Qn 9.1-1

CHAPTER 9

Second Smallest Element (A, n)

```
min = A[1]
~~Secondmin = A[2]~~
 if A[2] > A[1] then secondmin = A[2]
                else secondmin = A[1], min = A[2].
for i = 3 to n
    if ( A[i] > min )
        if( A[i] < secondmin)
            Secondmin = A[i]
    else
        min = A[i]
```

min ↓	smin ↓					
5	7	3	4	2	1	9

Find min using D&C and find smin using

min – linear time

D&C → $\underline{\log n}$

Smin → linear time. → $O(n)$.

$\Big\}$ $n + \lfloor \log n \rfloor - 2$ comparisons.

Pn 9.1 - 2

CHAPTER 9

MinMax(A) // Min & Max simultaneously in $\lfloor 3n/2 \rfloor$ comparisons.

```
min = A[1]
max = A[1]

for i = 1 to n
    if ( A[i] > A[i+1] )
        if ( A[i] > max )
            max = A[i]
        if ( A[i+1] < min )
            min = A[i+1]
    else
        if A[i] < min
            min = A[i+1]
        if A[i+1] > max
            max = A[i+1].
    i++;
```

3 comparisons.

$3(n/2)$ comparisons

for n = even : $3(n-2)/2 + 1$ comparisons.

$= 3(n/2) - 3 + 1$ comparisons

$= 3(n/2) - 2$ comparisons. $= 3\lceil(n/2)\rceil - 2$ comparisons

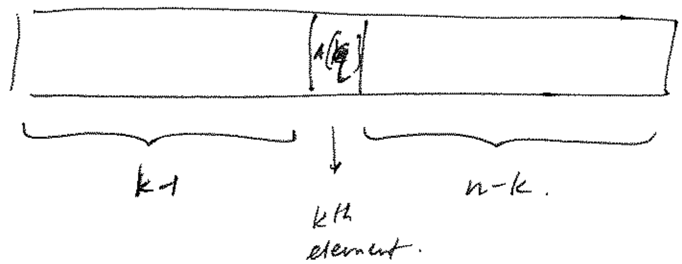
for n = odd : $3(n/2)$ comparisons

$= \lfloor 3(n+1/2) \rfloor$ comparisons. lower bound $= 3\lceil n/2 \rceil - 2$ comparisons.

Qn <u>9.2-2</u>

Indicator random variable X_k. } <u>independent</u>

$T(\max(k-1, n-k))$



$X_k = I\{$the subarray $A[p..q]$ has exactly k elements$\}$.

X_k = depends on an element, not the size of the array. = ~~0~~ <u>1 for the ~~ith~~ ~~···~~ k</u> and <u>0 for all others</u>.

$T(\max(k-1, n-k))$ = ~~O for k~~

$\max(T(k-1) + T(n-k))$

Qn 9.2-3

```
Randomized Select (A, p, r, k)     // Iterative version of random
                                            select.
~~while p!(p>r)~~
if p==r
    return A[p]

while (p > r)
    q = Randomised-partition (A, p, r)
    i = q - p + 1                  // pivot.
    if i == k
        return A[q]
    else if i < k
        p = p ; r = q - 1
    else
        p = q + 1 ; r = r
        k = i - k.
```

Qn 9.3-1

(*) Groups of 7.

$\Rightarrow$ $\frac{n}{7}$ medians.

$\Rightarrow$ For the ~~mead~~ median of median-of-medians.

$$4\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{7}\right\rceil\right\rceil - 2\right) = \frac{4n}{14} - 6.$$

elements are strictly smaller

elements are strictly larger.

$$\Rightarrow \quad T(n) = T\left(\frac{3n}{14}\right) + T\left(\frac{n}{7}\right) + O(n).$$

by subs^n $T(n) \le cn$.

$$T(n) \le \frac{4cn}{14} + \frac{cn}{7} + an.$$

$$= \frac{6cn}{14} + an.$$

$$= cn + \left(-\frac{8cn}{14} + an\right)$$

$$-\frac{8cn}{14} + an \le 0.$$

$$\Rightarrow \quad an \le \frac{8cn}{14}$$

$$\Rightarrow \quad c \ge \frac{14a}{8}.$$ $\Rightarrow$ linear time.

(*) Groups of 3

- divided into $\frac{n}{3}$ medians.

→ For the median of medians

$$2\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{3} \right\rceil \right\rceil - 2\right) = \frac{n}{\cancel{2}3} - 6. \quad \text{elements}$$

larger   smaller!

