USA Tutor

Cow at Large

Analysis by David Yang

Statement Summary

- We’re given a tree with N<=1e5 nodes
- Bessie is starting from node K
- Find the minimum amount of farmers that you need to block off Bessie.

Sample Input

1 2

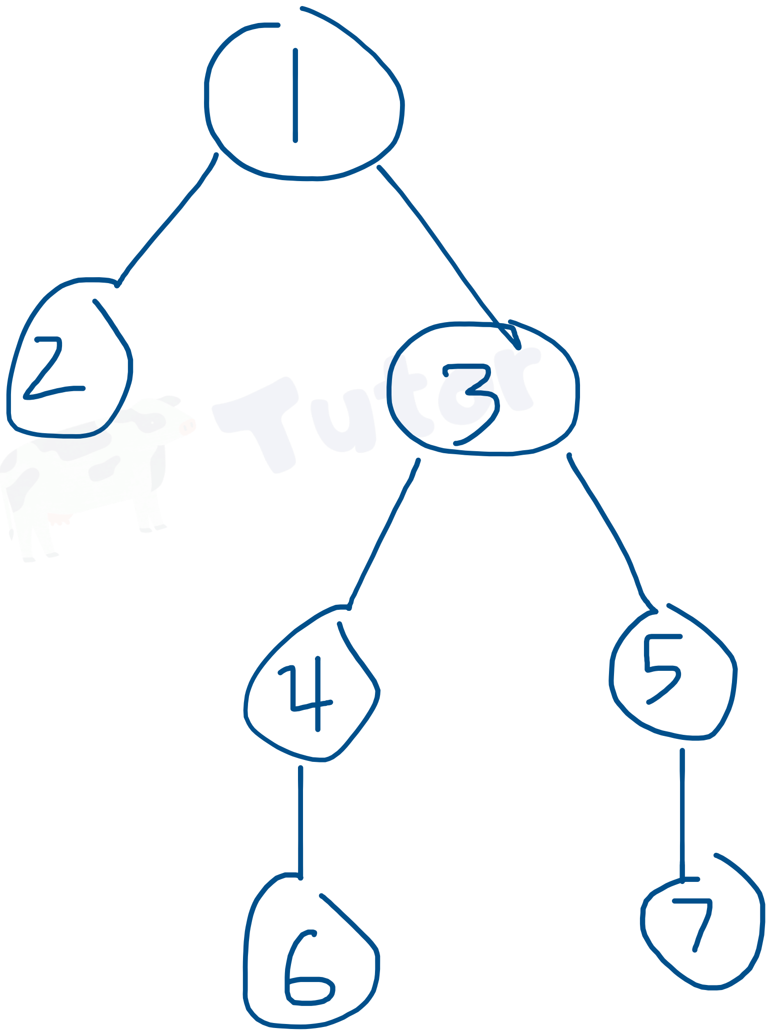
1 3

3 4

3 5

4 6

5 7



Sample Input

1 2

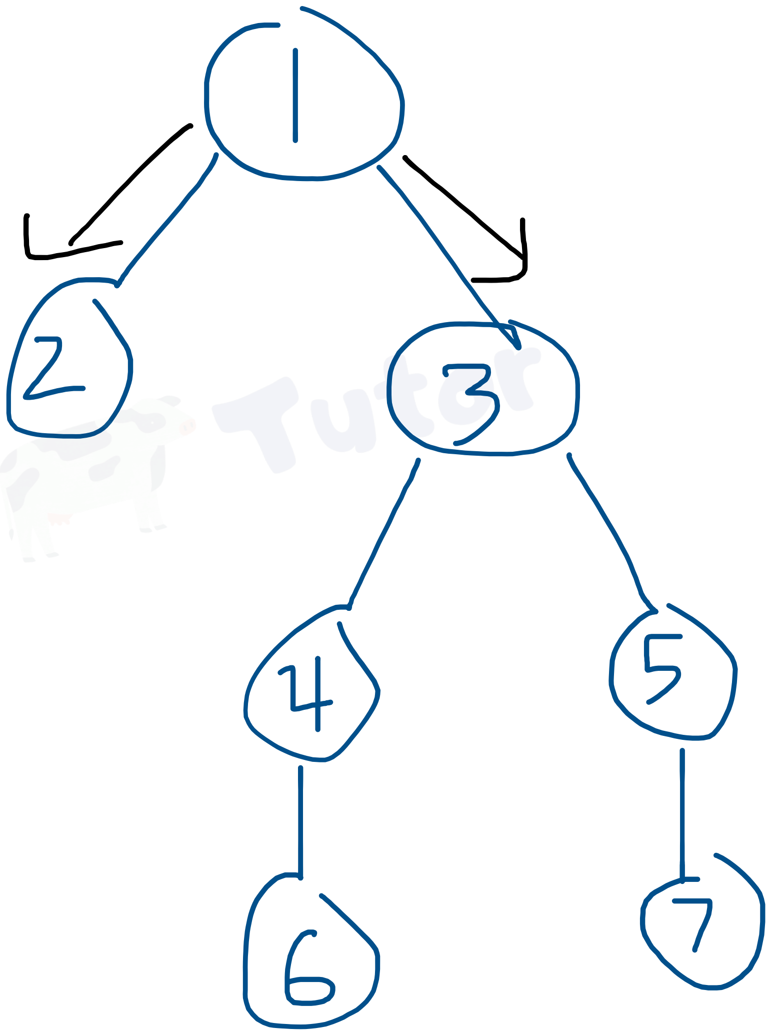
1 3

3 4

3 5

4 6

5 7



Sample Input

