数据结构与算法

Data Structure and Algorithm

XXIV. 动态规划

授课人：Kevin Feng
翻　译：王　落　桐

课前回顾

- 数据结构及算法
- 数学回顾
- 数组（Array）和数组列表（Array List）
- 递归 vs. 迭代
- 二分法搜索
- 分治法
- 链表
- 栈和队列

课前回顾

- 哈希（Hash）
- 树（Tree）
- 堆（Heap）
- 图论（Graph）
- 双向指针（Two Pointers）
- 滑动窗口（Sliding Window）

动态规划 | Dynamic Programming

- 拆分（Divide）：将一个复杂问题拆分成一系列的简单子问题，每一次解决一个子问题并将其结果存储起来。理想情况下用基于内存的数据结构。
- 查找（lookup）：在下一次遇到相同的子问题的时候，直接查找之前计算过的结果而不是重新计算。理想情况下，使用这种方法可以以适当的增大内存占用为代价节省计算时间。
- 存贮子问题的答案以避免从新计算的技术叫做 memorization（记忆化）
- 分治法：把问题拆分成若干独立子问题，解决每个子问题之后合并子问题的结果作为原始问题的结果。
    - Top-Down
- 动态规划：将问题拆分成一些列重复的子问题以得到越来越大的子问题的答案
    - Bottom-up

DP问题 | DP Problems We Have Seen

- 斐波那契数列（Fibonacci）：：f(n) = f(n-1)+f(n-2),

  where f(1)=1, f(2)=1

- 汉诺塔（Hanoi Tower）

- 子序列的最大和

1维动态规划

- 目前我们所见到的DP问题都是1D的，让我们来看一下其他例子：
- 问题描述：给定n，找到不同的将n写成1，3，4相加的方法
- 例如：n=5，答案是6
- 5 = 1 + 1 + 1 + 1 + 1
  = 1 + 1 + 3
  = 1 + 3 + 1
  = 3 + 1 + 1
  = 1 + 4
  = 4 + 1

- 问题重现：

$$D_n = D_{n-1} + D_{n-3} + D_{n-4}$$

- 解决基本问题：

- $D_0 = 1$
- $D_n = 0$ for all negative n
- Alternatively, can set: $D_0 = D_1 = D_2 = 1$, and $D_3 = 2$

- 整合：

```
D[0] = D[1] = D[2] = 1; D[3] = 2;
for(i = 4; i <= n; i++)
    D[i] = D[i-1] + D[i-3] + D[i-4];
```

入室抢劫：

- 假设你是一个职业抢劫犯，你打算洗劫一个街道。每一个房子里有一定数量的钱，限制你的唯一条件是相邻的房子的安保系统是相连的，如果你抢劫相邻的房子那么安保系统就会惊动警察。
- 给定一个非负整数的列表代表每个房子当中的钱，计算在不惊动警察的情况下你可以抢劫到的最多的钱

- 进阶问题：
- 该街道的所有房子是圆形排列的。也就是说第一家和最后一家也是邻居。安保系统的设置同上问题

组织聚会

- 你的公司在组织一个公司的年会。你们公司的组织形式非常严格，如：一个以主席为根的树状结构。现在这个聚会有一个限制：雇员和他的直接领导不能同时受邀参加年会。你希望能够让更多的人来参与到年会中。

- 进阶问题：
- 假设这是一个募款会，组织方会提前知道每个人的捐助意向，所以如何使收到的款最大。

瓷砖问题

- 假定给定一块n*2的地板，瓷砖的大小为1*2。计算需要使用的瓷砖数量。瓷砖可以水平放置：1*2。也可以竖直放置：2*1。

- 进阶问题（很有挑战）
- 将一个3*n的地板用2*1的瓷砖铺满共有多少种方式？

最小台阶问题

- 有一个楼梯，每次可以走1层或者2层，cost数组表示每一层所需要花费的值。
- 你可以从第0节或者第1节台阶开始走，一旦你交了过路费，你可以上一个台阶或者两个台阶。试计算登顶的最少花费

解码方式

- 一段包含着A-Z的短信用一下方式进行编码：
  - 'A' -> 1
  - 'B' -> 2
  - ...
  - 'Z' -> 26
- 给定一段编码的短信，计算解码的方式。

