动态规划

杜伟桦

动态规划问题（dp）

- 动态规划是一种在数学、计算机科学和经济学中使用的，通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法。

- 动态规划相较于朴素算法的优秀在于每个子问题只计算一次而减少了冗余。因此，恰当地划分子问题是采用动态规划方法的关键。

动态规划问题（dp）

- **重叠子问题：**
- 将问题重新组合成子问题；为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。
- **最优子结构：**
- 最优子结构的意思是局部最优解能决定全局最优解。简单地说，问题能够分解成子问题来解决。

要素

- 状态
- 转移方程
- 初始条件

例1.最长不下降子序列

- 给定一个数列a_1、a_2、……a_n，求其最长的不下降的子序列的长度。

例1.最长不下降子序列

- 状态：F[i]表示a[i]结尾的最长上升序列长度
- 转移方程：$F[i]=\max(F[j]+1)\quad j<i\ ,\ a[j]\le a[i]$

- $O(n^2)$

例1.最长不下降子序列

- 优化：

- G[i]表示长度为i的不降子序列末尾元素的最小值
- 当计算F[i]时，二分j，使得G[j]≤a[i]
- F[i]=j+1
- G[F[i]]=a[i]

- O(n*log(n))

例2.背包问题

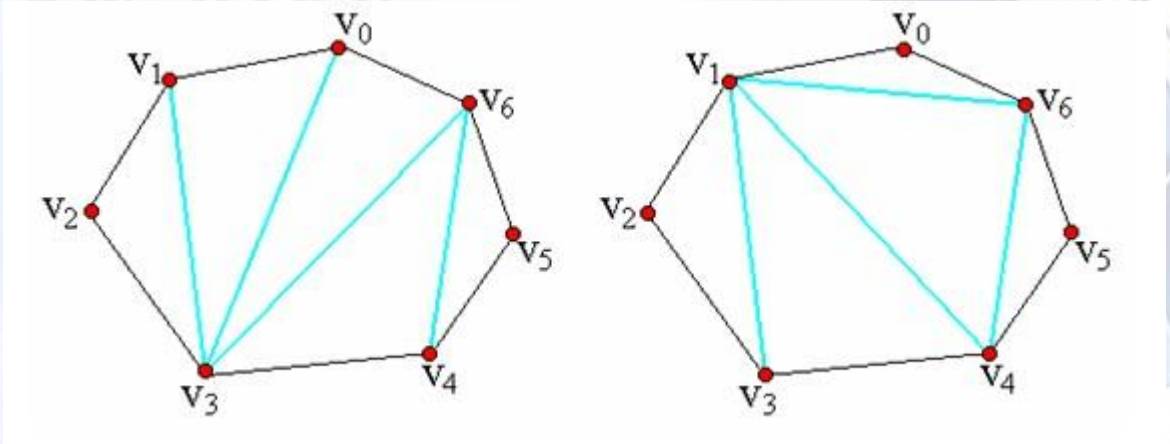
- 0/1背包
- 完全背包
- 多重背包

多重背包的优化

- 优化：
- F[i][v]=max{F[i-1][v-k*c[i]]+k*w[i] | 0<=k<=n[i]}
- 尝试采用单调队列
- 困难：决策区间（v-c[i]，v-2*c[i]，v-3*c[i]，……）不连续。
- 我们发现，如果按“模c[i]同余”将v分类计算，每一类中决策区间都是连续的。

例3.凸多边形的三角剖分

- n个顶点的凸多边形，给定每个点的点权。要求划分成n-2个互不相交的三角形，定义划分代价为这些三角形三个顶点权值乘积之和。求最小的划分代价。



例3.凸多边形的三角剖分

- 状态：
- F[i,j]表示按顺时针顺序从点i到点j这些顶点所构成的凸多边形的最小划分代价。
- 转移：
- 考虑到点i点j间的边一定是某个三角形的一边，枚举这样的三角形（即枚举另一点k），来寻找最优决策。
- $F[i,j]=\min\{F[i,k]+F[k,j]+w[i]*w[j]*w[k]\}$ $i+1\le k\le j-1$
- 区间动态规划；$O(n^3)$

例4.Brackets

- 由()和[]组成的括号序列（无优先级区别）
- 有个脑残把其中所有的[和]都替换成了(
- 给你一个替换后的序列，求可能的原序列个数

- 例如：((((()))
- 原序列可能为：[](((())))或([](()))或(([]()))或((([])))

例4.Brackets

- 思路1：区间DP
- F[i][j]表示[i,j]这个区间构成一个合法的括号序列的方案数
- 从括号序列的定义角度考虑转移：
- 1、如果A是括号序列，则(A)和[A]是括号序列
- 若s[i]=(且s[j]=(，F[i][j]+=F[i+1][j-1]
- 若s[i]=(且s[j]=)，F[i][j]+=F[i+1][j-1]
- 2、如果A、B分别是括号序列，则AB是括号序列
- 这个转移是O(n)的。小心重复计数。
- O(N^3)

