DP

吴清月

数字三角形

- 有一个由数字组成的三角形，第i行有i个数，一共有n行，你需要找出来一条从顶端到底端的路径，每次可以走向左边一个或者右边一个，你需要找出一条经过的数字权值最大的路径。
- $n \leq 1000$

- 来源：IOI1994压轴题
- 设f[i][j]表示到达第i行第j列的最大权值，直接DP。

最长上升子序列

- 给你一个长度为n的序列A，问你最长上升子序列长度。
- $n \leq 1000, n \leq 100000$

- 模板题，大家都会吧？
- 设f_i表示以i结尾的最长上升子序列的长度。
- 直接转移是$O(n^2)$的，二分或树状数组优化到$O(n \log n)$

最长公共子序列

- 给你两个序列，问这两个序列的最长公共子序列。
- $n, m \leq 3000$

- 设f[i][j]表示第一个序列的前i个和第二个序列的前j个元素的最长公共子序列的长度。
- 然后如果a[i]=b[j]那么f[i][j]=f[i-1][j-1]+1，否则f[i][j]=max(f[i-1][j],f[i][j-1])。

背包问题

- n种物品，每一个物品有一个价值v和一个体积w，你有一个大小为m的背包，你需要装进去最大价值的物品。
- $n, m \leq 1000$

- 01背包：每一种物品只有一个。NOIP2005采药
  - 设f[i][j]表示前i种物品，背包容量为j，则f[i][j]=f[i-1][j-w[i]]+v[i]
- 完全背包：每一种物品有无限个。洛谷1616疯狂的采药
  - 还是设f[i][j]，这时候f[i][j]=f[i][j-w[i]]+v[i]。
- 多重背包：每一种物品有有限个。
  - 方法1：二进制拆分，一种物品按照个数拆成log个。时间复杂度$O(nm\log n)$。
  - 方法2：单调队列。时间复杂度$O(nm)$。

背包问题

- **二维费用背包：每一种物品有两个重量。**
  - 多加一维表示多出来的那一个重量。
- **分组背包：每一组内部只能选择一个。**
  - 设f[i][j]表示前i组，然后枚举组内的物品进行转移。
- **有限制的背包：每一个物品可能需要先选择另一个物品才能选。NOIP2006金明的预算方案**
  - 树形DP，如果方案数较少可以枚举方案。

NOI1995 合并石子

- 有n堆石子围成一个圈，每次你可以选择把两堆相邻的石子合并，并且得到两堆石子的大小之和的分数。
- 求最小得分和最大得分。
- $n \le 100$

- 区间DP，设f[i][j]表示将从i到j的石子合并成一堆的最小/最大得分。
- 然后枚举最后一次合并的情况进行转移，时间复杂度$O(n^3)$。
- 类似的题：NOIP2006能量项链

SCOI2005 互不侵犯

- 在 $n \times n$ 的棋盘里面放K个国王，使他们互不攻击，共有多少种摆放方案。国王能攻击到它周围的八个格子。
- $1 \leq n \leq 9$

SCOI2005 互不侵犯

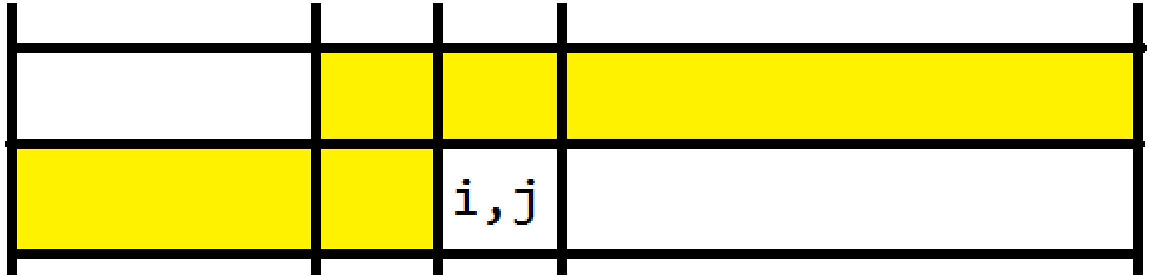
- 状压DP入门。
- 设f[i][j][s]表示放了前i行，一共放了j个，第i行放的位置集合是s的方案数。
- 然后枚举下一行放的状态t进行转移。
- 转移合法当且仅当t&(t<<1),t&(t>>1),s&t,s&(t<<1),s&(t>>1)都是0。
- 时间复杂度$O(3^n \cdot n^4)$，但是常数很小，可以直接过。

