绪论

动态规划：记忆法

01-F1

圣人不记事，所以常记得；
令人忘事，以其记事。

有人建议不妨置备一本签名簿，供来访者留下自己的名字，就像怀特山那样；可是，天哪！我的记性非常好，用不着那个玩意儿

邓俊辉
deng@tsinghua.edu.cn

fib(): 递归

$$fib(n) = fib(n-1) + fib(n-2)$$

0 1 1 2 3 5 8 13 21 34 55 89 ...

```
int fib(n)  { return (2 > n) ? n : fib(n-1) + fib(n-2); }  //为何这么慢?
```

❖ **复杂度:** $T(0) = T(1) = 1,\ T(n) = T(n-1) + T(n-2) + 1,\ \forall\, n > 1$

- **若令** $S(n) = [T(n) + 1]/2$ **, 则** $S(0) = 1 = fib(1)$ **,** $S(1) = 1 = fib(2)$
- **故有** $S(n) = S(n-1) + S(n-2) = fib(n+1)$

  $T(n) = 2 \cdot S(n) - 1 = 2 \cdot fib(n+1) - 1 = \mathcal{O}(fib(n+1)) = \mathcal{O}(\phi^n)$
- **其中** $\phi = (1 + \sqrt{5})/2 \approx 1.618$

❖ **也可以...猜...** $T(n) = \mathcal{O}(\alpha^n)$

于是 $\mathcal{O}(\alpha^n) = \mathcal{O}(\alpha^{n-1}) + \mathcal{O}(\alpha^{n-2})$ **,** $\alpha^2 = \alpha^1 + \alpha^0$ **, 解得** $\alpha = (1 + \sqrt{5})/2 = \phi$

封底估算

$$\phi^{36} \approx 2^{25}$$

$$\phi^{43} = 2^{43*25/36} = 2^{1075/36} \approx 2^{1080/36} = 2^{30} \approx 10^{9}\ flo = 1\ sec$$

$$\phi^{5} \approx 10^{1}$$

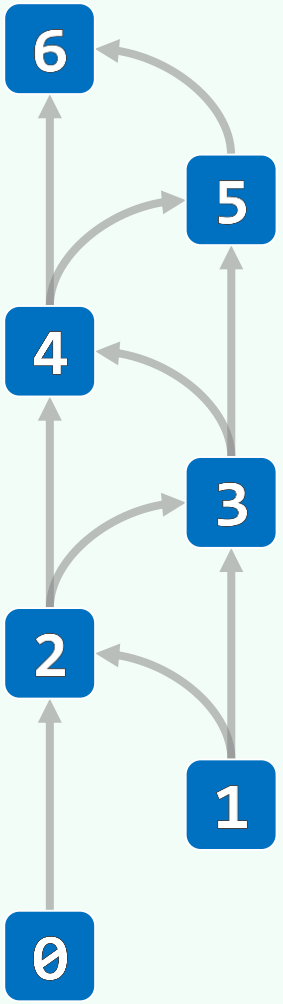
$$\phi^{67} \approx 10^{14}\ flo = 10^{5}\ sec \approx 1\ day$$

$$\phi^{92} \approx 10^{19}\ flo = 10^{10}\ sec \approx 10^{5}\ day \approx 3\ century$$

递归

❖ 低效的根源在于，有

大量重复的递归实例



❖ 先后出现的递归实例，共计 $\mathcal{O}(\phi^n)$ 个

而去除重复之后，总共不过 $\mathcal{O}(n)$ 种...

Memoization: 记住答案，直接“抄袭”

```
def f(n)
    if ( n < 1 ) return trivial( n );


    return f(n-X) + f(n-Y)*f(n-Z);
```

