Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

动态规划探究

硕士研究生 程浩卿

2018年10月21日

内容提要

- 背景知识
  - 动态规划
  - 最优子结构
  - 重叠子问题
- 动态规划实例探究
  - House Robber
  - LIS
  - LCS
  - 01背包
- 动态规划与其他决策方法对比

背景知识

动态规划基本概念

- 动态规划
  - 指用来解决多阶段决策过程的最优化问题的一种方法，并非某一特定算法
  - 对于可用动态规划求解的问题，一般有两个特征—最优子结构与重叠子问题

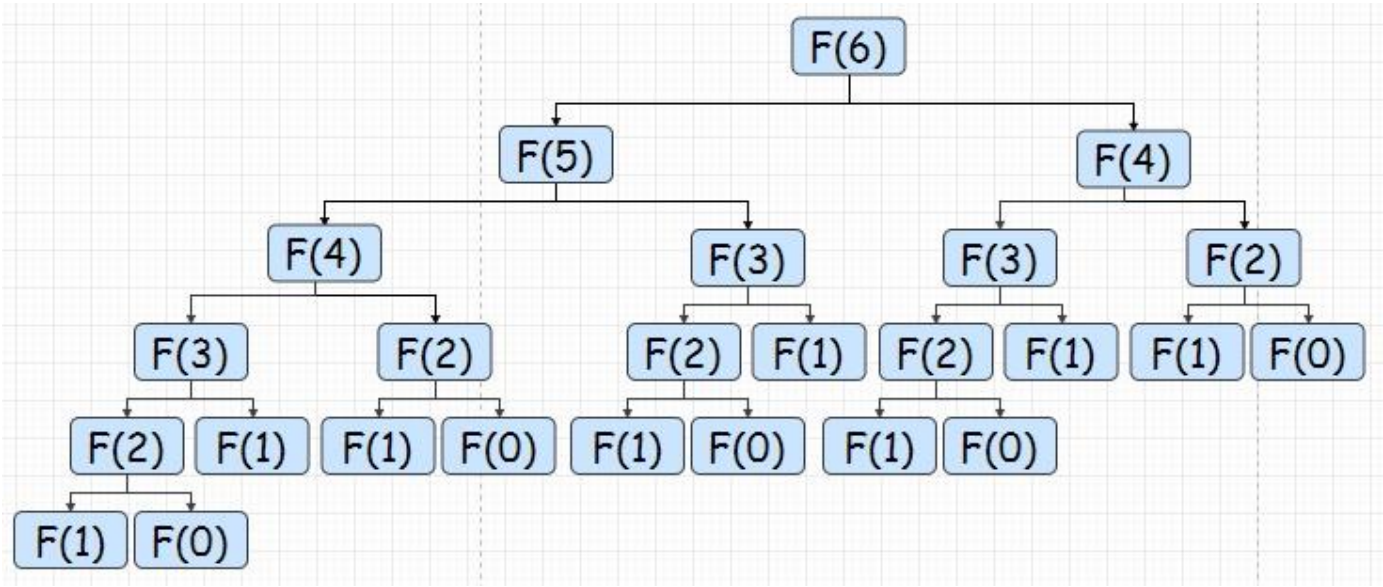
- 最优子结构
  - 一个问题的最优解包含其子问题的最优解

- 重叠子问题
  - 一个问题的递归算法会反复求解相同的子问题

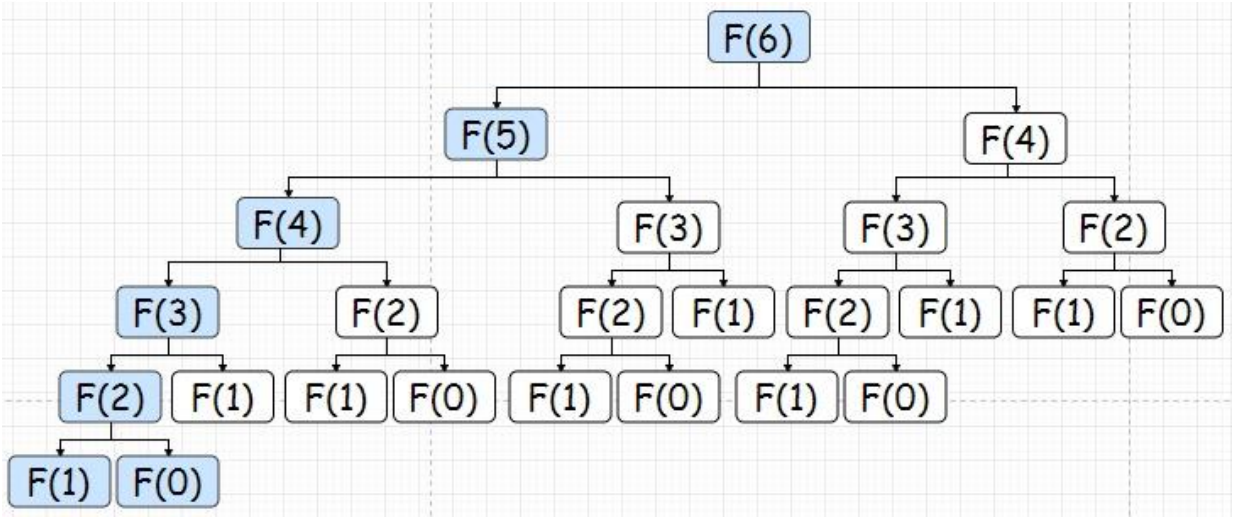
斐波那契数列

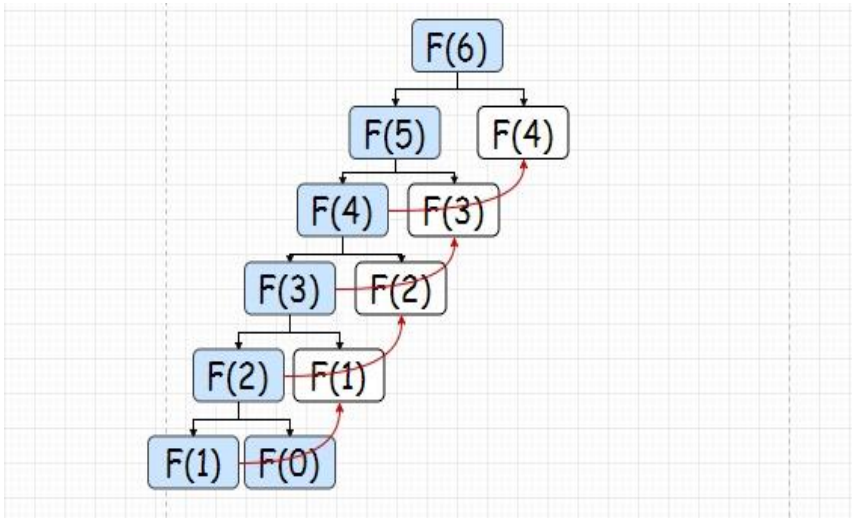
- **斐波那契递推公式**

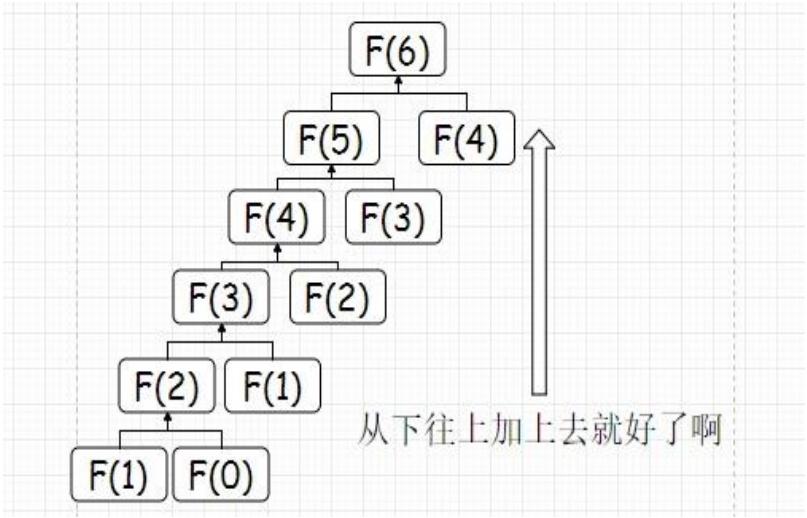
$$F(n) = \begin{cases} F(n-1) + F(n-2) & n \geq 2 \\ 1 & n = 0 \ or \ 1 \end{cases}$$



斐波那契数列


F(6)
F(5)
F(4)
F(4)
F(3)
F(3)
F(2)
F(3)
F(2)
F(2)
F(1)
F(2)