独特二叉树搜索路径

- 给定n，用1…n个数字来表达Binary Search Tree，问有多少种表达方式

最大子序列乘积

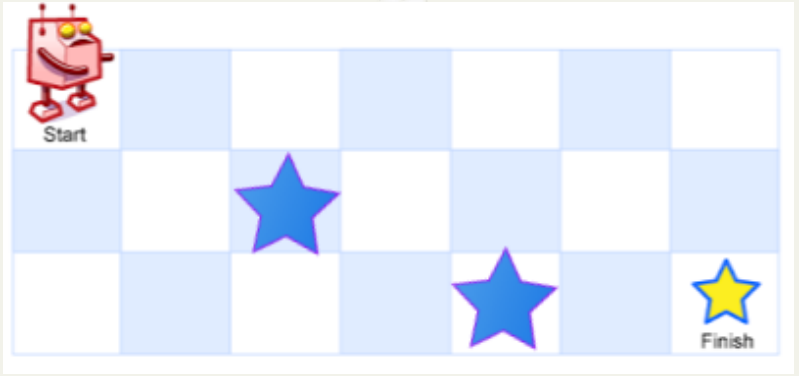
- 找到array中的连续子序列，该子序列的乘积最大。
- 咦？不是做过最大子序列和吗？有什么不一样？

二维动态规划

- 独特路径：Unique Path
- 在棋盘上移动：Moving on Checkerboard
- 最大方形：Maximum Square
- 0/1减缩问题：0/1 Knapsack
- 最大公共子序列：Longest Common Substring
- 最长增长子序列：Longest Increasing Subsequence
- 矩阵链：Matrix Chain
- 移动销售：Sales Travelling
- Floyd-Warshall 算法：Floyd-Warshall Algorithm

独特路径

- 一个机器人被放置在棋盘的左上角，棋盘有（m*n）个格子。机器人只可以往下移动或者往右移动。机器人的目标是到达右下角（如图）
- 可以有多少种不重复的路径：



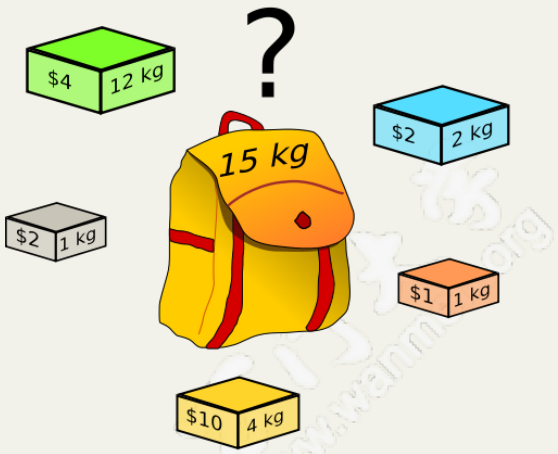
- 进阶问题：目前考虑如果在棋盘中有一些路障。计算不重复的路径。

在棋盘上移动

- 给定一个有n行n列的棋盘。每个格子上有不同的值代表该格子的价值。目标是找到获利最大的移动路线（从第一行的某个格子到最后一行的某个格子）
- 移动方法：
- 1. 移动到下一行的之前一列（up then left）
- 2. 移动到下一行（UP）
- 3.移动到下一行的下一列（UP then right）

0/1 包裹

- 现有若干物品，每一个物品有重量和价值两个属性。现还有一个书包，该书包有一定的承载上限。试找到一种装包方法使得包内物品的重量不超过其限定值且使包内物品的价值最大



最长公共子串

- 给定一个序列： $x_1, x_2, x_3, x_4, \ldots, x_n$
- 子序列定义为一串按照顺序的标志，但是并不一定是连续的序列
- 给定两个序列：$X = x_1\, x_2 \ldots x_n$ and $Y = y_1\, y_2 \ldots y_m$
- 最大公共子序列必须满足其同时是X和Y的子序列。我们想要找到最大公共子序列的长度。