1 2

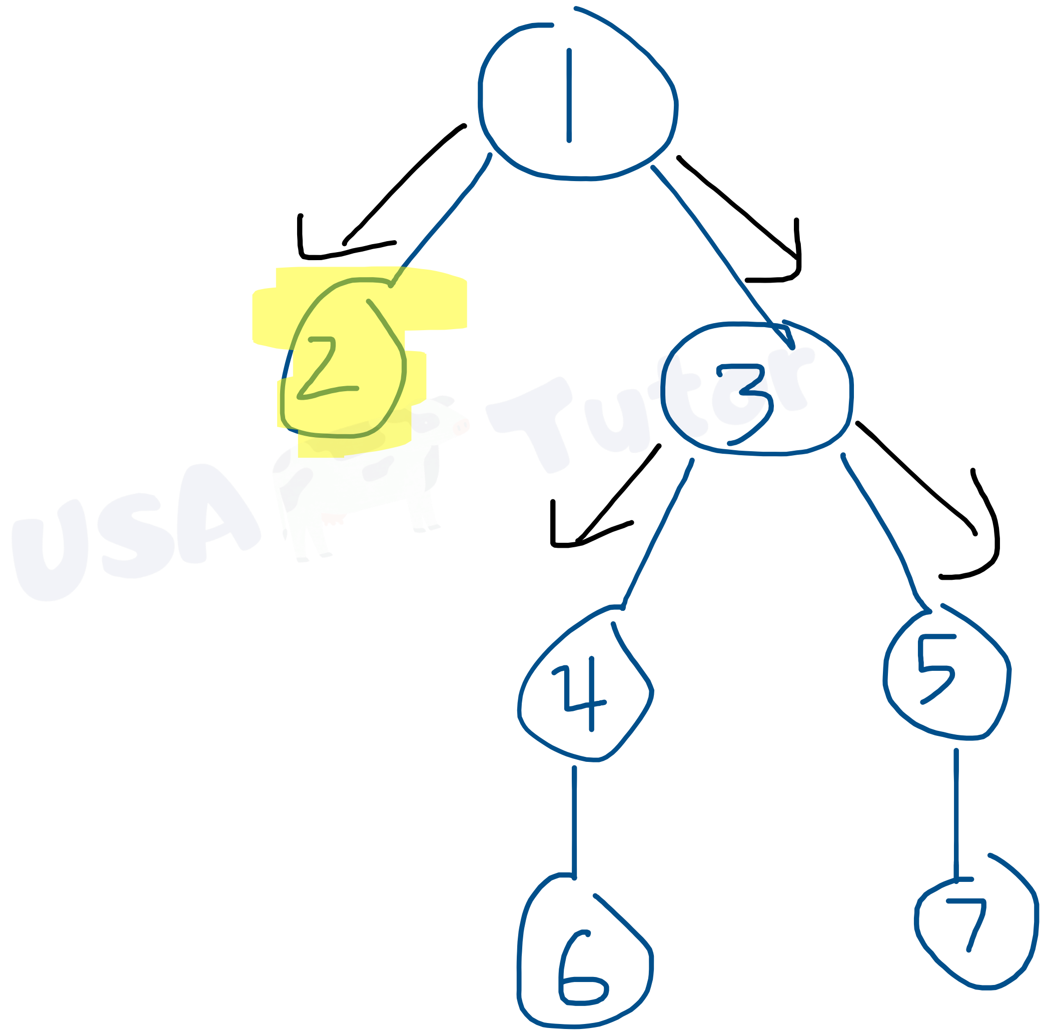
1 3

3 4

3 5

4 6

5 7



Sample Input

1 2

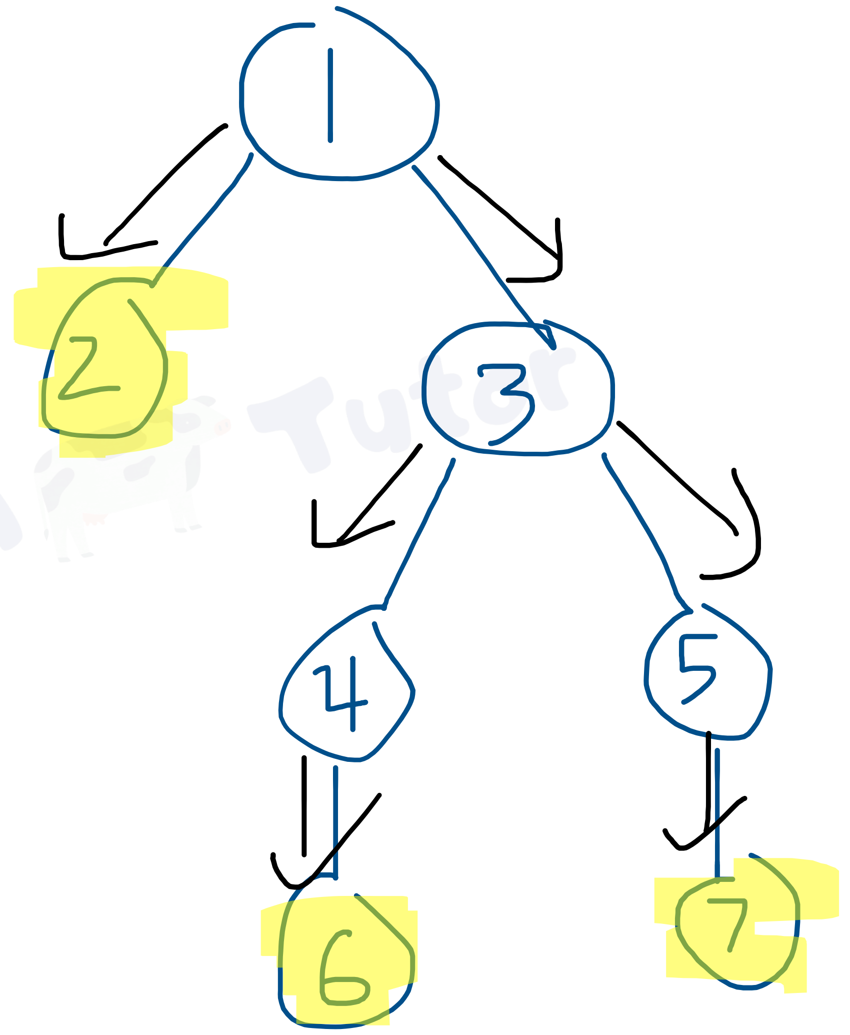
1 3

3 4

3 5

4 6

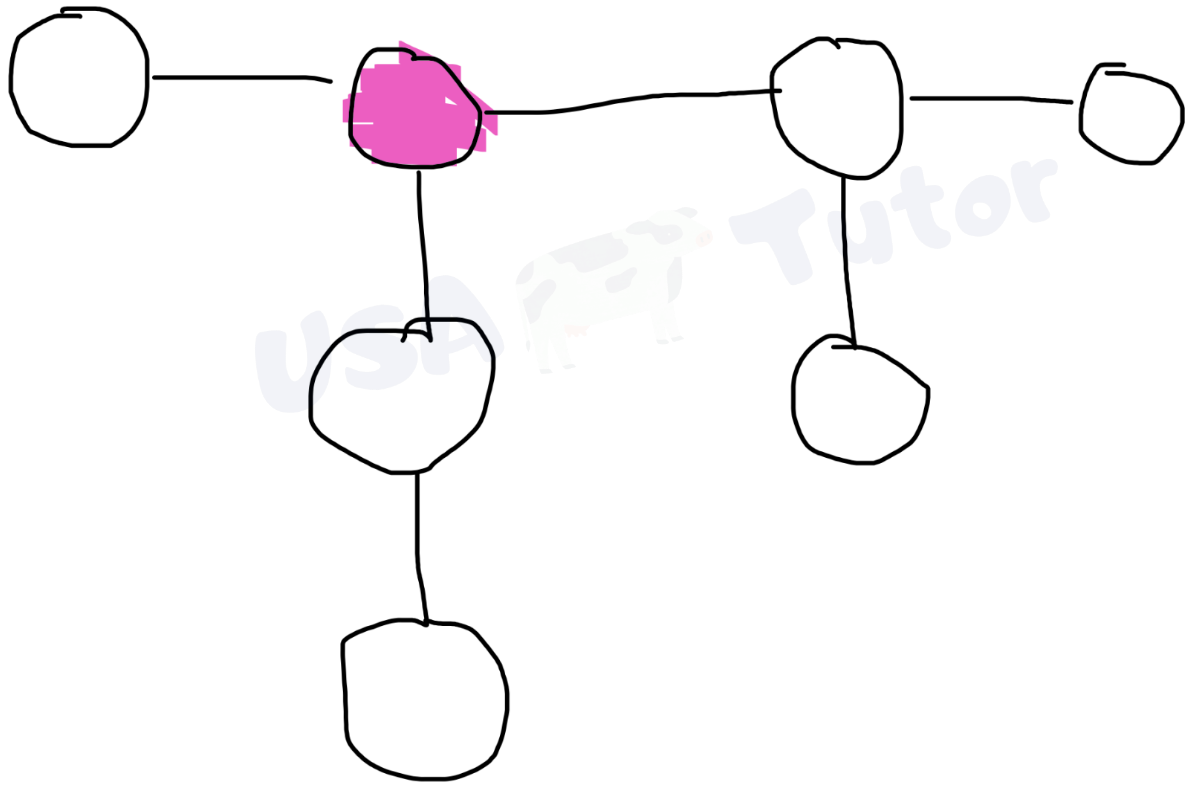
5 7



