Adit Kotwal
NYU ID: ask9100
30th March, 2021

Fundamental Algorithms - Mid term

Q1.] Asymptotics

a) a) $\log(n^9 + 37) \Rightarrow \Theta(\log n)$ [Ignoring constants]

b) $\frac{n^2+2}{1+n^4 . 2^{-n}} \Rightarrow \frac{2^n n^2}{2^n+n^4} + \frac{2^n . 2}{2^n+n^4} \Rightarrow \Theta(n^2)$

c) $(\log \sqrt{n}) . \log(37n^2+45) \Rightarrow \Theta((\log n)^2)$

d) $3^n + (\log n)^{6 \log n} \Rightarrow \Theta(3^n)$

e) $10^{207} n + \log n! \Rightarrow \Theta(n \log n)$ [$\because \log n! = n \log n$]

f) $\frac{3^n - 1000}{2^n+1} \Rightarrow \frac{3^n}{2^n+1} - \frac{1000}{2^n+1} \Rightarrow \Theta(1.5^n)$

g) $8^{(n+1)/3} = 8^{n/3} . 8 \Rightarrow \Theta(8^n)$

h) $4^{2n} + 1000 = (4^2)^n + 1000 = \Theta(16^n)$

b) Ranking functions in increasing order of growth.

$\Theta(\log n) < \Theta((\log n)^2) < \Theta(n \log n) < \Theta(n^2) < \Theta(1.5^n) < \Theta(3^n)$
$< \Theta(8^n) < \Theta(16^n)$

Q2.] Recurrences

a) 1. $a = 1$, $b = 32/31$, $f(n) = \Theta(\log n)$, $g(n) = n^{\log_b a} = n^{\log_{32/31} 1} = n^0 = 1$

Even though $\Theta(\log n)$ is greater than $\Theta(1)$, it is not polynomially greater. ~~If~~ It falls between case 2 and case 3.

∴ Master Theorem is not applicable. for $T(n) = T(31n/32) + \log n$

2. $a = 4$, $b = 2$, $f(n) = n^2$, $g(n) = n^{\log_b a} = n^{\log_2 2^2} = n^2$

This is case 2

∴ ~~$T(n) = \Theta(\log f(n)) = \Theta(\log n^2) = \Theta(\log n)$~~

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

$= \Theta(n^2 \log n)$ for $T(n) = 4T(n/2) + n^2$

3. $a = 9$, $b = 8$, $f(n) = \Theta(\log n!) = \Theta(n \log n)$

$g(n) = \Theta(n^{\log_b a}) = n^{\log_8 9} \approx n^{1+\epsilon}$ where $0 < \epsilon < 1$

∴ $f(n) = O(g(n)) = O(n^{\log_b a - \epsilon})$

This is case 1

∴ $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_8 9})$ for $T(n) = 9T(n/8) + 700 \log(n!)$

4. $T(n) = 8T(n-1) + 35$, $T(0) = 0$

Dividing by 8^n

$$\frac{T(n)}{8^n} = \frac{8T(n-1)}{8^n} + \frac{35}{8^n}$$

$$\Rightarrow \frac{T(n)}{8^n} = \frac{T(n-1)}{8^{n-1}} + \frac{35}{8^n}$$

Q2.] Recurrences

b) $T(n) = \Theta(1)$, $n \leq 5$

$= \frac{2}{3} T(n) + \frac{1}{3} T(n/2)$, $n > 5$

Expected Value :

1. $E(T(n)) = 7 + \frac{2}{3}(1) + \frac{1}{3}(0)$

$= 23/3$

2. When $n = 0$:

$T(n) = \cancel{\frac{1}{3} T(n)+} \frac{1}{3} T(n/2)$

3. When $n = 1$:

$T(n) = \frac{2}{3} T(n)$

4. $\therefore T(n) = \frac{2}{3} T(n) + \frac{1}{3} T(n/2) + \Theta(1)$

$\Rightarrow \frac{1}{3} T(n) = \frac{1}{3} T(n/2) + \Theta(1)$

$\Rightarrow T(n) = 1 T\left(\frac{n}{2}\right) + \Theta(1)$

$a = 1$, $b = 2$, $f(n) = \Theta(1)$

5. $g(n) = n^{\log_b a} = n^{\log_2 1} = n^0 = \Theta(1)$

$\therefore$ This is case 2 of master theorem.

