

CS381 Homework 0 Problem 1

Connor Couetil

January 18, 2026

1 Exercise 1.3.1

You can tell where you are in a rotated array by comparing against the first element of the array. That way, you tell if you are in the rotated portion or merely the shifted portion.

Recursive Algorithm

RotationIndex(array, start, end)

1. If $start == end$
 - A. Return 1
2. $midpoint = \lceil (end - start + 1)/2 \rceil + start$
3. $neighbor = (midpoint - 1) \bmod \text{Length}(array)$
4. If $array[midpoint] < array[neighbor]$
 - A. Return $midpoint$
5. If $array[midpoint] > array[1]$
 - A. Return $\text{RotationIndex}(array, midpoint, end)$
6. Return $\text{RotationIndex}(array, start, midpoint)$

Runtime Analysis $O(\log n)$ due to recurrence relation $\approx S(n) = S(n/2) + c$, $S(1) = C$

2 Exercise 1.3.2

For every point in a binary search, we check the neighbors of the midpoint for a peak, otherwise we follow the upslope to the peak of the mountain.

Recursive Algorithm

MountainIndex(array, start, end)

1. If $start == end$
 - A. Return 1
2. $midpoint = \lceil (end - start + 1)/2 \rceil + start$
3. $left = (midpoint - 1) \bmod \text{Length}(array)$
4. $right = (midpoint + 1) \bmod \text{Length}(array)$

5. If $array[midpoint] > array[left]$ and $array[midpoint] > array[right]$
 - A. Return $midpoint$
6. If $array[midpoint] > array[left]$
 - A. Return $MountainIndex(array, midpoint, end)$
7. Return $MountainIndex(array, start, midpoint)$

Runtime Analysis $O(\log n)$ due to recurrence relation $\approx S(n) = S(n/2) + c, S(1) = C$

3 Exercise 1.3.3

Recursive Algorithm The heuristic is to follow a downslope or upslope until the local minimum. If we follow the slope, we either meet a boundary condition, or we must meet a point at which the slope turns.

Assuming the array has a non-zero length.

LocalMinimum(array, start, length)

1. If $length == 1$
 - A. Return $start$
2. If $array[start] < array[start + 1]$
 - A. Return $start$
3. If $array[start + length - 1] < array[start + length - 2]$
 - A. Return $start + length - 1$
4. return $Search(array, start, length)$ // into recursion

Search(array, start, length)

1. $midpoint = start + length/2$
2. $array[midpoint] < array[midpoint - 1]$ and $array[midpoint] < array[midpoint + 1]$
 - A. Return $midpoint$
3. If $array[midpoint] > array[midpoint + 1]$ // downslopes to right
 - A. Return $Search(array, midpoint, length/2)$
4. Return $Search(array, start, length/2)$ // upslopes to right

Runtime Analysis $O(\log n)$ due to recurrence relation $\approx S(n) = S(n/2) + c, S(1) = C$

4 Exercise 1.3.4