URBAN EDGE 17/1/20

Module 1 & 2

T1	T2	10^6 instructns per second.
$O(n^2)$	$O(2^n)$	
$\Rightarrow n^2$ instructns $= n^2 \times 10^{-6}$ s	$\Rightarrow 2^n$ instructns $= 2^n \times 10^{-6}$ s	$\Rightarrow$ 1 instructn $= 10^{-6}$ s
if $n=100$, time taken $= 100^2 \times 10^{-6}$ s $= 10^{-2}$ s $= 0.01$ s	time taken $= 2^{100} \times 10^{-6}$ $= 1.267 \times 10^{24}$ s $= 4.01 \times 10^{16}$ years	

Key comparison technique used in sorting can only go down till nlogn, & its complexity can't be reduced further.
∵ every problem has an inherent complexity

22/1/20

c^n → exponential
n^3 → cubic
n^2 → quadratic
nlogn → log-linear
n → linear
logn → logarithmic
const

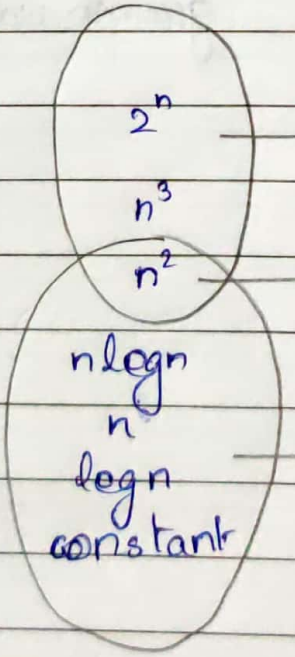
↑ growth rate (↑ upwards)

URBAN EDGE 22/1/20

Input Size	$\log n$	$33n$	$4n+50$	$13n^2$	n^3	2^n
10	$\log 10 \times 10^{-6}$ s $= 1 \times 10^{-6}$ s	330×10^{-6} s $= 330\,\mu s$	$90\,\mu s$	1.3 ms	1 ms	1 ms
100	$2\,\mu s$	3.3 ms	$450\,\mu s$	130 ms	1 s	4×10^{16} years
1000	$3\,\mu s$	33 ms	4 ms	13 s	16 min	
10,000	$4\,\mu s$	330 ms	40 ms	22 min	11 days	
1,00,000	$5\,\mu s$	3.3 s	≈ 0.4 s	36 hours	31 years	

Computer speed = 10^6 instructions per second

$\Rightarrow$ 1 instruction = 10^{-6} s



$\Omega(n^2)$

$\Theta(n^2)$

$O(n^2)$

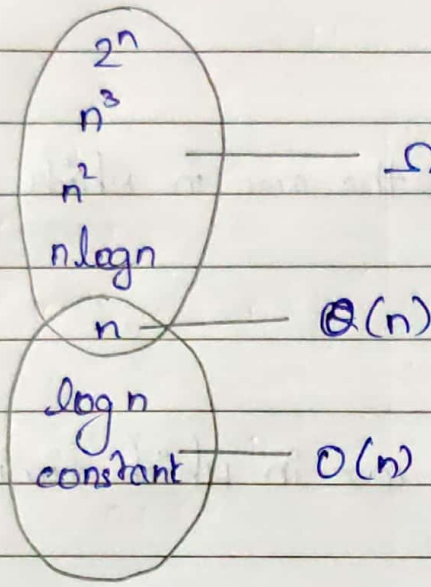
$n^2, n\log n, n, \log n,$ constant

$g(n)$ $f(n)$

$O(n^2) \Rightarrow$ Max. n^2 time

$\Omega(n^2) \Rightarrow$ Min n^2 "

URBAN EDGE 22/1/26



$O(n) \Rightarrow n, \log n, \text{constant}$

$\Omega(n) \Rightarrow n, n\log n, n^2, n^3$

$O(g(n))$

Let $g(n)$ be a function from non-negative integers to positive real numbers. Then $O(g(n))$ is the set of funct^{n}s $f(n)$ also from non-negative integers to +ve real no.s such that for some real constant $c>0$ & some non-negative integer const n_0, then $f(n) \leq c \cdot g(n) \; \forall \; n \geq n_0$

$\Omega(g(n))$

$f(n) \geq c \cdot g(n)$, for $n \geq n_0$

$\Theta(g(n))$

$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

$o(g(n))$

$f(n) < c \cdot g(n)$, for $n \geq n_0$

$\omega(g(n))$

$f(n) > c \cdot g(n)$, for $n \geq n_0$

$O(g(n))$

A funct^n $f(n) \in O(g(n))$ if

$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$, including the case in which limit is zero

$\Omega(g(n))$

A funct^n $f(n) \in \Omega(g(n))$ if

$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$, including the case in which limit is ∞

$\Theta(g(n))$

A funct^n $f(n) \in \Theta(g(n))$ if

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, for some constant c such that $0 < c < \infty$

$o(g(n))$

A funct^n $f(n) \in o(g(n))$ if

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

$\omega(g(n))$

A funct^n $f(n) \in \omega(g(n))$ if

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

① S.T $2n+3$ is in $O(n)$

ans $f(n) = 2n+3$

$g(n) = n$

$\frac{f(n)}{g(n)} = \frac{2n+3}{n} = 2 + \frac{3}{n}$

URBAN EDGE 24/1/20

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \left(2+\frac{3}{n}\right) = 2 + \lim_{n\to\infty} \frac{3}{n} = 2 < \infty$

$\therefore 2n+3 \in O(n)$

(2) S.T $3n^3+4n$ is not in $O(n)$

ans $f(n) = 3n^3+4n$

$g(n) = n^2$

$\frac{f(n)}{g(n)} = \frac{3n^3+4n}{n} = 3n^2+4$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} (3n^2+4) = \infty \not< \infty$

$\therefore 3n^3+4n \notin O(n)$

(3) S.T n^2 is not in $O(n\log n)$

ans $f(n) = n^2$

$g(n) = n\log n$

$\frac{f(n)}{g(n)} = \frac{n^2}{n\log n} = \frac{n}{\log n}$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n}{\log n} = \lim_{n\to\infty} \frac{1}{\frac{1}{n}} = \lim_{n\to\infty} n = \infty \not< \infty$

↓ L'Hospital Rule

$\therefore n^2 \notin O(n\log n)$

(4) S.T $3n^3+4n$ is in $\Omega(n^3)$

ans $f(n) = 3n^3+4n$

$g(n) = n^3$

$\frac{f(n)}{g(n)} = \frac{3n^3+4n}{n^3} = 3+\frac{4}{n^2}$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \left(3+\frac{4}{n^2}\right) = 3+\lim_{n\to\infty} \frac{4}{n^2} = 3>0$

