Problem 1. Let A be an array of n real values. If we start from some position and then look at these values in a cyclic manner, we see a pattern where the values initially increase monotonically and then decrease monotonically. For example, $A = (10, 20, 30, 25, 15, 0, 5)$ has the property: by inspecting the values in the order "$0, 5, 10, 20, 30, 25, 15$", we observe the pattern mentioned earlier. On the other hand, $A = (5, 20, 30, 25, 15, 0, 10)$ does not have the property. Design an algorithm to find the maximum value in A in $O(\log n)$ time.

Solution. Case 1: The array is already unimodal, looking as if concaving up or down.

This can be verified by checking $A[1], A[2]$ and $A[n-1], A[n]$, determining the monotonicity.

For the case of concaving up, we just need to compare $A[1]$ and $A[n]$.

For the case of concaving down, we apply a ternary search.

Case 2: We can first eliminate the pit created around the minimum element in A. Notice that if the array is already decreasing initially, then the element in the final position is no less than the first element by observing monotonicity. Otherwise, we eliminate the tail on the right hand side of A.

This can be done by binary lifting. Taking steps of 2^k (the largest k satisfying $1 + 2^k \leq n$), $2^{k-1}, \ldots, 4, 2, 1$ (denote the current position as p), add it to p if $A[p + 2^i] < A[p]$.

Now the sequence formed by $A[p], A[p+1], \ldots, A[n]$ must be unimodal, applying ternary search once should allow us to find the maximum element.

Problem 2. Let $P_1, \cdots, P_m$ be m arbitrary convex polygons, each of which has no more than k vertices. Let ℓ be a line in the plane such that all of $P_1, \cdots, P_m$ fall on the left side of ℓ. Now, fix a point p on ℓ. We want to turn ℓ counterclockwise with p as the pivot, and stop as soon as ℓ hits a vertex of any polygon. Design an algorithm to find in $O(m \log k)$ time the first vertex hit.

Solution. Consider the case consisting only one convex polygon. We can reduce the problem using the algorithm described by Problem 1.

Note that we have to find the point P_i minimizing the angle formed by positive y-axis, p, and P_i ranged from $(0, \pi]$ (we can either tweak the algorithm a bit or add a minus sign ahead of these angles so to retrieve minimum angle using the original algorithm).

Minimizing all m angles found in the m polygons, we solve this problem in $O(m \log k)$ time.

Problem 3. Let S be a set of n points in $\mathbb{R}^2$. You are given an integer $\widehat{k}$ that is guaranteed to be larger than or equal to the number of vertices on the convex hull of S. Give an algorithm that computes the convex hull in $O(n \log \widehat{k})$ time.
(Hint: Arbitrarily divide S into groups of size $\widehat{k}$ and apply the result of Problem 2.)

Solution. Arbitrarily divide S groups of points with size $\widehat{k}$ and use Graham's Scan Algorithm, this computes $\lceil \frac{n}{\widehat{k}} \rceil$ convex hulls in $O(n/\widehat{k} \cdot \widehat{k} \log \widehat{k}) = O(n \log \widehat{k})$ time.

Now, it is guaranteed that the downmost point among the vertices of these $\lceil n/\widehat{k} \rceil$ convex hulls is included on the convex hull of S, this vertex can be found by scanning through all points in $O(n)$ time.

Using the result of Problem 2, we can compute the next vertex in P (counterclockwisely), in $O(n/\widehat{k} \cdot \log \widehat{k})$ time.

Since it is known that $\widehat{k}$ is the number of vertices on the convex hull of S, there will only be $\widehat{k} - 1$ vertices needed to be found by the Gift Wrapping Algorithm above. This takes $O(\widehat{k} \cdot n/\widehat{k} \cdot \log \widehat{k}) = O(n \log \widehat{k})$ time for running.

Problem 4. Design an algorithm to compute the convex hull of n 2D points in $O(n \log k)$ time, where k is the number of points on the convex hull.

Solution. Assuming that a guess of k, $\widehat{k}$, is less than k, our algorithm should be able to terminate once we have discovered more than $\widehat{k}$ hull vertices, the running time will be $O(n \log \widehat{k})$.

For the case that $\widehat{k} > k$, the algorithm above is able to identify that all points on the convex hull have been found and terminate (by noticing that it has returned to the first point on the convex hull of P). This should take $O(n \log \widehat{k})$ time, but we will show next that this should not be a worry for us.

Lemma. Suppose an algorithm runs in $O(n \log \widehat{k})$ time with a guess of an output size $\widehat{k}$, reporting failure if the actual output size $k < \widehat{k}$, and returning the required output if $k \geq \widehat{k}$. Assuming a guessing sequence with $\widehat{k}_i = 2^{2^i}$, we are able to derive the output correctly in $O(n \log k)$ time.

This lemma finishes proving the bound of this guessing algorithm, giving $O(n \log k)$ time complexity bound as required.

Proof of Lemma. Note that the guessing sequence ends at an integer satisfying $\widehat{k}_i = 2^{2^i} \geq k$.

The total running time would be

$$\begin{aligned} O(n \log 2^{2^0} + n \log 2^{2^1} + n \log 2^{2^2} + \ldots + n \log 2^{2^i}) &= O(n \cdot (1 + 2^1 + 2^2 + \ldots + 2^i)) \\ &= O(n \cdot 2^i) \\ &= O(n \log k). \end{aligned}$$