Dynamic Programming

SOCP'21
ANCC, IIT-DELHI

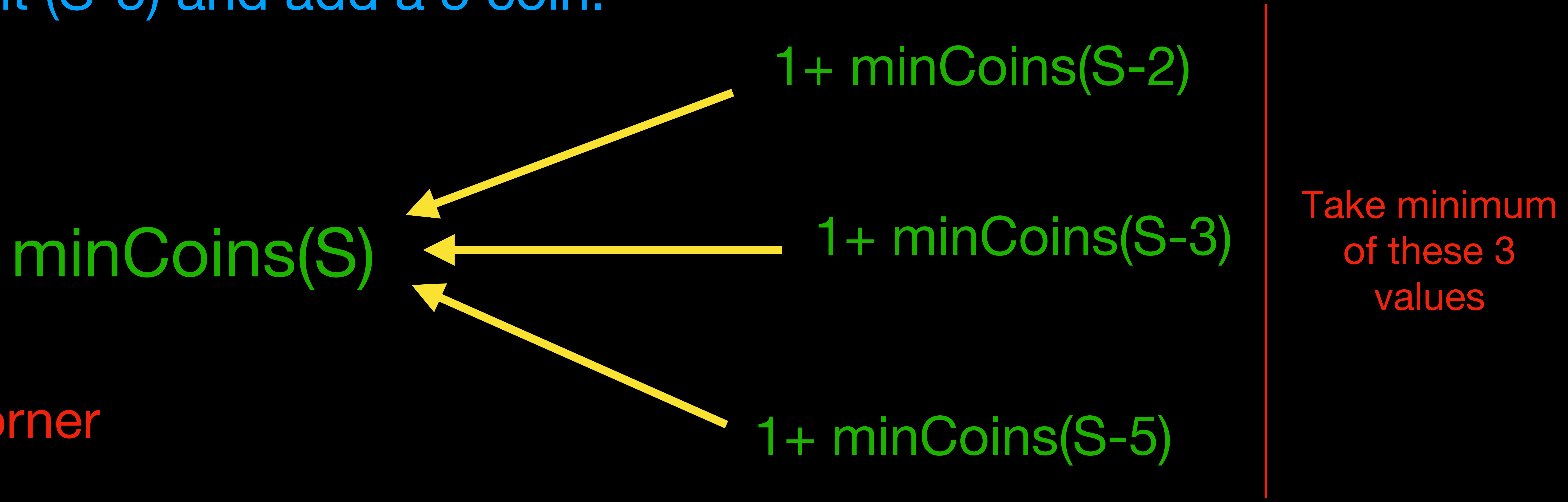
Coin Problem

- Given infinite number of coins of denominations 2, 3 and 5, find the minimum number coins needed to make an amount S.
- Example:
- S=14
- 14 = 2+2+2+2+2+2+2 (7 coins)
- 14 = 3+3+3+3+2 (5 coins)
- 14 = 5+5+2+2 (4 coins)
- It can be shown that we cannot do better than 4 coins.

Does naive greedy approach work?

Recursion to the Rescue

- Notice that we can create amount S in any 3 of the following ways-
- 1) Take amount (S-2) and add a 2 coin.
- 2) Take amount (S-3) and add a 3 coin.
- 2) Take amount (S-5) and add a 5 coin.



