动态规划

Dynamic Programming

方泓杰

OJ

bzoj: http://www.lydsy.com/JudgeOnline/
Uoj: http://uoj.ac/
Hdu: http://acm.hdu.edu.cn/
Poj: http://poj.org/
Luogu: http://www.luogu.org/
Loj: http://loj.ac/

目录
CONTENTS

PART 01

动态规划初探 & 基本dp

动态规划初探

什么是动态规划?

解决决策问题最优化的一种方法
在OI中应用广泛

什么样的问题可以用动态规划?

三个条件：相同子问题、无后效性、最优子结构

无后效性是啥?

每个子问题的决策不能对后面其他未解决的问题产生影响。

相同子问题，最优子结构又是啥?

子问题可以按照同样方法分割成更小的子问题
一个决策的子决策也是最优的

怎么学好动态规划?

积累常见状态，对问题进行合理划分
具有一定的转移技巧

还有啥?

没啥了……

Q&A

动态规划是动的吗？

· 是……吧？动态规划主要的思想就是利用目前已知结果推得下一步，然后重复这个步骤。从这个意义上是动的吧？

怎么让动态规划动起来？

· 认真听讲。

· 有听过的例题可以给其他人留些思考时间。

设计状态

· 一个好的状态设计是成功的四分之一。 ——我说的

对于动态规划分为许多类别，有一些共性的设计状态的方法，可以解决问题。当然在不同的题目中也要学会创新性的设计状态。

1. 序列dp

序列dp，就是和序列有关的dp问题，非常基本的有LCS、LIS等等。

一般来说，状态的选择有以下几种：

· 一个维度$[i]$表示前i个元素组成的状态。

· 一个维度$[i]$表示用了第i个元素，它与其他$1 \ldots i-1$的元素组成的状态。

· 两个维度$[i][0/1]$表示前i个元素，其中第i个元素用(1)或不用(0)所组成状态。

LIS

LIS：最长上升子序列

朴素的想法是用一个维度 $[i]$ 表示以第 i 个元素结尾的最长上升子序列。那么枚举前面所有值比它小的，选取dp值最大的即可。复杂度为 $\Theta(n^2)$ 。

怎么优化?

~~我会三维数据结构!~~

我会二分!

考虑记录当前长度为x的上升子序列的最后一个数的最小值为d_x。

显然， d_x单调上升，故二分出小于当前数的最大的d即可转移。

时间复杂度优化为$\Theta(n \log n)$

LCS

两个串之间的最长公共子序列
· or 子串?

$f_{i,j}$表示a串前i个，b串前j个字符所组成的最长公共子序列（子串）
子序列转移：

$$f_{i,j} = \begin{cases} \max(f_{i,j-1}, f_{i-1,j}) & (a_i \neq b_j) \\ f_{i-1,j-1} + 1 & (a_i = b_j) \end{cases}$$

子串转移：

$$f_{i,j} = \begin{cases} 0 & (a_i \neq b_j) \\ f_{i-1,j-1} + 1 & (a_i = b_j) \end{cases}$$

Noip2015 子串

有两个仅包含小写英文字母的字符串 A 和 B。现在要从字符串 A 中取出k个互不重叠的非空子串，然后把这k个子串按照其在字符串 A 中出现的顺序依次连接起来得到一个新的字符串,请问有多少种方案可以使得这个新串与字符串 B 相等?

注意：子串取出的位置不同也认为是不同的方案。

e.g. n=6, m=3, k=2, A=“aabaab”，B=“aab”
答案=7

数据范围$1 \le n \le 1000, 1 \le m \le 200, 1 \le k \le m$
TL：1s，ML：256Mb

Noip2015 子串

令$f_{i,j,d}$表示A串前i个字符中选择d个子串匹配了B串的前j个字符。

怎么转移？

我们并不知道某个字符在一个状态中是否选择，也就是说并不知道什么时候要开一个新子串。

怎么办？

改状态！

令$f_{i,j,d}$表示A串前i个字符中选择d个子串匹配了B串的前j个字符而且**必须选择A串的第i个字符**。

考虑转移？

Noip2015 子串

令$f_{i,j,d}$表示A串前i个字符中选择d个子串匹配了B串的前j个字符而且**必须选择A串的第i个字符**。

$$f_{i,j,d}=\begin{cases} f_{i-1,j-1,d}+\sum_{l=1}^{j-1} f_{i-1,l,d-1} & (A_i=B_j) \\ 0 & (A_i\neq B_j) \end{cases}$$

时间复杂度$\Theta(nm^3)$，观察转移形式，发现是前缀和形式，即可用前缀和优化为$\Theta(nm^2)$。

观察**空间是否超出**，发现超出，使用滚动数组。

前缀和 & 滚动数组

· 每个人可以有不同的实现方式

前缀和，即记录一个数组从第一项到第i项的值为s_i，那么要提取$[a, b]$的和，只需利用$s_b - s_{a-1}$即可。

滚动数组，即观察上题转移式，i维只和$i-1$维有关，且答案只与第n维有直接关系，故可以利用调整转移顺序（不建议）或开一个2维数组代替来实现。

```
int f[2][M][M];

int pre = 0, cur = 1;

for (int i=1; i<=n; ++i) {
    f[cur][...] = f[pre][...]
    swap(now, pre);
}
```

