算法设计与分析

第四章 动态规划

户保田
e-mail:hubaotian@hit.edu.cn

哈尔滨工业大学（深圳）计算机学院

本讲内容

4.1 动态规划的原理

4.2 矩阵乘法问题

4.3 最长公共子序列

4.4 背包问题

4.5 最优二分搜索树

斐波那契数列

第1个月：有1只新生母兔，还未成熟，共有1只母兔

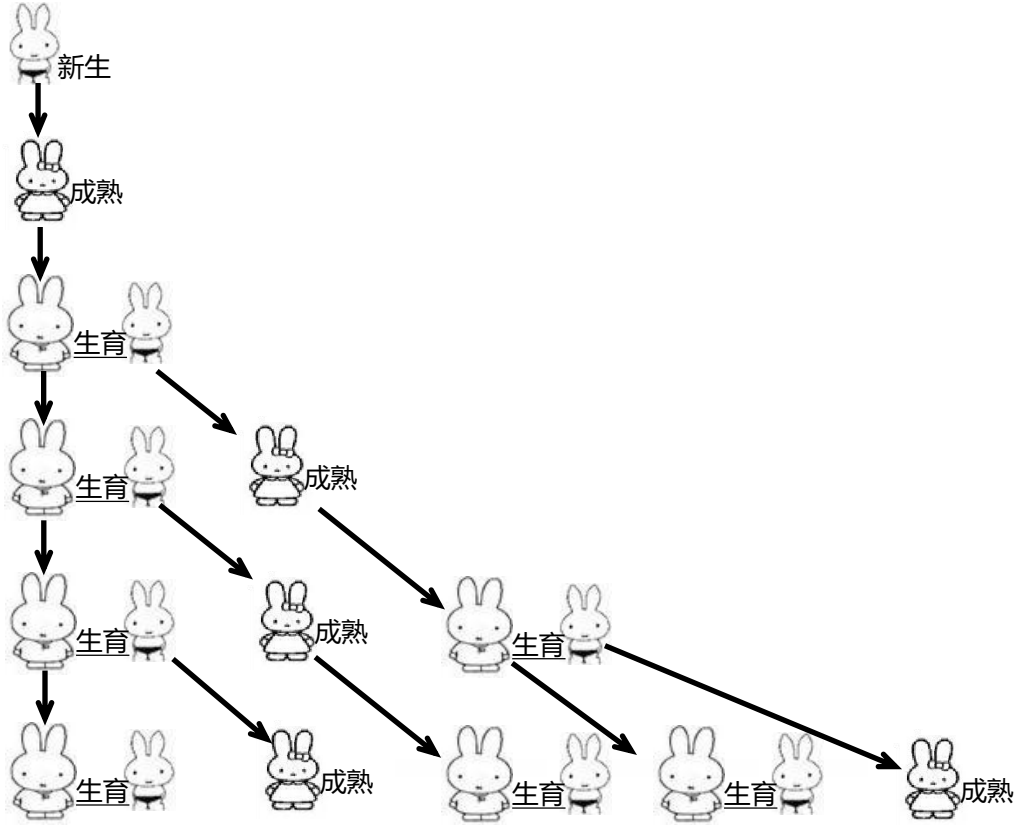
第2个月：无母兔生育，1只母兔成熟，共有1只母兔

第3个月：1只母兔生育，无新母兔成熟，共有2只母兔

第4个月：1只母兔生育，另1只母兔成熟，共有3只母兔

第5个月：2只母兔生育，另1只母兔成熟，共有5只母兔

第6个月：3只母兔生育，另2只母兔成熟，共有8只母兔



每个月母兔数的数列是：1, 1, 2, 3, 5, 8, 13, 21, 34, ...

$$F(n)=\begin{cases}1 & ,n=1\\ 1 & ,n=2\\ F(n-1)+F(n-2), & n>2\end{cases}$$

斐波那契数列(自顶向下)

