动态规划实战

七月算法　**曹鹏**

2015年4月13日

提纲

- 二维数组路径最小和
- 最大子数组和
- （最短）编辑距离
- 总结与思考

例1 二维数组路径最小和

- 一个m行n列的二维数组，每个元素是一个非负数，从左上角走到右下角，每次只能朝右或者下走，不能走出矩阵，使得总和最小。(Leetcode 64)
  - 思路——枚举
    - m行,n列。(m + n -2)步 (m – 1)步向下，(n -1)步向右，任意选择，路径条数一共是C(m + n - 2, m -1)，枚举是万能的！
    - 实际不可行

例1 续

- dp?
  - dp[i][j]表示从左上到达(i,j)的最小值
  - dp[i][j] = min(dp[i – 1][j], dp[i][j – 1]) + a[i][j]
    - 从上边过来dp[i – 1][j] + a[i][j]
    - 从左边过来dp[i][j – 1] + a[i][j]
  - 初值 (下标从0开始)
    - dp[0][0] = a[0][0]
    - dp[0][j > 0] = dp[0][j – 1] + a[0][j]
    - dp[i > 0][0] = dp[i – 1][0] + a[i][0]
  - 复杂度: 时间O(m * n), 空间O(m * n)

例1 续2

- 空间优化——省掉一维
  - dp[i][j]只与dp[i – 1][j] , dp[i][j – 1]有关
  - 对每个i,正向循环j
    - 之前的dp[j – 1]是“新的”， dp[j]还是旧的
    - dp[j] = min(dp[j – 1], dp[j]) + a[i][j] 更新
- 贪心的返例——哪小往哪走

0	1	100
2	100	100
1	1	1

例2 最大子数组和

- 一个整数数组，一个非空的子数组(连续一段数)，使得它的和最大 (Leetcode 53)
  - 思路1 暴力枚举，起点i = 0..n – 1，终点j = i…n – 1，求和i..j，时间复杂度$O(n^3)$
  - 思路2 “聪明”枚举，起点0..n – 1, 终点j = i..n – 1，“顺便”求和，时间复杂度$O(n^2)$
  - 思路3 分治
    - 分：两个基本等长的子数组，分别求解 T(n / 2)
    - 合：跨越中心点的最大子数组合（枚举）O(n)
    - 时间复杂度 O(nlogn)

例2 续

- 思路4 dp
  - dp[i]表示以a[i]结尾的最大子数组的和
  - dp[i] = max(dp[i – 1] + a[i], a[i])
    - 包含a[i – 1]：dp[i – 1] + a[i]
    - 不包含a[i – 1]: a[i]
  - 初值 dp[0] = a[0]
  - 答案？最大的dp[0..n – 1]
  - 时间复杂度 O(n), 空间复杂度O(n)
  - 空间优化: dp[i]要存么？
    - endHere = max(endHere + a[i], a[i])
    - 结果answer = max(endHere, answer)

例2 续2

- 思路5 另外一种线性枚举
  - 定义
    - sum[i] = a[0] + a[1] + a[2] +…+ a[i] i>=0
    - sum[-1] = 0
  - 则 对 0 <= i <= j
    - a[i] + a[i + 1] + … + a[j] = sum[j] – sum[i – 1]
  - 我们就是要求这样一个最大值
    - 对j我们可以求得当前的sum[j]，取的i – 1一定是之前最小的sum值，用一个变量记录sum的最小值
    - 时间O(n),空间O(1)

例3 编辑距离

- 给定两个字符串S和T，求把S变成T所需要的最少操作次数。操作包括:在任意位置增加一个字符、减少任意一个字符以及修改任意一个字符。(Leetcode 72)
  - 思路——bfs
    - 题目里有“最小”字样，符合bfs关键词：上界 len(S) + len(T)
    - 实际不可行
  - 思路dp，有删除操作很麻烦

例3 续

- 换个角度思考问题，变为字符串“对齐问题”：
  - S = “ABCF” T = “DBFG”

S	A	B	C	F	–
T	D	B	–	F	G

  - 对应位置相同不扣分，不同则扣一分(修改)
  - 两个特殊字符“–”不会对应
  - S位置“–”代表增加字符
  - T位置“–”代表删掉字符
  - 使扣分最少

例3 续2

- dp[i][j]表示S的前i个位置和T的前j个位置对齐的最少得分
- dp[i][j] = min(dp[i – 1][j – 1] + same(i , j), dp[i – 1][j] ＋ 1, dp[i][j – 1] + 1)
  - dp[i – 1][j – 1] + same(i, j) 对应S第i个字符和T第j个字符对齐
  - dp[i – 1][j] + 1对应S第i个字符和-对齐，即删掉S中第i个字符
  - dp[i][j – 1] + 1对应T第j个字符和-对齐，即在S中加入该字符
- 初值
  - dp[0][j] = j, dp[i][0] = i (i ≥ 0, j ≥ 0)
- 时空复杂度 O(length(S) * length(T))
- 空间优化——省掉一维
  - 对每个i, 正向循环j
    - 注意保存dp[i – 1][j – 1]因为j – 1已经是“新值”

总结与思考

- dp问题注意
  - 递推式
  - 初值
  - 空间优化
- 多练习，多思考
  - Leetcode (leetcode.com) 题号 85, 91, 97, 120, 131, 132, 139, 140, 152