2. 区间dp

区间dp，就是与区间相关的dp问题。

一般来说，状态的选择有以下几种……
不对，基本就一种！$f_{l,r}$表示做完$[l,r]$区间的最优解。

怎么转移？将一个区间分成两个，找断点k，从$f_{i,k}$和$f_{k+1,j}$转移而来。
实现方式？记忆化搜索 or 按长度枚举

合并石子

有n堆石子，每堆有a_i个。按一定顺序合并，每次只能移动相邻两堆石子，合并费用为新一堆石子的数量。求合并成一堆最小花费。

数据范围$1 \leq n \leq 100$
TL: 1s, ML：256Mb

进阶版1：
环形合并石子。定义第一堆和最后一堆相邻，其他条件同上。

进阶版2：？？

合并石子 基本版

定义$f_{l,r}$表示合并完$[l,r]$后，所需要的最小花费。明显有

$$f_{l,r} = \min_{k=l}^{r-1}\left(f_{l,k} + f_{k+1,r}\right) + s_{l,r}$$

其中，$s_{l,r}$表示$[l,r]$的石子总数。
时间复杂度 $\Theta(n^3)$

记忆化搜索
实现方法可以多样化。

```
int f[M][M];
bool vis[M][M];

inline void solve(int l, int r) {
    if(l == r) return ...;
    if(vis[l][r]) return f[l][r];
    vis[l][r] = 1;
    int ret = INF;
    for (int k=l; k<r; ++k)
        ret = min(ret, solve(l, k) + solve(k+1, r));
    return f[l][r] = ret;
}
```

合并石子 进阶版1

将原序列复制一份接在后面，进行相同的dp。

下标	1	2	3	4	5	6	7	8	9	10	11	12
数据	1	2	3	4	5	6	1	2	3	4	5	6

例如上图，那么答案就是从
[1, 6], [2, 7], [3, 8], [4, 9], [5, 10], [6, 11]的区间选择了。

时间复杂度仍为$\Theta(n^3)$

3. 背包dp

背包dp，假设你们已经熟练掌握了01背包和完全背包的解法。

主要状态设计：$f_{i,j}$表示考虑到第i个物品，用了j的空间。
根据需要加入其他维度或限制条件。

多重背包？

多重背包

有n种物品，每种物品有个数c、价值v、重量w。背包有容量m。

将个数用二进制进行拆分，使得仅用$\log c$ 个物品就可以表示出任何个数。例如，有一个物品有14个，那么分解成1、2、4、7四种，即可组合表示出1-14所有个数。例如10=7+2+1等等。

那么就拆分成$n \log c$个物品做01背包，复杂度$\Theta(nm \log c)$。

Bzoj1578 股票市场

给定一个D天的S只股票价格矩阵，以及初始资金M；每次买股票只能买某个股票价格的整数倍，可以不花钱，约定获利不超过500000。最大化你的总获利。

数据范围$2 \leq S \leq 50, 2 \leq D \leq 10, 1 \leq M \leq 200000$
TL：1s，ML：256Mb

Bzoj1578 股票市场

首先每天过后剩余的钱尽量多是最优的。
因为连续买股票相当于每天买完第二天卖，然后再买。
所以对于D天，每天做一次70w的完全背包即可。
复杂度$O(D \times 700000)$

Noip2014 飞扬的小鸟

1. 界面是一个长为n，高为m的二维平面，其中有k个管道（忽略管道的宽度）。
2. 小鸟始终在游戏界面内移动。小鸟从游戏界面最左边任意整数高度位置出发，到达游戏界面最右边时，游戏完成。
3. 小鸟每个单位时间沿横坐标方向右移的距离为 1，竖直移动的距离由玩家控制。如果点击屏幕，小鸟就会上升一定高度 X ，每个单位时间可以点击多次，效果叠加；如果不点击屏幕，小鸟就会下降一定高度 Y。小鸟位于横坐标方向不同位置时，上升的高度X和下降的高度 Y 可能互不相同。
4. 小鸟高度为0或者小鸟碰到管道时，游戏失败。小鸟高度为m时，无法再上升。

现在，请你判断是否可以完成游戏。如果可以，输出最少点击屏幕数；否则，输出小鸟最多可以通过多少个管道缝隙。

数据范围$5 \le n \le 10000, 5 \le m \le 1000, 0 \le k \le n$
TL：1s，ML：512Mb

Noip2014 飞扬的小鸟

我们考虑一个简单的状态。$f_{i,j}$表示飞到(i,j)的最小步数。

$$f_{i,j}=\min\left(\min\left\{f_{i-1,j-k*X}+k\right\},f_{i-1,j+Y}\right)$$

这样复杂度为$\Theta(nm^2)$。

考虑完全背包的优化方法，我们是不是也可以不枚举k呢。

首先$f_{i-1,j+Y}$的转移可以放在最后转移，对该轮没有影响。

考虑$\min\left\{f_{i-1,j-k*X}+k\right\}$，这不就是完全背包吗?