$\therefore 3n^3+4n \in \Omega(n^3)$

(5) S.T $3n+5$ is in $\omega(1)$

ans $g(n) = 3n+5$

$g(n) = 1$

$\frac{g(n)}{g(n)} = \frac{3n+5}{1} = 3n+5$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} (3n+5) = \infty = \infty$

$\therefore 3n+5 \in \omega(1)$

(6) S.T $3n+5$ is not in $o(n)$

ans $g(n) = 3n+5$

$g(n) = n$

$\frac{g(n)}{g(n)} = \frac{3n+5}{n} = 3+\frac{5}{n}$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \left(3+\frac{5}{n}\right) = 3 \neq 0$

$\therefore 3n+5 \notin o(n)$

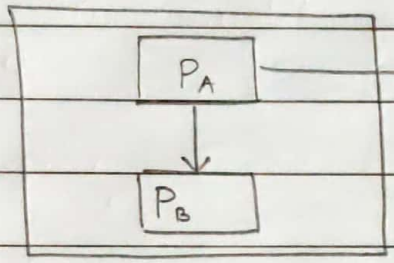
URBAN EDGE 29/1/20

Control Structures

(i) Sequencing
(ii) If-then-else
(iii) For loop
(iv) While loop
(v) Recursion

(i) Sequencing

P_A - Part A, P_B - Part B



Time taken $t_A = \Theta(n)$

$t_B = \Theta(n^2)$

Computatn time $= t_A + t_B = \max\{t_A, t_B\} = \max\{\Theta(n), \Theta(n^2)\}$

$= \Theta(n^2)$

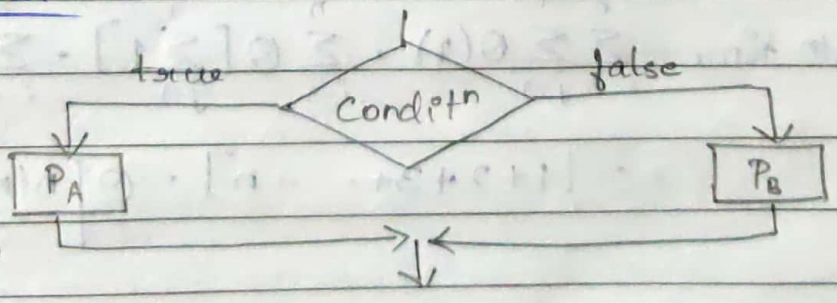
If $t_B = O(n^2) \Rightarrow t_A + t_B = \max\{t_A, t_B\} = \max\{\Theta(n), O(n^2)\}$

$= O(n^2)$

If $t_B = \Theta(n^2) \Rightarrow t_A + t_B = \max\{t_A, t_B\} = \max\{\Theta(n), \Theta(n^2)\}$

$= O(n^2)$

(ii) If-then-Else



$t_A = O(n^2)$

$t_B = O(n)$

Computatn time $= t_A | t_B = \max\{t_A, t_B\} = \max\{O(n^2), O(n)\} = O(n^2)$

(iii) For Loop

```
for i=1 to n
{
    P(i)
}
```

$t_i \rightarrow$ time taken for $P(i)$

$$\text{Computatn time} = \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \Theta(1) = \Theta\left[\sum_{i=1}^{n} 1\right] = \underline{\underline{\Theta(n)}}$$

```
for i=1 to n
{ for j=1 to n
  { P(i,j)
  }
}
```

$$\text{Computatn time} = \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} \Theta(1) = \sum_{i=1}^{n} \Theta(n) = \underline{\underline{\Theta(n^2)}}$$

```
for i=1 to n
{ for j=1 to i
  { P(i,j)
  }
}
```

$$\text{Computatn time} = \sum_{i=1}^{n}\sum_{j=1}^{i} \Theta(1) = \sum_{i=1}^{n} \Theta\left[\sum_{j=1}^{i} 1\right] = \sum_{i=1}^{n} \Theta(i) = \Theta\left[\sum_{i=1}^{n} i\right]$$

$$= \Theta[1+2+3+\ldots+n] = \Theta\left[\frac{n(n+1)}{2}\right] = \underline{\underline{\Theta(n^2)}}$$

- Avoid nested loops as far as possible.

URBAN EDGE 29/1/20

```
for {i=2 to m-1}
{ for j=3 to i
  { P(i,j)
  }
}
```

Computatn time $= \sum_{i=2}^{m-1} \sum_{j=3}^{i} O(1) = \sum_{i=2}^{m-1} O\left[\sum_{j=3}^{i} 1\right] = \sum_{i=2}^{m-1} O(i) = O(m^2)$

(iv) While Loop

```
while (n>0)
{ =
  n←n-1
}
```

O(n)

```
while (n>0)
{ =
  n←n/2
}
```

O(logn)

(v) Recursion

```
fact(n)
{ if n=0 return 1
  return n * fact(n-1)
}
```

```
recursion(n)
{ Base case
  recursion(n-1)
            n/2
}
```

Recurrence Equatn :- T(n) = T(n-1) + non-recursive cost

```
fib(n)
{ if n=0 return 0
  fib(n-2) + fib(n-1)
}
```

Recurrence Equatn:

$T(n) = T(n-2) + T(n-1) + 1$

URBAN EDGE 29/1/20

```
recursion(n)
{ Base case
   =
   recursion(n/2)
   recursion(n/2)
}
```

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

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

```
binarysearch(n)
{   O(1)
 if a[mid] == key return mid
    binarysearch(n/2)
 else
    binarysearch(n/2)
}
```

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

```
recursion(n)
{ for i=1 to n
   { =
   }
   recursion(n/2)
   recursion(n/2)
}
```

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

Recurrence Equation

The running time of the recursive algorithm can be obtained by a recurrence equation.