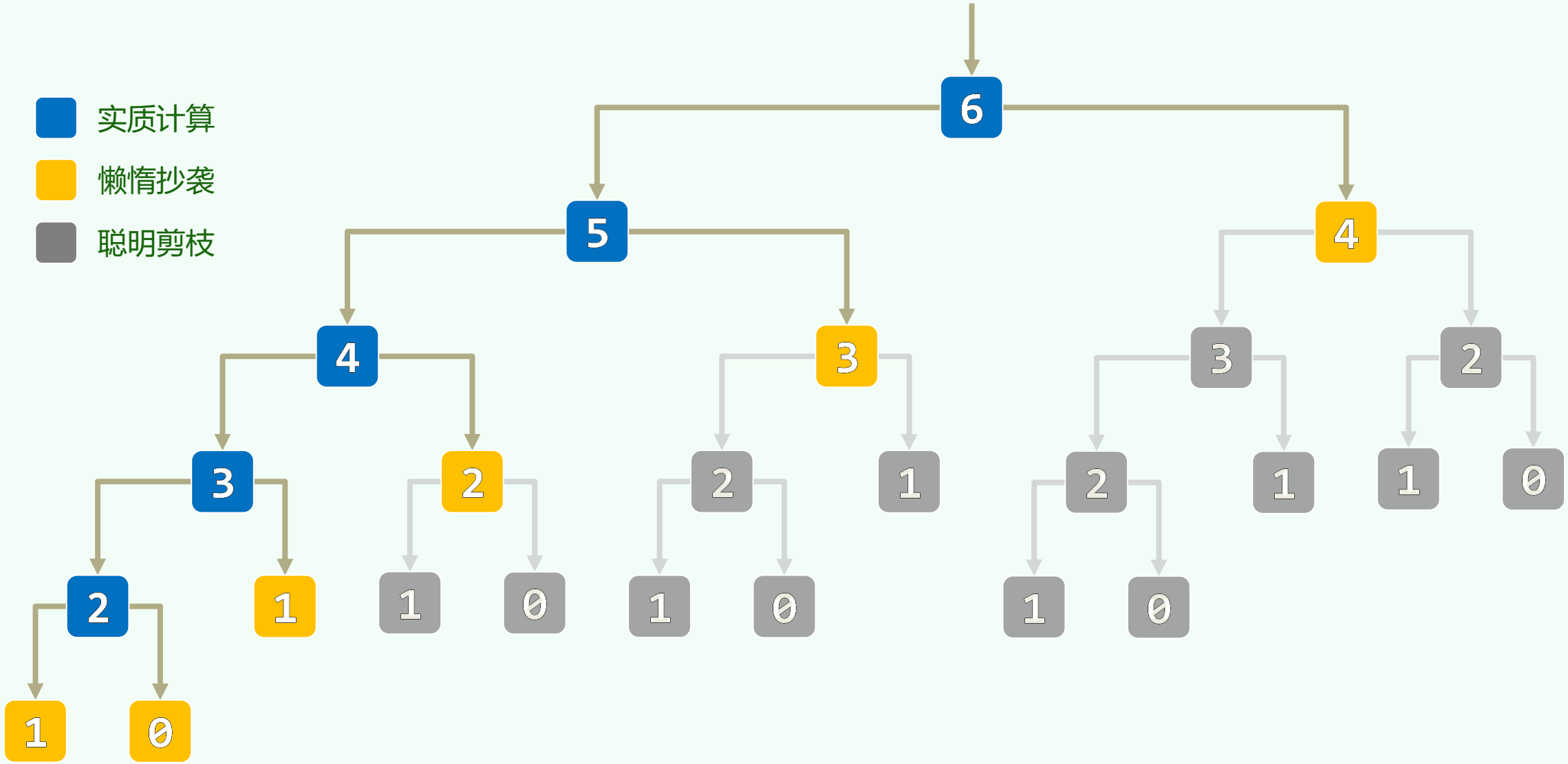
```
T M[ N ]; #init. with UNDEFINED

def f(n)
    if ( n < 1 ) return trivial( n );

 # recur only when necessary &
    always write down the result
    if ( M[n] == UNDEFINED )
        M[n] = f(n-X) + f(n-Y)*f(n-Z);

    return M[n];
```

Memoization: 实例: fib(6)



Dynamic Programming：颠倒计算方向

❖ 由自顶而下递归，改为自底而上迭代

❖
```
f = 1; g = 0; //fib(-1), fib(0)
while ( 0 < n-- ) {
   g = g + f;
   f = g - f;
}
return g;
```

❖ T(n) = O(n)，而且仅需O(1)空间！