SCOI2005 互不侵犯

- 100分怎么了？我还能更优！
- 考虑优化上面那个算法。
- 首先我们发现，每一行的状态数量是有限的，因为王不能相邻。
- 所以可以预处理每一行的状态。
- 处理完之后我们发现每一行可以转移到的状态也是很有限的，所以还是可以预处理。

SCOI2005 互不侵犯

- 还有另一种优化思路。
- 直接枚举两个集合是$O(3^n)$的，这样太慢了。能不能优化一下呢？
- 答案是肯定的。
- 我们设f[i][j][k][s]表示当前到了第i行第j列，已经填了k个王，上一行的状态是s。这里的s指的是这样的一种状态：



- 这样我们只需要枚举(i,j)这一个格子的状态了，里面的3^n就变成了2^n。
- （当然在这道题里面不一定会更快）

NOI2001 炮兵布阵

- 在$n \times m$的棋盘放若干个炮兵，有的位置不能放，每一个炮兵的攻击范围如下：

P	P	H	P	H	H	P	P
P	H	P	H	P	H	P	P
P	P	P	H	H	H	P	H
H	P	H	P	P	P	P	H
H	P	P	P	P	H	P	H
H	P	P	H	P	H	H	P
H	H	H	P	P	P	P	H

- 你需要放置最多的炮兵。问最多能放多少。
- $n \leq 100, m \leq 10$。

NOI2001 炮兵布阵

- 首先对m状压。
- 设f[i][s1][s2]表示到了第i行，这一行放的状态是s1，上一行放的状态是s2的方案数。
- 然后枚举下一行放的情况，判断是否合法进行转移。
- 复杂度是$O(n8^m)$，显然过不了。
- 但是注意到每一行的合法状态非常有限，因为相邻两个位置至少差3。
- 经过爆搜发现其实合法的状态只有70不到……
- 所以复杂度变成了$O(70^3n)$，可以通过本题。

九省联考2018 一双木棋

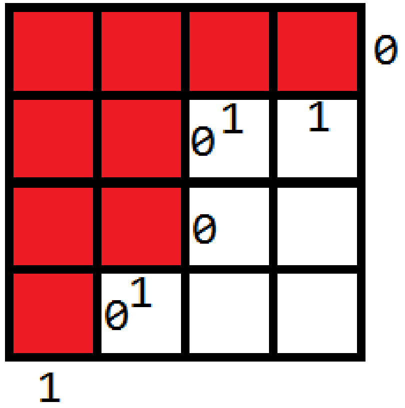
- 有一个$n \times m$的棋盘，每一个位置有一个$A_{i,j}$和一个$B_{i,j}$。
- 双方一黑一白轮流在上面落子。一个位置能够落子，当且仅当这个位置左上两个位置都已经有了棋子。
- 先手得分是所有黑棋位置的$A_{i,j}$之和，后手得分是所有白棋位置的$B_{i,j}$之和，双方都希望自己的得分减对方的得分最大。
- 问最优策略下最后先手得分减后手得分的结果。
- $n, m \le 10$

九省联考2018 一双木棋

- 当时考的时候靠特判拿了40分……
- 我们考虑设f[s]表示从状态s出发，最后先手-后手的得分。
- 对于转移，我们考虑枚举哪些位置可以落子，假设落子后能够到达的所有状态是t，那么f[s]=max(A[i][j]+f[t])（黑棋先）或f[s]=min(f[t]-B[i][j])（白棋先）。
- 这样我们就有了转移的思路了。
- 可是要怎么记录状态呢？

