* K th largest Element in an Array

$O(n\log n)$, $O(1)$ → - sort and take k^{th} → noob

$O(n\log k)$, $O(k)$ → • Priority Queue & take k^{th} - only items are given in ~~prio~~ order when ~~popping~~, ~~Polling~~ polling, peeking

• Quick Select → relates to quick sort.

pivot

3 2 1 5 6 $\boxed{4}$
l r
i j

if j less than pevot, i++ & then swap i, j
else j++

$O(n^2)$ worst case

3 2 1 5 6 4 ⇒ 3 2 1 5 6 4 ⇒ 3 2 1 5 6 4

3 2 1 5 6 4 ⇒ 3 2 1 5 6 4

now swap pivot & (i+1)

$O(n)$ best case, $O(1)$

3 2 1 4 5 6

less than 4 → greater than 4

check pevot place from right & if == k, return value.

p k
l r

if place < k
l = p+1 place

else place > k
r = p-1 place

$\Theta(n)$ guaranteed

We can optimize this using a random pivot position from (l, r) range.

select random value & swap with r.
now also pivot is at the end.

~~More~~

$$T(n) = T(n/2) + (n-1)$$

$n \downarrow n/2 \downarrow n/4 \downarrow n/8$

$$T(n/2) = T(n/4) + (n/2 - 1)$$

$$T(1) = 0$$

$$T(n) = (n-1) + (n/2 - 1) + (n/4 - 1) + \cdots$$

tree depth

$$\sum_{i=0}^{(\log n)-1} \left(\frac{n}{2^i} - 1\right)$$

$$n\left(\sum_{i=0}^{(\log n)-1} \frac{1}{2^i}\right) - \sum_{i=0}^{(\log n)-1} 1$$

$$n\left(\sum_{i=0}^{\log n - 1} \left(\frac{1}{2}\right)^i\right) - \log(n)$$

$$n\left(\frac{1-\left(\frac{1}{2}\right)^{\log n}}{1-\frac{1}{2}}\right) - \log n$$

No : Date: / /

$$2n\left[1-\left(\frac{1}{2}\right)^{\log n}\right]-\log n$$

$$2n\left[1-\frac{1}{2^{\log n}}\right]-\log n$$

$$b^{\log_b K}=K$$

$$2^{\lg n}=n$$

$$\cancel{\log n}\ \lg 2^{\lg n}=\cancel{\lg}\ \lg n$$

$$\lg n \lg 2=\lg n$$

$$2n\left[1-\frac{1}{n}\right]-\log n$$

$$\frac{2n}{n}(n-1)-\log n$$

$$a^b=c$$

$$2(n-1)-\log n$$

$$\log_a c=b$$

leading complexity

$$\therefore O(n)$$

Scanned with CamScanner