简单动态规划

北京大学 洪华敦

动态规划

- 由已知状态去计算未知的状态
- 一般没有后效性
- 等价状态较少

求斐波那契数列

- Fib[1..n]是已知状态
- Fib[n+1]是未知状态
- 有已知得到未知：Fib[n+1]=Fib[n]+Fib[n-1]

最长上升子序列

- 给定一个长度为 n 的数组，求最长上升子序列的长度
- $f[i]=\max(f[j]+1)$. $(a[i]>a[j],\ i>j)$

方格路径

- 给定一个 N*N 的方格，每个格子上有权值，现在要找一条从左上到右下，每次只能向下或向右走的路径，使得权值之和最大

- f[i][j]->f[i][j+1]

- f[i][j]->f[i+1][j]

买票

- 有 n 个人排队买票，第 i 个人要花 s[i] 块钱，但如果 i-1 和 i 一起买的话只要花 d[i] 块钱

- 求最少花多少钱才能买到所有人的票

- n<=10^5

买票

- f[i] 表示前 i 个人买票花的最少总金钱数
- $f[i]=max(f[i-1]+s[i],f[i-2]+d[i])$
- O(n)

动态规划的优化

- 通过分析无用状态，减少状态数
- 用数据结构优化
- 优化转移

CCPC

- 给定一个小写字母字符串 S，定义一个字符串的收益为里面CCPC作为连续的子串的出现次数
- 现在你可以往 S 里面插入任意一个字符，第 i 次插入付出 i-1 的代价
- 求最大的收益
- |S|<=10^5

CCPC

- f[x][i][j]：考虑了S[1..x]，和CCPC匹配了 i 位，加了 j 个字符的最大总收益
- O(n^2)
- 观察到 j>2时都是无用状态
- O(n)

树形dp

- 状态设计与树结构相关的dp

树形依赖背包

- 给定一棵 n 个点的有根树，每个结点有一个重量 v[i]，现在你需要选一个包含根的连通块，使得里面的结点的重量之和不超过 M 且尽量最大

- n<=2000，0<=v[i] , M<=2000

树形依赖背包

- 求出dfs序
- f[i][j]表示考虑完了dfs序 <i 的所有点，当前的重量之和为 j
- f[i][j] -> f[i+1][j]
- f[i][j] -> f[ed][j]
- f[i][j] -> f[i+1][j+v[i]]
- O(nM)

求直径

- f[x] 表示以点 x 为根的子树里的直径
- g[x] 表示以点 x 为根的子树里，以 x 为一端的直径
- $g[x]=\max(g[y]+w)$
- $f[x]=\max(f[y],g[x]+g[y]+w)$
- $O(n)$

黑暗森林

- 给定一棵树，每条边有一半的概率长度为 0 或 1
- 求期望的直径
- n<=300

黑暗森林

- 枚举 L，计算直径长度 <=L 的概率
- f[x][i] 表示，以 x 为根的子树，直径不超过 L，且以 x 为一端的直径为 i 的概率
- 合并：枚举 i , j，要求 i+j<=L
- 时间复杂度：O(n^3)

最远点

- 给定一棵带正边权的无向树，求每个点的最远点
- n<=10^5

最远点

- 随便选一个点当根，变成有根树
- 最远点要么在子树里，要么在子树外
- 子树里 : f[x]表示以 x 为根的子树里 x 的最远点离他的距离

最远点

- 子树外：
- g[x] : 点 x 离以 x 为根的子树外的点的最远距离
- f[x] 直接 bfs计算
- 然后枚举父亲来计算 g
- O(n)

树上独立集

- 给一棵树，每个点有点权，求一个点权和最大的独立集
- n<=10^5

树上独立集

- f[x] 表示 x 没有选的情况下，以 x 为根的子树里的最大点权和独立集
- g[x] 表示 x 选的情况下，以 x 为根的子树里的最大点权和独立集
- f[x]=sum(max(f[x],g[x]))
- g[x]=v[x]+sum(f[x])
- O(n)

区间dp

- 状态都是一个个区间的 dp

回文串

- 给定一个长度为 n 的字符串 S，求最少插入多少字符串使得他能变成回文串
- n<=5000

回文串

- f[l][r] 表示将 S[l..r] 变成回文串至少需要插入几个字符
- f[l][r] <— f[l+1][r] , f[l][r-1] , f[l+1][r-1]
- 时间复杂度：O(n^2)

取元素

- 有一个 n 个元素的序列 a，你每天可以从两端中任意取一个数，第 i 天取出的数 w 对答案的贡献为 i*w，求最大化取完时的贡献和

- n<=5000

取元素

- f[l][r] 表示序列只剩下 a[l..r]，接下来最大的贡献之和的值
- f[l][r] ——> f[l][r-1]
- f[l][r] ——> f[l+1][r]
- O(n^2)