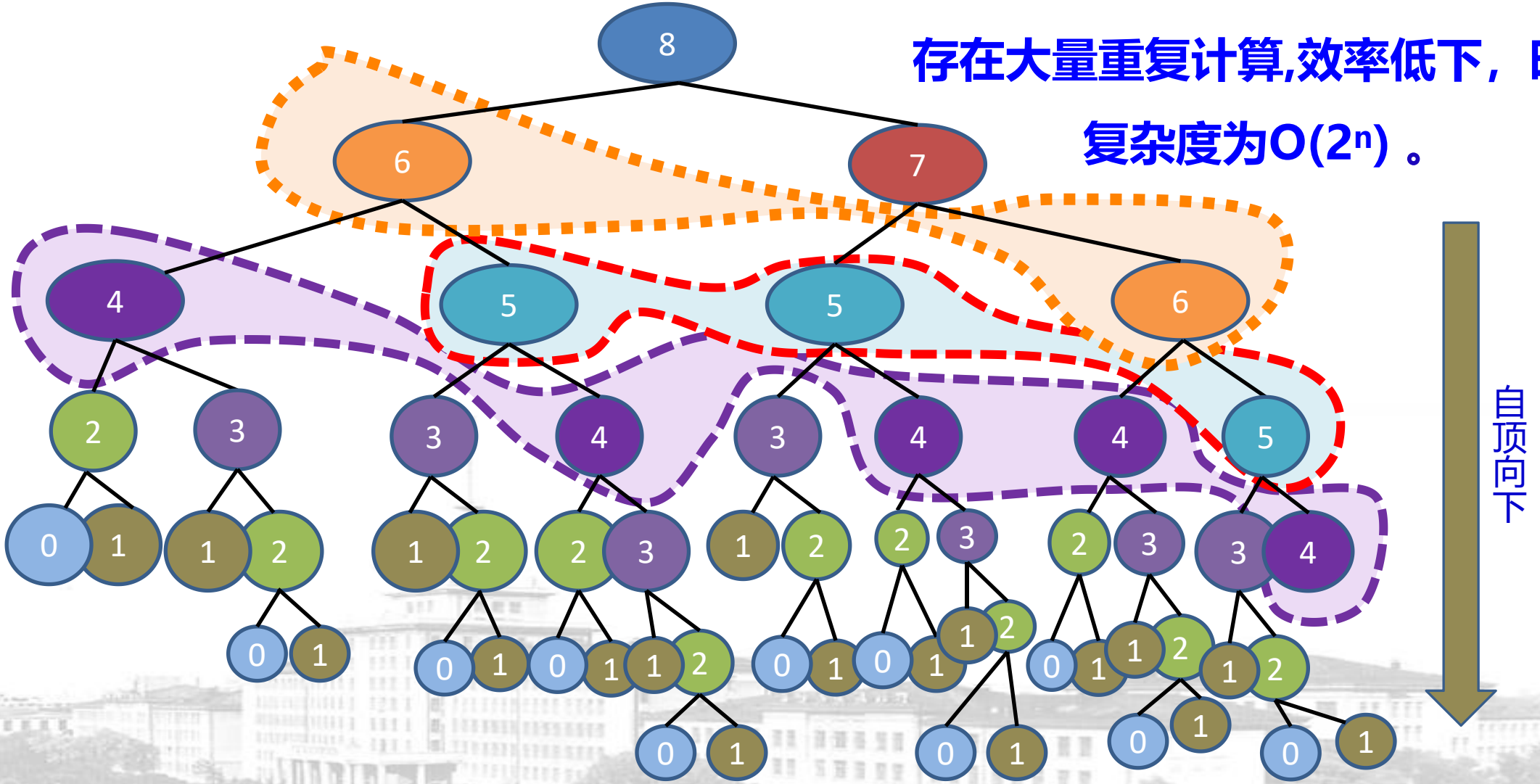
$$F(n)=\begin{cases}1 & ,n=1\\ 1 & ,n=2\\ F(n-1)+F(n-2), & n>2\end{cases}$$

如何设计算法，编写程序，给定任意的n求$F(n)$?

```
public int fib(int n)
    {
        if (n < 1)
        {
            return -1;
        }
        if (n == 1 || n == 2)
        {
            return 1;
        }
        return fib(n - 1) + fib(n - 2);
    }
```

斐波那契数列(自顶向下)

