Date 10/08/19
Page No.

Algorithm

An algo. is any well-defined computational procedure that takes some I/P & produce some O/P.

In other words, we can also say algo. is a sequence of computational steps that transform I/P into O/P.

Algorithm Approach

There are 5 approaches :-

1. Incremental Approach — Insertion sort, Selection sort.
2. Divide & Conquer Approach - Quick, Merge, Heap.
3. Greedy Approach - SSSP (Dijkastra, Bellmanford), Knapsack
4. Dynamic Approach - Floyd Warshall's
5. Back-tracking Approach - N-queen, Sum of Subsets

21/08/19

Asymptotic Notation

are used to describe time complexity of the algo.

There are 3 notation that generally used :-

O → Big Oh - Upper bound - Worst Case.
Ω → Big Omega - Lower bound - Best Case
Φ → Theta - Average bound - Average.

Analysis of the Algorithm

Analyzing the algo. has come to mean predicting the resources that the algo. requires

Occasionally resources such as memory, communicational bandwidth or computer

Scanned by CamScanner

Time complexity → Nature of I/P.
of insertion sort

Date / /
Page No.

hardware but most often it is computational time that we want to measure.

22/08/19

Insertion Sort (A, n)

		Cost	Time.
1.	for j = 2 to length[A]	C_1	n
	{		~~(n-1)~~
2.	key ← A[j]	C_2	$(n-1)$
3.	i = j - 1.	C_3	$\sum_{j=2}^{n} t_j \; (n-1)$
4.	while i > 0 && A[i] > key	C_4	$\sum_{j=2}^{n} (t_j)$
	{		
5.	A[i+1] ← A[i]	C_5	$\sum_{j=2}^{n} (t_j - 1)$
6.	i--;	C_6	
	}		
7.	A[i+1] ← key	C_7	$(n-1)$
	}		

$$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4\left(\sum_{j=2}^{n} t_j\right) + C_5 \sum_{j=2}^{n} (t_j - 1) + C_6\left(\sum_{j=2}^{n} t_j - 1\right) + C_7(n-1)$$

Best Case:

$$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_7(n-1)$$

$$= C_1 n + C_2 n - C_2 + C_3 n - C_3 + C_7 n - C_7.$$

$$T(n) = \underbrace{n(C_1 + C_2 + C_3 + C_7)}_{a} - \underbrace{(C_2 + C_7)}_{b}$$

Scanned by CamScanner

$T(n) = n \cdot a + b$.

$T(n) = a \cdot n + b$.

$\boxed{T(n) = \Omega(n)}$

Worst case:

$$T(n) = C_1 n + C_2(n-1) + C_3(n-1) + C_4\left(\frac{n(n+1)}{2} - 1\right) + C_5\left(\frac{n(n+1)}{2}\right) + C_6\left(\frac{n(n-1)}{2}\right) + C_7(n-1)$$

$$= C_1 n + C_2 n - C_2 + C_3 n - C_3 + C_4\left(\frac{n^2+n-2}{2}\right) + C_5\left(\frac{n^2-n}{2}\right) + C_6\left(\frac{n^2-n}{2}\right) + C_7 n - C_7.$$

$$= C_1 n + C_2 n - C_2 + C_3 n - C_3 + \frac{C_4 n^2}{2} + \frac{C_4 n}{2} + C_4 + \frac{C_5 n^2}{2} - \frac{C_5 n}{2} + \frac{C_6 n^2}{2} - \frac{C_6 n}{2} + C_7 n - C_7.$$

$$= \frac{n^2}{2}\left[C_4 + C_5 + C_6\right] + n\left[C_1 + C_2 + C_3 + \frac{C_4}{2} + \frac{C_5}{2} - \frac{C_6}{2} + C_7\right] - \left[C_2 + C_3 - C_4 + C_7\right].$$

$$= \frac{n^2}{2} \cdot a + n[b] + c.$$

$$= a \cdot n^2 + b.$$

Scanned by CamScanner

$= b + an^2$

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

Avg. case of Insertion sort is like the worst case.

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

23/08/19

Analysis of the selection sort:

The selection sort's algo. idea is based on to finding the min./max element in unsorted array & then it is put at the correct position in a sorted array

Selection sort (A).

		Cost	time
1.	$n \leftarrow$ length [A]	C_1	1
2.	for $j \leftarrow 1$ to $(n-1)$	C_2	n
3.	do smallest $\leftarrow j$	C_3	$(n-1)$
4	for $i \leftarrow j+1$ to n	C_4	$\sum_{i=2}^{n+1} t_j$
5.	do if $A[i] < A[\text{smallest}]$.	C_5	$\sum_{i=2}^{n+1}(t_j-1)$
6.	then smallest $\leftarrow i$.	C_6	"
7.	Exchange $A[j] \leftrightarrow A[\text{smallest}]$	C_7	$(n-1)$

$t_j = (n-1) + (n-2) + \cdots 1.$

$= \frac{(n^2-n)}{2}$

$T(n) = C_1 + C_2 n + C_3(n-1) + C_4 \sum_{i=2}^{n+1} t_j + C_5 \sum_{i=2}^{n-1}(t_j-1) + C_6 \sum_{i=2}^{n-1}(t_j-1) + C_7(n-1).$

Scanned by CamScanner

The time complexity of selection sort in all the cases is $O(n^2)$.

Best case - $\Omega \rightarrow O(n^2)$
Worst case - $O \rightarrow O(n^2)$
Avg case - $\phi \rightarrow O(n^2)$.

Ques Execute the data using selection sort step by step
5, 2, 1, 4, 3.

Ques Show the operation of insertion sort step by step.
12, 56, 34, 21, 67, 89, 57.

i →	1	2	3	4	5	6	7.
A -	12	56	34	21	67	89	57

Step 1 Hence, n = 7, i = 1, j = 2, key, A[j] = 56.

while i > 0 && A[i] > key
A[i+1] ← A[i], i--;

→ 1 > 0 && 12 > 56 . false.
A[i+1] ← key
A[1+1] ← 56
A[2] ← 56.

12	56	34	21	67	89	57

Scanned by CamScanner

Step 2

while $i \geq 0$ && $A[i] > key$
$A[i+1] \leftarrow A[i]$, $i--$;

where, $i = 2$, $j = 3$, key, $A[j] = 56, 34$.

→ $2 \geq 0$ && $56 > 34$. true.
$A[2+1] \leftarrow A[2]$
$A[3] \leftarrow 56$

| 12 | 34 | 56 | 21 | 67 | 89 | 57 |

Step 3.

$i = 3$, $j = 4$, key $A[j] = 21$

while $i \geq 0$ && $A[i] > key$
{ $A[i+1] \leftarrow A[i]$, $i--$;
}
$A[i+1] \leftarrow key$.

$3 \geq 0$ && $56 > 21$. true.
$A[3+1] \leftarrow A[3]$
$A[4] \leftarrow 56$.

| 12 | 34 | 21 | 56 | 67 | 89 | 57 |

key, $A[j] \leftarrow 34$

$2 \geq 0$ && $21 > 34$. false.
$A[i+1] \leftarrow key$
$A[3] \leftarrow 34$.

| 12 | 21 | 34 | 56 | 67 | 89 | 57 |

Scanned by CamScanner

Step 4 $i = 4, j = 5, key \leftarrow A[j] = 67$

while $i \geq 0$ && $A[i] > key$

{ $A[i+1] \leftarrow A[i]$; $i--$

}

$A[i+1] \leftarrow key$

$4 \geq 0$ && $56 > 67$. False.

$A[4+1] \leftarrow 67$

$A[5] \leftarrow 67$.

| 12 | 21 | 34 | 56 | 67 | 89 | 57 |

Step 5: $i = 5; j = 6, key \leftarrow A[j] = 89$.

$5 \geq 0$ && $67 > 89$. False.

$A[i+1] \leftarrow key$

$A[6] \leftarrow 89$.

| 12 | 21 | 34 | 56 | 67 | 89 | 57 |

Step 6 $i = 6; j = 7, key \leftarrow A[j] = 57$.

$6 \geq 0$ && $89 > 57$. True.

$A[i+1] \leftarrow key$

$A[7] \leftarrow 57$.

| 12 | 21 | 34 | 56 | 67 | 57 | 89 |

$i = 5, j = 6; key \leftarrow A[j] = 57$.

$5 \geq 0$ && $67 > 57$. True.

Scanned by CamScanner

A[5+1] ← ~~key~~ A[5].