九省联考2018 一双木棋

- 一个合法方案对应一条从左下走到右上的折线。
- 在这条折线上，我们用0代表向上，1代表向右。
- 比如下面这种状态：



- 就对应了10100110这样一个二进制串。
- 这样对于$n \times m$的网格我们就能用一个长度为$n+m$的二进制串表示。
- 转移就是将一个01变成10，是$O(n+m)$的，时间复杂度$O\left((n+m)2^{n+m}\right)$

BZOJ1026 windy数

- 我们定义不含前导零且相邻两个数字之差至少为2的正整数被称为windy数。求$[A,B]$之间的windy数的个数。
- $1 \le A \le B \le 2 \cdot 10^9$

BZOJ1026 windy数

- 数位DP的特点：
  - 超大的数据范围（10^9起步，甚至可能10^{400}）
  - 在数位上搞事情（各个数位之和，相邻数位）
- 数位DP入门。
- 由于可以差分，所以我们只需要考虑求$[1,n]$之间的数的个数。
- 我们把一个数字从高位到低位写下来，然后在数位上DP。
- 设f[i][j]表示到了从高到低第i位，这一位为j的方案数。
- 转移就枚举下一位填什么。
- 似乎忽略了什么？

BZOJ1026 windy数

- 下一位不是想填啥就填啥的。
- 比如n=23456，那么如果你前两位是23，第三位就不能超过4，但是如果前两位小于23，第三位就没有上界的限制。
- 所以我们加一维表示当前的位是否卡到了上界。
- 也就是f[i][j][0/1]表示前i位，这一位是j，是否卡到上界。
- 时间复杂度$O(100k)$，其中k是位数。

AHOI2009 同类分布

- 求$[a,b]$中各位数字之和能整除原数的数的个数。
- $1 \le a \le b \le 10^{18}$

AHOI2009 同类分布

- 还是数位DP。
- 首先枚举各位数字之和x是多少，这个最大是$9 \times 18 = 162$。
- 然后就可以设f[i][j][k][0/1]表示前i位，这个数字%x是j，各位数字之和是k，是否卡上界的方案数。
- 时间复杂度$O(162 \cdot 18 \cdot 162 \cdot 10 \cdot 10) = 47239200$。
- 我的代码不吸氧会T一个点。

CodeForces 908G

- 令$S(i)$表示把i中的所有数位上的数拿出来，从小到大排序后形成的新的数。比如$S(50394) = 3459$。
- 给你一个X，问你$\sum_{i=1}^{X} S(i)$。
- $X \leq 10^{700}$

CodeForces 908G 2700

- 我知道是数位DP，可是怎么DP呢？
- 对于每一个数（0到9），我们考虑算它的贡献。
- 换句话说就是看最后d这个数字在重排后排在第i位的方案数，然后乘10^i后加到答案上。
- 首先枚举一个d，然后设f[i][j][0/1]表示前i个数位，比d大的数已经有了j个，是否卡上界的方案数。
- 转移就枚举这一位填的数x，如果x>d那么就转移到f[i+1][j+1]，如果x<d那么就转移到f[i+1][j]。
- 最后让ans+=d*f[n][j][0/1]*10^j。
- 但是仍然有一个问题：对于相同的数位我们该怎么办呢？

CodeForces 908G 2700

- 对于相同的数位，我们钦定原来在前面的数位还在前面。
- 这样需要加一维0/1表示现在这一位是否填到了填d的位置。这样如果这一位是0那么枚举到d不需要让j，否则枚举到d就需要让j+1。
- 然后就可以转移了。由于我写的是大力特判所有情况所以代码长成这个样子：
- 复杂度$O(10\cdot 700\cdot 700\cdot 10)$

