动态规划(上) - 滚动数组，划分，博弈与区间型
Sliding Array & Partition, Game, Interval

主讲 侯卫东

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

Overview

- 滚动数组
  - Minimum Path Sum
- 划分型动态规划
  - Decode Ways I/II
- 博弈动态规划
  - Coin in a line I/II/III
- 区间型动态规划
  - Burst Balloons

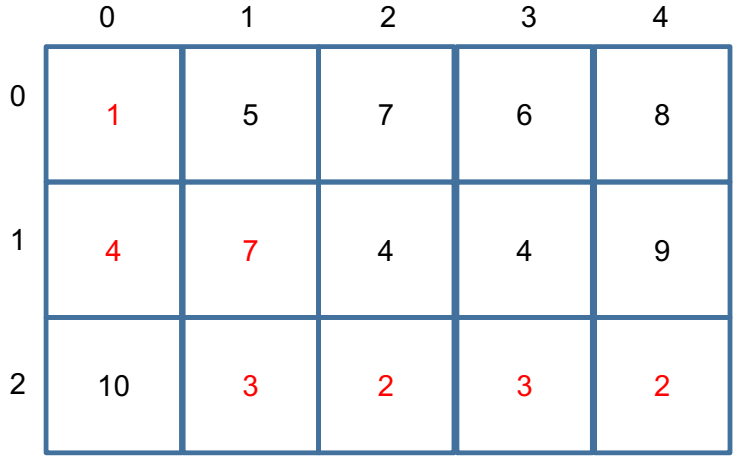
滚动数组 - DP空间优化神器

Minimum Path Sum

http://www.lintcode.com/problem/minimum-path-sum/
http://www.jiuzhang.com/solutions/minimum-path-sum/

LintCode 110

- 给定m行n列的网格，每个格子(i, j)里都一个非负数A[i][j]
- 求一个从左上角(0, 0)到右下角的路径，每一步只能向下或者向右走一步
- 使得路径上的格子里的数字之和最小
- 输出最小数字和



分析

- 确定状态：
  - 无论用何种方式到达右下角，总有最后一步：向右 或者 向下
  - 右下角坐标设为(m-1, n-1)，则前一步一定是在(m-2, n-1)或者(m-1, n-2)
  - 设从(0, 0)**走到(i, j)**的路径最小数字总和为f[i][j]
- 转移方程：f[i][j] = min{f[i-1][j], f[i][j-1]} + A[i][j]
- 初始条件和边界情况：
  - f[0][0] = A[0][0]
  - i = 0 或 j = 0，则前一步只能有一个方向过来
- 计算顺序：
  - f[0][0..n-1]
  - f[1][0..n-1]
  - …
- 时间复杂度：O(MN)　空间复杂度：O(MN)

