PEAK FINDER

One-dimensional version

1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i

a - i are numbers

Position 2 is a peak if and only if $b \geq a$ and $b \geq c$

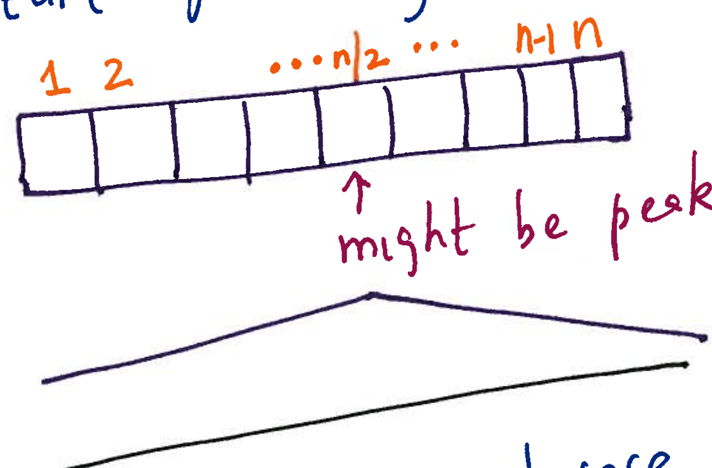
Position 9 is a peak if $i \geq h$

Problem: Find <u>a</u> peak if it exists.

6	7	4	3	2	1	4	5
	↑						↑

STRAIGHTFORWARD ALGORITHM

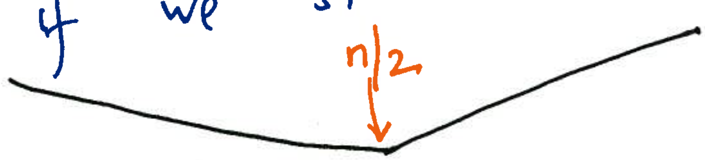
Start from left



Look at n/2 elements
Could look at n elements

$\Theta(n)$ complexity worst case

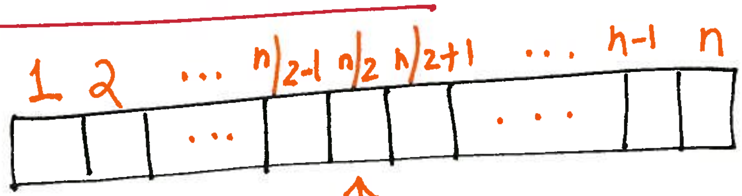
What if we start in the middle?



Look at n/2 elements

Can we do better?

Divide & Conquer



Look at $n/2$ position

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

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

Else $n/2$ position is a peak

WHY?

$$a[n/2] \geq a[n/2 - 1]$$
$$a[n/2] \geq a[n/2 + 1]$$

What is the complexity?

