1D Peak Finding

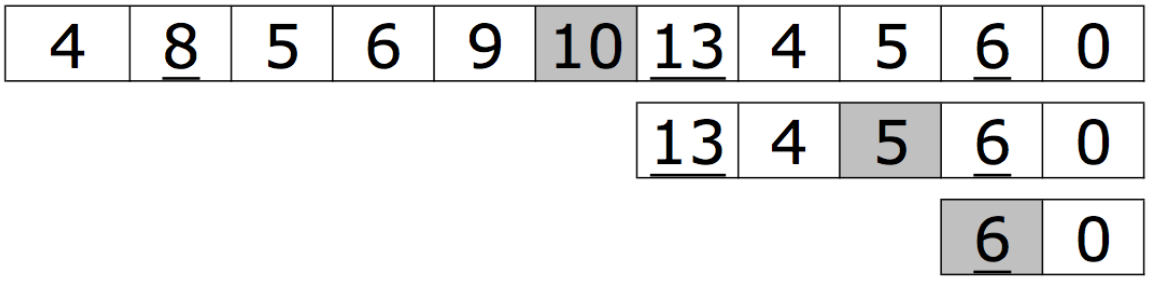
Objective

Given an array A with n elements, find the index i of the peak element $A[i]$ where $A[i] \geq A[i-1]$ and $A[i] \geq A[i+1]$. For elements on the boundaries of the array, the element only needs to be greater than or equal to its lone neighbor to be considered a peak. Or, say $A[-1] = A[n] = \infty$.

Algorithm

Given an array A with n elements:

- Take the middle element of A, $A[\frac{n}{2}]$, and compare that element to its neighbors
- If the middle element is greater than or equal to its neighbors, then by definition, that element is a peak element. Return its index $\frac{n}{2}$.
- Else, if the element to the left is greater than the middle element, then recurse and use this algorithm on the left half of the array, not including the middle element.
- Else, the element to the right must be greater than the middle element. Recurse and use this algorithm on the right half of the array, not including the middle element.



Runtime Analysis

When we recurse, we reduce size n array into size $\frac{n}{2}$ array in $O(1)$ time (comparison of middle element to neighbors). Show recursion in the form of "Runtime of original problem" ="Runtime of reduced problem" + "Time taken to reduce problem". Then use substitution to keep reducing the recursion.

$$T(n) = T(\frac{n}{2}) + c \tag{1}$$
$$T(n) = T(\frac{n}{4}) + c + c \tag{2}$$
$$T(n) = T(\frac{n}{8}) + c + c + c \tag{3}$$
$$T(n) = T(\frac{n}{2^k}) + ck \tag{4}$$
$$\text{Substitute } k = \log_2 n \tag{5}$$
$$T(n) = T(\frac{n}{2^{\log_2 n}}) + c\log_2 n \tag{6}$$
$$= T(1) + c\log_2 n \tag{7}$$
$$= O(\log n) \tag{8}$$

2D Peak Finding

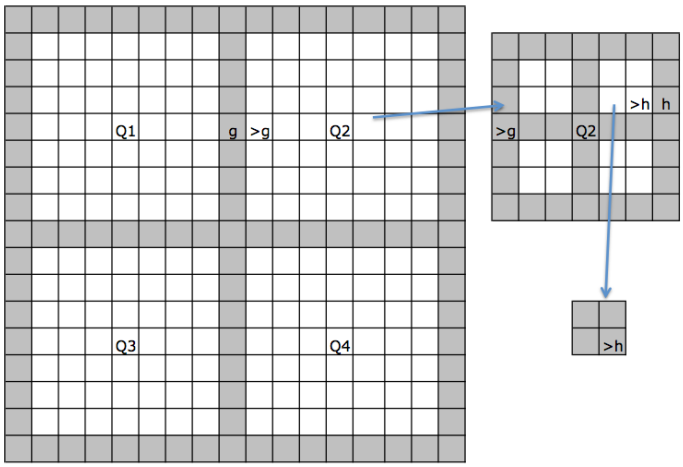
Objective

Given an nxn matrix M, find the indices of a peak element $M[i][j]$ where the element is greater than or equal to its neighbors, $M[i+1][j], M[i-1][j], M[i][j+1], andM[i][j-1]$. For elements on the boundaries of the matrix, the element only needs to be greater than or equal to the neighbors it has to be considered a peak.

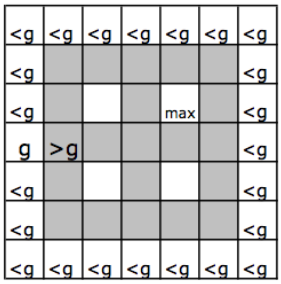
Algorithm

Given an nxn matrix M:

- Take the "window frame" formed by the first, middle, and last row, and first, middle, and last column. Find a maximum element of these $6n$ elements, g = $M[i][j]$.
- If g is greater than or equal to its neighbors, then by definition, that element is a peak element. Return its indices (i, j).
- Else, there's an element that neighbors g that is greater than g. Note that this element can't be on the window frame since g is the maximum element on the window frame, thus this element must be in one of the four quadrants. Recurse and use this algorithm on the matrix formed by that quadrant (not including any part of the window frame)



Proof of Correctness



Claim 1: If you recurse on a quadrant, there is indeed a global peak in that quadrant.

Proof: The quadrant we selected contains an element larger than g. Thus we know that the maximum element in this quadrant must also be larger than g. Since g is the maximum element surrounding this quadrant, the maximum element in this quadrant must be larger than any element surrounding this quadrant. This element must be greater than or equal to all of its neighbors since it is greater than all elements within the quadrant and directly outside of the quadrant, so the maximum element in this quadrant must be a global peak.

Claim 2: If you find a peak on the submatrix, then that peak is a global peak.

Proof: The window frame of the submatrix contains an element larger than g. Say m is the maximum element on the window frame. Since g is the largest element directly surrounding the submatrix, that means m is larger than all the elements surrounding the submatrix. If m is a peak in the submatrix and m is on the boundary, m must be a global peak since it is guaranteed that m is greater than any neighbors outside the scope of the submatrix. If m is a peak in the submatrix and m is not on the boundary, then clearly m is greater than or equal to its four neighbors and thus is a global peak.

Claim 3: You will always find a peak on the submatrix

Proof: In the case that we don't find a peak on the window frame of a matrix, we recurse to try to find a peak in a strictly smaller matrix. Eventually, if you keep not finding a peak, you will recurse into a small enough matrix such that the window frame covers the entire matrix (i.e. if the number of rows and columns are both 3 or below). By claim 1, there is indeed a global peak in this matrix if we recursed down to it. Since we're examining the entire matrix, we must find that global peak.

By claim 2 and claim 3, using this algorithm, we will always find a peak and that peak will be a global peak.

Runtime Analysis

When we recurse, we reduce nxn matrix into $\frac{n}{2}$x$\frac{n}{2}$ matrix in $O(n)$ time (finding the maximum of $6n$ elements). Show recursion in the form of "Runtime of original problem" ="Runtime of reduced problem" + "Time taken to reduce problem". Then use substitution to keep reducing the recursion.

$$T(n) = T(\frac{n}{2}) + cn \tag{9}$$

$$T(n) = T(\frac{n}{4}) + c\frac{n}{2} + cn \tag{10}$$

$$T(n) = T(\frac{n}{8}) + c\frac{n}{4} + c\frac{n}{2} + cn \tag{11}$$

$$T(n) = T(1) + cn(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8}...) \tag{12}$$

$$= O(n) \tag{13}$$