$$F(n)=\begin{cases}1 & ,n=1\\ 1 & ,n=2\\ F(n-1)+F(n-2), & n>2\end{cases}$$

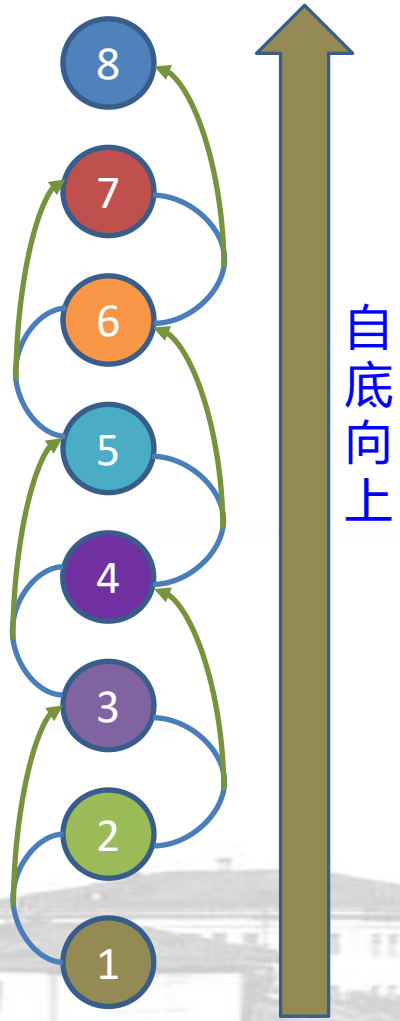


存在大量重复计算,效率低下，时间复杂度为$O(2^n)$。

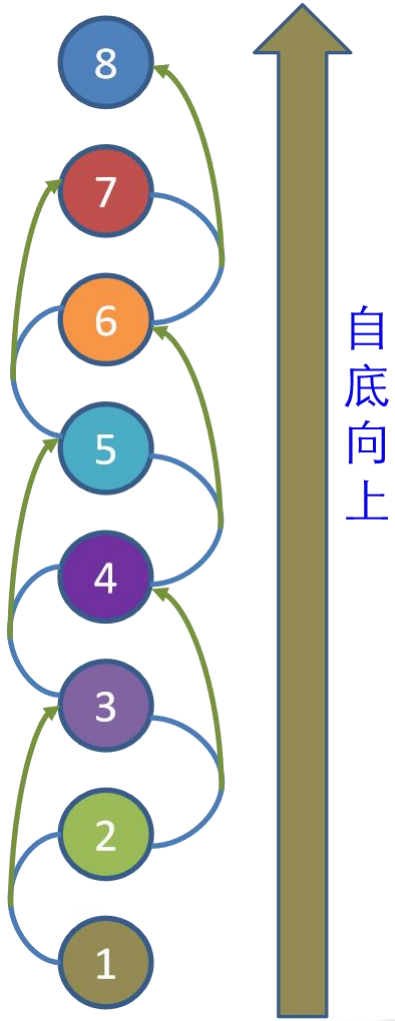
斐波那契数列(自底向上)

$$F(n)=\begin{cases}1 & ,n=1\\ 1 & ,n=2\\ F(n-1)+F(n-2), & n>2\end{cases}$$

- 首先计算最小的问题
  - 记录 F[0],F[1]的值
- 然后计算大一点的问题
  - 记录 F[2]的值
- …
- 继续从较小问题的解，计算大问题的解
  - 记录 F[n-1]的值
- 最后计算最终的问题.
  - 得到 F[n]的值



斐波那契数列(自底向上)



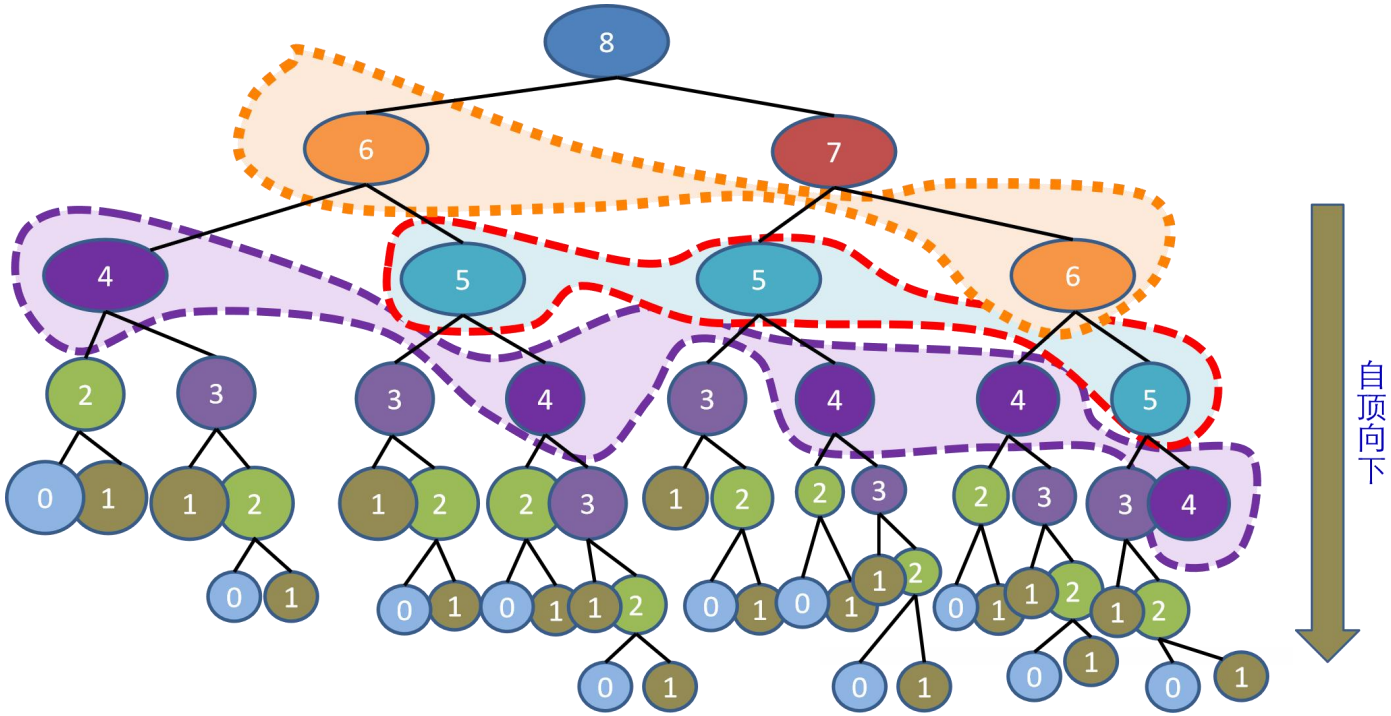

```
public int fib(int n)
    {
        fib_n_1 = 1,fib_n_2= 1, fib_n=0;
        if (n < 1)
        {
            return -1;
        }
        if (n == 1 || n == 2)
        {
            return 1;
        }
        for (i = 3; i <= n; ++i)
        {
            fib_n = fib_n_1 + fib_n_2;
            fib_n_2 = fib_n_1;
            fib_n_1 = fib_n;
        }
        return fib_n ;
    }
```

