$= 0 + 2 = 2$

Binary lifting

There is a tree of size N, rooted at 0, answer Q queries: given V & K, find kth ancestor of v.

Brutforce (q * N)

each query in Brute force => O(N)

↓ we want

O(logn)

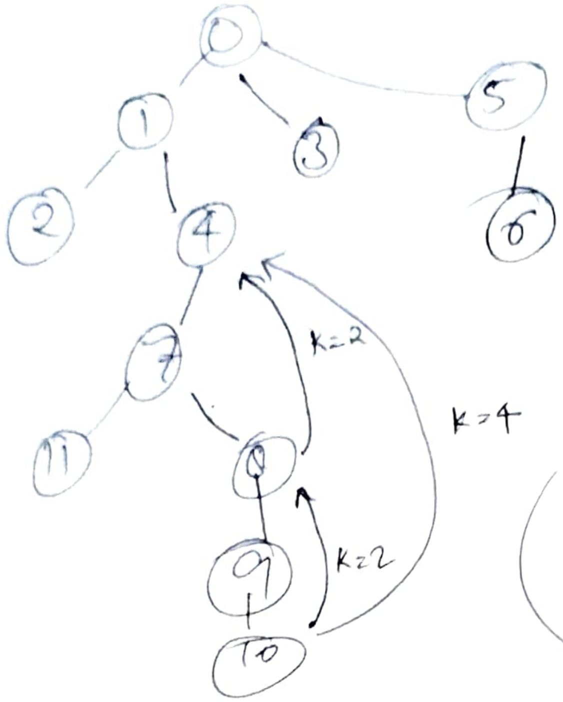
How to improve from O(N) to O(log(N)) ?

divide by 2

- BS
- D&C

Power of 2

- segment tree
- binary lifting
- sparse tables.



```
int up[N][log];
up[V][J] => 2^J-th ancestor of V
```

```
for v=0...... N-1
    up[V][0] = ~~up~~ parent[V]
    ~~for J=1~~
    up[V][1] = up[up[V][0]][0]
    up[V][2] = up[up[V][1]][1]
    up[V][3] = up[up[V][2]][2]
    . . . . .
```

```
for V = 0 -- N-1.
    up[V][0] = par[V]
    for J=1... logN-1:
        up[V][J] = up[up[V][J-1]]
                   [J-1]
```

T.C. & S.C => O(Nlog(N))

If Parent[i] < i ; then above code is true

else do as: ① run dfs.
② written below:

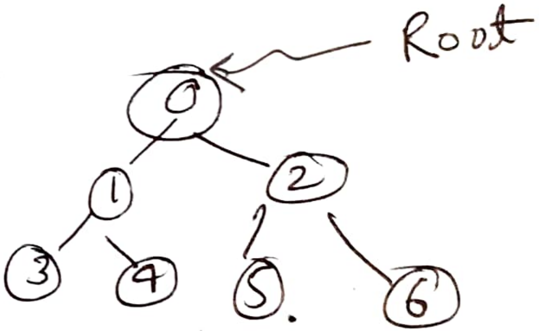
```
int up[N][log]:
up[V][J] ---- 2^J - th ancestor of V

for v = 0 ---- N-1:
        up[V][0] = Parent[V]
for J = 1 ...... LogG -1:
    for v = 0 . . . . . N-1:
        up[V][J] = up[up[V][J-1]][J-1]
```

Ex => 19 = 16+2+1 => ~~up~~ x = up[4][0]
4 =>
x = up[x][1]
x = up[x][4]

How to avoid extra (over) binary lifting :- 8

Ex ⇒ Suppose the root of binary in given tree has the following binary lifting procedure.



Root = 0 ; assume Parent[root] = -1

up[0][0] = Parent[0]

now if we apply same logic as in binary lifting code then there will be a problem.

Problem ⇒ There is no ancestor for the root.

Solution ⇒ Parent [root] = 0 ; // Magical Line

Intution ⇒ for above ex; root = 0

up[0][0] = Parent[0] = 0

up[0][1] = up[up[0][1-1]][1-1] = up[0][0] = 0

up[0][2] = up[up[0][2-1]][2-1] = up[0][1] = 0

⋮

up[0][J-1] = up[up[0][J-2]][J-2] = up[0][J-2] = 0

up[0][J] = up[up[0][J-1]][J-1] = up[0][J-1] = 0

⋮

up[0][log] = up[up[0][log-1]][log-1] = up[0][log-1] =

Conclusion ⇒ Similarly, for all the nodes which have no possibili of binary lifting Same logic will be applied.

dfs for Binary lifting General code.

```
void dfs(int node)
// assuming tree to be directed graph i.e. as in pointer wala tree.
{
    ~~up[node][~~
    for ( int v : children[node])
        up[v][0] = node;
        for(int j=1; j<log; j++){
            up[v][j] = up[up[v][j-1]][j-1];
        }
        dfs(b)
    }
    ~~dfs(~~.
}
```