TREE

Date .....

tree can be defined as a collection of entities (nodes) linked together to simulate a hierarchy.

Degree of tree :- max degree among all nodes.

* max^m no. of nodes possible at any level i is 2^i.

* max^m no. of nodes of height $h = 2^{h+1} - 1$
* min^m no. of nodes of height $h = h+1$

i.e $n = 2^{h+1} - 1$

$n+1 = 2^{h+1}$

$\log_e(n+1) = \log_e 2^{h+1}$

$\log_2(n+1) = h+1$

$$h = \log_2(n+1) - 1$$

h → height

n → no of node.

	max nodes	min nodes
Binary tree	$2^{h+1} - 1$	$h+1$
Full Binary tree	$2^{h+1} - 1$	$2h+1$
Complete Binary tree	$2^{h+1} - 1$	2^h

	min height	max. height
Binary tree	$\lceil \log_2(n+1) \rceil - 1$	$n-1$
Full Binary	$\lceil \log_2(n+1) \rceil - 1$	$\frac{n-1}{2}$
Complete Binary	$\lfloor \log_2(n+1) \rfloor - 1$	$\log n$

Spiral

Date .....

perfect Binary tree:-

all internal nodes have 2 children and all leaves are at same level.

④ Complete Binary tree:-

all level are compleately ficlled (except possibly the last level) and the last level have node as left as possible.

Spiral

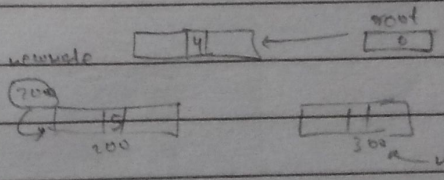
⊕ Implementation using recursion

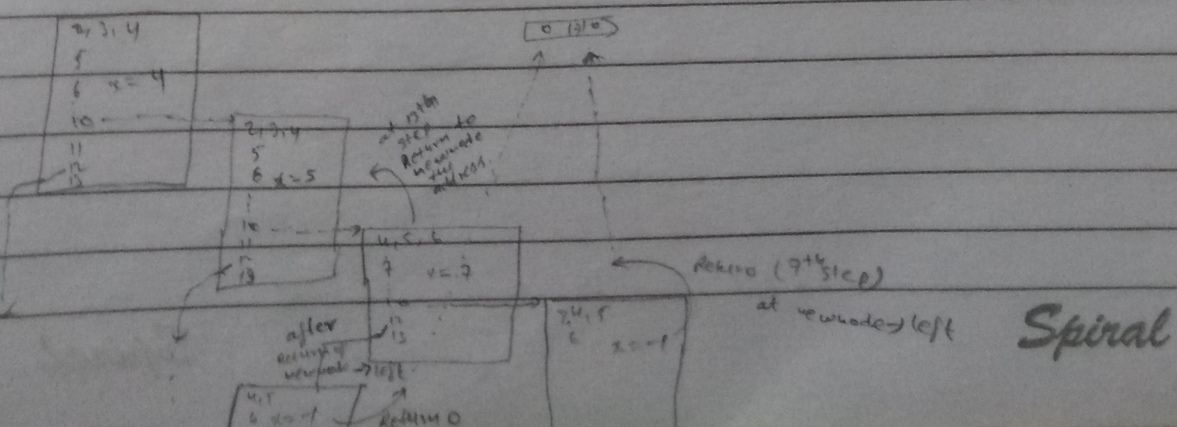
Date .....

```
void main()
{
    struct node *root;
    root = 0;
    root = create();
}
```

```
struct node
{   int data;
    struct node *left, *right;
};
```

```
struct node* create()
{
1   int x;
2   struct node *newnode;
3   newnode = (struct node*) malloc (sizeof (struct node));
4   printf("Enter data (-1 for no node) : ");
5   scanf("%d", &x);
6   if (x == -1)
7   { return 0;
    }
8   newnode -> data = x;
9   printf("Enter left child of %d", x);
10  newnode -> left = create();
11  printf("Enter right child of %d", x);
12  newnode -> right = create();
13  return newnode;
}
```





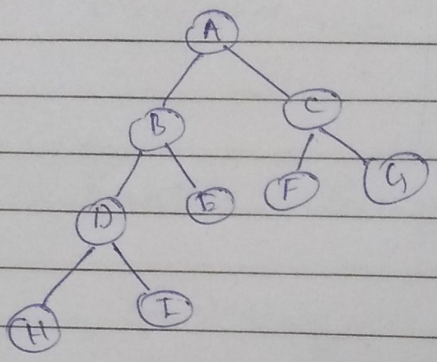
Spiral

5.4

{frist make complete Binary tree}

Array Representation of Binary tree Date .....

(sequential Representation)



case I

if a node is at i^{th} index :-

→ left child would be at = $[(2*i)+1]$

→ Right child would be at = $[(2*i)+2]$

parent would be at = $\left[\frac{(i-1)}{2}\right]$

check for B → is at index 1
so, i=1

so, left child = $(2*i)+1$ = 3rd index
so, D

right child = $(2*i)+2$ = 4th index
= E

A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

case I

A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9

case II

case II

node is at i^{th} index

left child at = $(2*i)$

right child at = $[(2*i)+1]$

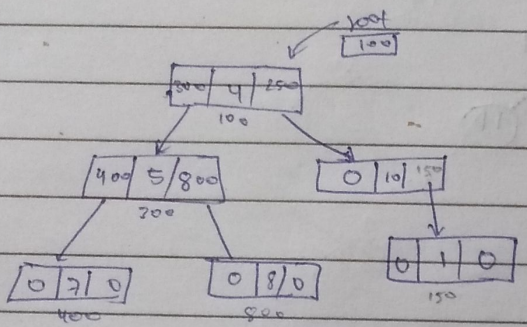
parent at = $\left[\frac{i}{2}\right]$

Spiral

Binary tree x traversals

Date .....

```
void main()
{
    struct node * root;
    printf("preorder is :");
    preorder (root );
    postorder ( root );
    inorder ( root );
```



preorder → Root, Left, Right

4, 5, 7, 8, 10, 1

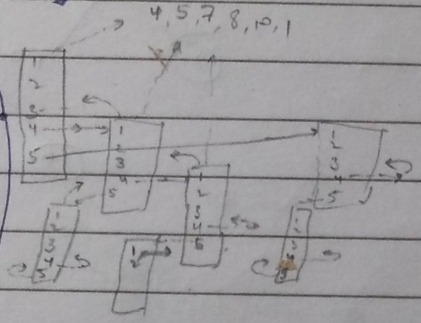
postorder → L, Right, Root

7, 8, 5, 1, 10, 4

inorder → L, Root, Right

7, 5, 8, 4, 10, 1

```
void preorder ( struct node* &root )
{
    if (root == 0) } Base of recursion
        return;
    printf("%d", root -> data);
    preorder (root -> left);
    preorder (root -> right);
}
```




```
void postorder ( struct node* &root )
{
    if (root == 0)
        return;
    postorder (root -> left);
    postorder (root -> right);
    printf("%d", root -> data);
}
```

```
void inorder ( struct node* &root )
{
    if (root == 0)
        return;
    inorder (root -> left);
    printf("%d", root -> data);
    inorder (root -> right);
}
```

