

The main body of Graham-Scan is referred to as Graham-Scan-Core. It processess points in P (notice that now they are properly sorted) one by one and gradually construct the convex hull. It maintains a *stack* data structure, called S, and keeps the following variant:

Invariant: after processing the first k points of P, the convex hull of these k points, i.e., $CH(p_1, p_2, \cdots, p_k)$ will be stored in the stack S: the list of points in S from bottom to top gives the list of the vertices of the convex hull in counter-clockwise order.

How to guarantee above invariant? Consider the general case when we are processing p_k. Now we know that S stores $CH(p_1, p_2, \cdots, p_{k-1})$, and the goal is to determine $CH(p_1, p_2, \cdots, p_k)$. Following above Property 2, we know that $CH(p_1, p_2, \cdots, p_k) = CH(CH(p_1, p_2, \cdots, p_{k-1}) \cup \{p_k\})$. Hence we only need to consider the points in S and the current point p_k.

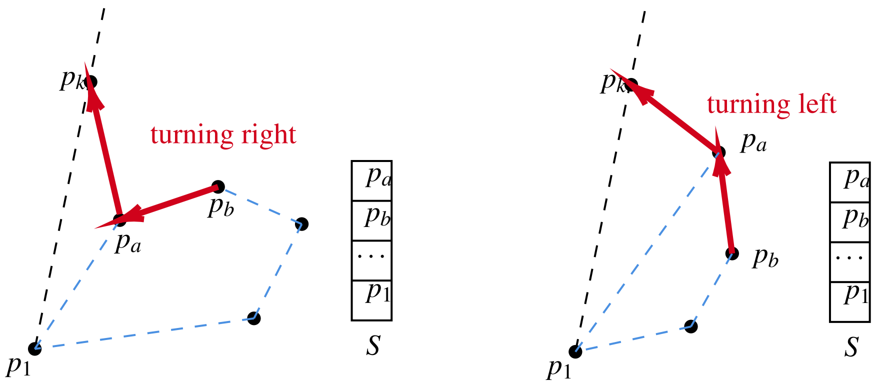


Figure 3: Procedure to decide if top element of S is on $CH(p_1, p_2, \cdots, p_k)$.

But the issue is that, points in S, which is exactly $CH(p_1, p_2, \cdots, p_{k-1}$, may not be in $CH(p_1, p_2, \cdots, p_k)$. We therefore need to *determine* and *exclude* such points in S. How? See Figure 3 for an example. We use the following procedure to determine if the top element, called p_a, of S is on $CH(p_1, p_2, \cdots, p_k)$: we check the *orientation* of the three points $p_b \to p_a \to p_k$, where p_a and p_b are the top and second top elements of S respectively. If it's "turning right", then p_a will not be on $CH(p_1, p_2, \cdots, p_k)$. Why? Because in this case p_a will be within triangle $p_1 p_b p_k$ (the angle of p_a is in the middle of that of p_b and p_k). If this happens, we can remove p_a safely, i.e., by calling *pop (S)*, as p_a cannot be in the convex hull of the first k points. After we remove p_a from S, we will need to continue to examine the top element of S iteratively. If the orientation is "turning left", then actually we get the convex hull of the first k points—that's the points in S plus p_k, where p_k will be pushed into S and become the top element of S. We will give a formal proof after the algorithm.

The complete Gramham-Scan algorithm is illustrated with pseudo-code below.

```
Algorithm Graham-Scan ($P$)
    calculate point $p_*$ in $P$ with smallest $y$-coordinate;
    sort $P$ in ascending order of the angles w.r.t. $p_*$ and $(\infty, 0)$;
    rename points in $P$ so that points in $P$ are following above order;
    Graham-Scan-Core ($P$);
end algorithm;

Algorithm Graham-Scan-Core ($P = \{p_1, p_2, \cdots, p_n\}$)
    init empty stack $S$;
    push ($S, p_1$);
    push ($S, p_2$);
    push ($S, p_3$);
    for $k = 4$ to $n$
        while ($S$ is not empty)
            let $p_a$ and $p_b$ be the top two elements of $S$;
            check the orientation of $p_b \to p_a \to p_k$;
            if "turning right": pop (S) and then continue the while loop;
            if "turning left": break the while loop;
        end while;
        push ($S, p_k$);
    end for;
end algorithm;
```

