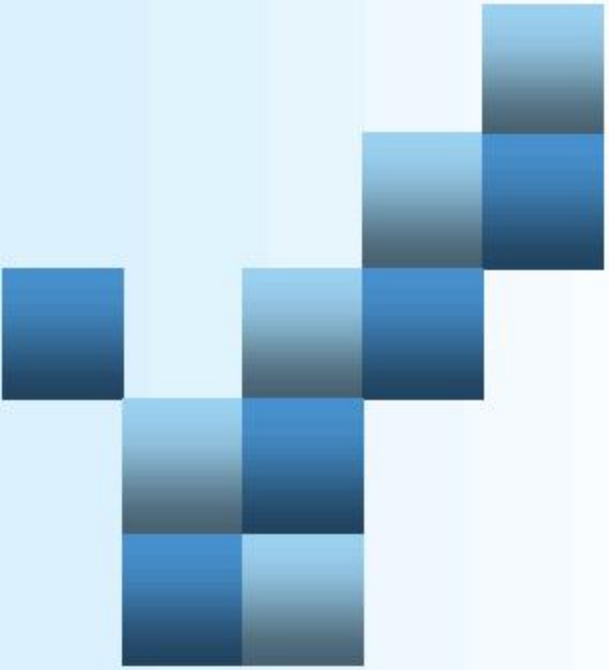


动态规划
Dynamic Programming

~~2016-8-1~~ 2017-8-8

动态规划题目有什么难点？~~（咋感觉眼熟）~~

- （1）状态是什么？（2017年）
  - 最大的难点
  - 如何发现状态？
    - 几何直观（从上到下，从左到右，斜向）
    - 已知的顺序结构（时间轴，坐标轴）
    - 依赖关系
    - 先发现一个完全无法接受的状态，再删除不要的重叠的状态
- （2）时间效率
- 出题人喜欢考察哪一部分？-时间效率
- （~~2016年~~ 2017年）
  - 出题容易
  - 有常见套路

四个决定时间效率的因素~~（咋感觉也眼熟）~~

- (1)状态总数
- (2)每个状态的决策数
- (3)每次状态转移所需的时间
- (4)寻求转移过程的相关性

(1)，减少状态总数

- (1-1)合并相似的状态
- (1-2)改变状态表示;
- (1-3)选择适当的规划方向;

- *审查题目中的信息

1-减少状态总数

1.1-合并相似的状态

- 有的时候
- 只有这一步做好了
- 才会发现是一道动态规划

- 圆桌问题

- 给定 n 个座位的圆桌，有 m 个人，问有多少可能的序列 a[1], a[2], ..., a[m] 规定：
  - m 个人依次抵达
  - 第 i 个人抵达后坐在 a[i] 的位置
- 且对于给定的 k 要求任意时刻连续就位的 group 不超过k

1-减少状态总数
1.1-合并相似的状态

- 圆桌问题

- 我们需要一个状态表示：$m*2^n$规模的状态

- 考虑状态：
  - S1:...oooooxxxoooooxxxooo...
  - S2:...oooooxxxooooxxxoooo...
- 从初始状态到S1,S2的方案相同（为什么？考虑构成）
- 合并为同一个状态！

1-减少状态总数
1.1-合并相似的状态

- 圆桌问题

- 具体的位置是不需要考察的，只需要知道有多少段
- 改变思考角度：
  - 有一个圆桌，一开始没有凳子
  - 每一个人来的时候自己带来了凳子（给自己坐，或者当作间隔）
- 考虑状态 i, j：来了i个人，分成了j组

- F[i][j] = F[i-1][j]*(i+j) + F[i-1][j-1]*(j-1)
- 回到原问题？组合数学！

1-减少状态总数
1.2-改进状态表示

青蛙过河

- 宽度为L的小河上有一座独木桥,青蛙想过河
- 河中有一些荷叶
- 河宽和青蛙一次跳过的距离都是正整数
- 0,1,.....,L
- 坐标 0 的点位于河的一侧,坐标 L 的点位于河的另一侧
- 青蛙从 0 开始,不停的向坐标为 L 的点的方向跳跃
- 一次跳跃的距离是 S 到 T 之间的任意正整数(包括 S,T)
- 当青蛙跳到或跳过坐标为 L 的点时,就算已经越过河了