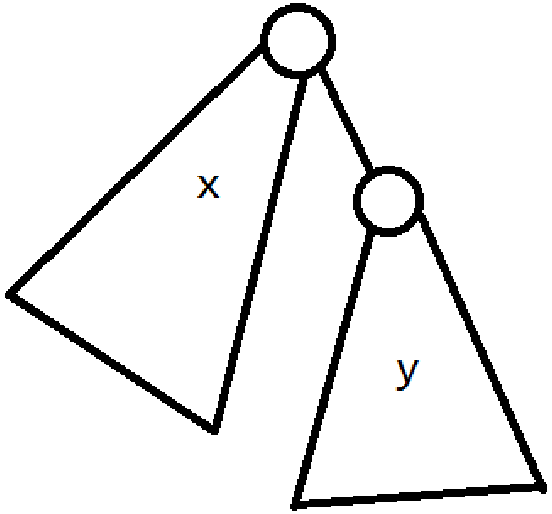
```cpp
for(int k=0;k<10;k++)
{
	if(k>digit[i+1])
	{
		if(k>x)
		{
			f[i+1][j+1][0][0]+=f[i][j][0][0];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
		}
		else if(k==x)
		{
			f[i+1][j][1][0]+=f[i][j][0][0];
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
		}
		else
		{
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j][1][0]+=f[i][j][1][0];
		}
	}
	else if(k==digit[i+1])
	{
		if(k>x)
		{
			f[i+1][j+1][0][0]+=f[i][j][0][0];
			f[i+1][j+1][0][1]+=f[i][j][0][1];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
			f[i+1][j+1][1][1]+=f[i][j][1][1];
		}
		else if(k==x)
		{
			f[i+1][j][1][0]+=f[i][j][0][0];
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j][1][1]+=f[i][j][0][1];
			f[i+1][j][0][1]+=f[i][j][0][1];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
			f[i+1][j+1][1][1]+=f[i][j][1][1];
		}
		else
		{
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j][0][1]+=f[i][j][0][1];
			f[i+1][j][1][0]+=f[i][j][1][0];
			f[i+1][j][1][1]+=f[i][j][1][1];
		}
	}
	else
	{
		if(k>x)
		{
			f[i+1][j+1][0][0]+=f[i][j][0][0];
			f[i+1][j+1][0][0]+=f[i][j][0][1];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
			f[i+1][j+1][1][0]+=f[i][j][1][1];
		}
		else if(k==x)
		{
			f[i+1][j][1][0]+=f[i][j][0][0];
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j][1][0]+=f[i][j][0][1];
			f[i+1][j][0][0]+=f[i][j][0][1];
			f[i+1][j+1][1][0]+=f[i][j][1][0];
			f[i+1][j+1][1][0]+=f[i][j][1][1];
		}
		else
		{
			f[i+1][j][0][0]+=f[i][j][0][0];
			f[i+1][j][0][0]+=f[i][j][0][1];
			f[i+1][j][1][0]+=f[i][j][1][0];
			f[i+1][j][1][0]+=f[i][j][1][1];
		}
	}
	f[i+1][j+1][0][0]%=MOD;
	f[i+1][j+1][0][1]%=MOD;
	f[i+1][j+1][1][0]%=MOD;
	f[i+1][j+1][1][1]%=MOD;
	f[i+1][j][0][0]%=MOD;
	f[i+1][j][0][1]%=MOD;
	f[i+1][j][1][0]%=MOD;
	f[i+1][j][1][1]%=MOD;
}
```

洛谷1273 有线电视网

- 有一棵n个点的树，每一个叶子节点有一个权值A_i，每一条边有一个权值C_i，你需要选出来一个叶子节点的集合，使得这些节点到根的路径上的所有边的边权-所有叶子节点的权值大于等于0，在此基础上使得选出来的叶子节点最多。
- $n \leq 3000$