动态规划四个组成部分

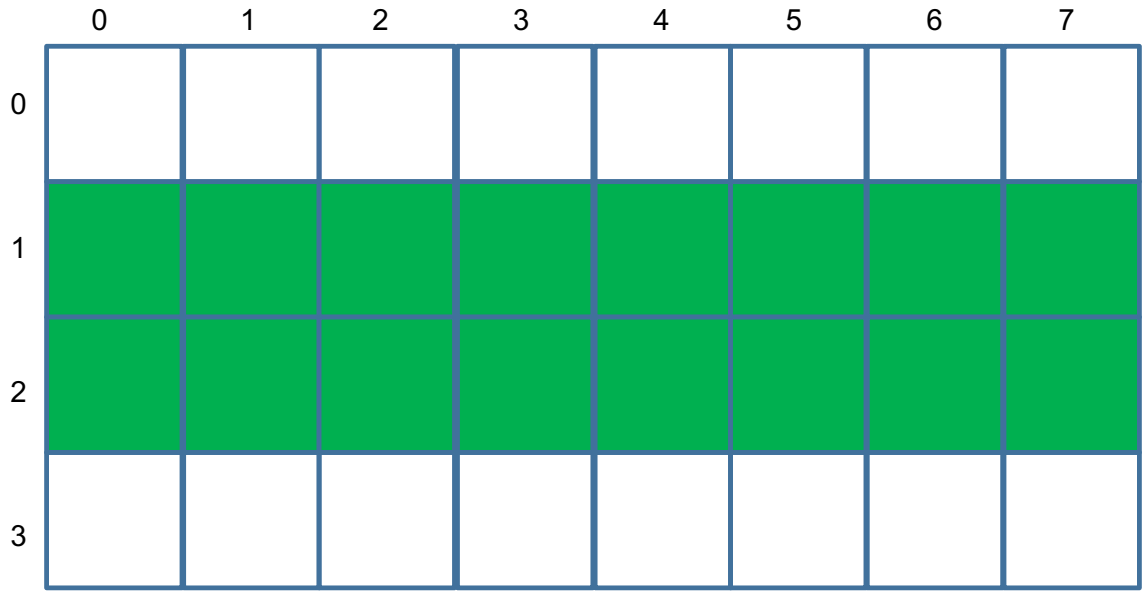
九章算法

四个组成部分

- 确定状态
  - 研究最优策略的最后一步
  - 化为子问题
- 转移方程
  - 根据子问题定义直接得到
- 初始条件和边界情况
  - 细心，考虑周全
- 计算顺序
  - 利用之前的计算结果

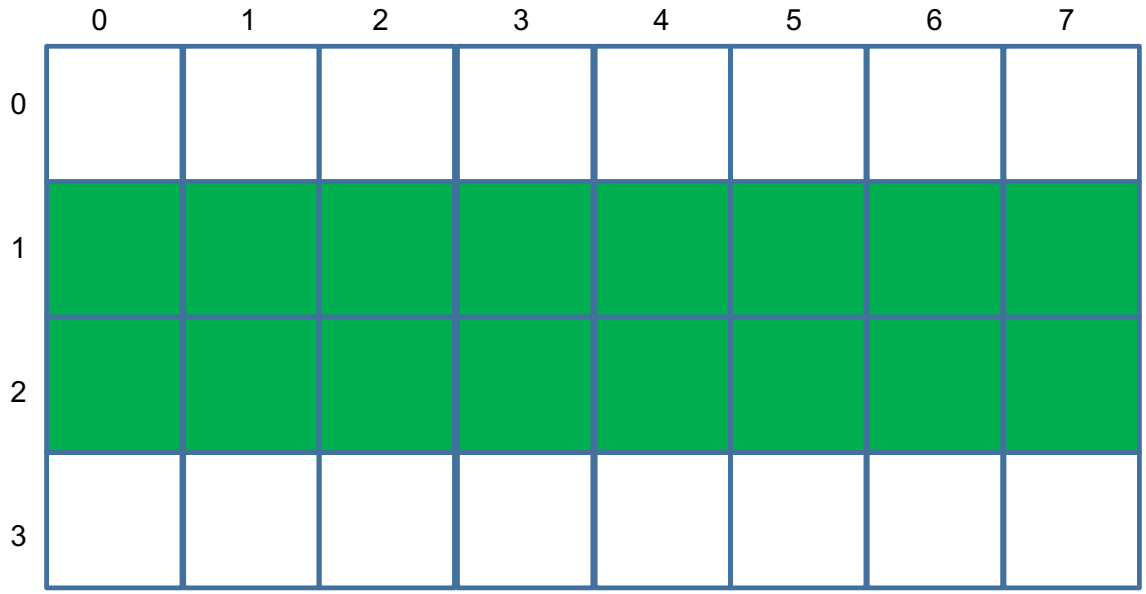
空间优化

- f[i][j] = min{f[i-1][j], f[i][j-1]} + A[i][j]
- 计算第i行时，只需要第i行和第i-1行的f



空间优化

- 所以，只需要保存两行的f值：f[i][0..n-1]和f[i-1][0..n-1]
- 用滚动数组实现



编程

滚动数组相关问题

Unique Paths
https://www.lintcode.com/problem/unique-paths/

Fibonacci
https://www.lintcode.com/problem/fibonacci/

划分型动态规划

九章算法

- 常见动态规划类型

- 给定长度为N的序列或字符串，要求划分成若干段
  - 段数不限，或指定K段
  - 每一段满足一定的性质

划分型动态规划

九章算法

- 常见动态规划类型

- 给定长度为N的序列或字符串，要求划分成若干段
  - 段数不限，或指定K段
  - 每一段满足一定的性质

Decode Ways

http://www.lintcode.com/problem/decode-ways/
http://www.jiuzhang.com/solutions/decode-ways/