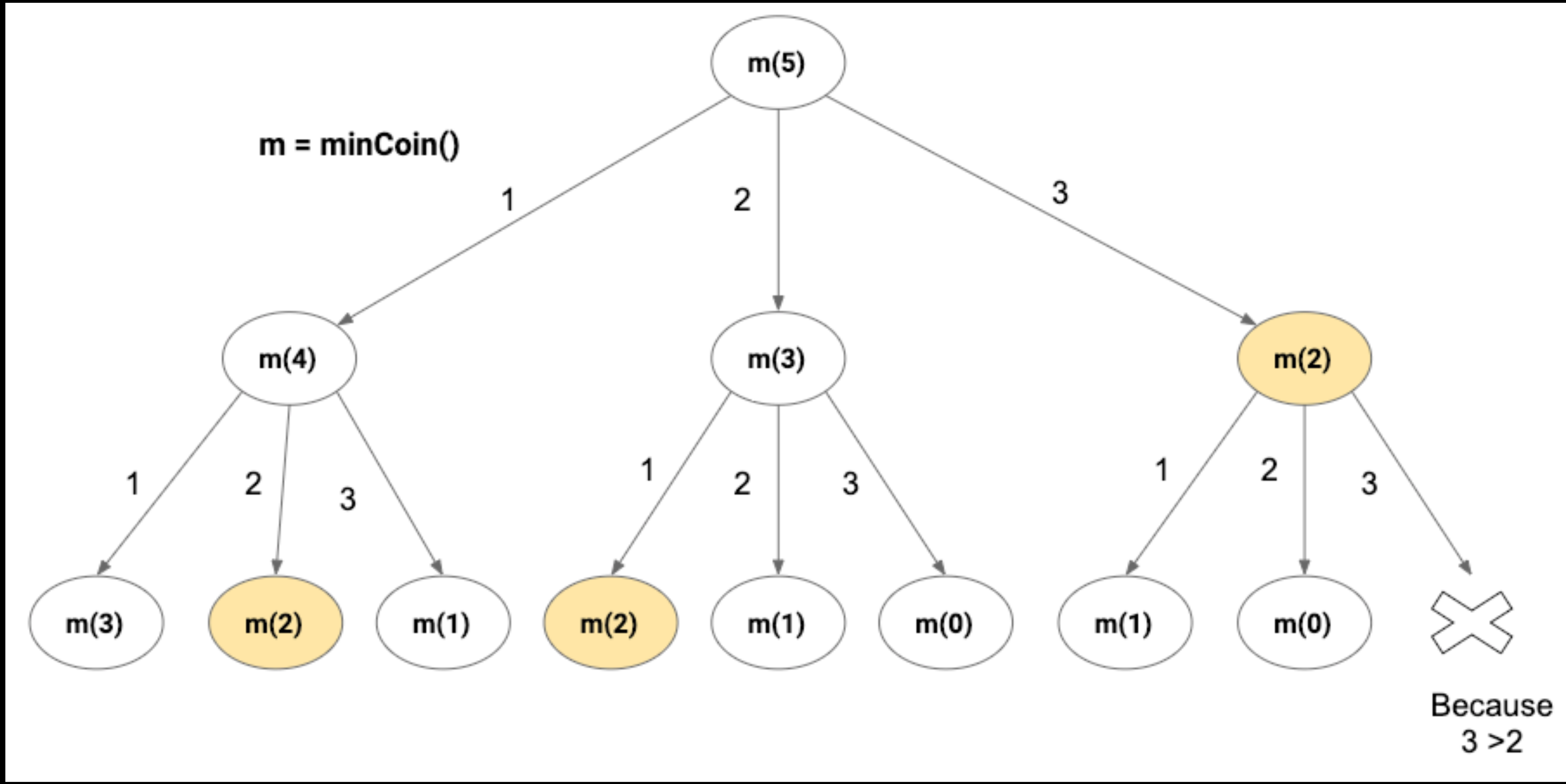
Beware of corner cases!

Recursion Code

```
int minCoins(int S) {

    if(S<0) {
        return 1e9;
    // For an amount of S<0, it is impossible to make it
    // with any number of coins, hence we return infinity
    }

    if(S==0) {
        return 0; //For an amount of S=0, we need 0 coins.
    }

    // 3 ways of reaching S
    int s1 = 1+minCoins(S-2);
    int s2 = 1+minCoins(S-3);
    int s3 = 1+minCoins(S-5);

    // we take the minimum of the three
    int res = min(s1,min(s2,s3));

    return res;
}
```

