Def.: in the array $A[1:n]$ the element $x$ is a majority element, if more than half of the elements are $x$-es

$A[i] = x$ for more than half of $i \in \{1, 2, \dots, n\}$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 4 | 3 | 1 |

1 is not majority element

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 5 | 1 |

1 is a majority element

Question: is there a majority element in $A$?

If the elements are from $\mathbb{N}$ (or $\mathbb{R}$)

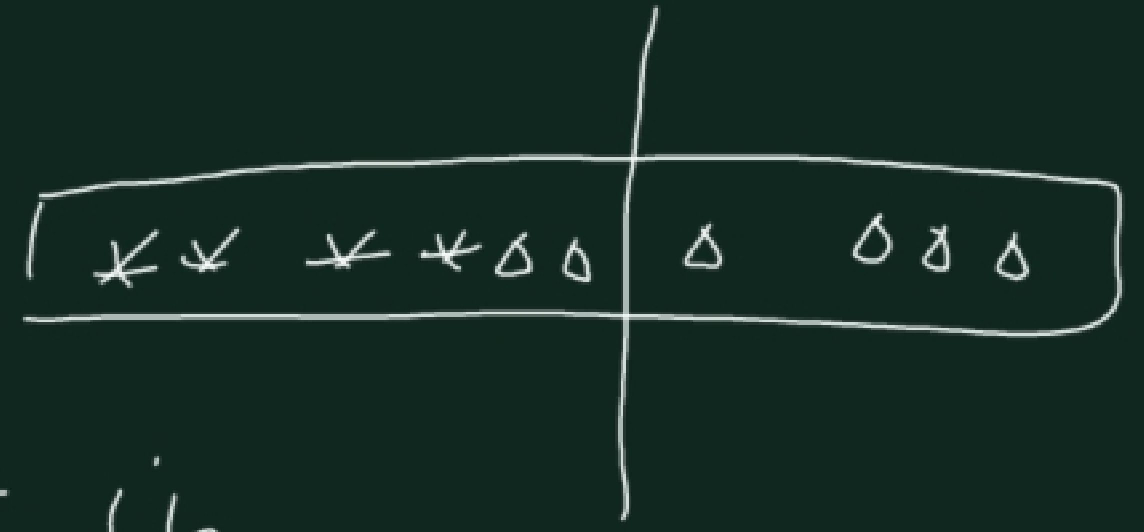
then easy: sort + count

$O(n \log n)$ $\quad$ $O(n)$

If the elements are from a set where no sorting is available $\rightarrow$ harder.

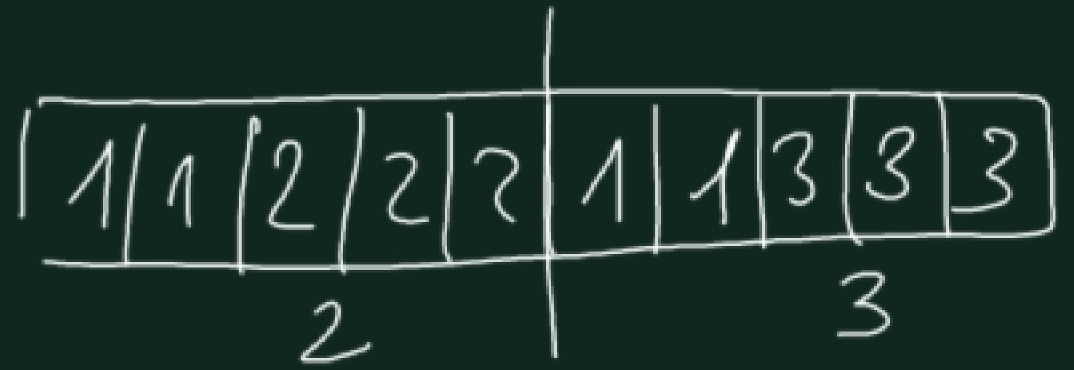
We can check if $A[i] = A[j]$ or not.

