

IOI2014 国家集训队第一次作业试题泛做部分

ACM/ICPC World Finals 赛题选解

广东实验中学 黄施霖

Contents

1	Volume I	4
1.1	2013 A: Self-Assembly	4
1.2	2013 B: Hey, Better Bettor	4
1.3	2013 C: Surely You Congest	6
1.4	2013 D: Factors	6
1.5	2013 E: Harvard	7
1.6	2013 F: Low Power	7
1.7	2013 H: Matryoshka	8
1.8	2013 I: Pirate Chest	8
1.9	2013 J: Pollution Solution	9
1.10	2013 K: Up a Tree	9
2	Volume II	11
2.1	2012 A: Asteroid Rangers	11
2.2	2012 B: Curvy Little Bottles	11
2.3	2012 C: Bus Tour	12
2.4	2012 D: Fibonacci Words	12
2.5	2012 E: Infiltration	13
2.6	2012 G: Minimum Cost Flow	13
2.7	2012 L: Takeover Wars	14
3	Volume III	15
3.1	2011 A: To Add or to Multiply	15
3.2	2011 C: Ancient Messages	15
3.3	2011 F: Machine Works	16
3.4	2011 H: Mining Your Own Business	16
3.5	2011 I: Mummy Madness	17

3.6	2011 J: Pyramids	17
3.7	2011 K: Trash Removal	18
4	Volume IV	19
4.1	2010 B: Barcodes	19
4.2	2010 C: Tracking Bio-bots	19
4.3	2010 D: Castles	20
4.4	2010 E: Channel	20
4.5	2010 G: The Islands	21
4.6	2010 I: Robots on Ice	22
4.7	2010 J: Sharing Chocolate	23
5	Volume V	24
5.1	2009 A: A Careful Approach	24
5.2	2009 B: My Bad	24
5.3	2009 D: Conduit Packing	24
5.4	2009 E: Fare and Balanced	25
5.5	2009 F: Deer-Proof Fence	25
5.6	2009 G: House of Cards	26
5.7	2009 H: The Ministers' Major Mess	26
5.8	2009 J: Subway Timing	26
6	Volume VI	28
6.1	2008 A: Air Conditioning Machinery	28
6.2	2008 B: Always an Integer	28
6.3	2008 E: Huffman Codes	29
6.4	2009 F: Glenbow Museum	29
6.5	2008 G: Net Loss	30
6.6	2008 I: Password Suspects	30
6.7	2008 J: The Sky is the Limit	31
6.8	2008 K: Steam Roller	31
7	Volume VII	32
7.1	2007 A: Consanguine Calculations	32
7.2	2007 H: Marble Game	32
7.3	2007 I: Water Tanks	32
7.4	2006 A: Low Cost Air Travel	33
7.5	2006 B: Remember the A La Mode!	33
7.6	2006 D: Bipartite Numbers	34
7.7	2006 E: Bit Compressor	34

7.8	2006 G: Pilgrimage	35
7.9	2006 I: Degrees of Separation	36
7.10	2006 J: Routing	36
8	Volume VIII	37
8.1	2005 C: The Traveling Judges Problem	37
8.2	2005 H: The Great Wall Game	37
8.3	2005 I: Workshops	37
8.4	2005 J: Zones	38
8.5	2004 H: Tree-Lined Streets	38
8.6	2003 A: Building Bridges	39
8.7	2003 B: Light Bulbs	39
8.8	2003 J: Toll	40
9	Volume IX	40
9.1	2002 A: Balloons in a Box	40
9.2	2002 E: Island Hopping	40
9.3	2002 G: Partitions	41
9.4	2002 H: Silly Sort	41
9.5	2001 A: Airport Configuration	42
9.6	2001 B: Say Cheese	42
9.7	2000 B: According to Bartjens	42
9.8	2000 E: Internet Bandwidth	43
9.9	2000 F: Page Hopping	43
10	Volume X	44
10.1	1999 H: flooded!	44
10.2	1998 A: Crystal Clear	44
10.3	1998 B: Flight Planning	45
10.4	1998 D: Page Selection by Keyword Matching	45
10.5	1998 E: Petri Net Simulation	46
10.6	1998 G: Spatial Structures	46