洛谷1273 有线电视网

- （可算摆脱数位DP了）
- 首先设f[i][j]表示在i的子树中，选择j个叶子，点权-边权的最大值。
- 同时发现这个数组是单调的，所以最后找到第一个f[1][i]>=0就是答案。
- 那么怎么转移呢?
- 考虑枚举第一棵子树内选了多少个点，第二棵子树内选了多少个点：
- 那么f[node][x]+f[to][y]->f[node][x+y]。
- 这样转移是$O\left(\sum_{fa_i=fa_j} size_i \cdot size_j\right)$的。



洛谷1273 有线电视网

- 我们来分析一下时间复杂度。
- 抛开这个DP，我们考虑另一个时间复杂度相同的算法：
- for fa[i]=fa[j]
-   for x in i
-     for y in j
-       ……
- 这样也是$O\left(\sum_{fa_i=fa_j} size_i \cdot size_j\right)$的。
- 而我们发现，一个点对(x,y)只会在第一层循环枚举到LCA的时候被枚举到一次，所以这样的时间复杂度是$O(n^2)$的。

洛谷1273 有线电视网

- 总结：
- 有时候我们需要解决一类树上背包问题（也就是从树里面选择若干个点），这时候往往要设f[i][j]表示i的子树里面选择j个点。
- 转移的时候枚举两棵子树各自选了多少个点，这样的时间复杂度是$O(n^2)$的。

JSOI2016 最佳团体

- 有一棵n个点的树，每一个点有一个P_i和一个S_i。
- 你需要选出k个点，满足如果x被选中，那么x的父节点也要被选中。
- 你需要最大化你选出的所有点的$\frac{\sum P_i}{\sum S_i}$。
- $1 \leq k \leq n \leq 2500$

JSOI2016 最佳团体

- 就是一个随堂测试啊。
- 首先看到比值，二分一个答案v。
- 那么我们只需要看是否满足$\frac{\sum P_i}{\sum S_i} > v$，也就是是否满足$\sum(P_i - vS_i) > 0$
- 让每一个点的权值等于$P_i - vS_i$，然后做一个树形DP。
- 设f[i][j]表示i的子树选择j个点的最大权值。最后看f[1][k]是否>0。
- 时间复杂度$O(n^2 \log v)$。

HAOI2015 树上染色

- 给你一棵n个节点的树，你需要选出来k个节点染成黑色，剩下的染成白色，你的收益是所有黑点两两之间的距离+所有白点两两之间的距离。
- 求最大收益。
- $0 \le k \le n \le 2000$

HAOI2015 树上染色

- 直接两两距离肯定没法统计啊，考虑怎么统计答案。
- 对于一条边，它带来的收益=两边黑点的数量之积+两边白点的数量之积。
- 用上面的套路。设f[i][j]表示i这个子树，选择了j个黑点的最大收益。
- 枚举两个子树里面选择了几个点，然后算边的贡献即可。
- 时间复杂度$O(n^2)$。