A[6] ← 67.

| 12 | 21 | 34 | 56 | 57 | 67 | 89 |

Selection sorting: 5, 2, 1, 4, 3.

1	2	3	4	5
5	2	1	4	3

Step 1.

for i ← j+1 to 5.
  if A[i] < A[smallest]
    smallest ← i.
Exchange A[j] ↔ A[smallest]

i = 1, j = 1, n = 5., A[j] = 5

for i ← 1+1
  if A[2] < A[1]
    2 < 5. true.
    2 ← i
A[j] ↔ A[smallest]
A[j] → 2.

2	5	1	4	3

Scanned by CamScanner

Step 2 $i = 3$, $j = 2$, i

if $A[i] < A[j]$

$1 < 5$ . . true.

$A[j] \leftrightarrow A[i]$

| 5 | 1 | 2 | 4 | 3 |

| 2 | 1 | 5 | 4 | 3 |

$A[i] < A[j]$. true.

$A[j] \leftrightarrow A[i]$.

| 1 | 2 | 5 | 4 | 3 |

Step 3 $j = 3$; $i = 4$;

if $A[j] < A[j]$

4 5 < 5 . true

$A[j] \leftrightarrow A[i]$

| 1 | 2 | 4 | 5 | 3 |

Step 4 $j = 4$, $i = 5$.

$A[i] < A[j]$

$3 < 5$. true.

$A[i] \leftrightarrow A[j]$.

| 1 | 2 | 4 | 3 | 5 |

Scanned by CamScanner

Steps:

$j = 3$; $i = 4$,

$A[i] < A[j]$

$3 < 4$. True.

$A[i] \leftrightarrow A[j]$

| 1 | 2 | 3 | 4 | 5 |

30/08/19

Divide & Conquer Approach.

works by recursively breaking down a problem in two of the same sub problems, until this becomes simple enough to understand

Analysis of D&C approach.

D & C algo. proceed as :-

1. Divide the I/P problem into sub problems
2. Conquer on sub-problems.
3. Combine the solution of the sub-problems to obtain the sol.n of original problem.

The analysis of algorithm may be carried out by the several methods. Some of them are as :-

1. Guess method
2. Substitution method.
3. Recursion tree
4. Master Method. (Cook-book method)

Scanned by CamScanner

Quick Sort (L, i, j)

1. i ← Pivot
2. K ← Partition (L, Pivot, j)
3. quick sort (L, Pivot, K-1)
4. quick sort (L, K+1, j)

Partition (L, m, n)

```
Pivot ← m
Pivot Element ← L[m]
  x ← m+1
  y ← n.
while (x ≤ y)
{
  while (L[x] < Pivot Element).
    x++;
  while (L[y] > Pivot Element)
    y--;
  if (x ≤ y)
  Exchange (L[x], L[y])
  Else
  Exchange (L[y], Pivot Element)
  return y;
```

Ques Apply the quick sort with the technique of divide & conquer.
19, 10, 25, 11, 13, 6, 9, 4, 26.

Scanned by CamScanner

1	2	3	4	5	6	7	8	9
19	10	25	11	13	6	9	4	26

P.E ↑ x ↑ y ↑

$n = 9$, P.E = 19, $A[x] = 10$, $A[y] = 26$

$x = 2$, $y = 9$.

while $(x < y)$; $x = 2 \Rightarrow$; $x = 3$

while (y 10 < 19.

26 > 19, $y = 9, 8$.

4 > 19

if $(x < y)$.

$L[x] \longleftrightarrow L[y]$

$\Rightarrow L[3] = 4$; $L[8] = 25$.

19	10	4	11	13	6	9	25	26

while $(x < y)$; $x = 4$ & & $= 8$.

11 < 19.

13 < 19

6 < 19

9 < 19

25 < 19.

$x = 8.7$

4 > 19. ; $y = 8$.

$x = 7$, $y = 8$.

Scanned by CamScanner

Analysis of the Quick Sort.

B.C = $T(n) = \Omega(n \log n) \Rightarrow T(n/2) + T(n/2) + n \Rightarrow 2T(\frac{n}{2}) + n$

W.C = $T(n) \rightarrow O(n^2)$

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

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

Scanned by CamScanner

Heap sort.

It is the sorting technique based on binary heap data structure. It is similar to selection sort where we first find the max. element and place the max. element at the end.

Algo.

1. First, build the max heap.
2. At this point, the largest element at the root, replace it with the last element of the tree.
   Repeat this step, until the size of Decrease the tree by 1.
   Repeat step 2, till the size of tree is greater than 1.

Ques Show the operation of partition using Quick sort on following data set.

25, 20, 17, 40, 18, 55, 12, 21, 30, 48

1	2	3	4	5	6	7	8	9	10
25	20	17	40	18	55	12	21	30	48

n = 10, P.E = 25, x = 2 ; y = 10.
A[x] = 20 ; A[y] = 48.

while (x ≤ y)
{
20 < 25
17 < 25
40 < 25. false.

x = 2 3 4.

Scanned by CamScanner

y = 10 9 8.

48 > 25. True

30 > 25. True

21 > 25. false.

if. $(x \leq y)$ $A[x]$ ⇒ 4 ≤ 8. : True.

Exchange $A[y], A[y]$.

A[4] ↔ A[8].

1	2	3	4	5	6	7	8	9	10
25	20	17	21	18	55	12	40	30	48

n = 10, x = 4, y = 8.

x = 4 ≯ 6

21 < 25. True

18 < 25. True

55 < 25. false

y = 8 7.

40 > 25. True

12 > 25. false.

6 < 7. . True.

A[6] ↔ A[7]

1	2	3	4	5	6	7	8	9	10
25	20	17	21	18	12	55	40	30	48

x = 6 ; y = 7. ; n = 10.

x = 6 7.

12 < 25. True.

55 < 25. false.

y = 7 6 5

55 ≯ 25. True

12 > 25. True

y = 7 6.

55 > 25. True

12 > 25. false.

Scanned by CamScanner

$x = 7$, $y = 6$.

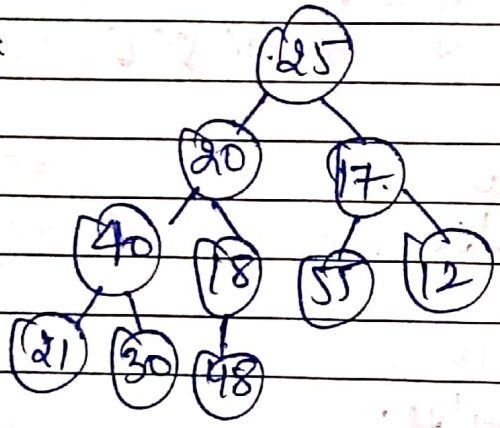
$7 < 6$. false.

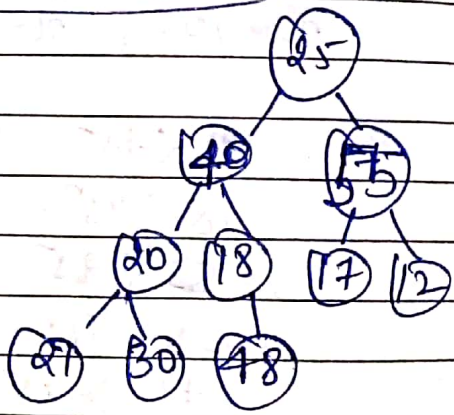
$A[y]$. $\leftrightarrow$ Pivot element .

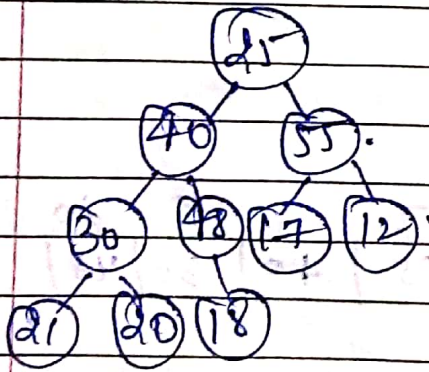
12	20	17	21	18	25	55	40	30	48

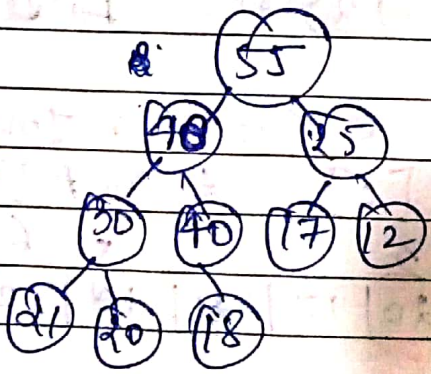
Ques. Perform the operation of heap sort on above data.