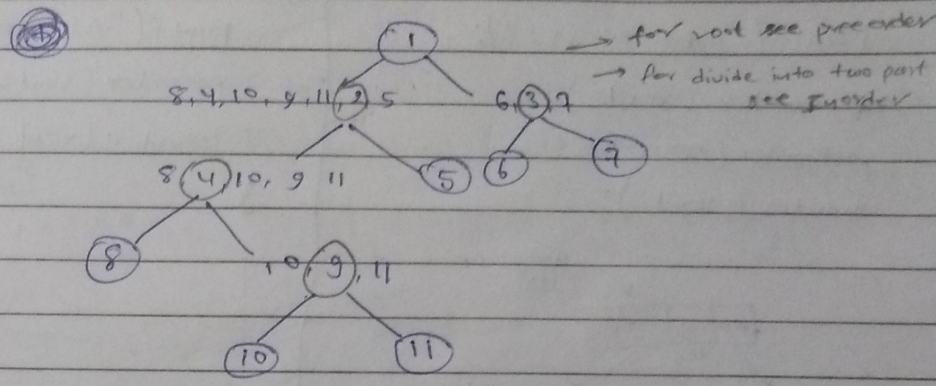
Spiral

Construct a Binary tree from preorder & Inorder

Date .....

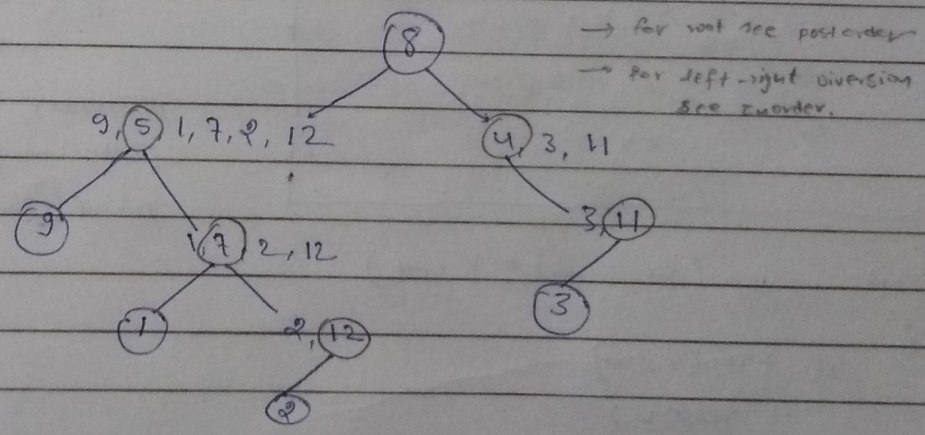
preorder :- 1, 2, 4, 8, 9, 10, 11, 5, 3, 6, 7 (Root, L, R)

Inorder :- 8, 4, 10, 9, 11, 2, 5, 1, 6, 3, 7 (L, Root, R)

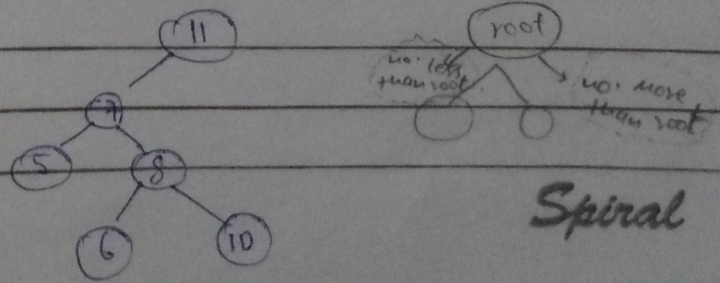


postorder :- 9, 1, 2, 12, 7, 5, 3, 11, 4, 8 (L, Right, Root)

Inorder :- 9, 5, 1, 7, 2, 12, 8, 4, 3, 11 (L, root, Right)



★ Draw Binary search tree by inserting the number :-
11, 7, 5, 8, 6, 10



Spiral

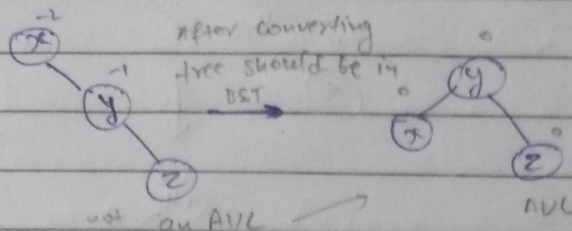
AVL Tree

Date .....

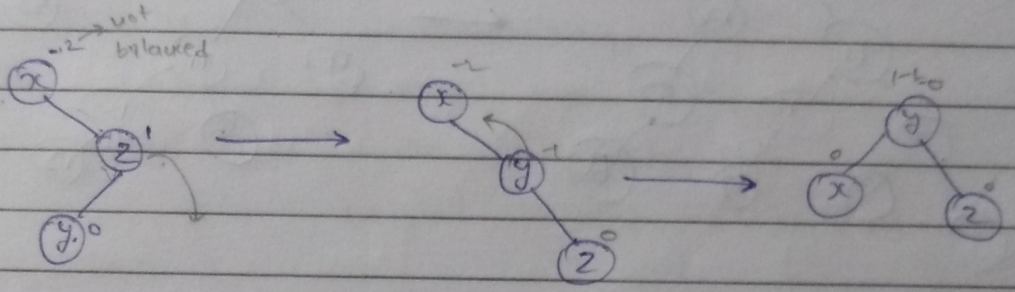
(1) It is a BST

(2) difference between left subtree and right subtree must be = {-1, 0, 1}.