A recurrence is an equation or an inequality that describes a func^n in terms of it's values on smaller inputs

eg: Binary search, Factorial, Fibonacci

URBAN EDGE 29 / 1 / 20

Iteration Method

$T(n) = T(n-1) + 1$, where $T(1) = \Theta(1)$ → ①

Put $n = n-1$ in ①

$T(n-1) = T(n-1-1) + 1 = T(n-2) + 1$ → ②

Substitute equation ② in equation ①

$T(n) = [T(n-2) + 1] + 1 = T(n-2) + 2$ → ③

Put $n = n-2$ in ① ⇒ $T(n-2) = T(n-3) + 1$ ~~$= T(n-4) + 2$~~ → ④

Substitute eq. ④ in eq. ③

$T(n) = T(n-3) + 3$ → ⑤

In general,

$T(n) = T(n-k) + k$ → ⑥

$k \geq 1$

At base case,

$n - k = 1 \Rightarrow k = n-1$ → Substitute in eq. ⑥

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

31/1/20

① Solve the recurrence $T(n) = T(n/2) + n$

ans. $T(n) = T(n/2) + n$ → ①

Put $n = n/2$ in eq. ①

$T(n/2) = T(n/4) + n/2$ → ②

Substitute eq. ② in eq. ①

$T(n) = [T(n/4) + n/2] + n$ ~~$= T(n/4) + 3n/2$~~ → ③

~~Substitute~~ Put $n = n/4$ in ①

$T(n/4) = [T(n/8) + n/4]$ → ④

Substitute eq. ④ in eq. ③

$T(n) = $ ~~$T(n/4) +$~~ $T(n/8) + n/4 + n/2 + n$ → ⑤

In general, $\lg \to \log_2$

$$T(n) = T\left(n/2^k\right)_{k\geq 1} + \sum_{j=0}^{k-1} \frac{n}{2^j} \longrightarrow (6)$$

At the base case,

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow \log n = \log(2^k)$$

$$\Rightarrow \log n = k \log 2 \Rightarrow k = \log_2 n \Rightarrow k = \lg n$$

Substitute in eq. (6)

$$T(n) = T(1) + \sum_{j=0}^{\lg n - 1} \frac{n}{2^j} = \Theta(1) + n \sum_{j=0}^{\lg n - 1} \frac{1}{2^j}$$

Imp ★

$$\sum_{j=0}^{\infty} \frac{1}{2^j} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = \frac{1}{1-x} \text{ where } x = 1/2 = \frac{1}{1-1/2} = 2$$

$$T(n) < \Theta(1) + n \cdot 2$$

$$\therefore T(n) \in \Theta(n)$$

② Solve the recurrence $T(n) = 2T(n/2) + n$

ans $T(n) = 2T(n/2) + n \longrightarrow (1)$

Put $n = n/2$ in (1) $\Rightarrow T(n/2) = 2T(n/4) + n/2 \longrightarrow (2)$

Substitute (2) in (1)

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

Put $n = n/4$ in (1) $\Rightarrow T(n/4) = 2T(n/8) + n/4 \longrightarrow (4)$

Substitute (4) in (3)

$$T(n) = 8T(n/8) + 3n \longrightarrow (5)$$

In general,

$$T(n) = 2^k T\left(n/2^k\right) + kn \longrightarrow (6)$$

$k \geq 1$

At the base case,

$\frac{n}{2^k} = 1 \Rightarrow k = \lg n$

$a^{\lg b} = b^{\lg a}$

Substitute in eq. ⑥

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

~~$T(n) \leq n\Theta(1) + n\lg n$~~ $= 2^{\lg n} T(1) + n\lg n$

~~$\therefore T(n) \in \Theta(n \log n)$~~ $= 2^{\lg n}\Theta(1) + n\lg n$

$= n^{\lg 2}\Theta(1) + n\lg n$

$\therefore T(n) \in \Theta(n \log n)$

5/2/20

Recursion Tree Method

Each node in the recursion tree has 2 fields: the size field & the non-recursive cost field

T(size)	Non-recursive cost	→ Node structure

Phase 1:- Construct the recursion tree

Phase 2:- To evaluate the recursion tree

(i) Sum the non-recursive cost of all nodes at the same depth. This is called the row sum.

(ii) Sum those row sums over all depths.

① Solve $T(n) = 2T(n/2) + n$ using the recursion tree.

ans. Phase 1:- Build the tree

Hoark copy

$T(k) = 2T(k/2) + k$ Substitute $k = n \Rightarrow 2T(n/2) + n$

No. of children ↓ Non-recursive cost

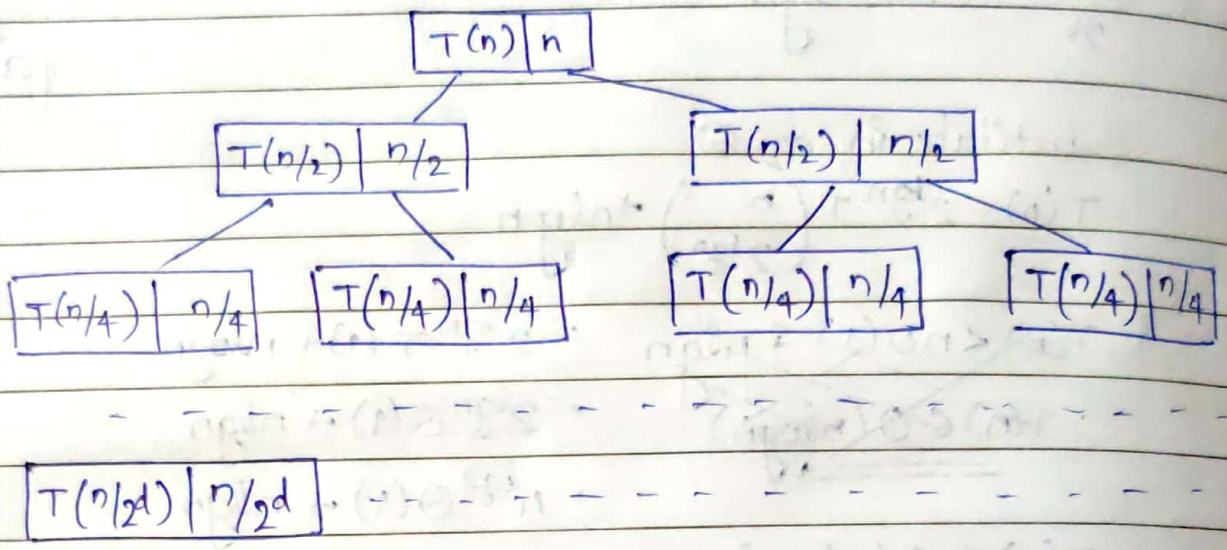
URBAN EDGE 5/2/20

ROW sum

n

$\frac{n}{2}+\frac{n}{2}=n$

$\frac{n}{4}+\frac{n}{4}+\frac{n}{4}+\frac{n}{4}=n$



$T(n/2^d) \mid n/2^d$

Sum of row sums = $n+n+n+\cdots$

$\frac{n}{2^d}=1 \Rightarrow n=2^d \Rightarrow d=\lg n$

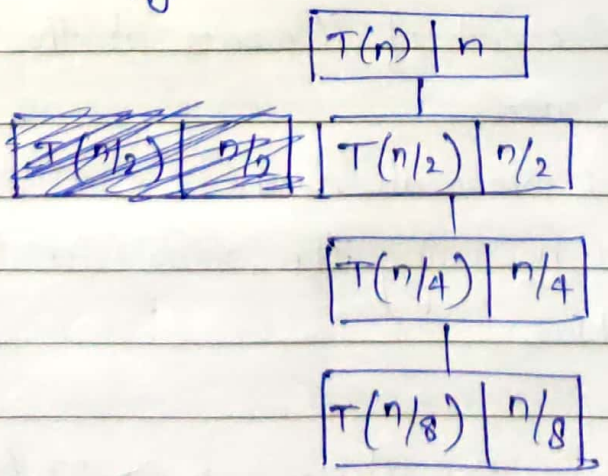
$\lg n+1$ levels for tree $\Rightarrow n$ added $\lg n+1$ times