1 Volume I

1.1 2013 A: Self-Assembly

题目大意

一个结构体由若干个在同一平面的分子组成，每个分子的形状为正方形，正方形的四条边上有连接标识，表示这条边可以与另一个分子的哪种边相邻。连接标识有两种：一种由一个大写字母加上正负号组成，另一种由两个 0 组成，表示该边不能与其他分子的任何边相邻。两条边可以相邻当且仅当它们的字母相等且符号相反。分子可以旋转和翻转，且每条边可以不与其它分子的任何边相邻。给定 n 种分子，每种分子的数目有无限多个，试判断它们是否可能组合成一个无限大的结构体。

算法分析

将每种分子抽象成一个点，一个分子若可以与另一个分子相邻，则将它们对应的点连一条无向边，形成一个无向图。可以证明，当图中存在环时，可以构造出一个无限大的结构体。

那么我们可以对该图进行 DFS 以判断是否存在环。但构造该图的时间复杂度为 $O(n^2)$ ，对于 $n \leq 40000$ 的数据范围，显然无法承受。注意到边的种类只有 $m = 26$ 种，我们可以将原图进行变换，即将边变成点，点变成边，同时将相同种类的点合并，这样原图就被压缩成只有 m 个顶点的图。构图的时间复杂度为 $O(n)$ ，DFS 判断环的存在性的时间复杂度为 $O(m^2)$ ，总时间复杂度为 $O(n + m^2)$ 。

1.2 2013 B: Hey, Better Bettor

题目大意

赌场开设了一种赌博游戏，玩家赌赢了赚 1 元，赌输了要向赌场支付 1 元。玩家可以玩任意多次，且可以在玩了任意次后退出游戏。退出游戏时，若玩家总资金少了，赌场会返还玩家亏损资金的 $x\%$ ，若玩家赚了，赚来的资金可以据为己有。给定 x 和一次游戏获胜的概率 p ，试计算出玩家在最佳赌博策略下获得的收益的期望值。

算法分析

令 $\lambda = 1 - x/100$ ，容易发现，玩家每一步的决策只和玩家当前赚（亏）的资金数有关，与之前进行了多少次游戏无关。设 $f(k)$ 为玩家当前赚了 k 元（ $k < 0$ 表示亏了 $-k$ 元）时，在最佳赌博策略下获得的期望收益。容易得到如下转移方程：

$$f(k) = \max\{k, \lambda k, pf(k+1) + (1-p)f(k-1)\}$$

最优策略必然是，亏了 a 元或者赚了 b 元时退出赌博，当前赚的资金在 $(-a, b)$ 之间时，继续赌博。则有 $f(-a) = -\lambda a$ ， $f(b) = b$ ，对于任意 $-a < k < b$ 有

$$f(k) = pf(k+1) + (1-p)f(k-1)$$

将上式变形得

$$f(k+1) = \frac{1}{p}f(k) - \frac{1-p}{p}f(k-1)$$

令 $g(k) = f(k-a)$, 则 $g(0) = -\lambda a$, $g(a+b) = b$, $g(k+1) = \frac{1}{p}g(k) - \frac{1-p}{p}g(k-1)$ ($0 < k < a+b$)。设 α 和 β 为特征方程 $x^2 - \frac{1}{p}x - \frac{1-p}{p} = 0$ 的两根, 则 $g(k) = c_1\alpha^k + c_2\beta^k$, 易解得 $\alpha = 1$, $\beta = \frac{1-p}{p}$ 。分别将 $k = 0$, $k = a+b$ 代入可解得

$$g(k) = \frac{(\beta^k - 1)(\lambda a + b)}{\beta^{a+b} - 1} - \lambda a$$

显然, 答案为 $f(0) = g(a) = \frac{(\beta^a - 1)(\lambda a + b)}{\beta^{a+b} - 1} - \lambda a$ 。