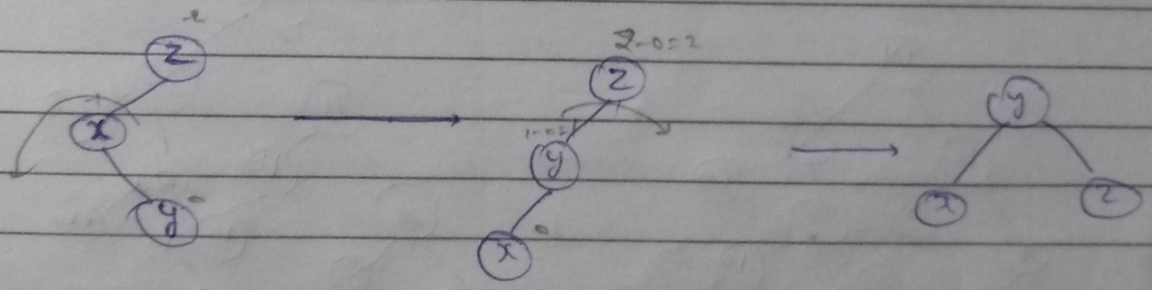
ex :-



ex :-



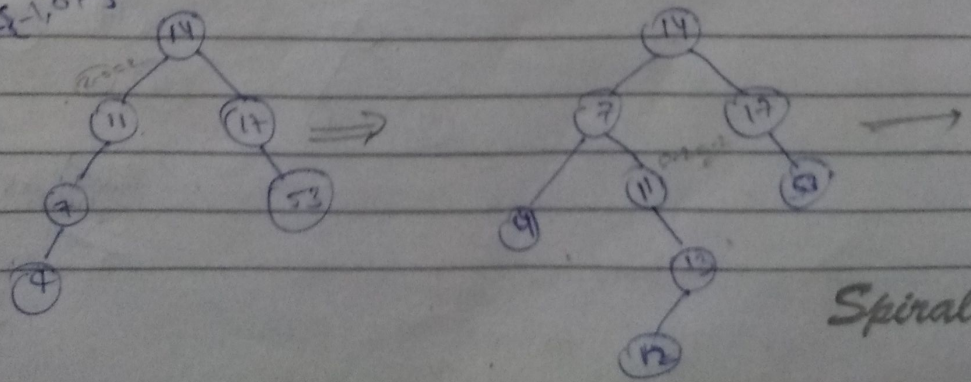
ex :-



(*) Construction of AVL Tree by inserting the following data.

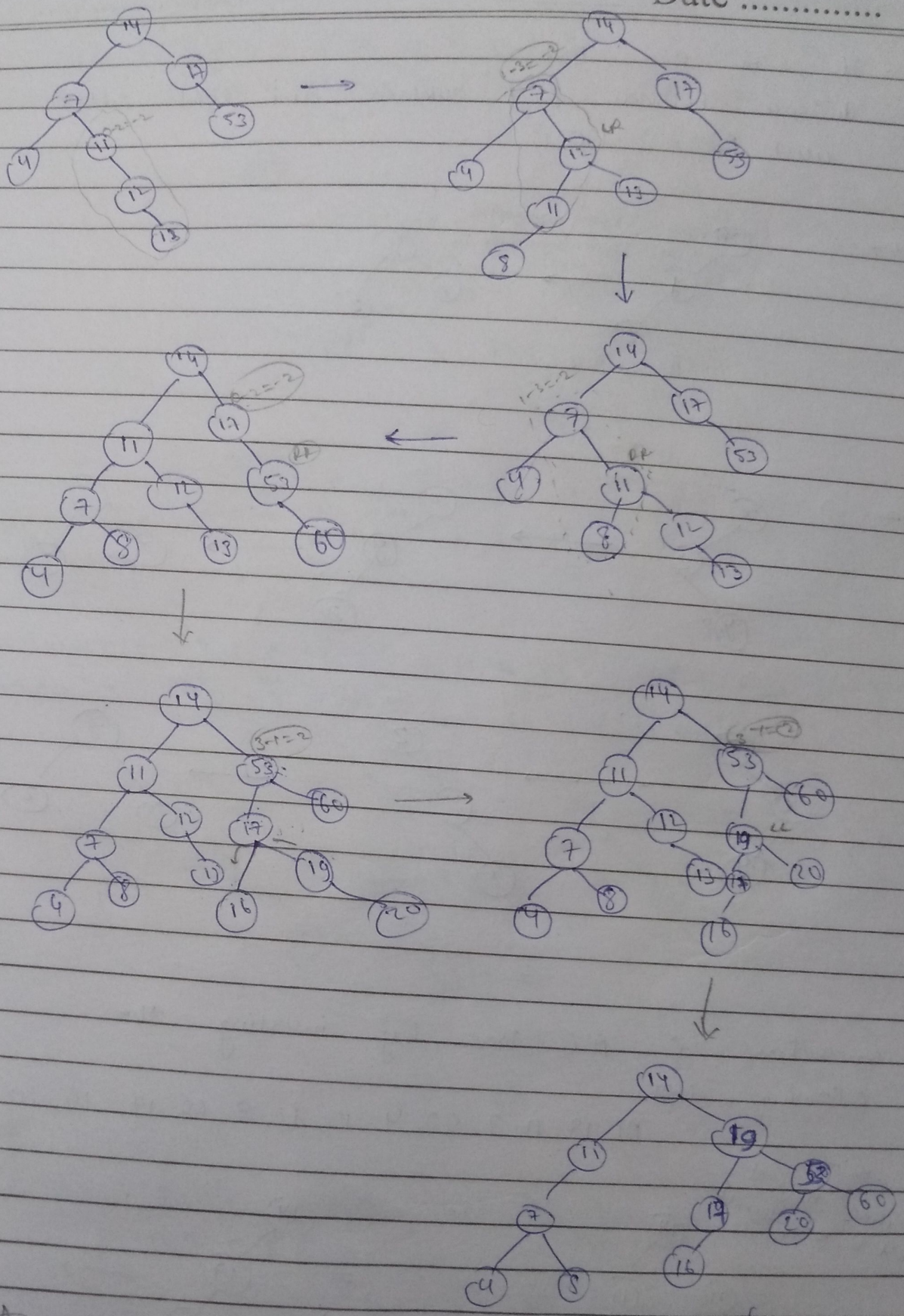
14, 17, 11, 7, 53, 4, 13, 12, 8, 60, 19, 16, 20

insert are by one and check weight should be {-1, 0, 1}



Spiral

Date .....



similarly for Deletion like
Delete and then Balance the factor.

