```python
def order_clockwise(self, p):
    points = []
    for point in p:
        points.append((point.x(), point.y()))
    sorted_points = sorted(points, key=lambda point: atan2(point[1], point[0]),
reverse=True)
    out = []
    for point in sorted_points:
        q = QPointF();
        q.setX(point[0])
        q.setY(point[1])
        out.append(q)
    return out


def get_leftmost_index(self, points):
    leftmost_point = points[0]
    for i in range(1, len(points)):
        if points[i].x() < leftmost_point.x():
            leftmost_point = points[i]

    return points.index(leftmost_point)


def get_rightmost_index(self, points):
    rightmost_point = points[0]
    for i in range(1, len(points)):
        if points[i].x() > rightmost_point.x():
            rightmost_point = points[i]

    return points.index(rightmost_point)


def convex_hull(self, sorted_points):
    if len(sorted_points) <= 3:
        return self.order_clockwise(sorted_points)  # return this in clockwise
ordering

    l = self.convex_hull(sorted_points[0:len(sorted_points) // 2])
    r = self.convex_hull(sorted_points[(len(sorted_points) // 2):])
    hull_points = self.merge(l, r)
    hull = self.order_clockwise(hull_points)
    return hull


def merge(self, l, r):
    upper_tangent = self.find_upper_tangent(l, r)
    lower_tangent = self.find_lower_tangent(l, r)
    # hull = [upper_tangent[0], upper_tangent[1], lower_tangent[0],
lower_tangent[1]]
    # hull = [lower_tangent[0], lower_tangent[1]]
    hull = l[l.index(lower_tangent[0]):]
    hull = hull + l[0:l.index(upper_tangent[0])]
    hull.append(upper_tangent[0])
    hull = hull + r[r.index(upper_tangent[1]):]
    hull = hull + r[0:r.index(lower_tangent[1]) - 1]

    return hull


def find_upper_tangent(self, l, r):
    l_idx = self.get_rightmost_index(l)
    r_idx = self.get_leftmost_index(r)

    prev_l = []
    prev_r = []
    is_left_tangent, is_right_tangent = False, False

    slope = float('inf')
    while (not is_left_tangent) and (not is_right_tangent):
        while not is_left_tangent:
            new_slope = self.get_slope(l[l_idx], r[r_idx])
            if new_slope > slope:
                l_idx = prev_l.pop()
                is_left_tangent = True
                is_right_tangent = False
            else:
                prev_l.append(l_idx)
                l_idx -= 1
                if l_idx < 0:
                    l_idx = len(l) - 1
                slope = new_slope

        slope = float('-inf')
        r_idx = self.get_leftmost_index(r)

        while not is_right_tangent:
            new_slope = self.get_slope(l[l_idx], r[r_idx])
            if new_slope < slope:
                r_idx = prev_r.pop()
                is_right_tangent = True
                is_left_tangent = False
            else:
                prev_r.append(r_idx)
                r_idx += 1
                if r_idx > len(r) - 1:
                    r_idx = 0
                slope = new_slope

    return [l[l_idx], r[r_idx]]


def find_lower_tangent(self, l, r):
    l_idx = self.get_rightmost_index(l)
    r_idx = self.get_leftmost_index(r)
    prev_l = []
    prev_r = []
    is_left_tangent, is_right_tangent = False, False

    slope = float('-inf')

    while (not is_left_tangent) and (not is_right_tangent):
        while not is_left_tangent:
```

```python
            new_slope = self.get_slope(l[l_idx], r[r_idx])
            if new_slope < slope:  # If slope decreases, we've found it.
                l_idx = prev_l.pop()
                is_left_tangent = True
            else:
                prev_l.append(l_idx)
                l_idx -= 1
                if l_idx > len(l) - 1:
                    l_idx = 0
                slope = new_slope
                is_right_tangent = False

        slope = float('inf')
        r_idx = self.get_leftmost_index(r)

        while not is_right_tangent:
            new_slope = self.get_slope(l[l_idx], r[r_idx])
            if new_slope > slope:
                r_idx = prev_r.pop()
                is_right_tangent = True
            else:
                prev_r.append(r_idx)
                r_idx -= 1
                if r_idx < 0:
                    r_idx = len(r) - 1
                slope = new_slope
                is_left_tangent = False

    return [l[l_idx], r[r_idx]]


def get_slope(self, p1, p2):
    if p1 == p2:
        return float('inf')
    return (p2.y() - p1.y()) / (p2.x() - p1.x())
```