设 $h(a, b) = \frac{(\beta^a - 1)(\lambda a + b)}{\beta^{a+b} - 1} - \lambda a$ 。 $h(a, b)$ 关于 b 的偏导为:

$$\frac{\partial h}{\partial b} = \frac{\beta^a - 1}{(\beta^{a+b} - 1)^2} \left\{ \beta^{a+b} [1 - (\lambda a + b) \ln \beta] - 1 \right\}$$

显然 $\frac{\beta^a - 1}{(\beta^{a+b} - 1)^2} > 0$, 记 $u(a, b) = \beta^{a+b} [1 - \ln \beta (\lambda a + b)] - 1$, 有

$$\frac{\partial u}{\partial b} = -\beta^{a+b} (\ln \beta)^2 (\lambda a + b) < 0$$

故 $u(a, b)$ 关于 b 单调递减, 且 $\forall a, \exists b_0 \geq 0$, 当 $b > b_0$ 时, $u(a, b) < 0$ 。由此可以推出当 a 确定时, $h(a, b)$ 为关于 b 的单峰函数。

$h(a, b)$ 关于 a 的偏导为:

$$\frac{\partial h}{\partial a} = \frac{\beta^a (\beta^b - 1)}{(\beta^{a+b} - 1)^2} \left[(a\lambda + b) \ln \beta - \lambda (\beta^{a+b} - 1) \right]$$

显然 $\frac{\partial h}{\partial a} = \frac{\beta^a (\beta^b - 1)}{(\beta^{a+b} - 1)^2} > 0$, 设 $v(a, b) = (a\lambda + b) \ln \beta - \lambda (\beta^{a+b} - 1)$,

$$\frac{\partial v}{\partial a} = \lambda (1 - \beta^{a+b}) \ln \beta < 0$$

故 $v(a, b)$ 关于 a 单调递减, 且 $\forall b, \exists a_0 \geq 0$, 当 $a > a_0$ 时, $u(a, b) < 0$ 。由此可以推出当 b 确定时, $h(a, b)$ 为关于 a 的单峰函数。

设 $\phi(a) = \max \{h(a, b)\}$, 由上述讨论, $\phi(a)$ 为单峰函数。由于本题中 a, b 都是整数, 故我们可以二分查找出最小的 a_0 使得 $\phi(a_0 + 1) - \phi(a_0) < 0$, $\phi(a_0)$ 即为所求, 计算 $\phi(a)$ 时, 二分查找出最小的 b_0 使得 $h(a, b_0 + 1) - h(a, b) < 0$, 则 $\phi(a) = h(a, b_0)$ 。计算 $h(a, b)$ 的时间复杂度为 $O(\log C)$ (C 为 a 和 b 二分的上界)。总时间复杂度为 $O(\log^3 C)$ 。

下面讨论 C 的取值。显然当 $h(a, b)$ 取到极大值时, 有 $u(a, b) = 0$, $v(a, b) = 0$ 。当 $v(a, b) = 0$ 时, $(a\lambda + b) \ln \beta = \lambda (\beta^{a+b} - 1)$,

$$a = \log_{\beta} \frac{(a\lambda + b) \ln \beta - \lambda}{\beta^b} < \log_{\beta} [(a\lambda + b) \ln \beta - \lambda] < \log_{\beta} 10^6 = \frac{6}{\log \beta}$$

10^6 是 $(a\lambda + b) \ln \beta - \lambda$ 的一个较粗略的上界。

当 $u(a, b) = 0$ 时, $\beta^{a+b} [1 - \ln \beta (\lambda a + b)] = 1$,

$$b = \frac{1 - \lambda a \ln \beta - \frac{1}{\beta^{a+b}}}{\ln \beta} < \frac{1}{\ln \beta}$$