6. $\therefore T(n) = \Theta(n^{\log_b a} \lg n)$

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

c) 1. Expected Value $= \frac{2}{3} T(n) + \frac{1}{3} BLA\left(T\left(\frac{n}{2}\right)\right)$

$= \frac{2}{3} T(n) + \frac{1}{3} BLA(0) + 5$

$= \frac{2}{3}(1) + \frac{1}{3}(0) + 5$

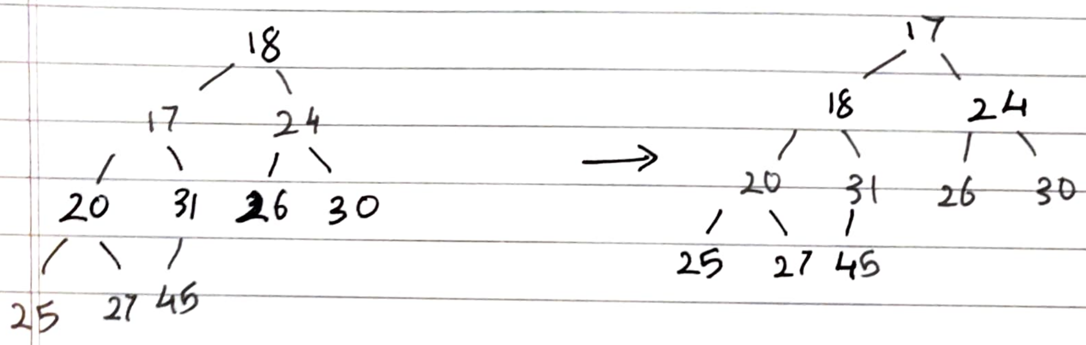
$= 17/3$

2. When $n = 0$, $T(n) = n + \frac{1}{3} BLA(T(n/2)) \Rightarrow S(n) = n + \frac{1}{3} T(S(n/2))$

3. When $n = 1$, $T(n) = n + \frac{2}{3} T(n) \Rightarrow S(n) = n + \frac{2}{3} S(n)$

Q3.] Priority Queues and Heaps

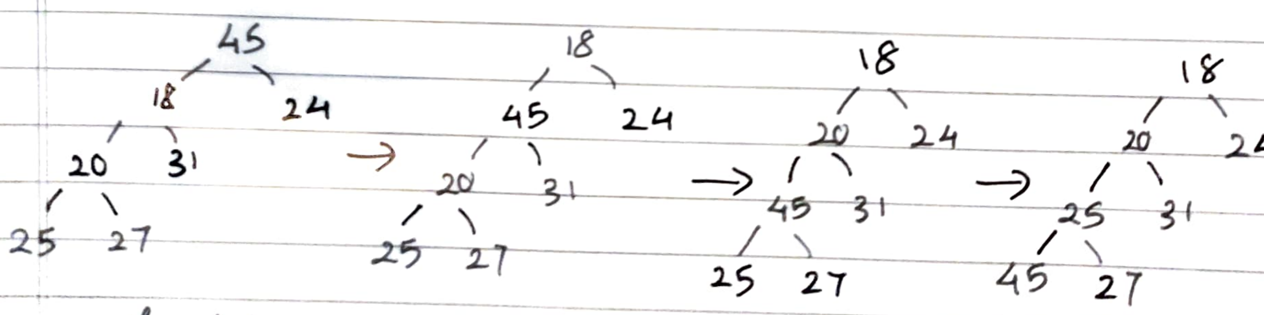
a) A = [18, 17, 24, 20, 31, 26, 30, 25, 27, 45]



A is not a min heap since the root has an element which is greater than one of its children. This violates the min heap property.

This becomes a min heap by replacing 17 with 18.

b) Extract Min: 17 (Root will be extracted)



~~Final Heap~~:

(Final Heap)

c) Heap-Increase-Key (A, i, key, n)

Heap-Extract-Min (A, i)

Min-Heap-Insert (A, i, key)

Min-Heapify (A, n)

Q4.] Sorting and Order Statistics.

a) $$\sum_{j=n/3}^{2n/3-1} \text{Rank}(j)$$

Find Rank (A, i)

```
1. Convert each element to list which holds value and index
2. RadixSort (A)
3. Loop through sorted list
4. If original index lies within given range [n/3 to 2n/3 - 1],
   then return Sum upto current index.
```

b) Running time of Counting Sort ⇒ $O(n+i)$ Where $i \leftarrow$ # comparisons.

c) Running time of Heap Sort ⇒ $O(n \log i)$

d) Running time of Insertion Sort ⇒ $O(n^2)$ Independent of i

e) The number of passes of Counting Sort ~~is~~ will be n when Radix sort is using base n

Time for each of these passes is n^n

Q5.] a) Binary Trees

(i) → Preorder walk : Root - Left - Right

⇒ 100, 25, 98, 28, 50, 40, 20, 91

→ Inorder walk: Left - Root - Right

⇒ 98, 25, 28, 50, 100, 20, 40, 91

→ Postorder walk: Left - Right - Root .