避免了大量重复计算,效率高，

时间复杂度为O(n) 。

Why?

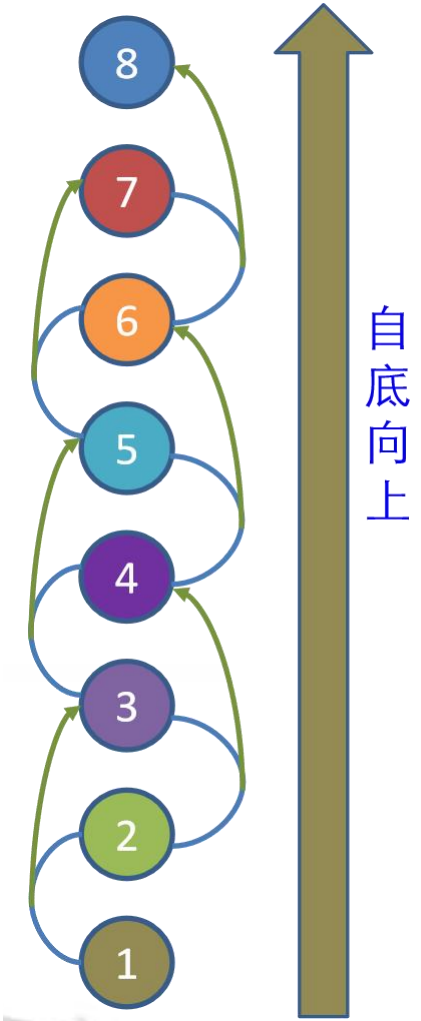
- 分治技术（**无记忆递归**）
  - 子问题是相互独立的
  - 分治方法将重复计算公共子问题，效率很低
  - 是一种自顶向下设计算法的思维



“那些遗忘过去的人注定要重蹈覆辙。” ——乔治·桑塔亚纳

What?

- 动态规划的特点（有记忆的迭代）
  - 把原始问题划分成一系列子问题
  - 求解每个子问题仅一次，将其结果保存在一个表中，以后用到时直接存取，不重复计算，节省计算时间
  - 自底向上地计算
- 适用范围
  - 优化问题：给定一组约束条件和代价函数，搜索具有最小或最大代价的优化解
  - 很多优化问题可分为多个子问题，子问题相互关联，且子问题的解可重复使用



“那些遗忘过去的人注定要重蹈覆辙。”——乔治·桑塔亚纳

How?

- 使用动态规划的条件
  - 优化子结构（正确性）
    - 当一个问题的优化解包含了子问题的优化解时，这个问题具有优化子结构
    - 缩小子问题集合，只需使用优化解中包含的子问题，降低时间复杂性
    - 优化子结构使得我们能自底而上地完成求解过程
  - 重叠子问题（必要性）
    - 在问题的求解过程中，很多子问题的解将被多次使用

How?

- 动态规划算法的设计步骤
  - 自顶向下分析优化解的结构
  - 根据优化解的结构递归地定义最优解的代价
  - 自底向上计算优化解的代价并保存，并获取构造最优解的信息
  - 根据构造最优解的信息构造优化解

动态规划起源

- 动态规划是一种算法设计的范式（paradigm）或思想，不是解决某一具体问题的具体算法
- 理查德·贝尔曼（Richard Bellman）在1950年代首次提出了这个名字，当时他正为美国兰德公司工作，主要为美国空军和政府项目服务。

> “It’s impossible to use the word, dynamic, in the pejorative sense…I thought dynamic programming was a good name. It was something not even a Congressman could object to.” ——*理查德·贝尔曼*

- 理查德·贝尔曼（1920年8月26日 - 1984年3月19日），美国应用数学家，美国国家科学院院士。