Crucial observation:



if $x$ is a majority element in $A[1:n]$ then $x$ is a majority element in one of $A[1:\underbrace{\lfloor \frac{n+1}{2} \rfloor}_{q}]$ or $A[\lfloor \frac{n+1}{2} \rfloor + 1 : n]$

$$q := \left\lfloor \frac{n+1}{2} \right\rfloor$$




$\lightning$ if $x$ is not a maj. element in any of the subarrays then the # of $x$-es in $A[1:n]$ is $\leq \frac{\cancel{q}}{2} + \frac{n-\cancel{q}}{2} = \frac{n}{2}$ $\lightning$ $x$ was a maj. element in $A[1:n]$

# Divide & conquer algorithm

1. If $A$ has 1 element $\Rightarrow \checkmark$ it is a majority.

If $A$ has $\geq 2$ elements

$$A[p:r] \qquad r \geq p+1$$

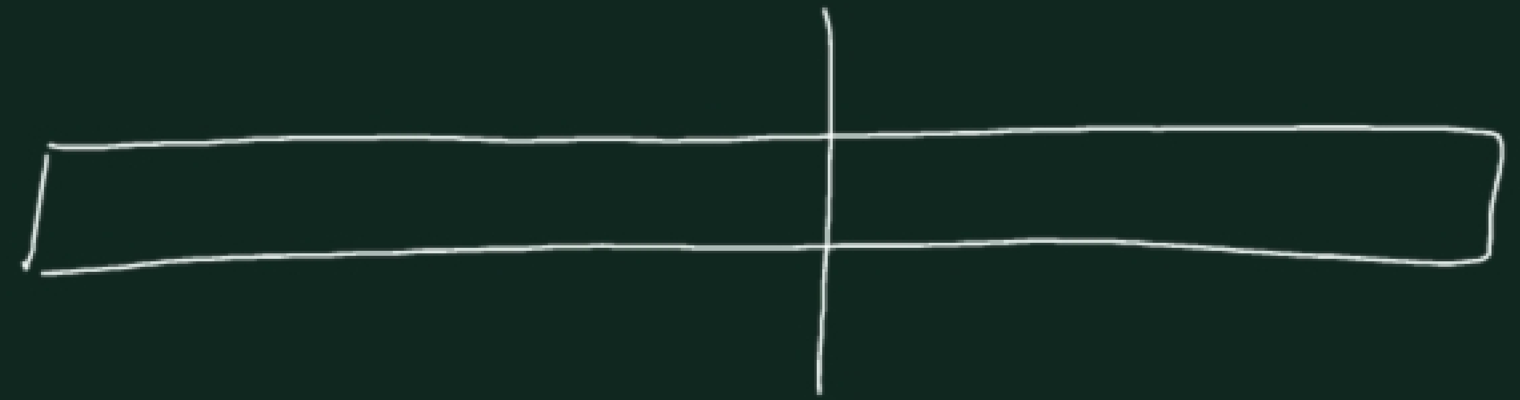
then divide $A$ into 2 subarrays

$q := \left\lfloor \frac{p+r}{2} \right\rfloor$ and

2. recursively solve

the problem for $A[p:q]$ and $A[q+1:r]$

③ Combine answers from the recursive calls



the answer for $A[p:r]$

$\Theta(1)$

4 cases:

- no | no ⟶ $\boxed{\text{no}}$
- yes, "x" | no ⟶ count the # of "x"-es in $A[p:r]$
  - 2 cases:
    - $> \frac{r-p}{2} \Longrightarrow \boxed{\text{yes, "x"}}$
    - $\leq \frac{r-p}{2} \Longrightarrow \boxed{\text{no}}$
- no | yes "y" ⟶ same with "y"
- yes "x" | yes "y" ⟶ same with <u>both</u> "x" and "y"