$T(n)=n+n+\cdots=n\lg n$

$\therefore T(n) \in \Theta(n\lg n)$

② Solve $T(n)=T(n/2)+n$

ans. Work copy $\rightarrow T(k)=T(k/2)+k \Rightarrow T(n/2)+n$



ROW sum

n

n/2

n/4

n/8

Sum of row sums = $n+n/2+n/4+n/8 \leq 2n$

$\therefore T(n) \in O(n)$ or $\Theta(n)$

Common Recurrence Equations

(i) Divide & Conquer

$T(n) = bT(n/c) + f(n)$

The main problem of size n can be divided into b sub-problems ($b \geq 1$) of size n/c where $c > 1$, $f(n)$ is the non-recursive cost.

b is called the branching factor

(ii) Chip & Be Conquered

$T(n) = bT(n-c) + f(n)$

The main problem of size n can be chipped down to b sub-problems of size $n-c$ where $c > 0$, $f(n)$ is the non-recursive cost.

Divide & Conquer - General Case

The general divide & conquer recurrence eq. is

$$T(n) = bT(n/c) + f(n)$$

The size parameter decreases by a factor of c each time the depth increases. Therefore, at the base case,

$$\frac{n}{c^D} = 1 \Rightarrow n = c^D$$

$$\Rightarrow \log n = \log(c^D)$$

$$\Rightarrow \log n = D \log c$$

$$\Rightarrow \boxed{D = \frac{\log n}{\log c}}$$

If the branching factor is b, the no. of nodes at depth D is,

$L = b^D$

$\log L = \log(b^D) \Rightarrow \log L = D \log b$

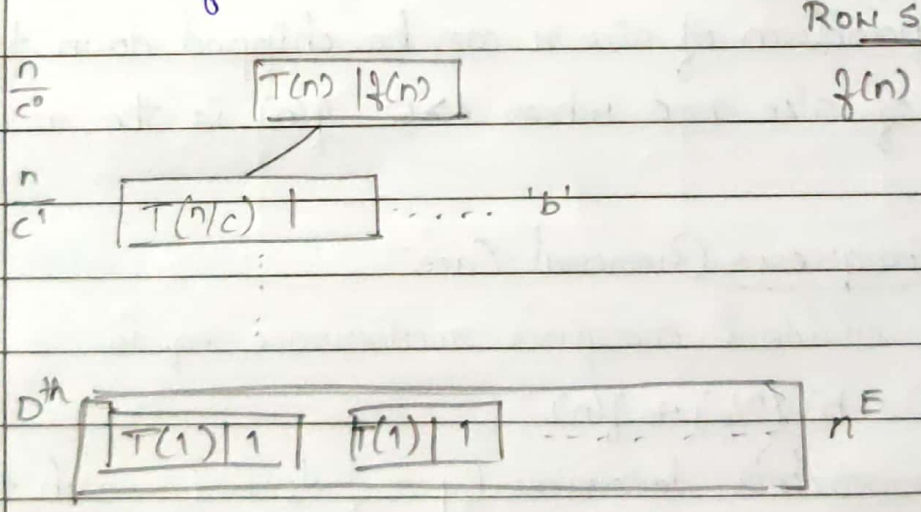
$\Rightarrow \log L = \frac{\log n}{\log c} \cdot \log b$

$\Rightarrow \log L = \frac{\log b}{\log c} \cdot \log n$

$\log L = E \cdot \log n$, $E = \frac{\log b}{\log c}$ → critical exponent

$\boxed{L = n^E}$

The no. of leaves in the recursion tree is $L = n^E$



Conclusion:-

(i) If the row sums form an increasing geometric series, then $T(n) \in \Theta(n^E)$ where $E = \frac{\log b}{\log c}$; critical exponent

i.e. the cost is proportional to the no. of leaves in the recursion tree

URBAN EDGE 5/2/20

(ii) If the row sums remain const. $T(n) \in \Theta(f(n)\log n)$

(iii) If the row sums form a decreasing geometric series, then $T(n) \in \Theta(f(n))$ which is proportional to the non-recursive cost of the root.

** Imp Master Theorem

The soln of the recurrence equation $T(n) = bT(n/c) + f(n)$ where $E = \frac{\log b}{\log c}$; critical exponent

(i) If $f(n) \in O(n^{E-\varepsilon})$ for any +ve ε, then $T(n) \in \Theta(n^E)$, which is proportional to the no. of leaves in the recursion tree.

(ii) If $f(n) \in \Theta(n^E)$, then $T(n) \in \Theta(f(n)\log n)$ as all node depths contribute equally.

(iii) If $f(n) \in \Omega(n^{E+\varepsilon})$ for any +ve ε, & $f(n) \in O(n^{E+\delta})$ for some $\delta \geq \varepsilon$, then $T(n) \in \Theta(f(n))$, which is proportional to the non-recursive cost of the root of the recursion tree.

12/2/20

① Solve the recurrence $T(n) = T(3n/4) + 1$ using Master theorem.

ans: $T(n) = T(3n/4) + 1$

$b = 1, c = 4/3, f(n) = 1$

$E = \frac{\log b}{\log c} = \frac{\log 1}{\log \frac{4}{3}} = 0$

$\therefore n^E = n^0 = 1$

Case 1:- $f(n) \in \Theta(n^{E-\varepsilon})$

$\Rightarrow 1 \in O(n^{0-\varepsilon})$

$\Rightarrow n^0 \in O(n^{0-\varepsilon})$ ⇒ Case 1 failed.