把向上跳的操作看作重量为X，价值为1的物品。

利用完全背包相同的转移方式优化即可。

时间复杂度$\Theta(nm)$。

PART 02

树形dp

简单题1

给出一个以1号节点为根的树，求每个节点的子树大小。
数据范围$1 \leq n \leq 10^5$

树形dp一般通过dfs实现，一般先遍历儿子，得到儿子信息再更新父亲。

```
inline void dfs(int x, int fa) {
    sz[x] = 1;
    for (int i=head[x]; i; i=nxt[i]) {
        if(to[i] == fa) continue;
        dfs(to[i], x);
        sz[x] += sz[to[i]];
    }
}
```

简单题2

给出一个以1号节点为根带点权的树，求每个节点的子树权值和。
数据范围$1 \leq n \leq 10^5$

简单题2·改

点权改为边权？

转移方式稍加修改。

简单题3

给出一个带边权的树，求树上每个点所能到达的最远距离。
数据范围$1 \leq n \leq 10^5$

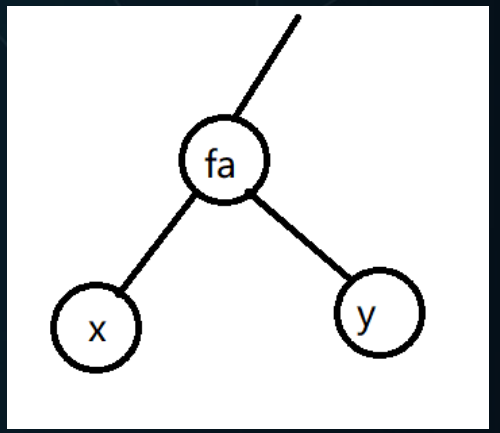
Up-down dp（树形dp的一种，简单来说就是先从下往上dp一次，再从上往下dp一次）

先按照正常顺序dp（用儿子的更新父亲），再用父亲的值更新儿子。

简单题3

令fi_x, se_x分别表示x往不同子树方向可延伸出的最长距离和次长距离。
顺便记录是往哪一个儿子的方向。（这个可以通过儿子更新父亲做到）

接下来才是答案的更新，比如我们要更新x的答案。
在x子树内的路径已经更新了，从fa上边下来的路径也已更新了，下面就剩下x的兄弟们，比如说从y来的路径没有更新。



而fa的子树信息已经取得了，用fa子树内到fa点的最长距离更新即可。
如果最长距离恰好为x所在子树，那么用次长距离即可。
最后更新完即为答案。复杂度$\Theta(n)$。

简单题4

给出一个无权树，求树的重心。
数据范围$1 \leq n \leq 10^5$

重心定义：找到一个点，其所有的子树中最大的子树节点数最少，那么这个点就是这棵树的重心，删去重心后，生成的多棵树尽可能平衡。

这是一个比较重要的题目，之后~~点分治（noip不考吧）~~啥的~~会经常用~~。

对每个节点记录子树节点个数和去掉这个点分成的若干块的块的最大值。比较后选取最小的即可。时间复杂度$\Theta(n)$。

简单题5

给出一个带边权树，求树的直径。
数据范围$1 \leq n \leq 10^5$

直径定义：树中任意两点间最大距离。

树的直径一定为某一点到其不同子树的最长链+次长链。
然后按照简单题3的做法仅记录最长、次长即可。
然后dp就行了。复杂度$\Theta(n)$。

简单题5

给出一个带边权树，求树的直径。
数据范围$1 \leq n \leq 10^5$

结论：从任意点开始，找离这个点最远的点x；再从x开始，找到离x最远的点y，那么$x \to y$就是树的直径。

证明？反证法。
具体可以参考这里。
复杂度$\Theta(n)$。

Luogu2015 二叉苹果树

有一棵苹果树，它是一棵二叉树，共 n 个节点，树节点编号为 $1\sim n$ 。编号为 1 的节点为树的根，边可理解为树的分枝，每个分支都长 着若干个苹果 现在要减去若干个分支，保留m个分支，使得这m个分支中的苹果数量最多。

数据范围$1 \leq n \leq 10^4, 0 \leq m \leq 50$
TL：1s，ML：256Mb
Oj#455

二叉苹果树

此题中，我们发现不能单纯用一维状态来表示，需要另开一维记录保留了多少枝条。那么即$f_{x,i}$表示节点x，留下i个枝条获得最多苹果个数。

考虑转移，共三种情况：

1. 全部给左儿子：$f_{lson,i-1}+w_{lson\to x}\to f_{x,j}$
2. 全部给右儿子：$f_{rson,i-1}+w_{rson\to x}\to f_{x,j}$
3. 分一些给左儿子，一些给右儿子：枚举给左儿子k条，那么有

$$f_{lson,k}+f_{rson,i-k-2}+w_{lson\to x}+w_{rson\to x}\to f_{x,j}$$

最后$f_{1,m}$即是答案。

时间复杂度$\Theta(nm^2)$

简单题6

给出一棵树，每条边边权为1，问两点之间路径长度为k的点对有多少个?