- 问：青蛙要想过河最少需要踩到的荷叶数

1-减少状态总数
1.2-改进状态表示

- 状态？
  - 坐标位置
  - 荷叶的位置

1-减少状态总数
1.2-改进状态表示

- 寻找无向图中长度为 k 的简单路径

- 给定无向图 G=(V,E)，常数 k。
- 对于每一对结点 (u,v)，询问是否有从u到v经过恰好k个结点的简单路径？

- k <= 6
- |V| * |E| * <= 1,500,000

1-减少状态总数
1.2-改进状态表示

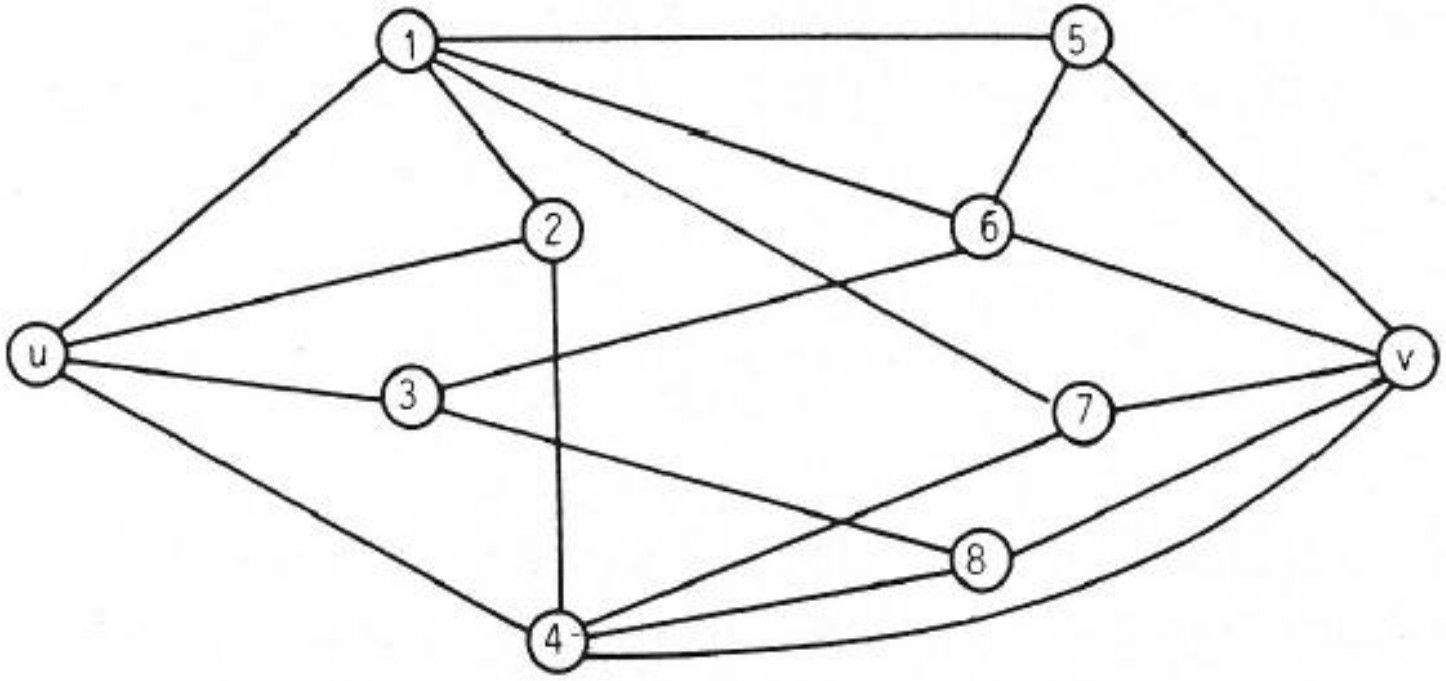
- F(source,target,len,S)
  - 从source出发至target结束
  - 经过的点集为S（不包括source, target）
  - 经过的点集大小为len
  - 这样的方案是否存在?

- 转移的时候，需要考虑结点x
- 问是否存在不包含x的集合S，满足F(source,target,len,S)