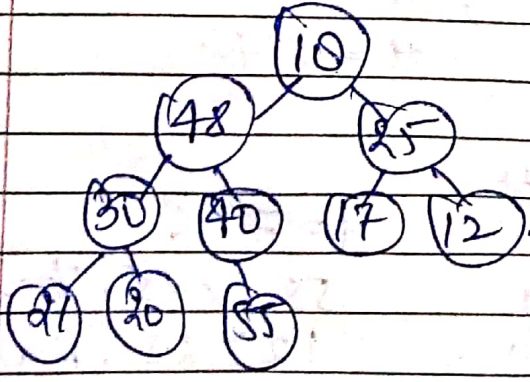
1	2	3	4	5	6	7	8	9	10
25	20	17	40	18	55	12	21	30	48





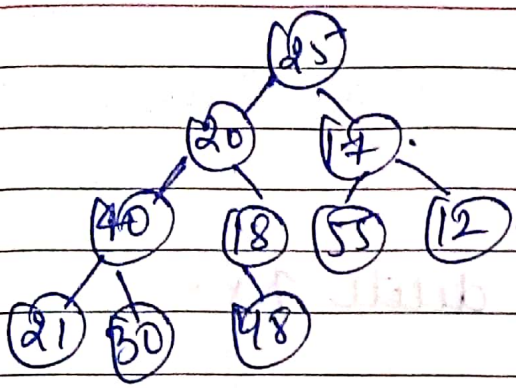


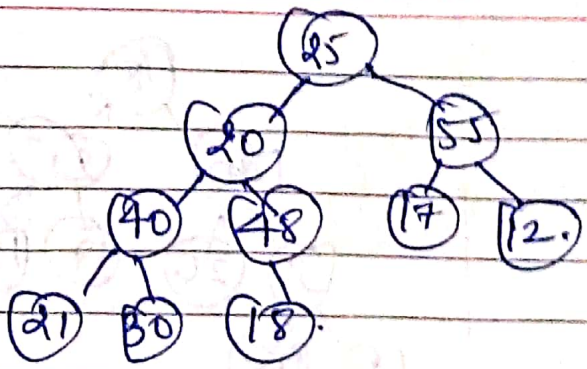


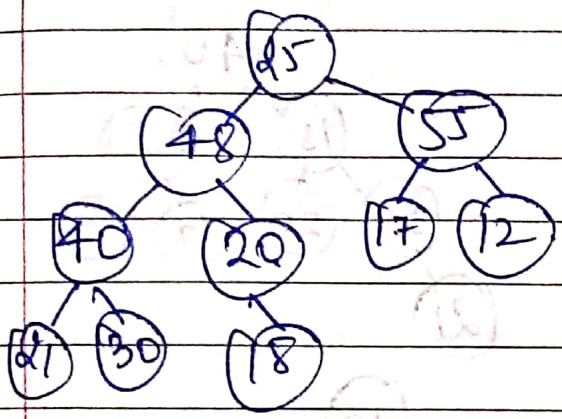


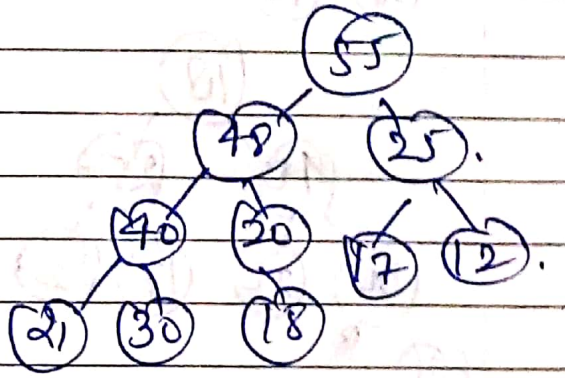
delete 55.

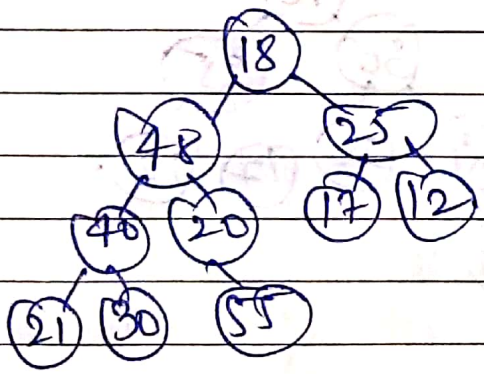
Scanned by CamScanner



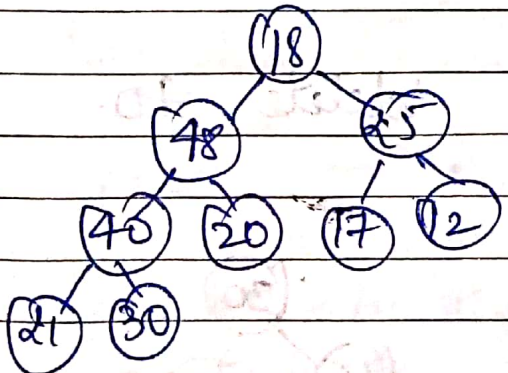


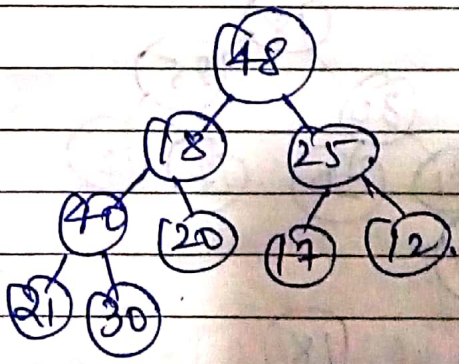


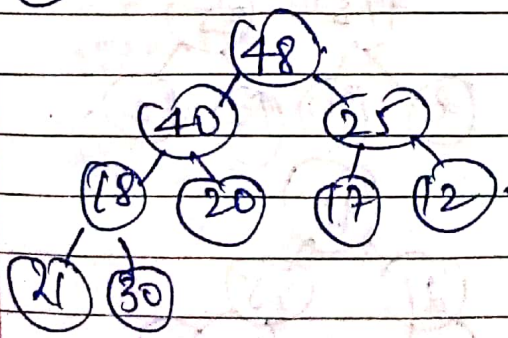


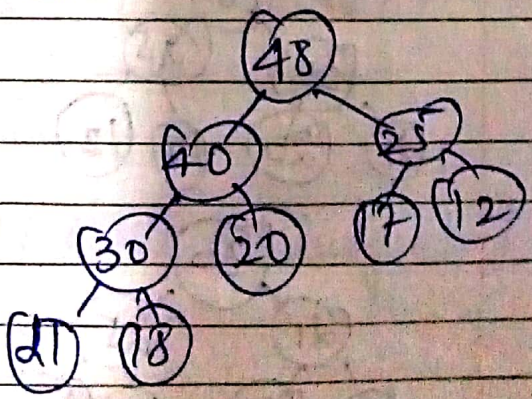


Delete 55.

