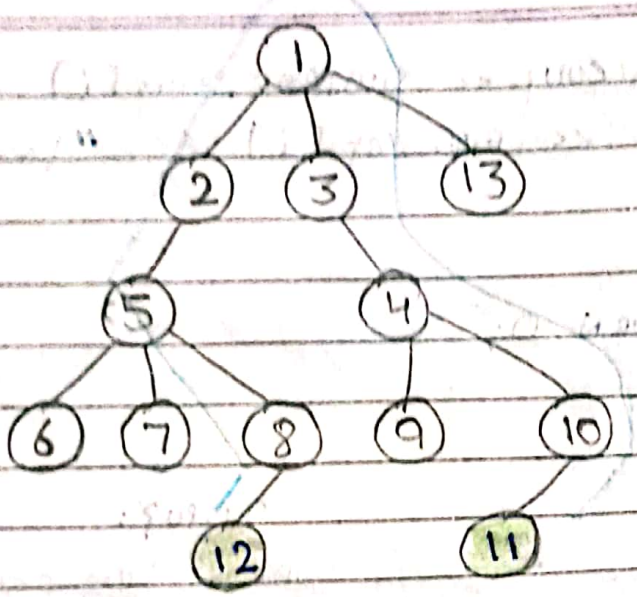
DIAMETER OF A TREE

Diameter is the length of the longest path b/w any 2 nodes in the tree.

~~AVL~~ TREE:



Here, (12) → (11) is the diameter of the tree.

APPROACH 1: (BRUTE FORCE) T.C: $O(n^2)$

→ Assume every node as ~~root node~~ and calculate ~~maximum~~ maximum depth from it.

e.g ROOT → (1) ⇒ 4 (MAX-DEPTH)
ROOT → (2) ⇒ 5 (" ")
|
ROOT → (12) ⇒ 8 (" ")
ROOT → (11) ⇒ 8 (" ")

Now, the maximum of its MAX-DEPTH will be, Diameter of this tree.

In this we will run DFS for every node $O(n)$ and T.C for running DFS is almost around $O(n)$
∴ Total T.C ⇒ $O(n^2)$.

APPROACH 2: T.C : O(n)

-> With any node as root node find the MAX-DEPTH node.
e.g: For (1) => (12) or (11)
For (2) => (11)
For (3) => (12).

∴ We can observe that it will always find one end of our diameter.

-> Now with the MAX-DEPTH node as root node find maximimum depth and that will be our diameter as well.

CODE

-> Maximum capacity of nodes.
```
CONST INT N = 1e5+10;
```
-> Adjacency list
```
VECTOR<INT> G[N]
```
-> Array to store depth
```
INT DEPTH[N];

VOID DFS(INT VERTEX, INT PAR = -1)
{
  FOR(INT CHILD : G[VERTEX])
  {
    IF(CHILD == PAR) CONTINUE;
    DEPTH[CHILD] = DEPTH[VERTEX] + 1;
```

```
            DFS (CHILD, VERTEX);
        }
    }

INT MAIN ()
{
    INT n;
    CIN >> n;

    FOR (INT i=0; i<n-1; i++)
    {
        INT V1, V2;
        CIN >> V1 >> V2;
        G[V1].PB (V2);
        G[V2].PB (V1);
    }

    // Calling DFS function on 1 as
       root node
    DFS (1);

    // To store maximum depth when
       root node is 1.
    INT MX_DEPTH = -1;
    // To store maximum depth's node
       when root node is 1.
    INT MX_D_NODE;

    // Calculating mx_depth and mx_d_
       node when root is 1.
```

```
FOR (INT i=1; i <= n; i++)
{
    IF (MX_DEPTH < DEPTH[i])
    {
        MX_DEPTH = DEPTH[i];
        MX_D_NODE = i;
    }
    DEPTH[i] = 0;
}
```

→ Now running DFS on that max depth's node, which will be one end of our diameter

```
DFS (MX_D_NODE);
MX_DEPTH = -1;
```

→ Calculating mx_depth when that node is one end of our diameter

```
FOR (INT i=0; i <= n; i++)
{
    IF (MX_DEPTH < DEPTH[i])
    {
        MX_DEPTH = DEPTH[i];
    }
}

COUT << MX_DEPTH << "\n";

RETURN 0;
```

Page No.
Date

INPUT:

13
1 2
1 3
1 13
2 5
3 4
5 6
5 7
5 8
8 12
4 9
4 10
10 11

OUTPUT:

8