可能的S太多了！

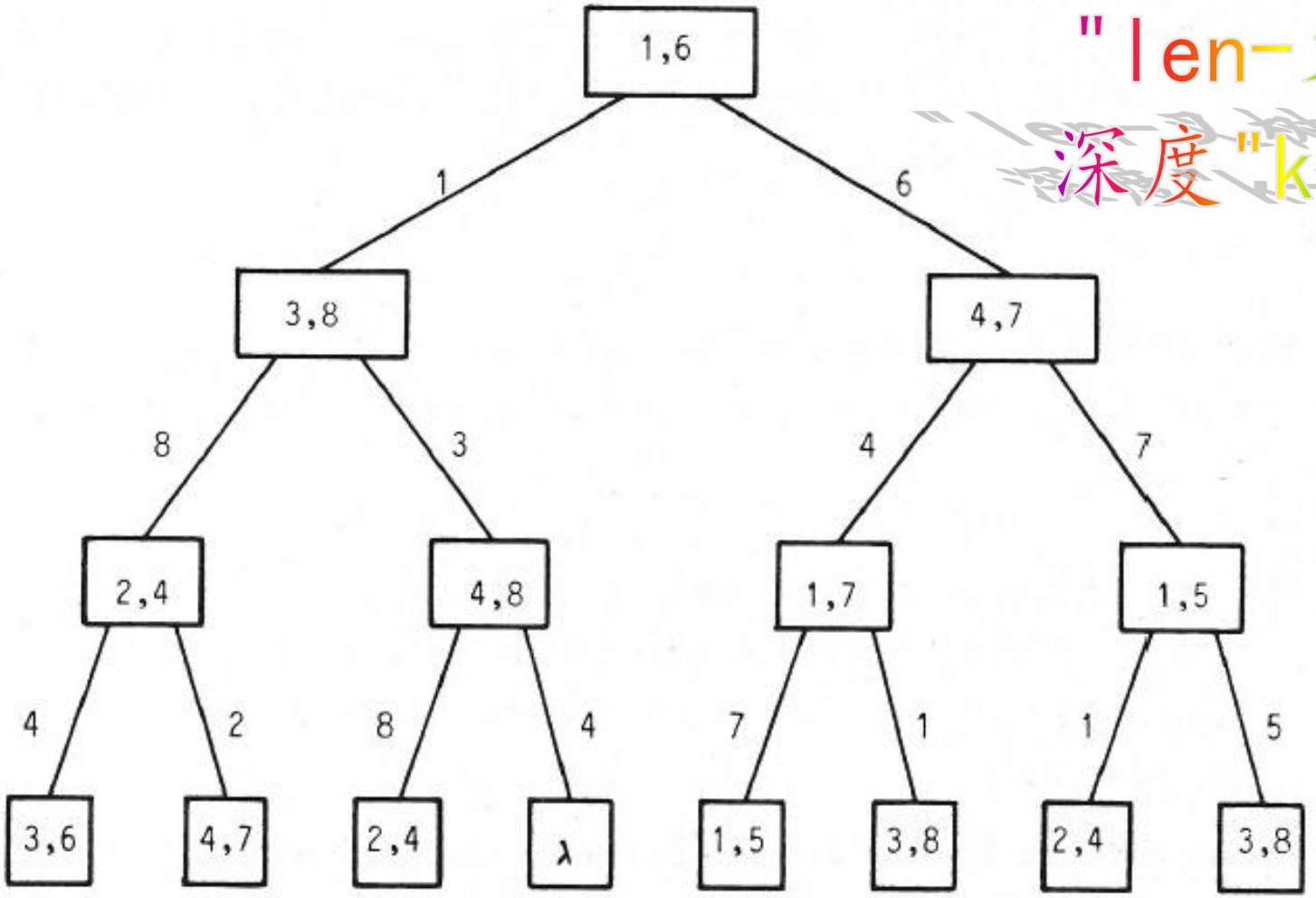
1-减少状态总数
1.2-改进状态表示

- Sample：
- {S|F(u,v,2,S)} = {{2,4},{1,5},{1,6},{1,7},{3,6},{3,8},{4,8},{4,7}}



1-减少状态总数
1.2-改进状态表示

- {S|F(u,v,2,S)} = {{2,4},{1,5},{1,6},{1,7},{3,6},{3,8},{4,8},{4,7}}



"len-叉树"

深度"k-len"

1-减少状态总数
1.3-选择适当的规划方向

- KYKNEION ASMA

Problem Description

On the last day before the famous mathematician Swan's death, he left a problem to the world: Given integers n and a_i for $0 \leq i \leq 4$, calculate the number of n-digit integers which have at most a_i-digit i in its decimal representation (and have no $5, 6, 7, 8$ or 9). Leading zeros are not allowed in this problem.

Input

There is one integer T $(1 < T \leq 10)$ in the beginning of input, which means that you need to process T test cases. In each test case, there is one line containing six integers representing n and a_0 to a_4, where $2 \leq n \leq 15000$ and $0 \leq a_i \leq 30000$.

1-减少状态总数
1.3-选择适当的规划方向

- 方案一：考虑状态(p,x)，前p个数字，总计用了x个
  - 状态数：5n
  - 转移：线形

- 方案二：考虑所有合法（不合法情况组合，共计S个
  - 每一个组合对应长度为n的信息
  - 不同组合之间的转移是线形的
    - 每一次转移是O(1)的
  - $S = 3^5$. $S = 5*2^5$

- 利用本题常数5的特性，方案二定义状态的方法可以使“状态数”x“转移”
- 更优！

1-减少状态总数
1.3-选择适当的规划方向

数组

- 给定 3n 个整数$X_1,X_2,\ldots,X_{3n}$
- 将{1,2,...,3n}分为3个长度为n的序列A,B,C
- 最大化$S = sum\{(X_{Ai}-X_{Bi})*X_{Ci} | 1<=i<=n\}$

- n <= 20

1-减少状态总数
1.3-选择适当的规划方向

- 不妨假设X已经从小到大排序好了

- Fact1: 对于同一组(A_i,B_i,C_i)一定有$A_i > C_i > B_i$
- Fact2: 如果$C_i > C_{i+1}$，则$A_i > A_{i+1}$且$B_i < B_{i+1}$
- Fact3: $B_i = i$

- 按照 i 从小到达选取 A_i 和 C_i
- 则：[3n-i+1,3n]一定已经选取，[n+1,2n-i]一定没有选取

- 只需要记录[2n-i+1,3n-i]段的选取情况即可

2-减少每个状态决策数的基本策略

- (2-1)利用最优决策的单调性
- (2-2)优化决策量
- (2-3)合理组织状态
- (2-4)细化状态转移

- *观察转移方程

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- Motivation（2-1，2-2）：
  - 排除掉一些不可能转移方向
  - 减少整体的转移量

- 2D2D-型：
  - f[i][j] = $\min_{i<k<=j}$ {f[i][k-1]+f[k][j]+w[i][j]}

- 1) 四边形不等式
  - – f[i][j] + f[i'][j'] <= f[i'][j] + f[i][j'] , i<=i'<=j<=j'
- 2）区间单调
  - – f[i'][j] <= f[i][j'] , i<=i'<=j<=j'

