试题泛做

——镇海中学顾嘉瀚

ACM/ICPC World Finals 2013

试题编号：2013-A　　**名称**：Self-Assembly

题目大意

给定n种分子，每个分子是一个正方形，每条边一个标识 00 或大写字母和加减号，00 不能与任何东西相连，A+可以与 A-相连，B+可以与 B-相连，依次类推。分子可以旋转，翻转，每种有无限个。问能不能组成一个无限的结构。

$n \leq 40000$

算法讨论

我们将每个分子看做一个点，将能够相连的点连一条边（每个面和其能贴合的点只连一条边）。我们可以发现，分子能不能组成无线结构等价于这张图中是否存在一个环（包括自环）。如果这张图存在一个环，那么我们将多个这个环在同一处拆开然后顺序相接，这样就成为一个无线结构。同时因为分子可以旋转、翻转，所以可以保证其它边不会贴合。然后如果这张图中不存在环，那么即使将所有能够连接的边连上，那么将所有能连的面连完之后，其仍然只能形成一个只有限个点的分子。

对于找环的问题，我们其实只需把每个面表示表示成一个点，那么就只有$m = 26 * 2 = 52$个点。对于每个面，我们将其与同一分子中其它面的对应面相连。这样可以发现，在新图中有环等价于原图中有环。然后在这张新图中暴力找环找环进行判断即可。

时空复杂度

时间复杂度$O(n + m^3)$　空间复杂度$O(n + m^2)$

试题编号：2013-D　　**名称**：Factors

题目大意

设$f(k) = k$的质因子排列方案数。求最小的k使得$f(k) = n$。

$n, k < 2^{63}$

算法讨论

我们可以发现，由于n和k有大小限制，所以能成为答案的k非常少。设$k = p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}$,那么$f(k) = \frac{(a_1+a_2+\cdots+a_n)!}{a_1!a_2!\dots a_n!}$。由于$f(k)$只和$k$的质因数次数有关，和质

因数大小无关，不妨令$a_1 \geq a_2 \geq \cdots \geq a_n > 0$，因为我们要使$k$尽量小，所以就要使质因数尽量小，显然我们可以得到$p_1 = 2$，$p_2 = 3$，…，$p_n$ =第n个质数。

然后直接枚举序列a，并计算出对应的k和$f(k)$。然后可以发现，在k满足小于2^{63}的情况下，有效的序列a不超过40000 个。那么我们在递归枚举时，如果当前乘积k大于等于2^{63}时直接跳出，否则就直接计算$f(k)$，判断是否满足条件。这样我们就预处理出了每个n对应的最小的k。最后在读入每个n后直接输出即可。

时空复杂度

时间复杂度$O(C)$　　空间复杂度$O(C)$

试题编号：2013-F　　**名称**：Low Power

题目大意

有n个机器，每个机器有2个芯片，每个芯片可以放k个电池。每个芯片能量是k个电池的能量的最小值。两个芯片的能量之差越小，这个机器就工作的越好。现在有$2nk$个电池，已知它们的能量，我们要把它们放在n个机器上的芯片上，使得所有机器的能量之差的最大值最小。

$2nk \leq 10^6, 1 \leq pi \leq 10^9$。

算法讨论

我们称每个芯片上电池的最小值成为有效值。

首先我们可以看做将$2nk$个电池分成n组，每组$2k$个，每组会分为两部分。显然每组的最小值必定是有效值，那么我们必然会将次小值分到到另一部分使得两部分的差最小。那么每组的有效值只有最小值和次小值。

然后将所有电池排序，我们可以发现一个性质，每组的有效值必定在序列中连续。证明如下：

1、如果两个同组有效值中间有不是有效值的数，那么我们将其后一个有效值与其交换，那么两个有效值的差显然更小，而那个非有效值在后移之后也不会成为新的有效值。

2、若四个有效值i,j,k,l满足i,j同组和k,l同组，$i \leq k \leq j \leq l$。那么我们将j和k的组别互换，由于$k-i \leq j-i$，$l-j \leq l-k$，所以$\max(k-i, l-j) \leq \max(j-i, l-k)$，所有解肯定不会变差。下面我们证明交换后$i,j,k,l$任然是有效值：若$k=j$那么将其交换显然没有关系。若$k<j$，那么因为在原来的组别中$k$为最小值，$l$为次小值，所以组中不可能存在数$x$满足$j \leq x < k$,所以$k$仍然是有效值。

3、若四个有效值i,j,k,l满足$i \le k \le l \le j$（其余同上），那么我们将也将j，k互换，那么$\max(k-i,l-j) \le j-i = \max(j-i,l-k)$，同样不会变差。而且四个数仍然是有效值的证明与上面相同。

然后我们二分答案，设当前答案为m，然后从1到n扫描。由于每组必须要有$2*k$个数，所以如果还有t组有效值没有确定，那么剩下的数个数必须大于$2kt$，所以扫描时每次选择第一对满足$a[i+1]-a[i] \le m$，然后判断能否找到n个有效值即可。

时空复杂度

时间复杂度$O(nk\log(p)+nk\log(nk))$　　空间复杂度$O(nk)$

试题编号：2013-H　　**名称**：Матрёшка

题目大意

俄罗斯套娃是一些从外到里大小递减的传统的俄罗斯木头玩偶组成的。当你打开一个俄罗斯套娃时，里面就会露出一个同样的俄罗斯套娃，再打开，就会再露出一个，不断重复。有n个玩偶放在在一行上，每个都有一个整数的大小，你需要重新组装套娃集，一个完好的套娃集内的玩偶大小是从 1 到某个数字m。在组装套娃集时，你必须遵守下列规则：

1.你只能将一个玩偶或者套娃集放入一个更大的玩偶中

2.你只能把相邻两个俄罗斯套娃组合在一起

3.已经被合并的玩偶是不能再重新拆出来的。

每打开一个玩偶所需时间为1，求将n个玩偶重新拼成一些完好的俄罗斯套娃的最小代价。

$1 \le n \le 500$，$1 \le$ 玩偶大小 ≤ 500

算法讨论

当一个区间$[i,j]$组成了一个套娃时，显然方式唯一。$f[i][j]$表示区间$[i,j]$组成一个套娃的最短时间（无论是否完整），$g[i][j]$表示区间$[i,j]$能否组成一个完整的套娃。设长度$l=j-i+1$，那么枚举中间点k。对于将区间$[i,k]$和区间$[k+1,j]$合并所需的时间，我们可以发现将$[i,j]$中所有套娃的大小排序，由于没有重复大小的套娃，这些套娃的顺序确定。那么考虑若它的一个前缀中的套娃都来自原先同一个大套娃，那么这些套娃都不会被拆开，所以找出最长的来自同一个套娃的前缀设长度为x，那么需要打开的套娃数量是$l-x$。那么转移方程为：

$$f[i][j]=\min(f[i][k]+f[k+1][j]+l-xk)\{i \le k < j\}$$

对于这个过程，如果直接排序时间为$O(n^3 \log n)$，可能无法通过。所以我们在做区间 DP 时，可以第一维枚举长度l，第二维枚举区间开头i，那么对前l个数排序，然后当做到$i+1$时，在序列中删去i，加入$i+l$，这样时间复杂度就为$O(n^3)$。当我们从i到$j-1$枚举k时，记录每个套娃来自哪一个大套娃。由于k增加时，套娃k从来自第二个套娃变成来自第一个套娃。这样我们考虑两个过程：

1、序列第一个套娃来自后一个大套娃，这样会使前缀长x不断递减，当套娃k的在序列中的位置p小于x，那么我们就使$x=p-1$。

2、序列第一个套娃来自前一个大套娃，这样会使前缀长x不断递增，这时套娃k的在序列中的位置p必然大于x，这时我们只要在每次套娃k的归属变化之后，不断判断序列中第$x+1$个套娃是否来自前一个套娃，如果是则使$x=x+1$，重复这个判断，由于不断递增这个过程只进行$O(l)$次。

最后我们用$h[i]$表示区间$[1,i]$都组成了完整的一组套娃所需的最少时间，显然转移方程为$h[i]=\min(h[j-1]+f[j][i])\{0<j<i$，且$g[j][i]=true\}$。那么$h[n]$就是最后的答案。

时空复杂度

时间复杂度$O(n^3)$　空间复杂度$O(n^2)$

试题编号：2013-J　　**名称**：Pollution Solution

题目大意

给定一个圆心在原点，直径在x轴上的半圆，以及一个简单多边形，问两者重叠的面积为多少。

$n \le 100$，$r \le 1000$

算法讨论

首先求出这个多边形的边与半圆交点的横坐标。然后按横坐标（包括$-r$和r）排序。将原多边形的边按横坐标分拆成线段，然后逐段计算。对于每一段，维护这一段中的线段从上至下的顺序。然后从上至下枚举。

首先对于每一条线段，编号为$2*i-1$的线段和编号为$2*i$的线段之间为多边形所包含的部分。那么我们所计算的面积应该是圆弧以下的多边形包含的面积。设一条线段或圆弧对应的面积为这一条曲线和x轴以及直线$x=x_1$和$x=x_2$所围成的面积。对于一条线段，设其两端的端点为(x_1,y_1)，(x_2,y_2)，那么其对应的面积$S=(x_2-x_1)*\frac{y_1+y_2}{2}$。对于一段从$x_1$到$x_2$的圆弧，其对应的面积：

$S = \int_{x_1}^{x_2} \sqrt{r^2 - x^2}\, dx = \frac{1}{2}(x\sqrt{r^2 - x^2} + r^2 \tan^{-1} \frac{x}{\sqrt{r^2 - x^2}}) \Big|_{x_1}^{x_2}$。

那么对于每一条在圆弧下的线段，设其标号为i，那么它对答案的贡献可以表示为$S_i * (-1)^{i+1}$，同时如果圆弧作为了一块区域的上边界，那么加上圆弧对应的面积。最后的答案即是每一段的面积和。

时空复杂度

时间复杂度$O(n^2)$　空间复杂度$O(n)$

ACM/ICPC World Finals 2012

试题编号：2012-B　　**名称**：Curvy Little Bottles

题目大意

有一条多项式曲线$f(x)$，将曲线的$[X_{low},X_{high}]$这段绕x轴旋转一周就能形成一个曲面。将这个曲面作为一个瓶子的侧面，再将X_{low}处封死，我们就能得到一个瓶子。接下来你要在这个瓶子上标记刻度，体积每隔inc标记一次，标记满八次后停止。求这个瓶子的容积以及每个标记的位置。

多项式每项次数为自然数，最高次数$n \leq 10$。

算法讨论

对于多项式积分：$\int \mathrm{f}(x)dx = \int \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \frac{a_i}{i+1} x^{i+1} + C$。

那么设$s(x) = \pi f(x)\text{^}2$。这可以直接将$s(x)$展开求值。然后对$s(x)$求$[X_{low},X_{high}]$范围内的定积分，计算出瓶子体积。对于每个标记，由于$s(x)$恒大于0，所以积分单调递增，所以我们可以直接二分高度h，然后对$s(x)$求$[X_{low},h]$范围内的定积分，与标记体积比较然后调整h范围直到一定精度即可。

时空复杂度

时间复杂度$\mathrm{O}(n^2 + \frac{nV}{inc} * log\frac{1}{eps})$ 空间复杂度$\mathrm{O}(n)$

试题编号：2012-C　　**名称**：Bus Tour

题目大意

给定一个n个点m条边的无向图，每条边有一个长度。点0为起点，点$n-1$为终点，点1到点$n-2$每处有一个客人，你需要设计一个接送客人的顺序，使得你的客车从点0出发，要到点$1\sim n-1$的每一个点去接客，然后送到点$n-1$，再将每个客人送到原来位置，最后返回点0，同时使得经过的路程最短。同时接送客人的顺序要满足先接到的前$\left\lfloor\frac{n-2}{2}\right\rfloor$个客人要先送到。

注意到达一个点后可以先不接客人，也可以先不放下客人。

$n \leq 22$，$m \geq 2$，时限$4s$

算法讨论

这道题由于时限较宽，可以使用状压 DP。首先我们可以预处理出每两个点的最短路$e[i][j]$。然后然后用i表示这$n-2$个客人是否被接到的状态，j表示当前车的位置，$f[i][j]$表示在这状态下的最短路程。方程表示为$f[i|2^{k-1}][k] = \min(f[i][j] + e[j][k])$，满足$j \in [0, n-1]$，$k \in [1, n-2]$。

然后我们以0为起点做一次 DP 用$g[i][j]$表示，然后以$n-1$为起点做一次 DP 用$h[i][j]$表示，然后设先接到的前$\left\lfloor\frac{n-2}{2}\right\rfloor$人表示的状态为$i$，同时枚举第一次接到前$\left\lfloor\frac{n-2}{2}\right\rfloor$个人后接到的人为$j$和第二次放下前$\left\lfloor\frac{n-2}{2}\right\rfloor$个人后放下的人为$k$，前一段路程是$S_1 = g[i|2^{k-1}][k] + h[(2^{n-2}-1)\text{^}i][k]$，后一段路程$S_2 = h[i|2^{k-1}][k] + g[(2^{n-2}-1)\text{^}i][k]$，那么最终答案$S = \min(S_1 + S_2)$。

时空复杂度

时间复杂度$O(2^n * n^2)$　空间复杂度$O(2^n * n)$

试题编号：2012-D　　**名称**：Fibonacci Words

题目大意

定义字符串$F(0) = '0', F(1) = '1', F(n) = F(n-1) + F(n-2)$，每次询问一个正整数$n$和字符串$p$，问$p$在$F(n)$中的出现次数。

$n \le 100$，p的长度$l \le 100000$

算法讨论

首先，我们思考$F(n)$构造的过程——$F(n) = F(n-1) + F(n-2)$，那么p在$F(n)$中出现有三种情况 1、p完全在$F(n-1)$中。2、p完全在$F(n-2)$中。3、p横跨$F(n-1)$和$F(n-2)$。显然对于前两种情况，我们可以直接递推得解。

首先设$L(n) = F(n)$的长度。可以发现，$L(n)$单调递增。对于第三种情况，设x为$L(x)$大于l的最小值$(x \le n)$。那么首先我们直接构造出$F(x)$和$F(x+1)$，并且直接用 kmp 算法计算p在$F(x)$和$F(x+1)$中的出现次数。显然$L(x) = \frac{\sqrt{5}}{5}\left(\left(\frac{1+\sqrt{5}}{2}\right)^x - \left(\frac{1-\sqrt{5}}{2}\right)^x\right)$。我们令$\phi = \frac{1+\sqrt{5}}{2}$，那么时间复杂度$O(\sum_0^x L(i)) = O(\sum_0^x \phi^x) = O(\frac{\phi^{x+1}-1}{\phi-1}) = O(l)$。

同时可以观察可到:

