CSE 317: Design and Analysis of Algorithms — Quiz 1

Wednesday, September 17, 2025. Max marks: 10. Time: 15 minutes

Let $A = \{a_0, a_1, \ldots, a_{n-1}\}$ be a circular sequence of n distinct positive integers. In a circular sequence, the neighbors of element a_i are $a_{(i-1+n) \bmod n}$ and $a_{(i+1) \bmod n}$.
An element a_i is called a *circular peak* if it is larger than both of its neighbors. That is $a_i > a_{(i-1+n) \bmod n}$ as well as $a_i > a_{(i+1) \bmod n}$.

Design a divide-and-conquer algorithm that runs in $O(\log n)$ time to find the index of any single circular peak element. You may assume a circular peak always exists in an array/sequence of distinct positive integers.

Understanding the Question

We are asked to find an element in a circular array that is strictly greater than its two circular neighbors. In a circular array, the first element's left neighbor is the last element, and the last element's right neighbor is the first element. Our goal is to design an $O(\log n)$ algorithm, rather than the obvious $O(n)$ algorithm that checks every element one by one. The question guarantees that at least one such peak always exists.

Existence of a Circular Peak

Why must a circular peak exist?

- Start anywhere and walk around the circle.
- If the next element is larger, continue walking.
- Since numbers are distinct, you cannot increase forever in a circle.
- Eventually you must drop, and just before the drop you are at a peak.

Thus, at least one circular peak always exists.

Examples of different arrays:

- Strictly increasing: $[1, 2, 3, 4, 5]$. The last element 5 has neighbors 4 and 1, both smaller $\Rightarrow$ peak.
- Strictly decreasing: $[9, 7, 5, 3, 1]$. The first element 9 has neighbors 1 and 7, both smaller $\Rightarrow$ peak.
- Random unsorted: $[10, 3, 7, 12, 6]$. Peaks at 10 and 12.

Brute Force $O(n)$ Solution

The simplest method is to check each element and test if it is larger than both neighbors.

- For i from 0 to $n-1$: check if $A[i] > A[(i-1+n)\%n]$ and $A[i] > A[(i+1)\%n]$.
- If true, return i.

This clearly works but takes $O(n)$ time.

Algorithm (Divide and Conquer, $O(\log n)$)

We can do better using a binary-search style approach:

- Pick $mid = (lo + hi)//2$.
- If $A[mid]$ is greater than both neighbors, return mid.
- Else if $A[mid] < A[mid+1]$, then there must be a peak on the right side. Set $lo = mid+1$.
- Else ($A[mid] < A[mid-1]$), then there must be a peak on the left side. Set $hi = mid-1$.
- Repeat until a peak is found.

This works because in a circular array, if you are on an ascending slope, you must eventually come down, so there will be a peak in that direction.

Pseudocode

```
function CircularPeak(A, n):
    lo = 0, hi = n-1
    while lo <= hi:
        mid = (lo+hi)//2
        L = A[(mid-1+n)%n]
        M = A[mid]
        R = A[(mid+1)%n]

        if M > L and M > R:
            return mid
        else if M < R:
            lo = mid + 1
        else:
            hi = mid - 1
```

Dry Run

Example 1 (Unsorted): $A = [5, 9, 2, 8, 6]$

Iteration	lo	hi	mid	(L,M,R)	Decision
1	0	4	2	(9,2,8)	$M < R \Rightarrow$ go right ($lo = 3$)
2	3	4	3	(2,8,6)	M is peak $\Rightarrow$ return 3

Example 2 (Sorted Increasing): $A = [1, 2, 3, 4, 5]$

Iteration	lo	hi	mid	(L,M,R)	Decision
1	0	4	2	(2,3,4)	$M < R \Rightarrow$ go right ($lo = 3$)
2	3	4	3	(3,4,5)	$M < R \Rightarrow$ go right ($lo = 4$)
3	4	4	4	(4,5,1)	M is peak $\Rightarrow$ return 4

Example 3 (Sorted Decreasing): $A = [9, 7, 5, 3, 1]$

Iteration	lo	hi	mid	(L,M,R)	Decision
1	0	4	2	(7,5,3)	$M < L \Rightarrow$ go left ($hi = 1$)
2	0	1	0	(1,9,7)	M is peak $\Rightarrow$ return 0

Running Time

Each iteration does $O(1)$ work. The interval is halved each time, just like binary search. Therefore, the algorithm runs in $O(\log n)$ time.

Marking Criteria (10 points total)

- **Base attempt (3 marks):** Defines circular peak OR shows some effort toward the algorithm OR did linear search (everyone attempting gets this).
- **+1 mark:** Mentions divide-and-conquer idea (start from middle, not linear scan).
- **+1 mark:** Mid index calculated correctly ($(low + high)//2$).
- **+1 mark:** Left and right neighbors calculated correctly (with mod).
- **+1 mark:** Checks peak condition properly ($A[mid] > left$ && $A[mid] > right$).
- **+1 mark:** Correct branching logic (move to larger neighbor side).
- **+1 mark:** Explains algorithm halving $\rightarrow O(\log n)$ complexity.
- **+1 mark:** Clear, structured presentation, complete algorithm (pseudocode or well-ordered explanation).

Note: Examples are not necessary for marks. Any format of algorithm description is acceptable. The rubric focuses on essential reasoning steps.