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- 最优排序二叉树
- ● 所谓二叉排序树是指具有下列性质的非空二叉树
  - (1)若根顶点的左子树不空,则左子树的所有顶点值均小于根顶点值
  - (2)若根顶点的右子树不空,则右子树的所有顶点值均不小于根顶点值
  - (3)根结的左右树也分别为二叉排序树;

- 【输入】
  - 第 1 行为关键字数 n
  - 第 2 行为 2n 个正整数,依次为n(1≤n≤2000)个关键字的权值 k_i 和查找频率 $p_i(1≤i≤n)$;
- 【输出】
  - 对应二叉排序树的总查找长度
  - – $\Sigma_{1<=i<=n}\ p_i\ (\ depth(k_i) + 1)$ 的最小值。

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- C[i][j] 表示：
  - 顶点i~j对应的子树的最小查找长度, w[i][j] 从i->j的频率和。
- C[i][i] = p_i.
- C[i][j] = w[i][j] + $\min_{i<k<=j}$ {C[i][k-1]+C[k][j]}.

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- 石子合并问题
- ● 在一个操场上摆放着一排n<=2000堆石子。现要将石子
- 有次序地合并成一堆。规定每次只能选相邻的 2 堆石子
- 合并成新的一堆,并将新的一堆石子数记为该次合并的
- 得分。

- 【输入】
  - 第 1 行为石子堆数 n;
  - 第 2 行为 n 个正整数,依次给出 n 堆的石子数;
- 【输出】
  - 将 n 堆石子合并成一堆的**最小**得分。

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- ● f[i][j] : 合并i~j需要的最小费用
- ● f[i][j] = $\min_{i<k<=j}$ {f[i][k-1]+f[k][j]} +d[i]+...+d[j]

2-减少每个状态决策数的基本策略
2.1-利用最优决策的单调性

- 邮局
- ● 按照递增顺序给出一条直线上坐标互不相同的 n 个
- 村庄,要求从中选择 p 个村庄建立邮局,每个村庄使
- 用离它最近的那个邮局,使得所有村庄到各自所使用
- 的邮局的距离总和最小。

- 【输入】
  - 第 1 行为村庄数 n 和邮局数p(1≤p≤n≤2000)
  - 第 2 行为 n 个村庄的 x 坐标;
- 【输出】
  - 最小距离和

2-减少每个状态决策数的基本策略
2.2-优化决策量

- Motivation（2-1，2-2）：
  - 排除掉一些不可能转移方向
  - 减少整体的转移量

- 贪心 结论 规律 ...
- （哪里有大量的贪心题呀？）

2-减少每个状态决策数的基本策略
2.2-优化决策量

- 石子合并
- ● 在一个圆形操场的四周摆放 n 堆石子(1≤n≤2000),现
- 要将石子有序地合成一堆。规定每次只能选相邻的两
- 堆合并成新的一堆,并将新的一堆的石子数记为该次
- 合并的得分。

- 【输入】
  - 第 1 行为堆数 n;第 2 行为每堆的石子数;
- 【输出】
  - n-1 次合并后的**最大**得分总和。

2-减少每个状态决策数的基本策略
2.2-优化决策量

- ● f[i][j] = max$_{i<k<=j}$ {f[i][k-1]+f[k][j]}+d[i]+...+d[j]
- ●
- ● s[i][j] = k*
- ● s[i][j] in {i+1,j}

2-减少每个状态决策数的基本策略
2.2-优化决策量

- Winning an Auction

- N 件物品
- Alice有A元钱，Bob有B元钱。
- 对N件物品进行出价：
  - 如果Alice出的钱更高，则物品归Alice
  - 如果Bob出的钱更高，则物品归Bob
  - 如果一样：奇数轮归Alice，偶数轮归Bob
- 问：最优情况下，Alice和Bob分别会获得多少物品

2-减少每个状态决策数的基本策略
2.2-优化决策量

- Winning an Auction

- 要先想清楚状态！
- F[i][a][b]：
  - 还有i个物品的时候
  - Alice，Bob分别有a, b的钱
  - 这时候最优方案进行下去，结束的时候Alice有F[i][a][b]的钱
- 那么这时候Alice和Bob的目标分别是什么？
  - Alice：要最大化F[i][a][b]
  - Bob：最小化F[i][a][b]
- 枚举所有策略？

2-减少每个状态决策数的基本策略
2.2-优化决策量

- Winning an Auction

- F[i][a][b]：
  - 还有i个物品的时候
  - Alice，Bob分别有a, b的钱
  - 这时候最优方案进行下去，结束的时候Alice有F[i][a][b]的钱
- 有可能的策略一定是双方都满意的，也就是均衡点！
- 怎么求均衡点？

- 均衡点唯一，找到后转移O(1)

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- 尤其对于二异型的状态：
  - 或为真或为假
- 例如：
  - 做不能做到？
  - 能不能抵达？
  - 可不可行？