Spiral

INTRODUCTION TO Red-Black tree

Date .....

→ it is a self Balancing BST

→ Every node is either Black or Red

→ every ~~node~~ a leaf which is NIL is Black

→ if node is Red then its children are black.

→ every path from a node to any of its descendent NIL node has same no. of Black nodes.
i.e. black height should be balanced

NOTE : A Tree containing all the node black is a Red - Black tree.

① root always be Black
② Nill is Black always.

(*) Insertion in Red - Black Tree Date .....

(1) If tree is empty, create newnode as root node with color black.

(2) If tree is not empty, create newnode as ~~black~~ leaf node with color (Red)

(3) If parent of newnode is black then exit.

(4) If parent of newnode is (Red) then check the ~~co~~ color of parent's sibling of newnode

(a) if color is black or NULL then do suitable rotation & recolor.

(b) if color is Red then color & also check if parent's parent's parent of newnode is not root node. then recolor it & recheck.

Spiral

B- Tree

Date .....

property

order → A max^m number of children a tree's node can have.

① balanced m-way tree. (m = 5) → min^m node = $(\frac{m}{2}) = 2.5 = 2$

② generalization of BST in which a node can have more than one key & more than 2 children

③ maintains sorted data

④ all leaf node must be at same level.

- B- Tree of order m has following properties

→ every node has max m child.

→ Min. children :-

leaf → 0

root → 2

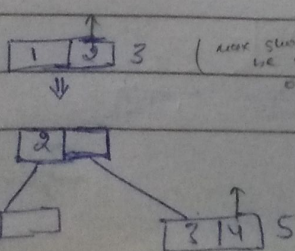
internal node $\lceil \frac{m}{2} \rceil$ for m=5 $\lceil \frac{5}{2} \rceil = (2.5) = 3$

→ every node has max. (m-1) keys.

→ Min. keys :- root node → 1

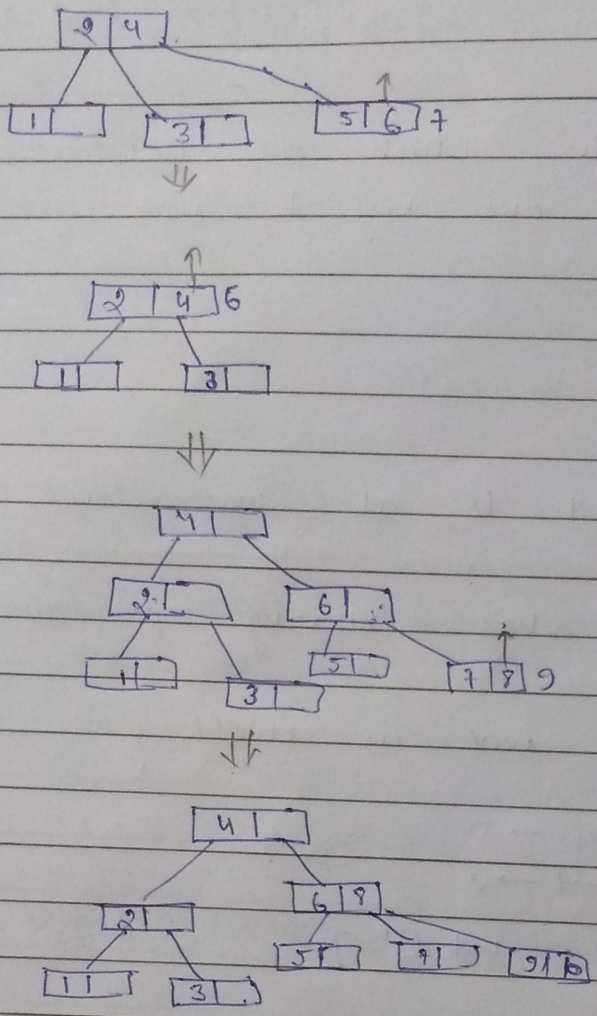
⑤ creat a B- tree of order 3 by inserting value from 1 to 10.

⇒ given Max Key = (m-1) = 3-1 = 2



Spiral

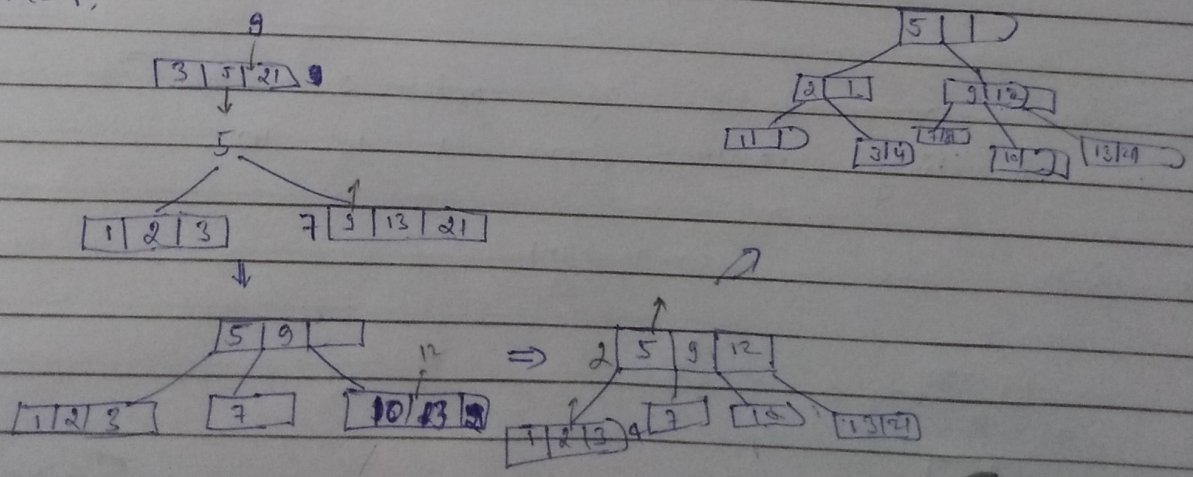
Date .....