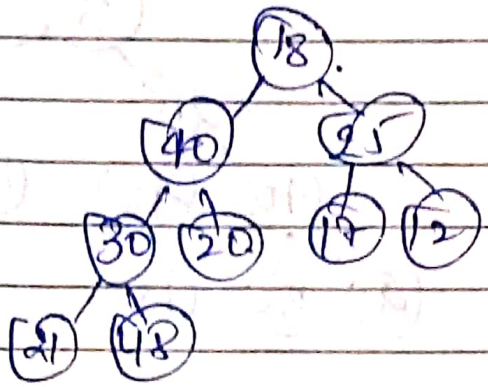




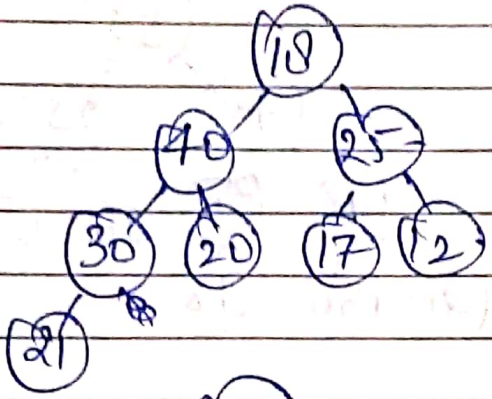


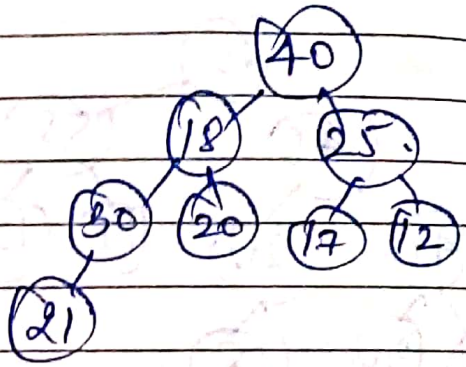


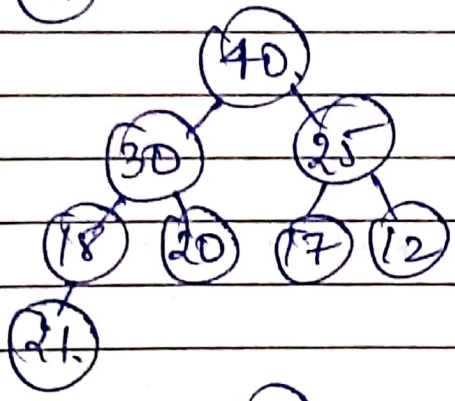
Scanned by CamScanner

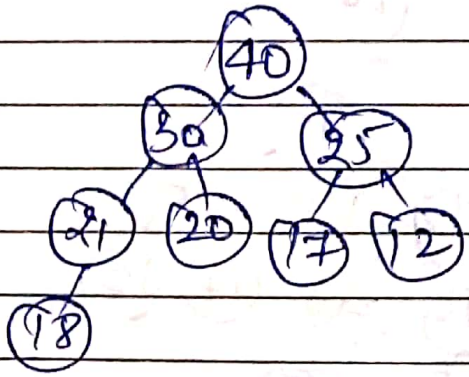


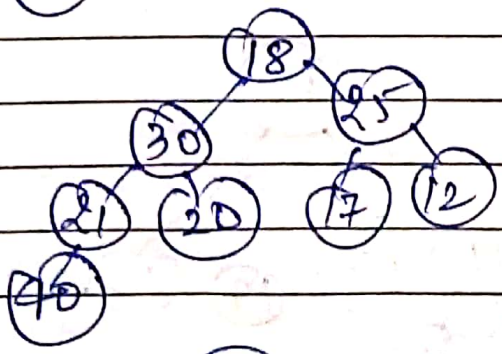
delete 48.



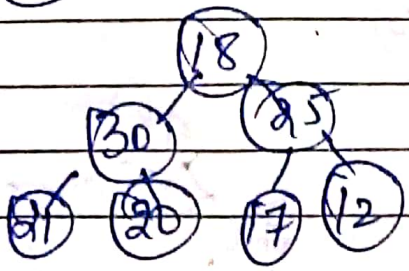


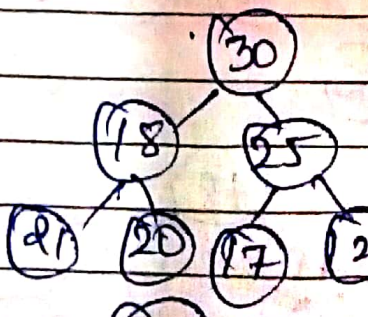


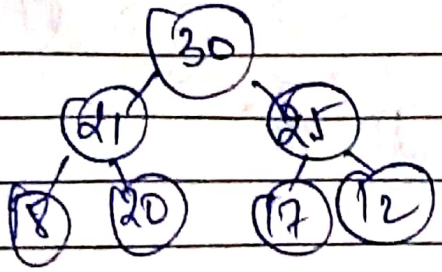


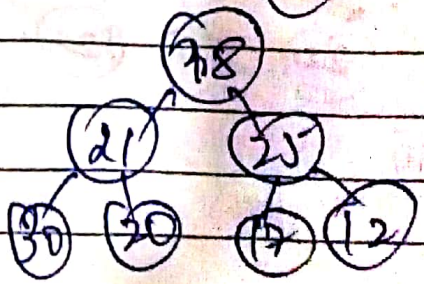


delete 40.



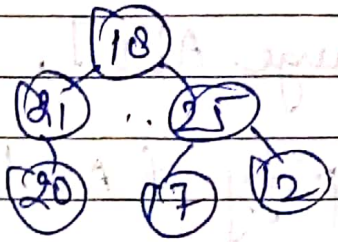


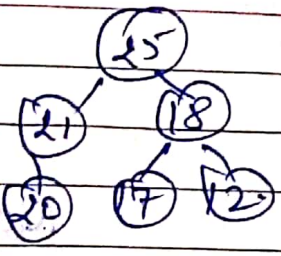


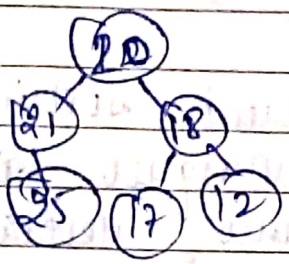


Scanned by CamScanner

delete 30.







Algorithm of the heap sort. (bid max)

Max-Heapify (A, i)

1. l ← left (i)
2. r ← right (i).
3. if l <= A.size && A[l] > A[i]
   largest ← l.
   else
   largest ← i.
   if r <= A.size && A[r] > A[largest]
   largest ← r
   else
   largest ←

Scanned by CamScanner

Exchange A[i], A[largest]

Max-heapify (A, largest).

5/09/19

Merge sort

is one of the most efficient sorting algo. works on divide & conquer principle, breakdown a list into several sub-lists, until each sublist consist of the single element and merging those sublist in a manner that results into a sorted list.

Algo. for Merge sort (A, l, r)

if $r > l$

1. find the middle point to divide the array into two halves
   m = middle = $\frac{(l+r)}{2}$
2. Call merge sort for first half
   Merge sort (A, l, m)
3. Call merge sort for second half
   Merge sort (A, m+1, r)
4. Merge the two halves sorted into step 2 & 3.
   Call merge (A, l, r).

Scanned by CamScanner

Performance Analysis:

Let given input size = n

then, after partition the size of sublist = $n/2$

so, $n = n/2 + n/2$

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

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + \theta(n)$

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

i.e., the solution of $T(n) = O(n \log n)$.

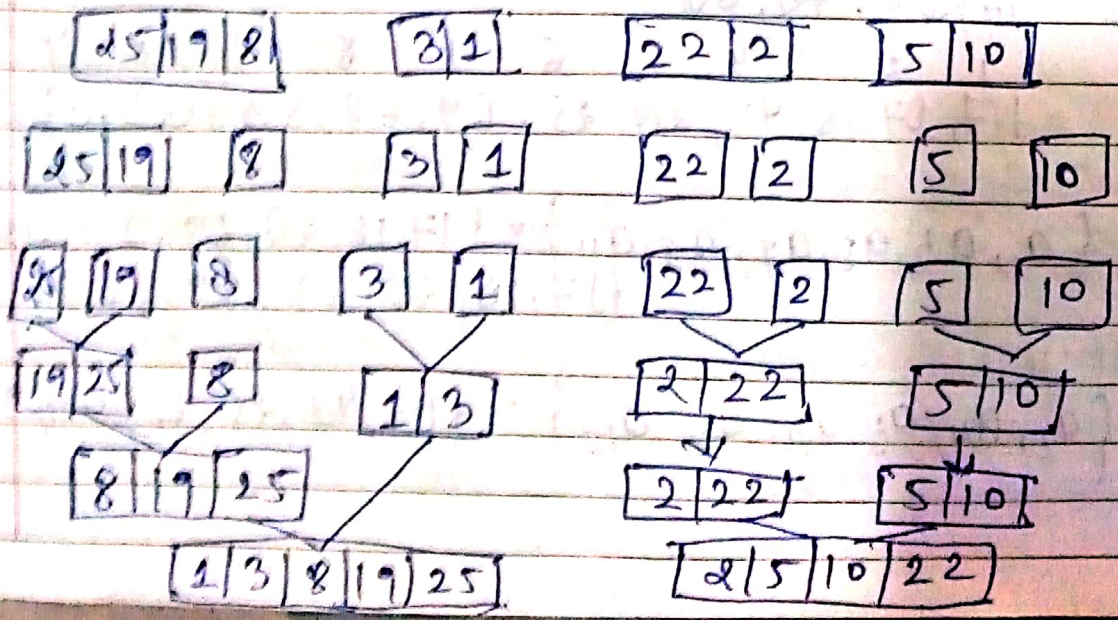
Ques Show the operation of the merge sort on the following array.

