Recursion

Key note:

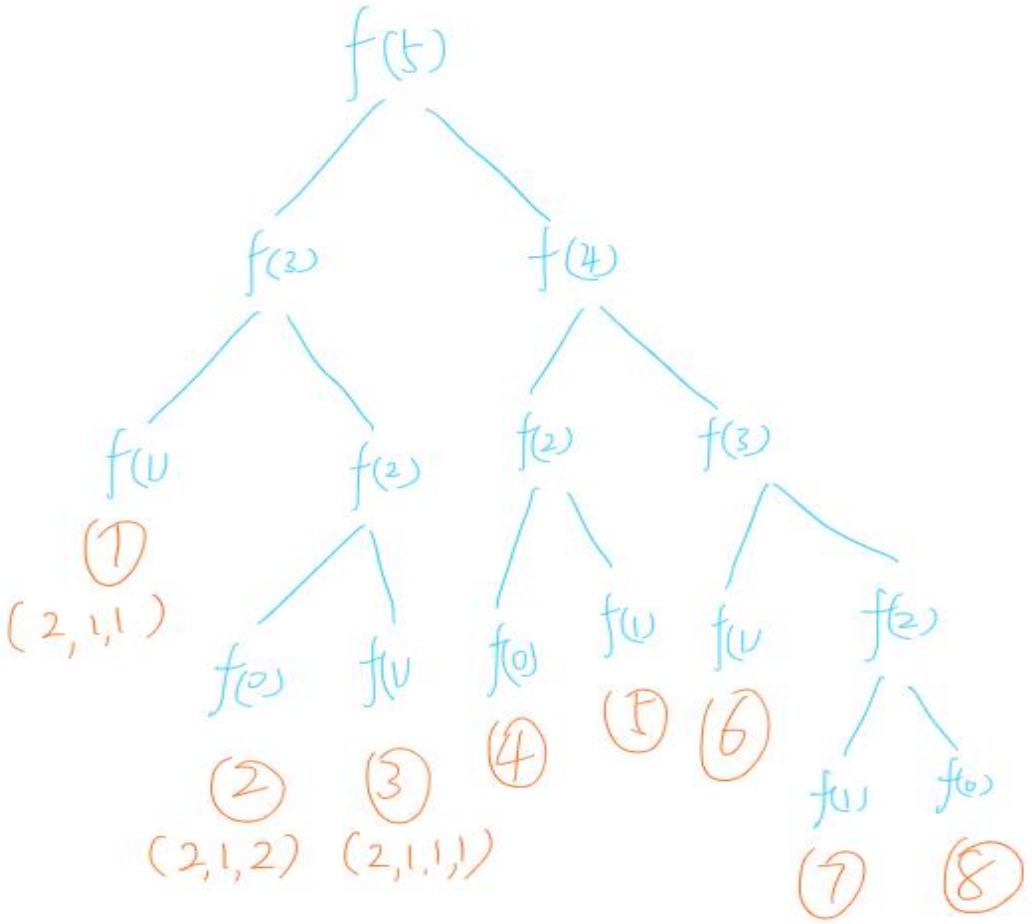
a) Find the relationship between current function and the next function.
b) Find the end condition.

Q1: If a frog can jump 1 step or 2 steps. How many possible ways the frog can jump X steps?

a) Relationship between each step: Assume we have n steps. Totally, there are f(n) possible ways to jump. Each time there are two possible: 1 step or 2 steps.
Case 1, if jump 1 step first, then we will have n-1 steps remaining which is f(n-1) possible ways.
Case 2, if jump 2 step first, then we will have n-2 steps remaining which is f(n-2) possible ways
Therefore: f(n) = f(n-1) +f(n-2)

b) End condition:
When n < 0. There are 0 possible ways to jump, so f(n) = 0 ***(Not count, since we limit n=1, so do not need to worry here.)
When n == 0. There are 0 possible ways to jump, so f(n) = 1 ***(This count, since it is a way to jump)
When n = 1. There are 1 possible way to jump, so f(n) = 1

```
public int  recusion(int a) {
   if(a <= 0){
      return 1;
   }
   else if(a == 1){
      return 1;
   }
   else{
      return recusion(a - 1) + recusion(a-2);
   }
}
```



Improvement: The problem here is too many repeated calculations. Expensive!

