Problem 3 ④.

$\star n = right - left + 1$

```
T quantile(T*a, int left, int right, int k){
  if(right <= left + 10){
    InsertionSort(a, left, right); — θ(n²) ①
    return a[k + left]; — θ(1) ①
  } else {
    int smallsize = (right - left + 1)/5; — θ(1) ①
    T*b = new T[smallsize]; — θ(1) ①
    for(int i = 0; i < smallsize; i++){
      b[i] = quantile(a, left + 5*i, left + 5*(i+1) - 1, 2); — T((left + 5*(i+1) - 1) - (left + 5*i) + 1) ②
    }                                                          = T((left + 5i + 5 - 1) - (left + 5i) + 1)
    T pivot = quantile(b, 0, smallsize - 1, smallsize/2); — = T(5)   Input size for each recursive call = 5 ③
    int p = linearSearch(a, left, right, pivot); — θ(n) ①
    a.swap(p, right); — θ(1) ①
    int m = partition(a, left, right); — θ(n) ①
    if(left + k == m){ return a[m]; } — θ(1) ①
    else if(left + k < m){ return quantile(a, left, m - 1, k); }
    else { return quantile(a, m + 1, right, k - (m + 1 - left)); }
  }
}
```

But

Since we know that (right <= left + 10) InsertionSort will always sort a constant # of elements for T(5). ∴ Each iteration that assigns a value to b[i] is constant θ(1). ∴ Constant-time sorting.

$T(smallsize - 1 - 0 + 1) = T(\frac{n}{5} - 1 + 1) = T(\frac{n}{5})$

Size of array: $\frac{n}{5}$

$\sum_{i=0}^{\frac{(right-left+1)}{5}} \theta(1) = \theta(1) * (\frac{n}{5}) \approx \theta(n)$ ②

④ - The for loop which assigns values to b[i] iterates for $\lfloor n/5 \rfloor$ times.

∴ Size of b is $\lfloor n/5 \rfloor \rightarrow \lfloor n/5 \rfloor$ medians.

- smallsize/2 = $\lfloor \frac{n}{5}(\frac{1}{2}) \rfloor = \lfloor n/10 \rfloor$ → # of medians smaller/larger than the median of medians.

∴ Each subgroup of 5 elements contain: 1 median, 2 elements smaller than median, and 2 elements larger than median.

∴ There are $\lfloor n/10 \rfloor$ medians smaller than the median of medians and $2\lfloor n/10 \rfloor$ elements that are smaller than the medians, hence smaller than the median of medians. (approx.)

Furthermore, within the subgroup containing the median of medians, we also have 2 elements smaller than the median of medians and 2 elements larger than the median of medians. (approx.)

∴ There are at least $\lfloor n/10 \rfloor + 2\lfloor n/10 \rfloor + 2 \approx 3n/10$ elements in a that are <= the pivot, the median of medians.

Scanned by CamScanner

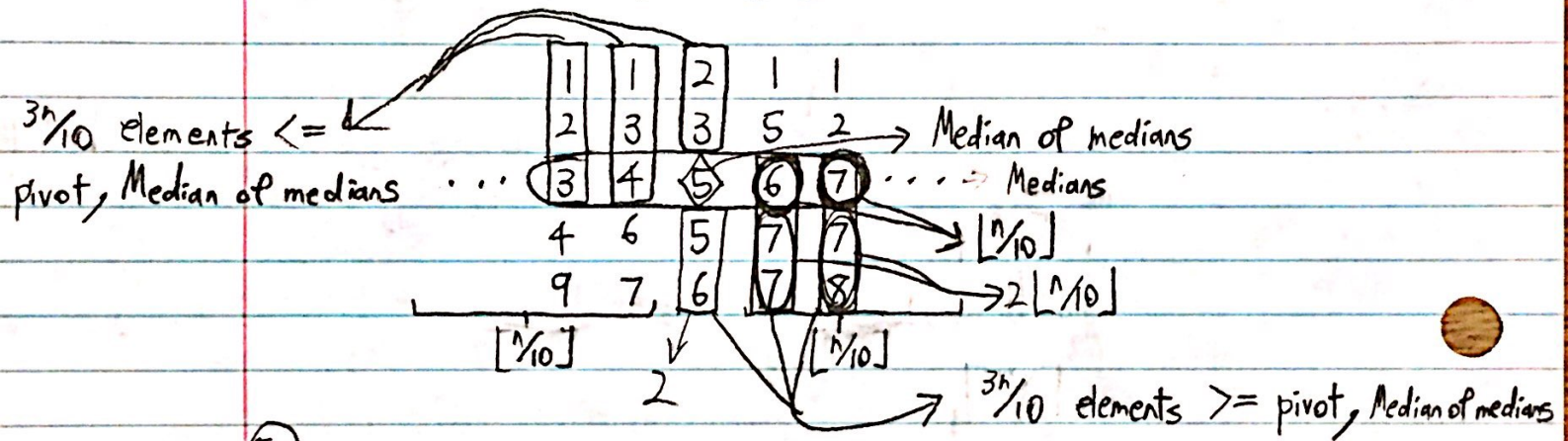
approx. approx.

Since there are also $\lfloor n/10 \rfloor$ medians larger than the median of medians which each have 2 elements in their subgroups of 5 that are also larger than the median of medians, along with 2 elements in the subgroup containing the median of medians that are larger than the median of medians,

∴ There are also at least $\lfloor n/10 \rfloor + 2\lfloor n/10 \rfloor + 2 \approx 3n/10$ elements in a that are >= the pivot, the median of medians.

E.g.