根据以上分析, $a \in (0, \frac{6}{\log \beta})$, $b \in (0, \frac{1}{\ln \beta})$ 。当 $\beta = \frac{5001}{4999}$ 时, $C = \frac{6}{\log \beta} \approx 34539$

1.3 2013 C: Surely You Congest

题目大意

在一个城市里有 n 个路口和 m 条连接路口的道路，有 c 个人要同时驱车从不同的路口到达城市 1。他们只会选择最短路径。如果有两辆车在同一时间沿着相同道路的同方向移动，该道路会出现堵塞。给出开车通过每条道路的时间，以及每个人出发的路口，问在不允许出现堵塞的情况下，最多有多少人能到达市中心。

算法分析

设从 u 到 1 的最短路径的长度为 $d(u)$ ，对于每条边 (u, v) ，设其长度为 $w(u, v)$ ，如果有 $d(u) + w(u, v) = d(v)$ ，那么保留这条边从 u 到 v 的方向。剩下的图一定是一个 DAG，我们构造一个对应的网络：对于 DAG 的每一条边 (u, v) ，在网络中连一条有向边 (u, v) ，容量为 1。

开始时在顶点 u 和顶点 v 的两个人，如果有 $d(u) = d(v)$ ，那么这两个人到达相同顶点的时间一定相同。将所有人按照所在点的 d 值分组，不同组的人不会相互影响，我们只需分别求出每组的最优解并累加。每组的最优解是一个单源多汇的最大流问题，考虑到网络为单位容量网络，我们只需采用最简单的 Ford-Fulkerson 增广路算法求最大流。

Ford-Fulkerson 算法每次增广的复杂度为 $O(n + m) = O(m)$ ，总增广次数为 $O(c)$ ，故总时间复杂度为 $O(mc)$ 。

1.4 2013 D: Factors

题目大意

设 $f(k)$ 表示 k 的素因子排列方案数，给定 $n(n < 2^{63})$ ，求出最小的 k ，使得 $f(k) = n$ ，数据保证 $k < 2^{63}$ 。

算法分析

设 $k = p_1^{a_1} p_2^{a_2} \dots p_m^{a_m}$ ，则

$$f(k) = \frac{(a_1 + a_2 + \dots + a_m)!}{a_1! a_2! \dots a_m!}$$

计算知 m 不超过 20，对于任意一组 $(a_1, a_2, \dots, a_m) (a_1 \geq a_2 \geq \dots \geq a_m)$ ，显然取 p_1, p_2, \dots, p_m 为前 m 个素数可使 k 最小。设 $g(n) = k(f(k) = n)$ ，我们可以暴力枚举所有的 (a_1, a_2, \dots, a_m) ，计算出这种情况下最小的 k ，并更新 $g(\frac{(a_1 + a_2 + \dots + a_m)!}{a_1! a_2! \dots a_m!})$ 。 $g(n)$ 可以用一个哈希表存储。最后用 $O(1)$ 时间处理询问。

时间复杂度较难估计，但可行的 (a_1, a_2, \dots, a_m) 非常少，故实际效率非常高。

1.5 2013 E: Harvard

题目大意

有一个单片机有 $b(b \leq 13)$ 个内存库（从 0 到 $b-1$ 编号）和一个 BSR 寄存器（初始值未定义），每个内存库有 $s(s \leq 13)$ 个数据单元。访问数据单元的指令是一个有序数对 (a, f) ，当 $a = 0$ 时访问 0 号内存库的第 f 个变量，当 $a = 1$ 时访问 BSR 寄存器所指向的内存库的第 f 个变量。另有一个修改 BSR 数值的指令。现有一个在该单片机上运行的程序，程序中有变量访问操作和简单的循环，试确定一个变量到内存库的映射，使得程序运行的指令数最少，变量不超过 13 个。