(2) Construct a B-Tree of order 4 with following set of data.

5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8

⇒ M=4,



Spiral

Deletion in B-Tree

Date .....

case 1 if target key is in leaf node

case 2 if target key is in internal node.

order (M) = 5

min children = $\lceil \frac{M}{2} \rceil = 3$ (root)

max children = 5

min keys = $\lceil \left(\frac{M}{2}\right) - 1 \rceil$ (that is min 2)

max keys = 4 (ceiling)

Leaf node

- that leaf node contain more than minimum no. of keys → delete simply
- that leaf node contain of keys. → borrow from left / borrow from right / or merge the both minimum key after bringing parent key and then delete that

Case 2 check for internal node

bring the greatest from left or smallest from right leaf.

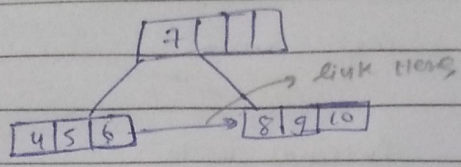
and both have minimum key then merge that both leaf node, and delete that parent.

Spiral

B^+ Tree

Date .....

Ex :-



Heap

Date .....

Max Heap

⇒ for every node i, the value of node is less than or equal to its parents value.

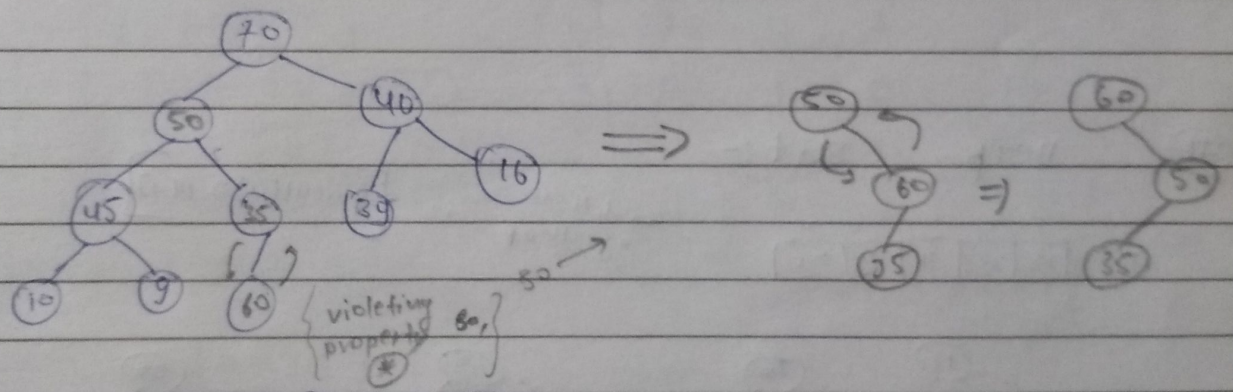
Min Heap

for every node i, the value of node is greater than or equal to its parent value

(*) $A[parent(i)] \geq A[i]$ { except root node } $A[parent[i] \leq A[i]$

tree should be complete tree. or as left as posible

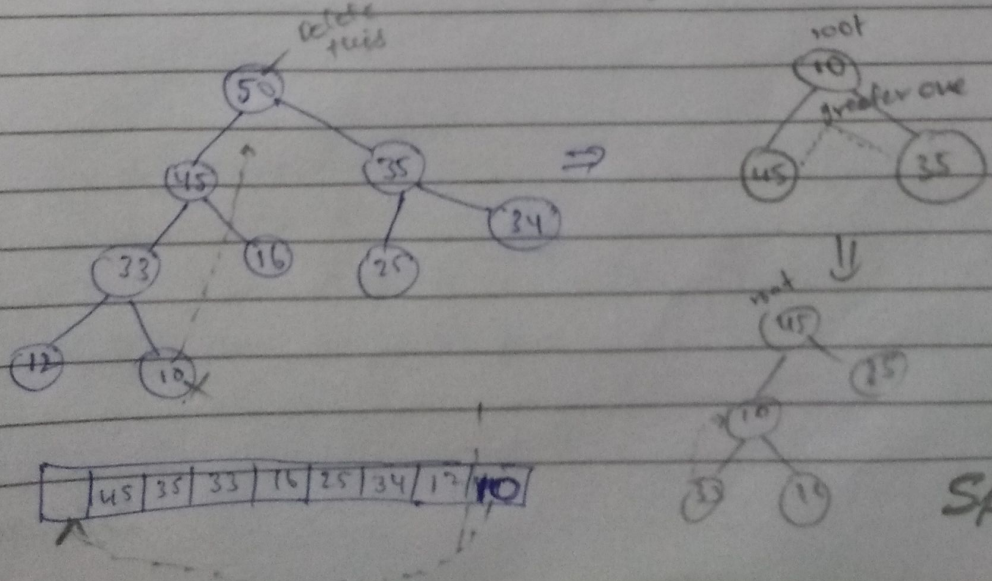
Insertion



70	50	40	45	35	39	16	10	9	60
0	1	2	3	4	5	6	7	8	9

→ for parent (i) = [i/2] = 4

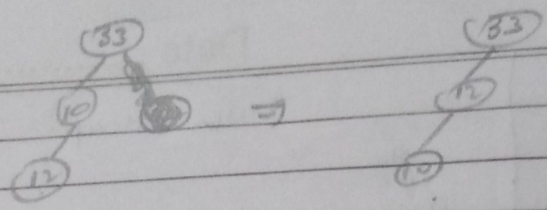
Deletion



	45	35	33	16	25	34	12	10

Spiral

Date .....



Heap

45	33	35	17	16	25	34	10
0	1	2	3	4	5	6	7

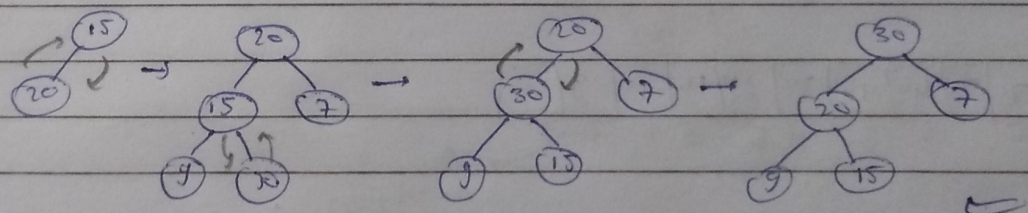
left child of i = $2*i+1$

right child of i = $2*i+2$

(#) Heap Sort :-

	0	1	2	3	4
A	15	20	7	9	30

condition of Heap is, [parent[i] ≥ A[i]]



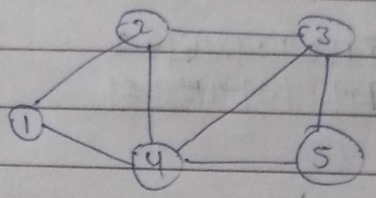
Representation of graph in computer

Date .....

