Sandip Nepali
191724

1) what is the difference between data type and data structure.

Data Type

- Data Type is the kind of variable which is being used to throughout the Program.
- Implementation through data types is a form of abstraction implementation
- It can hold value and not data, so it is data less.
- values can directly be assigned to the datatype variables.
- No problem of time Complexity

eg. int, float, double etc.

Data structure

- Data structure is the collection of different kinds of data The entire data can be represented ~~of object~~ using an object an can be used throughout the entire program.
- Implementation through Data structure is called Concrete implementation.
- It can hold different types of data within on single object.
- The data is assigned to the datastructure object using some set of algorithms and operation like push, pop etc
- problem of time Complexity.

eg. stacks, queue, tree etc.

Sandip Nepali
191724

2) What is abstract data type? What is the important of ADT in software development?

⇒ ADT are like user defined data types which define operation on values using function without specifying what is there inside the function and how the operation are performed.

eg. Stack ADT

Operations:

Initialize() – Initialize it to be empty.
Push() – Insert Element into the stack.
Pop() – delete element into the stack
isEmpty – check if stack is empty
isFull – check if stack is Full.

Important of ADT:-

- Let say if someone wants to use the stack in the program, then he can ~~in~~ simply use push and pop operation without knowing its implementation.
- ADT provides abstraction.
- Also, if in future, the implementation of stack changed array from linked list client program will work same way without affected.

Sandip Nepali
191724

3.) Write an algorithm to push and pop data on to stack.

Push ( )

1. If Tos = Max-1 Then display "stack is full and stop."
2. Else
   - 2.1 Read data
   - 2.2 Tos ← Tos+1
   - 2.3 stack [tos] ← data.

Pop ( )

1. If Tos = -1 Then display "stack is empty & stop".
2. data ← stack [tos]
3. tos ← tos-1
4. Return data

4.) Convert the following infix expression to postfix using stack.

A + (B * (C − (D / E ^ F) * G)) * H

Scanned character	content of stack	post-fix expression
A		A
+	+	A
(	+(	A
B	+(	AB
*	+(*	AB
C	+(*	ABC
−	+(−	ABC*
(	+(−(	ABC*
D	+(−(	ABC*D
/	+(−(/	ABC*D
E	+(−(/	ABC*DE
^	+(−(/^	ABC*DE
F	+(−(/^	ABC*DEF
)	+(−(/^	ABC*DEF^/
*	+(−*	ABC*DEF^/
G	+(−*	ABC*DEF^/G
)	+	ABC*DEF^/G*−
*	+*	ABC*DEF^/G*−
H	+*	ABC*DEF^/G*−H
empty	empty	ABC*DEF^/G*−H*−

∴ ABC*DEF^/G*−H*−

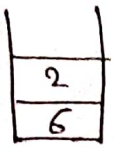
5) Evaluate the following postfix expression stack

623+- 382/+*2$3+

1.  Tos = -1

2. push 6

3. Push 2



4.) Push 3

5) ~~Push~~

$OP_1 = 3$

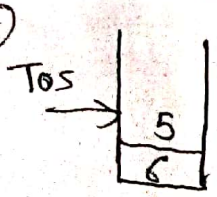
$OP_2 = 2$

result = $OP_2 + OP_1$

= 3+2

= 5

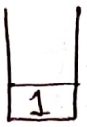
6)



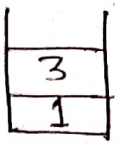
7) $OP_1 = 5$

$OP_2 = 6$

Result = $OP_2 - OP_1$ = 6-5 = 1

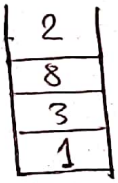


8) Push 3



9) push 8

10) push 2



11) $OP_1 = 2$

$OP_2 = 8$

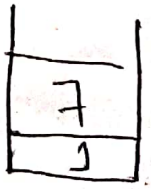
Res = OP_2 / OP_1 = 4

12)

$OP_1 = 4$

$OP_2 = 3$

Res = 3+4 = 7

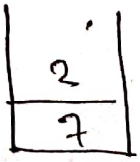


13) $OP_1 = 7$

$OP_2 = 1$

Res = 7*1 = 7

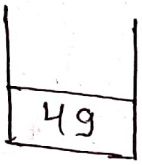
14) push (2)



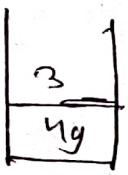
15) $OP_2 = 2$

$OP_1 = 7$

Res = $7^{12} = 49$



16) push 3



17) $OP_2 = 49$

$OP_1 = 3$

Res = 49+3 = 52

$\Rightarrow$  TOP = -1

∴ Result = 52 #

Sandip Nepali

6) Write an Algorithm to Enqueue() and Dequeue() values on to queue.

Solution :-

Enqueue() :- Adding new element in the queue.
Dequeue() :- Removing element from the front.

Algorithm

* Enqueue() :-

1) If Rear = Maxsize - 1. Display "Queue is full and stop"
2) Read data
3) If Rear = front = -1
4) front ← Rear ← Rear + 1
5) q[rear] ← data
6) else
7) Rear ← Rear + 1
8) q[rear] ← data

* Dequeue()

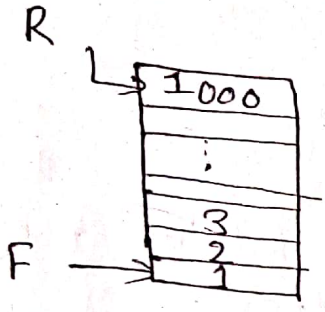
1) If front = Rear = -1 Display "Queue is empty & stop"
2) If front = Rear
3) item ← q[front]
4) Front ← Rear ← -1
5) return item
6) else
7) item ← q[Front]
8) front ← front + 1
9) return item

7) What are the limitation of linear queue how this can be corrected using circular queue.

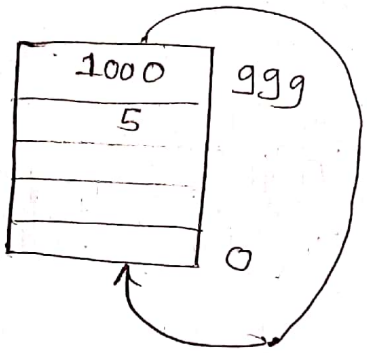
Soln:-

~~Problem~~ The limitation of linear queue :-

int a[1000]



Dequeue 999 times



Enqueue

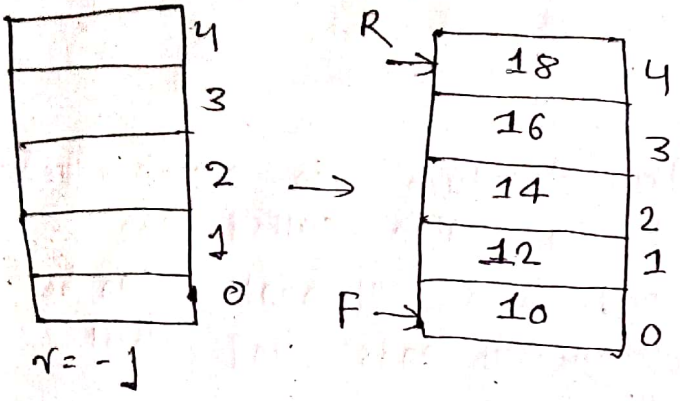
queue is Full

Since First in First out & Last in last out The Problem with linear queue is If we Dequeue the element of (n-1) times where n is the Max size of of queue then while enqueue the new elements it shows the message queue is full i.e we can't put new elements even there is space.

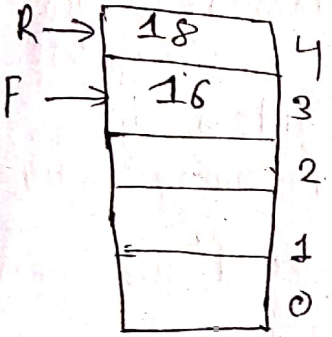
Circular queue ~~overcome this~~ Solve this limitation

```
# define Max-size 5
  int main ( ) {
  int eq [Max-size];
  return 0;
  }
```

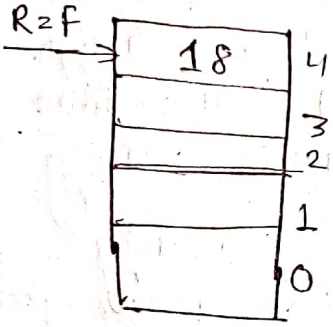
Sandip ~~Hetpo~~ Nepal



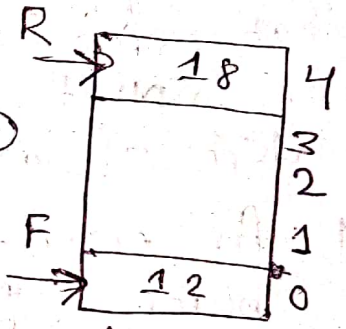
(*) Dequeue - 3

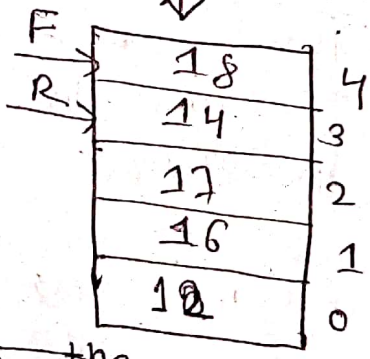


(*) Further 1 dequeues



Enqueue (12)





In circular, if we Dequeue the elements of (n-1) times. ~~if there n is the Max size of~~ Then we can enqueue the element if there is space.

8) What is Priority queue? Explain its Types.

Ans:-

A priority queue is a data structure in which each element has been assigned a value called the priority of the element and an element can be inserted and deleted not only at ends but at any position of the queue.

A Priority queue is collection of element such that each element has been assigned an explicit or implicit priority and such that order in which element are deleted using following rule:

1) An element of higher priority ~~proceed~~ processed before any element of lower priority.

2) Two element with same priority are processed to ~~order~~ the order in which they were inserted in queue

There are Two types of Priority Queue :-

a) Ascending Peiority queue

b) Descending Paiority queue.

Ascending queue:

In ascending Priority queue element can be inserted in any order, but while deleting element from the queue, always element associated with lowest Priority ~~queue~~ number will be deleted.

Sandip Nepal

Descending queue:-
In descending queue element can arrive in any order but while removing element associated with highest priority number will be delcted.