We cannot find a +ve ε which satisfies the condition

Case 2:- $f(n) \in \Theta(n^E)$

$\Rightarrow 1 \in \Theta(n^0)$

$\Rightarrow n^0 \in \Theta(n^0)$

$\Rightarrow 1 \in \Theta(1) \Rightarrow$ Case 2 passed

$\therefore T(n) \in \Theta(1 \cdot \log n)$

(2) Solve the recurrence $T(n) = 4T(n/2) + n^3$

ans. $b=4, c=2, f(n)=n^3$

$E = \frac{\log b}{\log c} = \frac{\log 4}{\log 2} = 2$

$\therefore n^E = n^2$

Case 1:- $f(n) \in O(n^{E-\varepsilon})$

$\Rightarrow n^3 \in O(n^{2-\varepsilon}) \Rightarrow$ Case 1 failed

We cannot find a +ve ε that satisfies the condition

Case 2:- $f(n) \in \Theta(n^E)$

$\Rightarrow n^3 \in \Theta(n^2) \Rightarrow$ Case 2 failed

Case 3:- $f(n) \in \Omega(n^{E+\varepsilon})$

$\Rightarrow n^3 \in \Omega(n^{2+\varepsilon})$ ~~Case 3 passed~~ $\Rightarrow$ Accepted $\Rightarrow \varepsilon = 1$

~~+TH~~ $f(n) \in O(n^{E+\delta})$

$\Rightarrow n^3 \in O(n^{2+\delta})$, $\delta > \varepsilon \Rightarrow$ Case 3 passed $\Rightarrow \delta = 1$

$\therefore T(n) \in \Theta(n^3)$

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

ans. $b=4, c=2, f(n)=n$ $E = \log b / \log c = \log 4 / \log 2 = 2$

URBAN EDGE 12/2/20

Case 1:- $f(n) \in O(n^{1+\varepsilon})$

$\Rightarrow n \in O(n^{2-\varepsilon})$

$\varepsilon = 1 \Rightarrow n \in O(n^1) \Rightarrow$ Case 1, accepted

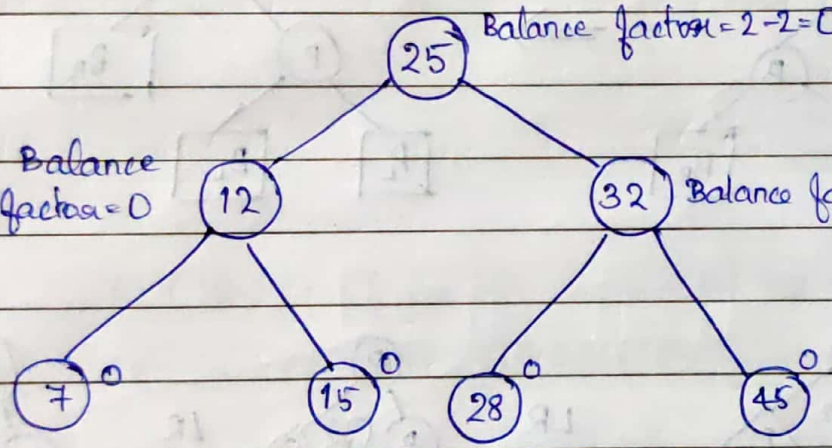
$\therefore T(n) \in O(n^2)$

** Imp Some qn. 10 or 8 marks

AVL Trees

Height-balanced tree

Balance factor = | Height of left subtree − Height of right subtree |

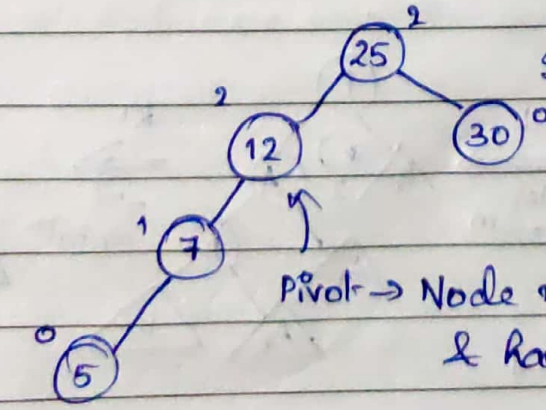


If balance factor = 0 or 1 or −1 then the tree is an AVL tree

If a single node has balance factor other than 0 or 1 or −1, then it is not an AVL tree

18/2/20

25, 30, 12, 7, 5



Left child, Left subtree

LL Rotation

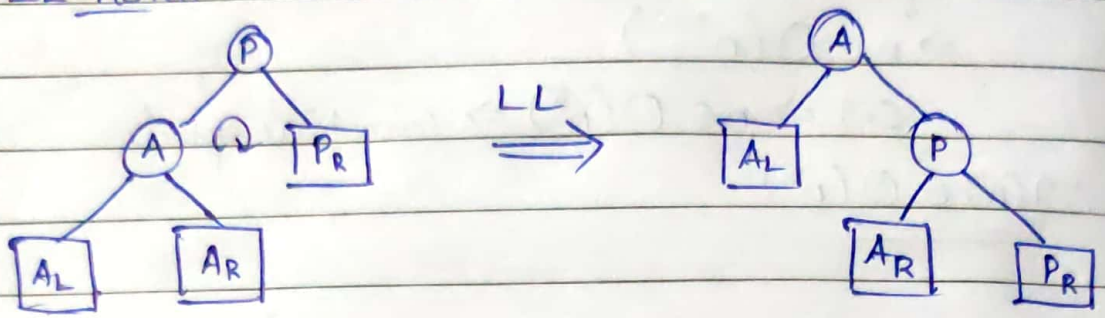
Single: LR "

RR "

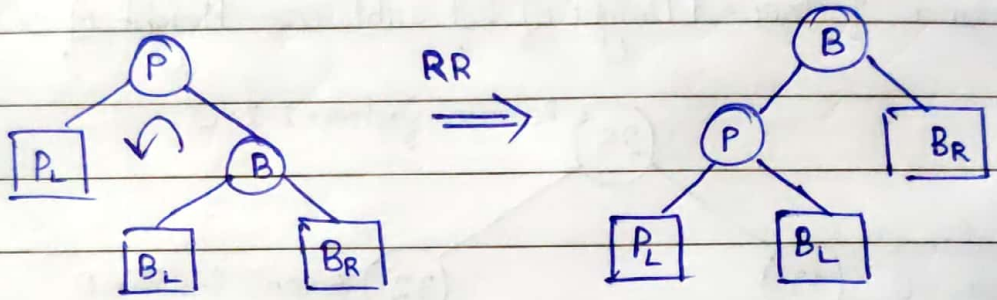
Double: RL "

Pivot → Node nearest to the newly inserted node & having non-permissible balance factor