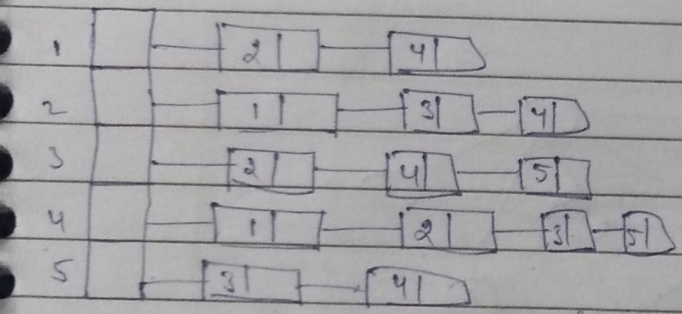
① Adjacency Matrix :-

② Adjacency list :-

① Adjacency matrix :-



② list



① (represent edges to its of)

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

5×5

$O(n+2e)$ — no. of edges.

space complexity $= O(n^2)$

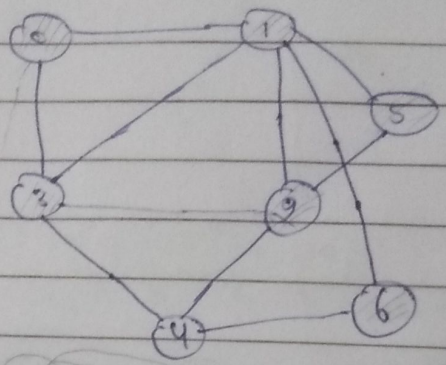
graph traversal :-

① BFS → (level order) → breath first search

② DFS → depth first search

Spiral

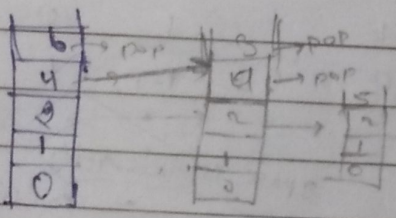
Date .....

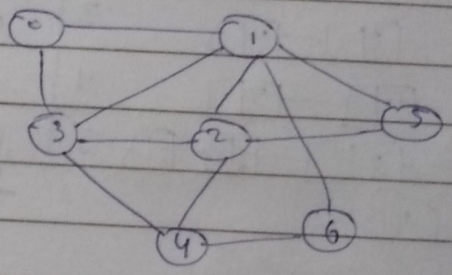


Queue :- 0 1 3 2 5 6 4

Print :- 0 1 3 2 5 6 4

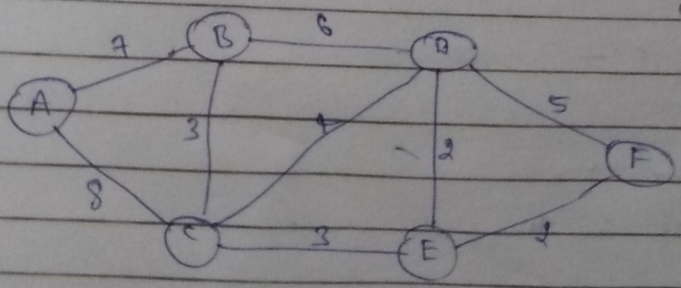
② DFS :-





result → 0, 1, 2, 4, 6, 3, 5

① Prim's Algorithm :- There should not be cycle and there should be one of node, but



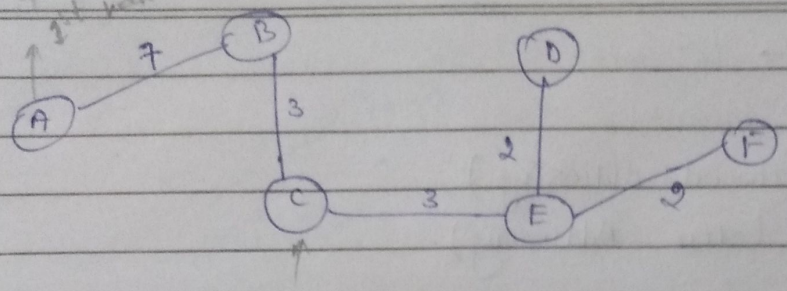
at first remove the duplication of edges.

Spiral

No cycle allowed

Date .....

take 1st node then find



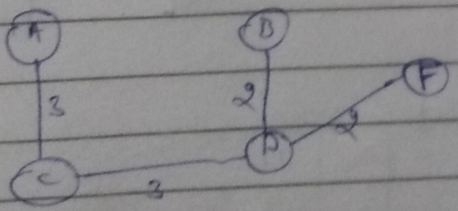
If you are here then see for previous node covered also that which has minimum weight of edge then take that node.

(2) By Kruskals Algorithm :-

(i) first write in increasing order of edge weight.

BD = 2
DE = 2
AC = 3
CD = 3
BE = 4
AB = 5
BC = 6
AF = 7
FC = 8

(ii) take from above and make graph make sure that there should not be cycle.



Spiral

Hasing

Date .....

(*) Type of Hashing :→

(i) open hasing (closed Addressing)
(ii) closed Hashing (open Addressing)

A:- 3, 2, 9, 6, 12, 13, 7, 12 $\boxed{m = 10}$

use division method and closed addresing to store these values,

given, ~~h(k) = 2k + 3~~ $\boxed{m=10}$ Hash table size 10

key	location (u)	probs (for linear Hashing)
3	[(2×3)+3] % 10 = 9	1
2	[(2×2)+3] % 10 = 7	1 (one or more shifted)
9	[(2×9)+3] % 10 = 1	1
6	= 5	1
11	= ~~5~~	2
13	= 9	2 (2 times shifted)
7	= 7	2 (6 time shifted)
12	= 7	6 →

0	
1	9
2	
3	
4	
5	6 → 11
6	
7	2 → 7 → 12
8	
9	3 → 13

separate chaining

linear probing method

0	12
1	9
2	12
3	
4	
5	6
6	11
7	2
8	7
9	3

Spiral

Date .....