9) What is the major advantage of using dynamic list (linked list) insted of using static list (array).

Solution:-

The advantages of using dynamic list using static list are:-

1) Dynamic data structure:
- A linked list is a dynamic arrangement so it can grow and shrink at runtime by allocating and deallocating memory. so there is no need to initialize size of linked list.

2) No Memory Wastage:- • In linked list, efficient memory utilization can be achieved since the size of linked list increase or decrease at run time. so there is no memory wastage.

3) Implementation:- Linear data structure like stack are queues are often easily implemented using linked list

4) Insertion and deletion operations: Insertion and deletion operation is quit ~~easy~~ easier in the linked-list. There is no shifts elements after the insertion and deletion of an element only the address of present pointer updated.

10) Write an algorithm to insert a node at nth position in doubly circular linked list.

Solution:-

Insert as nth node

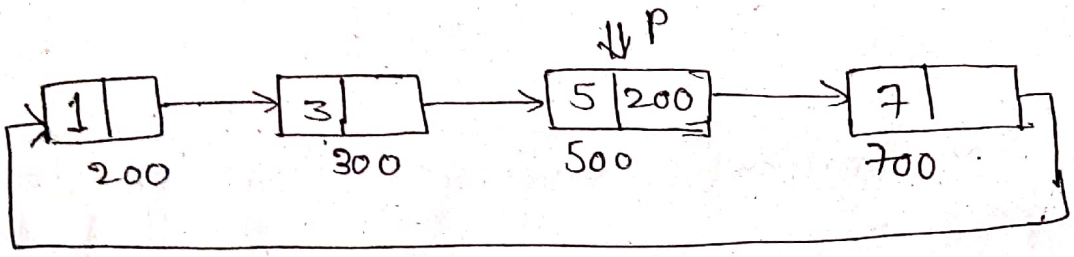
1. temp = (DCLL) * malloc (sizeof (DCLL))
2. temp → info = data
3. P = first
4. while (n > 2)
5. P = P → next
6. n--
7. End while
8. temp → next = P → next
9. temp → prev = P
10. P → next → prev = temp
11. P → next = temp.

Defining node

```
struct Doubly-Circular-linked-list {
    int info;
};
typedef struct Doubly-circular-linked-list DCLL;
DCLL *P, *q, *temp, *first = NULL.
```

11.) Write an algorithm to delete the last node of a ~~linear~~ singly circular linked list.

Solution:-



1. p= first
2. while (p→next != first)
3. p= p→next
4. End while
5. temp = p→next
6. p→next = first
7. free (temp).

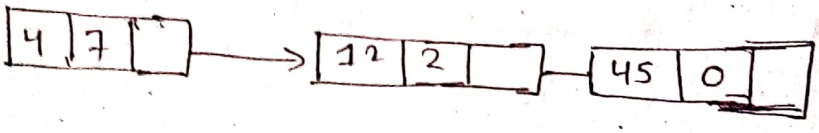
12) Write an algorithm to add two polynomial using linked-list?

Solution

struct node{
float coefficient;

Polynomial is a mathmatical expression that consists of variable and coefficients.

eg. $4x^7 + 12y^2 + 45$



Algorithm

1. loop around all values of linked list and follow step 2 and 3
2. If the value of the node's exponents is greater Copy this node to the result node and head towards the next node.
3. If the values of both node's exponent is same add the coefficient and then copy the added value with node to the results.
4. Print the resultant node.

18). Write a code and algorithm for Tower of Hanoi (TOH).

Algorithm

1) Transfer (n-1) disk from peg 'A' to peg 'B' using peg 'C' as auxillary.

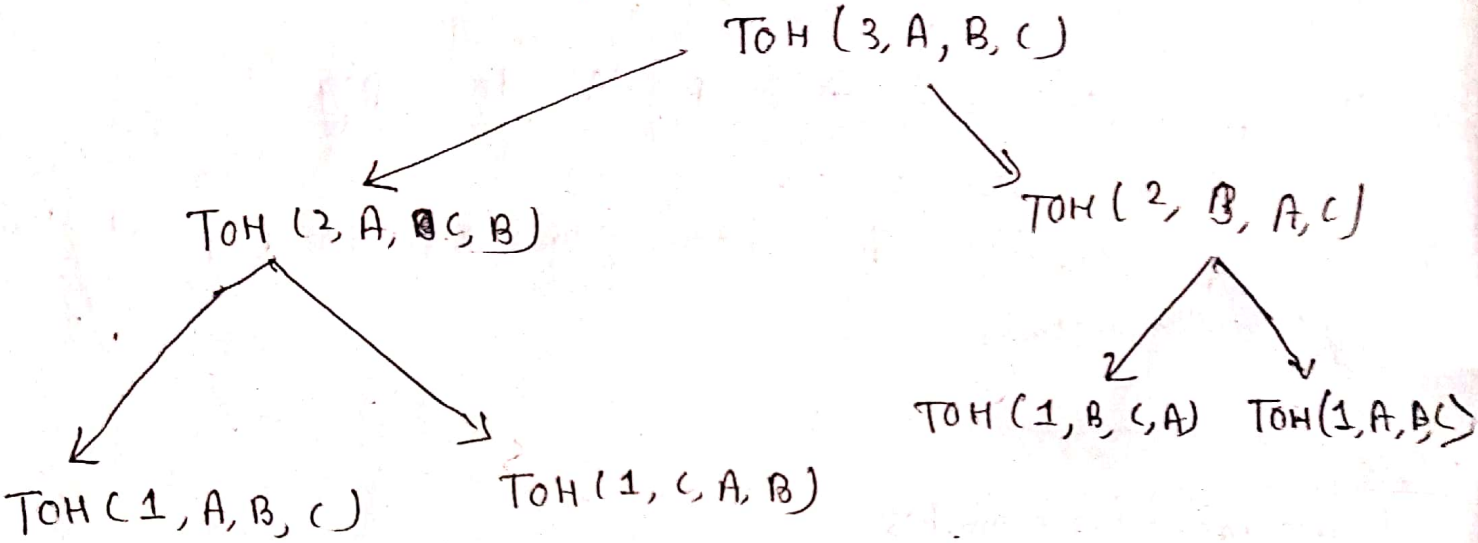
2) Transfer nth disk from peg 'A' to peg 'C'.

3) Transfer (n-1) disk from peg 'B' to peg 'C' using 'A' as auxillavy.

Code

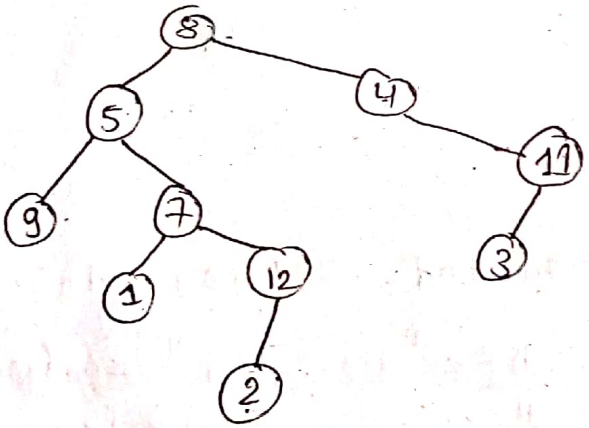
```
# include < iostream.h>
void towerofHanoi (int n, char from_rod, char to_rod, char
                                    aux_rod)
{
  if (n==0)
  {
   returns;
  }
  towerofHanoi (n-1, from_rod, aux_rod, to_rod);
  Cout<<" Move disk " <<n<< "from rod " << from_rod <<
                "to rod " << to_rod <<endl;
  towerOfHanoi (n-1, aux_rod, to_rod, from_rod);
}
int main () {
  int n=4;
  towerofHanoi (n, 'A', 'C', 'B' );
  return 0;
}
```

14.) Draw recursion tree for TOH when number of disk $n=3$.



15.) Perform In-order, pre-order and post order tree traversal ?

Ans :-



~~In-order :- 2 12 1 7 2, 1, 7 12, 9, 5~~

In-order :- 2, 12, 1, 7, 9, 5 8, 3, 11, 4

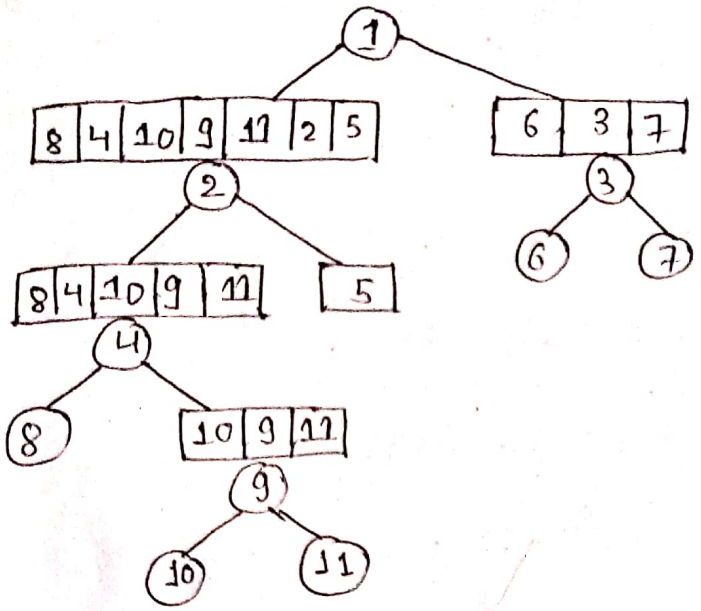
Pre-order :- 8 5 9 7 1 12 2 4 11 3

Post-order :- 2 12 1 7 9 5 3 11 4 8

16.) Construct the tree from the given values

Pre-order Traversal :- 1 2 4 8 9 10 11 5 3 6 7

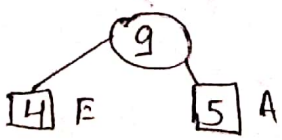
In-order Traversal :- 8 4 10 9 11 2 5 1 6 3 7



17) Construct Huffman tree for following values.