合并石子

- 有 n 个石子，第 i 个的大小为 w[i]，每次可以合并相邻两个，得到他们两个大小之和的代价，且他们会并成一个新的石头，新的石头的大小是旧的两个的大小的和
- 求最大的代价之和
- n<=300

合并石子

- f[L][R]表示合并第 L 个到第 R 个石头获得的最大价值和
- 考虑最后一步合并的两个石子是哪两个区间
- f[L][k]+f[k+1][R]+sum[L…R] —> f[L][R]
- O(n^3)

取数字

- 给一个序列 a[1..n]，每次可以拿走连续一段满足某些条件的数，拿完后序列会并起来
- 求最少几次把整个序列拿完
- n<=100

取数字

- f[L][R] 表示 a[L..R] 最少几次取完
- 考虑 a[L..R]里最后取的那一段现在是哪些
- 于是把 a 分成了若干段，分别做子任务
- 复杂度视具体情况而定

状压dp

- 一般都是大力把状态压缩成数字，方便递推

Valley Number II

- 给定一张无向图，每个点可能是高点或者低点，三个点(x,y,z)被称为一个valley当且仅当 x,z 是高点，y是低点，且存在边(x,y)和(y,z)

- 现在要求最多有几个valley，每个点最多只能在一个valley里

- N<=30，高点数量<=15

Valley Number II

- f[i][S]表示考虑到了第 i 个低点，高点的使用状态为 S，最多的valley数量
- 转移时，对于该低点，枚举两个高点即可。
- O(2^k*n^3)

奇怪的道路

- 求有多少无向图满足有n个点m条边，且每个点度数为偶数，并且对于每条边 (u,v) 都有 $|u-v|<=K$
- n,m<=30
- K<=8

奇怪的道路

- 状压一下前面K个点的奇偶性
- f[i][j][S][p]表示，当前考虑了前 i 个点，一共用了 j 条边，i-K..i 的度数的奇偶性是 S，且当前考虑到了边 (i-p,i)
- 只要考虑一下当前的边用不用即可
- O(nmK*2^K)

独立集

- 给定一张 n 个点的无向图，小 A 会随机一个 [1..n] 的排列，然后按这个排列的顺序来贪心选独立集
- 求选出来的独立集的期望值
- n<=20

独立集

- f[S]表示一顿选择后集合 S 为独立集的概率
- 考虑集合 T，表示哪些点可以与 S 并起来组成独立集
- 我们只关心排列里下一个 T 里的元素
- O(n*2^n)

矩阵

- 一个 n*m 的01矩阵，要求不能有相邻的 1 ，求方案数
- n,m<=12

矩阵

- 考虑枚举每个点下来，维护一条轮廓线就行了
- O(nm2^min(n,m))

King’s Visit

- 给一个 8*8 的方格，里面有障碍，一个King可以往周围8个格子走，给定起点，求一条最长的简单路径
- n,m<=8

King’s visit

- 插头 dp
- 记录一下每个点的度数以及连通性
- dp到某个点时，转移是枚举他到之前那些点有哪些出边
- 复杂度：玄学

多重背包

- 有 n 个物品，每个物品体积为 v[i]，且有 d[i] 个，求选一些物品使得体积和为 W 的方案数
- n,W<=5000

多重背包

- f[i][j] 表示考虑了前 i 种物品，体积和为 j 的方案数
- 朴素转移：
- f[i][S] -> f[i+1][S+k*v[i+1]]
- 可以用前缀和优化
- O(nW)

背包问题

- 有 n 个物品，每个物品有价值 w[i]，体积 v[i]，个数d[i]
- 求体积为 S 的背包最多可以装下多少价值的物品
- 1<=n, v[i]<=80，1<=w[i],d[i],S<=10^9

背包问题

- 先二进制分组，变成01背包
- 同一个二进制组下价值和不会超过 n*v[i]
- 所以可以f[i][S]表示当前考虑到第 i 位，已经用的体积为 S 的最大价值
- 时间复杂度：O((nlogn)*logS*n*v[i])

背包问题

- 有 n 个物品，第 i 种物品体积为 i，数量为 i，求填满大小为 n 的箱子的方案数
- n<=10^5

背包问题

- 分 i<=sqrt(n) 和 i>sqrt(n) 来做
- 前者：物品数量为 sqrt(n) 的多重背包，时间复杂度：O(n^1.5)
- 后者：相当于无限制背包，f[i][j] 表示选了 i 个物品，当前体积和为 j 的方案数
- f[i][j] -> f[i][j+i]
- f[i][j] -> f[i+1][j+sqrt(n)+1]
- 时间复杂度：O(n^1.5)