算法分析

先枚举哪些变量放在 0 号内存库中。访问一个不在 0 号库的变量 v_i 时，若最近被访问的不在 0 号库的变量 v_j 与 v_i 不在同一个库中，那么 BSR 寄存器就需要被修改。通过递归算法确定每对变量 v_i, v_j （ v_i, v_j 均不在 0 号库中）， v_i 恰好在 v_j 前一个被访问的次数。然后搜索出剩下各个变量所在的内存库并统计总操作次数。取所有方案的最优值。搜索时使用最小表示法，加上简单的最优性剪枝即可在极短的时间内通过全部测试数据。

1.6 2013 F: 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。现有 $2nk$ 个电池，已知它们的能量，要把它们放在 n 个机器上，使得所有机器两个芯片的能量差的最大值最小。

算法分析

首先将电池按照能量从小到大排序。容易发现，最优策略下，所有机器的两个芯片的能量大小，在电池能量的序列中相邻。

二分能量差的最大值 P ，将最优性问题转化为存在性问题。每次我们从小到大扫描整个序列，如果相邻两个电池的能量差不超过 P ，则将它们分别安装在新的机器的两个芯片中，直到所有的机器芯片都装上电池或发现不存在可行解为止。不存在可行解的充要条件是，前 $2mk$ 个电池恰能配对出 m 对电池，每对的能量差不超过 P ，而第 $2mk+1$ 和第 $2mk+2$ 个电池的能量差超过 P 。当上述条件发生时，电池将无法放置。

时间复杂度 $O(n \log n + n \log P_{\max})$ ， P_{\max} 为电池能量的最大值。

1.7 2013 H: Matryoshka

题目大意

有 n 个套娃排成一行，每次可以将相邻的两个套娃集合并成一个套娃集，合并时需要将两个套娃集拆开并重新组合，使得小的套娃一定在比它大的所有套娃的内部。拆开一个套娃并合上的代价是 1。一个包含 m 个套娃的完整套娃集，其所有套娃的大小从小到大排列必然为 $1, 2, \dots, m$ 。求将 n 个套娃合并为若干个完整套娃集的最小代价。

算法分析

设 $g[i, j]$ 表示将区间 $[i, j]$ 内的套娃合并成一个套娃集所需的最少操作次数（假如这个区间内有套娃大小相同，令 $g[i, j] = +\infty$ ）。那么显然

$$g[i, j] = \min \{g[i, k] + g[k + 1, j] + \text{merge}((i, k), (k + 1, j))\}$$

其中 $\text{merge}(A, B)$ 表示区间 A 和 B 内的套娃已合并为一个套娃集时，合并两个套娃集的代价。主要的问题是如何计算 $\text{merge}(A, B)$ 。

合并两个套娃时，如果对于某个大小的套娃，另一个套娃内不存在比他还小的，就不需要拆开这个娃娃。例如合并 $(1, 2, 5)$ 和 $(3, 4)$ 时，1 和 2 就不用拆开。

设 t 为 $[i, j]$ 内最小的娃娃的位置。当 $k < t$ 时，将 k 逆向枚举可以使得不需要拆开的娃娃个数单调不降。当 $k \geq t$ 时则正向枚举 k 。通过将 $[i, j]$ 区间内的娃娃有序化（维护一个支持插入删除的有序表），可以用均摊 $O(n)$ 的时间算出每种决策中不需要拆开的娃娃个数，这样计算 $g[i, j]$ 的总时间复杂度为 $O(n^3)$ 。

最后设 $f[i]$ 表示将前 i 个套娃合并成一些套娃集的最小代价，显然有

$$f[i] = \min \{f[j - 1] + g[j, i]\}$$

计算 $f[n]$ 的时间复杂度为 $O(n^2)$ 。

算法的瓶颈在于计算 $g[i, j]$ ，故总时间复杂度为 $O(n^3)$ 。

1.8 2013 I: Pirate Chest

题目大意

有一个表面为矩形的池塘，池塘四周都是竖直的悬崖，且悬崖足够高。经过调查，池塘表面可以被分为 $m \times n$ 的网格，每个方格内的所有点的水深是相同的。现要在池塘内放一个底面一边长不超过 a ，另一边长不超过 b 的长方体，当长方体放入池塘中时，长方体会下沉直至碰到池底，其边缘和网格对齐，且顶面和池塘表面平行。长方体会排开一部分水，使得水位上升。要求水位上升后，长方体顶面必须严格低于水面。求长方体的最大体积。