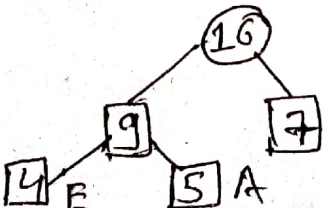
value	E	A	C	F	D	D
	4	5	7	12	15	25

Taking two minimum weights



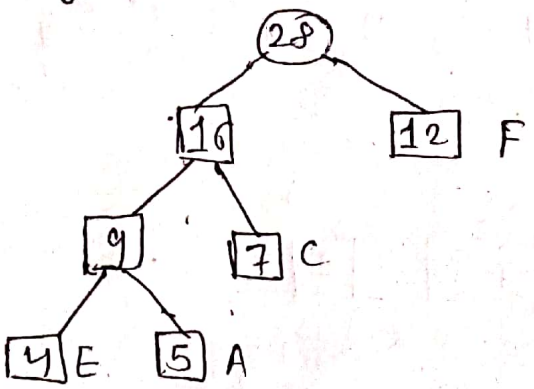
new weight list 9 7 12 15 25

Taking two minimum weight



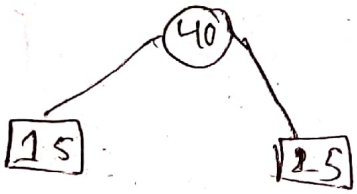
New weight list :- 16 12 15 25

Taking two minimum weight

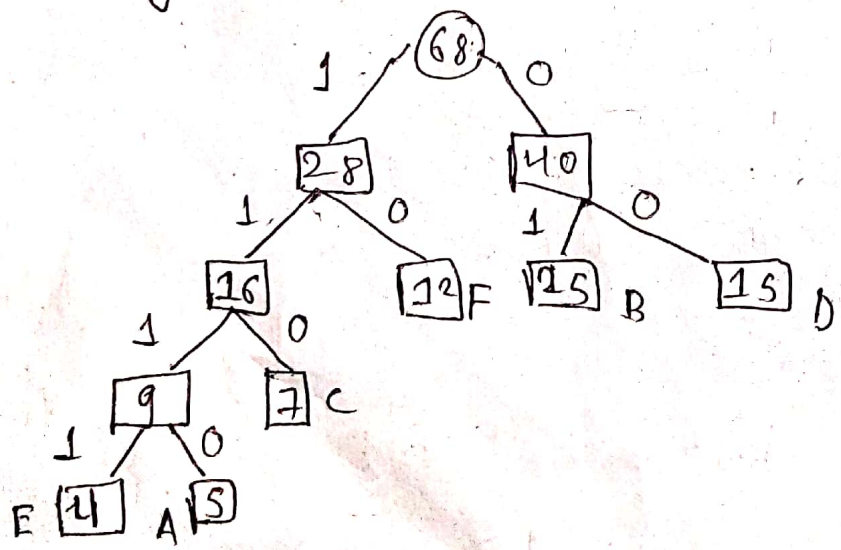


New weight list :- 28 15 25

Taking two minimum weight



New weight list :- 28 40



Huff-Man Code

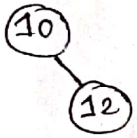
F : 1 1 1 1
A : 1 1 1 0
C : 1 1 0
F : 1 0
D : 0 0
B : 0 1

(18) Construct AVL Tree for following values

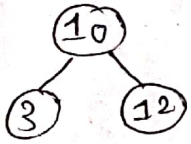
10, 12, 3, 42, 16, 18, 27, 30 6, 89, 22, 7 33

1.)

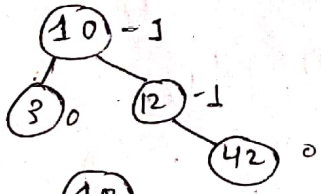
2.)



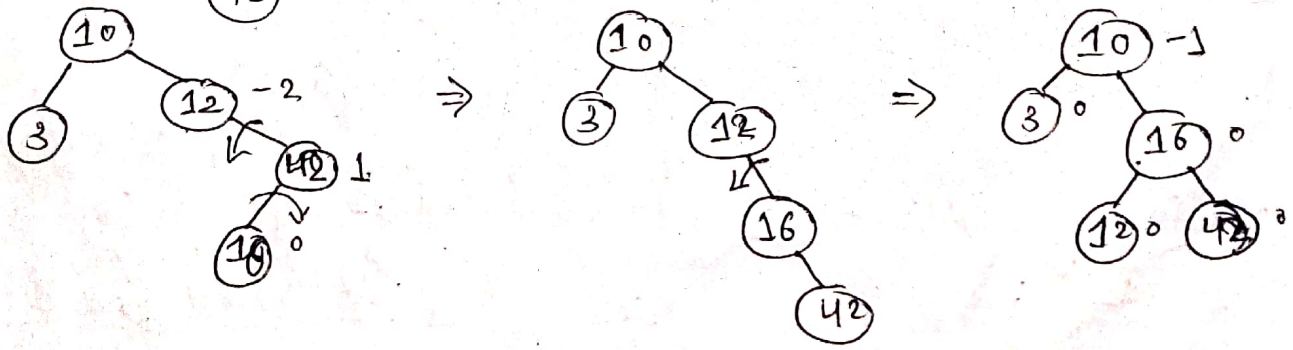
3.



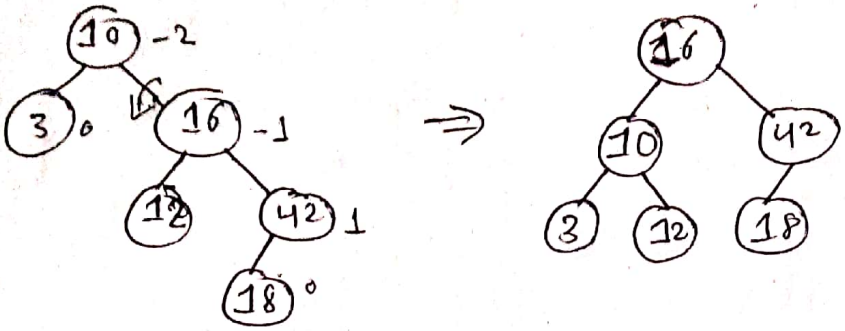
4.



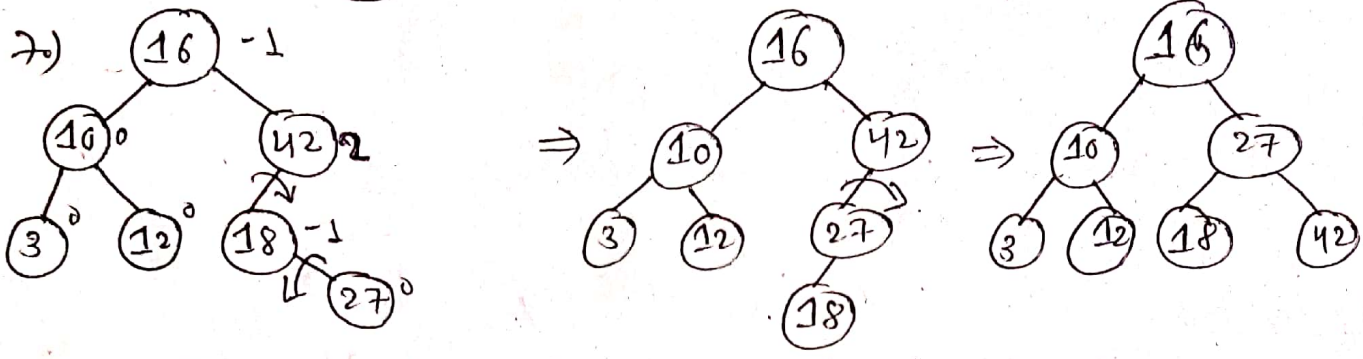
5.



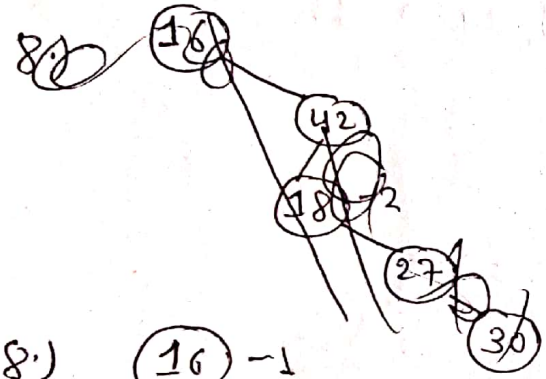
6.)



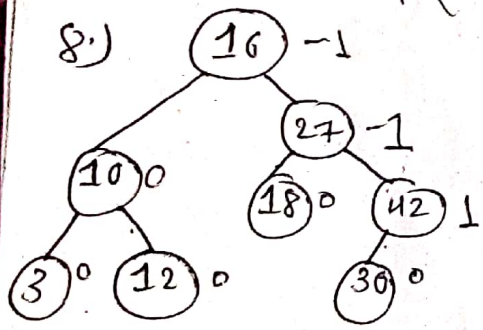
7.)



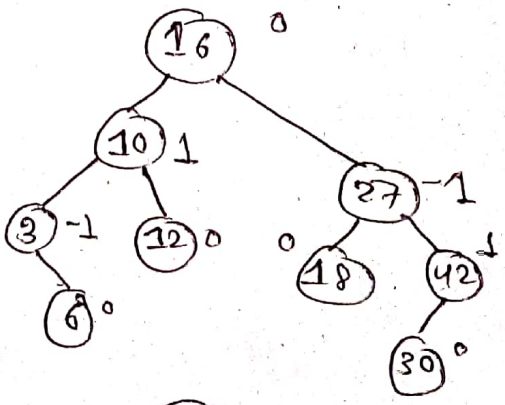
8.)



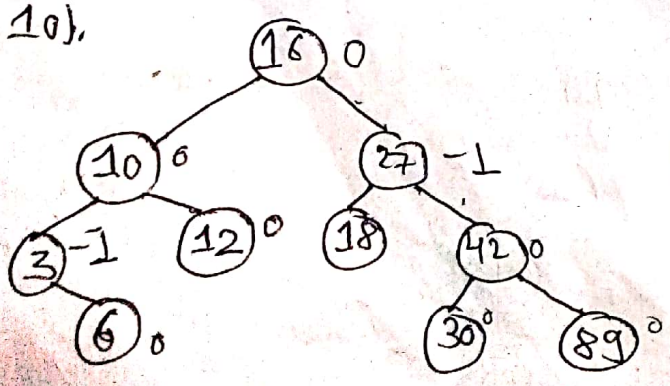
8.)