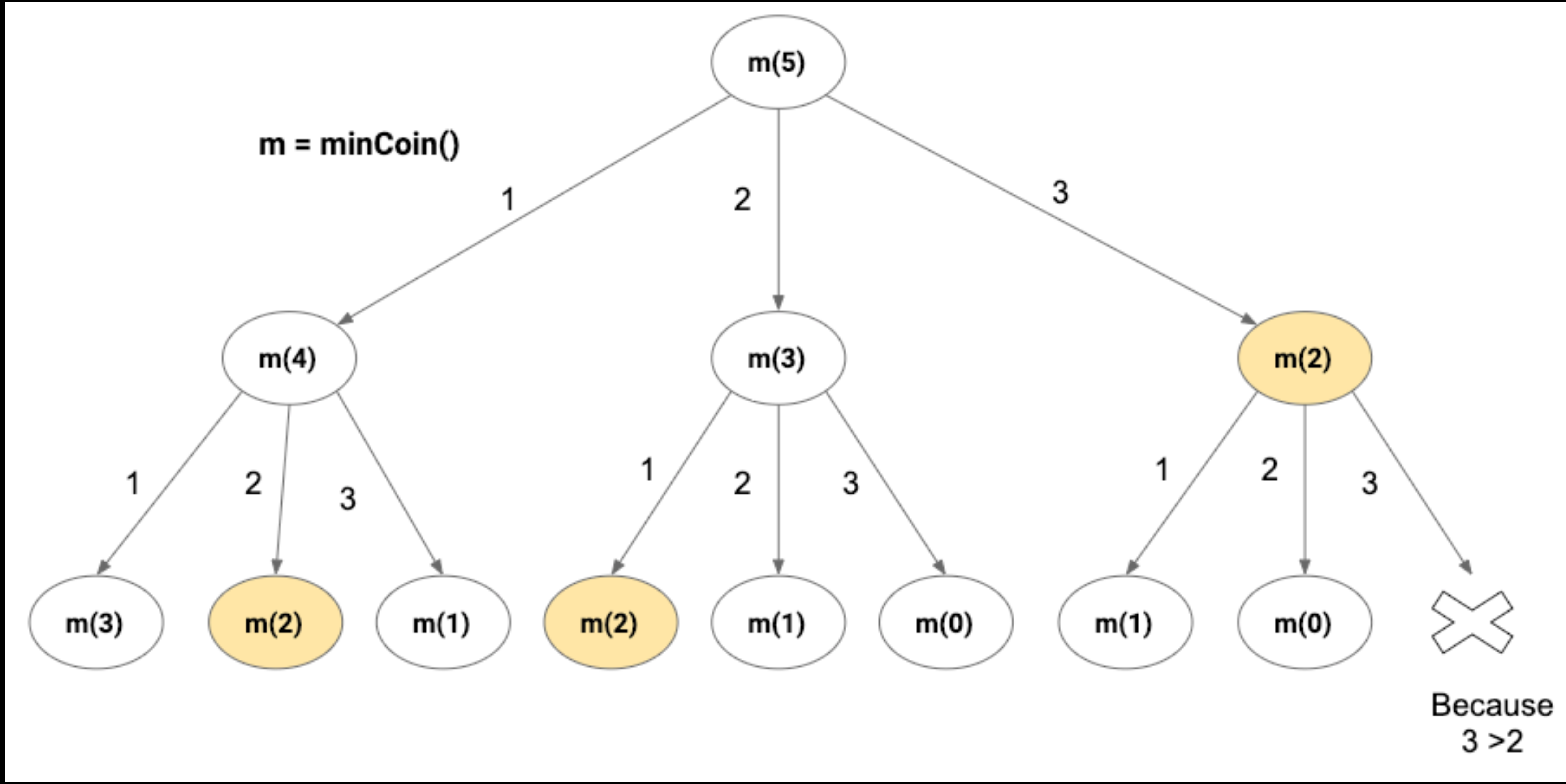
Time Complexity Analysis

- T(n) = T(n-2) + T(n-3) +T(n-5) + O(1) ===> **T(n) ~ O(3^n)** **Exponential !!**



We have ~n levels and there are ~3^n nodes at level n

Improving Time Complexity : Memoization



1. Notice that we are doing extra work when we are computing m(2), m(3) etc again and again.

2. Idea:
Somehow whenever we calculate minCoins(k) for the first time we should "remember" it so we can directly reuse that value instead of calling the recursive function again.

3. We can have an array called int memory[n] initialised fully with -1, where memory[k] will "remember" the value of minCoins(k).

4. If the value of memory[k] is not -1 that means we have already calculated minCoins(k) and we don't need to call the function.

Recursion + Memoization Code

```
int main()
{
    int S;
    S=39;
    memory = new int[S+1];

    for(int i=0;i<=S;i++) {
        memory[i]=-1;
    }

    cout<<minCoins(S);
    return 0;
}
```

Notice that time complexity is now O(N) because we spend O(1) time on each k thanks to memoization technique. Space complexity will also be O(N) for the memory[ ] array.

```
int *memory; //initialized as an array of size S+1
             //with all -1 entries

int minCoins(int S) {

    //base case
    if(S<0) {
        return 1e9;
    }
    if(S==0) {
        return 0;
    }

    //checking if we "remember" minCoins(S),
    //from some previous computation
    if(memory[S]!=-1){
        return memory[S];
    }


    int s1 = 1+minCoins(S-2);
    int s2 = 1+minCoins(S-3);
    int s3 = 1+minCoins(S-5);

    int res = min(s1,min(s2,s3));

    //before returning res, store it in the memory
    return memory[S]=res;
}
```

Thank you!