$\Theta(n)$

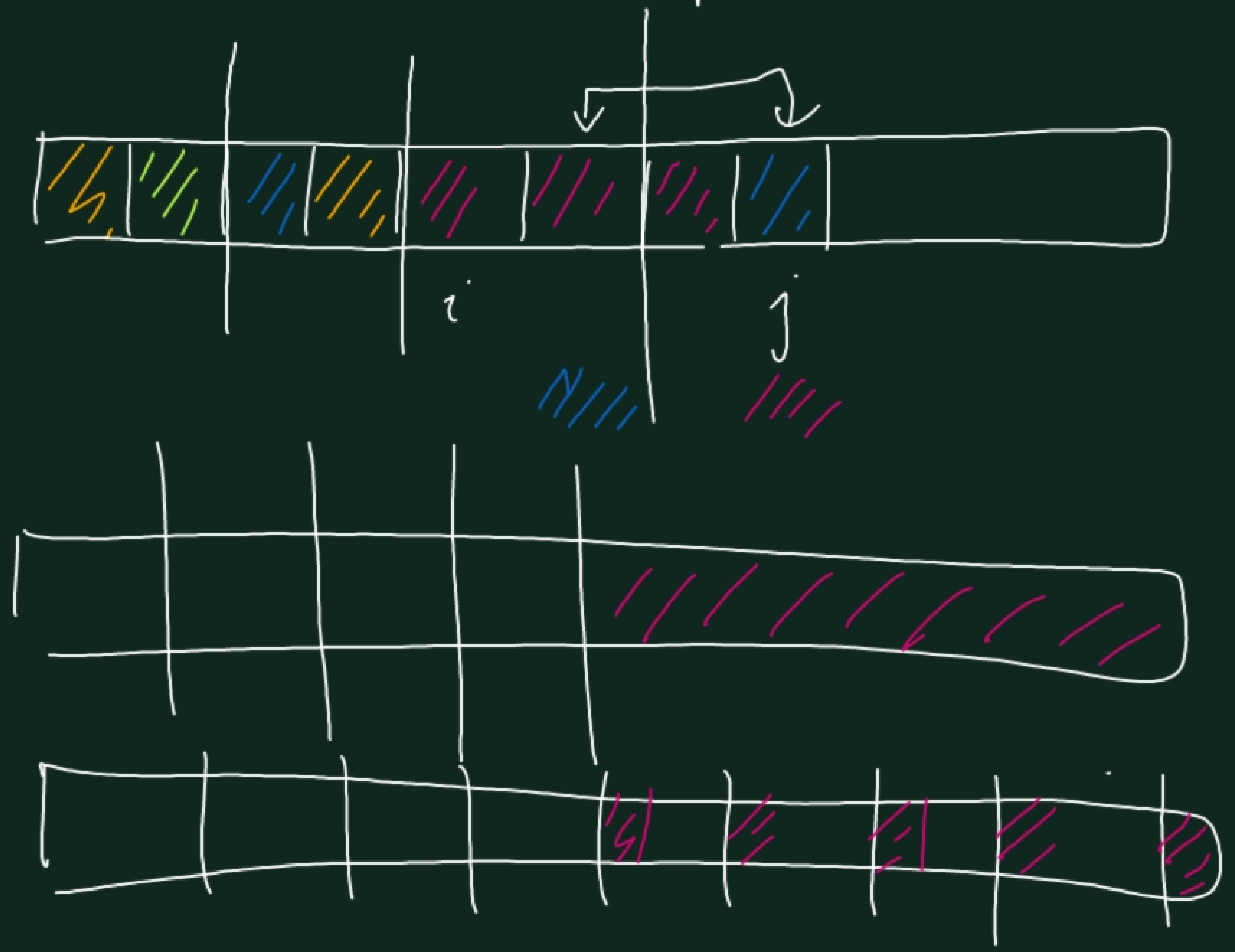
$T(n)$: time complexity for $A[1:n]$

$$T(n) = \underbrace{2 \cdot T\left(\frac{n}{2}\right)}_{\text{2 recursive calls}} + \underbrace{\Theta(n)}_{\text{final step}}$$