F(1)
F(1)
F(0)
F(2)
F(1)
F(1)
F(0)
F(1)
F(0)
F(1)
F(0)
F(1)
F(0)


F(6)
F(5)
F(4)
F(4)
F(3)
F(3)
F(2)
F(2)
F(1)
F(1)
F(0)


F(6)
F(5)
F(4)
F(4)
F(3)
F(3)
F(2)
F(2)
F(1)
F(1)
F(0)
从下往上加上去就好了啊

动态规划实例探究

House Robber

- 你是一个专业的小偷，计划偷窃沿街的房屋。每间房内都藏有一定的现金，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警

- 例如[1,3,2,5,4,3]，应当选择偷3-5-3得到最大收益11

House Robber

- 示例[1,3,2,5,4,3]

- 定义$F(i)$
  - 偷前i户人家所能获得的最大收益（最优状态）

- 该最优值与子问题最优值的递推关系（状态转移方程）
  - $F(i) = max\big(F(i-1), value[i] + F(i-2)\big)$

- 确定边界条件
  - $\begin{cases} F(0) = 0 \\ F(1) = value[1] \end{cases}$

House Robber

- 带备忘的自顶向下法
  - 按照递归编写程序，过程会保存已遇到子问题的最优值
- 自底向上法
  - 从规模最小的子问题开始，求解某问题当且仅当其相关子问题已经求解完成
- 变式问题
  - 街道为环形，即第1家与最后1家相邻
  - 街道呈树状排列