25, 19, 8, 3, 1, 22, 2, 5, 10.

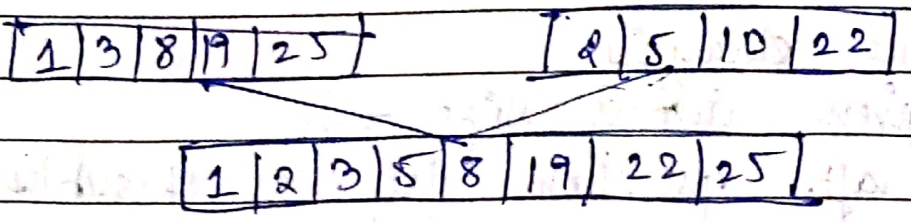
1	2	3	4	5	6	7	8	9
25	19	8	3	1	22	2	5	10

$m = \frac{1+9}{2} = \frac{10}{2} = 5$

sublist I- [25|19|8|3|1] [22|2|5|10] – sublist II.



Scanned by CamScanner



Shell Sort:

a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}

62, 83, 18, 53, 07, 17, 95, 86, 47, 69, 25, 28

5 → $\{a_1, a_6, a_{11}\} = \{62, 17, 25\} \Rightarrow \{17, 25, 62\}$

3 → $\{a_2, a_7, a_{12}\} = \{83, 95, 28\} \Rightarrow \{28, 83, 95\}$

1 → $\{a_3, a_8\} = \{18, 86\} \Rightarrow \{18, 86\}$

$\{a_4, a_9\} = \{53, 47\} \Rightarrow \{47, 53\}$

$\{a_5, a_{10}\} = \{07, 69\} \Rightarrow \{07, 69\}$

a_1 2 3 4 5 6 7 8 9 10 11 12

⇒ 17, 28, 18, 47, 07, 25, 83, 86, 53, 69, 62, 95.

3 → $\{a_1, a_4, a_7, a_{10}\} = \{17, 47, 83, 69\} = \{17, 47, 69, 83\}$

$\{a_2, a_5, a_8, a_{11}\} = \{28, 07, 86, 62\} = \{07, 28, 62, 86\}$

$\{a_3, a_6, a_9, a_{12}\} = \{18, 25, 53, 95\} = \{18, 25, 53, 95\}$

$\{a_4, a_8, a_{10}\} = \{47, 83, 69\} = \{47, 69, 83\}$

= 17, 07, 47, 69

1 2 3 4 5 6 7 8 9 10 11 12

17, 07, 18, 47, 28, 25, 69, 62, 53, 83, 86, 95

1 → $\{a_1, a_3, a_5, a_7, a_9, a_{11}\} = \{17, 18, 28, 69, 53, 86\}$

$= \{17, 18, 28, 53, 69, 86\}$

$\{a_2, a_4, a_6, a_8, a_{10}, a_{12}\} = \{07, 47, 25, 62, 83, 95\}$

Scanned by CamScanner

Comparison of the time complexity.

Sorting	Best Case	Avg. Case	Worst Case.
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Quick	$O(n \log n)$	$O(n^2)$	$O(n^2)$
Merge	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heap	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Linear type: Bucket	$O(n)$	$O(n)$	$O(n)$
Linear type: Radix	$O(n)$	$O(n)$	$O(n)$
Linear type: Counting	$O(n)$	$O(n)$	$O(n)$

Sorting in linear type:

⇒ Counting Sort

Ques Show the operation of counting sort on following data set :- 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2.

k → largest number given in the I/P.
size of array → 0 to k.
∴ 0 to 6.

Scanned by CamScanner

$n = 11$, $K = 6$.

	0	1	2	3	4	5	6
C =	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
C' =	2	2	2	2	1	0	2

← no. of times 0 to 6 occurs.

	0	1	2	3	4	5	6
C'' =	2	4	6	8	9	9	11

	1	2	3	4	5	6	7	8	9	10	11
R =	0	0	1	1	2	2	3	3	4	6	6

Ques 3, 6, 4, 1, 3, 4, 1, 4.

$n = 8$, $K = 6$.

	0	1	2	3	4	5	6
C =	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
C' =	0	2	0	2	3	0	1

	0	1	2	3	4	5	6
C'' =	0	2	2	4	7	7	8

	1	2	3	4	5	6	7	8
R =	1	1	3	3	4	4	4	6

Radix Sorting.

Algo.

for i = 1 to d ∴ d → digit

do stable sort

Assign Ques Perform the radix sort on the following data set :- row, dog, sea, rug, row, mob, box, tab, tab, bar, ear, tar, wig, big, tea, now.

Scanned by CamScanner

Ques. 329, 457, 657, 839, 436, 720, 355

i.e. d = 3 ; for i = 0 to 3.

	i = 1	i = 2	i = 3
329	720	720.	329
457	355	329	355
657	436	~~436~~ 355	~~355~~ 457
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657.	839.

09/09/19

R-B tree.

R-B tree is a self-balancing tree. A BST can be a R-B tree iff it obey the following properties :-

1. The colour of root will be black.
2. Every node has a colour either red or black.
3. There are no 2 consecutive red nodes (a red node cannot hold red parent or red child).
4. Every path from a node (including root) to any of its leaf nodes (null) has the same no. of the black nodes.
5. The colour of the leaf will be always black.

Note: The colour of new inserted node is red.

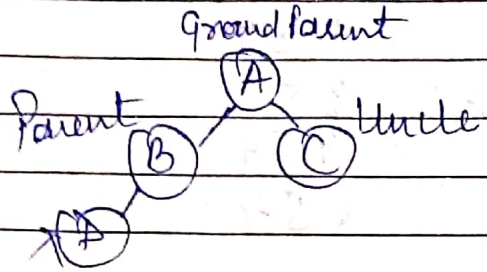
Balancing of RB tree.

In R-B tree we use two tools to do the balancing :- 1. Recolouring
2. Rotation

Scanned by CamScanner

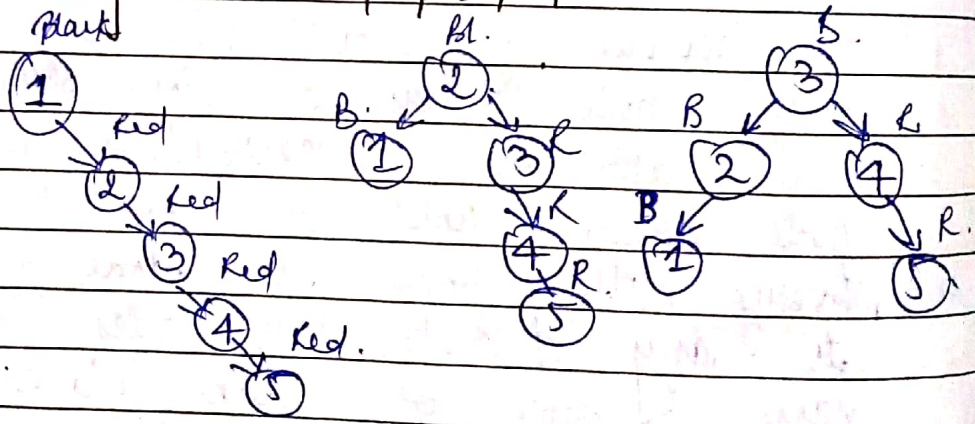
We try recolouring first, if recolouring does not work, we go for ~~the~~ rotation.

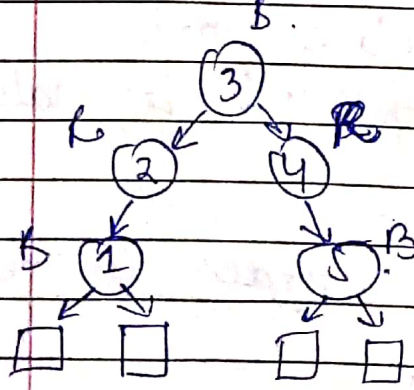
The algo. has mainly two cases depending on the colour of the uncle.



If the colour of uncle is red, we do recolouring
If colour is black then, we do rotation / recolouring

Ques RB tree for — 1, 2, 3, 4, 5.





