Tanmay Garg CS20BTECH11063

Algo Quiz 2 (mid term)

1. When selecting for pivot, we use median finding algorithm then recursive calls will change

let the array be partitioned in the worst case scenario ie.

$\cancel{\frac{7n+6}{10}}$

~~every~~ ~~terms~~

after dividing atleast $\left(3\left(\frac{n}{5}\right)\cdot\frac{1}{2}-2\right)$ elements

must be ~~larger~~ less than median & $\frac{7n}{10}+6$ on

the other side

$\therefore$ for large n the partition would look like

$\frac{3n}{10}, \frac{7n}{10}$

$\therefore$ the no. of calls $= 1+2+4+\dots+2^{\log_{\frac{10}{7}}(n)}$

$= 2^{\log_{10/7}(n)+1} - 1$

$= 2 \cdot n^{\log_{10/7} 2} - 1$

$\approx 2 \cdot n^{1.94} - 1$ no. of calls

2. $T(n) = n^{1/3} \cdot T(n^{1/3}) + 1$

$T(1) = T(2) = 1$

let us take ~~$n = 2^k$~~

$\therefore$ ~~$T(2^k)$~~ let us divide entire eq^n by $\sqrt{n}$

$$\frac{T(n)}{\sqrt{n}} = \frac{T(n^{1/3})}{n^{1/6}} + \frac{1}{\sqrt{n}}$$

let $\frac{T(n)}{\sqrt{n}} = P(n)$ $\therefore P(n) = P(n^{1/3}) + \frac{1}{\sqrt{n}}$

let $n = 2^k$ $\therefore P(2^k) = P(2^{k/3}) + \frac{1}{\sqrt{2^k}}$

let $P(2^k) = R(k)$ $\therefore R(k) = R\left(\frac{k}{3}\right) + \frac{1}{\sqrt{2^k}}$

let $k = 3^b$ $\therefore R(3^b) = R(3^{b-1}) + \frac{1}{\sqrt{2^{3^b}}}$

let $R(3^b) = S(b)$ $\therefore S(b) = S(b-1) + \frac{1}{\sqrt{2^{3^b}}}$

$\therefore R(k) \geq R\left(\frac{k}{3}\right) + 1$

$\therefore R(k) \approx \log_3 k$

$\therefore P(n) \approx \log_3 \log_2 n$

$T(n) \approx \sqrt{n} \log(\log n)$

$\therefore \underline{T(n) = O(\sqrt{n} \log(\log n))}$

TOPIC .... DATE ....

3. So the total no. of blocks with block size 13 = $\left\lceil \frac{n}{13} \right\rceil = k$

Time to find median of all groups = $O(n)$

we have $\left\lceil \frac{n}{13} \right\rceil$ medians

We need to make recursive calls to all these k groups

$\therefore$ time to find median for these = $T\left(\frac{n}{13}\right)$

$\therefore$ no. of elements less than median $\geq \frac{7n}{26} - 7$

" " " greater " $\geq \frac{7n}{26} - 14$

$\therefore$ $\cancel{T(n) \leq T\left(\frac{n}{13}\right) + O(n) + T\left(\frac{15n}{26}\right)}$

$$\cancel{T(n) = \begin{cases} T\left(\left\lceil \frac{n}{13} \right\rceil\right) + c\left\lceil \frac{n}{13} \right\rceil + O(n) & n \geq 13 \\ c & n < 13 \end{cases}}$$

$$T(n) = \begin{cases} T\left(\left\lceil \frac{n}{13} \right\rceil\right) + O(n) & n \geq 13 \\ O(n) & n \leq 13 \end{cases}$$

TOPIC .... DATE ....

4. Given 2 arrays $A[1 \ldots m]$, $B[1 \ldots n]$
alphabet $= \{0, 1, 2\}$
let us use the original $Edit(i,j)$ with some modification

if $i = 0$, $\therefore$ j insertions & the cost would be $j \times 0.75$

if $j = 0$ $\therefore$ i deletions & cost would be $i \times 0.75$

$$\cancel{ModEdit(i,j) = \begin{cases} i \times 0.75 & \text{if } j = 0 \\ j \times 0.75 & \text{if } i = 0 \\ \min \begin{cases} ModEdit(i, j-1) + 0.75 \\ ModEdit(i-1, j) + 0.75 \\ \begin{cases} ModEdit(i-1, j-1) + 0.5\,|A[i] - B[j]| & (\text{if } A[i] \neq B[j]) \\ ModEdit(i-1, j-1) & \text{else} \end{cases} \end{cases} \end{cases}}$$

$$ModEdit(i,j) = \begin{cases} i \times 0.75 & \text{if } j = 0 \\ j \times 0.75 & \text{if } i = 0 \\ \min \begin{cases} ModEdit(i, j-1) + 0.75 \\ ModEdit(i-1, j) + 0.75 \\ \begin{cases} ModEdit(i-1, j-1) + 0.5\,|A[i] - B[j]| & \text{if } A[i] \neq B[j] \\ ModEdit(i-1, j-1) & \text{else} \end{cases} \end{cases} & \text{else} \end{cases}$$

for substitution if characters are same then edit distance is $Edit(i-1, j-1)$, if different then edit distance $Edit(i-1, j-1) + \underbrace{0.5\,|A[i] - B[j]|}_{\text{cost to edit}}$

TOPIC .... DATE....................................

5. · Our algorithm would be to go through the entire array to find the indices where 1 occurs since all the 1s are continuous, it would be preferable to go through the array.

· We may also use a binary search type algorithm but the metric to compare the subarrays can be sum of all elements in that subarray

· using the above would still take more than O(n) time

· There can be an algorithm based on the total no. of cases which can work in Ω(log n) time but an extra metric to compare would be required whose computation time will always be O(n)