$F(x+2)=F(x+1)+F(x)$

$F(x+3)=F(x+2)+F(x+1)=F(x+1)+F(x)+F(x+1)$

$F(x+4)=F(x+3)+F(x+2)=F(x+1)+\cdots+F(x+1)+F(x+1)+F(x)$

$F(x+5)=F(x+4)+F(x+3)$

$=F(x+1)+\cdots+F(x)+F(x+1)+\cdots+F(x+1)$

……

由于$L(x)$和$L(x+1)$均不小于l，所以从$F(x+2)$开始中间横跨部分只与$F(x)$和$F(x+1)$有关，分三种情况：

1、$n=x+2$，横跨部分为$F(x+1)+F(x)$。

2、$n=x+2k(k>1)$，横跨部分为$F(x+1)+F(x+1)$。

3、$n=x+2k+1$（$k>0$），横跨部分为$F(x)+F(x+1)$。

那么我们可以用 kmp 算法计算出$F(x)$的后$l-1$个字符与$F(x+1)$前$l-1$个字符组成的字符串中p出现次数，以及前为$F(x+1)$后为$F(x)$和前为$F(x+1)$后为$F(x+1)$的情况。然后根据发现的规律直接递推计算即可。

时空复杂度

时间复杂度$O(l)$　　空间复杂度$O(l)$

试题编号：2012-E　　**名称**：infiltration

题目大意

给定一张有n个点的有向图，每两个点之间都会连一条有向边。问这张图的最小覆盖集的点数（点可以覆盖本身）。

$n\leq 75$

算法讨论

首先我们可以发现由于这张图非常密集，答案必然会非常小。我们可以证明答案小于$O(\log n)$：对于任意由x个点组成的子图，显然它有$\frac{x(x-1)}{2}+x=\frac{x(x+1)}{2}$条边（包括自环），那么由抽屉原理可知必然存在能够覆盖超过$\frac{x}{2}$个点（包括其本身）的节点。那么从n个点开始每次取能覆盖未覆盖的点最多的点，那么这样的操作每次使得未覆盖的点至少减少一半，那么次数就不会超过$O(\log n)$。

这样我们可以用搜索来解决这个问题，首先用上述的贪心方法先构造一个上界，那么我们就可以方便的进行最优性剪枝，即选择的节点个数超过这个上界就可以直接跳出。然后在判断是否完全覆盖时，要使用位运算优化，即使用两个 long

long 类型来表示是点被覆盖的状态。当前状态S用 1 表示未被覆盖，0 表示已被覆盖，每个点的覆盖集合T_i用 1 表示能够覆盖，0 表示不能覆盖。那么当我们选择点i时，那么我们就可以使$S = S\text{^}(S\&T_i)$，最后判断S是否等于0即可。这样就可以通过这道题目。

时空复杂度

时间复杂度$O(n^{\log n})$　　空间复杂度$O(n)$

试题编号：2012-K　　**名称**：Stacking Plates

题目大意

现在有n堆盘子，每堆最多h个盘子，每个盘子有一个正整数直径。在一堆盘子中，上面的盘子直径必须小于等于下面的盘子。有两种操作：

1、将一堆盘子上面的任意个盘子抬起，形成新的一堆。

2、在将一堆盘子放在另一堆盘子的堆顶。

求把所有盘子合并成一堆的最小操作次数。

$n, h \leq 50$

算法讨论

显然每堆盘子里大小相同的盘子不会被分开，所以可以直接缩成一个盘子。设剩下的盘子数为m，然后对盘子从小到大排序。假设开始时将所有的盘子全部拆开然后在组装起来，那么操作次数即为$m - n + m - 1$。

然后我们发现对于排序后的序列，对于那些大小相同的盘子，它们之间的顺序是可以互换的，而且如果相邻两个盘子来自同一堆，那么在开始拆开时就不用将其拆开，之后也不用组装，那么操作次数就减少了两次。

那么我们就可以用动态规划来解决这个问题。将序列分成 l 个部分，每个部分盘子大小相同。设$f[i][j]$表示在第i部分结尾可以为j时的相邻来自同一堆的盘子对数。所以当第 i 部分没有来自第 j 堆的盘子时，$f[i][j] = -\infty$。同时设$g[i][j]$则表示在第i部分开头为j时的对数，$s[i]$表示个$g[i][j]$中的最大值。那么显然$g[i][j] = \max(f[i][j] + 1, s)$。同时如果第$i$部分盘子数等于1，那么需要简单特殊判断。否则如果盘子数大于1，那么$f[i][j] = \max(g[i][k])$，且满足$k \neq j$。而且可以用前缀和优化使计算$f[i][j]$的过程做到$O(n)$。那么最终答案即为$2 * m - n - 1 - 2 * s[l]$。

同时我们可以用循环数组来对算法进行优化。那么每次在计算出第i部分的g数组后，扫描前一部分的所有盘子将f数组清空，然后再用第i部分的g数据更新出新的f数组。这样就只需操作$O(m)$次即可，最后的时间复杂度即为排序所用时间。

时空复杂度

时间复杂度$O(nh\log(nh))$　　空间复杂度$O(nh)$

试题编号：2012-L　　**名称**：Takeover Wars

题目大意

给定两个公司 A 和 B，两个公司分别有n和m个子公司，每个子公司有一个价值。一次操作有两种情况 1、将自己拥有的两个子公司合并，新公司为原公司价值和。2、用自己拥有的一个子公司吞并对方的一个价值比当前子公司小的子公司，同时自身价值不变。两个公司 A、B 轮流操作，从 A 开始，当一个公司没有子公司时算输。问最终那个公司会获胜。

$n,m \leq 100000$，题目保证两边不会相同价值的子公司出现。

算法讨论

首先非常显然的是，每次吞并必然是用自己最大的公司去吞并对方所能吞并的最大的公司。同时，如果己方吞并了对方一个公司且，对方也吞并本方的一个公司，那么显然损失少的那个公司是占优了。然后我们可以依次推出下两个结论：

1、每次吞并必然是吞并对方最大的子公司。有于题目保证两边不会相同价值的子公司出现，那么如果己方无法吞并对方最大的子公司，而去吞并较小的一个，那么对方下一次就一定可以吞并己方的最大子公司。这样显然是不合算。

2、每次合并必然是合并自己最大的两个子公司。因为注意到如果不是合并两个最大的子公司，那么必然是为了保护那个较小的子公司不被对方吞并。但是由 1 可得，对方吞并己方必定是吞并己方最大的子公司，不会吞并其余较小的公司，那么这样做就没有意义。

由此我们就可以发现，由于每次合并最大的两个子公司，那么每个公司下辖的子公司中只有一个是由合并产生。那么这个公司如果可以被对方吞并，那么这个公司最后必定会输，因为以后合并出来的公司必定比现在的公司小，那么依然可以被对方吞并。

这样的话，我们只需要枚举先手的第一步，之后的每一步就都可以唯一确定。因为如果先手第一步选择合并，那么后手最大值如果大于合并值，那么可以直接选择吞并，先手必败。如果后小于合并值，那么只有合并一种选择，之后先手同理。如果先手选择吞并对方最大值，那么后手新的最大值必然小于对方的最大值，那么只能选择合并，之后就同第一种后手的情况。那么最后只需判断两种情况中是否存在先手胜的情况即可。

时空复杂度

时间复杂度$O(n\log n + m\log m)$　　空间复杂度$O(n+m)$

ACM/ICPC World Finals 2011

试题编号：2011-A　　**名称**：To Add or to Multiply

题目大意

给定两个操作:1、加上一个数a。2、乘上一个数m。

求一个最短的操作序列，使得所有整数$x \in [p,q]$，依次进行操作后得到的值$y \in [r,s]$，若有多个最短找出字典序最小的一个。

$1 \le a,m,p,q,r,s \le 10^9$

算法讨论

设$y = kx + b$，由于y单调递增，所以$k * p + b \ge r, k * q + b \le s$。显然$k = m^n$，$b = a * w$。枚举$n$，然后得到$w$的取值范围。

对于每个n，可以使$w = C_n * m^n + C_{n-1} * m^{n-1} + \cdots \ldots + C_1 * m + C_0$，那么我们的操作序列就是:$C_n$A 1$M$ C_{n-1}A 1M …… C_2A 1M C_1A 1M C_0A。那么序列长度$l = \sum_{i=0}^{n} C_i$。由于A的字典序小于M，所以当序列长度相同时，越前面的C应该越大。若存在$C_i \ge m (i < n)$，那么使之减少m，然后使C_{i+1}增加1，这会使解变得更优。所以除C_n外，其余每个都C_i 小于m。

设解出的w的取值范围为$u \le w \le v$，若$u = v$，那么序列直接被确定。若$u < v$则将u, v拆分，从n到1枚举系数C_i，若系数相同，那么这一位就已经确定。若第i项系数不同，设其中的较小值为t，那么我们可以确定$C_1 = C_2 = \cdots \ldots = C_{i-1} = 0$，因为如果其中有一项非0，由于$C_i \ge t$那么这个解，必定劣于解$C_i = t + 1$，$C_1 = C_2 = \cdots \ldots = C_{i-1} = 0$。同时我们发现$C_i$必定等于$t$或$t + 1$。那么之后我们只需要判断$C_i = t$是否符合要求就可以确定$C_i$的值。

枚举n求出每一个n的最优解，然后直接比较得到最优解。注意当$m = 1$是只计算$n = 0$的情况。

时空复杂度

时间复杂度O($\log^2 s$)　　空间复杂度O($\log s$)

试题编号：2011-C　　**名称**：Ancient Messages

题目大意

已知六种象形文字的形状，要求识别出给定图像中包含的象形文字，0 表示白色，1 表示黑色，图像用$n * m$个十六进制数给出，并且满足以下条件：

1、图像仅包含题目给出的六种象形文字。

2、每个图像至少有一个有效的象形文字。

3、每个黑色像素都是一个有效象形文字的一部分。

4、每个象形文字由一个联通的黑色像素块组成，对于每个黑色像素，至少在其上下左右至少有一块黑色像素块。

5、象形文字互不接触，而且不存在一个象形文字在另一个的内部。

6、如果有两块黑色像素对角线相接处，则必然存在一块公共接触的黑色像素。

$n \leq 200$，$m \leq 50$

算法讨论

观察图形我们可以发现，每个象形文字内部包含的白色区块数量各不相同，同时由于文字不接触且互不包含，所以我们可以 bfs 每个白色区块，然后只计算那些只与一个黑色连通块接触的白色连通块。然后计算出每个黑色连通块相连的可计算白色区块个数，然后直接判断然后排序即可。

时空复杂度

时间复杂度$O(nm)$　　空间复杂度$O(nm)$

试题编号：2011-E　　**名称**：Coffee Central

题目大意

在一个城市里有n个咖啡馆，这些咖啡馆都坐落在坐标系的整点上，且坐标满足$1 \leq x_i \leq d_x$，$1 \leq y_i \leq d_y$。有m个询问，每次询问有一个最大距离p，我们需要找出与之曼哈顿距离不大于p的咖啡馆数目最多的整点，输出这个最多的咖啡馆数目以及这个整点的坐标，如果有多个点满足，纵坐标小的优先，如果纵坐标相同，横坐标小的优先。

$n \leq 500000$，$d_x, d_y \leq 1000$，$q \leq 20$，$m \leq 1000000$

算法讨论

首先将所有的坐标以原点为中心，逆时针旋转$45°$，然后向外伸长$\sqrt{2}$倍。那么(x, y)求变化为$(x-y, x+y)$。设原先两个点(x_1, y_1)、(x_2, y_2)变化后为(x_1', y_1')、(x_2', y_2')，那么可以发现$|x_1 - x_2| + |y_1 - y_2| = max(x_1' - x_2', y_1' -$

y_2')。那么每个点对应的在原坐标系上在最大距离p之内的点，在新坐标系中是一个边长为$2p$的正方形。 所以我们将所有点映射到新坐标系中。在新坐标系中，用$a[i][j]$表示在坐标(i,j)的咖啡店数，用$f[i][j]$表示横坐标不大于i纵坐标不大于j的咖啡馆个数，显然$f[i][j]=f[i-1][j]+f[i][j-1]-f[i-1][j-1]+a[i][j]$。然后对于求在原坐标的点$(x,y)$最大距离$p$的范围内点数，那么找出其映射坐标$(x',y')$，那么总数$S=f[x'+p][y'+p]-f[x'-p-1][y'+p]-f[x'+p][y'-p-1]+f[x'-p-1][y'-p-1]$。

那么对于每个询问，直接枚举所有的点，计算出在p范围内的点数，然后找到总数最大的那个点即可。注意p的范围可能较大，需要考虑一下边界。

时空复杂度

时间复杂度$O(d_x d_y q)$　　空间复杂度$O(d_x d_y)$

试题编号：2011-F　　**名称**：MachineWorks

题目大意

给定n台机器，每台机器可以在D_i天买入，买入价P_i元，卖出价Q_i元（$P_i > Q_i$），从D_i+1天开始到卖出前一天每天赚G_i元。每天最多只能有一台机器。开始有C元，共D天。在$D+1$天必须卖出机器。求最大收益。

$n \leq 100000$

算法讨论

首先由于机器必定卖出，那么买入一台实际可简化为亏损P_i-Q_i元。设$f[i]$为到D_i天的最大收益，那么如果在Di天买入第i台机器（满足$f[i]>P_i$），那么之后第x天收益为$G_i*(x-D_i-1)+f[i]-(P_i-Q_i)=G_i*x+f[i]-G_i*(D_i+1)-P_i+Q_i$。设$b[i]=f[i]-G_i*(D_i+1)-P_i+Q_i$，那么$f[i]=max(G_j*D_i+B_j)$，$j<i$。那么对于两个决策$j$和$k$满足$G_j<G_k$，那么若$G_j*D_i+B_j<G_k*D_i+B_k$,变换成斜率式为$\frac{B_j-B_k}{G_k-G_j}<D_i$。显然$D_i$递增，所以可以用斜率优化 DP，维护下凸壳。由于插入的G_i不单调，所以可以用分治法解决这个问题。

时空复杂度

时间复杂度$O(n\log n)$　　空间复杂度$O(n\log n)$

试题编号：2011-H　　**名称**：Mining Your Own Business

题目大意

给定一片矿，这个矿由n个挖矿点组成，中间用m条双向通道连接。要求在挖矿点到地面建造安全竖井，使得无论哪一个挖矿点坍塌无法通过之后，其他挖矿点的工人都能从竖井逃到地面。编程求出最少需要建造多少个安全竖井，以及在满足最少的前提下有多少种方案。保证整个图联通。

$1 \le m \le 50000$，$2 \le n \le 50000$

算法讨论

如果我们发现如果割点坍塌，那么每个分支互不干涉，所以必须都要有安全竖井。所以首先求出这张图的割点，然后将每个非割点于自己相连的非割点缩成同一个点。显然这样会使整张图变成一棵树。对于每个叶节点，这个点首先不会是割点，否则必定会连接两个及以上的节点，所以其父节点一定是割点。那么如果父节点坍塌，那么这个连通分量至少需要一个竖井。如果是非叶节点，那么这个点至少连接两个叶节点，如果有一个点坍塌，那么必定有路径通向另一个叶节点。所以答案就是缩点之后叶节点的数目，方案数是每个叶节点包含原节点的乘积。