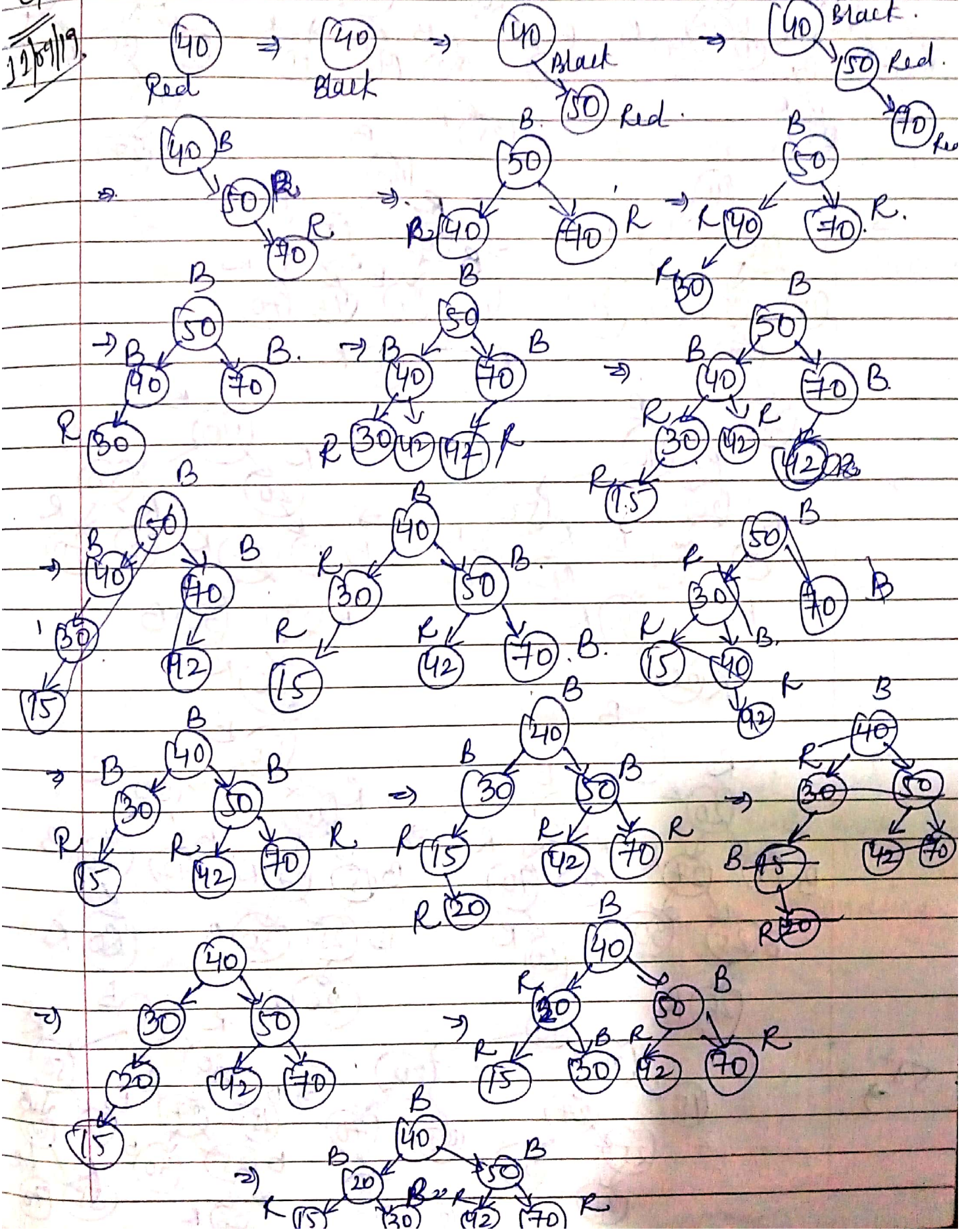
Assign Ques Create a R-B tree for the following data
40, 50, 70, 30, 42, 15, 20, 25, 27, 26, 60, 55.

Scanned by CamScanner

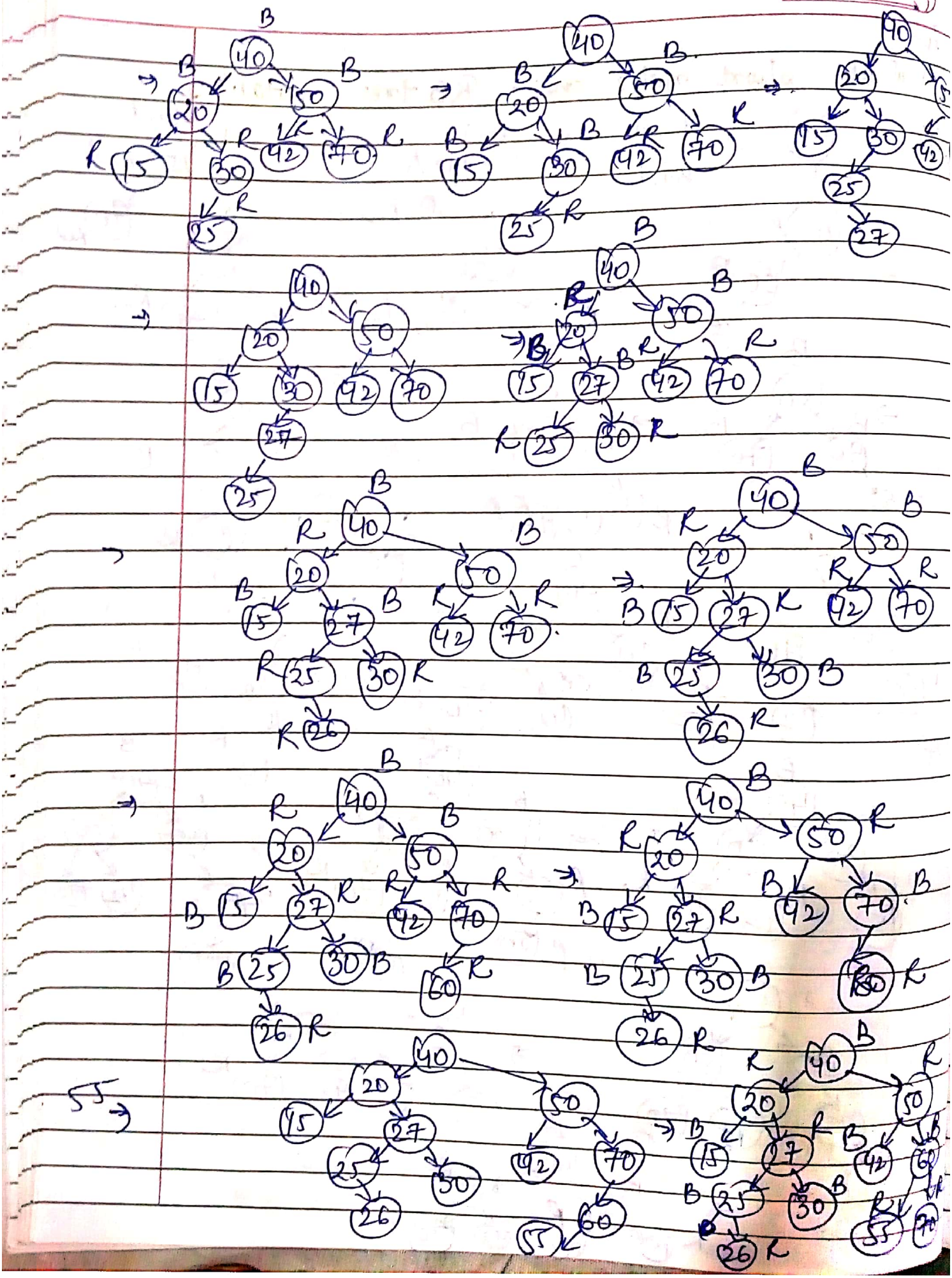
Assign

Ques Write short notes on :- RB tree rotations.

12/09/19



Scanned by CamScanner



Scanned by CamScanner

B-Tree

B-tree is the self balanced tree in which every node contains multiple keys & have more than 2 child.

Here the no. of keys in a node & no. of children for a node depends on the degree of the tree & order of the tree. Every B-tree has an order & degree

If degree = t, Then,

min. no. of keys = t-1

& max. no. of keys = 2t-1.