(ii) Quadratic probing :- store the value not linearly but according to given equation

$(h(k) + i^2)$, i = no. of index.

or $(h(k)+i)\%m$ where $i \in [0, m-1]$

= index of prog next if slot is not empty i++

same as linear nothing before but shifting taken place according to this value,

Let $h(k) = 2k+3$

$m = 10$

(iii) Double Hashing :-

$h(k) = 2k+3$ $m=10$ $h_2(k) = 3k+1$

If collision occurs then insert k at first free place from $(u + V \times i) \% m$

$V = h_2(k) \% m$

square as earlier but also

like in 1st Hashing

for 11 we will find the next place as,

$(5 + V \times i) \% m$ $h_2(k) = 3 \times 11 + 1 = 34$

i) $(5 + 4 \times 0) \% m$ $V = 34 \% 10 = 4$

ii) 5 → not empty

Now i=1

$(5 + 4 \times 1) \% m$

$= 9$ → not empty

Now i = 2

$(5 + 4 \times 2) \% 10$

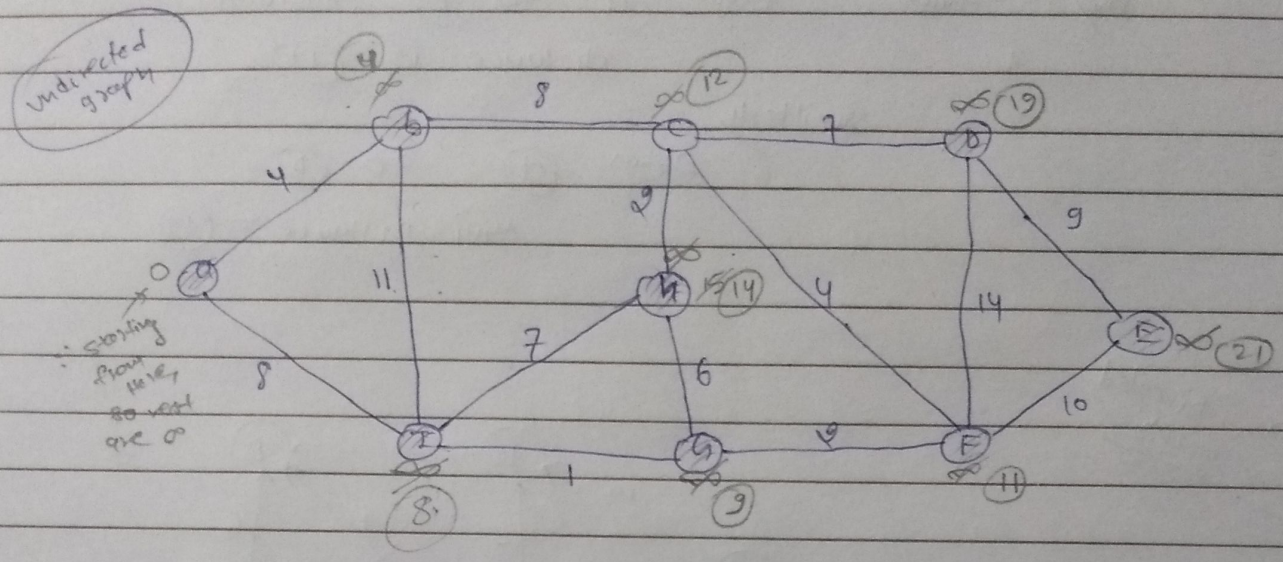
$= 3$ → Now insert and probe = 3

Spiral

for finding shortest path in a graph,

Dijkstra Algorithm

Date .....



in question source / starting node will be given, as

Here we will start from a

source
previous calculated weight
cost of destination to source

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

d(v) = du + c(u,v)

~~at~~ for (b) :

0 + 4 < ∞

true, so,

d(v) → (b) = 4

for (I) : from a

0 + 8 < ∞

true, so,

d(v) → (I) = 8

from (b)

4 + 11 < ∞

15 <]

But 8 is less so 8 is ok

Spiral

Date .....

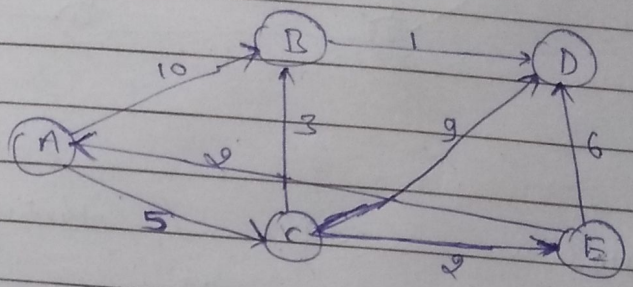
Hence after seeing the diagram we can say that from (a) to (F) minimum distance is (11)

Similarly

from (a) to (D)

Mini Distance = (19)

for Directed graph



let source vertices is (A)

	A	B	C	D	E
A	0	∞	∞	∞	∞
A C		10	5	∞	∞
A C E		8		14	7
A C E B		8		13	
A C E B D				9	

selected min

A+B

A+C

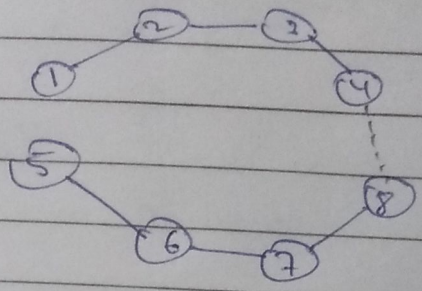
modified path from A to D 5+9

should be minimum cost

Spiral

Disjoint set data Structure. Date .....

① Find
② Union



$S_1 = \{1, 2, 3, 4\}$.

$S_2 = \{5, 6, 7, 8\}$.