LIS

- 给定一个无序的非空整数数组，求出其中最长递增子序列的长度

- 例如[10,9,2,5,3,7,101,18]，最优值为4，其中一个最优解为[2,5,7,101]

LIS

- 示例[10,9,2,5,3,7,101,18]

- 定义$LIS(i)$
  - 以数组中第i个数结尾的LIS长度（条件最优状态）

- 该最优值与子问题最优值的递推关系（状态转移方程）
  - $LIS(i) = max(1 + LIS(j))$　　$j < i \ \ and \ \ arr[j] < arr[i]$

- 确定边界条件
  - $\begin{cases} LIS(0) = 0 \\ LIS(1) = arr[1] \end{cases}$

LCS

- **求两序列$x = [x_1, x_2, \cdots, x_m]$和$y = [y_1, y_2, \cdots, y_n]$的最长公共子序列的长度（DNA序列比对）**

- $x = [1,3,4,5,6,7,7,8]$
- $y = [3,5,7,4,8,6,7,8,2]$
- **最优值为5，其中一个最优解为**$[3,5,6,7,8]$

LCS

- 示例$x = [1,3,4,5,\quad 6,7,7,8], y = [3,5,7,4,8,\quad 6,7,8,2]$

- 定义$LCS(i,j)$
  - 考虑x 序列的前i 个元素，y 序列的前j 个元素时的LCS长度（最优状态）

- 该最优值与子问题最优值的递推关系（状态转移方程）
  - $$LIS(i,j) = \begin{cases} LIS(i-1,j-1)+1 & x[i] = y[j] \\ max\big(LIS(i-1,j), LIS(i,j-1)\big) & x[i] \neq y[j] \end{cases}$$

- 确定边界条件
  - $LIS(i,j) = 0 \quad i = 0 \ or \ j = 0$

LCS

下标j / 下标i		0	1	2	3	4	5	6	7	8	9
	$S2_j$	3	5	7	4	8	6	7	8	2	
0	$S1_i$	0	0	0	0	0	0	0	0	0	0
1	1	0									
2	3	0									
3	4	0									
4	5	0									
5	6	0									
6	7	0									
7	7	0									
8	8	0									

下标j / 下标i		0	1	2	3	4	5	6	7	8	9
	$S2_j$	3	5	7	4	8	6	7	8	2	
0	$S1_i$	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	3	0	**1**	**1**	**1**	1	1	1	1	1	1
3	4	0	1	1	1	**2**	2	2	2	2	2
4	5	0	1	2	2	**2**	**2**	2	2	2	2
5	6	0	1	2	2	2	2	**3**	3	3	3
6	7	0	1	2	3	3	3	**3**	4	4	4
7	7	0	1	2	3	3	3	3	**4**	4	4
8	8	0	1	2	3	3	4	4	4	**5**	5

01背包问题

- 有N件物品和一个容量为V的背包。第i件物品的体积是$c[i]$，价值是$w[i]$。求解将哪些物品装入背包可使价值总和最大

- $c = [1,3,2,6,2]$，$w = [2,5,3,10,4]$
- $N = 5$，$V = 12$
- 最优值为21，选择第1,2,4,5件物品

01背包问题

- $c = [1,3,2,6,2]$，$w = [2,5,3,10,4]$，$N = 5$，$V = 12$

- 定义$F(i,j)$
  - 考虑前i 个物品，背包容量为j 时的最大收益（最优状态）

- 该最优值与子问题最优值的递推关系（状态转移方程）
  - $$F(i,j) = \begin{cases} F(i-1,j) & j < c[i] \\ max(F(i-1,j), F(i-1,j-c[i]) + w[i]) \end{cases}$$

- 确定边界条件
  - $F(i,j) = 0 \quad i = 0 \ or \ j = 0$

01背包问题

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	2	2	2	2	2	2	2	2	2	2	2
2	0	2	2	5	7	7	7	7	7	7	7	7	7
3	0	2	3	5	7	8	10	10	10	10	10	10	10
4	0	2	3	5	7	8	10	12	13	15	17	18	20
5	0	2	4	6	7	9	11	12	14	16	17	19	21

- **变式问题**
  - 完全背包问题，每个物品数量不限
  - 多重背包问题，每个物品数量为一定值

穷举、贪心与动态规划

穷举、贪心与动态规划

- 穷举（搜索）：每个阶段的最优状态是由之前所有阶段的状态的组合得到的
- 贪心算法：每个阶段的最优状态都是由上一个阶段的最优状态得到的
- 动态规划：每个阶段的最优状态可以从之前某个阶段的某个或某些状态直接得到而不管之前这个状态是如何得到的

参考资料

[1] 算法导论15、16章.

[2] LeetCode.

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。