数据范围$1 \le n \le 5 \times 10^4, 0 \le k \le 500$

TL：1s，ML：256Mb

简单题6

很像up-down dp
但是可以处理得简单一点。
$f_{x,i}$表示以x为子树中离x距离为i的点的个数。
设y为x的儿子，那么可以通过$f_{y,i-1}$转移得到。
然后只要将父亲的贡献加进来即可。
从上往下处理，设x的父亲为p，那么处理x的时候，p已经处理过了。

$$f_{x,i} = f_{x,i} + f_{p,i-1} - f_{x,i-2}$$

注意更新顺序从k->1，这样不会重复更新。
复杂度$O(nk)$
来源codeforces 161D

简单题7

给出一颗树，有m种颜色，第k种颜色是特殊颜色，树上最多有x个特殊颜色点。
你需要把整个树染色，且保证特殊颜色节点以下条件：
1. 与其相连的不能有特殊颜色节点。
2. 与其相连的节点的颜色序号必须小于k。
问有多少种满足要求的树。

数据范围$1 \le n \le 10^5, 1 \le m \le 10^9, 1 \le k \le m, 1 \le x \le 10$
TL：2s，ML：256Mb

简单题7

实际上，颜色只有3类：①是特殊点，为k；②不是特殊点，但是可以和特殊点相连（小于k）；③不是特殊点，不能和特殊点相邻（大于k）。
那么就可以设计状态了。
$f_{x,i,0/1/2}$表示以x为根子树内有i个特殊点，x节点类型为1/2/3
转移直接背包合并即可。
将x节点的f'_x和x节点的儿子y的f_y合并，得到新的f_x

$$f_{x,i,0}=\sum f_{y,k,1}\times f'_{x,i-k,0}\,,f_{x,i,1}=\sum\left(f_{y,k,0}+f_{y,k,1}+f_{y,k,2}\right)\times f'_{x,i-k,1}$$

$$f_{x,i,2}=\sum\left(f_{y,k,1}+f_{y,k,2}\right)\times f'_{x,i-k,2}$$

时间复杂度$O(nk^2)$ 来源codeforces 855C

PART 03

概率与期望dp

期望是什么

有n个选择，每个选择收益为v_i，概率为p_i，且满足$\sum p_i = 1$，则期望获得收益$E = \sum p_i v_i$。
期望的加权和公式：$E(ax + by) = aE(x) + bE(y)$。

一个简单的例子（Uva12230 Crossing Rivers）
· 有个人准备坐速度为v的船过宽度为L河，船是不断处于往返两个河岸过程中的。当他到达河岸时，船随机在某一个位置。求期望过河时间。
最坏$\frac{3L}{v}$（刚走），最好$\frac{L}{v}$（刚到）
由于船的位置随机，所以时间在$\left[\frac{L}{v}, \frac{3L}{v}\right]$之间均匀随机，故期望为$\frac{2L}{v}$。

骰子

一个n面的均匀骰子，期望投多少次使得每面都有至少一次朝上。

数据范围$1 \leq n \leq 5 \times 10^6$
TL：1s，ML：256Mb

骰子

假设现在已经出现了x面，出现新的一面概率是$\frac{n-x}{n}$，也就是期望出现新的一面需要投$\frac{n}{n-x}$次。于是就是一个求和问题了。复杂度$\Theta(n)$。

机器人

在一个环上有n个格子，有m个w_i表示连续的一段命令，每个w_i表示机器人沿顺时针或者逆时针方向前进w_i格，已知机器人是从1号点出发的求最后机器人停在环上$[l,r]$区间的概率。

数据范围$1 \le n \le 200, 1 \le m \le 10^6$
TL：1s，ML：256Mb

机器人

$f_{i,j}$表示第i次操作后，机器人到j的概率。

$$f_{i-1,j+m} \rightarrow f_{i,j}$$
$$f_{i-1,j-m} \rightarrow f_{i,j}$$

用滚动数组优化即可，复杂度$\Theta(nm)$

Collecting Bugs

一个软件有s个子系统，会产生n种bug。某人一天发现一个bug，这个bug属于某种bug，发生在某个子系统中。求找到所有的n种bug，且每个子系统都找到bug，这样所要的天数的期望。