$S_1 \cap S_2 = \phi$ so ~~disjoint~~ set

Now when we add (4,8) edge Here then see both are from different disjoint set so union perform

i.e. $S_R = \{1, 2, 3, 4, 5, 6, 7, 8\}$

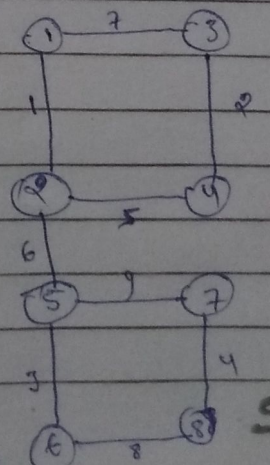
Note, as if both belongs to same set will say cycle exists so don't make the edge

Ex ÷ Now (1,5) e
so, cycle exist,
don't combine

Ex ÷ How we will find a cycle in any graph.

⇒ first take universal set

$U = \{1, 2, 3, 4, 5, 6, 7, 8\}$.



Step 01 ÷ take edge one by one (according to edge weight and) remove from universal set.

Q8

$S_1 = \{1, 2\}$ $S_2 = \{3, 4\}$ $S_3 = \{5, 6\}$.

Spiral

Date .....

$S_4 = \{7, 8\}$.

for edge 5,

now, (2, 4)

$\in S_1$ $\in S_2$

so union it

i.e. $S_5 = \{1, 2, 3, 4\}$.

Now edge 6,

(1, 5)

$\in S_5$ $\in S_3$

so, union it

$S_6 = \{1, 2, 3, 4, 5, 6\}$.

Now we have, $S_6 = \{1, 2, 3, 4, 5, 6\}$

$S_4 = \{7, 8\}$.

similarly for edge 7

(1, 3)

$\in S_1$ $\in S_6$

so,

there is cycle don't combine it

for 8,

(6, 8)

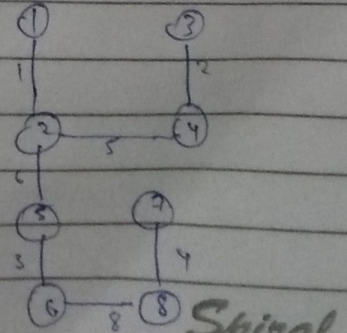
$\in S_6$ $\in S_8$

union it

$S_7 = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

just like kruskal algorithm

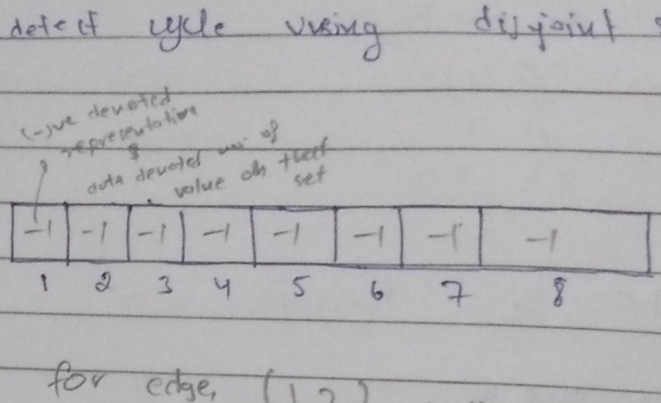
Hence, final Diagram or MST will be,

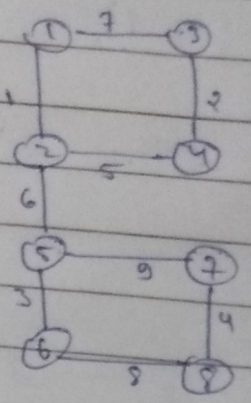


Spiral

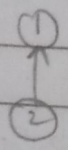
Date .....

② detect cycle using disjoint set

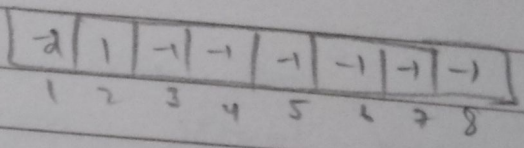




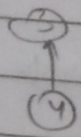
for edge, (1,2) both representative is -1
so, both are parent Now union it

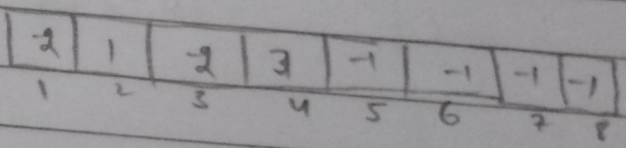


and update

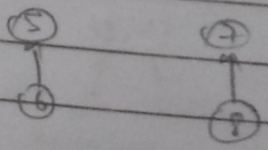


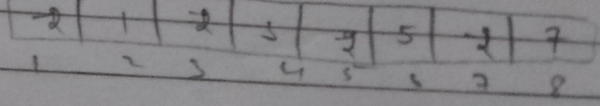
Similarly for edge 2, (3,4)





for 3, (5,6) & 4, (7,8)



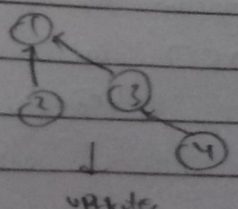


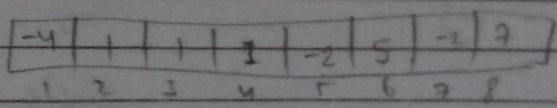
Also, for 5, (2,4)

representative = 1

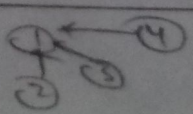
representative = 3,

so, union,





update,



Spiral

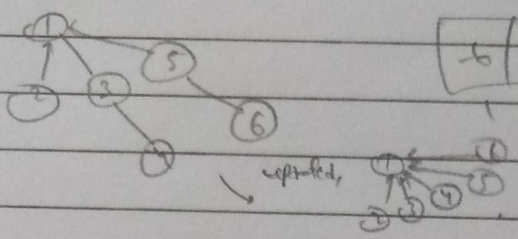
Date .....

for 5. (2, 5)

representative = 1

representative is itself.

So, union,

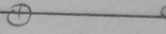


-6	1	1	1	1	1	-2	7
1	2	3	4	5	6	7	8

updated,

for. 7 (1, 3)

representative representative



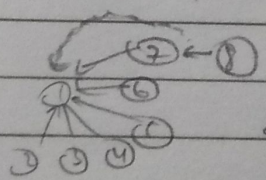
So, in same set

can't combine Because form cycle,

for 8. (6, 8)

rep.. = 1 rep.. = 7

union,



after update

will repeat 1

-8	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8

=

Here MST found

Spiral