但是这样在树只有一个节点的时候是错误的。因为当$n \ge 2$时，如果只有一个竖井，当有竖井的这个节点坍塌，其余点的人无法逃离。所以一张图至少需要2个竖井。所以当树节点只有一个时，竖井的数量应该是2，方案数应为n个点任意选择2个的方案数。

时空复杂度

时间复杂度$O(n+m)$　空间复杂度$O(n+m)$

试题编号：2011-J　　**名称**：Pyramids

题目大意

一个底面边长为$n(n \ge 2)$的高金字塔需要$\sum_{i=1}^{n} i^2$个石块，一个底面边长为n（$n>2$）的矮金字塔需要的石块数则为$\begin{cases}\sum_{i=0}^{n}(2i+1)^2 \\ \sum_{i=1}^{n}(2i)^2\end{cases}$，现有$N$个石块，要修建一些金字塔，要求用完所有石块且金字塔两两不同。如果有解，则先使金字塔个

数尽量少，如果相同，则取将金字塔按大小排序之后的字典序最大的一个。

$N \leq 1000000$

算法讨论

首先将所有大小不大于N的金字塔大小预处理出来，可以发现所有的金字塔数m不超过320个，同时我们可发现当N非常大的时候，答案都非常小。所以我们可以直接用迭代加深搜索，使得答案s从1开始递增逐个判断找到最小值即可，直到s超过金字塔个数，或是最小的s个金字塔大小总和超过N。由于出现无方案时，N都比较小，非常快就能出解。所以这样做就能取得非常好的效果。

时空复杂度

时间复杂度$O(s^m)$　　空间复杂度$O(m)$

试题编号：2011-K　　**名称**：Trash Removal

题目大意

给定一个n个点不自交的多边形，将多边形旋转平移使之通过一个管道，求管道的最小宽度。

$n \leq 100$

算法讨论

首先求出这个多边形的凸包。那么我们可以证明，在最优解的情况下，凸包至少有一条边紧贴管道壁。显然如果在最优解的情况下是两点紧贴两壁，那么我们将其旋转，那么在两个方向中的必有一个方向两壁间距离会减小，直到不能旋转即有一条边贴住管道壁的情况。

那么我们只要枚举一条凸包上的边，那么其余所有点到边所在直线距离的最大值即为当这条边贴住管道时管道的最小宽度。由于这道题的n非常小，我们可以用非常暴力但方便的一个做法，即直接先枚举两个点，同叉积判断其余所有点是否在这条线段的同一侧，如果是那么这条线段必定是凸包上的一条边，然后直接枚举找到其余点到这条线段所在直线的最大值即可。然后对所有可行解取最小值即为管道最小宽度。

时空复杂度

时间复杂度$O(n^3)$　　空间复杂度$O(n)$

ACM/ICPC World Finals 2010

试题编号：2010-B　　**名称**：Barcodes

题目大意

给定一个由n个宽区域和窄区域组成的条形码。这个条形码对应着一个字符序列。每个序列有两个告诉你构造方法的校验码。告诉你每个字符所对应的的条形码，且每两个字符的条形码之间有一个窄区间分隔。每个条形码有开始标志和结束标志，而且条形码可能反向。已知宽区间的长度是窄区间的2倍，且所有条形码的长度有5%的误差。给你n个区间的长度，判断能否识别成一个字符序列，如果可以，判断校验码是否正确，如果校验码正确，输出原序列。

$n \leq 150$,每个宽度≤ 250

算法讨论

首先我们要判断每个区间为宽区间还是窄区间。对着n个区间从小到大进行排序，然后设每个为a_i。设窄区间的标准长为x，那么每个窄区间的长度$\in [0.95x, 1.05x]$，每个宽区间的长度$\in [1.9x, 2.1x]$，那么我们取使$a_{i+1} - a_i$最大的i，那么如果这个条形码可以被识别，那么$a_1 \sim a_i$为窄区间，$a_{i+1} \sim a_n$为宽区间。然后我们得到x的范围的约束：$\begin{cases} a_1 \geq 0.95x \\ a_i \leq 1.05x \end{cases}$且$\begin{cases} a_{i+1} \geq 1.9x \\ a_n \leq 2.1x \end{cases}$，则$\min\left(\frac{a_i}{1.05}, \frac{a_n}{2.1}\right) \leq x \leq \min(\frac{a_1}{0.95}, \frac{a_{i+1}}{1.9})$，判断$x$是否存在即可。

判断完之后，将序列正向和逆向分别做一次解码。将每个字符对应的条形码看成一个二进制数，那么我们就可以用一个十进制整数来表示。然后根据题目描述进行解码并做出判断即可。

时空复杂度

时间复杂度$O(n\log n)$　　空间复杂度$O(n)$

试题编号：2010-C　　**名称**：Tracking Bio-bots

题目大意

在一个m行n列的地图中，有 w 个横着的墙，每个墙只占一行，且占据的是连续几个格子，机器人不能经过墙。右上角是出口，机器人只能向上走或者向右走。问有几个出发点不能安全到达出口，墙不能作为出发点。

$m, n \leq 10^6$，$w \leq 1000$

算法讨论

虽然m,n最大有10^6，但是w最大只有1000，所以我们可以将坐标离散化，将一些小格合并成一个大格。对于行和列我们可以分开计算。设第i堵墙横向从p_i到q_i，然后我们可以将p_i-1和q_i排序成新序列r_i，然后每个r_i所在格对应的横向长度即为r_i-r_{i-1}。列也以相同方式处理。然后这样新图的横纵坐标范围只有$2*\mathrm{w}$即2000。那么我们就可以从点$(n-1,m-1)$开始直接扫描判断每个格子是否能到达终点，如果不能那么答案中增加这个格子的横向长度与纵向长度的乘积。最后输出格子数总和即可。注意终点可能会被被墙覆盖，这里比较容易出错。

时空复杂度

时间复杂度$\mathrm{O}(w^2)$　空间复杂度$\mathrm{O}(w^2)$

试题编号：2010-D　　**名称**：Castles

题目大意

给定一棵树，每个节点有一个城堡。你带领军队从任意一个节点出发，攻克所有的城堡。每个城堡i需要至少a_i个人才能攻克，攻打时有m_i个人会战死，且需要有g_i个人在攻克后留守，攻克一个城堡后可以从这个城堡出发沿树上的边到另一个城堡，同时每条边同一个方向只能走一次。问攻克所有城堡需要的最少人数。

$n\le 100$

算法讨论

枚举开始的节点作为根，做树形 DP。对于每一个节点，设攻克以这个节点为根的子树需要的人数为$f[i]$，同时记录$s[i]=\sum_{k\in i\text{的子树}} g_k+m_k$。那么每个对于 i 的子节点$j$，我们需要$f[j]$个人沿$j$节点向下，最后有$f[j]-s[j]$的人能够返回。$f[j]$显然有两个来源，一是从$j$的兄弟节点返回的人，二是$i$节点所需要增派的人。因为我们要使$f[i]$足够小，那么我们要之前返回的人数越多越好，所以我们对于 i 的所有节点按$f[j]-s[j]$从大到小排序，然后按顺序逐个模拟，即先使用之前返回的人，不够再从$f[i]$增派。设到子节点j时，剩下的从子树返回的人数为$w[j]$，那么我们可以列出递推式：

$w[j]=w[j-1]-\min(w[j-1],f[j])+f[j]-s[j]$，

$f[i]=\sum\max(f[j]-w[j-1],0)$。

最后对于所有的根，取最小值即可。

时空复杂度

时间复杂度$O(n^2 \log n)$　　空间复杂度$O(n)$

试题编号：2010-G　　**名称**：The Islands

题目大意

在平面上有n个从西向东排列的岛屿，岛屿横坐标互不相同。一个人从1号岛出发向东，到达n号点后在向西回到1号点，给定两个岛屿标号p, q，要求两条路径都要至少经过这两个岛中的一个，同时满足每个岛都至少经过一次。求行驶的最短路程。

$n \leq 100$

算法讨论

首先由于两条路径本质相同，不妨看做两条从1号点到达n号点的路径。显然两条路径除终点外经过的点各不相同，因为两点之间线段最短，走重复的点显然不如直接跳过这个点。那么不妨令第一条路径经过点p，第二条路径经过点q，设$f[i][j]$表示第一条路径到达i，第二条路径到达j的最短路径和，那么对于下一个点$k = \max(i, j) + 1$。那么当$k \neq q$时，$f[k][j]$就可以用$f[i][j] + \text{dis}(i, k)$更新，当$k \neq p$时，$f[i][k]$就可以用$f[i][j] + \text{dis}(j, k)$更新。同时为了计算方便，可以添加第$n + 1$个点使第$n + 1$个点与第 n 个点坐标相同。那么最后的最短路径即为$f[n][n + 1]$。

对于输出方案，我们可以直接记录$f[i][j]$从哪个位置转移，然后从n和$n + 1$逆推然后倒序输出即可。

时空复杂度

时间复杂度$O(n^2)$　　空间复杂度$O(n^2)$

试题编号：2010-I　　**名称**：Robots on Ice

题目大意

一个机器人需要经过一个$n * m$网格中的每一个方格，起点为$(0,0)$，终点为$(0,1)$，且每个方格只能经过一次。网格中有3个地方为登记点，第i个登记点必须是机器人第个$\left\lfloor \frac{nmi}{4} \right\rfloor$经过的格点，机器人每一步只能向上下左右四个方向移动一格。求可行的路径数。

$n, m \leq 8$

算法讨论

这是一道搜索题，但是需要一些优化。首先，我们可以维护每个点到下一个目标点距离的下界，显然如果当前时间加上这个下界仍然到不了下一个点，那么可以直接退出。对于这个下界，我们可以在已经到达某个登记点后，对下一个登记点进行 bfs，求出每个点在当时到达下一个登记点的最短距离，显然之后每个点的最短距离不会比这个小。而且在 bfs 时，如果 bfs 到的点少于已经走过的点，说明这张图已经被隔断，那么就可以直接退出。

同时我们可以维护每个点的度数，首先使终点的度数加1。然后对于每个点，如果存在周围的点度数为0，那么可以直接退出。如果存在度数为1的点，如果只有一个，那么下一步一定是走这个点，如果有多个，那么显然可以直接退出。对于度数，我们可以在访问时及时维护。这样就可以通过此题。

时空复杂度

时间复杂度$O(4^{nm})$　　空间复杂度$O(nm)$

试题编号：2010-J　　**名称**：Sharing Chocolate

题目大意

开始给定一个长为x宽为y的矩形,x、y为正整数。之后每次可以选择一个矩形将其横向或纵向切开分成两个边长为正整数的矩形。问能不能在几次操作之后分成n个矩形，每个矩形的面积给定。

$1 \le x, y \le 100$，$n \le 15$

算法讨论

首先由于n比较小，我们可以用状态压缩的方法。可以用$s[i]$表示状态 i 包含的所有矩形的面积和。$f[i][j]$表示其中一条边长为j的矩形能否分成表示为状态i的矩形。那么如果存在状态i的子集$k(0 < k < i)$，满足若$f[k][j] \cup f[i^k][j] = 1$，那么$f[i][j] = f[i]\left[\frac{s[i]}{j}\right] = 1$，最后判断$f[2^n - 1][x]$是否为真即可。但是由于要对于任意$i \in [1,2^n)$枚举子集，这样直接做的时间复杂复是$O(x * 3^n)$，无法通过此题。

但是我们可以发现，对于一个状态 i，它所有可能使$f[i][j]$为真的j，必然满足j|s[i]且$1 \le j, \frac{s[i]}{j} \le \max(x, y)$。我们先对所有$p \in [1,10000]$，预处理出满足$1 \le d, \frac{p}{d} \le \max(x, y)$的约数$d$，那么对于每个$i$只判断$s[i]$所对应的数，这样就可以通

过本题。

时空复杂度

时间复杂度$\mathrm{O}(C*3^n)$　空间复杂度$\mathrm{O}(x*2^n)$

C表示对于$p\in[1,10000]$，p可能有的最大的约数个数。

试题编号：2010-K　**名称**：Paperweight

题目大意

给定一个由两个有一个公共面的四面体拼接而成的实心物体$D-ABC-E$。物体中存在一个给定点F。将物体放到一个平面上，求在满足物体稳定的情况下，这个点到平面距离的最大值和最小值。每个点用三维坐标表示。

稳定的定义是当物体重心向任意方向移动0.2个单位后，物体不会发生位移。

算法讨论

首先通过简单物理知识可知，一个物体在放置时不会发生位移当且仅当其重心的投影在其支撑面上。首先我们可以求出四面体$D-ABC$的重心$\mathrm{G}_1=\frac{A+B+C+D}{4}$，$E-ABC$的重心$\mathrm{G}_2=\frac{A+B+C+E}{4}$。同时我们用$\mathrm{d}(P,\alpha)$表示点$P$到平面$\alpha$的有向面积。那么设平面上有三个点$A$、$B$、$C$，那么$\mathrm{d}(P,\alpha)=\frac{(\overrightarrow{AB}\times\overrightarrow{AC})\cdot\overrightarrow{AP}}{|\overrightarrow{AP}|}$，那么根据物理原理，物体的重心$G$即在$G_1$和$G_2$的连线上，且$\frac{\overrightarrow{G_1G}}{\overrightarrow{GG_2}}=\left|\frac{\mathrm{d}(E,面ABC)}{\mathrm{d}(D,面ABC)}\right|$。

之后枚举每个支撑面。注意由于D、E中可能与ABC中的其中两点四点共面，所以支撑面的形状有三角形和四边形两种情况。

对于三角形的情况，首先我们可以在6个点中枚举三个点OPQ组成一个支撑面，同时满足其余点R均要满足在这个面的同一边，即$\mathrm{d}(R,面OPQ)$同时大于等于0或小于等于0。那么重心G在这个平面的投影$G'=G-\mathrm{d}(G,面OPQ)*\frac{\overrightarrow{OP}\times\overrightarrow{OQ}}{|\overrightarrow{OP}\times\overrightarrow{OQ}|}$。首先$G'$要在$\Delta OPQ$内部，那么$G'$需要满足$\begin{cases}(\overrightarrow{OP}\times\overrightarrow{OG'})\cdot(\overrightarrow{OQ}\times\overrightarrow{OG'})\le 0\\(\overrightarrow{PO}\times\overrightarrow{OG'})\cdot(\overrightarrow{PQ}\times\overrightarrow{OG'})\le 0\\(\overrightarrow{QO}\times\overrightarrow{OG'})\cdot(\overrightarrow{QP}\times\overrightarrow{OG'})\le 0\end{cases}$。同时对于

移动0.2个单位仍然不动，那么要满足$\begin{cases}|\overrightarrow{GO}\times\overrightarrow{GP}|/|\overrightarrow{OP}|\geq 0.2\\|\overrightarrow{GO}\times\overrightarrow{GQ}|/|\overrightarrow{OQ}|\geq 0.2\\|\overrightarrow{GP}\times\overrightarrow{GQ}|/|\overrightarrow{PQ}|\geq 0.2\end{cases}$。如果两者都满足，那么$d(F,$面$OPQ)$即可作为一个可行解。