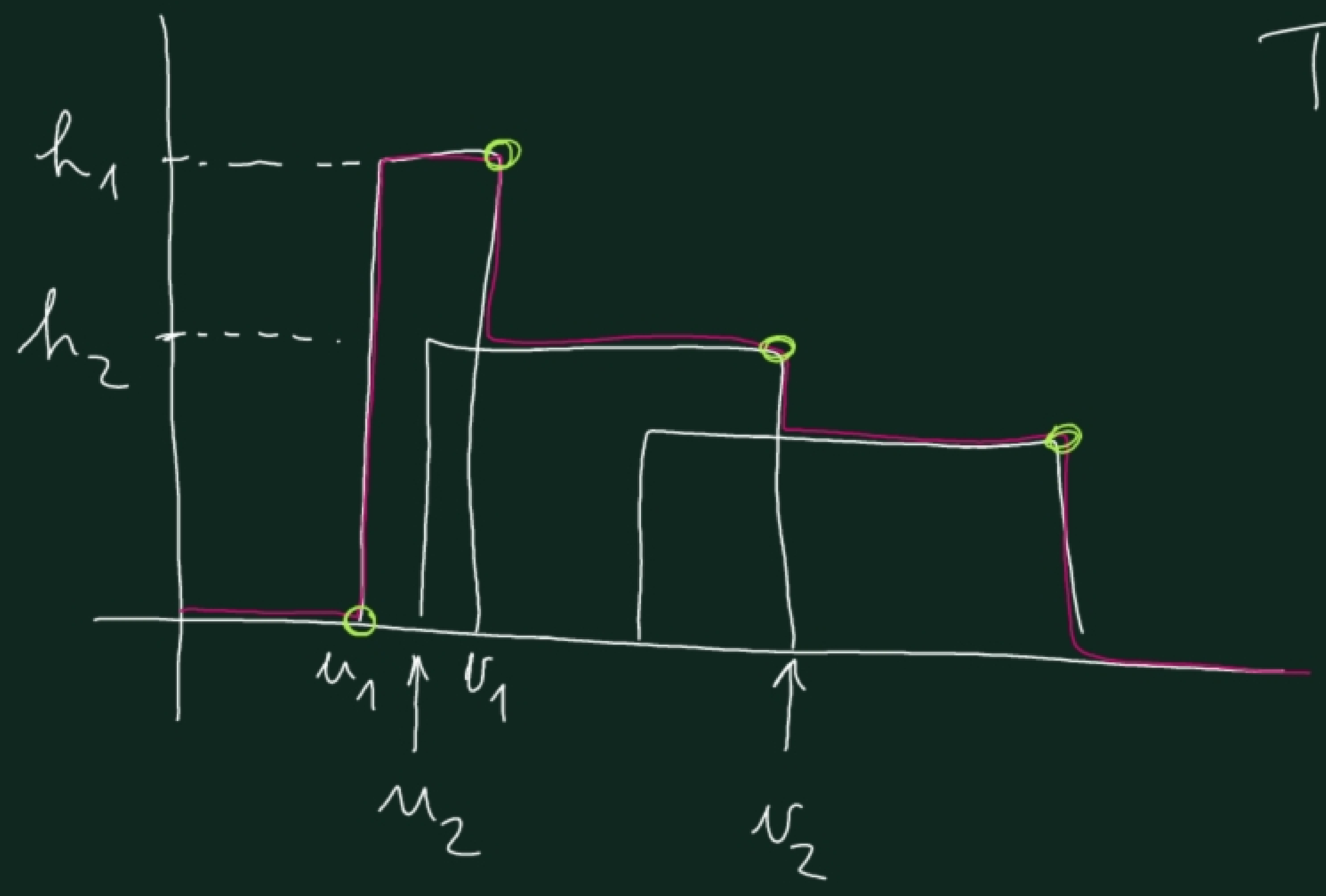
$$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

