USA  Tutor

Fruit Feast

Analysis by David Yang

Statement Summary

You are given a maximum fullness of T<=5e6

You can eat 2 types of fruits:

- Orange gives you A<=T fullness
- Lemon gives you B<=T fullness

You can also drink water at most 1 time

- Water halves your fullness

Structure of the DP

I keep telling you there's an array, but what's in the array?

So the array is THE DP:

Array[state][state][state][state]… = random value

What's a state?

What's the value?

Recursive Idea

In any given time, we have 2 options

+A

+B

Or you can use up a special function

/2

Recursive Idea

```
int maxS;
recurse (int sum, int a, int b, bool water){
        if(sum>T) return;
        maxS = max(sum, maxS);
        if(water) recurse(sum/2, a,b, false);
        recurse(sum+a, a, b water);
        recurse(sum+b, a, b, water);
}
```

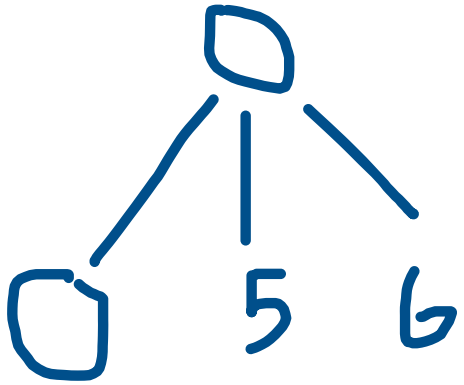
Sample Input

8 5 6

0

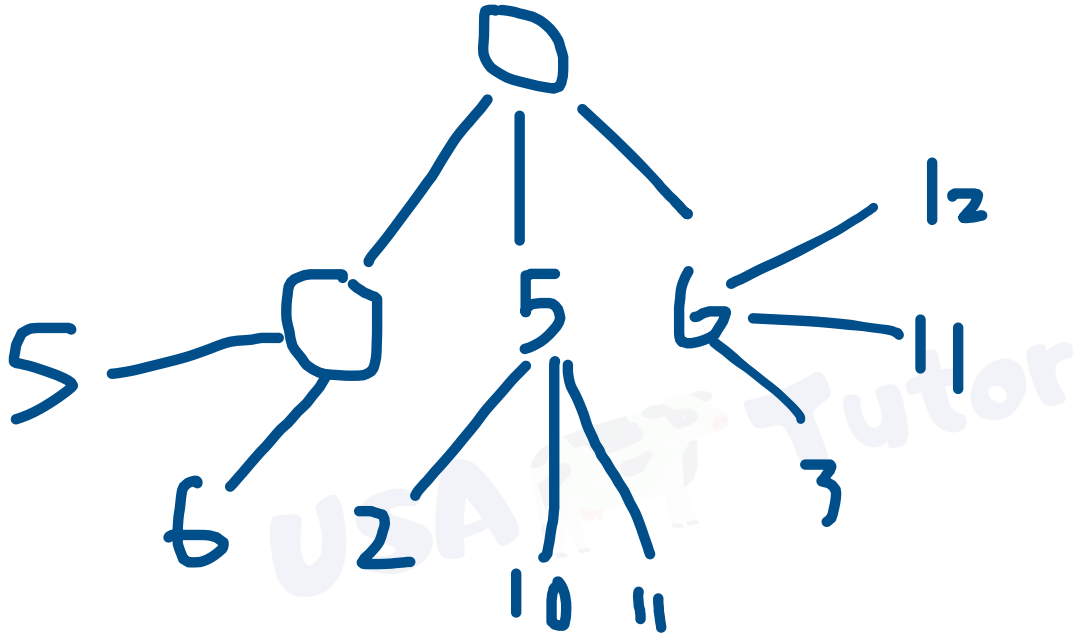
Sample Input

8 5 6



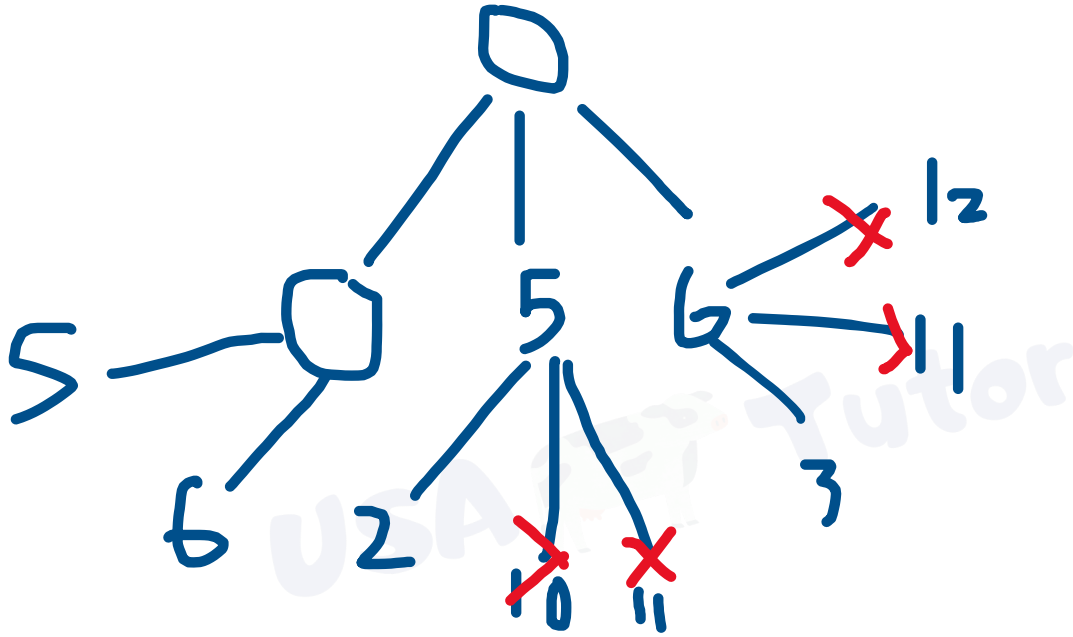