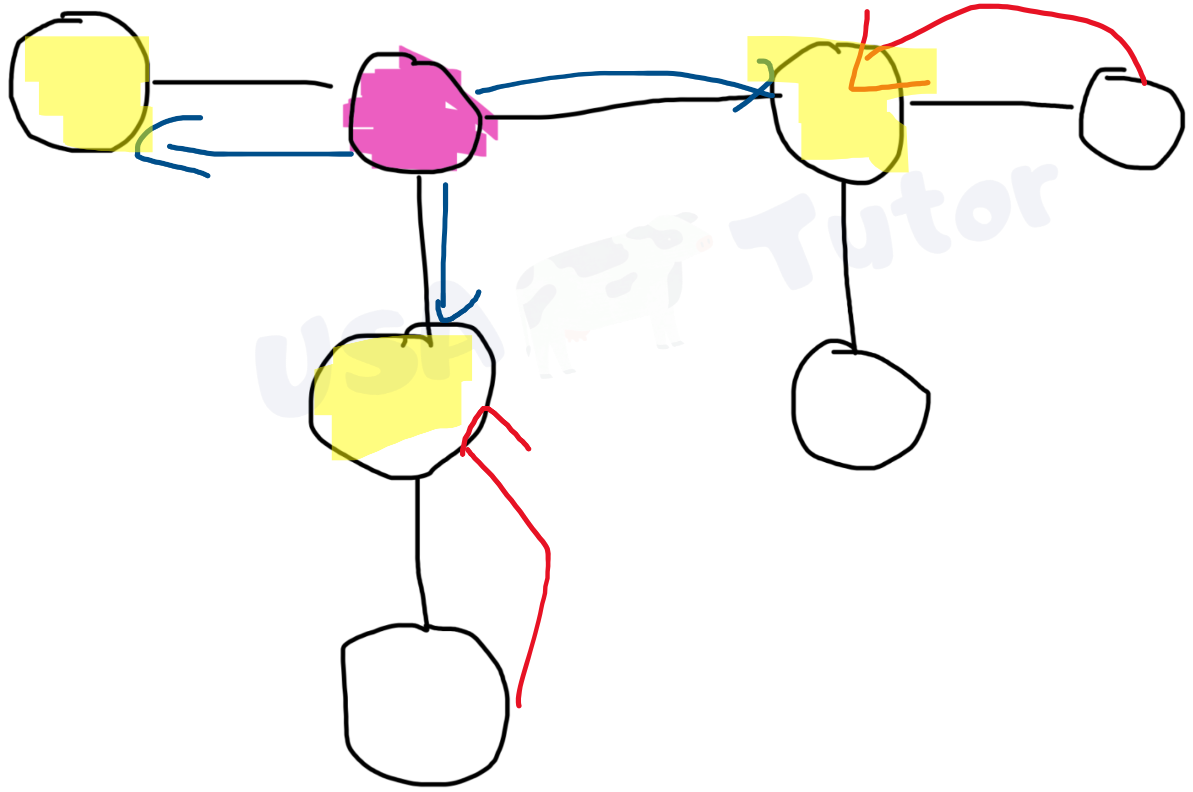
Analysis of Sample

- We see that Bessie is able to travel to 3 distinct leaf nodes
- Therefore, we must deploy 3 farmers
- What if we had a different sample?