需要注意的是：bug的数量是无穷大的，所以发现一个bug，出现在某个子系统的概率是$\frac{1}{s}$，属于某种类型的概率是$\frac{1}{n}$。

数据范围$1 \leq n, s \leq 2000$。

TL：1s，ML：256Mb

Collecting Bugs

$f_{i,j}$表示已经找了i个bug，存在j个子系统中，剩余的期望天数。
那么很显然$f_{n,s}=0$，从后往前更新。
1. 找到的bug是旧的，存在的系统也是旧的。
概率：$p_1=\frac{i}{n}\times\frac{j}{s}$，还需要的期望$f_{i,j}$。
2. 找到的bug是新的，存在的系统是旧的。
概率：$p_2=\frac{n-i}{n}\times\frac{j}{s}$，还需要的期望$f_{i+1,j}$。
3. 找到的bug是旧的，存在的系统是新的。
概率：$p_3=\frac{i}{n}\times\frac{s-j}{s}$，还需要的期望$f_{i,j+1}$。
4. 找到的bug是新的，存在的系统也是新的。
概率：$p_4=\frac{n-i}{n}\times\frac{s-j}{s}$，还需要的期望$f_{i+1,j+1}$。

Collecting Bugs

根据期望的加权和公式，有

$$f_{i,j} = p_1 f_{i,j} + p_2 f_{i+1,j} + p_3 f_{i,j+1} + p_4 f_{i+1,j+1} + 1$$

移项后从后向前转移即可，最终答案就是$f_{0,0}$。复杂度$\Theta(\mathrm{ns})$。

Noip2016 换教室

给出一张无向图$G = (V, E)$作为学校地图。
小A选了n门课，这n门课依次上课。现在他可以更换m门课，每门课原来在点c_i上课，更换后在d_i上课，更换成功概率为p_i。从一个教室到另一个教室的体力消耗为两个教室之间的距离。求最小期望体力消耗。

数据范围$1 \leq n, m \leq 2000, 1 \leq V \leq 300, 1 \leq E \leq 90000$
TL：1s，ML：512Mb

Noip2016 换教室

· 肯定可以先floyd预处理出点对之间最短路。
· 然后考虑dp。一个简单的想法是 $f_{i,j}$ 表示前i个课程，选择了j个要换的，期望体力消耗最小值。
· 发现这样并不能知道前一个是否换过课程，无法确定最短路。
考虑更改状态表示：
· $f_{i,j,0/1}$表示前i个课程，选了j个要换的，其中最后一个是（1）否（0）换，期望消耗体力最小值。
那么转移就非常好想了。

$$f_{i-1,j,0} \to f_{i,j,0}, f_{i-1,j,1} \to f_{i,j,0}$$
$$f_{i-1,j-1,0} \to f_{i,j,1}, f_{i-1,j-1,1} \to f_{i,j,1}$$

至于转移的时候用期望定义算体力消耗即可。复杂度$\Theta(nm + V^3)$

Bzoj3566 概率充电器

有n个结点，相互之间通过$n-1$条边连接，形成一棵树。
第i个节点有a_i的概率直接充电，第i条边有b_i的概率导电。
求期望有电的结点数。
数据范围$1 \leq n \leq 10^6$

TL：1s，ML：256Mb

概率充电器

· 树形dp与概率期望dp的组合。
· 考虑如何设计状态：既符合树形dp的要求，也要包含一般概率期望dp。
· 按照树形dp的思路，有两种可以使得节点x充上电的方法：从儿子或父亲导电而来。

设计状态：

f_x表示不管父亲及祖先，x充上电的概率。

g_x表示x点父亲向x点导电的概率。

这样设计状态在转移的时候需要用到概率加法，比较麻烦。

考虑换一种状态设计？直接反面设计即可。

f_x表示不管父亲及祖先，x充不上电的概率。

g_x表示x点父亲不向x点导电的概率。

概率充电器

f_x表示不管父亲及祖先，x充不上电的概率。

g_x表示x点父亲不向x点导电的概率。

f_x转移比较简单，$f_x = (1 - a_x) \times \prod(f_{son} + (1 - f_{son}) \times (1 - b_{son \to x}))$

先用一遍dfs把f_x处理出来，然后用类似up-down dp的方法把父亲信息传递下去，得出g_x。

父亲有没有电可以通过x节点和x节点的爷爷传递，也可以通过x节点的兄弟传递。通过x节点兄弟或x节点的爷爷传递，仍然没有电的概率：

$$t = g_{fa} \times f_{fa} \div (f_x + (1 - f_x) \times (1 - b_{fa \to x})$$

那么$g_x = t + (1 - t) \times (1 - b_{fa \to x})$。

最后每个节点有电概率就是$1 - f_x \times g_x$。

时间复杂度$\Theta(n)$

PART 04

其他dp

6. 状态压缩dp

状态压缩：将一个复杂的状态利用二进制/三进制/k进制表示成一个数，成为dp的一维，从而进行dp。
二维的可能会稍稍快些，因为位运算。

状压dp有一个很明显的特点：数据范围小，一般最多不超过20。
（当然这也有可能是搜索题，因题而异）

位运算

以C++为例：
假如压缩成为了一个二进制状态S，那么取第i位的数：$(S \gg i)\&1$
第i位原来为0，现在改为1：$S|(1 \ll i)$。
以此类推……