4 3 2 1 9, 7 6 3 4 1, 3 2 5 6 5, 6 7 5 1 7, 8 7 7 2 1, ...



(5.) m = position of pivot

- in the 2nd last else if statement where if condition (left + k < m) evaluates to true, the array for the recursive call is at most $n - (\lfloor 3n/10 \rfloor + 2) \approx 7n/10$ in size. The same goes for the last else statement.

approx./less

(6) ∴ Recurrence Relation:

$$T(n) = \underbrace{T(n/5)}_{\text{Finding the median of medians in b.}} + \underbrace{T(7n/10)}_{\text{The size of array at most that we need to recurse in 2nd last / last statements.}} + \underbrace{\theta(n)}_{\text{Sum of all the constants and the linear runtime to partition function.}}, \quad n \geq 10$$

Or else we would just do InsertionSort for $n < 10$ elements, executed by the starting if statement

runtime

InsertionSort $n/5$ subgroups of max. 5 elements each, taking $\theta(1)$ and the

Base Case: $T(1) = \theta(1)$, $T(2) = \theta(1)$ } $n < 10$, $T(n) = \theta(1)$

Scanned by CamScanner

*size = array size

Problem 3b).

$T(n) = T(n/5) + T(7n/10) + \theta(n)$

Work done / Runtime:

i = 0: $T(n)$ size $= n$, $\theta(n)$ — $= \theta(n)$

i = 1: $T(n/5)$ size $= n/5$, $\theta(n/5)$ + $T(7n/10)$ size $= 7n/10$, $\theta(7n/10)$ — $= \theta\left(\left(\frac{1}{5}+\frac{7}{10}\right)^1 n\right)$

i = 2: $T\left(\frac{n}{5}\times\frac{1}{5}\right)$ size $= \frac{n}{25}$, $\theta\left(\frac{n}{5}\times\frac{1}{5}\right)$ + $T\left(\frac{n}{5}\times\frac{7}{10}\right)$ size $= \frac{7n}{50}$, $\theta\left(\frac{n}{5}\times\frac{7}{10}\right)$ + $T\left(\frac{7n}{10}\times\frac{1}{5}\right)$ size $= \frac{7n}{50}$, $\theta\left(\frac{7n}{10}\times\frac{1}{5}\right)$ + $T\left(\frac{7n}{10}\times\frac{7}{10}\right)$ size $= \frac{49n}{100}$, $\theta\left(\frac{7n}{10}\times\frac{7}{10}\right)$ — $= \theta\left(\left(\left(\frac{1}{5}\right)^2 + 2\left(\frac{1}{5}\times\frac{7}{10}\right) + \left(\frac{7}{10}\right)^2\right)n\right)$

$= \theta\left(\left(\frac{1}{5}+\frac{7}{10}\right)^2 n\right)$

i = 3: $T\left(\frac{1}{5}^3 n\right)$ $T\left(\frac{1}{5}^2\times\frac{7}{10}\right)$ $T\left(\frac{1}{5}\times\frac{7}{10}^2\right)$ $T\left(\frac{7}{10}^3\right)$ $T\left(\frac{1}{5}^2\times\frac{7}{10}\right)$ $T\left(\frac{1}{5}\times\frac{7}{10}^2\right)$ $T\left(\frac{7}{10}^2\times\frac{1}{5}\right)$ $T\left(\frac{7}{10}^3\right)$

$\theta\left(\frac{1}{5}^3 n\right)$ $\theta\left(\frac{1}{5}^2\times\frac{7}{10}\right)$ $\theta\left(\frac{1}{5}\times\frac{7}{10}^2\right)$ $\theta\left(\frac{7}{10}^3\right)$ $\theta\left(\frac{1}{5}^2\times\frac{7}{10}\right)$ $\theta\left(\frac{1}{5}\times\frac{7}{10}^2\right)$ $\theta\left(\frac{7}{10}^2\times\frac{1}{5}\right)$ $\theta\left(\frac{7}{10}^3\right)$ — $= \theta\left(\left(\frac{1}{5}+\frac{7}{10}\right)^3 n\right)$

⋮

Since this is an unbalanced recursion tree, there would be multiple different layers for each recursive call as they all reach recursive call on an array of size 10 (→ Base case) or less after different number of layers of recursive calls to quantile().

Most importantly,

Branch from $T(n/5)$ will reach the base case when $\left(\frac{n}{5}^i\right) = 10$,

$\therefore$ Layers $i = (\log_5 n - \log_5 10) \approx \log_5 n$

Branch from $T(7n/10)$ will reach the base case when $\left(\frac{7}{10}\right)^i n = 10$,

$\therefore$ Layers $i = (\log_{10/7} n - \log_{10/7} 10) \approx \log_{10/7} n$

$\therefore$ By assuming we will have at most $\log_{(10/9)} n$ layers when $\left(\frac{1}{5}+\frac{7}{10}\right)^i n = 10$, ($\to 1/(\frac{1}{5}+\frac{7}{10})$)

Total Amt. of work: $\sum_{i=0}^{\theta(\log_{(10/9)} n)} \left(\frac{9}{10}\right)^i n = n\left(\frac{1-\left(\frac{9}{10}\right)^{\log_{(10/9)} n}}{1-\frac{9}{10}}\right)$

$\leq n\left(\frac{1}{\frac{1}{10}}\right) = 10n$

$\therefore$ Algorithm runs in: $\theta(n)$

At the i^{th} layer,

Work done / Total Runtime $\approx \theta\left(\left(\frac{1}{5}+\frac{7}{10}\right)^i n\right)$

Scanned by CamScanner