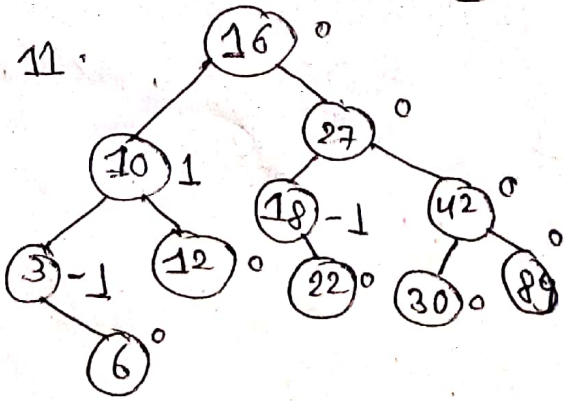
9.)

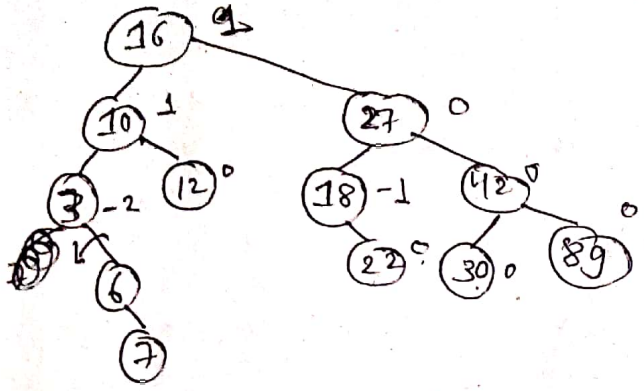


10).

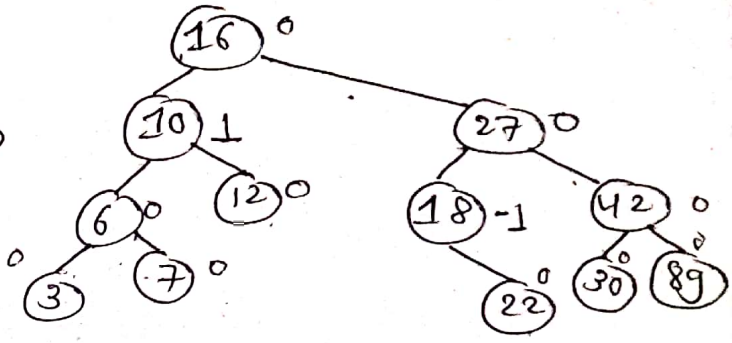


11.

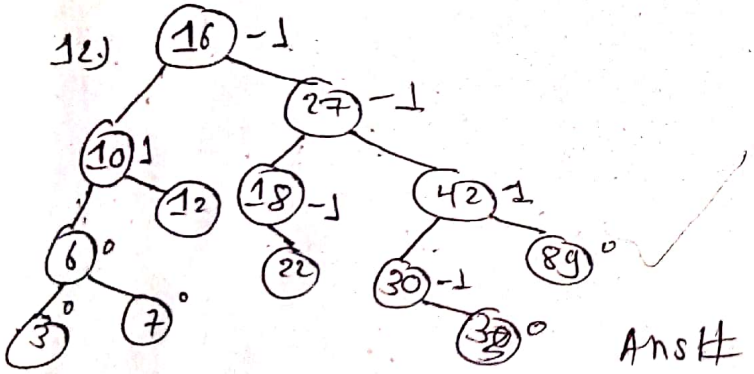



16
10
27
3
12
18
42
6
22
30
89
7

⇒

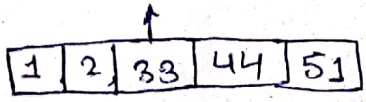

16
10
27
6
12
18
42
3
7
22
30
89

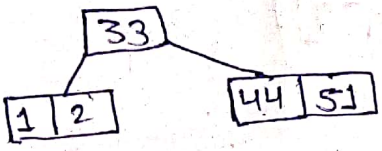
12)