LintCode 512

九章算法

- 题意：有一段由A-Z组成的字母串信息被加密成数字串
- 加密方式为：A→1, B→2, …, Z→26
- 给定加密后的数字串S[0…N-1]，问有多少种方式解密成字母串

- 例子：
- 输入：12
- 输出：2 (AB 或者 L)

分析

- 确定状态：
  - 最后一步：一定有最后一个字母, A, B, …, 或Z
  - 这个字母加密时变成1, 2, …, 或26
  - 需要知道数字串前N-1和N-2个字符的解密方式数
- 转移方程：f[i] = f[i-1] | S[i-1]对应一个字母 + f[i-2] | S[i-2]S[i-1]对应一个字母
- 初始条件和边界情况：
  - f[0] = 1，即空串有1种方式解密
  - 如果i = 1, 只看最后一个数字
- 计算顺序：
  - f[0], f[1], …, f[n]
- 时间复杂度O(N), 空间复杂度O(N)

Decode Ways II

http://www.lintcode.com/en/problem/decode-ways-ii/
http://www.jiuzhang.com/solution/decode-ways-ii/

LintCode 676

九章算法

- 题意：有一段由A-Z组成的字母串信息被加密成数字串
- 加密方式为：A→1, B→2, …, Z→26
- 给定加密后的数字串S[0…N-1]，问有多少种方式解密成字母串
- 其中可能出现*字符，可以被替换成为1~9中的任何一个字符

- 例子：
- 输入：1*
- 输出：18 (11~19各有两种方式)

分析

- 确定状态：
  - 和Decode Ways基本相同
  - 需要知道数字串前N-1和N-2个字符的解密方式数

情况一：最后一个字符翻译成字母

- S[i-1] =‘0’ ：**不能翻译成字母**
- S[i-1] ∈{‘1’, …, ‘9’} ：**1种方式翻译成一个字母**，共f[i-1]种方式
- S[i-1] =‘*’ ：**9种可能翻译成一个字母**，共9*f[i-1]种方式

题目分析

九章算法

情况二：最后两个字符翻译成字母

- S[i-2] = ‘0’ ：不能翻译成字母
- S[i-2] = ‘1’
  - S[i-1] ∈{‘0’, …, ‘9’} ，**1种可能翻译成一个字母**，共f[i-2]种方式
  - S[i-1] = ‘*’ ， **9种可能翻译成一个字母**，共9*f[i-2]种方式
- S[i-2] = ‘2’
  - S[i-1] ∈{‘0’, …, ‘6’} ，**1种可能翻译成一个字母**，共f[i-2]种方式
  - S[i-1] ∈{‘7’, …, ‘9’} ，**不能翻译成字母**
  - S[i-1] = ‘*’ ， **6种可能翻译成一个字母**，共6*f[i-2]种方式
- S[i-2] ∈{‘3’, …, ‘9’} ：**不能翻译成字母**
- S[i-2] = ‘*’
  - S[i-1] ∈{‘0’, …, ‘6’} ，**2种可能翻译成一个字母**，共2*f[i-2]种方式
  - S[i-1] ∈{‘7’, …, ‘9’} ，**1种可能翻译成一个字母**，共f[i-2]种方式
  - S[i-1] =‘*’ ，**15种可能翻译成一个字母**，共15*f[i-2]种方式

编程

课间休息五分钟

博弈类动态规划

Game DP

博弈型动态规划

九章算法

- 博弈为两方游戏
- 一方先下，在一定规则下依次出招
- 如果满足一定条件，则一方胜
- 目标：取胜

博弈

九章算法

- 先手：先出招的一方
- 出招后，先手换人，新的先手面对一个新的局面

Coins in a line

https://www.lintcode.com/problem/coins-in-a-line/
https://www.jiuzhang.com/solutions/coins-in-a-line/