BZOJ4987 Tree

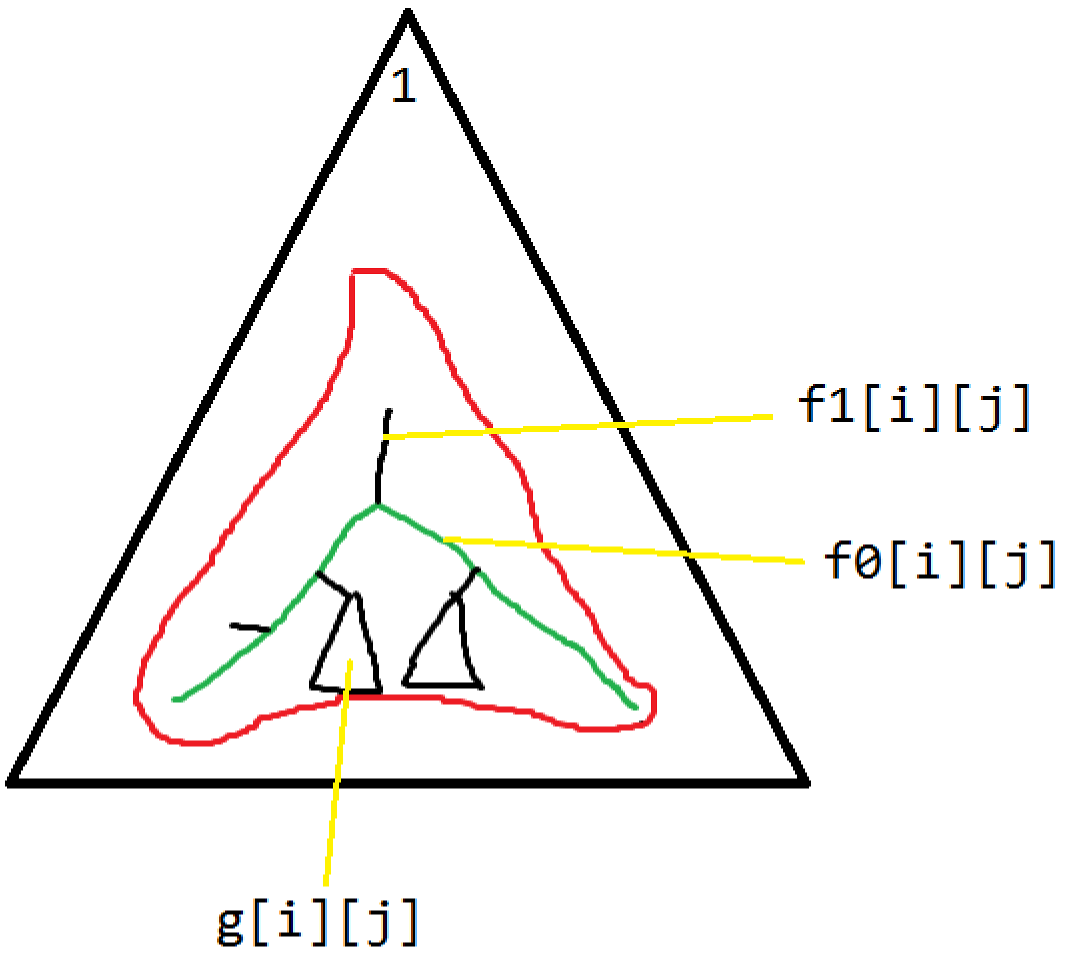
- 从前有棵树。
- 找出k个点$A_1, A_2, \ldots, A_k$，使得$\sum dis(A_i, A_{i+1})$最小。
- $1 \le k \le n \le 3000$

BZOJ4987 Tree

- （何时能摆脱树上背包啊）
- 这道题很难，如果你把它秒掉了那么你已经暴打讲课人了。
- 首先我们考虑$k = n$的情况。
- 如果$k = n$，那么答案就是遍历所有点的最小代价。
- 这时候除了起点到终点的那条路径，其余的所有路径都经过了两次。
- 所以这时候答案就是$\sum v_e -$直径长度。
- 有了这个，我们考虑更一般的情形。
- 如果$k \neq n$，那么问题就是选出来k个点，最小化
- $2 \times$(这k个点虚树的总边长)$-$(这k个点的直径长度)
- 这个可以用树形DP求出来。

BZOJ4987 Tree

- 首先这k个点一定连通。
- 设g[i][j]表示在i这棵子树里选出来j个点，虚树总边长的最小值。
- 设f0[i][j]表示在i这棵子树里选出来j个点，而且这j个点的虚树直径有一端是j，构成的虚树总边长×2－直径长度的最小值。
- 设f1[i][j]表示在i这棵子树里选出来j个点，而且这j个点的虚树直径两端都不是j，构成的虚树总边长×2－直径长度的最小值。



BZOJ3846 Hero Meet Devil

- 给你一个由AGCT组成的字符串S，你需要计算对于所有长度为m的也由AGCT组成的字符串T，与S的最长公共子序列长度为i的字符串的数量。
- 你需要对于所有的i求出答案。
- $|S| \leq 15, 1 \leq m \leq 1000$