班服

要开运动会了，神犇学校的n个班级要选班服，班服共有100种样式，编号1~100。现在每个班都挑出了一些样式待选，每个班最多有100个待选的样式。要求每个班最终选定一种样式作为班服，且该班的样式不能与其他班级的相同，求所有可能方案的总数，由于方案总数可能很大，所以要求输出mod 1000000007后的答案。

数据范围$1 \leq n \leq 10,1 \leq T \leq 10$，每班待选班服不超过100种。

TL：1s，ML：256Mb

班服

考虑到班级数量不多，但是班服数量多。
将班级进行状态压缩而不是服装。
$f_{i,S}$表示前i件衣服班级达到S这个状态。
显然有$f_{i-1,S} \to f_{i,S}$
而且$f_{i-1,j-\{k\}} \to f_{i,S}$，其中k是这个班可以穿的衣服。
那么复杂度$O(T \times 100 \times 2^n)$

Noip2016 愤怒的小鸟

有一架弹弓位于(0,0)处，每次 Kiana 可以用它向第一象限发射一只红色的小鸟，小鸟们的飞行轨迹均为形如 $y = ax^2 + bx$ 的曲线，其中a, b是 Kiana 指定的参数，且必须满足$a < 0$。当小鸟落回地面（即x轴）时，它就会瞬间消失。

在游戏的某个关卡里，平面的第一象限中有 n 只绿色的小猪，其中第 i 只小猪所在的坐标为 (x_i, y_i)。如果某只小鸟的飞行轨迹经过了 (x_i, y_i)，那么第 i 只小猪就会被消灭掉，同时小鸟将会沿着原先的轨迹继续飞行；如果一只小鸟的飞行轨迹没有经过 (x_i, y_i)，那么这只小鸟飞行的全过程就不会对第 i 只小猪产生任何影响。

现在 Kiana 想知道，至少需要发射多少只小鸟才能消灭所有的小猪。由于她不会算，所以希望由你告诉她。多组数据，数据组数T。

数据范围$1 \le T \le 15, 1 \le n \le 18$

TL：2s，ML：512Mb

Noip2016 愤怒的小鸟

先求出两两猪的解析式和能穿过其他哪些猪。
f_S表示覆盖状态为S，用的小鸟的最少数量。
那么枚举两只猪，设加入后覆盖状态变为S’，那么可以用$f_S + 1$来更新$f_{S'}$。时间复杂度$\Theta(Tn^2 2^n)$。

Bzoj4057 kingdoms

有一些王国陷入了一系列的经济危机。在很多年以前，他们私底下互相借了许多钱。现在，随着他们的负债被揭发，王国的崩溃不可避免地发生了……

现在有n个王国，对于每对王国A和B，A欠B的钱被记为d_AB（我们假设有$d_{BA} = -d_{AB}$成立）。如果一个王国入不敷出（即需要支付超过所能获得的钱），它就可能破产。每当一个王国破产，与它相关的所有债务关系都会被去除，无论是正是负。而王国们的破产不是一瞬间完成的，而是第一个王国破产后，接下来可能破产的王国再继续破产，直到剩下的王国经济都是稳定的。不同的结局将取决于谁先破产，尤其是有的结局只会留下一个王国。请你计算，对于每个王国，是否存在一种结局使得该王国是唯一的幸存者。

数据范围$1 \leq n \leq 20$。TL：1s，ML：256Mb

Bzoj4057 kingdoms

用f_S记录状态S是否能够达到，其中状态S代表所有王国破产/不破产。
对于每个状态，搜索接下来可以破产的王国，进行转移。
最后的答案要求的状态肯定就是除了某个王国，其他所有王国都破产。
复杂度$\Theta(n^2 2^n)$。

Codeforces gym 101343 J

给出n个序列，构造一个序列包含这n个串，求序列最小长度。

数据范围$1 \le n \le 15, 1 \le len \le 100$
TL：1s，ML：256Mb

Codeforces gym 101343 J

预处理出任意i子串后加j子串的最小长度。
也就是i的后缀和j的前缀重合的最小长度。
然后状压dp，设$f_{S,i}$表示S这个状态下，最后的串为第i个的最小长度。
转移只要寻找下一个接进来的串即可。

Codeforces 757D Felicity's Big Secret Revealed

给定一个01串，一个有效的n切割定义如下：一个横杠代表一次切割，第一条横杠前面的01串不算，最后一条横杠后面的01串不算，将两个横杠中的01串转化成十进制数字，假设这些数字的最大值是MAX且这些数字囊括了1-MAX的所有数字，则称为一次有效切割。求2~n+1次有效切割的切法。