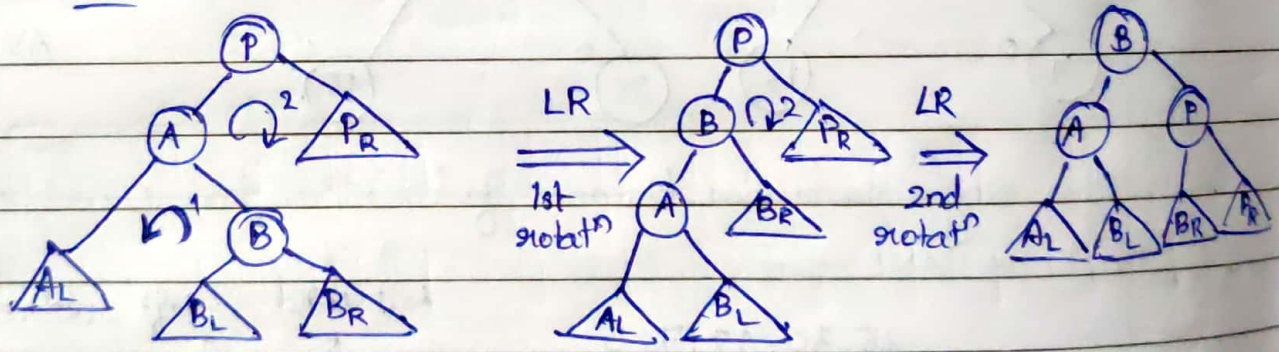
(i) LL Rotatn



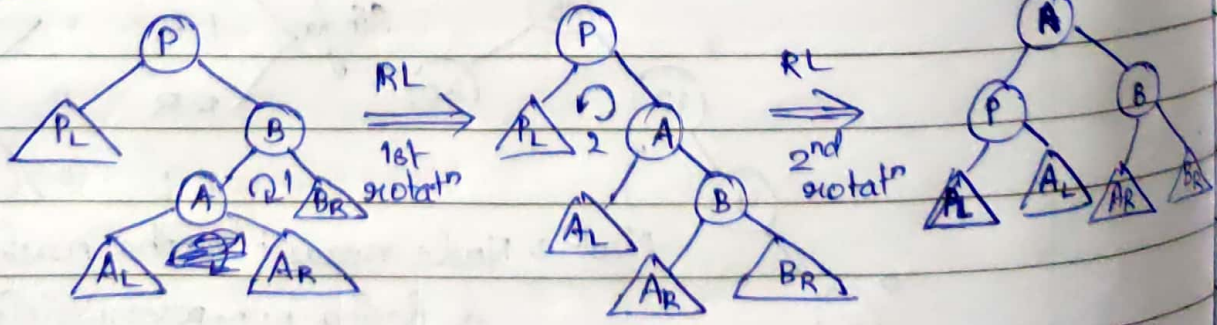
(ii) RR Rotatn



(iii) LR Rotatn



(iv) RL Rotatn



URBAN EDGE 19/2/20

AVL Rotat^n's

- In 1962 2 Russian mathematicians G I M Adelson-Velskii & E M Landis
- So AVL rotat^n
- 4 cases of rotat^n's

B-Trees

- m-way search tree

→ An m-way search tree T is a tree in which all nodes are of degree <= m

→ Each node in the tree contains the following attributes

P_0	K_1	P_1	K_2	P_2	K_n	P_n

where $1 <= n <= m$,

K_i $(1 <= i <= n)$ are key values in the node

P_i $(0 <= i <= n)$ " pointers to subtrees

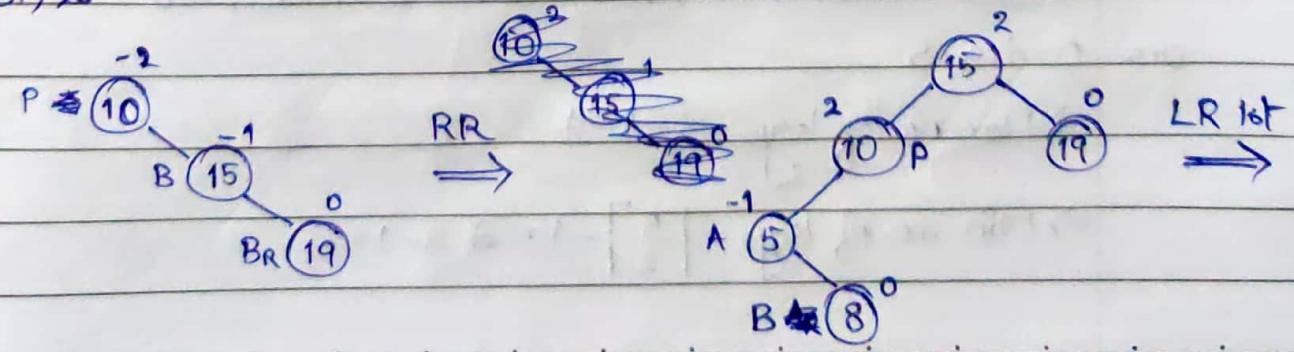
- A B-Tree T of order m is an m-way search tree that is either empty, or it satisfies the following properties