LintCode 394

九章算法

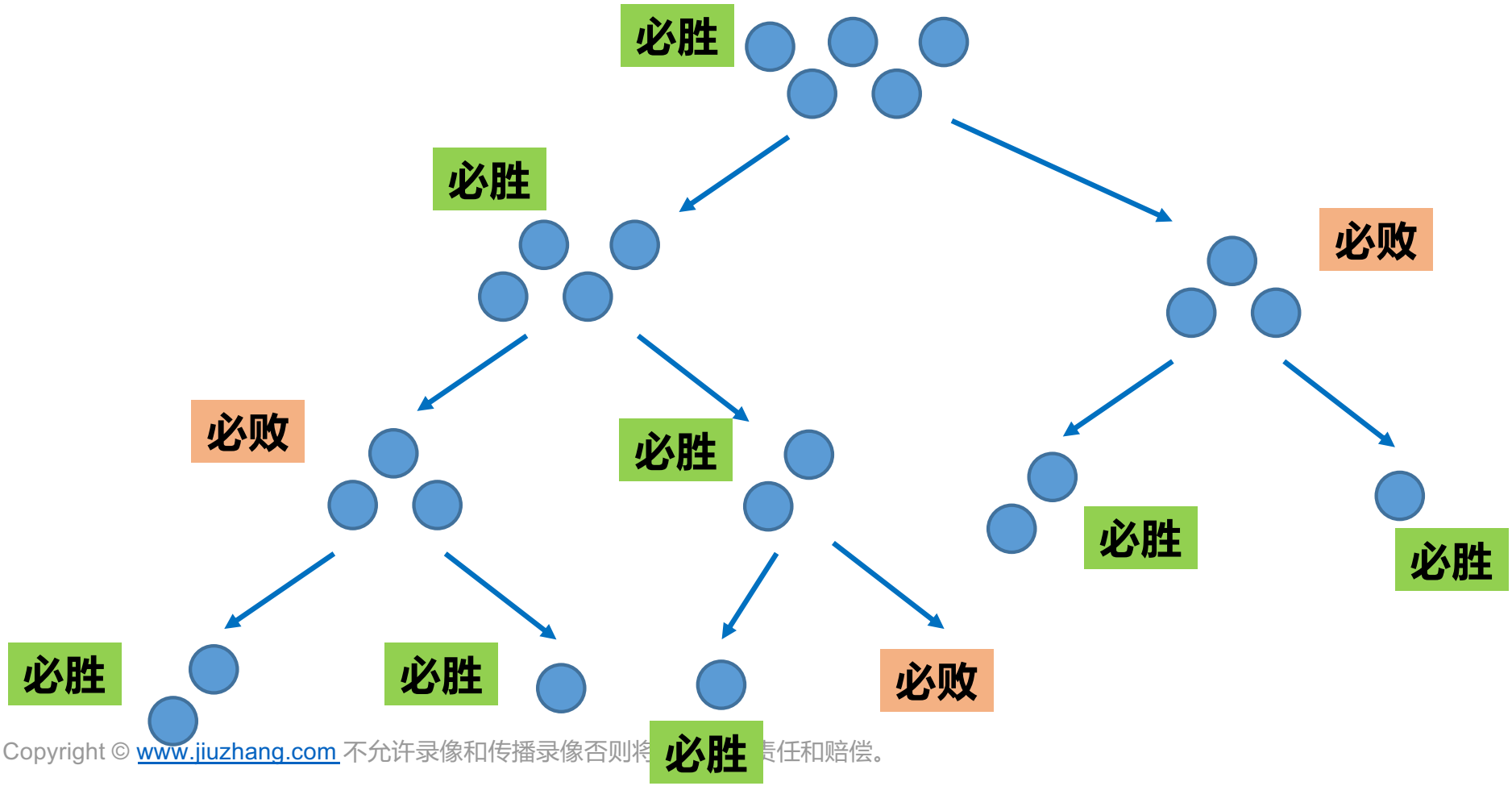
- 有一排N个石子，Alice, Bob两人轮流取石子
- 每次一个人可以从最右边取走1个或2个石子
- 取走最后石子的人胜
- 问先手Alice是否必胜 (先手必胜: true,先手必败: false)

- 例子：
- 输入：N=5
- 输出：true （先手取走2个石子，剩下3个石子，无论后手怎么拿，先手都可以取走最后一个石子）

分析

- 面对N个石子，先手Alice第一步可以拿1个或2个石子
- 这样后手Bob就面对N-1个石子或N-2个石子
- 先手Alice一定会选择能让自己赢的一步
  - 因为双方都是采取最优策略
- 怎么选择让自己赢的一步
- 就是走了这一步之后，对手面对剩下的石子，他必输