16
27
10
12
18
42
6
22
30
89
3
7

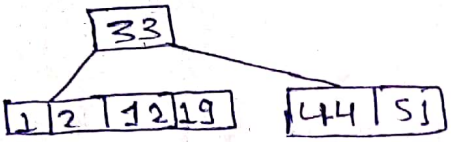
Ans H

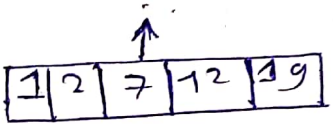
19. Construct B-tree of order -5



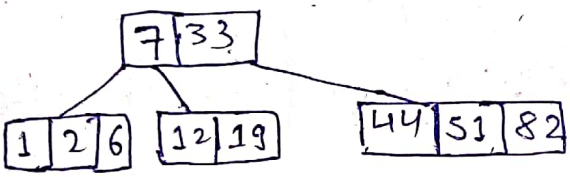


Insert 12, 19

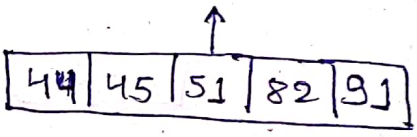


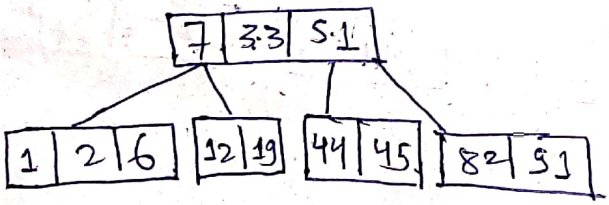


Insert 7, 6, 82

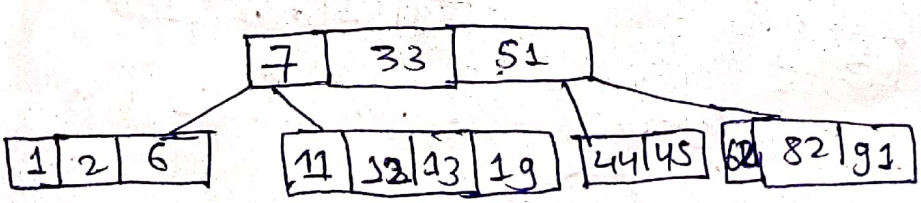


Insert 91, 45



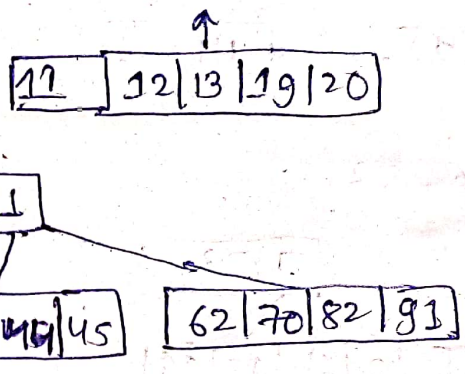


Insert 62, 11, 13

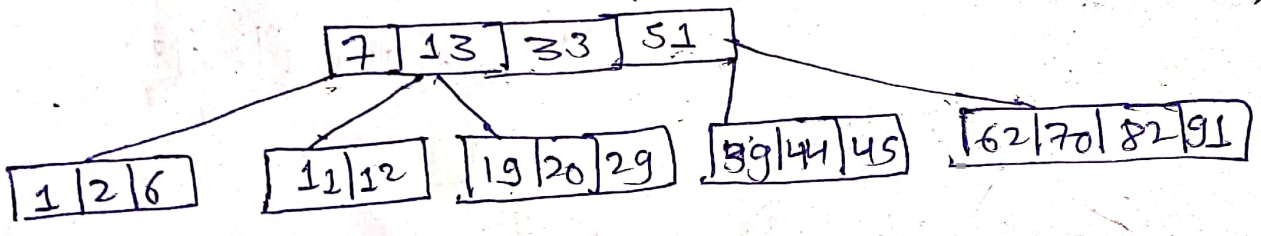


Insert 2 9, 3 9

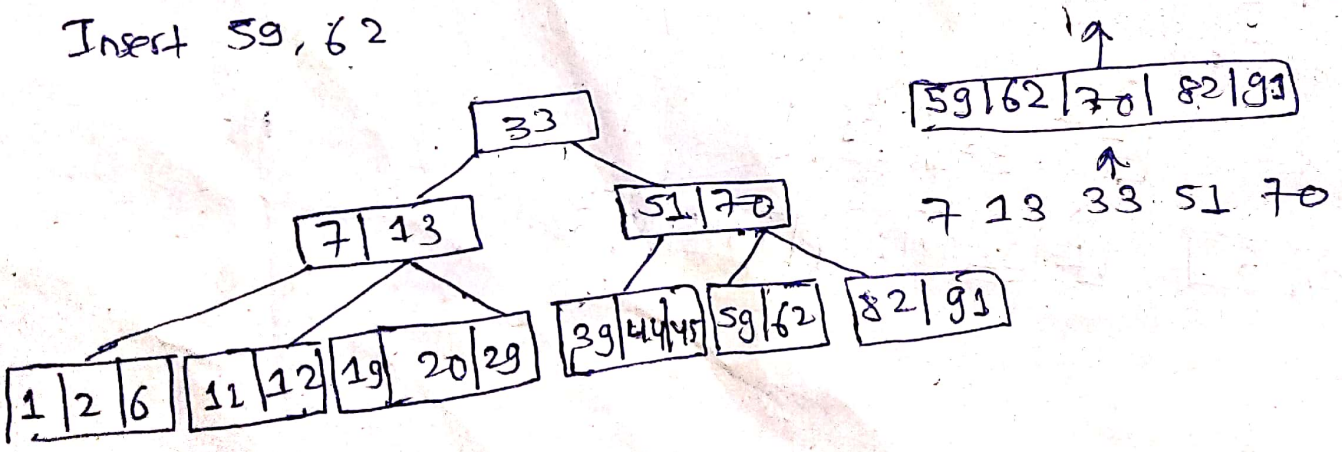
Insert 70, 20



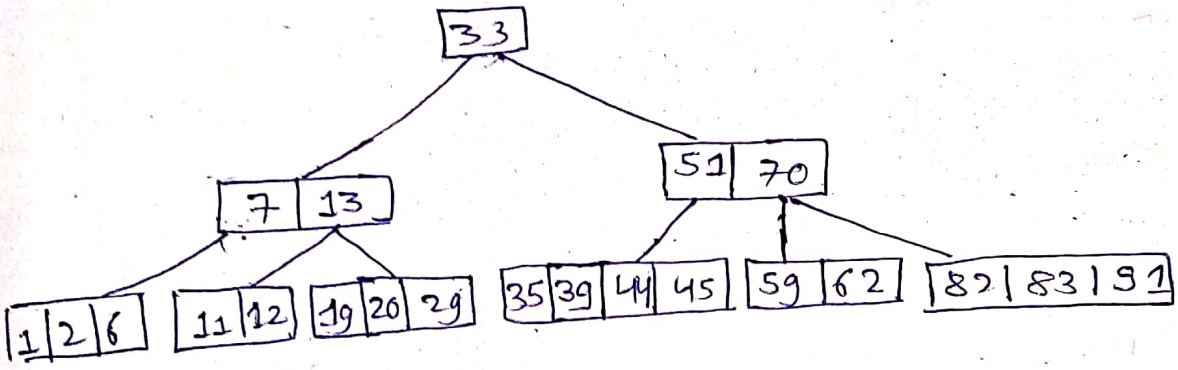
Insert 29, 39



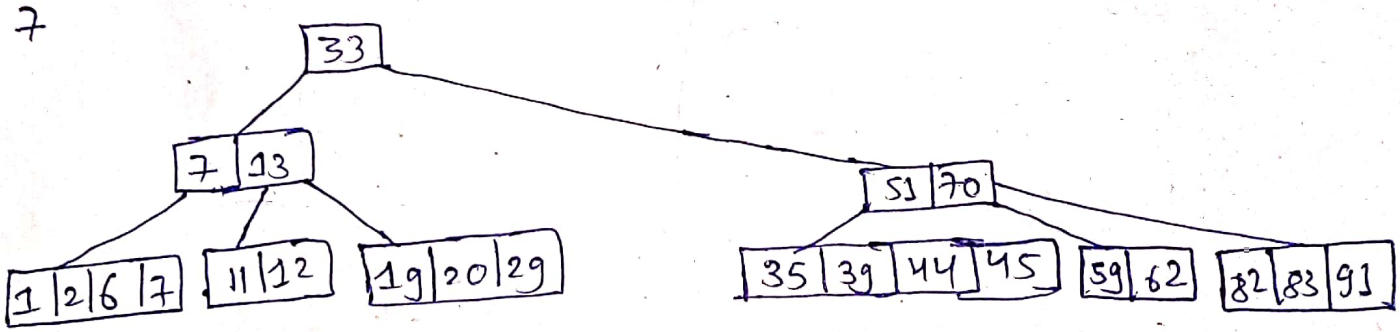
Insert 59, 62



Insert 83 35



Insert 7



2. Use Binary search technique to find a key = 40 in in this following sequence

1, 8, 12, 13, 24, 30, 34, 40, 50, 60, 80, 90, 92.

soln:-

key = 40

1) LB = 0

UB = 12

Mid = $\frac{12}{2} = 6$

A[6] = 34

key > A[Mid]

2) LB = Mid + 1 = 7

UB = 12

Mid = $\frac{7+12}{2} = 9$

A[9] = 60

A[9] > key

3) ~~LB = Mid - 1 = 8~~

LB = 7

UB = Mid - 1 = 8

Mid = $\frac{7+8}{2} = 7$

A[7] = 40. key is matched #

21 Input the following data into hash table of size 10.

Input sequence: 1, 27, 6, 87, 47, 7, 8, 17, 37, 67

Hash Function is key Mod 10

Solution:-

1) Insert 1

$H(k, i) = (H(k) + i) \bmod 10$

$H(1, 0) = (1 \bmod 10 + 0) \bmod 10 = 1$

2) Insert 27

$H(27, 0) = (27 \bmod 10 + 0) \bmod 10 = 7$

3) Insert 6

$H(6, 0) = (6 \bmod 10 + 0) \bmod 10 = 6$

4) Insert 87 $= (87 \bmod 10 + 0) \bmod 10 = 7$ (collision)

~~5) Insert 47 = (47 Mod 10 + 0) Mod 10 =~~

$= (87 \bmod 10 + 1) \bmod 10 = 8$

5) Insert 47 $= (47 \bmod 10 + 0) \bmod 10 = 7$ (collision)

$H(47, 1) = (47 \bmod 10 + 1) \bmod 10 = 8$ (collision)

$H(47, 2) = (47 \bmod 10 + 2) \bmod 10 = 9$

6) Insert 7

$H(7, 0) = (7 \bmod 10 + 0) \bmod 10 = 7$ (Collision)

$H(7, 1) = (7 \bmod 10 + 1) \bmod 10 = 8$ (collision)

$H(7, 2) = (7 \bmod 10 + 2) \bmod 10 = 9$ (collision)

$H(7, 3) = (7 \bmod 10 + 3) \bmod 10 = 0$

Insert 8:

$H(8,0) = (8 \bmod 10 + 0) \bmod 10 = 8$ (collision)
$H(8,1) = (8 \bmod 10 + 1) \bmod 10 = 9$ (collision)
$H(8,2) = (8 \bmod 10 + 2) \bmod 10 = 0$ (collision)
$H(8,3) = (8 \bmod 10 + 3) \bmod 10 = 1$ (collision)
$H(8,4) = (8 \bmod 10 + 4) \bmod 10 = 2$

Insert 17

$H(17,0) = (17 \bmod 10 + 0) \bmod 10 = 7$ (collision)
$H(17,1) = (17 \bmod 10 + 1) \bmod 10 = 8$ (collision)
$H(17,2) = (17 \bmod 10 + 2) \bmod 10 = 9$ (collision)
$H(17,3) = (17 \bmod 10 + 3) \bmod 10 = 0$ (collision)
$H(17,4) = (17 \bmod 10 + 4) \bmod 10 = 1$ (collision)
$H(17,5) = (17 \bmod 10 + 5) \bmod 10 = 2$ (collision)
$H(17,6) = (17 \bmod 10 + 6) \bmod 10 = 3$

Insert 37

$H(37,0) = (37 \bmod 10 + 0) \bmod 10 = 7$ (collision)
$H(37,7) = (37 \bmod 10 + 7) \bmod 10 = 4$

Insert 67

$H(67,8) = (67 \bmod 10 + 8) \bmod 10 = 5$

0	1	2	3	4	5	6	7	8	9
7	1	8	17	37	67	6	27	87	47

Using Collision Handling H(key) + f(i) where f(i) = i*i

Insert 1.

$H(k,i) = (H(k) + c_1 i + c_2 i^2) \bmod m$

$H(1,0) = (1 \bmod 10$

$H(k,i) = (H(k) + i^2) \bmod m$

$H(1,0) = (1 \bmod 10 + 0*0) \bmod 10 = 1$

Insert 27

$H(27,0) = (27 \bmod 10 + 0*0) \bmod 10 = 7$

Insert 6

$H(6,0) = (6 \bmod 10 + 0*0) \bmod 10 = 6$

Insert 87

$H(87,0) = (87 \bmod 10 + 0*0) \bmod 10 =$ collision

$H(87,1) = (87 \bmod 10 + 1*1) \bmod 10 = 8$

Insert 47

$H(47,0) = (47 \bmod 10 + 0*0) \bmod 10 = 7$ collision

$H(47,2) = (47 \bmod 10 + 2*2) \bmod 10 = 1$ collision

$H(47,3) = (47 \bmod 10 + 3*3) \bmod 10 = 6$ collision

$H(47,4) = (47 \bmod 10 + 4*4) \bmod 10 = 3$

Insert 7

$H(7,0) = (7 \bmod 10 + 0*0) \bmod 10 = 7$ collision

$H(7,4) = (7 \bmod 10 + 4*4) \bmod 10 = 3$

$H(7,5) = (7 \bmod 10 + 5*5) \bmod 10 = 2$

22 sort the following sequence of number using quick sort and Heap sort.