$$T(n) = T(n/2) + \Theta(1)$$

To compare $a[n/2]$ to neighbors

$$= \Theta(1) + \dots + \Theta(1) \quad (\log_2 n \text{ times})$$

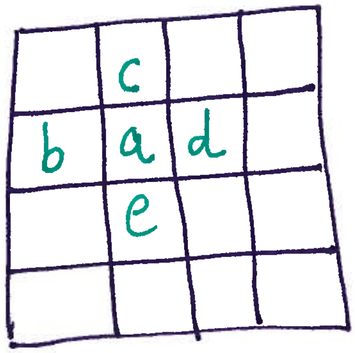
$$T(n) = \Theta(\log_2 n)$$

$n = 1{,}000{,}000$ $\quad \Theta(n)$ algo $\quad$ 13 s in python impl

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

Argue that the algorithm is correct

2-Dimensional Version

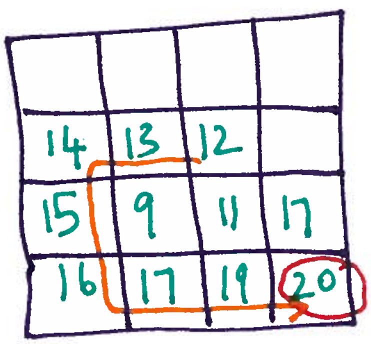


a is 2D peak iff

$a \geq b$, $a \geq d$, $a \geq c$, $a \geq e$

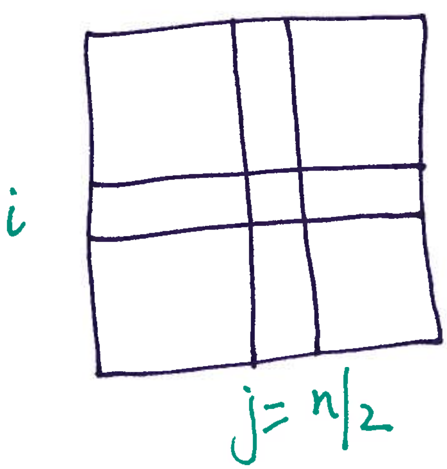
Greedy ascent algorithm

$\Theta(n^2)$ algorithm



peak

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



Pick middle column $j = n/2$

Find a 1D peak at i, j

Use (i, j) as a start point on row i to find 1D-peak on row i

Attempt #1 Fails

Problem: 2D peak may not exist on row i

		10	
14	13	12	
15	9	11	
16	17	19	20

end up with 14 which is not a 2D peak

Attempt #2

Pick middle column $j = n/2$

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

10	8	10	10
14	13	12	11
15	9	11	21
16	17	19	20

pick this column

17 global maximum for column

go with

10	10
12	11
11	21
19	20

pick this column

10
11
21
20

find 21

COMPLEXITY OF ATTEMPT #2

n rows, m columns

$$T(n, n) = T(n, n/2) + \Theta(n)$$

to find global maximum on a column (n rows)

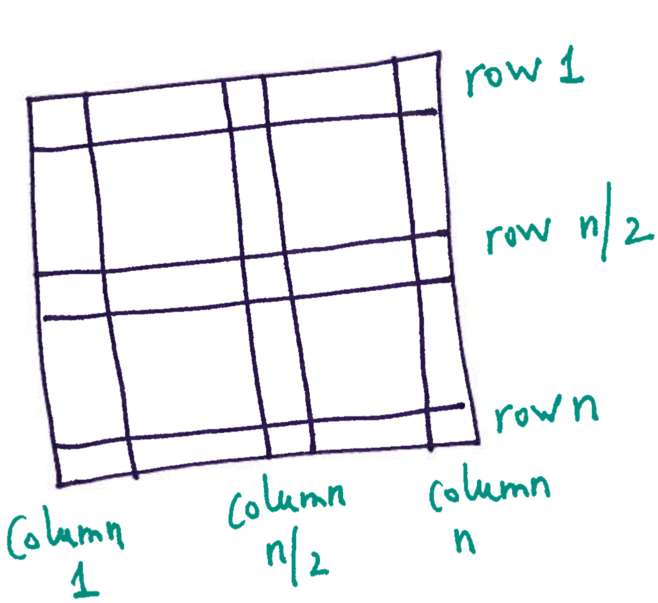
$$T(n, n) = \underbrace{\Theta(n) + \ldots \Theta(n)}_{\log m}$$

$$= \Theta(n \log m)$$

$$= \Theta(n \log n)$$

if $m = n$

CAN WE DO BETTER? ATTEMPT #3



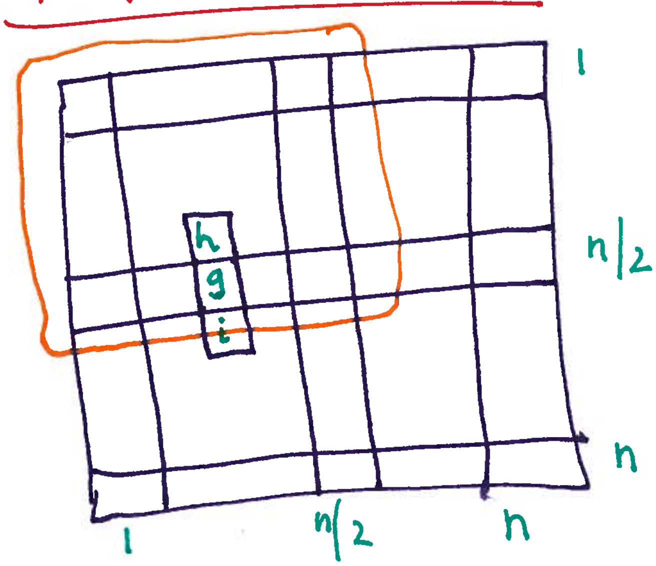
Find maximum of rows 1, n/2, n & cols 1, n/2, n

6n numbers → $\Theta(n)$ time

Call it g.

Check if g is 2D-peek. If so, return it. If not, pick a quadrent which contains a number strictly bigger then g, and recurse on only that quadrent.

WHY IT WORKS.



if h > g pick marked quadrent

h is greater then all numbers in the boundary of the chosen quadrent!

Therefore, we will find a 2D-peek in the smaller-sized quadrent.

COMPLEXITY

Since we are recursing $\log n$ times, one might think that the complexity of algo is $\Theta(n \log n)$

Upon closer inspection . . .

$$T(n) = T(n/2) + \Theta(n)$$

$T(n)$: $n \times n$ size problem

$T(n/2)$: quadrant is of size $n/2 \times n/2$

$\Theta(n)$: To compute maximum of boundary & middle rows and columns

$$= \Theta(n)$$

because $n + n/2 + n/4 \cdots + 1$

$$\leq 2n$$