比如 | 1 | 11 | 10 | 就是一个有效切割。

数据范围$1 \leq n \leq 75$。
TL：4s，ML：256Mb

Codeforces 757D Felicity's Big Secret Revealed

虽然数据范围为$1 \le n \le 75$，但是我们可以发现，$1 \le maxn \le 20$。
令$f_{i,S}$表示在i前面有切割，1-20中能表示出来的数有哪几个。
枚举下一个切割的位置j，$f_{j+1,S+\{p\}} \mathrel{+}= f_{i,S}$。
时间复杂度$O(2^{20} \times n^2)$
常数优秀就过啦！

Bzoj1076 奖励关

有n种宝物，k关游戏，每关游戏随机给出一种宝物，可捡可不捡。每种宝物有一个价值（有负数）。每个宝物有前提宝物列表，必须在前面的关卡取得列表宝物才能捡起这个宝物，求期望收益。

数据范围$1 \leq k \leq 100, 1 \leq n \leq 15.$
TL：1s，ML：256Mb

Bzoj1076 奖励关

观察宝物数量很少，意味着我们可以将宝物拥有数量的状态进行压缩。
用15位的二进制数来表示宝物拥有状态，我们很容易设计出dp状态。
$f_{i,s}$表示前i关，已有宝物状态为S的期望收益。
这里我们采取倒推，因为倒推的时候我们知道所有状态都是有效的，不会出现无效变成有效的情况。

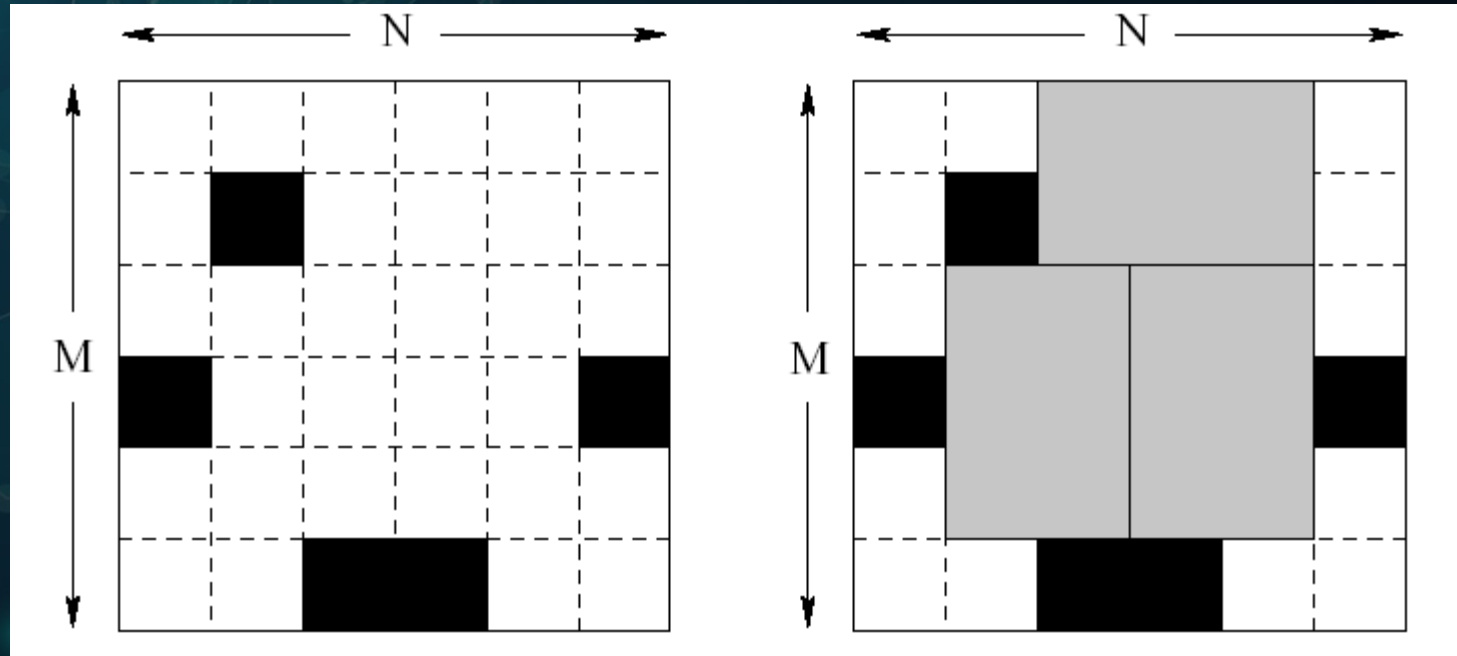
$$f_{i,s}=\frac{1}{n}\sum_{j=1}^{n}\max\left(f_{i+1,s+\{j\}}+p,f_{i+1,s}\right)$$

最后$f_{1,\emptyset}$就是答案。复杂度$\Theta(nk2^n)$

Poj1038 Bugs Integrated Inc.