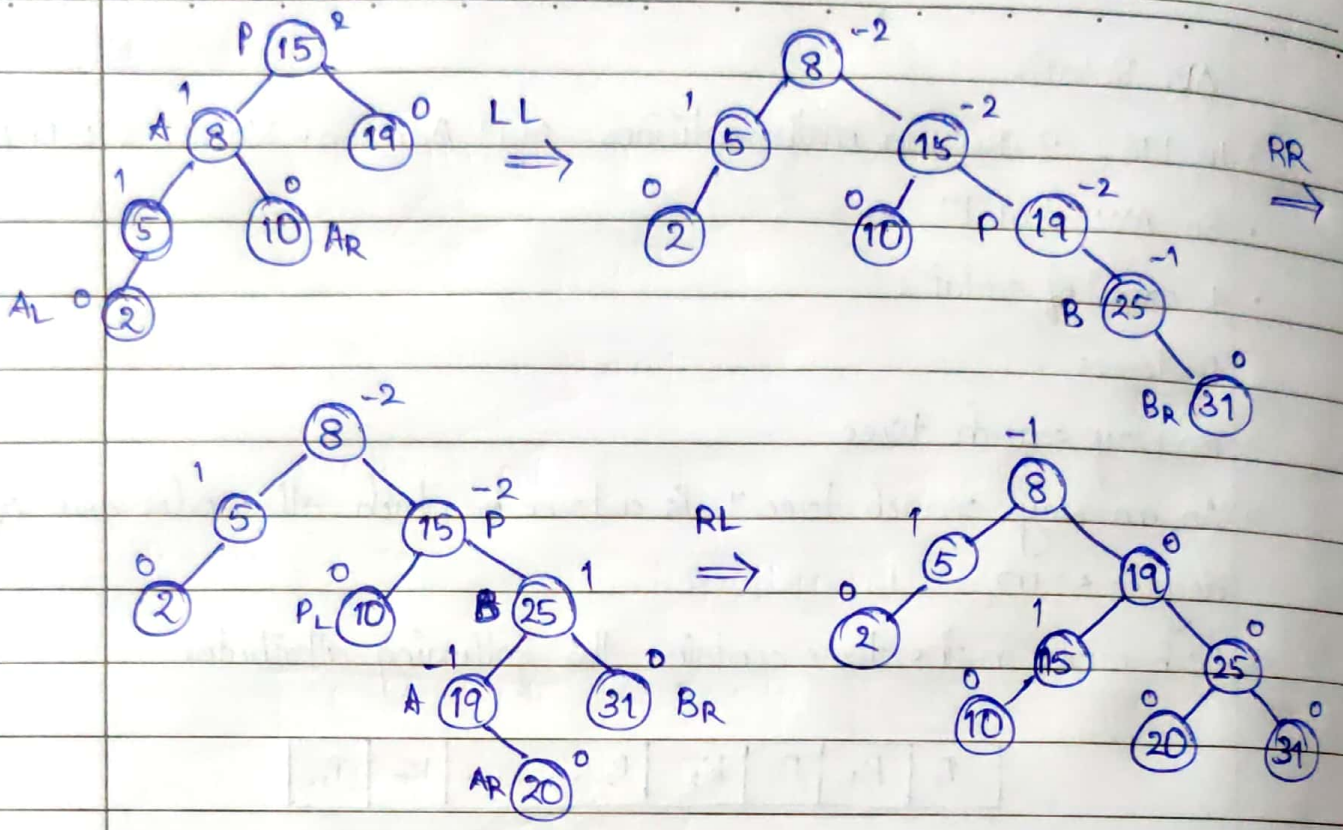
→ Root node is a leaf or has atleast 2 children

→ Failure nodes are at same level

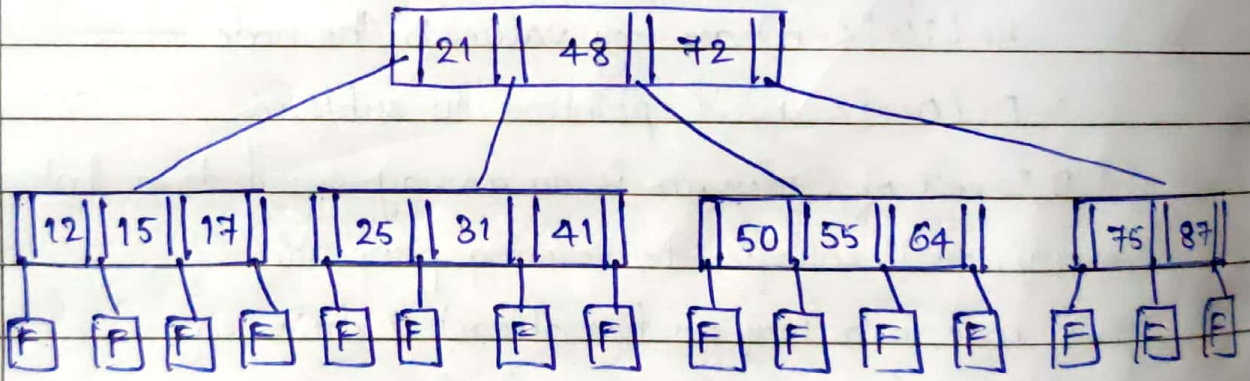
① Build an AVL tree using the following values: 10, 15, 19, 5, 8, 2, 25, 31, 20

ans





B-Trees ~~B-Trees~~



① Create a B-Tree of order 5 with values: 10, 20, 50, 60, 40, 80, 100, 70, 130, 90, 30, 120, 140, 25, 35, 160, 180

ans: Order = 5

⇒ Max no. of keys = 5 - 1 = 4

⇒ Min. no. of keys = $\lceil \frac{5}{2} \rceil - 1$ = ~~2~~ 3 - 1 = 2

URBAN EDGE 19 / 2 / 20

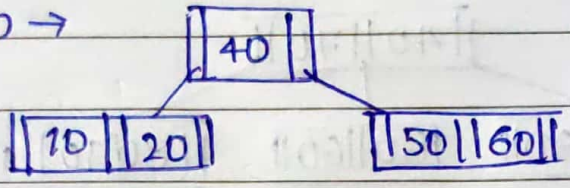
Insert 10 → [10]

Insert 20 → [10 | 20]

Insert 50 → [10 | 20 | 50]

Insert 60 → [10 | 20 | 50 | 60]