12, 2, 24, 56, 19, 17, 13, 90, 44, 61, 27, 45

quick short

pivot = 12

↓up ↓up

12, 2, 24, 56, 19, 17, 13, 90, 44, 61, 27, 45

↑d ↑d

pivot = 24

↓up ↓up

2, (12) 24 56 19 17, 13, 90, 44, 61 27, 45

↑d ↑d

↓up ↓up

2 (12) 24 13 19 17 56 90 44 61 27 45

↑d ↑d

↓up ↓up

2 (12) 17 13 19 (24) 56 90 44 61 27 45

↑d ↑d

pivot = 17

2 (12) 13 (17) 19 (24) 56 90 44 61 27 45

↓up

pivot = 56

2 (12) 13 (17) 19 (24) 56 90 44 61 27 45

↑d ↑d

↓up ↓up

2 (12) 13 (17) 19 (24) 56 45 44 61 27 90

↑d ↑d

↓up ↓up

2 (12) 13 (17) 19 (24) 56 45 44 27 61 90

↑d ↑d

2 (12) 13 17 19 (24) 27 45 44 (56) 61 90

↓up

pivot = 27

2 12 13 17 19 (24) 27 45 44 (56) 61 90

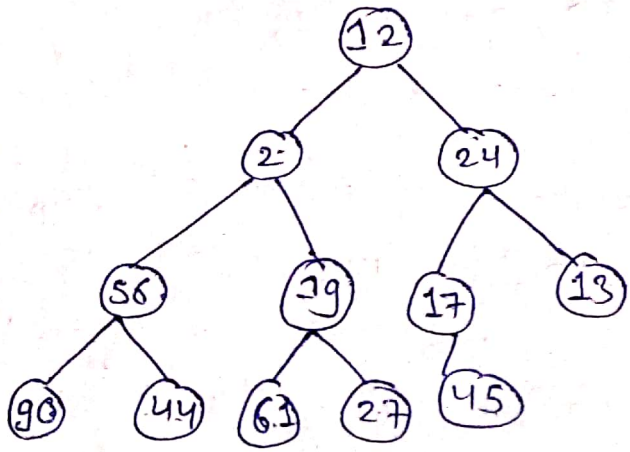
↑d ↑d

2 (12) 13 17 19 (24) 27 44 45 56 61 90

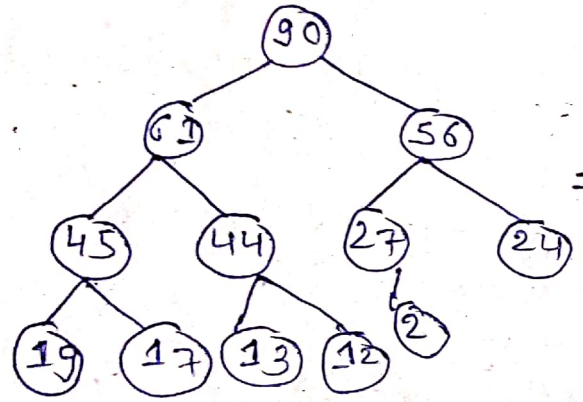
The sorted sequence is

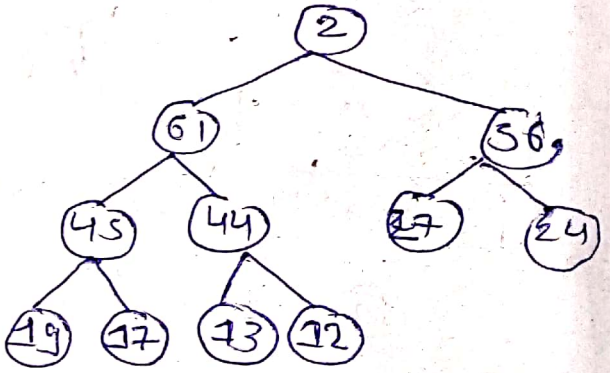
2 12 13 17 19 24 27 44 45 56 61 90

Heap sort



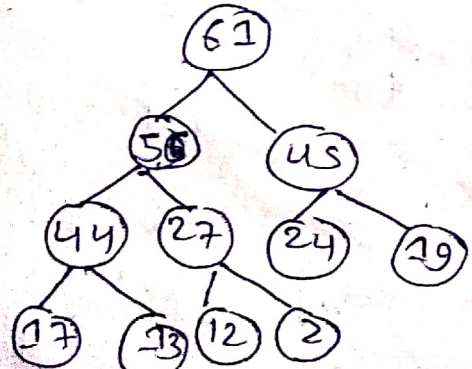
heapify

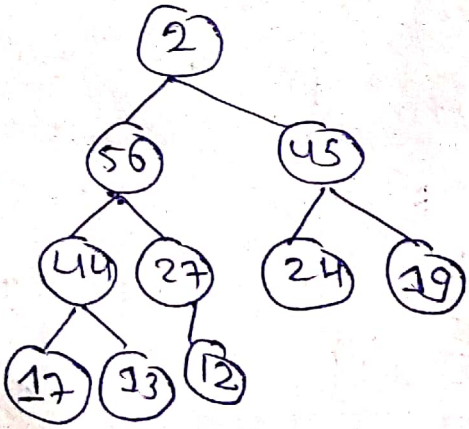




O/p list: 90

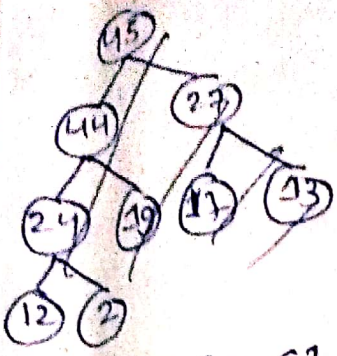
Not max heap



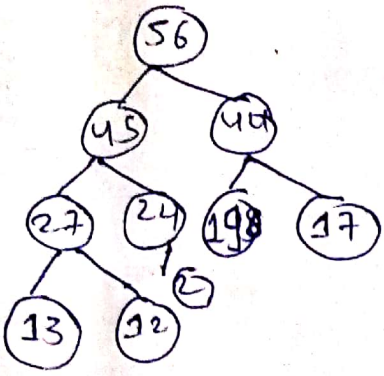


Op list: 90, 61

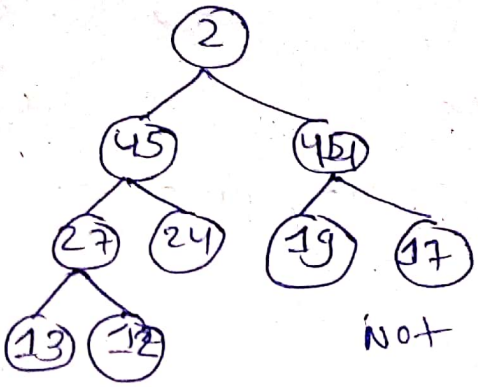
Not max heap



O/p list: ~~90, 61,~~

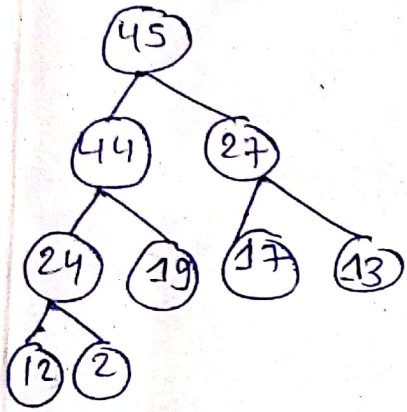


⟹

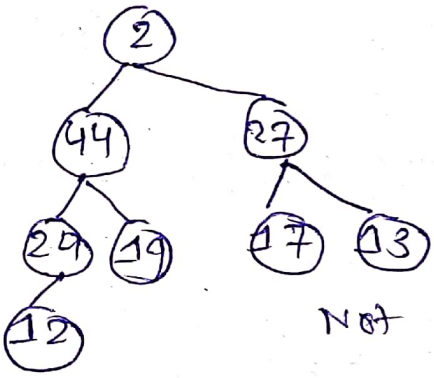


Not max heap

O/p list: 90, 61, 56

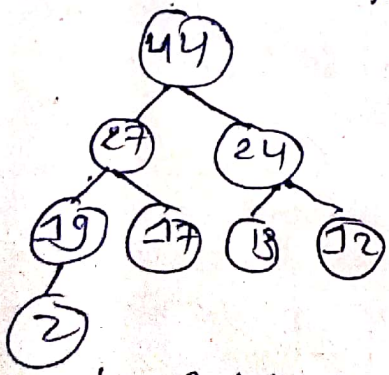


⟹

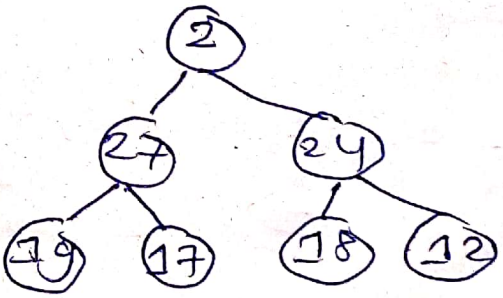


Not max heap

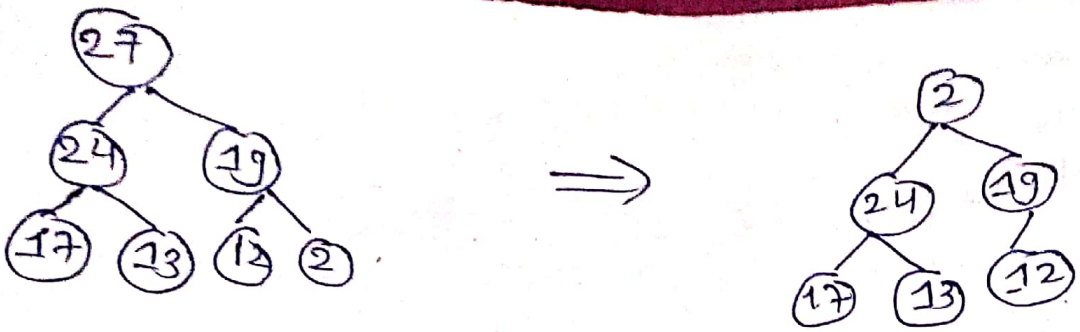
O/p: list : 90, 61, 56, 45



⟹

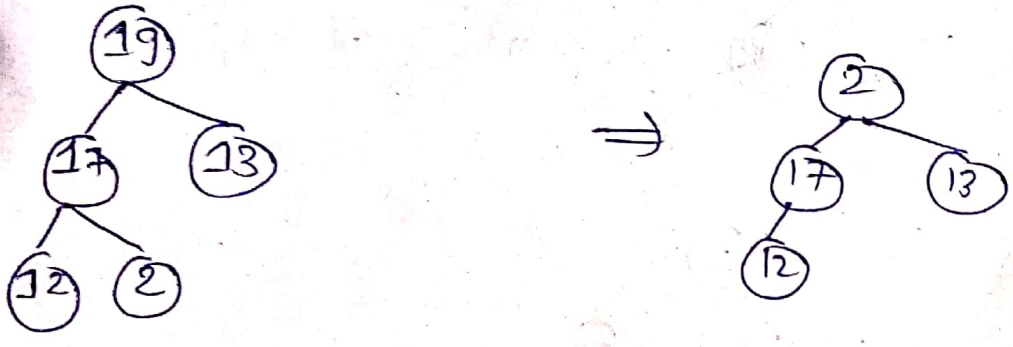


O/p list : 90, 61, 56, 45, 44

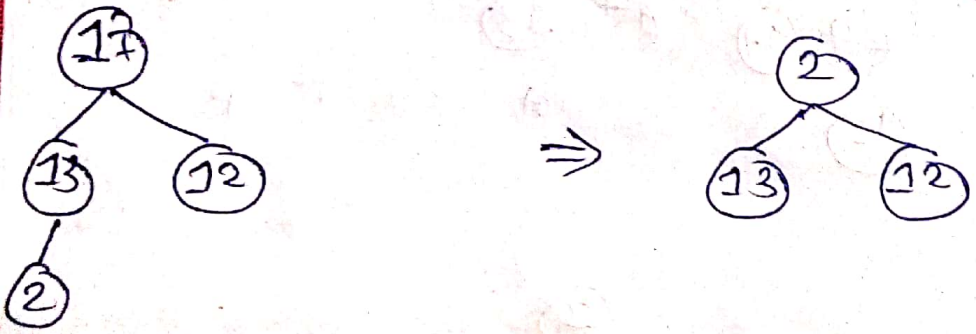


o/p list: 90, 61, 56, 45, 44, 27

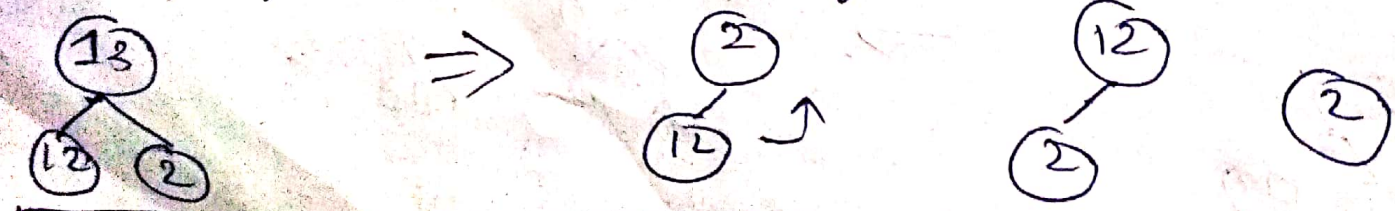
o/p: 90, 61, 56, 45, 44, 27, 24



o/p: list: 90, 61, 56, 45, 44, 27, 24, 19



o/p: 90, 61, 56, 45, 44, 27, 24, 19, 17



o/p: 90, 61, 56, 45, 44, 27, 24, 19, 17, 13, 12, 2

23) What is the sorting explain the difference between following

a) Internal and External sorting

b) Stable and Unstable sorting

Ans) Sorting :- The arrangement of data or element in increasing or in decreasing.