最大递增子序列

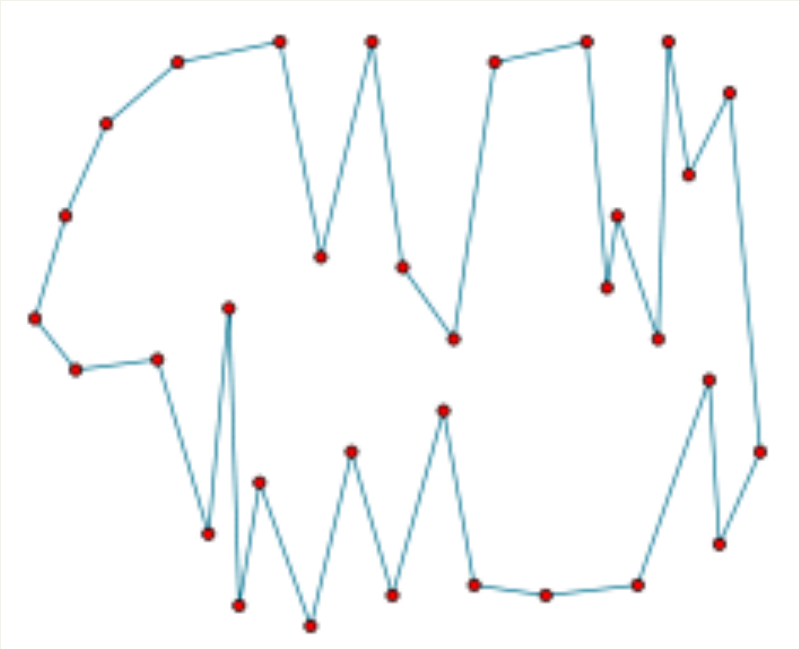
- 给定一个长度为N的数组，找出一个最长的单调自增子序列（LIS）（不一定连续，但是顺序不能乱）。
- 例如：给定一个数组｛10，22，9，33，21，50，41，60，80｝，则其最长的单调递增子序列为{10，22，33，50，60，80}，长度为6.

矩阵链

- A是一个p*q的矩阵，B是q*r的矩阵
- 则矩阵的乘积可以定义为：
- $C = A \times B = \begin{pmatrix} c_{11} & \cdots & c_{1r} \\ \vdots & \ddots & \vdots \\ c_{p1} & \cdots & c_{pr} \end{pmatrix}$ is defined by
- $C_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}$
- 使用标准算法：需要p*q*r次乘法
- 有乘法结合律，$(\boldsymbol{A} \times \boldsymbol{B}) \times \boldsymbol{C} = \boldsymbol{A} \times (\boldsymbol{B} \times \boldsymbol{C})$。然而需要的乘法次数不一定一样。找到最优路径（可以写出括号的可以给出较少的乘次数）

移动销售

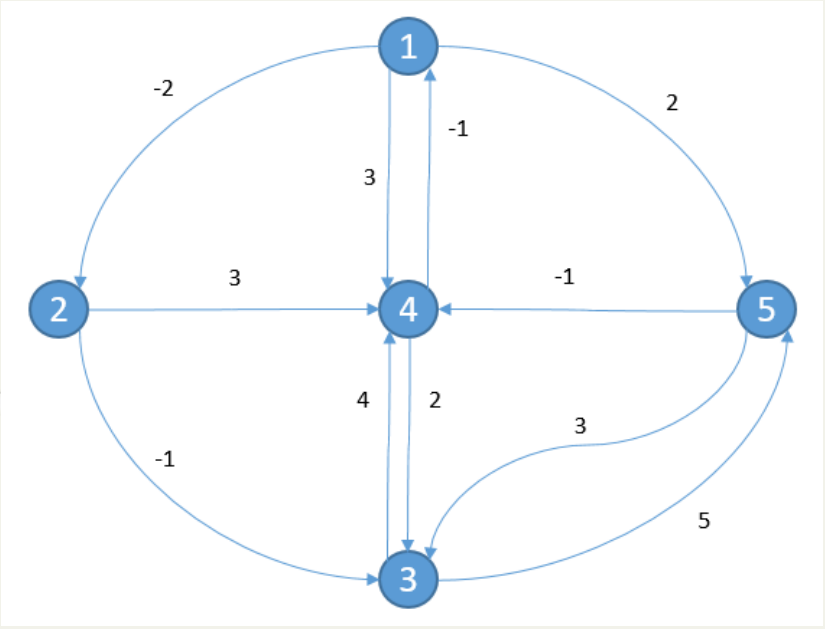
- 在平面上给出n个点，将这些点连成最短的路径（通过欧氏距离测定），要求：从任一点出发，所有点只能访问一次，最终回到起点



Floyd-Warshall 算法

- 我们的目标是寻找从点i到点j的最短路径。

- 从任意节点i到任意节点j的最短路径不外乎2种可能，1是直接从i到j，2是从i经过若干个节点k到j。所以，我们假设Dis(i,j)为节点u到节点v的最短路径的距离，对于每一个节点k，我们检查Dis(i,k) + Dis(k,j) < Dis(i,j)是否成立，如果成立，证明从i到k再到j的路径比i直接到j的路径短，我们便设置Dis(i,j) = Dis(i,k) + Dis(k,j)，这样一来，当我们遍历完所有节点k，Dis(i,j)中记录的便是i到j的最短路径的距离。



数据结构与算法

Data Structure and Algorithm

XXIV. 动态规划结束

授课人：Kevin Feng
翻　　译：王　落　桐