If order = m ; Then,

min no. of keys = m/2 -1

max no. of keys = m-1

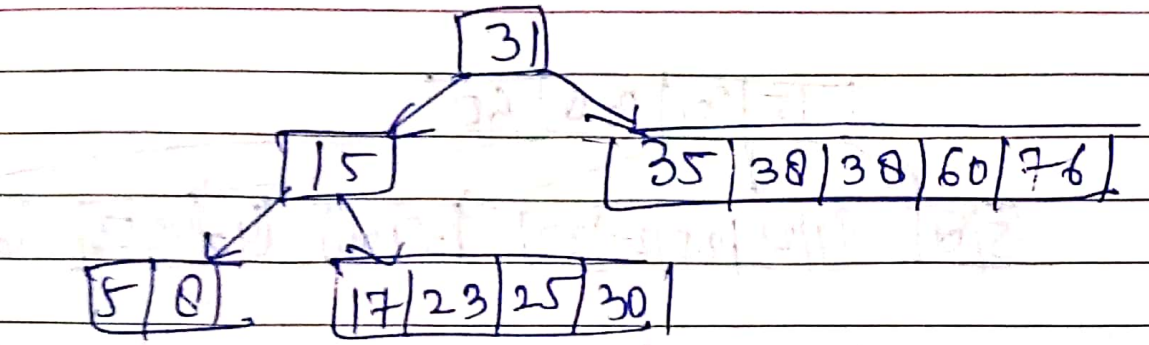
Apart from the above constraint there are following properties of B-tree :-

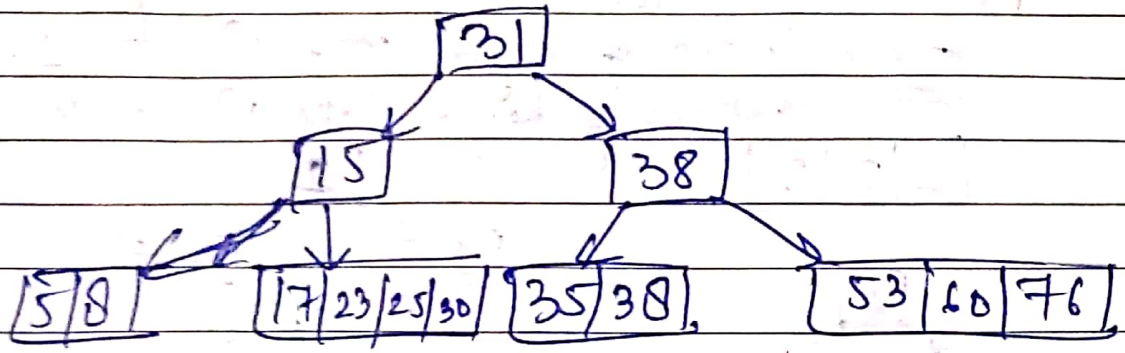
- All the leaf nodes must be at the same level.
- All nodes except the root node must have atleast (t-1) or (m/2-1) keys. and max. of (2t-1) or (m-1) keys.
- All the keys values are in ascending order

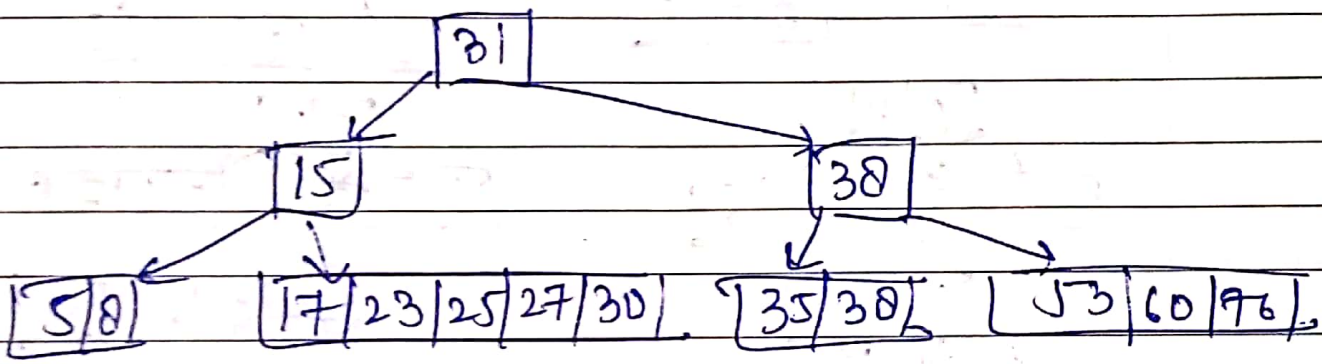
Ques Create a B-tree of following data :-
25, 31, 30, 76, 5, 60, 38, 8, 30, 15, 35, 17, 23, 53, 27, 43, 65, 40. &
Degree of tree = 3.