Now we show that, when the while-loop breaks, i.e., the orientation is "turning left", then $S \cup \{p_k\}$ becomes the convex hull of $\{p_1, p_2, \cdots, p_k\}$. The proof is based on the definition of convex-hull: we will verify that $S \cup \{p_k\}$ is the smallest, convex polygon that includes all $\{p_1, p_2, \cdots, p_k\}$. First, we verify that $S \cup \{p_k\}$ forms a convex polygon. This is because, all consecutive 3 points of S are "turning left", as this is guaranteed by the algorithm: we push p_k right after verifying $p_b \to p_a \to p_k$ are turning left. To verify that it is circularly "turning left", we further need to show that two more consecutive 3 points, namely, (p_a, p_k, p_1) and (p_k, p_1, p_2), are also turning left. These are easy to see as $p_k p_1$ are the rightmost line and all other points, including p_1 and p_a are on the right side of this line. Second, we verify that the convex polygon formed by $S \cup \{p_k\}$ contains all points in $\{p_1, p_2, \cdots, p_k\}$. To see this, we just need to show that all poped points are not possible to be in $CH(p_1, p_2, \cdots, p_k)$, and this is true, as every poped point is inside of a triangle (Figure 3), and therefore must be inside of the convex hull. Third, we show that $S \cup \{p_k\}$ is the smallest convex polygon that include all first k points. This is obvious as $S \cup \{p_k\}$ is a subset of the first k points.

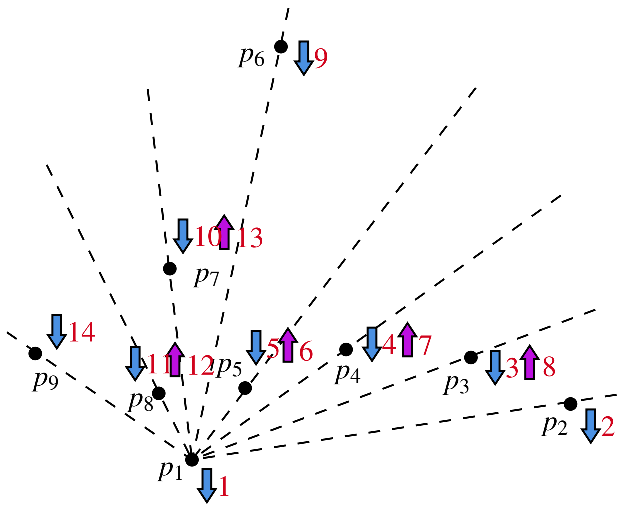


Figure 4: Running Graham-Scan-Core on the example given in Figure 2. Blue and pink arrows show the push and pop operations; the red numbers show the sequential of these operations.

The first 3 lines of Gramham-Scan algorithm corresponds to the preprocessing steps, which produce a sorted list of points. The main body of the algorithm, referred to as "Graham-Scan-Core" here, gradually construct the convex hull and guarantees above invariant. When the algorithm terminates, the convex hull of all points are stored in stack S.

Graham-Scan-Core can be regarded as an independent algorithm (you will find its use in the divide-and-conquer algorithm for convex hull). Its input is a list of points P, in which the first point, i.e. p_1, is the point with smallest y-coordinate, and the following points, i.e. $p_2, p_3, \cdots, p_n$, are placed in ascending order by their angle w.r.t. p_1 and $(+\infty, p_1.y)$. The output of Graham-Scan-Core is $CH(P)$ which will be stored in the stack S.

There are two technical details in implementing above algorithms. First, in the preprocessing step, for each point p_i do we really need to calculate the actual angle of $\angle p_i p_* p_\infty$, where $p_\infty = (+\infty, p_*.y)$? No. Notice that the only purpose of calculating these angles is sorting these points. Therefore, only the relative order matters but not the actual angles. Hence, we can calculate $\cos(\cdot)$ of the angle and use it to sort, as $\cos(\cdot)$ is monotone over $[0, \pi]$. Clearly, $\cos(\angle p_i p_* p_\infty) = \overrightarrow{p_* p_i} \cdot (1,0) / |\overrightarrow{p_* p_i}|$.

Second, in Graham-Scan-Core, how to determine the orientation of 3 points (i.e. $p_b \to p_a \to p_k$)? We will use "right-hand-rule". Define two vectors: $\overrightarrow{A} := \overrightarrow{p_b p_a}$ and $\overrightarrow{B} = \overrightarrow{p_a p_k}$. Let $\overrightarrow{A} = (x_A, y_A)$ and $\overrightarrow{B} = (x_B, y_B)$ be their coordinates, which can be calculated from the coordinates of p_b, p_a and p_k. The right-hand-rule gives that

$$\begin{cases} x_A y_B - x_B y_A > 0 & \text{if and only if} \quad p_b \to p_a \to p_k \text{ ``turning left''} \\ x_A y_B - x_B y_A < 0 & \text{if and only if} \quad p_b \to p_a \to p_k \text{ ``turning right''} \\ x_A y_B - x_B y_A = 0 & \text{if and only if} \quad p_b, p_a, p_k \text{ ``colinear''} \end{cases}$$