例4.Brackets 优化

- 如果只有()，判定其是否为括号序列，只需从左至右扫描，记录“(”的个数-“)”的个数的值，始终大于等于0就可以说明是括号序列。
- 但是如果是()和[]，仅保证“(”的个数-“)”的个数的值，和“[”的个数-“]”的个数的值都始终大于等于0还无法说明是括号序列。如([)]。

例4.Brackets 优化

- 我们可以构建一个新的模型：
- 仅允许将给出序列的“(”改成“)”，使之为括号序列。
- 可以发现，这样的序列与所求序列一一对应。
- 改后序列：((((()))
- 原来序列：([](()))
- 新式序列：((()(())))
- 任何一个合法的新式序列与题目给出的序列对照就可以唯一确定一个合法的原序列。
- 新序列中()若在给出序列中为()则说明原序列中为()
- 新序列中()若在给出序列中为((则说明原序列中为[]

例5.二叉树独立集计数

- 对于一棵二叉树，独立集是指两两互不相邻的节点构成的集合。求一棵二叉树的独立集个数。

例5.二叉树独立集计数

- 状态：
- 对于每个节点x，有二元组（f[x]，g[x]）
- f[x]表示以x为根的子树中，不包含节点x的独立集个数。
- g[x]表示以x为根的子树中，包含节点x的独立集个数。

- 最后的答案就是F[root]=f[root]+g[root]

例6.幼儿园的游戏

- 一天，幼儿园的孩子们与老师们排排坐，形成了一个N*M的矩形队列，一共有K个老师在这个队列中。老师要求每个小朋友都和四周(即上下左右)的任意两个小朋友牵手。
- 小朋友不可以与老师牵手，求牵手的方案数。
- n≤8； m≤1000； k≤100

例6.幼儿园的游戏

- 状态：
- F[i,s]；i表示当前处理到第i列，s表示状态。
- 考虑一下，s需要记录哪些信息才能够转移。
- s其实只要记录每一行是否向右边牵手。所以用2^n来表示s
- 从F[i,s1]转移到F[i+1,s2]，判断一下s1、s2不冲突即可转移。（冲突就是手伸出去没人接着）

例7.扔鸡蛋

- 现在有N个鸡蛋，一座摩天大楼有M层，其中有一个楼层K，如果你从K层或者K层以上扔一个鸡蛋，这个鸡蛋会碎掉，而从K层以下鸡蛋则不会碎。现在希望你用最少的次数求出K。

例7.扔鸡蛋

- 思路1

- 用f[i][j]表示用j个鸡蛋测i层楼最少需要多少次。
- 注意到每次扔一个鸡蛋，会出现两种情况，碎掉或者没有碎，这个是人为不可以控制的。
- f[i][j]=min(max(f[k-1][j],f[i-k][j-1])+1)
- O(NM^2)
- 由于在鸡蛋无限的时候上界是logM次。所以最多只要logM个鸡蛋。
- O(M^2logM)

例7.扔鸡蛋

- 思路2
- 还有一种创造性的想法是改变状态：
- g[i][j]表示用i个鸡蛋j步最多可以确定高度为g[i][j]的高楼。
- g[i][j]=g[i-1][j-1]+g[i][j-1]。
- O(MlogM)

例8.精简序列

- 给你一个长度为N的正整数序列$A_1,A_2…A_N$，你需要踢掉一些数（踢完了以后右边的数字自动往左补），使得踢完后满足$A_i=i$的数字尽量多。

- $N\leq100000$。

例8.精简序列

- F[i]表示考虑前i位使第i位满足a[i]=i的答案。
- a[i]<i的已经没用了
- j能转移到i（j<i）的条件是，1≤a[i]-a[j]≤i-j。
- O(n^2)

例8.精简序列

- j能转移到i（j<i）的条件是，1≤a[i]-a[j]≤i-j。
- 转换一下：j-a[j]≤i-a[i]且a[j]<a[i]。
- 那么每个数都变成了一个二元组（i-a[i],a[i]）。
- 把第一个数升序排列，要求的就是第二个数的最长上升子序列。
- O(NlogN)

单调队列

$$f(x)=\min_{k=b[x]}^{x-1}\{g(k)\}+w[x]$$，**其中，b[x]随 x 不降。**

- 这个方程怎样求解呢？我们注意到这样一个性质：如果存在两个数j,k，使得j<=k，而且g(k)<=g(j)，则决策j是毫无用处的。因为根据b[x]单调的特性，如果j可以作为合法决策，那么k一定可以作为合法决策，又因为k比j要优，（注意：在这个经典模型中，“优”是绝对的，是与当前正在计算的状态无关的），所以说，**如果把待决策表中的决策按照k排序的话，则g(k)必然是不降的。**

单调队列

- 这样，就引导我们使用一个**单调队列**来维护决策表。对于每一个状态f(x)来说，计算过程分为以下几步：
- 1.队首元素出队，直到队首元素在给定的范围中。
- 2.此时，队首元素就是状态f(x)的最优决策，
- 3.计算g(x)，并将其插入到单调队列的尾部，同时维持队列的单调性（不断地出队，直到队列单调为止）。
- 重复上述步骤直到所有的函数值均被计算出来。不难看出这样的算法均摊时间复杂度是O(1)的。

例9.