- 如何存储状态？
  - 直接维护F[s]？
  - 维护下标为F[s]的位置？
  - 更多？

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- 求最长单调上升子序列

- $F[i][j] = OR_{i_2(a[i_2]<a[i])} F[i_2][j-1]$
- $= OR_{x(x<a[i])} (T[x]>=j-1)$

- 若 $j' = max_{x(x<a[i])} T[x]$，则 $F[i][j]$ 只在 $j>j'$ 的时候为真

2-减少每个状态决策数的基本策略
2.3-合理组织状态

叠 放 箱 子 （ 不讲了 ）

- ● 某港口有一批箱子,将其编号,分别为 1 至 N。每一个箱子的尺寸
- 规格是一样的,现在要将其中某些箱子叠放起来,箱子叠放的规则如
- 下:
  - 1、每个箱子上最多只能直接叠放一个箱子;
  - 2、编号较小的箱子不能放在编号较大的箱子之上;
  - 3、每个箱子都给出了自身重量与可承受重量,每个箱子之上的所有箱子重量之和不得超过该箱的可承受重量。

- ● 为了节约堆放场地,希望你编程从中选出最多个箱子,使之能够在满
- 足条件的情况下叠放起来。

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- ● 设函数 f(i,j),其含意是:前 i 个箱子中最多可选出 f(i,j)
- 个箱子叠放起来,上面还可承受重量 j
  - $w_0[i]$ 为第 i 个箱子的本身重量
  - $w_1[i]$为第 i 个箱子可承受的重量
- ● $f(i,j)=\max\{f(i-1,j+w_0[i])+1 \mid w_1[i]\geq j, f(i-1,j)\}$
- ●
- ● 内存？
- ● 如果要求给出策略，内存如何优化？

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- 树的染色

- 给定 n 个结点的有根树 T，要求对其中每一个结点染色（颜色编号 1 到 m）。
- 1）要求相邻结点颜色不同
- 2）如果存在多个（至少2个）与第 i 号结点相邻的结点的颜色都是 j 则需要罚金（penalty）Pij
- 求最少罚金？

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- 第一想法：
  - 状态，对于每一个结点 i
  - i 的颜色是 j
  - i 的子节点拥有的颜色集 s
- 太多了！对于结点 i 和颜色 j，我们只需要其所有子节点的子节点是否有颜色 j。
- 第二想法：
  - 状态，对于每一个结点 i
  - i 的颜色是 j
  - i 的所有子节点的颜色都不是 k
- 转移想不清楚（想不清楚也是动态规划考题的一个重要潜在考点）

2-减少每个状态决策数的基本策略
2.3-合理组织状态

- 第二想法：
  - 状态，对于每一个结点 i
  - i 的颜色是 j
  - i 的所有子节点的颜色都不是 k
- 转移想不清楚

- 第三想法：
  - 状态，对于每一个结点 i
  - i 的颜色是 j
  - i 的父节点颜色是 k
- 怎么转移？

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- 细化状态：（~~“我有新名字啦：分布式DP”~~）
  - 弊端：
    - 状态数增加
    - 希望不要增加太多
  - 可能的优点：
    - 减少转移

- 目标：状态数x转移 更小

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- 考虑如下的转移方程
- 下标0<=i<=n，0<=j<=m
- 边界条件：
  - f(i,j) = 0 : i=0 or j=0
- 已知转移为：
  - $f(i,j) = \min_{1<=k<=i}$
    - { $f(i-k,j-k)+c_1$ | condition one,
    - $f(i-k,j)+c_2$ | condition two,
    - $f(i,j-1)+c_3$ | condition three }.

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- $f(i,j) = \min_{1<=k<=i} \{f(i-k,j-k)+c_1, f(i-k,j)+c_2, f(i,j-1)+c_3\}$

- 思考：（fix i and j）
- 如果f(i,j)的最优转移从第一部分过来，即
  - $f(i,j) = f(i-k^*,j-k^*) + c_1$
- 且如果令$f_1(i-1,j-1)$是强制(i-1,j-1)从第一部分转移的情况下的最优方案，则
  - $f(i,j) = f_1(i-1,j-1)+c_1$

- 维护$f_1(i,j)$, $f_2(i,j)$, $f_3(i,j)$和f(i,j)
- 状态规模$O(n^2)$，转移O(1)

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- Keep In Touch

- 给定DAG　G=(V,E)，其中|V|<=50
- 有一些合法的三元状态，且再指定其中一个状态s

- 问从s出发走下去的方案数

- （BestCoder Round #86）

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- Keep In Touch

- F[i][j][k] 从状态(i,j,k)出发的方案数
- 枚举所有下一状态(i',j',k')

- F_1[i'][j][k] 从某个状态(?,j,k)出发
- 且?首先抵达i'
- F_2[i'][j'][k] 从某个状态(?,*,k)出发
- 且?首先抵达i', *首先抵达j'
- 状态 O(n^3) 转移 O(n)