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

There is a faster algorithm which is NOT Divide & conquer: $\Theta(n)$



Skyline problem:




given $(u_i, v_i, h_i)$ $\quad i = 1, 2, \dots, n$

Task: compute green points

Compute for

$i = 1, \dots, \lfloor \frac{n}{2} \rfloor$ and

$i = \lfloor \frac{n}{2} \rfloor + 1, \dots, n$

and MERGE

## Problem:

Largest jump / leap : We have an array $A[1:n]$ of integer numbers. Find the index pair $1 \le i \le j \le n$ such that

$$A[j] - A[i] \text{ is maximal.}$$

| | | | | |
|---|---|---|---|---|
| 5 | 4 | 3 | 2 | 1 |

← here $i=j$ is the best

Divide & conquer algorithm to solve this for $A[p:r]$

1. If $p=r \implies (i,i)$ gives the largest leap : $0$

   If $p+1 \leq r \implies q := \lfloor \frac{p+r}{2} \rfloor$

   divide $A[p:r]$ into 2 subarrays
   $A[p:q]$ and $A[q+1:r]$.

2. Recursively solve the problem for these subarrays

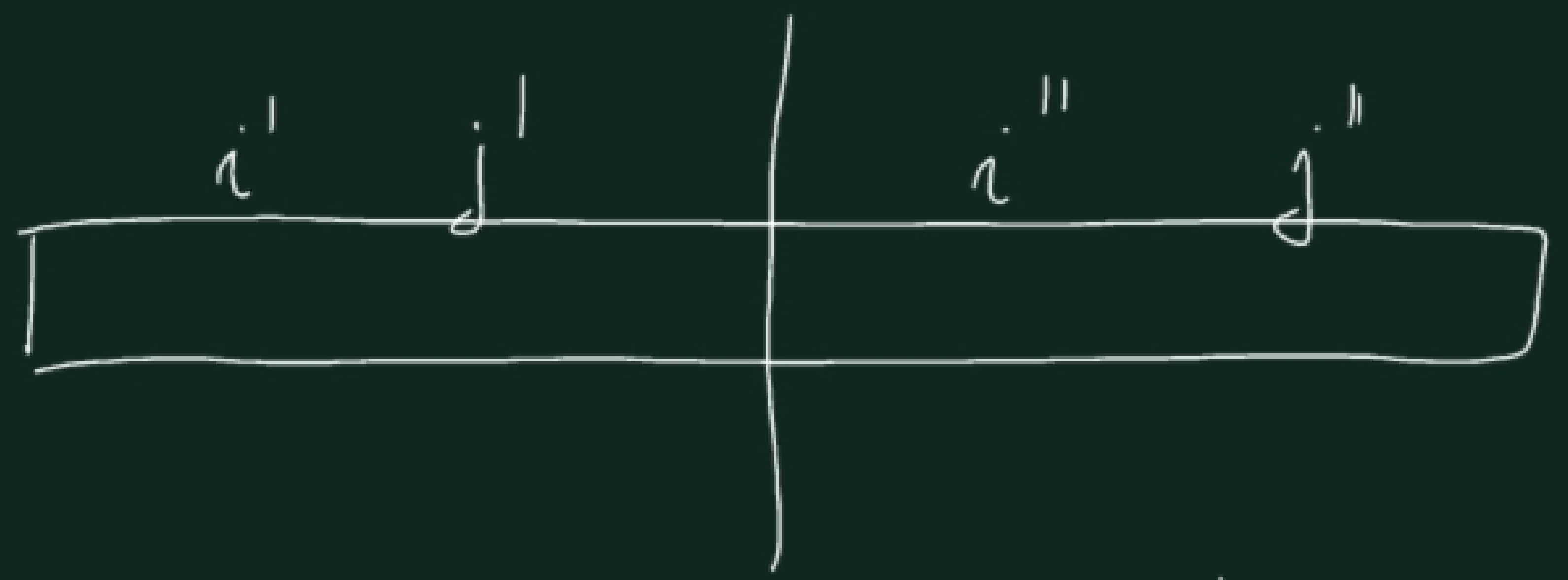
   if the answers are $i'$ and $j'$ for $A[p:q]$
   and $i''$ and $j''$ for $A[q+1:r]$

Observation: if $i \le j$ gives $\overset{\text{a}}{\text{the}}$ maximal leap in $A[p:r]$ then there are 3 cases

Case 1: $p \le i \le j \le q$, then $i = i'$ and $j = j'$

Case 2: $q+1 \le i \le j \le r$, then $i = i'$ and $j = j''$

Case 3: $p \le i \le q < j \le r$ ?