博弈动态规划：必胜 vs 必败



博弈动态规划：必胜 vs 必败

九章算法

知识点

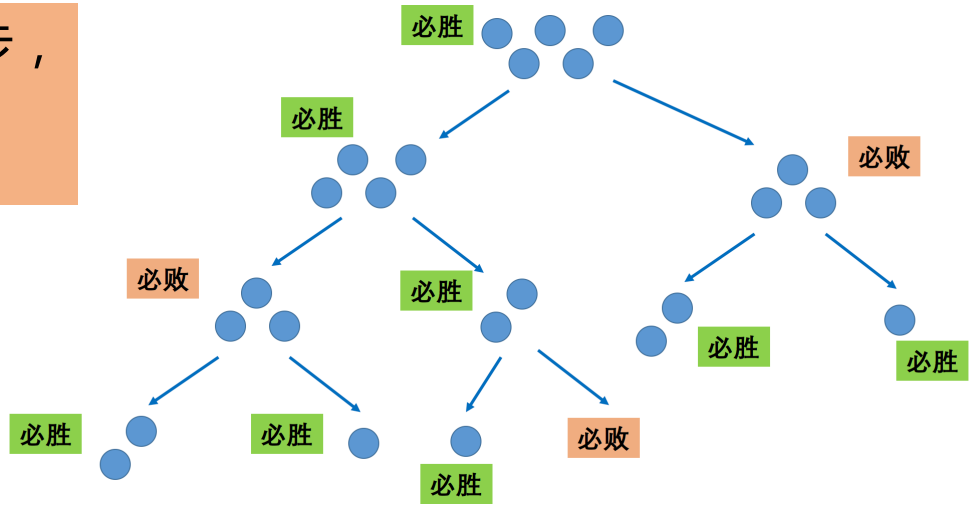
如果取1个或2个石子后，能让剩下的局面先手必败，则当前先手必胜

知识点

如果不管怎么走，剩下的局面都是先手必胜，则当前先手必败

宗旨

必胜：在当下的局面走出一步，
让对手无路可逃
必败：自己无路可逃



分析

- 状态：设f[i]表示面对i个石子，是否先手必胜(f[i] = TRUE / FALSE)
- 转移方程：f[i] = f[i-1] == FALSE OR f[i-2] == FALSE
- 初始条件和边界情况：
  - f[0] = FALSE --- 面对0个石子，先手必败
  - f[1] = f[2] = TRUE --- 面对1个石子或2个石子，先手必胜
- 计算顺序：f[0], f[1], f[2], …, f[N]
- 如果f[N] = TRUE则先手必胜，否则先手必败
- 时间复杂度O(N)，空间复杂度O(N)，可以滚动数组优化至O(1)

Coins in a Line II

https://www.lintcode.com/problem/coins-in-a-line-ii/
https://www.jiuzhang.com/solutions/coins-in-a-line-ii/

LintCode 395

九章算法

- 给定一个序列a[0], a[1], …, a[N-1]
- 两个玩家Alice和Bob轮流取数
- 每个人每次只能从左边取1或2个数
- 双方都用最优策略，使得自己的数字和尽量比对手大
- 问先手是否必胜
  - 如果数字和一样，也算先手胜

- 例子：
- 输入：[1, 2, 2]
- 输出：True

分析

九章算法

- 这题是一道博弈题，目标是让自己拿到的数字之和不比对手小
- 设己方数字和是A，对手数字和是B，即目标是A>=B，等价于A-B>=0。即如果S_A=A-B， S_B=B-A，Alice的目标是最大化S_A，Bob的目标是最大化S_B
- 当一方X面对剩下的数字，可以认为X就是**当前的先手**，他的目标就是最大化S_X=X-Y
- 当他这一步取走数字的和为m后，对手Y变成先手，同理他也要最大化S_Y=Y-X
- 对于X来说，**$S_X = -S_Y + m$**
  - 其中，m是当前这步的数字，$-S_Y$是对手看来的数字差取相反数（因为先手是X）