2-减少每个状态决策数的基本策略
2.4-细化状态转移

- 细化状态转移有的时候更多的是为了简化代码
- 插头类动态规划

- 最简单的插头状态：
  - m 个插头
  - 搜索一行的所有可能变化，得到两行之间的转移

- 一般常见的插头状态：
  - m+1 个插头
  - 搜索 (i,j) 一个格子的情况

3-减少状态转移时间的基本策略

- (3-1)、减少决策时间
  - 第一目标：对于每一个可能的前一状态，O(1)
- (3-2)、减少计算递推式的时间
  - 第二目标：快速考虑多个前置状态

3-减少状态转移时间的基本策略
3.1、减少决策时间

- LOSTCITY（不说了）
- ● 现给出一张单词表、特定的语法规则和一篇文章:
- ● 文章和单词表中只含 26 个小写英文字母 a...z。单词表中的单词只有名词,动词和辅词这三种词性,
- ● 且相同词性的单词互不相同。单词的长度均不超过 20。语法规则可简述为:
  - ● 名词短语:任意个辅词前缀接上一个名词;
  - ● 动词短语:任意个辅词前缀接上一个动词;
- ● 句子:以名词短语开头,名词短语与动词短语相间连接而成。
- ● 文章的长度不超过 1000。且已知文章是由有限个句子组成的,句子只包含有限个单词。将这篇文章划分成最少的句子,在此前提之下,要求划分出的单词数最少。

3-减少状态转移时间的基本策略
3.1、减少决策时间

- 状态：
  - Fv(i)：“前i个字符划分为以动词为结尾，且在i<M的时候允许带任意个辅词后缀”的最优分解方案下划分的句子数与单词数.
  - Fn(i)：“前i个字符划分为以名词为结尾，且在i<M的时候允许带任意个辅词后缀”的最优分解方案下划分的句子数与单词数.
- 注意，Fv(i)与Fn(i)存储的都是二元组(*,*)

- 如何快速决策[j+1,i]是不是动词，名词，辅词？
- j=1,2,3,...,i-1

3-减少状态转移时间的基本策略
3.2、减少计算递推式的时间

- 常见技巧二则：
- （1）利用数据结构
  - 快速区间计算 - 区间最值等
- （2）目标图形化（找到单调性）

3-减少状态转移时间的基本策略
3.2、减少计算递推式的时间

最大平均值问题

- 读入一列正数，$a_1, a_2, \ldots, a_N$，以及数F
  - 求一段长度大于等于F且**平均值最大**的子串

- 定义若$i \le j$，$ave(i, j) = (a_i + \ldots + a_j) / (j-i+1)$
- 目标：$Max\{ave(a, b) \mid a \le b-F+1\}$
- 范围：$F \le N \le 100\ 000$

- 例如N=4的序列中，F=2
  - 2, 5, 2, 5
  - ave(2, 4) = (5 + 2 + 5) / 3 = 4最大

3-减少状态转移时间的基本策略
3.2、减少计算递推式的时间

- 设部分和序列S_i为{a_i}前i项和，$S_0=0$
- ave(i, j) = [S_j - S_{i-1}] /[j - (i-1)]
- 过两点的直线：$P_{i-1}(i-1, S_{i-1})$，$P_j(j, S_j)$

- 问题转化：
- 平面上已知N+1个点，$P_i(i, S_i)$，$0 \le i \le N$
- 求横向距离大于等于F的两点连线的最大斜率

4-寻求转移过程的相关性（整体上减少转移时间）

- （4-1）重新整理状态
- （4-2）有周期性的转移 - F[s]与F[s+t]的转移相似
- （4-3）局部卷积形式

- 更多？
- 观察转移式的特殊性。

4-寻求转移过程的相关性

4.1、重新整理状态

- 区间dp -> 前缀dp
  - F[i][j] 考虑所有 F[i'][j'] ，i<=i'<=j'<=j
  - 前缀化： G[i] 对应 F[1][i]

- 树上dp -> 区间dp -> 前缀dp
  - 树的前序遍历 与 后序遍历

4-寻求转移过程的相关性
4.1、重新整理状态

- 树上背包

- n个物品的01背包问题，每一个物品对应树T上一个结点

- 一个结点x对应的物品取了
- 　则其父节点对应的物品一定也取了

- 经典树形DP，怎么做？
- 考虑树的DFS序列，等价于？区间动态规划！

4-寻求转移过程的相关性
4.1、重新整理状态

- 树上背包

- 考虑树的DFS序列，等价于？区间动态规划！

- 区间动态规划：小区间汇聚成大区间
- 前缀动态规划：不断扩大一个区间
  - 结点x一旦被选中，则可以尝试选择x子树内的结点
  - 否则，子树内不能用
  - 后序遍历中如何？