对于支撑面为四边形的情况，在枚举三角形时其实已经被枚举到。对于四点共面的情况，由于四个点必定是D、E和ABC其中两点，我们考虑枚举到DE其中一点，ABC其中两点的情况，那么属于某一个四边形必然会属于其中一个三角形。那么我们只需要判断，DE中另一个点是否在这个平面上，若是那么不妨设这两个点为O和R，其余两个为P和Q，然后判断是否满足$\begin{cases}\left|\overrightarrow{G'O}\times\overrightarrow{G'P}\right|/|\overrightarrow{OP}|\geq 0.2\\\left|\overrightarrow{G'O}\times\overrightarrow{G'Q}\right|/|\overrightarrow{OQ}|\geq 0.2\end{cases}\begin{cases}\left|\overrightarrow{G'R}\times\overrightarrow{G'P}\right|/|\overrightarrow{RP}|\geq 0.2\\\left|\overrightarrow{G'R}\times\overrightarrow{G'Q}\right|/|\overrightarrow{RQ}|\geq 0.2\end{cases}$即可。

最后将所有可行解的最大值和最小值输出即可。

时空复杂度

时间复杂度$O(1)$　　空间复杂度$O(1)$

ACM/ICPC World Finals 2009

试题编号：2009-A　　**名称**：A Careful Approach

题目大意

给定每驾飞机可以着陆的时间区间$[a_i, b_i]$（以分钟为单位，a_i、b_i是非负整数），这驾飞机可能在这个闭区间里的任意时间着陆（可以是实数）。你需要安排这些飞机的着陆时间，但是要使飞机的着陆时间点尽量均匀，即使得连续两个着陆的最小间隔尽量大，求这个最大值。

$n \leq 8$，$a_i, b_i \leq 1440$，答案精确到秒，并向上取整。

算法讨论

首先由于答案满足单调性，所以对答案二分。然后对于每一个答案m，我们使用状压 DP。用$f[i]$表示状态为i的飞机全部着陆的最早时间，开始时$f[0] = -\infty$，其余$f[i] = \infty$。那么对于一架未起飞的飞机j，如果满足$f[i] + m \leq b_j$，那么我们就可以用$\max(a_j, f[i] + m)$来更新$f[i|2^{j-1}]$。最后判断$f[2^n - 1]$是否为∞，如果是∞，那么m不可行。否则m可行。

同时这道题由于时间精确到秒，且向上取整，那么我们只需每$0.5s$设一个断点，对断点进行二分。这样就变成的整数二分，同时只需要对时间乘上2即可实现，最终答案即为最后剩下的断点取上整即可。

时空复杂度

时间复杂度$O(2^n \log t)$　空间复杂度$O(2^n)$

试题编号：2009-F　　**名称**：Deer-Proof Fence

题目大意

给定n棵树的坐标，可以用封闭的栅栏来将n棵树分隔，保证外界与每棵树的最小距离为r，求栅栏总长度的最下值。

$n \leq 9$

算法讨论

如果我们将其中几棵树围在一起，那么通过简单几何推导可知外围周长为所有点的凸包的周长加上一个半径为r的圆。由于点个数$n \leq 9$，所以可以用状态压缩的方法，每个点用1表示已经隔开，0表示未隔开，那么对于一个状态 i，可以将其所有点包住，那么直接求其所对应的凸包，算出对应周长，或是将 i 的两个子集合并得解，即$f[i] = \min(f[j] + f[i \hat{} j])$,满足$j\&i = j$。取两者中的较小值即可。最终答案救赎答案就是$f[2^n - 1]$。

时空复杂度

时间复杂度$O(n\log n * 2^n + 3n)$　　空间复杂度$O(2^n)$

试题编号：2009-H　　**名称**：The Ministers' Major Mess

题目大意

有n个议题和m个大臣，第i个大臣对k_i个议题发表通过或反对的看法。判断是否存在这n个议题的通过与未通过的方案，使得所有的大臣都有大于一半的意见被采纳。如果存在，那么判断每个议题是必定通过还是必定不通过或是都有可能。

$n \leq 100$，$m \leq 500$，$k \leq 4$

算法讨论

由于$k \leq 4$，那么当$k = 1$或2时，这个大臣的所有意见都要被采纳，当$k = 3$或4时，这个大臣只有一个议题不能被采纳。

显然这是一个 2-sat 问题，我们设点$2 * i - 1$为通过i议题，点$2 * i$为不通过这个议题。对于前一种情况，每个相关的议题都已经被确定，我们可以设置辅助点$2 * n + 1$和$2 * n + 2$，将每个议题的相反点与这两个点都不能共存。对于后一种情况，枚举任意两个议题，使得其相反情况不能共存。判断这个 2-sat 是否有解。

如果有解，显然对于每一个点i，以之为起点 dfs，如果同时遍历到了相反的两个点，那么这个点是不可取的。判断每个点是否能取。这样我们就求出了每个议题是否能通过的情况。

时空复杂度

时间复杂度$O(nmk^2)$　　空间复杂度$O(n + mk^2)$

试题编号：2009-I　　**名称**：Struts and Springs

题目大意

给定平面上$n+1$个不会相交的矩形，第0号矩形包含所有的矩形。每个矩形的上边界和下边界以及左边界和右边界有弹簧或是支杆连接，同时每个矩形的上下左右边界都和直接包含这个矩形的矩形的相对应的边界有弹簧或是支杆连接。弹簧会伸缩，而支杆长度不变。当一个矩形的外部长宽变化之后，边长对应的支杆长度不变，所有弹簧按同比例缩放。如果三根均为支杆，那么最上面或最右边的支杆会被替换成弹簧。调整m次最外部的矩形大小，问其余每个矩形的位置和边长。

$n,m \leq 500$。数据保证包含关系不会因大小调整而改变包含关系。

算法讨论

首先我们要确定矩形的包含关系。将所有矩形按照左边界的横坐标排序，当计算到第i个矩形时，从$i-1$到0扫描，那么找到的第一个包含矩形i的矩形即为直接包含它的矩形，设其为矩形j。显然如果矩形j不是直接包含矩形i，那么直接包含矩形i的矩形的位置必然在矩形j后，但是同时矩形j和矩形i之间的矩形没有包含矩形i 的矩形，所以这个假设不成立。

确定矩形之间的关系后，我们就可以发现这是一个树形的结构，那么只要直接从最外层的矩形开始，直接递归模拟即可。同时注意到一个矩形的长和宽变化的形式是一样的，所以这里用一个结构体来表示可以大大缩短代码长度。

时空复杂度

时间复杂度$O(n^2)$　空间复杂度$O(n)$

试题编号：2009-K　**名称**：Suffix-Replacement Grammars

题目大意

给定两个等长字符串和n个变换规则。每个变换规则表示如果串x为当前串的后缀，那么就可以把x换成等长的字符串y。每个规则可以使用多次。求将第一个字符串变换成第二个字符串所需的最小步数。

$n \leq 100$，串长度$l \leq 20$

算法讨论

首先我们考虑将s变换到t的过程，将所有变换之后的中间状态按照每次变换的后缀长度由大到小分段，那么显然在一个分段中，中间所有所有变换后缀的长度均小于这两端的变换后缀长度。那么这两个大的变换后缀之间的变换次数就不这两个变换后缀之前的部分无关，

这样我们就可以将所有出现的字符串编号，设$f[i][j]$表示将编号为 i 的后缀

变成编号为j的后缀所需的次数。显然开始时如果后缀x能变换成后缀y，那么$f[x][y]=1$，如果字符串y是x的后缀，那么$f[x][y]=f[y][x]=0$，其余初始值均为$+\infty$。

显然由于已经分层，那么对于后缀x和后缀y，枚举中间下一个分层的分割点z，同时z要满足两个条件：

1、首先后缀z的长度不大于后缀x和后缀y。这可由定义得到。

2、对于x和y的重叠部分，若出现不相同时还剩下l个字符，那么z的长度显然要大于等于l的长度。因为根据定义x与z之间的变换后缀和z与y 之间的变换后缀长度均不大于z的长度，那么如果z的长度小于l，那么不相同的部分就不会改变，显然这是不成立。

这样在预处理每对后缀x和后缀y之间可以用那些后缀z作为中间节点后，我们就可以得到一个转移方程$f[x][y]=min(f[x][z]+f[z][y])$。观察这个式子我们就可以发现这是一个三角不等式。所以我们可以用类似 flody 的算法的去解决这个问题。同时可以发现中间状态z的长度单调递增，那么根据 flody 算法原理，只要先将所有的后缀按长度从小到大排序后重标号，那么只需要做一遍 flody 就可以计算出所有后缀之间的“最短路”。就是字符串s和t间的最小变换次数就是其所对应点的“最短路”。

时空复杂度

时间复杂度$O(n^2l+n^3)$ 空间复杂度$O(n^2)$

ACM/ICPC World Finals 2008

试题编号：2008-A　　**名称**：Air Conditioning Machinery

题目大意

给定$n*m*l$一个长方体的方格型空间，给你一个入水口和入水方向，以及一个出水口和出水方向。现在有6个$1*2*3$的“L”型水管，两个开口的位置确定。水管的开口可以连接，但水管不能相互重叠而且不能超出长方体空间。求能否将入水口和出水口连通，如果可以输出水管的最少数量。

$n,m,l \leq 20$

算法讨论

对于每个点和方向，枚举“L”型水管的连接方式和连接方向。显然方式有2种，方向有4种。由于层数不超过6，所以直接搜索即可。

时空复杂度

时间复杂度 $O(8^6)$　空间复杂度$O(nml)$

试题编号：2008-B　　**名称**：Always an Integer

题目大意

给定一个形如$\frac{P(n)}{d}$的多项式，。其中，$P(n)$是一个系数是整数的多项式，d为正整数。判断是否n取任意正整数，这个多项式的值都为整数。

$P(n)$次数t不超过100，d和$P(n)$中系数均为32位有符号整数。

算法讨论

对于n属于Z_+，满足$\frac{P(n)}{d} \in Z$，那么只要满足$\begin{cases} d|P(1) \\ d|P(n+1)-P(n) \end{cases}$。显然对于第二个式子，每次相减后多项式的次数少1，但是n的范围没有改变。那么我们只要递归判断，直到最后多项式为常数为止即可。

但这样做有一点麻烦，其实我们可以发现，设第i次操作后的多项式为$P_i(n)$，开始多项式为$P_0(n)$，那么$\frac{P(n)}{d} \in Z$等价于$\forall i \in [0,t], d|P_i(1)$。那么我们可以用归纳法证明$\forall i \in [x,t], d|P_i(1) \Leftrightarrow \forall i \in [1,t-x+1], d|P_x(i)$：

1、当$x=t$时，上式显然成立

2、当$x < t$时，设到x为止上式成立，那么对于$x-1$，

$$\forall i \in [x-1,t], d|P_i(1) \Leftrightarrow \begin{cases} d|P_{x-1}(1) \\ \forall i \in [1,t-x+1], d|P_x(i) \end{cases}$$

$\Leftrightarrow \forall i \in [1,t-x+2], d|P_{x-1}(i)$。

所以我们只需从0到$t+1$枚举n计算每一个$P(n) mod\ d$的值，判断是否都为0即可。

时空复杂度

时间复杂度$O(t^2)$　空间复杂度$O(t)$

试题编号：2008-E　**名称**：Huffman Codes

题目大意

有n个结点，每个结点有一个正整数的权值，且权值之和为100。然后构建一棵哈夫曼树，每次选两个最小的合并，要求左子树的权值要小于等于右子树的权值。通向左子树的边标号为 0，通向右子树的边标号为 1。给定n个叶子从根到叶子C路径上的 01 串，求有几种权值构成的哈夫曼树编码与这些 01 串相同。

$n \leq 20$

算法讨论

首先观察这棵树，我们就可以发现，所有节点的值先从下到上后再从左到右单调递增。这个证明比较简单，首先对于叶节点，由排序不等式可得，数字越小，所在节点在树中的深度越大。然后对同一行而言，必然取小的那两个，然后再较大的两个，那么前两个数的和必然小于等于后两个数的和。

实际上满足这个条件的树非常少。这样，我们可以按顺序枚举每个叶节点的权值，使得生成出的树满足之前得出的性质。这里有两个可行性剪枝：

1、所有叶节点总和为100，所以最后一个不用枚举，可以直接计算。

2、设共有t个叶节点，已枚举x个叶节点，总和为s，那么第$x+1$个叶节点的权值z不超过$\left\lfloor \frac{100-s}{t-x} \right\rfloor$。

然后直接统计满足条件的树个数即可。

时空复杂度

时间复杂度$O(ans * n)$　空间复杂度$O(n)$

试题编号：2008-F　　**名称**：Glenbow Museum

题目大意

存在一些长度为n由 R 和 O 组成的序列，每个序列对应一个n个顶点的多边形。这个序列的每一个字符顺次代表了这个多边形的角度大小。多边形顶点的角度只能为$90°$或$270°$，R 表示$90°$，O 表示$270°$。问存在多少个这张的序列使得对应的多边形内存在一个点可以看到多边形的所有面积。

算法讨论

首先这个多边形有n个顶点，那么内角和为$(n-2)*180°$，设 R 有x个，O 有y个，那么列方程$\begin{cases} x+y=n \\ 90x+207y=(n-2)*180^\circ \end{cases}$解得$\begin{cases} x=\frac{n}{2}+2 \\ y=\frac{n}{2}-2 \end{cases}$，同时我们发现$n$必须为偶数，且不小于$4$。我们可以发现，如果存在两个连续的 O，那么就会产生一个内凹的形状，显然这种情况会存在视觉死角。那么所有的情况即使将所有的 O 插入到的一排 R 中，并且 O 不能相邻。注意图形是环形的，所以开头和结尾不能同时存在。所以我们分开头为 O 和开头不可以为 O 两种情况，答案即为$\mathrm{C}_x^y+\mathrm{C}_{x-1}^{y-1}=\mathrm{C}_{\frac{n}{2}+2}^{\frac{n}{2}-2}+\mathrm{C}_{\frac{n}{2}+1}^{\frac{n}{2}-3}=\mathrm{C}_{\frac{n}{2}+2}^{4}+\mathrm{C}_{\frac{n}{2}+1}^{4}$。

时空复杂度

时间复杂度$\mathrm{O}(1)$　　空间复杂度$\mathrm{O}(1)$

试题编号：2008-G　　**名称**：Net Loss

题目大意

给定一个范围在$[-1,1]$的函数$f(x)$，每个函数是一个n次的多项式即$\mathrm{f}(x)=\sum_{x=0}^{n}P_x x^x$，你需要求出两条相交线段$a_1x+a_0$和$b_1x+b_0$组成的函数$g$，满足线段交点的横坐标为$c$，使得$d(f,g)=\int_{-1}^{1}\big(f(x)-g(x)\big)^2dx$最小。

$n\le 10$，$-1\le pi\le 1$

算法讨论

首先将这个式子分拆：

$$d(f,g)=\int_{-1}^{c}\big(f(x)-(a_1x+a_0)\big)^2dx+\int_{c}^{1}\big(f(x)-(b_1x+b_0)\big)^2dx$$

