Prob. 1	Prob. 2	Prob. 3	Prob. 4

Problem 1.

1. We will use binary search on the points so points must be sorted and $O(1)$ time access is preferable. Thus, our algorithm takes its input as a sorted array of points.

   Since the polygon is convex, the angle to a reference point can increase but then it should decrease and if keeping going in the same way, after some time it will again increase. Let's define $a(x_i)$ is the angle $\angle abx_i$.

   - Find the index k of array where consecutive angles change sign. This is done by binary search with three angle calculations in each step of binary search.
     - $sign(a(x_k) - a(x_{k-1})) = -sign(a_{k+1} - a_k)$
   - Note down k because it is our tipping point where it will be used in the binary search for the intervals $[x_1, x_k]$ and $[x_k, x_n]$.
   - if $a(x_k) - a(x_{k-1})$ is positive, x_k is maximum point and then it starts to decrease. Otherwise, it is the minimum. According to k being minimum or maximum, binary search only takes left or right on the given intervals.
   - Binary search is run twice on the above two intervals to find the point that is doing maximum angle with function $a(x_i)$ as a comparison metric.
   - Finally, we compare two local maximum angles to find the maximum and return the max of them. Noted that if global maximum is k, then we actually one angle because both binary search queries will get the same result.

   Analysis:

   - We have used binary search to tipping point x_k in $|H|$ points so the steps requires $\log(|H|)$ time.
   - We had two intervals that are strictly less than $|H|$ so their complexity is also bounded by $\log(|H|)$.
   - In total:

$$\begin{aligned} \textit{tipping point} + \textit{search in subintervals} &= O(\log(|H|) + 2 \cdot O(\log|H|) \\ &= O(\log(|H|)) \end{aligned}$$

2. Analysis of the given algorithm:

   - Algorithm returns k points at the end and since these k points define convex hull, k equals to s.

- In the second loop of the algorithm, k counts from 1 to K. K is multiplied by itself in each step and when K is bigger than s, algorithm ends. Therefore, it takes us $O(\log(s))$ steps to pass over s.
- In each step, we are spending $O(\frac{n}{K} \cdot K\log(K))$ time for convex hull calculations, $O(n)$ time for the point with the smallest y-coordinate and $O(\frac{n}{K} \cdot K\log(K))$ for the nested loop(first loop counts until K and second loop counts until $m = \frac{n}{K}$ and inside of the loop is explained in the first part and it is $O(\log(K)$ since $|H_i|$ can be at most K).
- In total:

$$\begin{aligned} O(\log(s)) \cdot (2 \cdot (O(\frac{n}{K} \cdot K\log(K))) + n) &= O(\log(s)) \cdot O(n \cdot \log(K)) \\ &= O(n \cdot \log(s)) \textit{ where } O(K) \leq O(s) \end{aligned}$$

Problem 2.

Problem 3.

We've seen in class that we can create a Voronoi diagram in O($n \log n$). From that we can imagine that before the algorithm we define an array of size n keeping the closest point of p_i at the i-th position. The idea is simple, we just want to keep track of the closest point for each p_i while we contruct the Voronoi diagram. During the algorithm to build a Voronoi diagram every time we create an edge separating two points in P we check for those two points if this new neighbor is closer to the one in the array. Checking and updating the array are basic operations so it can be done in O(1). Thus we just add some O(1) operations for the current steps in the original algorithm so the result of this new algorithm is still in O($n \log n$).

Problem 4.