Sample Input

8 5 6

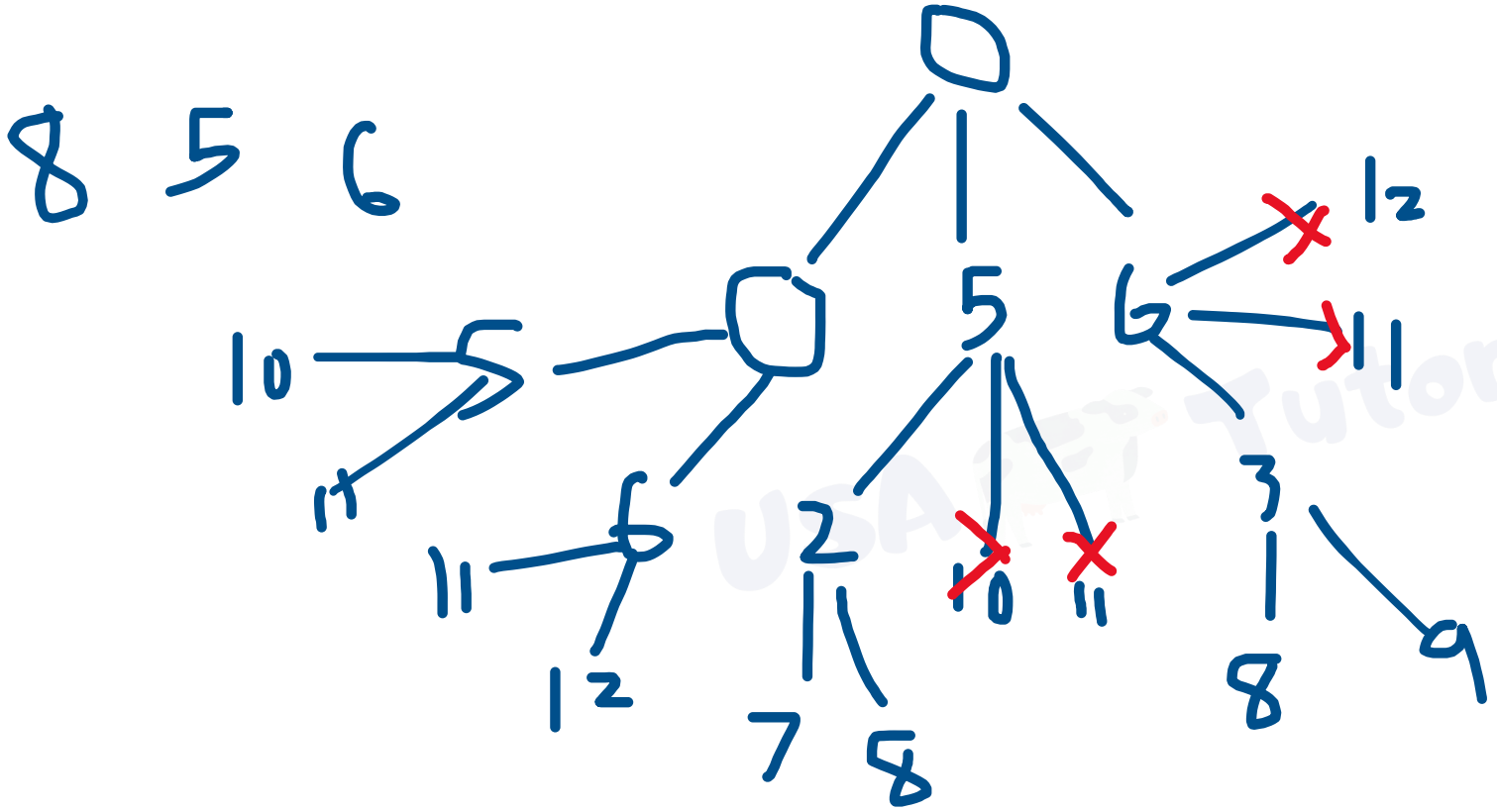


Sample Input

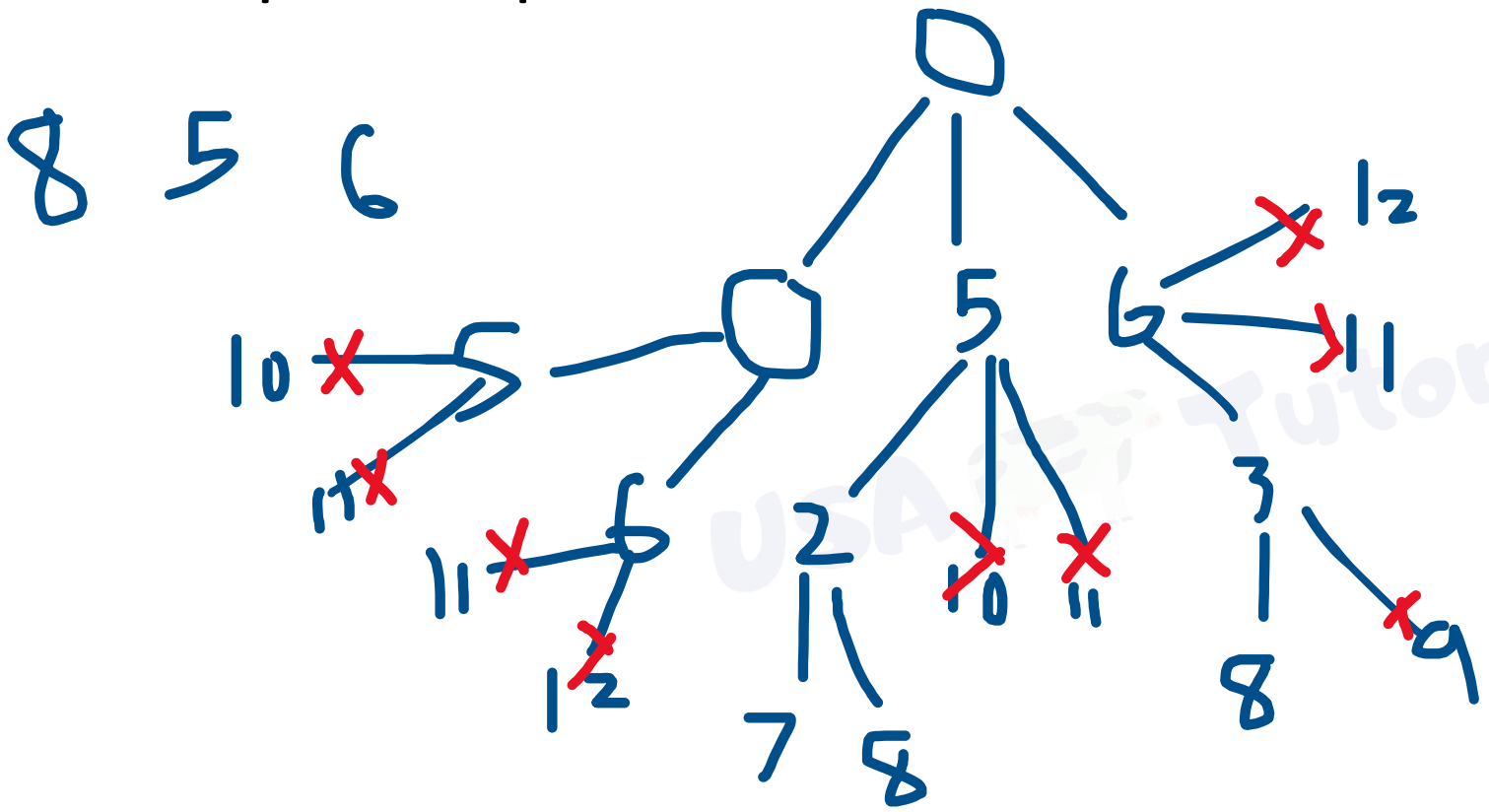
8 5 6



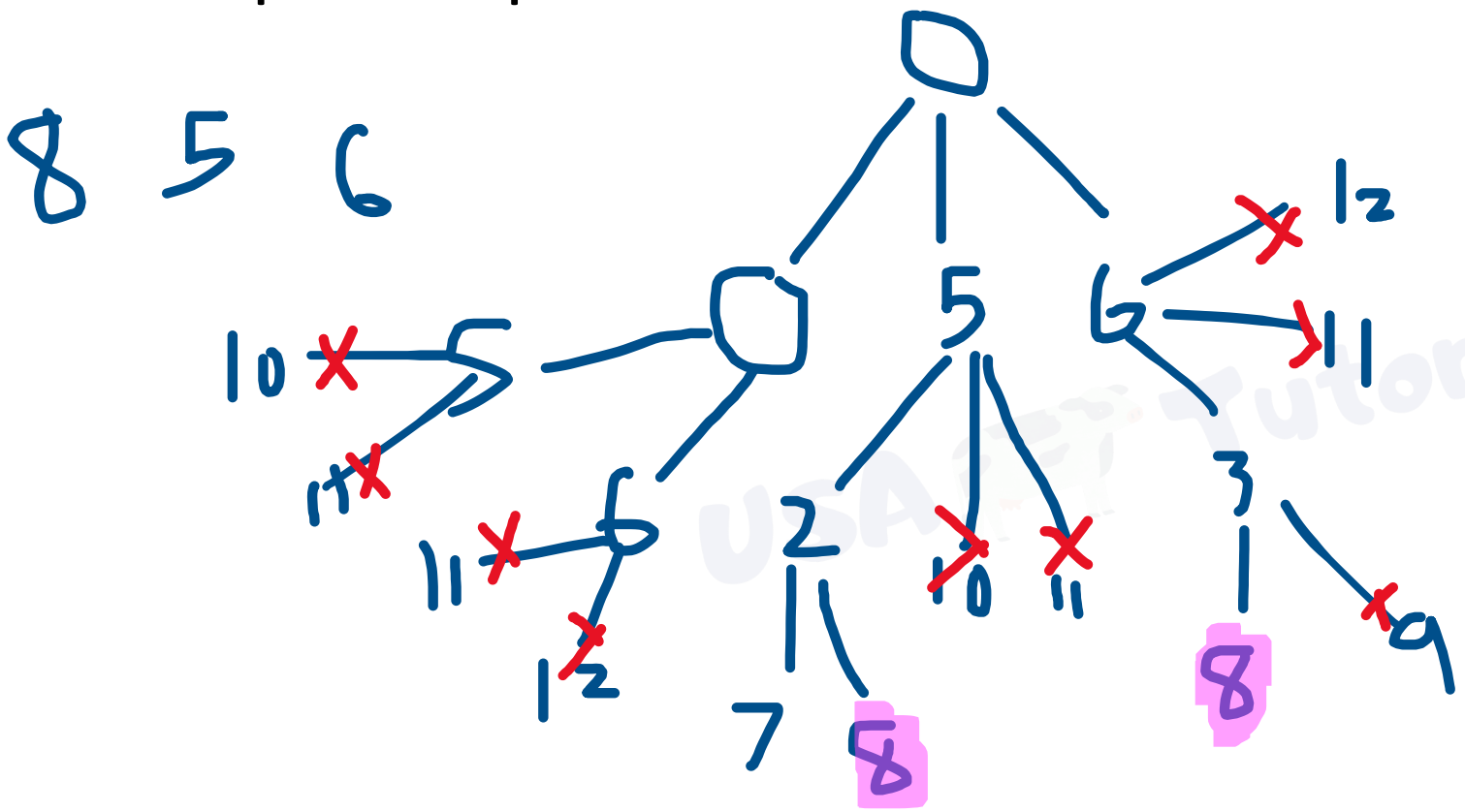
Sample Input



Sample Input



Sample Input



Technique

Similar to Milk Pails Silver (Feb 2016)

We have choices, and now we need an array

This is our DP

The DP

What are our States?

A state is a variable necessary to recreate an answer

(all paths lead to rome)

What are our transitions (where do we go next?)

This is pretty easy

All Possible States

A B = dp[A][B] = this seems wrong, A and B are constants

A T = dp[A][T] = wrong, A is a constant

A W (water) = wrong, A is a constant

All Possible States

B T = dp[B][T] = wrong, B is a constant

B W = dp[B][W]= wrong, B is a constant

T W = dp[T][W] = probably correct because everything else is wrong

T W = dp[current fullness][0/1 for water]

Given any current hungry level:

If you have drank water before, you must transition

Otherwise you can also try to drink water

Global

```
int dp[2][5000001];
```

Knapsack

```
int T, A, B;
cin>>T>>A>>B;
dp[0][0]=1;
for(int i=0; i<2; i++){
    for(int j=0; j<5000001; j++){
        if(dp[i][j]){
            if(j+A<=T) dp[i][j+A]=1;
            if(j+B<=T) dp[i][j+B]=1;
            if(i==0){
                dp[i+1][j/2]=1;
            }
        }
    }
}
```

Answer

```
int best=0;
for(int i=0; i<2; i++){
   for(int j=0; j<5000001; j++){
      if(dp[i][j] ){
         best = max(best, j);
      }
   }
}
```