In case 3 we have $i''' := \underset{p \le s \le q}{\text{argMin}} \left(A[s]\right)$

$$j''' := \underset{q+1 \le t \le r}{\text{argMax}} \left(A[t]\right)$$

$$(i, j) := (i''', j''')$$

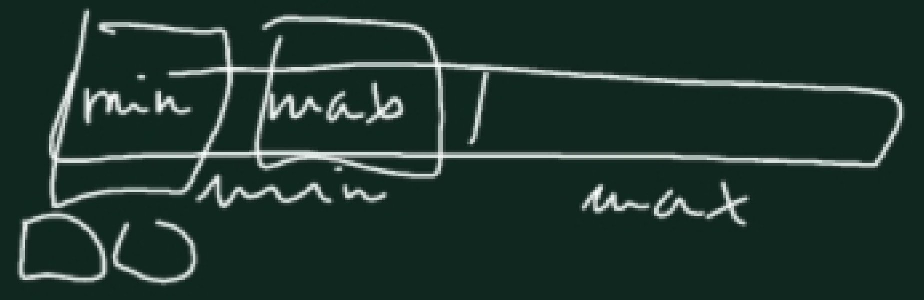
$\Theta(q-p)$

$\Theta(r-p)$ size of the array

$\Theta(r-q)$

We don't know which case we are in, so we check all 3 cases:

$$
\begin{array}{l}
i', j' \\
i'', j'' \\
i''', j'''
\end{array}
\qquad
\text{Max}\left\{
\begin{array}{l}
A[j'] - A[i'] \\
A[j''] - A[i''] \\
A[j'''] - A[i''']
\end{array}
\right\}
\begin{array}{l}
\longrightarrow \quad i := i' \quad j := j' \\
\xrightarrow{?} \quad i := i'' \quad j := j'' \\
\longrightarrow \quad i := i'' \quad j := j'''
\end{array}
$$




$$T(n) = 2T\left(\frac{n}{2}\right) + \underbrace{\boxed{\Theta(n)}}_{\text{counting min \& max}}$$

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

There is a divide & conquer alg. Which is faster.

It will have $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$

Idea: the algorithm provides the min and max elements' indeces from array A.

$$\text{LargestLeapMinMax}(A, p, r): \underbrace{i, j, \min, \max}_{\text{4 indeces as output}}$$

LLMM

if $p = r$ then RETURN$(p, p, p, p)$

$r \geq p+1$ then $q := \left\lfloor \frac{p+r}{2} \right\rfloor$ & recursive calls

$(i', j', \min', \max') := \text{LLMM}(A, p, q)$

$(i'', j'', \min'', \max'') := \text{LLMM}(A, q+1, r)$

Continued on next page

continued

find which is the largest

$$\left.\begin{array}{l} \quad A[j']-A[i'] \\ \text{or } A[j'']-A[i''] \\ \text{or } A[\max'']-A[\min'] \end{array}\right\} \text{ and set } i \text{ and } j \text{ to be these indeces}$$

also

$$\min := \arg\operatorname{Min}\{A[\min'], A[\min'']\}$$

$$\max := \arg\operatorname{Max}\{A[\max'], A[\max'']\}$$

RETURN $(i, j, \min, \max)$

This is all $\Theta(1)$ time compl.

So all together this new version has

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$$

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

Master Theorem

or

we know from a few weeks ago

Problem: array $A[1:n]$ with elements $\in \mathbb{Z}$ (integer)

we are looking for indeces $1 \leq i \leq j \leq n$

such that

$$A[i] + A[i+1] + \ldots + A[j] \text{ is maximal}$$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -1 | 3 | 2 | -5 | 4 | 2 | -2 | 1 |