对前一部分展开:

$$\int_{-1}^{c}\big(f(x)-(a_1x+a_0)\big)^2dx$$

$$=\int_{-1}^{c}(f(x)^2-2(a_1x+a_0)f(x)+(a_1x+a_0)^2)dx。$$

由于我们只考虑a_1、a_0、b_1、b_0四个值，所以我们可以直接将常数忽略。所以我们只需求$\int_{-1}^{c}(a_1x+a_0)^2dx-2\int_{-1}^{c}(a_1x+a_0)f(x)dx$。

对这两个简单积分计算求解并将a_1和a_0换为主元可得:

$$\int_{-1}^{c}(a_1x+a_0)^2dx=\frac{a_1^2}{3}x^3+a_1a_0x^2+a_0^2x\Big|_{-1}^{c}$$

$$=\frac{c^3+1}{3}a_1^2+(c^2+1)a_1a_0+(c+1)a_0^2$$

$$\int_{-1}^{c}(a_1x+a_0)f(x)dx$$

$$=\int_{-1}^{c}\left(a_1p_nx^{n+1}+\sum_{i=1}^{n}(a_1p_{i-1}+a_0p_i)x^i+a_0p_0\right)dx$$

$$=\frac{a_1p_n}{n+2}x^{n+2}+\sum_{i=2}^{n+1}\frac{a_1p_{i-1}+a_0p_i}{i}x^i+a_0p_0x\Big|_{-1}^{c}$$

$$=\sum_{i=2}^{n+2}\frac{p_{i-2}}{i}(c^i-(-1)^i)\,a_1+\sum_{i=1}^{n+1}\frac{p_{i-1}}{i}(c^i-(-1)^i)\;a_0$$

后半部分由于形式相同，所以同理可得。这样我们可以得到:

$$\mathrm{d(f,g)}=\mathrm{q}_1a_1^2+q_2a_1a_0+q_3a_0^2+q_4a_1+q_5a_0+q_6b_1^2+q_7b_1b_0+q_8b_0^2+q_9b_1+q_{10}\mathrm{b}_0$$

同时我们有条件$a_1c+a_0=b_1c+b_0$，使$b_0=a_1c+a_0-b_1c$带入，得到:

$\mathrm{d(f,g)}=\mathrm{r}_1a_1^2+r_2a_0^2+r_3b_1^2+r_4\,a_0a_1+r_6a_0b_1+r_7a_1+r_8a_0+r_9b_1$。

对其求偏导可得。$\begin{cases}S'(a_1)=2r_1a_1+r_4a_0+r_5b_1+r_7=0\\S'(a_0)=2r_2a_0+r_4a_1+r_6b_1+r_8=0\\S'(b_1)=2r_3b_1+r_5a_1+r_6a_0+r_9=0\end{cases}$。用高斯消元法将$a_1$、$a_0$、$b_1$求出，并计算出$\mathrm{b}_0$即可。

时空复杂度

时间复杂度$O(n)$　　空间复杂度$O(n)$

试题编号：2008-J　　**名称**：The Sky is the Limit

题目大意

给定平面上n个等腰三角形，保证所有三角形底边在X轴上且顶点在X轴上方。问从Y轴向下能看到的三角形边的长度。

$n \leq 100$

算法讨论

首先将每个三角形拆成两条线段，那么总共有$O(n)$条线段。然后枚举线段求两两之间的交点的横坐标，然后加上所有线段的端点的X坐标，将其排序，时间复杂度为$O(n^2 \log n)$。

我们延X轴从左往右扫描，然后就可以不断添加和删除线段。同时我们可以发现，在X轴上相邻的两个交点的横坐标x_1，x_2之间，所有线段的上下顺序不会改变。那么我们可以取$f(\frac{x_1+x_2}{2})$直接枚举进行比较得知每一段最上面的线段是那一条，并求出其长度。最后将每一段的长度想加即是最后的答案。这样，时间复杂度是$O(n^3)$，已经可以通过题目。

其实我们可以维护当前这一段上的线段从上到下的顺序，发现每一个交点表示一对线段的先后顺序被交换，同时在交换前这两条线段的位置是相邻的，且在交换前这两条线段。如果是多条线段交于同一点，则是所有经过这一点的线段的顺序全部倒置。由于最多有$O(n^2)$个交点，那么这样的交换最多进行$O(n^2)$次。同时每插入和删除一条线段时间为$O(n)$，进行$O(n)$次。那么这一部分就可以做到$O(n^2)$。

时空复杂度

时间复杂度$O(n^3)$或$O(n^2 \log n)$　空间复杂度$O(n^2)$

试题编号：2008-K　　**名称**：Steam Roller

题目大意

给定一张$n*m$的网格图，每个格点可以向上下左右连边。每条边的经过的时间给定。若在经过一条边之前或之后要改变方向，以及从起点出发或到达终点，那么这条边的时间翻倍。给定起点和终点的位置，求起点到终点所用的时间。

算法讨论

显然这是一道比较明显的拆点最短路。我们可以将每个点拆成五个点，分别表示从上下左右到达这个点且之后这个点不能改变方向的最短路和到这个点且下一步可以改变方向的最短路。将这五个点标号为1,2,3,4,0。那么对于两个点u、v，设从u到v的方向为x，权值为c，那么我们需要考虑四种情况：

1、u点不改变方向，且v点的方向也不改变。那么就从u_x向v_x连一条边权为c的边。

2、u点改变方向后到达v点，而且但在v点方向不改变，那么就从u_0到v_x连一条边权为$2c$的边。

3、u点不改变方向，但是v点要改变方向，那么就从u_x到v_0连一条边权为$2c$的边。

4、u点改变方向后到达v点，且v点也改变方向，那么就从u_0到v_0连一条边权为$2c$的边。

那么起点和终点的特殊情况同理，那么设起点为s，终点为t，那么最终答案即为s_0到t_0的最短路。由于这张图比较稀疏，可以使用二叉堆优化的 Dijkstra 算法。

时空复杂度

时间复杂度$O(nm\log(nm))$　　空间复杂度$O(nm)$

ACM/ICPC World Finals 2007

试题编号：2007-A　　**名称**：Consanguine Calculations

题目大意

给定一对父母和他们的一个孩子三个人中两个人的血型，输出未被告知的人所有可能的血型。

算法讨论

对于告知父母的血型，求孩子血型的这种情况，我们可以直接枚举计算。

对于告知一个父母的血型和孩子的血型求另一个父母的血型。我们只要枚举另一个父母的血型，然后计算出这两个人所有能生出的孩子的血型，判断其中是否有这个孩子的血型。然后输出所有正确的血型即可。

时空复杂度

时间复杂度$O(1)$　　空间复杂度$O(1)$

试题编号：2007-G　　**名称**：Network

题目大意

有n条信息，被分成m个信息包。每个信息包包含所属信息，开头编号和结尾编号。信息包依次通过，可以选择直接输出或放入缓存区，缓存区的信息包可以任何时间输出。每一条信息必须按顺序编号连续输出，并且无视信息间的顺序。求缓存区空间的最小值。

$n \leq 5$，$m \leq 1000$

算法讨论

首先对信包进行排序，就可以预处理出每条信息的开始信息包和每个信息包之后的那个信息包的位置。

首先枚举信息通过的顺序，那么我们就确定的信息包的通过顺序，那么只要按顺序扫描所有信息包。设当前的扫描到第i个信息包，判断当前可以输出的信息包的位置是否等于i，如果是那么直接输出信息包i，否则将信息包i放入缓存区。然后如果当前可以输出的信息包位置小于i，那么就从缓存区中取出，直到大于i为止。那么在这种信息输出顺序下的缓存区空间大小即为每个时刻缓存区信息包总和的最大值。

然后答案即为所有情况空间大小的最小值。

时空复杂度

时间复杂度$O(mlogm+n!*m)$　　空间复杂度$O(m)$

试题编号：2007-I　　**名称**：Water Tanks

题目大意

给定由n个矩形水箱，水箱$i(i<n)$与水箱$i+1$用水平管道连通，管道高度递增。水箱1开放，其余均封闭。向水箱1中倒水，由于水箱中存在空气，在水注入时空气被压缩，气压增大，会使水无法完全充满水箱。注意气体体积与压强的乘积不变。求当水倒到水箱1注满时，倒入水的总量。

$n\leq 10$，同时设定开始大气压为1，一米水柱的压强为0.097。

算法讨论

设每个水箱的高度为H_i，每根管道的高度为h_i。先注$2h_i$体积的水，使水没过第一根管道，这之后水箱内部的空气就会被密封。我们用v表示与最后一个水箱连通的空气体积，开始时$V=\sum_{i=2}^{n}H_i-h_1$,$P=1$。由于水管的高度递增，我们可以从水箱2开始逐个递推，对在每一个水箱$i(i>1)$我们讨论两个过程：

1、水箱i的水面从h_{i-1}涨到h_i。

在这个过程中，水面升高的同时水压在变化，我们假设当第一个水箱水面到达顶部时水箱i的水面上涨x米（以后均如此表示），我们就能列出方程：$0.097*\big(H_1-(h_{i-1}+x)\big)+1=\frac{pv}{v-x}$,这是一个二次方程，直接化简用求根公式取较小值，若$x<h_i-h_{i-1}$，说明水箱i最后的高度是x，且水不会到达水箱i之后的水箱。否则水箱i水面至少上升到x，同时空气体积减少h_i-h_{i-1}，气压做出相应改变。

2、水箱$i+1$水面从0涨到h_i。

在这个过程中由于水箱i的水面不会改变，所以水压不会改变，所以可列方程：$0.097(H_1-h_i)+1=\frac{pv}{v-x}$，这是一个一次方程，化简后解出$x$，若$x<h_i$，说明水不会到达水箱$i+1$之后的水箱，否则那么不断注水可以使得第$i+1$水箱的水面超过$h_i$，同时空气体积减少$h_i$，气压做出相应改变。但于此同时，水箱$i$上方的空气与后面的空气相互独立，可列出方程：$0.097*\big(H_1-(h_i+x)\big)+1=\frac{p(H_I-h_i)}{H_i-(h_i+x)}$，$x$取根中较小值。那$h_i+x$即为水箱 i 的最后高度。同时最后连通的气体的的体积会减少H_i-h_i，但是气压不会改变。

最后将每个水箱的最后高度相加即是答案（第一个水箱的最后高度是H_1）。由于运算中包含大量实数除法和开方，需要注意精度。

时空复杂度

时间复杂度$O(n)$　　空间复杂度$O(n)$

试题编号：2006-A　　**名称**：Low Cost Air Travel

题目大意

有n个城市，这些城市之间有m条飞机航线，每条航线会依次经过一些城市，可能会有城市经过多次，但相邻两城市不同，其价格为W_i。乘飞机时，我们只能选择一条起点在当前城市的航线，可以在中途下飞机，但还是要支付全额机票。现在有若干个询问，每次询问为一个长为L的城市序列，我们需要依次访问这些城市（起点在第一个城市，不一定要连续访问，但只有按照顺序的访问才是有效的），求最小的支出以及最优方案，保证最优方案唯一。

$L, m \leq 20$，一张机票和一条线路中的城市$c \leq 10$。

算法讨论

用$f[i][j][k]$表示已经到达过序列中的第i个点，现在乘坐航线j，在航线的第k个城市。首先设当前点为x，显然我们有两种选择

1、换乘航线，即选择一个以x为起始点的航班设为l，那么就可以用$f[i][j][k]+W_l$更新$f[i][l][1]$。

2、如果k不是当前航线的最后一个点，那么就可以到达下一点，那么就可以用$f[i][j][k]$更新$f[i][j][k+1]$。同时如果下一个点是序列的第$i+1$个点，那么就可以用$f[i][j][k]$更新$f[i+1][j][k+1]$。

显然这是一个分层图模型，我们可以在每一层跑一遍 spfa，在层与层之间直接枚举更新。初始情况为$f[1][i][1]=0$（满足航线 i 起点为序列的第一个点），其余所有状态为$+\infty$。然后最小支出即为$f[n][i][j]$中最小值。

对于输出最优方案，那么就在每个状态上记录上一个状态的位置，然后从最优情况倒推求解即可。

时空复杂度

时间复杂度$O(kLmc)$　　空间复杂度$O(mc+Lc)$

试题编号：2006-B　　**名称**：Remember the A La Mode!

题目大意

有P种饼干和I种冰激凌，每种的数量给定且饼干和冰激凌总数相同，第i种饼干和第j种冰激凌若可以组合，那么组合可得到$C_{i,j}$的利润，求利润的上界和下界。

$P, I \leq 50$

算法讨论

显然这是一道费用流的题目，将每种饼干和冰激凌当做一个点，从原点到每种饼干连容量为饼干个数，费用为0的边，然后从每种冰激凌到汇点连容量为冰激凌个数，费用为0的边。最后从每种饼干到其可组合的冰激凌连容量为$+\infty$，费用为两者组合利润的边。然后分别用最小费用最大流和最大费用最大流计算费用即可。注意费用为小数，在更新时要注意小数精度问题。

时空复杂度

时间复杂度$O(kn^3)$　　空间复杂度$O(n^2)$

试题编号：2006-D　　**名称**：BipartiteNumbers

题目大意

定义二段数是由a个x，b个y顺次连接形成的十进制数$(1 \le x \le 9，0 \le y \le 9，x \ne y，a > 1，b > 1)$。对于每个输入的$n$，求出大于$n$且是$n$的倍数的最小二段数。

$n < 100000$

算法讨论

首先我们可以预处理出$d[i][j]$表示当前模n余j的数在末尾加上最少的i的个数，使得当前数整除n。这相当于点(i,j)向$(i,(j*10+i)\%n)$连边，然后求出每个点到点$(i,0)$的最短路。显然$d[i][j]$不会超过n。

这样对于我们可以枚举前面的数字i和长度l。假设我们先不考虑这个二段数要求大于n，那么这个长度 l 同理也同样不会超过n，设前面模n的余数为m，对应的二段数即为l个 i 加上$d[j][m]$个j（满足$i \ne j$）。

那么由于二段数要大于n，所以当枚举到某一个i和l时最小值等于n时，由于可以发现在预处理的那张图是环加外向树。对于每个点(j,m)，能到达点$(j,0)$的路径长度的循环节即为$d[j][j\%n]+1$。所以我们可以得到新的两个可行解，即为l个i加上$d[j][m]+d[j][j\%n+1]+1$个j和 l+$d[i][i\%n+1]+1$个i加上的$d[j][m]$个j。然后在所有的解中找到最小的一个即可。

同时注意最终答案的解比较小，所以在枚举 l 时，当l长度超出已知最优解的长度时，我们可以直接跳出枚举下一个k，这样可以大大加快运行速度。

时空复杂度

时间复杂度$O(n)$　　空间复杂度$O(n)$。

试题编号：2006-E　　**名称**：Bit Compressor

题目大意

对于一个二进制串s，当n的二进制长度小于n时，可以将连续的n个'1'压缩成n的二进制表示。给定原串长度l和'1'的个数，以及压缩后的字符串，判断原串是否存在以及是否有多种可能。