Here we can use **dynamic programming**

Key note:

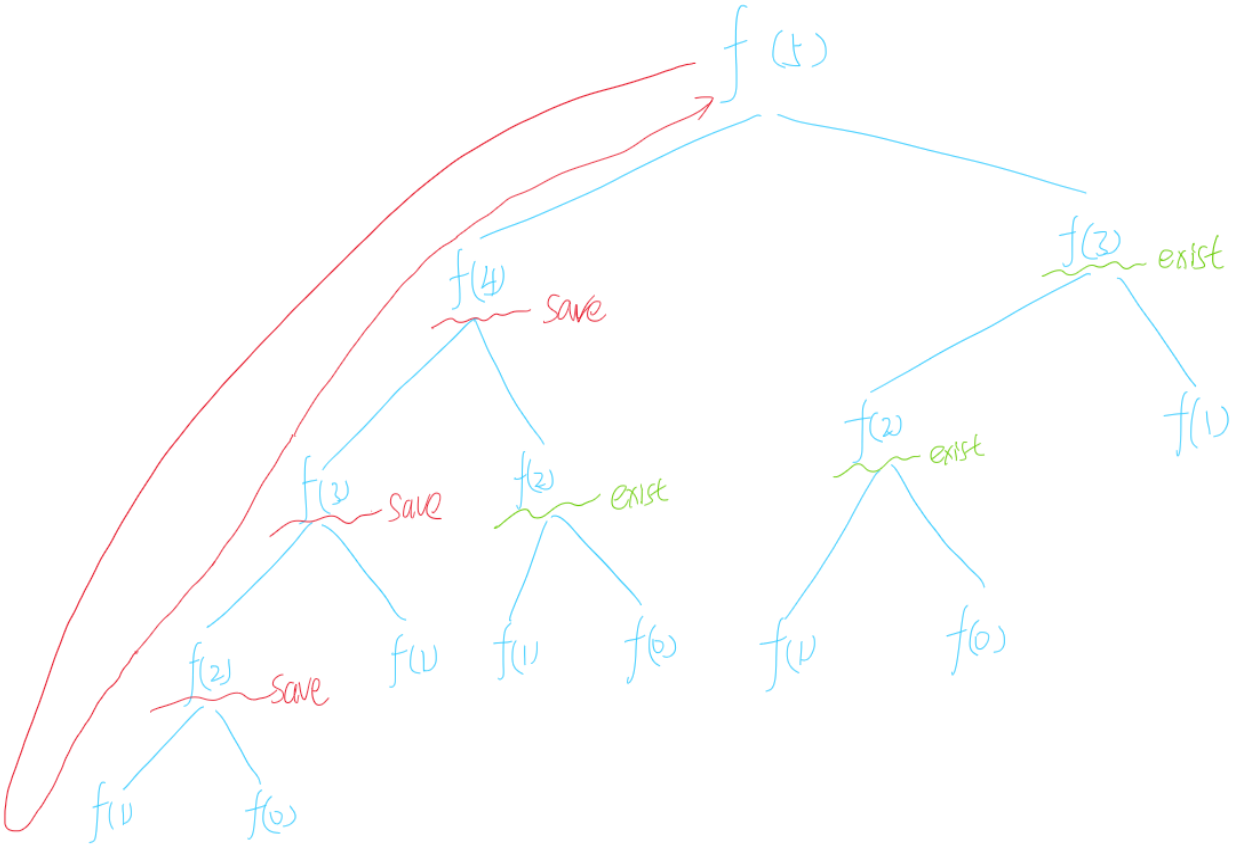
a) Save unique result in a map or other structure. Directly use without calculate again.

In this example, we can save f(4), f(3),f(2) when we first calculate. Therefore we can use anytime without do a repeat calculation.

```
   Map<Integer, Integer> map = new HashMap<>();
public int dynamic( int a ){
   if(a <= 0){
      return 1;
   }
   else if(a == 1){
      return 1;
   }
   else{
      if(map.containsKey(a)){
```

```
        return map.get(a);
      }
      else{
        int val = dynamic(a - 1) + dynamic(a-2);
        map.put(a,val);
        return val;
      }
    }

}
```



Q1: If a frog can jump 1 step, 2 steps, 3 steps ····· or X step each times. How many possible ways the frog can jump X steps?

```
Map<Integer, Integer> map = new HashMap<>();
public int dynamic( int a ){
  if(a <= 0){
    return 1;
  }
  else if(a == 1){
    return 1;
  }
```

```
  else{
    if(map.containsKey(a)){
      return map.get(a);
    }
    else{
      int val = 0;
      for(int i = 1 ;i<=a; i++){
        val += dynamic(a - i);
      }
      map.put(a,val);
      return val;
    }
  }

}
```