Internal sorting

- All the items to be sorted can be sorted in the primary memory of computer.
- The size of data in array is small.
- Less time is required.

External sorting

- All the data items to be sorted are stored on a relatively slower ~~medev~~ medium than primary such as external storage
- The size of data in array is large.
- slower than internal sorting.

(1) Stable sorting and unstable

Stable sorting maintain the relative order of records with equal keys is stable if not unstable

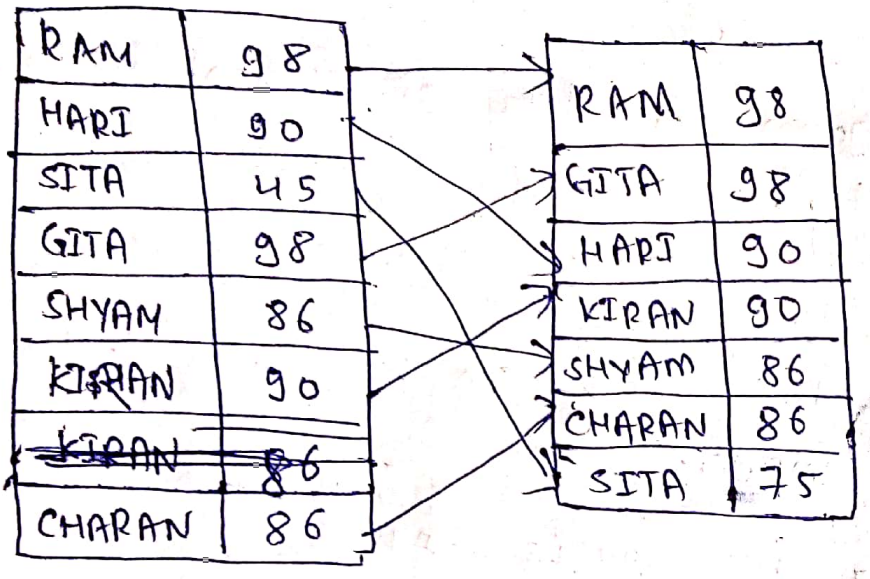
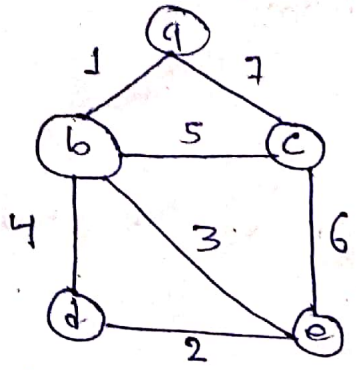


fig : stable sort.

24)

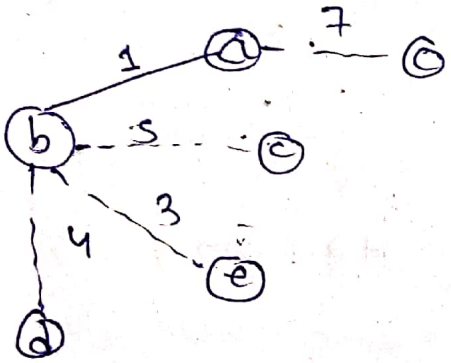
Prism Algorithm



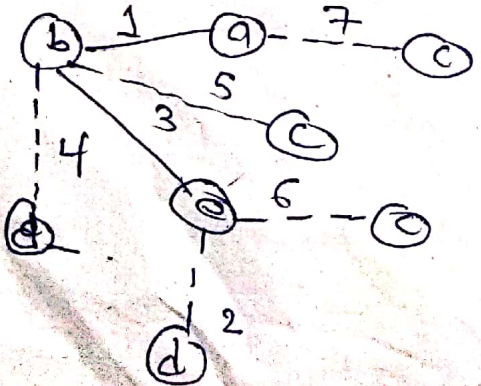
V = {a, b, c, d, e}

We start from b. the edges incident from the b are {b,a}, {b, c}, {b, e} and {b,d} with distance 1, 5, 3, 4. respectively. The minimum distance is 1.

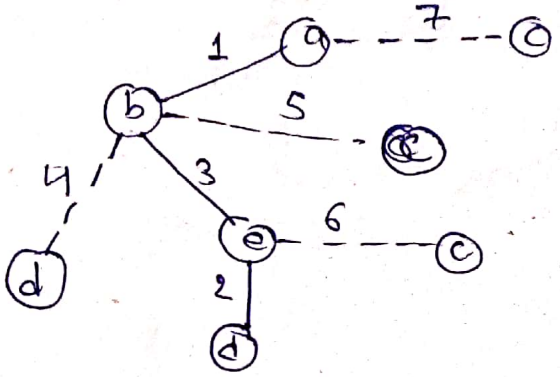
T = {{b, a}}



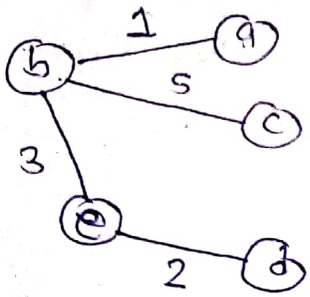
minimum distance among all the edges incident from and b is 3 so we include (e) vertex



minimum distance among all the edges incident from a, b, and e is d. so we include d.



we include vertex. ~~vertex~~ (c)



Cost of MST. = 1 + 3 + 5 + 2 = 11

Kruskal's Algorithm :-

The weight of the edges of the above graph is given in the below

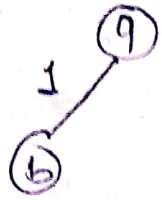
ab	ac	bc	be	bd	de	ce
1	7	5	3	4	2	6

sorte the edges in ascending order

ab	de	be	bd	bc	ce	ac
1	2	3	4	5	6	7

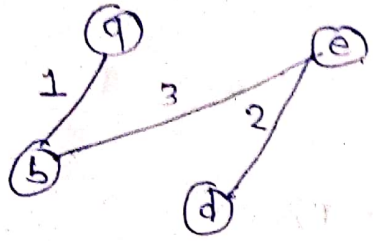
Step-1

First add the edge ab with distance 1.



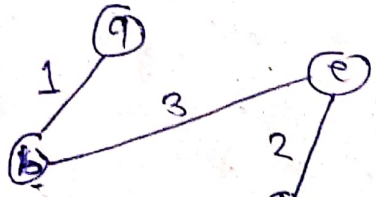
Step-2

Add the de with weight 2, as it not creating the cycle.



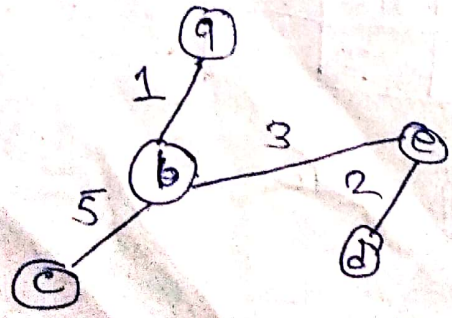
Step-3

Add be with weight 3, as it is not creating the cycle



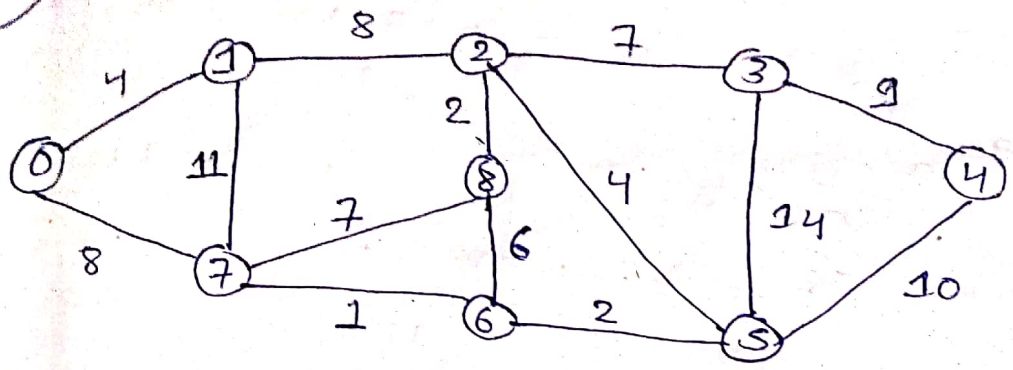
Step-4

Add bc with weight 5, as it is not creating the cycle

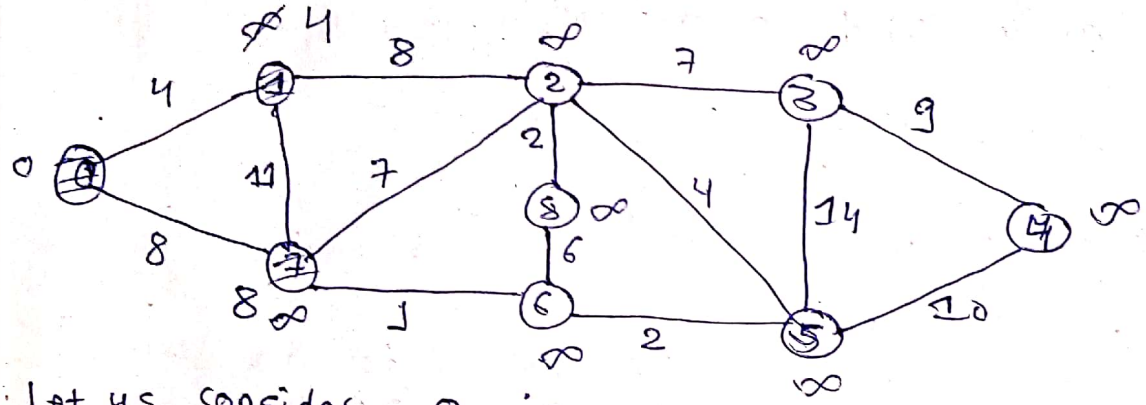


∴ The cost of Mst is :- 1 + 3 + 2 + 5 = 11

25



Solng:-



Let us consider 0 is source vertex. and assume that 0 as a source vertex and distance to all the other vertices is infinity.