- ……a……b……cA……d……e……fB……g……h……iC

4-寻求转移过程的相关性
4.2、有周期性的转移

- 幼儿园的游戏

描述

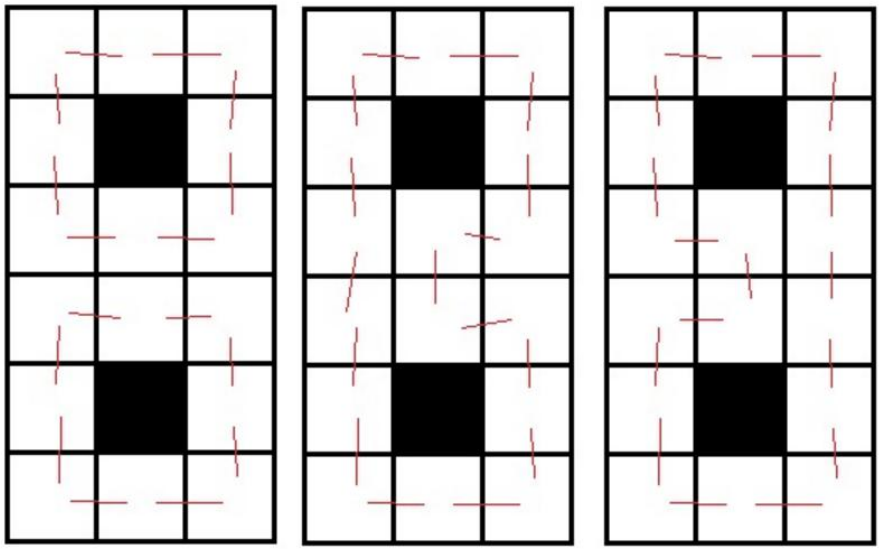
公元1770年，纪昀先生途经进香河，与百姓谈说人世，谈说友善。他所告诉人们的为友之道，流传至今，深深影响着当地的人们，甚至是幼儿园的孩童。

一天，多诺达新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，他们融入在了这个矩形队列之中，与小朋友们欢乐地唱着歌。“手拉手，我们永远都是好朋友！”于是老师要求每一位小朋友都和四周(即上下左右四个方向)的任意两个小朋友牵手。

作为幼儿园里最聪明的小朋友，你马上意识到这并不是单纯的游戏，而是一个非常有意义的问题。你非常希望知道一共有多少种不同的牵手方案，满足每一位小朋友都可以和四周恰好两位小朋友牵手。当然，每一位小朋友都只能和小朋友牵手，不能去和老师牵手。任何一位小朋友都不允许自己和自己牵手(即左手拉右手)。

你所需要知道的只是方案个数，因为答案可能太大，所以需要对1000000007取模。

4-寻求转移过程的相关性
4.2、有周期性的转移



对于100%的数据，N<=8,M<=2147483647，K<=100

4-寻求转移过程的相关性
4.2、有周期性的转移

- 插头DP？（... as an independent topic ...）

- 注意到k <= 100，如果k=0：
  - 将一行的转移视作整体
  - 转移具有周期性
- 没有障碍的时候，相邻行的转移可以表述为转移矩阵A
  - 时间复杂度：$O(2^{24}k\log n)$

- 如何做到更快？
  - 最终答案对应 $A^{n_1}B_1A^{n_2}B_2...A^{n_k}B_kx$

4-寻求转移过程的相关性
4.2、有周期性的转移

- 部分背包

- F[i][j] 由 F[i-1][j-v[i]], F[i-1][j-2v[i]], ..., F[i-1][j-t[i]v[i]] 转移来
- 对于所有 j ，在 mod v[i] 的意义下分组

- 特殊情况（ v[i]=1 ）
  - 单调队列
  - 对于每一个 j 求 j-t[i] 到 j-1 中的最优值

4-寻求转移过程的相关性
4.3、局部卷积形式

- Shell Necklace

Problem Description

Perhaps the sea's definition of a shell is the pearl. However, in my view, a shell necklace with n beautiful shells contains the most sincere feeling for my best lover Arrietty, but even that is not enough.

Suppose the shell necklace is a sequence of shells (not a chain end to end). Considering i continuous shells in the shell necklace, I know that there exist different schemes to decorate the i shells together with one declaration of love.

I want to decorate all the shells with some declarations of love and decorate each shell just one time. As a problem, I want to know the total number of schemes.

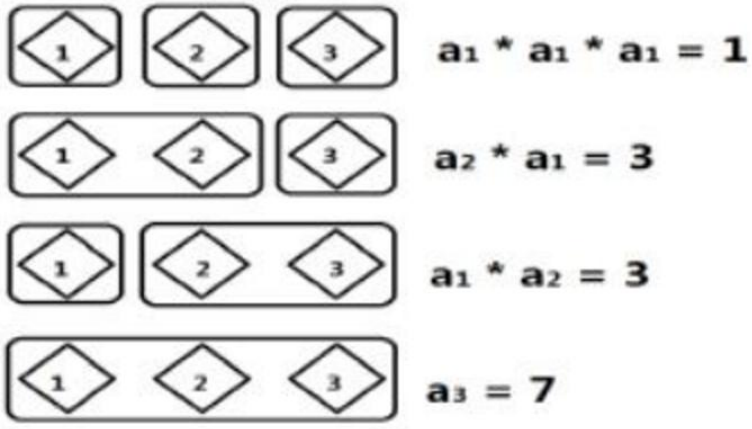


Figure 1: Hint for Sample.