⇒ 98, 50, 28, 25, 20, 91, 40, 100

(ii)

```
if isLeaf(v) then
    v.win-los = 1
else
    v.win_los = 0
    if not v.left then
        ~~return v.left Compute~~
        v = v.right
        v.win_los = Compute-Win(v)
    if Not v.right then
        v = v.~~to~~ left
        v.win_los = Compute-Win(v)
```

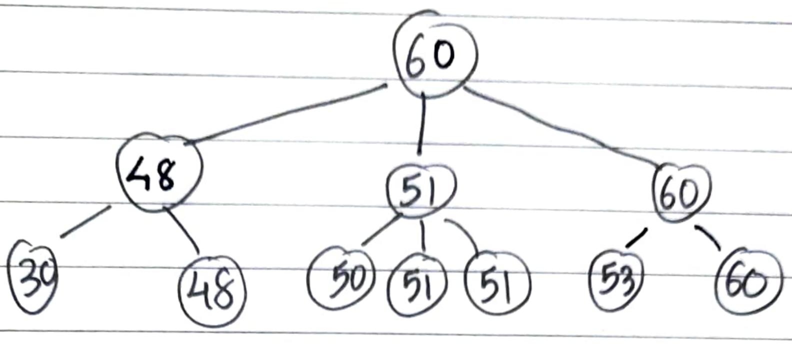
$$T(n) = 0 \quad , \quad n = 0$$
$$= 1 \quad , \quad n = 1$$

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

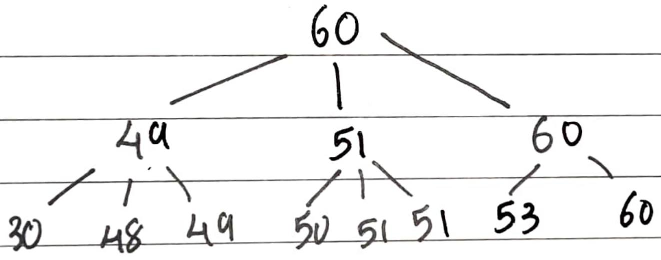
This solves to $T(n) = $ ~~O(lo~~ $O(n)$ [By Master Theorem]

Q5.] b) 2-3 Trees

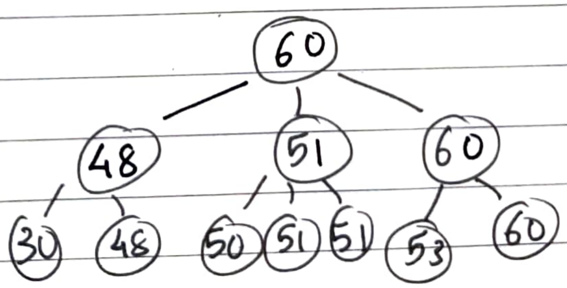
(i)



(ii) Insert 49



(iii) Delete 49

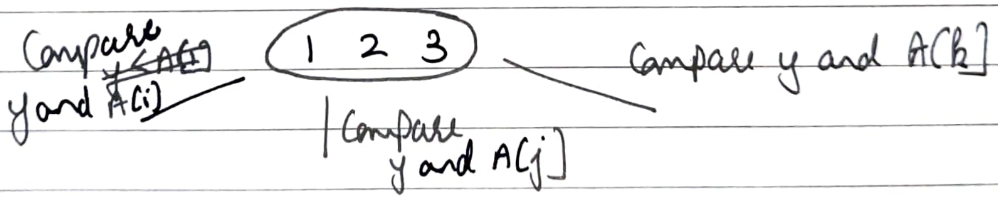


Q6.] Lower Bound and Divide and Conquer.

a)
```
if low >= high then
    return ~~highest~~ high
mid = $\lfloor (low + high) / 2 \rfloor$
if A[mid] < y
    return Modified-Bin-Search(A, y, mid+1, high)
else return Modified-Bin-Search(A, y, low, mid-1)
```

Invocation Call : Modified_Binary_Search(A, y, 1, n)
Worst Case Running time ⇒ $O(\log n)$

b) $n = 6$


Compare ~~y<A[i]~~ y and A[i]
1 2 3
Compare y and A[k]
Compare y and A[j]