$l \leq 16 * 1024$，s长度m小于40。

算法讨论

这道题显然可以搜索所有可能的情况。如果一段字符串从'1'开始，且这一段的后一个字符为'0'为或这一段为s的一个后缀，那么这一段字符串可能是被压缩过的。注意原串中的'11'的不会被压缩，所以当前串中'10'不会是压缩串，且当前串里的'11'可能是原串中的'11'或是'111'。然后加上简单的可行性剪枝——当前已经解压出来的串长度和'1'的个数不会比原串多。然后这样就可以通过此题。

时空复杂度

时间复杂度$O(2^m)$　　空间复杂度$O(m)$

试题编号：2006-G　　**名称**：Pilgrimage

题目大意

给定一个账目的一部分，有四种操作 1、加入x个人。2、减少x个人。3、每个人收x元。4、支出x元。同时在加入人时每个加入的人需要支付一些钱，减少人每个离开的人可以得到一些钱人，使得均资金不变。已知每个人得到和支出的钱必定是整数。求账单开始时可能的人数，如果有有无穷多个，输出其下限。

操作总数$n \leq 50$，每次数量变化不超过2000。且任何时候都至少有一个人。

算法讨论

我们设开始的人数为a。由于人数变化时每个人得到或支付的钱为整数，所以当人数变化后人数为x，总钱数为S，显然$x|S$。由于每次收钱时收入的钱是 x 的倍数，所以总钱数仍然整除人数，那么我们只需考虑支出的钱。设两次变化之间支出的钱总和为y，那么我们要使$x|S+y$，也就是$x|y$。

然后我们要设从开始到第i次人数变化之后人数变化了c_i，第i次变化到第$i+$

1次变化之间支出了了p_i。我们可以对于每一个列出不等式$a + c_i \geq 1$，且对于所有的$p_i \neq 0$可列出方程$a + c_i | p_i$,且。那么每次求出p_i的所有约数计算出a所有可能的值，满足所有等式的a即为符合的开始人数。对于找出满足所有满足条件的a，我们可以计算出满足第一个式子的所有解，然后将其带入之后的不等式和方程，如果不满足就将其删去，最后剩下的就为全部可行解。

注意如果每次p_i均为0，那么a就会有无限个值，而且我们只能解出a的下限。同时对于账目开始，我们只能列出不等式$a \geq 1$，而且由于开始时S对a的余数不确定，从开始到第一次变化之间的支出总额没有意义，这里很容易出错。

时空复杂度

时间复杂度$O(nC)$　空间复杂度$O(100000 * C)$

C表示1~100000中一个数最多可能有的约数个数。

试题编号：2006-I　　**名称**：Degrees of Separation

题目大意

给定n个人和m条认识关系，每条认识关系表示两个人之间互相认识，定义两个人的分离度是联系两个人需要经过的最小关系数。求有没有两个人无法联系，如果都能互相联系，输出网络中任意两人的最大分离度。

$n \leq 50$

算法讨论

将每个人看做一个点，每个认知关系看做一条边权为1的无向边。用 floyd 算法求出每对点之间的最短路，找出其中最长的一条即可。

时空复杂度

时间复杂度$O(n^3)$　空间复杂度$O(n^2)$。

ACM/ICPC World Finals 2005

试题编号：2005-C　　**名称**：The Traveling Judges Problem

题目大意

给定n个点和m条无向边，每条边有一个边权。给定t个点，可以选择一些边使得这t个点都与点d联通，求选择边的最小权值和以及这t个点到点d的路径。在权值和相同时选经过点最少的方案，如果经过的点相同，选择经过点字典序最小的一个。

$n \leq 20$，$t \leq 10$

算法讨论

首先我们可以发现，最后的图一定是一棵树。因为如果存在一个点的到另一个点有两条路径，那么只需保留短的那一条即可。

使$a_0 = d$，其余t个点用a_1到a_t表示。然后我们设每个i的点权为$2^n + 2^{i-1}$，然后使每条边的权值乘上$(n+1) * 2^n$。然后求点1和其它t个的点的斯坦纳树即可。将这$t+1$个点状态压缩。用$f[i][j]$表示将状态 i 对应的点连通且当前树根为j的权值最小和。显然初始状态为$f[2^i][a_i] = c[a_i]$。那么我们可以列出两个方程：

1、$f[i][j] = min(f[i][k] + e[j][k] + c[j])$

2、$f[i][j] = min(f[k][j] + f[l][j] - c[j])$，且状态$k$和状态$l$满足$k\&l = 0$，$k|l = i$。

设$s = min(f[2t + 1 - 1][j])$，那么最后的权值和为$\left\lfloor \frac{s}{(n+1)*2^n} \right\rfloor$，同时我们可以去处$s$在二进制下的后$n$位，从中得知在取到最小权值和时需要用到那些点，然后对这些点求最小生成树，然后输出d到t中每一个点的路径即可。

时空复杂度

时间复杂度$O(n^2 * 3t)$　空间复杂度$O(n * 2^t)$

试题编号：2005-E　　**名称**：Lots of Sunlight

题目大意

n幢公寓楼从东至西排列，每栋楼有m_i层。每层有一间公寓。公寓门牌号的后两位代表楼栋编号，公寓楼编号从东至西递增。门牌号的其余位则代表楼层。每层的高度为h，宽度为w，第i幢公寓和第$i+1$幢公寓的距离为d_i。

太阳从东到西以恒定角速度划过天空。阴影仅由楼房投射出。当一间公寓的

整块东侧或西侧外墙被太阳直射，或者当太阳处于公寓正上方时，公寓受到太阳直射。已知早上5点37分日出，于下午6点17分日落。

给定q间公寓，判断是否存在这间公寓，如果存在输出这一间公寓当天的太阳直射时间。

$n, m < 100$，$d, w, h \leq 100$，$q \leq 1000$

算法讨论

对于每一间公寓，考虑东墙直射和西墙直射两种情况。对于前一种情况，只要判断东墙的最低点是否被前一个楼层的最高点挡住。那么第x幢楼的第y层被第$i(i < x$且$m_i \geq x)$幢楼挡住的最大角度为$\tan^{-1}\frac{(m_i - x + 1) * h}{dis(x,i)}$，$dis(x,i)$表示第$x$幢东墙到$d_i$幢西墙的距离。那么东墙被直射的最小太阳高度角为东边所有楼挡住阳光的最大角度的最大值，西面同理。

手算可得一天中太阳光照时间为$45600s$，按照角度π均分。由于上面已经算出每间公寓开始直射和结束直射的太阳高度角，直接可以计算出这两个时间点从太阳升起开始所经过的秒数，再从太阳升起时间向后推即可。

时空复杂度

时间复杂度$O(qn)$　空间复杂度$O(n)$

试题编号：2005-H　　**名称**：The Great Wall Game

题目大意

在一个$n * n$的棋盘上有n个棋子，每个开始坐标给定。每个棋子可以向上下左右移动，且每个格子只能放一个棋子，问将所有棋子放在同一直线上至少需要几步。

$n \leq 15$

算法讨论

首先枚举最终直线的所有情况，总共有$2n + 2$种。然后我们考虑每个棋子的对应位置。显然每个格子只能放一个棋子的条件不需要考虑，因为如果这个棋子要经过另一个棋子才能到达对应位置，那么将这两个棋子的对应位置互换，答案显然不会变差。这样每个棋子到对应点的步数即为两点之间的曼哈顿距离。这样就变成了n个点的完全匹配问题，直接套用 km 算法即可。最后所有可能情况的的最小值即为答案。

时空复杂度

时间复杂度$O(n^4)$　空间复杂度$O(n^2)$

试题编号：2005-I　　**名称**：Workshops

题目大意

给定n个会议的人数和时间，m个房间的人数上限和关闭时间，一个会议只能在一个房间里开，一个房间也只能容纳一个会议，且人数上限应该大于等于会议的人数，时间应大于等于会议的时间。要求作出一种安排，使得不能被满足的会议最少，如果相同，再使不能被满足的人数最小。

$n, m \leq 1000$

算法讨论

这道题，我们可以用贪心来做。将所有房间按照时间从小到大排序，然后从小到大，每次选择未被选择且能被房间容纳的会议中人数最多的一个，如果存在则加到答案中即可。

证明如下：由于房间按时间排序，若某个会议的时间能被当前房间容纳，那么肯定也能被后面的房间容纳。如果这个房间不选择人数最多的那个，就会有两种情况。

1、如果你这个房间不选择任何会议。但是如果这个会议没有被其它房间选择，解肯定会变差，如果在其它房间被选择，那么就比直接被这个房间选择少了一个房间，解不会变优。

2、选择了其它能被容纳的会议。因为开始选出的是人数最多的会议，如果这个会议之后未被选择，那么解肯定不会比直接选这个会议优。如果在之后的房间选择了这个会议，由于这个会议人数不小于其它会议，那么之后的房间人数限制必然会满足其它会议，那么我们可以将这两个房间所容纳的会议交换，显然解不会改变。

这样我们就证明了这样做必然能得到最优解。

时空复杂度

时间复杂度$O(nm + m\log m)$ 空间复杂度$O(n + m)$

试题编号：2005-J　　**名称**：Zones

题目大意

给定n个圆和每个圆所包含的点的个数。由于一个点可以属于多个圆，所以

同时告诉你对于t个圆的集合，同时告诉你这个集合中的圆都包含的同时不在这个集合中的圆不包含的点的个数。求选择m个圆最多能覆盖多少个点。

$n \leq 20$，$t \leq 10$

算法讨论

直接枚举这m个点，然后计算这m个圆覆盖了多少的点。对于计算覆盖点的数目，由于有一些点是圆共有的，使S为每个被选择的圆覆盖点数的和。显然对于一个覆盖x个点的圆集合，如果有k个属于这个集合的圆同时被选，那么S需要减去$(k-1)*x$。然后对于所有求出的S取最大值即可。

时空复杂度

时间复杂度$O(C_n^m * nt)$ 空间复杂度$O(nt)$

ACM/ICPC World Finals 2004

试题编号：2004-E　　**名称**：Intersecting Dates

题目大意

已知n个日期区间的股票数据，现在要查阅m个日期区间的股票数据，问要查阅的日期区间里有哪些区间不是已知的，并且要把该并的区间并起来。

$n, m \leq 100$，日期年份在1700到2100之间。

算法讨论

显然从1700年到2100年总共有不超过150000天，那么直接将计算每个日期是从1700年1月1日开始的第几天，然后枚举每一个区间的所有值，并且标记每一天是否已经查阅和是否需要查阅。然后合并并输出所有需要查阅但是未被查阅的日期即可。

时空复杂度

时间复杂度$O(C)$　　空间复杂度$O(C)$

C为总共存在的天数。

试题编号：2004-H　　**名称**：Tree-Lined Streets

题目大意

给定二位平面内的n条街道，街道之间会交叉形成十字路口，保证所有相交为完全相交，街道不会重叠且三条街道不会交于同一点。我们可以在街道上种树，而且树与十字路口的距离不小于25米，同一条街道上树的距离不小于50米，求最多能种多少树。

$n \leq 100$

算法讨论

对于每一条街道我们可以扫描其它街道，求出所有在这条街道上的交点，对交点的横坐标进行排序，算出相邻两两间，即每一段都被十字路口分离。由于每棵树要距十字路口至少25米，且$50 = 25 * 2$，所以每一段上的树相互独立，那么设每一段的长度为S，其端点有T个为十字路口，那么每一段可种树$\left\lfloor \frac{S-25T}{50} \right\rfloor + 1$(要满足$S \geq 25T$)。答案即所有段上树的数量的和。

时空复杂度

时间复杂度$O(n^2 \log n)$　空间复杂度$O(n)$

试题编号：2004-I　　**名称**：Suspense!

题目大意

有两幢相邻的公寓，距离为d，高度分别为n和m。有人想要在两幢公寓之间建造一座微型的吊桥。这两根吊桥的缆绳被系在最高点的窗台上，每条缆绳会形成一个抛物线。水平的桥面正好比主缆的最低点的低1米，而且至少比两个窗台都低2米，同时至少比地面高一米。

同时每间公寓可能在窗台上养猫或养鸟。一只猫可以向上跳小于$0.5m$的距离，向下跳小于3米的距离。猫可以跳上桥并且会跳到有鸟的窗台上。这两座公寓的每个房间都是3米高，且所有窗台都比所在的房间的地面高1米。问在不伤害任何鸟的情况下，缆绳的最大长度是多少。

算法讨论

首先要使缆绳的长度越长，就要使桥面越低。首先可以分两种情况：1、猫不能从跳到桥上。2、桥上的猫不能跳到有鸟的窗台上。对于这两种情况的任何一种情况，都可以转化成同一个问题：在一个大区间中删去一些区间，问大区间中未被删去的最小值。

将所有区间画到数轴上。由于小区间的两端随楼层升高单调递增，且都是开区间。所以首先设开始高度$x = 1$，然后如果x属于某个小区间，那么将x赋值为该区间的右端，不断这样操作就可以找出题目要求的最小值。然
在此处键入公式。后在这两种情况中选择小的那个即可。

求出桥面的最小高度即求出缆绳最低点的最小高度以及两个窗台距缆绳最低点的竖直距离y_1和y_2。然后将通过最低点作为原点建立坐标系。那么这个抛物线为$y = ax^2$。并且经过点(x_1, y_1)和(x_2, y_2)。可列方程$\begin{cases} y_1 = ax_1^2 \\ y_2 = ax_2^2 \\ x_2 - x_1 = d \end{cases}$ 且$x_1 < 0$，$x_2 > 0$。可解出$a = \left(\frac{\sqrt{y_1}+\sqrt{y_2}}{d}\right)^2$，以及相应的$x_1$和$x_2$。对于那么抛物线长度

$$l = \int_{x_1}^{x_2} \sqrt{\left(\frac{dy}{dx}\right)^2 + 1}\,dx = \int_{x_1}^{x_2} \sqrt{4a^2x^2 + 1}\,dx$$

$$= \frac{x}{2}\sqrt{4a^2x^2 + 1} + \frac{1}{4a}\ln\left(x + \sqrt{x^2 + \frac{1}{4a^2}}\right)\Bigg|_{x_1}^{x_2}$$

注意由于这道题的数据非常多，所以需要用读入优化才能通过。

时空复杂度

时间复杂度$O(n+m)$　　空间复杂度$O(n+m)$

ACM/ICPC World Finals 2003

试题编号：2003-A　　**名称**：Building Bridges

题目大意

有一个n行m列的区域，每一个格子都是一个正方形，在某些格子上有建筑物。两个有建筑物的格子如果有角接触就算相连，它们就属于同一个建筑物。要在建筑物之间修建道路，道路必须是直线，且只能在正方形的边上建造，道路可以连接建筑物。道路之间可以互相穿过，但在这种情况下这两条道路并不连通。给定一个区域的建筑情况，你要算出连接所有建筑的最小道路数。如果不可能把所有建筑物连接在一起，也要尽量使无法连接的建筑物最少。在需要的道路数相同的情况下，尽量选择道路总长度最小的方案。