4-寻求转移过程的相关性
4.3、局部卷积形式

- $F(i) = Sum_{1<=j<i} \{F[j] * A[i-j]\}$

- 不妨考虑 $N = 2^k$ 是2的幂，并记 T(N) 为用时。
  - 假设我们已经处理好了前一半的F(i)
    - T(N/2)
  - 将前一半做卷积，并分配给所有目标位置
    - 将 $F(u) * A(v)$ 的值赋予 $F_0(u+v)$
    - O(NlogN)
  - 对于 i > N/2，考虑 $F(i) - F_0(i)$
    - F(i <= N/2) * A(j > N/2)
      - 均已知，O(NlogN)
    - F(i > N/2) * A(j <= N/2)
      - 问题还原为 N/2 规模的等价问题，T(N/2)
- 分析时间复杂度：$T(N) = 2*T(N/2) + O(NlogN) = O(Nlog^2N)$

今日训练安排

- Uvalive5133 Machine Works (WF2011)
- hdu5730 (2016 Multi-Univ Training Contest 1)
- hdu5519 (ACM/ICPC Shenyang 2015 onsite)
- hdu5951 (ACM/ICPC Shenyang 2016 onsite)
- hdu5807 (BestCoder Round #86)
- hdu6078 (2017 Multi-Univ Training Contest 4)
- BZOJ4601 (SDOI2016)
- BZOJ4910 (SDOI2017)

- **欢迎提问！**

~~插头类动态规划~~ 定向状态压缩

- 维护什么？
- 维护一行：（一行的什么？）
  - 一行的格子：是否使用了？连通性？相关性？
  - 一行的分界线：是否被跨越？
  - 一行
  - （什么是插头？）分界线是否被跨越+格子之间的连通性
- 细化转移！
- 维护与 (i,j) 以及之后的未访问格子 有关的格子

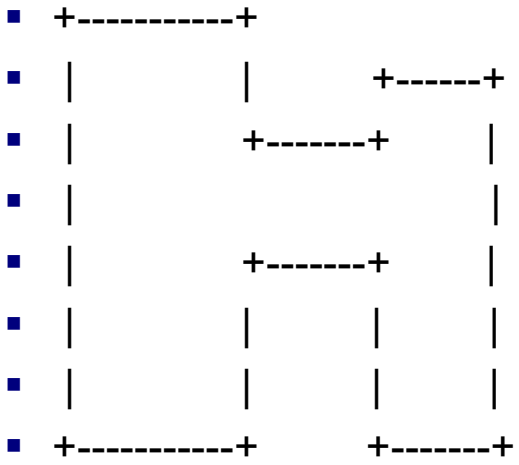
~~插头类动态规划~~ 定向状态压缩

- 维护什么？

- 维护与 (i,j) 以及之后的未访问格子 有关的格子！
- （的什么？）
  - 是否使用？连通关系？相对关系？
  - 对下面格子的影响？
  - （什么是插头？）对下面格子的影响+格子的连通关系

~~插头类动态规划~~ 定向状态压缩

- 一般格子，一般地图
- 格子的遍历（step by step）
  - “有关格子”的规模上界



  - 不同的遍历顺序会有不同的效果
- 维护相邻关系

~~插头类动态规划~~ 定向状态压缩

- 一般格子，一般地图
- 格子的遍历（step by step）
  - “有关格子”的规模上界
  - 不同的遍历顺序会有不同的效果
- 维护相邻关系
- “有关格子”与“未访问格子”之间有固定的关系
  - 相似性，规律性
  - 若没有？暴力依次处理每一对关系（例：足球）

~~插头类动态规划~~ 定向状态压缩

- 染色问题（求方案）：
  - 每一个格子如果染色了，则附近最多只能有2个格子染色了
  - 三染色问题
- 格子选取问题（求方案或带权最优解）：
  - 一定限制性选取格子
  - 每行每列每个对角线只能选取一个（两个，三个）
  - 任意3*3的格子只能选取最多5个（只能选取1，3，7个）
- 格子覆盖问题（方案或最优解）：
  - 放入1*2的骨牌覆盖
  - 放入大小4的骨牌（俄罗斯方块覆盖）

~~插头类动态规划~~ 定向状态压缩

- 路径覆盖（求方案或其他）：
  - 一个环覆盖
  - 任意多个环覆盖
  - 一条路径覆盖
  - 指定个数个环覆盖
  - 指定个数个路径覆盖
  - 指定长度的路径覆盖
- 特殊连通性（求方案或其他）：
  - 树覆盖（指定点的度，树的深度）：王字图覆盖，大字图覆盖
  - 二分图覆盖
  - 3-sat覆盖
  - 特殊性质的图覆盖：八字图覆盖，日（目）字图，田字图

~~插头类动态规划~~ 定向状态压缩

- 其它能想到的：
  - 无向图覆盖，使得s-t最小割不超过4
  - 随机漫步路径期望长度（每个格子不能经过2次）
  - 随机漫步抵达所有格子的期望用时（每个格子可以多次抵达）
  - 电阻覆盖，使得s-t电阻满足定值
  - 字符填充，使得不出现（出现）给定字典中的单词
  - 01填充，使得有要求多个洞
  - 黑白填充，使得最后的图上有QR code含指定信息
  - 填入一个双联通分量
  - 1*1和2*2的齿轮填充，要求no deadlock
  - 互相无法立刻吃子的中国象棋地图棋子位置分配
  - 计算半导体导电系数（每一个格子是一个概率p导电的半导体）
  - More...