在一个$n \times m$且有些格子损坏的网格上放最多的2×3芯片（芯片可以旋转），问最多放几个？如图最多3个。

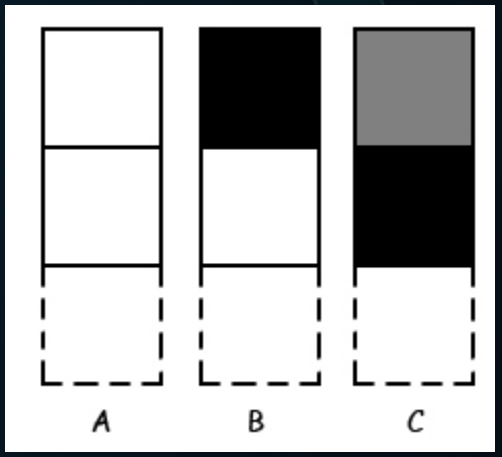
数据范围$1 \leq n \leq 150, 1 \leq m \leq 10$
TL：1s，ML：256Mb



Poj1038 Bugs Integrated Inc.

m很小，可以状压，那么如何进行状压?
观察到放置这列的时候最多是前两列有影响。所以记录前两列的信息即可。那么我们只要一个2^{2m}的状态。

其实我们可以进一步优化，对于每一行，仅有3种状态：
（坏/被占用：1；空：0；均可：？）
①00；②10；③?1
这样我们用三进制表示状态，状态总数优化到了3^m



Poj1038 Bugs Integrated Inc.

然后就分别考虑两个情况（横着放和竖着放）进行转移。
$f_{i,s}$表示到第i列，前两列状态为S的方案数。
如果有连续2行00，那么就可以放2*3的，如果连续3行10/00，那么就可以放3*2的，转移到下一个状态。当然也可以不摆放。
推荐用前面更新后面，这样可以方便写。

PART X

综合题选讲

以下内容包括但不限于今天所讲内容

背包

蛤布斯有n种商品，第i种物品的价格为a_i，价值为b_i。有m个人来向蛤布斯购买商品，每个人每种物品只能购买一个。第j个人有c_j的钱，他会不停选择一个能买得起的价格最高的商品买走（如果有多个则选择价值最高的）。你需要求出每个人购买的物品的价值和。

数据范围$1 \leq n, m \leq 100000, 1 \leq c_j \leq 10^{12}$

TL：1s，ML：256Mb

背包 Analysis

将物品从高到低排序。
对于每个人，先二分出目前最高能买的商品，再二分出能买的一段商品。
一直重复下去直到无法购买为止。

可以证明这个循环不超过$\log c_j$ 次。
时间复杂度$\Theta\left(m \log n \log c_j\right)$

Bzoj3043 incdec sequence

给定一个长度为n的数列，每次可以选择一个区间$[l,r]$，使这个区间内的数都加一或者都减一。问至少需要多少次操作才能使数列中的所有数都一样,并求出在保证最少次数的前提下,最终得到的数列有多少种。

数据范围$1 \leq n \leq 10^5, 1 \leq a_i \leq 2^{31}-1$
TL：1s，ML：256Mb

Bzoj3043 incdec sequence Analysis

差分序列，那么目标就是要让所有的数变为0。
我们在序列前增加一个位置假装也是差分的。
操作相当于每次选择一个地方+1，一个地方-1。
最后我们要求除了那两个假装差分的地方，其他全是0。
那么我们只要统计差分后序列正负数各自的和，就是最少次数了。
至于方案数，差距就是如果选择“假装在前面”的地方+1/-1，这样就可以统计了。复杂度$\Theta(n)$。

Bzoj4318 osu!

一共有n次操作，每次操作只有成功与失败之分，成功对应1，失败对应0，n次操作对应为1个长度为n的0/1串。在这个串中连续的x个1可以贡献x^3的分数，这x个1不能被其他连续的1所包含（也就是极长的一串1）。现在给出n，以及每个操作的成功率a_i，请你输出期望分数，输出四舍五入后保留1位小数。

数据范围$1 \leq n \leq 10^5$
TL：0.2s，ML：256Mb

Bzoj4318 osu!

考虑连续的1的个数不好维护，我们维护每个点的贡献。
考虑一个位置i，当i选了1，才会有贡献。如果前面有p个连续的1，贡献就是$(p+1)^3-p^3=3p^2+3p+1$
令i前面连续1的期望为x_i，连续1的个数的平方的期望为y_i
注意平方的期望不等于期望的平方！
显然$x_i=(x_{i-1}+1)a_i, y_i=(y_{i-1}+2x_i+1)a_i$
那么就知道每个位置的贡献了，时间复杂度$O(n)$。

参考

hzwer《树形dp入门》
n+e《dp总结》
fateice《概率与期望水题选讲》

Thanks