Letus assume that 0 represented by 'u' and 1 represented by 'v'. The distance between the vertices can be calculate by using the below formula

$$d(u) + c(u,v) < d(v)$$

$$0 + 4 < \infty$$

$$4 < \infty$$

So we will update d(v)

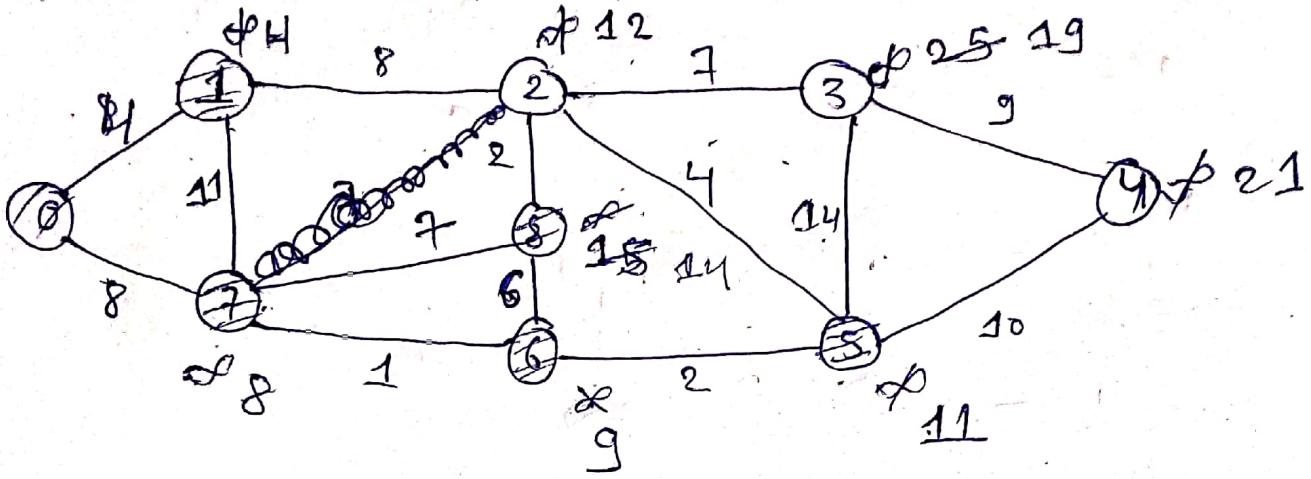
Now Consider vertex 0 as 'u' and 7 as 'v'.

$$0 + 8 < \infty$$

$$8 < \infty$$

So we will update d(v)

V(1) is selected since the distance is shortest.



Consider 1 as 'u' and 2 as 'v'.

$4+8 < \infty$

Consider 1 as 'u' and 7 as 'v'.

$4+11 < \infty$

$15 < \infty$ (Therere is no need to update)

V(7) is selected.

7 represented as 'u' and 6 represented as 'v'

$8+1 < \infty$

$9 < \infty$ so update V(6).

7 represented as 'u' and 8 represented as 'v'

$8+7 < \infty$

$15 < \infty$ updated V(8).

V(6) is selected.

6 represented as 'u' and 5 represented as 'v'

$9+2 < \infty$

$11 < \infty$ V(5) is updated.

6 represented as 'u' and 8 represented as 'v'

$9+6 < \infty$

$15 < \infty$ There is no need to changed.

Select v(5) due to shortest distance.

5 represented by 'u' and 4 represented by 'u'

11 + 10 < ∞

21 < ∞ so v(4) is changed.

5 represented by 'u' and 3 is represented by 'u'

11 + 14 < ∞

25 < ∞ v(3) is changed.

5 represented by 'u' and 2 represented by 'v'

11 + 4 < ∞

15 < ∞ There is no need to changed

v(2) is selected since 12 is shortest distance.

2 represented by 'u' and 8 represented by 'v'

12 + 2 < ∞

14 < ∞ v(8) is changed.

2 represented by 'u' and 3 represented by 'v'

12 + 7 < ∞

19 < ∞ v(3) is changed.

v(8) is selected since 14 is shortest distance

v(3) is selected since 19 is shortest distance

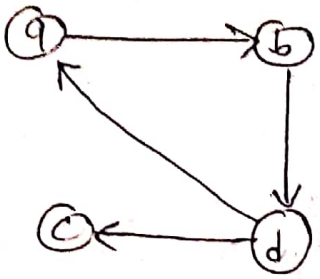
3 represented by 'u' and 3 represented by 'v'

19 + 9 < ∞

28 < ∞ There is no need to changed. uptaded.

∴ The shortest path between v(0) and v(4) is 21.

26) Find the transitive closure of following graph.



Solution

Adjacency Matrix

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

$R(0) =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

$R^{(1)} =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

$R^{(1)} =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

$R^{(2)} =$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$R^{(2)} =$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$R^{(3)} =$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$$R^{(3)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

$$R^{(4)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Transitive closure

$$T = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Ans

27) What is Big O notation and how it is useful to measure an ~~effica~~ efficiency of algorithm.

Solution

↳ Big O notation is the way to measure algorithm's efficiency.

↳ There are two parts to measuring efficiency
① Time Complexity
② Sape Complexity

Time Complexity :- Time complexity is a measure of how long the function takes to run in terms of its computational steps

Space Complexity :- It has to do with the amount of memory used by the function.

For example

We will search for the number 85 out of a range from 1-85

Our strategy is to start with the first number of the array and move up by one until ~~our target~~ we find our target number

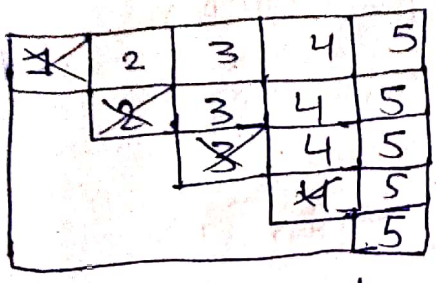


fig: linear search.

In round 1, we choose number one, that's not correct, so we move round 2 and eliminate number 1. step 2 is also not correct and we continue all the way until we select number 5. This method is called Linear search method. The Big O notation for linear search is O(N). The complexity is directly related to size of number.

28) Explain divide and conquer strategy algorithm.

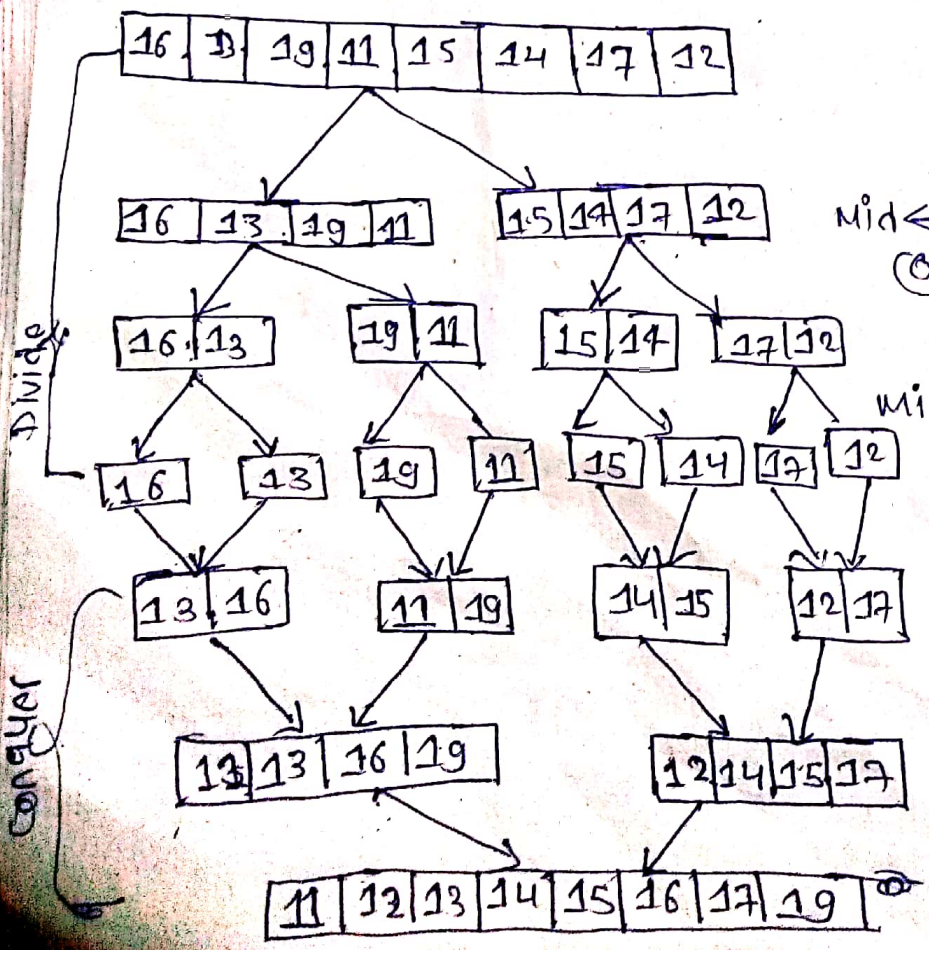
- A classical sequential sorting algorithm using divide and conquer approch.
- Unsorted list is first divided into half. Each half is again divided into two, continue ~~unit~~ until individuals numbers are obtained.
- Then pairs of numbers are combined (merged) into sorted list of two numbers. pairs of these list of two are combined to form four and continue until all numbers are merged into a single list.

Example

16, 13, 19, 11, 15, 14, 17 and 12

mid ← (left + right)/2

mid ← 7/2 = 3.5 = 3



Mid ← (left + right)/2

(0+3)/2 = 3/2 = 1

Mid ← (left + right)/2

0+1/2 = 0