BZOJ3846 Hero Meet Devil

- 看到$|S| \le 15$，先想到状压。
- 我们考虑当初是怎么求出来最长公共子序列的：
  - g[i][j]=g[i-1][j-1]+1
  - g[i][j]=max(g[i-1][j],g[i][j-1])
- 设f[i][s]表示长度为i，第i行的DP数组的样子是s的字符串的数量。
- 这个样子是s怎么定义呢？
- 我们不能直接压这个长度为15的DP数组。
- 但是我们注意到，这个DP数组差分后每一个位置的数就是0或者1了。
- 所以我们压缩差分数组，用一个长度为15的二进制数表示。
- 然后就可以转移了。时间复杂度$O(n2^m)$

CodeForces 1152D

- 给你一个n，把所有长度为$2n$的合法的括号序列拿出来插入到一棵Trie里面，然后问你这个Trie的最大匹配数量。
- $n \leq 1000$

CodeForces 1152D 2000

- 假设我们已经建出来了这棵字典树。
- 那么就树形DP好了，设f[i][0/1]表示i的子树里面，根节点匹配了/没有匹配的最大匹配数量。
- 现在的问题是由于点数太多，不能直接这样DP。
- 然而很多节点都是重复的。比如说，如果$n=4$，那么“()(”，“(()”这两个字符串下面链接的子树是一样的。
- 更进一步，每一个子树都可以表示成还需要插入i个左括号和j个右括号，所以直接设f[i][j][0/1]表示还需要插入i个左括号和j个右括号的子树里面，根节点匹配了/没有匹配的最大匹配数量。
- 时间复杂度$O(n^2)$。

CodeForces 1209E

- 你有一个$n \times m$的矩阵A，你可以把其中的任意列循环移位任意多次。
- 问最后所有行的最大值之和最大可以是多少。
- $n \leq 12, m \leq 2000$，40组测试。

CodeForces 1209E 2400

- rqy这一场被降智了没做出这道题，所以如果在座的各位秒掉了的话……
- 首先把所有列按照最大值排序，这样除去最大值前n大的列之外其余的列就都没用了。
- 证明：如果后面还要用到的话前面n列里面一定有一列没有用到，移动过来就会得到更优解。
- 然后设f[i][s]表示前i列已经确定了循环移位的情况，其中s这个集合里面已经确定了最大值的情况下答案的最大值。
- 然后我们枚举一个和s交集为空的集合t，表示第i+1行确定后新多出来的哪些行最大值确定了。然后枚举第i+1行的循环移位情况，进行统计。
- 时间复杂度$O(T3^n n^3)$，难以通过本题。

CodeForces 1209E 2400

- 考虑优化这个算法。
- 每次都枚举转移太浪费时间了，能不能预处理呢？
- 对于一个确定的集合t，怎么循环移位最优是可以预处理的，时间复杂度降到了$O(T3^n n)$，可以通过本题。
- （~~为什么rqy没做出来呢~~）

CodeForces 1188C

- 设一个序列b的美丽度为$\min(b_i - b_j)$。
- 给你一个序列a，问它的所有长度为k的子序列的美丽度之和。
- $2 \le k \le n \le 1000, 0 \le a_i \le 10^5$

CodeForces 1188C 2700

- ~~这真的是C题，一道难度是2700的C题，而且还和那一道2500的A题是同一场。~~
- 先排序，然后枚举美丽度v，然后我们尝试DP求出所有美丽度大于等于v的序列的个数，也就是相邻两个数只差必须大于等于v。
- 设f[i][j]表示长度为i，以j结尾的序列个数，转移枚举上一个元素：
- f[i][j]=f[i-1][k](a[k]<=a[j]-v)
- 记录一个前缀和就可以$O(nk)$了。
- 然后我们发现前面枚举美丽度是$O\left(\frac{v}{k}\right)$的，所以总复杂度变成了$O\left(\frac{v}{k}\cdot nk\right)=O(nv)$，然后就直接过掉了。
- （也不难啊，为什么难度是2700呢）

Q&A

- Thanks for listening!