Our Own Case

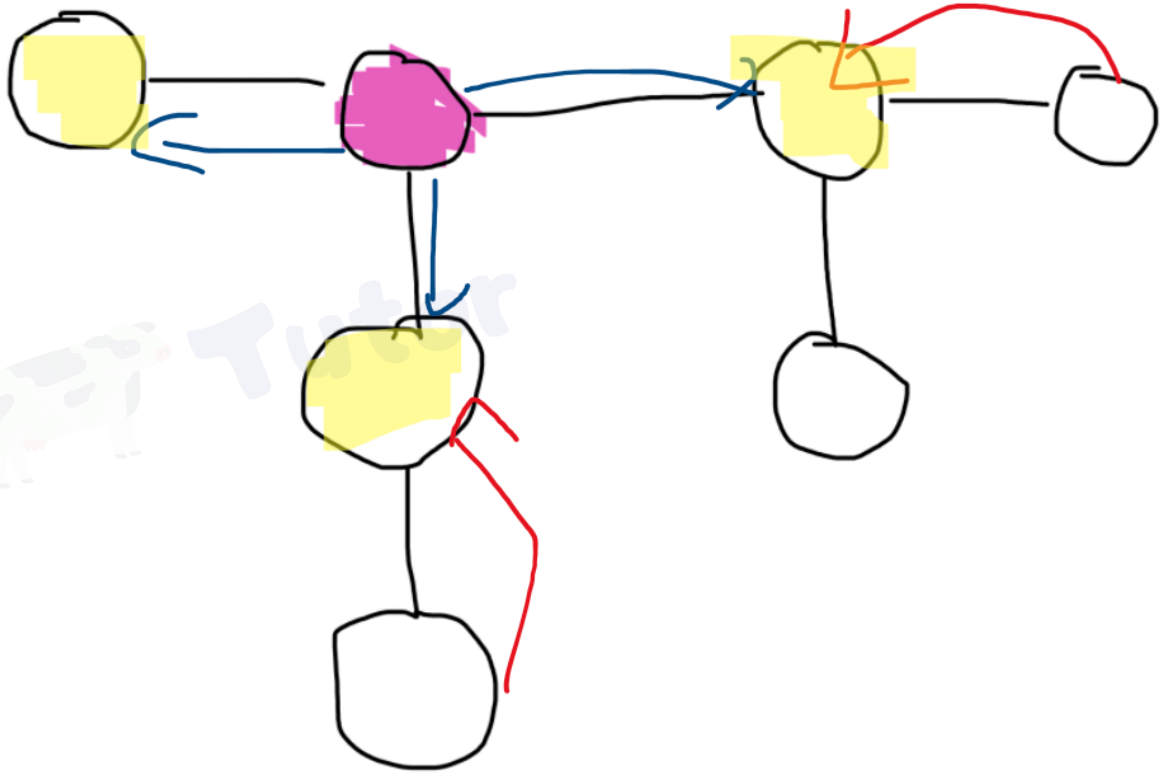


Our Own Case



Analysis of our case

- In our case, we can see that we need…
- 1 Farmer for Left. 1 Farmer for Right. 1 Farmer for Below.
- You can see that you save 1 farmers, because you can block off each highlighted exit with exactly 1 farmer.



Key Observations

Since this is a tree, Bessie will only ever go in 1 direction. IE: She will not go back

The answer is the sum of highlighted nodes, or blocked off exits.

We can block off an exit if the minimum distance from a leaf is smaller than the minimum distance from bessie

How to Code?

- 3 Parts.
- Find minimum distance for Bessie to get from node K to all other nodes.
- Find minimum distance from any node to a leaf
- Final DFS to get the solution.

DFS for Bessie

```
static void dfsBessie(int cur, int par){
        for(int next: adj[cur]){
            if(next==par) continue;
            bessieDist[next]= bessieDist[cur]+1;
            dfsBessie(next, cur);
        }
    }
```

DFS for Farmer

```
static void dfsFarmer(int cur, int par){
    for(int next: adj[cur]){
        if(next==par) continue;
        dfsFarmer(next, cur);
        farmerDist[cur]= Math.min(farmerDist[next]+1,
farmerDist[cur]);
    }
    if(adj[cur].size()==1) farmerDist[cur]=0;
  }
```

Final DFS

```
static void dfsSolve(int cur, int par){
        if(bessieDist[cur]>=farmerDist[cur]){
            res++; return;
        }
        for(int next: adj[cur]){
            if(next==par) continue;
            dfsSolve(next,  cur);
        }
    }
```