分析

九章算法

- 状态：设f[i]为一方在面对a[i..n-1]这些数字时，能得到的最大的与对手的数字差
- 转移方程：f[i] = max{a[i] - f[i+1], a[i] + a[i + 1] - f[i + 2]}
- 初始条件：f[n] = 0
- 计算顺序：f[n], f[n-1], …, f[0]
- 如果f[0]>=0，先手Alice必赢，否则必输
- 时间复杂度O(N)，空间复杂度O(N)

区间类DP

特点：

1. 求一段区间的解max/min/count
2. 转移方程通过区间更新
3. 大区间的值依赖于小区间

Burst Ballons

http://www.lintcode.com/problem/burst-balloons/

http://www.jiuzhang.com/solutions/burst-ballons/

LintCode 168

九章算法

- 给定N个气球，每个气球上都标有一个数字：$a_1, a_2, \ldots, a_N$
- 要求扎破所有气球，扎破第i个气球可以获得a[left]*a[i]*a[right]枚金币
  - left和right是与i相邻的下标
  - 扎破气球i以后，left和right就变成相邻的气球
- 求最多获得的金币数（设a[0]=a[N+1]=1）

- 例子：
- 输入：[3, 1, 5, 8]
- 输出：167
  - [3,1,5,8] → [3,5,8] → [3,8] → [8] → []
  - 金币3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167

分析：

- 确定状态：
  - 最后一步：一定有最后一个被扎破的气球，编号是i
  - 扎破i时，左边是气球0，右边是气球N+1，获得金币$1*a_i*1=a_i$
  - 此时气球1~i-1以及i+1~N都已经被扎破，并且已经获得对应金币—子问题
  - 状态：设f[i][j]为扎破i+1~j-1号气球，最多获得的金币数
- 转移方程：$f[i][j] = \max_{i<k<j}\{f[i][k] + f[k][j] + a[i] * a[k] * a[j]\}$
- 初始条件和边界情况：f[0][1] = f[1][2] = … = f[N][N+1] = 0
- 计算顺序：
  - f[0][1], f[1][2], f[2][3], …, f[N][N+1]
  - f[0][2], f[1][3], f[2][4], …, f[N-1][N+1]
  - …
  - f[0][N+1]
- 时间复杂度$O(N^3)$，空间复杂度$O(N^2)$

编程

Coins in a Line III

http://www.lintcode.com/problem/coins-in-a-line-iii
http://www.jiuzhang.com/solutions/coins-in-a-line-iii

LintCode 396

九章算法

- 给定一个序列a[0], a[1], …, a[N-1]
- 两个玩家Alice和Bob轮流取数
- 每个人每次只能取第一个数或最后一个数
- 双方都用最优策略，使得自己的数字和尽量比对手大
- 问先手是否必胜
  - 如果数字和一样，也算先手胜

- 例子：
- 输入：[1, 5, 233, 7]
- 输出：True （先手取走1，无论后手取哪个，先手都能取走233）

博弈

- 和Coins in a Line II类似，目标是让自己拿到的数字之和不比对手小
- 设己方数字和是A，对手数字和是B，即目标是A>=B
- 等价于A-B>=0
- 也就是说，如果Alice和Bob都存着**自己的数字和与对手的数字和之差**，分别记为 S_A=A-B， S_B=B-A
- 则Alice的目标是最大化S_A，Bob的目标是最大化S_B

分析

- 状态：设f[i][j]为一方先手在面对a[i..j]这些数字时，能得到的最大的与对手的数字差—区间型动态规划
- 转移方程：f[i][j] = max{a[i] - f[i+1][j], a[j] - f[i][j-1]}
- 初始条件：f[i][i] = a[i]
- 计算顺序：
  - 长度1：f[0][0], f[1][1], f[2][2], …, f[N-1][N-1]
  - 长度2： f[0][1], …, f[N-2][N-1]
  - …
  - 长度N： f[0][N-1]
- 如果f[0][N-1]>=0，先手Alice必赢，否则必输
- 时间复杂度O(N^2)，空间复杂度O(N^2)，也可以用记忆化搜索
- 区间+博弈

区间DP总结

最后都是求 0~n-1 这段区间的值

逆向思维：考虑最后一步如何做，而不考虑第一步如何做

今日重点三题

- Minimum Path Sum
  - 滚动数组优化
- Coins in a Line
  - 博弈问题
  - 必胜和必败状态
- Burst Balloons
  - 区间型动态规划
  - 消去型问题

Thank You