$n, m \leq 50$

算法讨论

这道题显然与最小生成树有关。首先用并查集来维护每一个连通块，然后因为每一座桥都是直线，那么枚举横向的$n+1$条和纵向的$m+1$条直线。显然一座桥不会与除起点终点外的其他点相连，否则我们把这座桥拆成两座，那么长度不会增大而且可以连接更多的点。那么我么直接计算这$n+m+2$条直线，每次将直线上相邻的两个点连边，所以这样总共就会有$O(nm)$条边。然后直接使用 Kruskal 算法计算出最小生成森林即可。

时空复杂度

时间复杂度$O(nm\log(nm))$　空间复杂度$O(nm)$

试题编号：2003-B　　**名称**：Light Bulbs

题目大意

有n盏灯和n个开关，每个开关控制一盏灯和它相邻的两盏灯。按开关会使其控制的灯关闭变打开，打开变关闭。给定n盏灯开闭的初始状态和目标状态，问按哪些开关能使按的开关数最小，如果相同需要使得开关编号序列的字典序最小。

$n \leq 1000$

算法讨论

设a_i表示第i盏灯需要操作的次数的奇偶性，x_i表示第i个开关是否需要按下。

之后可以列出方程$\begin{cases} x_1 \hat{} x_2 = a_1 \\ x_{i-1} \hat{} x_i \hat{} x_{i+1} = a_i \ (1 < i < n) \\ x_{n-1} \hat{} x_n = a_n \end{cases}$，然后我们枚举$x_1$就可以用前$n-1$个式子计算出所有的$x_i$，然后用最后一个式子就可以判断解是否存在。若存在，则取符合题目要求的解即可。

时空复杂度

时间复杂度$O(n)$　　空间复杂度$O(n)$

试题编号：2003-F　　**名称**：Combining Images

题目大意

递归定义一张颜色仅有 0、1 的图的四分树编码：若颜色都相同，编码为 1+颜色，若颜色不相同，编码为 0+左上编码+右上编码+左下编码+右下编码，边长保证为2的幂次。给定两张大小相同的图的四分树编码，求这两张图的并的四分树编码。

编码长度$n \leq 100 * 4$

算法讨论

首先将给定编码对应的树递归构建，然后递归构建新树。对于每一个节点，如果原先有一个为全 0，那么新树的这一部分也是全 0，如果原先有一个为全 1，那么新树的这一部分与另一棵树完全相同。如果两种情况都没有，就直接递归到子树。在返回时要注意判断子树是否完全相同，如果相同则要将其合并。

时空复杂度

时间复杂度$O(n)$　　空间复杂度$O(n)$

试题编号：2003-H　　**名称**：A Spy in the Metro

题目大意

有n个车站，相邻两个编号的车站之间有一条双向边，地铁行驶的时间已知。有m_1辆车从1开到n，给定它们的出发时间，有m_2辆车从n开到1，给定它们的出发时间。Maria 一开始在1点，要正好在 T 时刻在n点，每一时刻可以停在当前车站，或者搭乘任意一辆正好停靠在车站的地铁。问能不能在T时刻到达车站n，如果能，最小的等待时间是多少。

$n, m_1, m_2 \leq 50$，$T \leq 200$

算法讨论

这道题可以用动态规划求解。用$f[i][j]$表示在时刻i位置在站点j时所需的最少等待时间，用$a[i]$表示从第$i-1$个站点到第i个站点所需的时间$s[i]$表示从第一个站点到第i个站点所需时间。那么列出方程 $f[i][j] = min(f[i-1][j], f[i-a[j]][j-1]${如果时刻$i-s[j]$有车从1号站点发出}，$f[i-a[j+1]][j+1]${如果时刻$i-s[n]+s[j]$有车从$n$号站点发出})。答案就是$f[T][n]$的值。

时空复杂度

时间复杂度$O(nT+m)$ 空间复杂度$O(nT+m)$

试题编号：2003-I **名称**：The Solar System

题目大意

给出太阳系中一个行星的描述（就是说行星轨道的半长轴长度，半短轴长度，和它的环绕周期）以及第二个行星的描述（它的长轴和短轴）。假设第二个行星的轨道关于坐标轴对称（即中心在原点，焦点在x轴上），它沿逆时针方向运动，并且太阳位于x坐标非负的焦点处。你要计算第二个行星从轨道上的x坐标最大处（上图中的点B）开始，在经过一个指定的时间后到达的位置。

答案保留三位小数。

算法讨论

首先用开普勒第三定律可以算出第二个行星的运行周期$t_2 = t_1 * \left(\frac{a_2}{a_1}\right)^{\frac{3}{2}}$。然后用那么根据开普勒第二定律可以算出面积速度$V_s = \frac{\pi ab}{t_2}$。然后用极坐标来表示这个椭圆即$\rho = \frac{ep}{1+e\cos\theta}$，对椭圆面积积分$S = \left|\int_\alpha^\beta \left(\frac{ep}{1+ecos\theta}\right)^2 * \frac{1}{2} sind\theta\right| = \frac{e^2p^2}{2}\left|\int_\alpha^\beta \frac{1}{(1+ecos\theta)^2} d\theta\right|$。

对于计算$\int \frac{1}{(1+e\cos\theta)^2} d\theta = \frac{e\sin\theta}{(e^2-1)(e\cos\theta+1)} - \frac{2\tanh^{-1}\frac{(e-1)\tan\frac{\theta}{2}}{\sqrt{e^2-1}}}{(e^2-1)^{3/2}}$。因为在椭圆中$e \in [0,1)$，使$\frac{(e-1)\tan\frac{\theta}{2}}{\sqrt{e^2-1}} = di$，所以可计算得$2\tanh^{-1} di = \ln\frac{1+di}{1-di} = \ln\left(\frac{1-d^2}{1+d^2} + \frac{2d}{1+d^2} i\right) = \ln(cos\varphi + sin\varphi i) = \ln(e^{\varphi i})$，所以我们使$\varphi \in [0,2\pi]$。当$\varphi \in [0,\pi)$时$\varphi =$

$\cos^{-1}\frac{1-d^2}{1+d^2}$，当$\varphi \in (\pi, 2\pi)$时，$\varphi = 2\pi - \cos^{-1}\frac{1-d^2}{1+d^2}$。

所以我们开始将指定时间t减去t_2的整数倍使得$t < t_2$,然后将t与$\frac{t_2}{2}$比较判断该点在椭圆的上半部分还是下半部分，同时计算出所需面积$S = v_s t$。若是上半部分，取$\alpha = 0$，使β在$(0, \pi)$范围二分判断位置。若是在下半部分取$\alpha = \pi$，使β在$(\pi, 2\pi)$范围二分。得到较为精确β后，那么点的坐标即为$(c + cos\beta, sin\beta)$。同时防止输出-0.000这种情况，在输出时答案应加上eps。

时空复杂度

时间复杂度$O(\log\frac{\pi}{eps})$　　空间复杂度$O(1)$

试题编号：2003-J　　**名称**：Toll

题目大意

把一些货物从起点运到到终点。起点、终点和旅途中的任意一点都是城镇或村庄。进入一个村庄需要交纳一件货物，进入一个城镇时每携带20件货物需要交纳1件货物，不满20件的部分当20件计算，离开一个城镇或村庄不需要交纳任何货物。给定起点S、终点T，所有点的连接情况以及每个点是城镇还是村庄，求在起点时至少要有多少件货物才能保证选择合适的路线能使到达终点的时候还至少有p件货物。

$n \leq 52$

算法讨论

显然这是一道最短路题。我们可以从终点开始倒推，设起点为s，终点为 t，第i个点到终点所需货物为d_i,设开始时$d_t = p$,其余$d_i = \infty$。那么对于与点i相连的点 j，如果点i为村庄那么就可以用$d_i + 1$去更新d_j。如果点i为城镇就用$d_i + \left\lfloor\frac{d_i-1}{19}\right\rfloor + 1$去更新$d_j$。显然这个值是从终点开始不断增大，那么我们可以直接用 Dijkstra 算法进行计算，那么d_s就是最后的答案。

时空复杂度

时间复杂度$O(n^2)$　空间复杂度$O(n^2)$

ACM/ICPC World Finals 2002

试题编号：2002-A　　**名称**：Balloons in a Box

题目大意

给定三维空间中的一个长方体盒子，和n个点。每个点都可以放置一个气球，气球会一直均匀膨胀直到碰到其他气球或是盒子表面。你可以选择这n个点的顺序，然后求出盒子中空余的最小体积。

$n \leq 6$

算法讨论

枚举气球放置的顺序，然后求出每个气球的半径，从总体积中减去。最后在所有求出值中取最小值即可。

注意点可能在长方体外，需要特判。

时空复杂度

时间复杂度$O(n! * n2)$　空间复杂度$O(n)$

试题编号：2002-C　　**名称**：Crossing the Desert

题目大意

平面上有n个点，1号点可以得到食物，每个点可以储存食物和得到无限的水。一个人所携带的水和食物和的上限为m。每行走1单位距离消耗1单位的水和食物。求你从1号点出发，到达n号点所需的最少食物。

$n \leq 20$

算法讨论

我们设$f[i]$表示从i号点到n至少需要多少食物,用$d[i][j]$表示点i到j的距离。我们可以从n号点开始倒着转移，显然$f[n] = 0$。对于每个点i，设它的前一步为j，那么如果j可以从i处转移，至少从j可以到i，即$m \geq 2 * d[i][j]$。然后我们求从j出发至少需要多少食物才能使i处堆积$f[i]$食物。首先由于水可以在任一点无限得到，所以我们只需携带$d[i][j]$的水。那么如果从j到i不再返回，i 点可获得$m - 2 * d[i][j]$的食物，同时消耗$d[i][j]$的食物，如果从 j 到 i 往返运输食物，那么每次可获得$m - 3 * d[i][j]$的食物，同时消耗$2 * d[i][j]$的食物，由于最后一次一定不再返回，那么我们可以直接里列出方程：

$f[j]=\min(f[i]+\left(\left\lceil\frac{f[i]-(m-2*d[i][j])}{m-3*d[i][j]}\right\rceil*2+1\right)*d[i][j])$。

同时能够来回运输的前提是$m>3*d[i][j]$。这样我们我们可以用 spfa 直接计算出每个$f[i]$，那么$f[1]$即是答案。

时空复杂度

时间复杂度$O(kn^2)$ 空间复杂度$O(n)$

试题编号：2002-E **名称**：Island Hopping

题目大意

给定平面上n个岛屿的坐标和岛上的人数。将所有岛用电缆相连形成一个最小生成树。所有的电缆同时开始修建，单位时间内可以修建单位长度的电缆，问所有人与第一个岛连接的平均等待时间。

$n\leq 50$

算法讨论

显然，根据最小生成树的环割性质，对于每一种最小生成树，每个岛的等待时间相同。那么直接用 prim 算法任意构造出一个最小生成树，同时计算每个岛的等待时间，然后求带权平均即可。

时空复杂度

时间复杂度$O(n^2)$ 空间复杂度$O(n)$

试题编号：2002-G **名称**：Partitions

题目大意

一个矩形的划分是指把一个矩形分成若干个较小的、不重叠的子矩形。如果矩形B可以通过划分A的一个或多个子矩形得到，就说B比A更精细，A比B更粗糙，这一关系是偏序的。给定同一矩形的两种划分A和B，存在无穷多的矩形比它们都要精细，但这些划分中存在唯一最粗糙的划分，称之为A和B的下确界。同样地，比A和B都粗糙的划分中最精细的称为A和B的上确界。给定A和B，求出它们的下确界和上确界。

AB的长宽$w,h\leq 20$。

算法讨论

A和B的上确界非常简单，即直接将两个图形求并即可。对于下确界，首先对A和B两个图形求交，然后我们可以发现，有些边是不符合要求的。显然除了图形的边界，字符'|'的上下应为两个'_'或一个'|'，字符'_'左右应为两个'|'或一个'_'。如果不是那么就将这个图形改为空格。

由于可能开始符合要求的一个字符在某次删除一个字符后就不满足要求，那么在之前的操作中就不会将其删除。所以这个操作要进行多次才能将所有多出的字符全部删除。这样知道没有字符可以删除为止输出所剩图形即可。

时空复杂度

时间复杂度$O(kwh)$ 空间复杂度$O(wh)$

k为操作次数。

试题编号：2002-H　　**名称**：Silly Sort

题目大意

给定一个数字各不相同的序列，将其排序成升序序列。在排序过程中，可以交换两个数的位置。每次交换需要一个代价，代价是被交换的两个数的和。求排序的最小代价。

$n \leq 1000$

算法讨论

首先将这个序列排序，由于数字各不相同，所以就可以计算出每个数对应的位置。由于要求最小代价，所以必然在序列的置换中进行交换。那么对于每一个置换，设其长度为l，所有数的总和为s。有两种情况：

1、找出这个置换中最小的数，设其为t，那么显然我们每次都将t交换，共进行$l-1$次，代价为$s-t+(l-1)*t=s+(l-2)*t$。

2、找出全局最小值，设其为w，将w和t交换，然后每次都将w交换，进行$l-1$次后，然后将w和t换回，代价为$s-t+(l-1)*w+2*(w+t)=s+t+(l+1)*w$。

然后对每个置换求出最小代价，相加即可。

时空复杂度

时间复杂度$O(nlogn)$　　空间复杂度$O(nlogn)$

ACM/ICPC World Finals 2001

试题编号：2001-A　　**名称**：Airport Configuration

题目大意

有一个机场，北面是n个起始点，南面是n个目标点，所有的起始（目标）点都在同一水平线上，且相邻两个的水平距离是1，这两条水平线的距离也为1。已知每个起始点有几批旅客，他们分别去哪个目标点，有多少客流量，规定这些旅客只能水平或者垂直移动。现在有若干个机场分布方案，依次给出每个起始点和目标点，求出这些方案下的客流指数，并升序输出。一批旅客的客流指数为人数乘移动距离，一个方案的客流指数为所有旅客的客流指数之和。

$n < 25$

算法讨论

开始用邻接矩阵存贮每对城市间的客流量。对于每一个方案，直接枚举每对城市，然后把距离乘以客流量相加来计算客流指数。然后按客流指数和序号排序输出即可。

时空复杂度

时间复杂度$O(n^2)$　空间复杂度$O(n^2)$

试题编号：2001-B　　**名称**：Say Cheese

题目大意

给定一个三维空间中的起点和终点，和n个球形的孔。已知在孔中，移动不需要时间，在孔外，每移动1的距离需要$20s$。求起点到终点所需时间。

$n \leq 100$

算法讨论

将所有的孔（起点终点当做是$r = 0$的孔）看做一个点，那么两个孔i，j的距离就是$max(dis(i,j) - r_i - r_j, 0)$，然后计算起点到终点的最短路即可。

