classmate

Date

Page

PART 1 : Recurrances

1) $T(n) = T(n-3) + 3\log n$

Initial Guess : $T(n) = O(n\log n)$

$\Rightarrow T(n) \leq cn\log n$ s.t $c>0$

$T(n) = T(n-3) + 3\log n$

$= \cancel{T(n) - 3T + 3\log n}$

$= \cancel{O(n\log n) - 3T + 3\log n}$

$\Rightarrow c(n-3)\log(n-3) + 3\log n$

$\leq c(n-3)\log n + 3\log n$

$\leq cn\log n - c3\log n + 3\log n$

Removing lower order terms

$\leq cn\log n$

2) $T(n) = 4T(n/3) + n$

Guess $T(n) = O(n^{\log_3 4})$

Prove $T(n) \leq cn^{\log_3 4}$

$T(n) = 4T\left(\frac{n}{3}\right) + n$

$\leq 4c(n/3)^{\log_3 4} + n$

$\leq \frac{4}{3^{\log_3 4}} cn^{\log_3 4} + n$

$\leq \cancel{4/4}\, cn^{\log_3 4} + n$

Date
Page

$\leq C n \log_3^4$

$C n^{\log_3^4} + n$

$\leq C n^{\log_3^4}$ since $n < C n^{\log_3^4}$ we remove it

3) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

Guess $T(n) = O(n)$

$\Rightarrow T(n) \leq Cn$

$= T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

$\leq \frac{Cn}{2} + \frac{Cn}{4} + \frac{Cn}{8} + n$

$\leq \left(\frac{C}{4} + \frac{C}{16} + \frac{C}{64} + 1\right) n$

$\leq Cn$ //

4) $T(n) = 4T\left(\frac{n}{2}\right) + n^3$

Guess $T(n) = O(n^2)$

Show $T(n) = O(n^2)$

$= 4T\left(\frac{n}{2}\right) + n^3$

$\leq 4C\left(n/2\right)^2 + n^3$

$\leq 4c\frac{n^2}{4} + n^3$

$= (c+n)n^2$

let $c_1 \geq (c+n)$

$T(n) \leq c_1 n^2$

Part - 2.

1) $T(n) = 3T(n/2) + n^2$

*) $a = 3, b = 2$.

$n^{\log_2 3}$ is less tha n^2

we use MT - 3

$T(n) = \Theta(n^2 \log^0 n)$

$= \Theta(n^2)$

2) $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

This cannot be solved by master theorem as we cannot find the inequality of a, b & n.

3) $T(n) = 3T(n/4) + n\log n$

$a = 3, b = 4, n = n\log n$

$n^{\log_4 3} \approx n\log n$

since $n\log n$ is greater we use MT 3

$= \Theta(n^1 \log^1 n)$

$T(n) = \Theta(n\log n)$

Date
Page

4) $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

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

$a = 2,\ b = 2$

$\Rightarrow n^{\log_2 2} \simeq n \log n^{-1}$

$n = n$

here we use MT 2.

$T(n) = \Theta(\cancel{n\log}\ n^{\log_2 2} \log\log n)$

$= \Theta(n \log\log n)$

$T(n) = \Theta(n \log\log n)$

5) $T(n) = 0.5\,T(n/2) + 1/n$

We Cannot solve this using master theorem.

classmate

Date

Page

PART - 3

1) How would you represent a d-ary heap

Ans: The d-ary heap can be considered similar to a binary heap with Parent indexes and child Indexes

↳ Parent [i] = [i/d]

j^{th}-child = d.[i+j] where j = 0 ... d-1

The root would be at index i = 1

PART - 3

Q.2 A ~~d-ary~~ d-ary heap height → $\Theta(\log_b n)$

$$1 + d + d^2 + \dots d^{h-1} < n \le 1 + d + d^2 + \dots + d^h$$

$$\frac{d^h - 1}{d-1} < n \le \frac{d^{h+1} - 1}{d-1}$$

$$d^h < n(d-1) + 1 \le d^{h+1}$$

$$\therefore \cancel{h = [(\log_d(n(d-1)+1) \ge h+1}$$

$$h = [(\log_d(n(d-1)+1) - 1)]$$

PART - 3

Q.3 HEAPIFY (A, i, n, d)

j → i

for K = 0 & d = 1

if d×i+K ≤ n and A[d×i+K] > A[j]

then j = d×i+K

if j ≠ i

then Exchange A[i] ↔ A[j]

HEAPIFY (A, j, n, d)

EXTRACT MAX (A, n)

$max \leftarrow A(i)$

$A[1] \leftarrow A[n]$

$n = n - 1$

HEAPIFY $(A, 1, n, d)$

return max

The running time of the Heapify algorithm is $O(d \log_d n)$.
Since in Extract - max we have only Constant work.
The total running time is

$$O(d \log_d n)$$

PART - 3

Q 4. Insert (A, K, n, d)

$n = n + 1$

$A[n] = Max$

Increase-Key (A, i, K, n)

From Q5 we know the running time of Increase-Key is $O(\log_d n)$, Since in Insert we are not looping and just adding, So the total running time would be $O(\log_d n)$

PART - 3

Q 5. Increase-Key (A, i, j)

$A[i] \leftarrow max(A[i], K)$

if $K = A[i]$

while $i > 1$ and $A\left[\frac{i}{d}\right] < A[i]$

$i \leftarrow [i/d]$

The Implementation loops proportionally to the depth of the tree, therefore ~~A~~ the running time $O(\log_d n)$ time.
at each step the loop checks the i^{th} node to its Parent ~~oro~~ node and swaps them if it is found to violate the heap property
So the running time will be $O(\log_d n)$

PART - 4

Q 1

We know all elements are equal. So the ~~Randomized~~ ~~Rand~~ RANDOMIZED-QUICKSORT will always lead to result $q = r$.
then the running time will be recurrence of Same

$$\Rightarrow \quad T(n) = T(n-1) + \Theta(n)$$

$$= \Theta(n^2)$$

Q 2.

```
PARTITION'(A, P, r)
    x = A(P)
    low = P
    high = P
    for j = P+1 to r
        if A[j] < x
            y = A[j]
            A[j] = A[high+1]
            A[high+1] = A[low]
            A[low] = y
            low = low + 1
            high = high + 1
        else if A[j] == x
```

exchange A [high + 1] ↔ A [j]
high = high + 1
return (low, high)

Q3

```
QUICKSORT'(A, p, r)
  if p < r
    (low, high) = RANDOMIZED-PARTITION'(A, p, r)
    QUICKSORT'(A, p, low - 1)
    QUICKSORT'(A, high + 1, r)
```