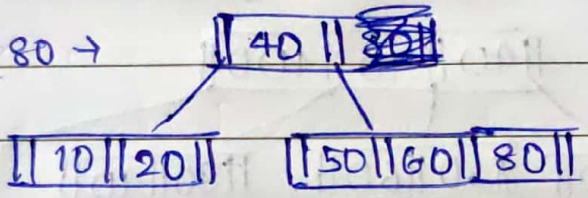
Insert 40 →



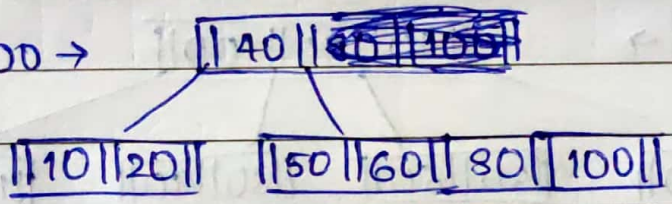
10 | 20 | 40 | 50 | 60

Take median & push 1 level up

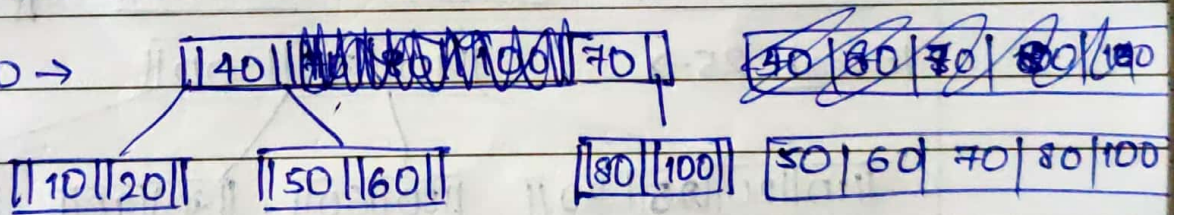
Insert 80 →



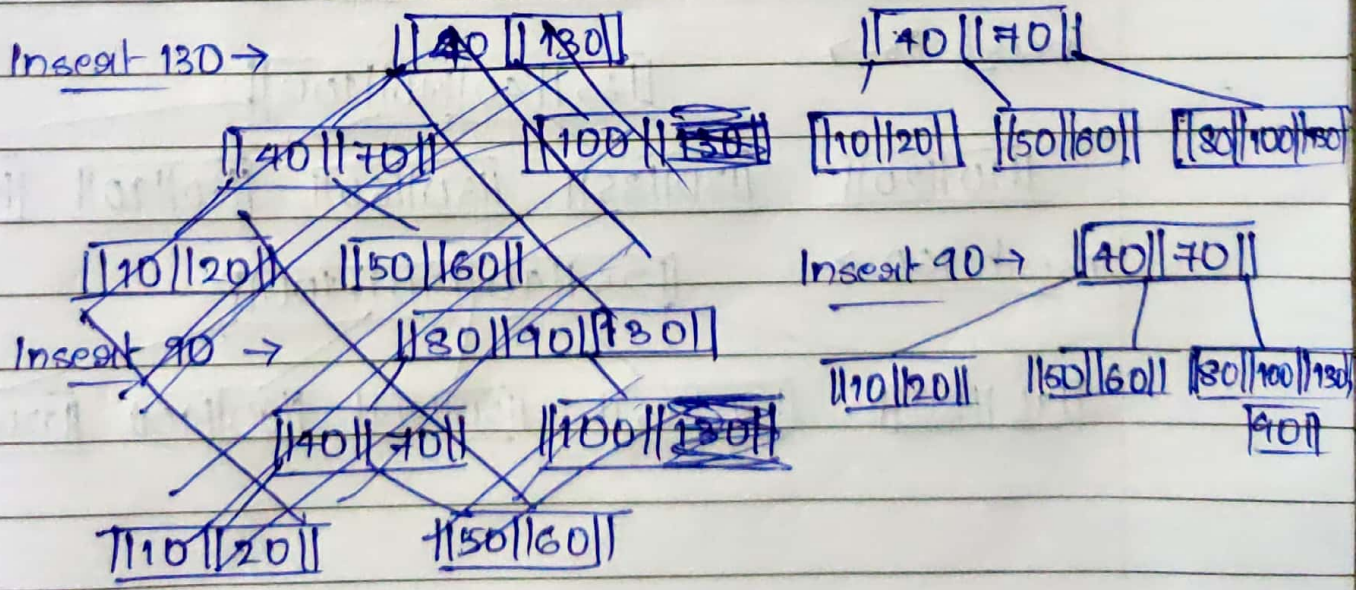
Insert 100 →



Insert 70 →

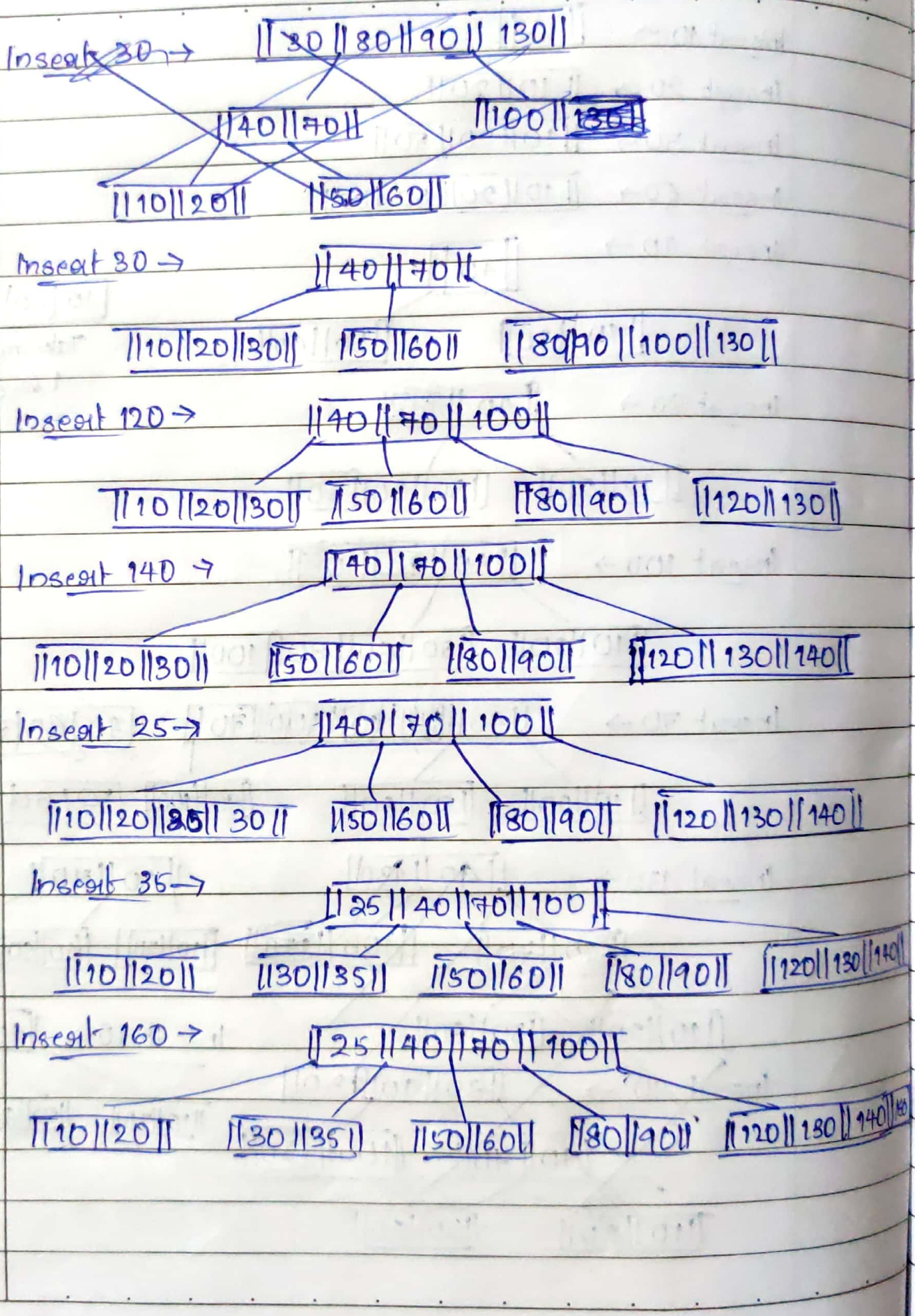


Insert 130 →



Insert 90 →

URBAN EDGE 19/2/20



URBAN EDGE 19 / 2 / 20

Insert 180 →