- 一个长度为n的数列，求出每连续m个数的最小值。
- 要求线性。

例9.

- 用f(x)表示以x作为**结尾**的有M个数的连续片段的最小值。套用单调队列，就可以在O(n)的时间内解决问题。

例10.oil

- 给出一条线段，在左端点点0与右端点n+1间有n个点，并且在0到x之间的所有点都是有油的，在每个点钻井判断是否有油需要时间ti，求能够知道x的最坏情况下最少需要多少时间。
- n≤2000

例10.oil

- 状态设计采用区间DP的思路
- F[i][j]表示判断[i,j]这个区间需要的时间。
- 朴素：
- F[i][j]=min{max(F[i][k-1],F[k+1][j])+t[k]}
- O(n^3)
- 由于F[i][k]随k单调不减，F[k][j]随k单调不增。所以找出分割点就可以了。用单调队列找出分割点。

例11.中国象棋

- 在一个n 行m 列的棋盘上放若干个炮，使得没有任何
- 一个炮能攻击另一个炮，求方案数模9999973。
- 1 ≤ n; m ≤ 100。

例11.中国象棋

- 每行每列不能放超过两个炮。
- 朴素动态规划？
- 用f(i; j) 表示到第i 行为止各列放炮数量为j（j 为m 位三进制数）的方案数。
- 转移只要枚举当前行在哪一列或哪两列放炮即可。
- 时间复杂度？
- O(3^m*m^2*n)。

例11.中国象棋

- 我们并不需要知道放了0/1/2 个炮的具体是哪几列，只
- 要知道放了0/1/2 个炮的各有多少列就可以了。
- 用f(i; j; k) 表示到第i 行为止有j 列放了0 个炮，有k 列放了1 个炮的方案数。
- 简单分类讨论，
- f(i; j; k) = f(i − 1; j; k) + (j + 1)f(i − 1; j +1; k−1)+
- (k + 1)*f(i−1; j; k+1) + k*(j+1)*f(i−1; j+1; k)+
- (j+2)*(j+1)/2*f(i − 1; j + 2; k − 2) + (k+2)*(k+1)/2*f(i − 1; j; k + 2)。
- O(m^2n)。

例题12.Raucous Rockers

- 有n 首歌长度分别为a1; a2; .. ; an，要从中选一些歌发
- 行m 张长度不超过t 的唱片，歌必须按编号顺序选进唱片且
- 只能被选一次，求最多能选多少首歌。
- $1 \le m \le n \le 1000$，$1 \le ai \le t \le 109$。

例题12.Raucous Rockers

- 用f(i,j,k) 表示到第i 首歌为止用了j 张唱片外加长度k
- 最多能选多少首歌。
- f(i,j,k) = max(f(i-1,j,k), f(i-1,j,k-ai)+1) k≥ai
- f(i,j,k) = max(f(i-1,j,k), f(i-1,j-1,t-ai)+1) ai>k
- O(mnt)。

例题12.Raucous Rockers

- 要使在给定长度的唱片中选的歌最多，也就是要使在给定选歌数量时所用唱片数和外加长度最少。
- 用f(i, j) 表示到第i 首歌为止选了j 首歌所用最少唱片数和外加长度（因此是二元组）。
- 定义(x,y) + z 表示用了x 张唱片外加长度y 时再选一首长度为z 的歌后所需唱片数和外加长度（也是二元组）。
- 对于第i 首歌，转移有不选和选两种，
- f(i, j) = min{f(i − 1, j), f(i − 1, j − 1) + ai}。
- O(n^2)。

斜率优化

$$F[i]=\min\{F[j]+Sum[i,j]\}+k(k为常数)$$

- 我们不能对其进行比较有效果的优化，因为它的转移，涉及到了关于i和关于j的一些数组，这时我们就需要用斜率优化了。
- 通常我们令k<j<i，且用j来更新F[i]比用j优。则有

$$F[j]+Sum[i,j]<F[k]+Sum[i,k]$$

- 并且我们都可以化成如下形式，X[j],Y[j]指只关于j的数，X[k],Y[k]亦之

$$\frac{Y[j]-Y[k]}{X[j]-X[k]}<F[i]$$

- 然后就可以对其进行单调队列优化了。

例题13.防御准备

- 有一排n 个检查点，第i 个点要么用ai 的花费建立防御
- 塔，要么要依靠其右侧离它最近的防御塔j 并用j – i 的花费
- 通过检查，求使所有点通过检查的最小总花费。
- 1 ≤ n ≤ 106，1 ≤ ai ≤ 10^9。

例题13.防御准备

- 用f(i) 表示到第i 个点为止（第i 个点必建立防御塔）所需最小总花费。
- f(i)=min{f(j)+(i-j)*(i-j-1)/2}+a(i)
- O(N^2)

例题13.防御准备

- 变形为：$f(i) = \min\{f(j) + j*(j+1)/2 - i*j\} + a(i) + i*(i-1)/2$
- 斜率优化