Time Complexity Analysis

1) Brute force

Algorithm SLOWCONVEXHULL(P)

Input. A set P of points in the plane.

Output. A list L containing the vertices of CH(P) in clockwise order.

```
1. E   / 0.
2. for all ordered pairs (p;q) 2 P P with p not equal to q
3.          do valid  true
4.                    for all points r 2 P not equal to p or q
5.                             do if r lies to the left of the directed line from p to q
6.                                       then valid  false.
7.                   if valid then Add the directed edge !pq to E.
8. From the set E of edges construct a list L of vertices of CH(P), sorted in clockwise order.
```

1. -> O(1)

2. ~ 7. -> O(n ^ 3);

8. -> O(nlogn) since I use graham scan to get extreme points in clockwise order. n itself is O(2n) = O(n) since every extreme points at most appear twice at this point, one is the endpoint of one edge of convex hull, the other is the one of another edge.

So O(n ^ 3) overall for brute force.

2) Graham Scan

Algorithm CONVEXHULL(P)

Input. A set P of points in the plane.

Output. A list containing the vertices of CH(P) in clockwise order.

```
1. Sort the points by x-coordinate, resulting in a sequence p1; : : : ; pn.
```

```
2. Put the points p1 and p2 in a list Lupper, with p1 as the first point.
3. for i 3 to n
4.      do Append pi to Lupper.
5.          while Lupper contains more than two points and the last three points in Lupper do not make a right turn
6.              do Delete the middle of the last three points from Lupper.
7. Put the points pn and pn−1 in a list Llower, with pn as the first point.
8. for i n−2 downto 1
9.      do Append pi to Llower.
10.         while Llower contains more than 2 points and the last three points in Llower do not make a right turn
11.             do Delete the middle of the last three points from Llower.
12. Remove the first and the last point from Llower to avoid duplication of the points where the
upper and lower hull meet.
13. Append Llower to Lupper, and call the resulting list L.
14. return L
```

1. -> O(nlogn);

2. -> O(1);

3. ~ 6. -> O(n);

7. ~ 11. -> O(n);

12. -> O(1)

13. -> O(n);

So O(nlogn) overall for graham scan.

Comparison table between graham scan and brute force with different input size

Input Size	**Brute Force**	**Graham Scan**
N = 10	0 ms = 0 s	0 ms = 0 s
N = 100	0 ms = 0 s	0 ms = 0 s
N = 1000	3019 ms = 3.019 s	31 ms = 0.031 s
N = 10000	253352 ms = 253.352 s = 4.2 mins	95 ms = 0.095 s
N = 100000	more than 5 mins	1890 ms = 1.89 s

As we can see, the running time required by brute force grows much fast than graham scan, especially the time is extremely large when n = 10000 or 100000. Furthermore, if we see n as in terms of brute force, n ^ 3 ,and we will get n = 15 for N = 1000, n = 65 for N = 10000 roughly, and then this n to calculate asymptotic running time for graham scan, which is that 17 for N = 1000 and 117 for N = 10000. In this way, we can have an intuitive understanding of difference in efficiency between two algorithms.