**PEAK FINDER**

One-dimensional version

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\underline{a} & b & c & d & e & f & g & h & i
\end{array}
\]  \(a-i\) are numbers

Position 2 is a peak if and only if \( b > a \) and \( b > c \)

Position 9 is a peak if \( i > h \)

Problem: Find a peak if it exists.

**STRAIGHTFORWARD ALGORITHM**

Start from left

\[
\begin{array}{cccccccccc}
1 & 2 & \cdots & n/2 & \cdots & N/2 & N
\end{array}
\]

\[\uparrow\] might be peak

Look at \( n/2 \) elements

Could look at \( n \) elements

\(\Theta(n)\) complexity worst case

What if we start in the middle?  

Look at \( n/2 \) elements
Can we do better?

1 2 ... $\lfloor n/2 \rfloor$-1 $\lfloor n/2 \rfloor$ $\lfloor n/2 \rfloor$+1 ... $n$-1 $n$

Divide & Conquer

Look at $n/2$ position

If $a[\lfloor n/2 \rfloor] < a[\lfloor n/2 \rfloor - 1]$ then only look at left half 1 .. $\lfloor n/2 \rfloor$ - 1 to look for peak

Else if $a[\lfloor n/2 \rfloor] < a[\lfloor n/2 \rfloor + 1]$ then only look at right half $\lfloor n/2 \rfloor$+1 .. $n$ to look for peak,

Else $n/2$ position is a peak.

Why? $a[\lfloor n/2 \rfloor] > a[\lfloor n/2 \rfloor - 1]$

$a[\lfloor n/2 \rfloor] > a[\lfloor n/2 \rfloor + 1]$

What is the complexity?

$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + \Theta(1) & \text{if } \log_2 n \text{ times} \\ \Theta(\log_2 n) & \text{else} \end{cases}$

$T(n) = \Theta(\log_2 n)$

$n = 1,000,000$

$\Theta(n)$ algo 13.5 s in python impl

$\Theta(\log n)$ algo 0.001 s

Argue that the algorithm is correct
2-Dimensional Version

$\begin{array}{ccc}
  c \\
  b & a & d \\
  e 
\end{array}$

$a$ is 2D peak iff $a \geq b, a \geq d, a \geq c, a \geq e$

Greedy ascent algorithm

$\Theta(n^2)$ algorithm

$\begin{array}{cccc}
  14 & 13 & 12 \\
  15 & 9 & 11 & 17 \\
  16 & 17 & 19 & 20 
\end{array}$

O peak

Extend 1D divide & conquer to 2D: Attempt #1

Pick middle column $j = \lceil n/2 \rceil$
Find a 1D peak at $i,j$
Use $(i,j)$ as a start point on row $i$ to find 1D-peak on row $i$
Problem: 2D peak may not exist on row i

end up with 14

which is not a 2D peak

ATTEMPT # 2

Pick middle column \( j = \frac{n}{2} \)

Find global maximum on column \( j \) at \((i,j)\)

Compare \((i,j-1), (i,j), (i,j+1)\)

Pick left cols if \((i,j-1) > (i,j)\)

(Similarly for right)

Solve the new problem with half the number of columns

When you have a single column, find global maximum and you're done
Example of Attempt #2

\[
\begin{bmatrix}
10 & 8 & 10 & 10 \\
14 & 13 & 12 & 11 \\
15 & 9 & 11 & 21 \\
16 & 17 & 19 & 20 \\
\end{bmatrix}
\]

pick this column

17 global maximum for column
go with

\[
\begin{bmatrix}
10 & 10 \\
12 & 11 \\
11 & 21 \\
19 & 20 \\
\end{bmatrix}
\]
pick this column

\[
\begin{bmatrix}
10 \\
11 \\
21 \\
20 \\
\end{bmatrix}
\]
find 21

Complexity of Attempt #2

\(h, \text{ rows, } m, \text{ columns}\)

\[T(h, n) = T(n, \frac{n}{2}) + \Theta(n)\]

to find global maximum on a column (n rows)

\[T(n, n) = \Theta(n) + \ldots + \Theta(n)\]

\[= \frac{\Theta(n \log m)}{\log m} = \Theta(n \log n)\]

\(\text{if } m = n\)
Find maximum of rows 1, n/2, n & cols 1, n/2, n
6n numbers $\to \Theta(n)$ time
Call it $g$.

Check if $g$ is 2D-peak. If so, return it.
If not, pick a quadrant which contains a number strictly bigger than $g$, and recurse on only that quadrant.

**WHY IT WORKS.**

- if $h > g$ pick marked quadrant
- $h$ is greater than all numbers in the boundary of the chosen quadrant!
  Therefore, we will find a 2D-peak in the smaller-sized quadrant.
Since we are recursing $\log h$ times, one might think that the complexity of the algo is $\Theta(n \log n)$.

Upon closer inspection...

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$$

$n \times n$ size problem

of size $n/2 \times n/2$

$= \Theta(n)$

because $n + n/2 + n/4 \ldots + 1 \leq 2n$