时空复杂度

时间复杂度$O(n^2)$　空间复杂度$O(n)$

试题编号：2001-F　　**名称**：A Major Problem

题目大意

在西方音乐中，用字母 A 到 G 和升调符号#和降调符号 b 组成的21个音符组成了以下的序列：C/B# C#/Db D D#/Eb E/Fb F/E# F#/Gb G G#/Ab A A#/Bb B/Cb。用/隔开的两个音符相同。一个大调音阶是从一个音符开始，在序列上间隔为$2-2-1-2-2-2$的音符中选择一个，要求大调音阶中的七个音符包含的大写字母不能重复，而且不能同时出现#或 b。给出两个大调音阶的开头，问这两个大调音阶是否存在，如果存在，输入一些音符，判断这个音符是否在第一个大调音阶中，如果在，输出第二个大调音阶上对应位置的音符。

组数$T \leq 100$

算法讨论

首先根据规则将所有音符对应的大调全部用搜索预处理出来。然后入读数据，在线计算即可。

时空复杂度

时间复杂度$O(1)$　　空间复杂度$O(1)$

试题编号：2001-H　　**名称**：Professor Monotonic's Network

题目大意

一个比较网络由若干个含两个输入端和两个输出端的比较器组成，一个比较器会比较它的两个输入端的值i_1和i_2，把它们放置在输出端o_1和o_2上，满足$o_1 \leq o_2$。一个比较网络由n个输入端和n个输出端。对于每个比较器，它的输入端要么是连在比较网络的输入端，要么连在某个比较器的输出端上。比较器之间的关系不会形成环。一个比较器工作需要1单位时间，且在输入端已被计算出来才能工作。一个比较网络被称为排序网络，当且仅当无论输入端的值如何，输出端的值都单调递增。给定一个n个输入值，k个比较器的比较网络，判断这个比较网络是不是排序网络，并求出这个比较网络的运行时间。

$n \leq 12$，$k \leq 150$

算法讨论

对于第一问，根据排序网络的$0-1$原理，使输入数值均为0或1，然后将所有

可能的输入模拟一遍，判断这个网络能否将每一组数值排序。如果可以那么此网络为排序网络，否则不是。证明参见《算法导论》。

对于第二问，从1到n枚举，记录每个数值最后被计算的时间点 t，那么第$\underline{i}$个比较器的两个数值x_i与y_i，其时间$t[x_i]$和$t[y_i]$可更新为$max(t[x_i], t[y_i]) + 1$。然后输出t中最大值即可。

时空复杂度

时间复杂度$O(k * 2^n)$　　空间复杂度$O(n + k)$

ACM/ICPC World Finals 2000

试题编号：2000-A　　**名称**：Abbott's Revenge

题目大意

给定一个$9*9$的迷宫，每个路口用二维坐标(x,y)表示，左上角为$(1,1)$，右下角为$(9,9)$。从每个方向进入一个路口都有一些方向可以继续走：直走，向左转再走，向右转再走，也有可能没有。给定起点的坐标和起点出去的方向（这一步就算起点就是终点也要走），求到终点的最短路径，并输出其中一种方案。

算法讨论

由于要求最短路径，用$f[i][j][k]$表示到达从点(i,j)出发，沿着方向k行走这种状态所需要的最小步数。由于边长为1，所以直接用 bfs 即可。

时空复杂度

时间复杂度$O(9*9*4)$ 空间复杂度$O(9*9*4)$

试题编号：2000-B　　**名称**：According to Bartjens

题目大意

给定一串长度为n的数字，你要在这串数字中插入至少一个+、-、*，使得这个算式没有多余的前导 0，没有负号，且按照正常的优先级计算的结果为2000。把所有可行的算式按字典序依次输出。

$n \leq 9$

算法讨论

直接枚举每个数字之间的符号，判断算式结果是否为2000，然后将所有可行解按字典序输出即可。

时空复杂度

时间复杂度$O(4^n)$　空间复杂度$O(n)$

试题编号：2000-C　　**名称**：Cutting Chains

题目大意

有n个链环，它们之间存在着一些连接情况，比如x和y两个链环相互嵌套在一起。每次打开一个链环后可以把嵌套在这个链环里的链环拿出来，也可以放一些新的链环进去，然后再把这个链环关上。求最小需要打开几次链环才能使得这个链环形成一条链。

$n \leq 15$

算法讨论

首先很显然每一个链环只会被打开一次。那么枚举每一个链环是否要被打开，那么对于每一组方案，首先假设将所有的与拆开链环的相连的链环取出，那么剩下的链环必然是单个或者是一条链。然后我们将剩下的链环重新连接，显然只要用拆开的链环来连接为拆开的链环即可。我们设拆开了x个链环，没有拆开的链环分成了y个单点或链，那么当且仅当$x \geq y - 1$时x可以作为一个解。然后在所有解中找出最小值即可。

时空复杂度

时间复杂度$O(n^2 * 2^n)$　空间复杂度$O(n^2)$

试题编号：2000-E　　**名称**：Internet Bandwidth

题目大意

给定一张n个点m条边的无向图，每条边都有一个流量上限，求从起点s到终点t的最大流量。

$n <= 100$，$m \leq \frac{n(n-1)}{2}$

算法讨论

直接套用最大流算法即可。

时空复杂度

时间复杂度$O(n^2 m)$ 空间复杂度$O(n + m)$

试题编号：2000-F　　**名称**：Page Hopping

题目大意

有一些编号为$1\sim100$的网页，它们之间存在着一些跳转关系，形如x可以通过一次点击跳转到y，保证从任意一个网站开始，通过点击能跳转到其他的所有网站。求网站之间最短距离的平均值。

算法讨论

将每个网页看成一个点，每个跳转看成边权为1的有向边。然后直接使用 floyd，求出每对点的最短路。那么平均值即为每对点的最短路总和除以点的对数即可。

时空复杂度

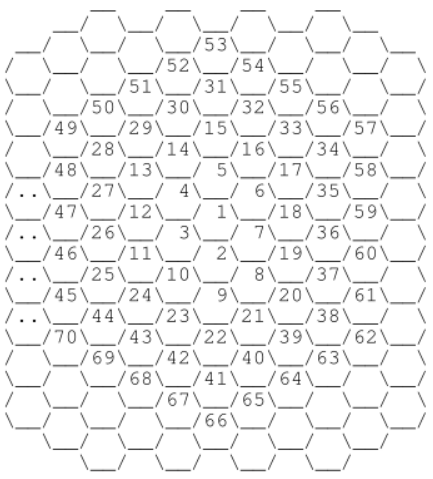
时间复杂度$O(n^3)$　空间复杂度$O(n^2)$

ACM/ICPC World Finals 1999

试题编号：1999-A　　**名称**：Bee Breeding

题目大意

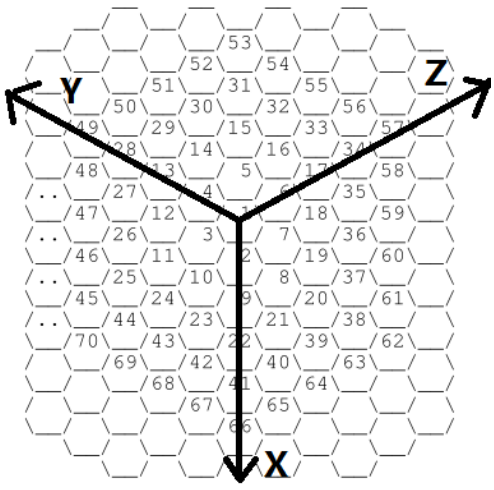
每个蜂房和相邻的六个蜂房相连，编号如图所示。每次询问两个编号的蜂房之间的最短路。



编号的最大值$n \leq 10000$

算法讨论

首先对图形建立一个坐标系：



那么每个位置都可以用三维坐标表示，并且坐标(x,y,z)等价于$(x+a,y+a,z+a)$。因此每个点的坐标不唯一。首先我们对图按照编号方式遍历一遍，将点$1\sim10000$所对应一个坐标预都处理出来。然后对于每个询问p和q，求出其坐标差为(x,y,z)，显然走$|x|+|y|+|z|$步必然能够从p走到q，这样的话，所有找出(x,y,z)等价坐标中所有坐标绝对值和最小的一个即可。

可以发现当x,y,z的中间值为0时达到最小值。证明很简单：当中间值大于0时，每个坐标减1会使至少两个坐标的绝对值减1，至多一个坐标的绝对值加1，总体和必定减小，中间值小于0时同理。这样答案就是$\max(x,y,z)-\min(x,y,z)$即可。

时空复杂度

时间复杂度$O(n+q)$　　空间复杂度$O(n)$

试题编号：1999-C　　**名称**：A Dicey Problem

题目大意

有一个标准的骰子和一个$n*m$的地图，地图的每一个格子都可能有一个数字或者一颗星星或者不存在。骰子放在地图的表面，底面正好与一个格子重合，骰子可以向水平和垂直共四个放行移动，前提是目的地是星星或者目的地上标着的数字和当前骰子顶面的数字相同。给定一个地图，骰子的起点坐标，骰子的顶面和面向地图下方的面，求一条从起点出发最后回到起点的路径，保证这样的路径要么不存在要么唯一。

$n,m\leq 10$

算法讨论

首先对骰子进行预处理，计算出当顶部数字为i时，数字j的左边数字$l[i][j]$和右边数字$r[i][j]$。

由于方案唯一，所以用$f[i][j][k][l]$表示在坐标(i,j)处顶部数字为k看到使得数字为 l 的状态是否能够到达。同时注意开始情况不算能够到达。然后直接搜索即可。

时空复杂度

时间复杂度$O(nm)$　空间复杂度$O(nm)$

试题编号：1999-E　　**名称**：Trade on Verweggistan

题目大意

给你 n 堆货物，每堆货物有b_i个箱子依次叠放，每个箱子有一个价格，而且要买一个箱子必须先买在这之上的箱子。每一个箱子值10元，问你最多能有多少利润，同时输出在最大利润的情况下买到的箱子的数量，输出最小的t个。

$n \leq 50$，$b \leq 20, t \leq 10$

算法讨论

显然每堆箱子是独立的，我们可以直接枚举每堆买到第几个箱子，算出最大值和取到最大值的箱子数目。然后总体最大值即为每堆的最大值之和，且每堆任意达到最大值的箱子数量的和即可以作为最终的箱子数量。

对于求箱子的数量，我们不能直接枚举每一堆的数量然后累加。但是，我们可以维护一个最终的答案序列s，显然s最多有t个元素。从1到n循环，到第i堆时，每次从s中选择一个元素，然后从这堆能达到最大利润的箱子数相加，存到一个临时序列中。显然这个临时序列最多有t^2个元素。这个序列排序取前t个不相同的值作为新的s序列，最后输出即可。这样后一个过程时间复杂度为$O(nt^2 \log t)$。如果这个过程用堆来维护可以做到$O(nt \log t)$。

时空复杂度

时间复杂度$O(nb + nt2\log t)$或$O(nb + nt\log t)$　空间复杂度$O(nb + t^2)$或$O(nb + t)$

试题编号：1999-H　**名称**：Flooded

题目大意

一块分成$m * n$格的地段，每格大小为 $10M \times 10M$。给定每格的高度高度。由于高出的水会向低处流，雨水会首先积在最低高度的区域中。假定在较高的区域中的积水（即使完全被更高的区域所包围）能完全排放到较低的区域中，并且水不会被地面吸收。

给定降雨量S，求积水的高度和该地段完全被淹没的区域的百分比（指该地段中高度严格低于积水高度的区域的百分比）。

$m, n \leq 30$

算法讨论

由于水的位置只与每格的高度有关，与位置无关，所以直接可以将这$m * n$个数排序。然后直接从小到大扫描。设这$n * m$个数为h_i，水位开始在h_1位置，当第i块已经被淹没。设$vi = v -$水涨到h_i所用去的体积，显然$v_1 = v$。将v_i与$i * (h_{i+1} - h_i)$比较，如果$v_i \leq i * (h_{i+1} - h_i)$，那么说明水无法淹没第$i + 1$块土地，跳出循环，否则说明第$i + 1$块土地被淹没，使$v_{i+1} = v_i - i * (h_{i+1} - h_i)$。这样不断枚举直到中途停下或者所有土地都被淹没为止。设最后被淹没的一块土地为编号为x，那么最终淹没百分比为$\frac{x}{nm} * 100\%$，水的高

度为$h_{\mathrm{x}}+\frac{v_x}{100x}$。

注意当$v=0$时，淹没百分比应该为0.00%

时空复杂度

时间复杂度$\mathrm{O}(mn\log(mn))$ 空间复杂度$\mathrm{O}(mn)$

ACM/ICPC World Finals 1998

试题编号：1998-B　　**名称**：Flight Planning

题目大意

一次飞行有n个航段，每一个航段都可以选择最佳的飞行高度，来减少飞行过程中消耗的燃料。飞机在海拔30000英尺的高度飞行的时候，每小时消耗燃料2000加仑，每比30000英尺高或低1000英尺，每小时消耗的燃料都会额外增加10加仑。飞机在起飞前、降落后的高度均为0。飞机每升高1000英尺，就会消耗50加仑的燃料（下降不会消耗燃料），改变高度的动作是瞬间完成的。在没有风影响的情况下，飞机的速度为400海里/小时，风速由一个关于飞行高度的线性函数决定，并给定每个航段20000英尺处的风速和40000英尺处的风速。飞机的飞行高度必须在20000~40000英尺之间，且必须是1000英尺的整数倍。求最小燃料消耗量，并输出方案，如果消耗相同，输出字典序最小的方案。

$n \leq 100$

算法讨论

显然我们可以用 dp 来解决这个问题。首先我们可以将飞行高度替换为$20-40$。由于我们要求字典序最小的方案，所以我们可以倒着递推，用$s[i][j]$表示在第i个航段高度为j时当前航段的耗油量，$f[i][j]$表示在第i个航段高度为j时的从第i个航段到第n个航段最小耗油量和。那么 $f[i][j] = min(f[i+1][k] + max(k-j,0)*50) + s[i][j]$。显然$f[0][0]$即为最后的最少耗油量。

对于计算方案，我们可以在转移时记录这个状态从哪个状态转移而来，然后对于相同的情况则选择高度小的那个。然后从末状态开始递推逐个输出即可。

时空复杂度

时间复杂度$O(n*21^2)$　空间复杂度$O(n*21)$

试题编号：1998-E　　**名称**：Petri Net Simulation

题目大意

给定n个库所以及这些库所现有的令牌。在这个 Petri 网中存在m个变迁，一个变迁有至少一个输入库所和一个输出库所，当一个变迁发生后每个输入库所都会减少一个令牌，每个输出库所都会增加一个令牌（如果一个库所出现多次的话会减少或增加多次），但一个变迁发生的条件是每个输入库所至少要有一个令牌（如果出现多次的话必须要有出现次数个令牌）。如果有多个变迁同时被允许，任意一个都可能发生。求能不能发生t次变迁，如果不能，最多能发生几次变迁，

以及当前有令牌的库所编号和拥有的令牌数。数据保证输出唯一。

$n, m \leq 100$，每个 Petri 网的所有变迁输入的整数序列的总长度s不超过20000。

算法讨论

由于保证输出唯一，所以对于每一次变迁可以直接枚举每个能够发生的变迁，任意选择一个进行操作即可。这样不断模拟到到达t次或是没有能够生的变迁为止。

时空复杂度

时间复杂度$O(st)$　空间复杂度$O(n+s)$