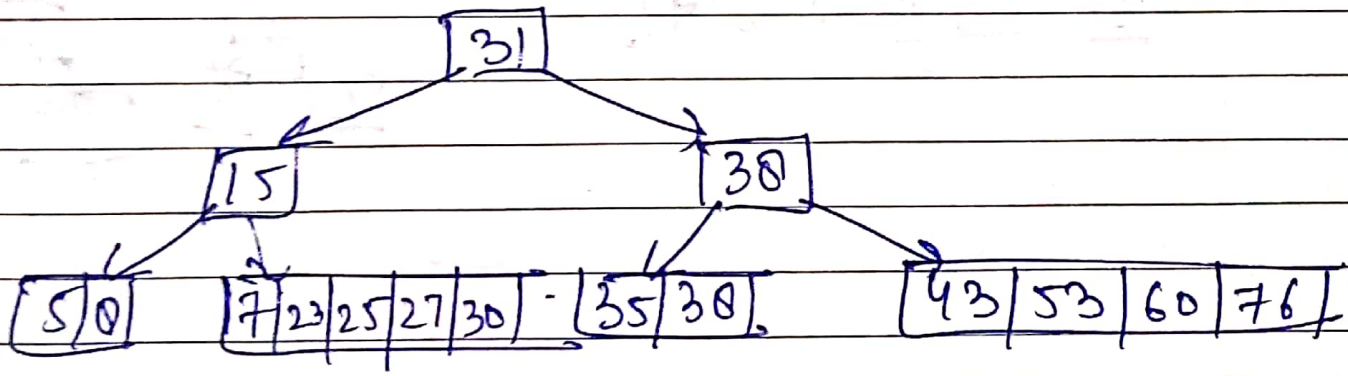
Scanned by CamScanner

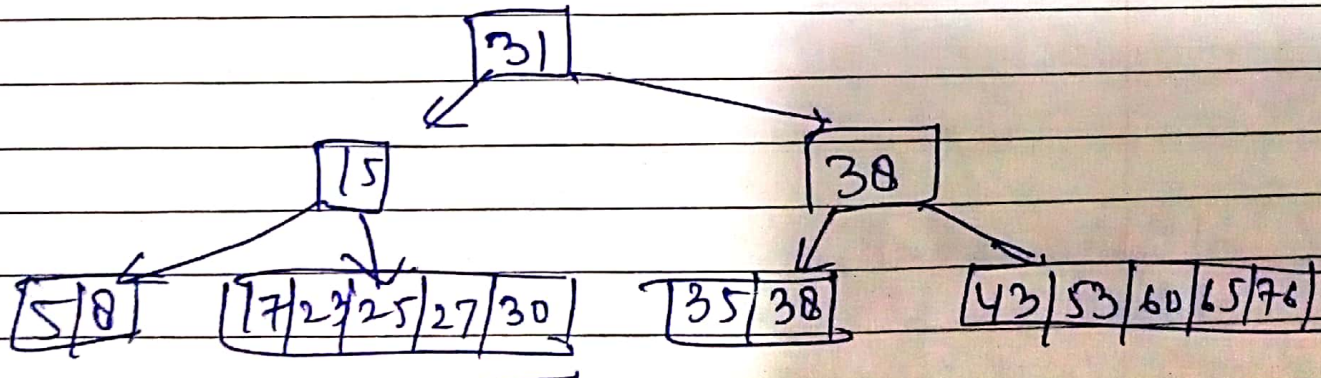
Scanned by CamScanner

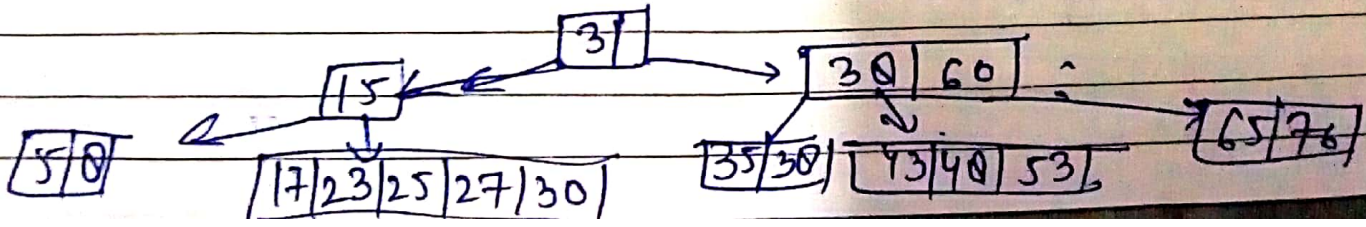




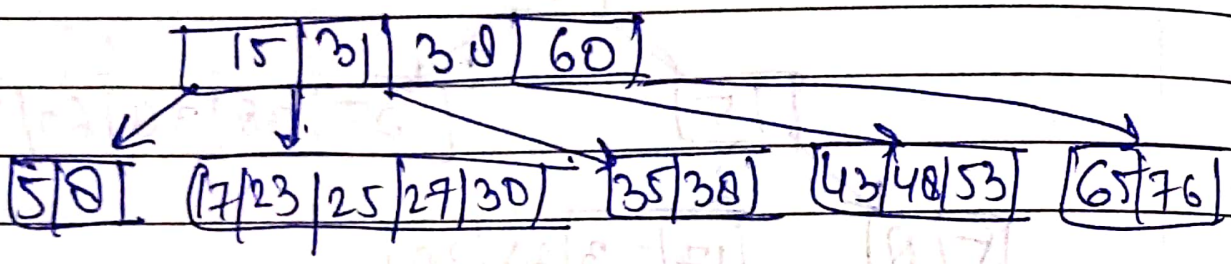








Scanned by CamScanner

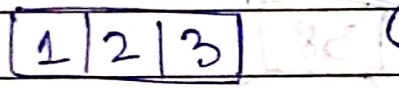


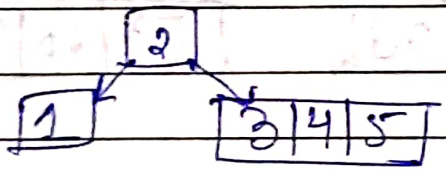
Ques for order $m=4$, create a B-tree for 1 to 10.

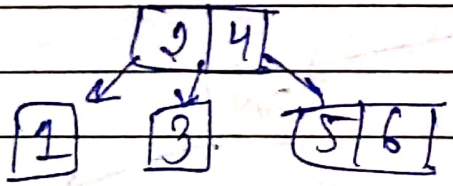
Sol

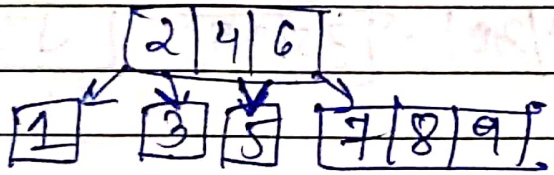
min no. of key = $(m/2 - 1) = 1$.

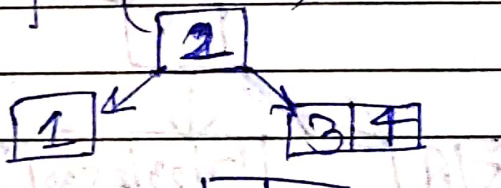
max no. of key = $(m-1) = 3$.

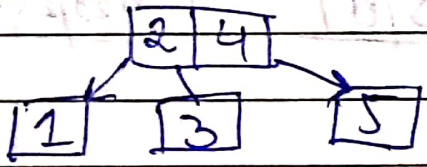


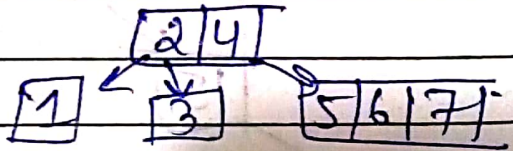


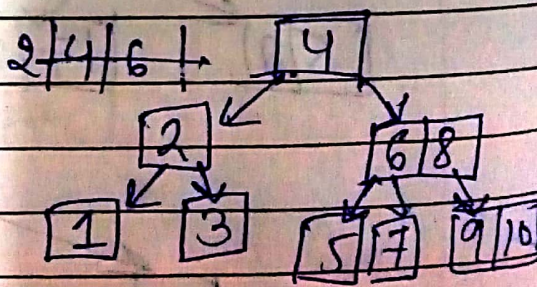












Scanned by CamScanner

Binomial Heap

$B_{k-1} + B_{k-1} = B_k$

$B_0 + B_0 \rightarrow B_1$

$B_1 + B_1 \rightarrow B_2$

$B_2 + B_2 = B_3$

$B_3 + B_3 \rightarrow B_4$

Q. Create Binomial heap for the following data:-
7, 2, 4, 17, 1, 11, 6, 8, 15, 10, 20.

No. of nodes $= 2^k$, where k = degree.

16/11/17

Recurrence Relation

Master Method.

is a procedure to solve the recurrence equation if it is exactly in the form of:-

$$T(n) = aT(n/b) + f(n)$$

where: $a \geq 1$, $b > 1$, $f(n) = (n^k \log^p n)$

Master method proposed the following 3 cases to solve the above recurrence relation:-

Case I: If $\log_b^a > k$, then $\theta(n^{\log_b^a})$

Case II: If $\log_b^a = k$, then $\theta(n^{\log_b} \log n)$

Case III: If $\log_b^a < k$, then if $P \geq 0$, then $\theta(n^k \log^p n)$
if $P < 0$ then $O(n^k)$.

Scanned by CamScanner

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

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$\therefore a = 2;\ b = 2;\ f(n) = n^1;\ k = 1$

$\log_2^2 = 1 \Rightarrow \log_b^a = k.$

Use Case II:

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

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

$a = 4;\ b = 2;\ k = 2;\ \log_b^a = \log_2^4 = 2\log_2^2$

$\log_b^a = k.$

Case I:

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

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

$a = 2;\ b = 2;\ k = 3;\ p = 0$

$\log_b^a = 1 < 3.$

Case III $\Rightarrow T(n) = \Theta(n^3 \log_3^0).$

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

$a = 4,\ b = 2;\ k = 1;\ d > 1.$

Case I: $\Theta(n)$

Scanned by CamScanner

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

$a = 4, \quad b = 2; \quad p = 1, \quad k = 1.$

Case II - $T(n) = \theta(n^{\log_2 4})$.

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

$a = 4, \quad b = 2; \quad k = 3 \; ; \log_b^a < k \; ; p > 0$

Case III $= T(n) = \theta(n^3)$.

$T(n) = 9T(n/3) + n^2$

$\log_b^a = \log_3^9 = 2\log_3^3 = 2 = k = 2.$

Case II.

$T(n) = \theta(n^2)$

$T(n) = 9T(n/3) + n.$

Case I.

$T(n) = T(2n/3) + 1 \Rightarrow 1 = n^0.$

$k = 0, \quad a = 1 \; ; \; b = 2/3. \Rightarrow \log_{3/2}^1 = 0.$

Case I

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

$a = 3; \quad b = 4 \; ; \; \log_4^3 = 0. \; ; \; k = 1$

Case III

$T(n) = 2T(n/2) + n\log n.$

Case II

Scanned by CamScanner

if $p > -1$, then $T(n) = \Theta(n^k \log^{p+1} n)$

if $p = -1$, then $T(n) = \Theta(n^k \log \log n)$

if $p < -1$, then $T(n) = \Theta(n^k)$.

17/09/19

Case III. If $f(n) = \Theta(n^{\log_b a + \epsilon})$.

then $T(n) = f(n)$.

with following regularity condition

$af(n/b) \leq c \cdot f(n)$

where c = constant.

Case II. If $f(n) = \Theta(n^{\log_b a} \log^p n)$.

then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$.

Case I. If $f(n) = \Theta(n^{\log_b a - \epsilon})$

then, $T(n) = \Theta(n^{\log_b a})$

Ques. $T(n) = T\left(\frac{n}{2}\right) + n(2 - \cos n)$

$a = 1$; $b = 2$. $k = 1$.

∴ Can't be solved.

Ques $T(n) = 64\left(\frac{n}{a}\right) - n^2 \log n$

∴ -ve $f(n)$ ∴ Can't be solved

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

$a = 3$; $b = 2$; $k = 2$

Scanned by CamScanner

~~Q(i)~~ $\log_b^a = 1.5 \Rightarrow K = 2 \Rightarrow \log_b^a < K$.

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

Q(ii) $T(n) = 4T(n/2) + n^2$

Case 2 ; $2\log_2^2 = 2$ //.

Q(iii) $T(n) = T(n/2) + 2^n$

~~Can't be solved~~ //

Scanned by CamScanner