18 Sept 202

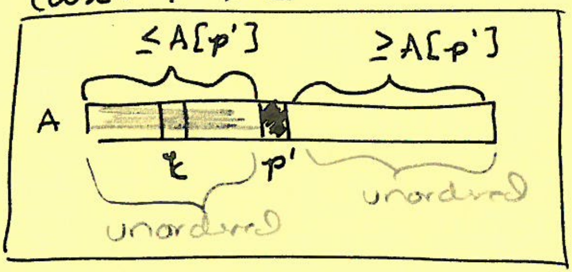
Quick Select

input: ① An array A of length n, ~~sorted~~ $\boxed{\text{not}}$ sorted from smallest A to largest $A[n]$
② An index $k \in \underline{n} = \{1, 2, \ldots, n\}$

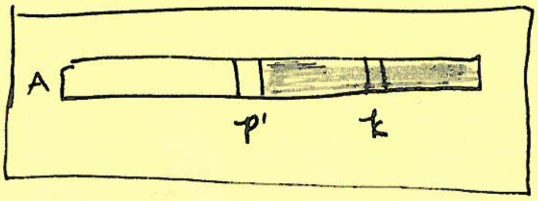
output: The k^{th} smallest element of A

```
QUICK SELECT (A[1...n], k)
  if n = 1          // k ∈ {1} => k = 1      } base case,
  | return A[1]                              } handled directly
  endif
  Choose a pivot p ∈ n.   // can be random, maybe not
  p' ← PARTITION (A, p)
  if p' = k
  | return A[p']
  else if  p' > k
  | return QUICKSELECT (A[1...p'-1], k)
  else  // p' < k
  | return QUICK SELECT (A[p'+1, ..., n], k-p')
  end else
```

case: $p' > k$



case: $p' < k$



Exercise: In groups, do the following:
① Create an array of length 15
② Follow this algorithm, diving into the recursions (today, the recursion fairy is napping)
③ What is the worst-case runtime? Why?

PARTITION (A, k)

→ input: Array A of length n
index $k \in \underline{n}$

→ output: ~~rearrange A AS SUCH~~
rearranges A so that
① all values $\leq A^{in}[k]$ ~~are now~~
are to the "left" of $A[k]$
② all values $\geq A^{in}[k]$ are to the "right" of $A[k]$
③ returns new index of $A[k]$.

ex: $[\pi, 6, 30, 18, 2.5]$, $k = 2$

1, 2, 3, 4, 5

new A: $[\underbrace{\pi, 2.5}_{\leq A[k_{in}]}, \underbrace{\boxed{6}}_{A[k]_{input}}, \underbrace{30, 18}_{\geq A^{input}[k]}]$

new index is 3

Worst-case runtime:

Let $T(n)$ denote the worst-case RT on input A of size n.

$$T(n) = \begin{cases} \overbrace{\Theta(1)}^{\text{end at line 2.}}, & n = 1 \\ \underbrace{\Theta(1)}_{\text{Lin 1 check}} + \underbrace{\Theta(1)}_{\text{Line 4, choose pivot}} + \underbrace{\Theta(n)}_{\text{Line 5}} + \max \begin{cases} \Theta(1) & \text{Lin} \\ T(p'-1) \\ T(n-p \end{cases} \end{cases}$$

there's still a choice of p

Let's simplify ~~realize worst p' choice~~

$$T(n) = \Theta(n) + \max\{T(p'-1), T(n-p')\}$$

Choose worst p':

$$T(n) = \Theta(n) + \max_{p' \in \underline{n}} \max\{T(p'-1), T(n-p')\}$$

We've seen this before!

IN The worst case, just like QSort,

$$T(n) = \Theta(n) + T(n-1) \in \Theta(n^2)$$

But, this algo: ① sort A

② return $A[k]$

runs in $\Theta(n\log n)$ worst case time!

What if... we could always choose a pivot such that our recurrence relation becomes

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

where $b > 1$

(ie. guarantee we slice off a % of the input

Using Master's:

$$\boxed{\begin{array}{l} a = 1, \ b = b \\ f(n) \in \Theta(n) \end{array}} \Rightarrow \log_b a = \log_b 1 = 0$$

variables

compare $f(n)$ to $\theta(n^{\log_b a})$

ie, $\Theta(n) >$ to $\Theta(n^0) = \Theta(1)$

$\Rightarrow$ Let's try case 3

(a) Find ε such that

$$\Theta(n) \in \Omega(n^{0+\varepsilon})$$

$\varepsilon = 1$ works!

(b) $c = 0.75$ and $n_0 = 1$

By case 3, $T(n) \in \Theta(n)$