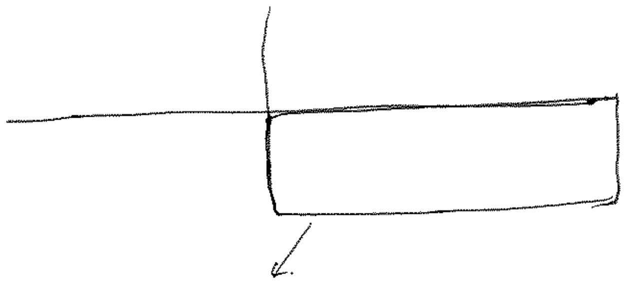
$$\Rightarrow T(n) = T(n/3) + T(n/3) + O(n).$$

$$= n\log_3 n > O(n) \Rightarrow \text{not } \underline{\text{linear time}}!$$

Qn 9.3-2

If $n \geq 140$

~~$T(n) \leq T(\lceil n/5 \rceil)$~~



$\rightarrow \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2 \right) * 3 - 1$

~~$\left(\frac{3*1}{2} \left\lceil \frac{n}{5} \right\rceil - 1 \right) = \frac{1}{2} * \frac{3n}{5} - 1$~~

~~$n = 140 \rightarrow \frac{3 * \overset{70}{140}}{2 * 5} - 1$~~

$n = 140 \Rightarrow \left(\left\lceil \frac{1}{2} \left\lceil \frac{140}{5} \right\rceil \right\rceil - 2 \right) * 3 - 1$

$= \left(\left\lceil \frac{1}{2} * 28 \right\rceil - 2 \right) * 3 - 1$

$= 12 * 3 - 1$

$= 35/140 = \boxed{1/4}$

$\rightarrow$ Hence $\approx \left\lceil \frac{n}{4} \right\rceil$

$= \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2 \right) * 3 - 1$

$= \left(\frac{n}{10} - 2 \right) * 3 - 1$

$= \frac{3(n-20)}{10} - 1 = \left(\frac{3n - 60}{10} - 1 \right) \approx \boxed{\frac{n}{4}}$ $n \leq 140$

Qn 9.3-5

Median subroutine → $\underline{A(n/2)}$ value.

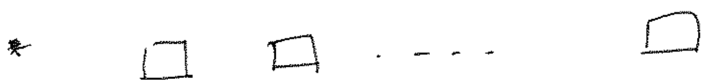
Arbitrary order statistic → $k \xrightarrow[\text{give}]{\text{should}}$ $\underline{A(k)}$ value

linear-time Algorithm?

InsertionSort( ).

↓

output ~~$\lceil k/5 \rceil$th element~~ $\cancel{\lceil k/5 \rceil}$ th element



$(n/5)$ $(k/5)$th elements

Q 9.3-6

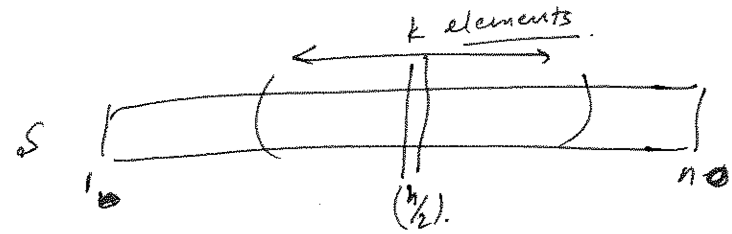
$m \longleftarrow \text{median}(A, p, r, n/2)$

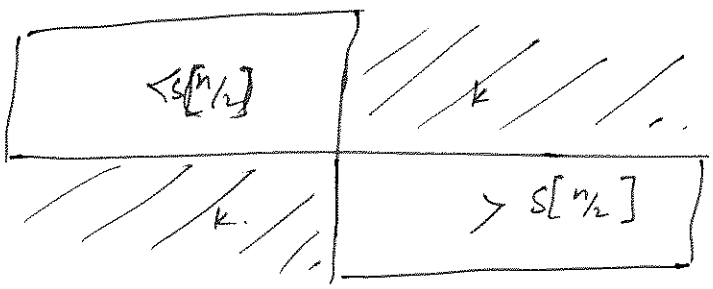
- ~~part~~ partition till we have k elements on ~~one~~ both sides.

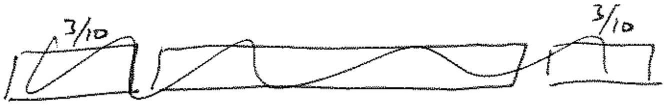
If k elements $\rightarrow$ return.

But $\rightarrow$ $O(n \log(n/k))$

Qn 9.3-7







PartitionTok(A, p, r, n/2, k)

q ← Pivot( ).

~~qq~~

if ~~k == n/4~~ $p - r + 1 == k$:

Return ~~$A[n/2 - k/2 \ldots n/2 \ldots n/2 + k/2]$~~

$A[p \cdots r]$

else if $q - p < n/2$:

partition.

else partition.

Qn 9-1

(a) Run merge-sort $\overset{O(n\log n)}{}$ and print $A[i]$ $\overset{O(1)}{}$

(b) Build Priority-queue $\rightarrow$ $O(n\log n)$ $\}$ $O(n\log n + i)$
Extract-Min i times $\rightarrow$ $O(i)$

(c) Find ith largest $\rightarrow$ $O(i)$. $\}$ $O(i\log i+n)$.
sort till i $\rightarrow$ $O(\cancel{x}\log i+n)$

Qn 9-2

(a) All get equal weights so the weighted median for $w = \frac{1}{n}$ for $i = 1 \ldots n$ will be the same as $w = 1$ for $i = 1 \ldots n$.

(b) Merge-Sort !
Stored weights separately and multiply by numbers while checking for sorting.

(c) Deterministic select !

(d) The weights are the distances and minimizing weights would minimizes distance, hence, it is the best solution possible.

(e) Weighted median !