算法分析

先考虑一个相对容易的问题：在一个 $m \times 1$ 的池塘中放一个底面长不超过 b ，宽为 1 的长方体，求长方体的最大体积。

首先，通过数学推导可以发现，在不影响长方体底面深度的情况下，底面积越大越好。

由于池塘的宽仅为 1，池塘各格点的深度可以看作一个序列 (d_1, d_2, \dots, d_m) ，如果最终长方体的左上角在 $(p, 1)$ ，右下角在 $(q, 1)$ ，我们称 $[p, q]$ 为长方体的位置区间。

设序列 d 在 $[l, r]$ 内的最小值位置为 $i (l \leq i \leq r)$ ，考虑位置区间在 $[l, r]$ 内最优的长方体 $V(l, r)$ ，设其位置区间为 $[p, q]$ ， $[p, q]$ 有三种可能的情形：

1) $p \leq i \leq q$ ，无论底面有多大，长方体底面的深度必然为 d_i ，必然有 $q - p + 1 = \max\{b, r - l + 1\}$ 。

2) $q < i$ ，该长方体必为 $V(l, i - 1)$ 。

3) $p > i$ ，该长方体必为 $V(i + 1, r)$ 。

由上述分析我们得到了一个分治算法，时间复杂度为 $O(m \log m)$ （RMQ 的复杂度），如果通过栈进行优化（类似 RMQ 转 LCA 的过程），我们将得到一个时间复杂度和代码量都相当优秀的线性算法。

回到原问题，我们可以枚举长方体一边的边长 r ，以及底面的第一行放在池塘的第几行，增加这些约束之后，再将相邻的 r 行压缩成一行，原问题就成功转化为上面分析过的问题。

将 a, b, m 看作与 n 同阶，枚举的时间复杂度为 $O(n^2)$ ，一次计算的时间复杂度为 $O(n)$ ，故总时间复杂度为 $O(n^3)$ 。

1.9 2013 J: Pollution Solution

题目大意

求一个半圆和一个简单多边形的交，保证简单多边形在半圆的上方。

算法分析

将多边形进行三角剖分，问题转化为求三角形和圆的交。分类讨论，不同的情形有 6 种，分类的依据是原点的对边所在直线和圆的交点方程的根的分布。时间复杂度 $O(n)$ 。

1.10 2013 K: Up a Tree

题目大意

有一位程序员用递归的方法编写了求二叉树先序、中序、后序遍历的代码。但中间的递归调用出现了错误（比如在先序遍历的代码里，将递归调用先序遍历写成了递归调用中序遍历），以至于运行程序时，产生了错误的遍历序列。分别给定由该错误代码生成的二叉树先序、中序、后续遍历序列，试推断该代码的 6 个递归调用分别调用了什么遍历函数（输出所有的可能），对于每种可能的错误，推出正确的遍历序列。如果有多解，输出先序遍历最小的一组，如果仍有多解，输出中序遍历最小的一组。

算法分析

枚举 6 个递归调用分别是什么，然后进行记忆化搜索，设 $f(S_1, S_2, S_3)$ 表示错误的先序、中序、后序遍历分别为 S_1, S_2, S_3 （若 S_1, S_2, S_3 中有空串，说明对应的项为未知）时最优的二叉树（用正确的先序、中序遍历表示），如果 S_1 的首项或 S_3 的末项存在，则该树的树根确定，否则需要在 S_2 中枚举根所在的位置，取最优的树作为答案。在根确定后，如果 S_2 非空，那么根的左右子树大小也确定，否则需要枚举左子树的大小，取最优的树作为答案。

设树的大小为 n ，则时间复杂度为 $O(n^5)$ ，其中隐藏着 $\frac{6!}{(2!)^3}$ 的常数，但记忆化搜索自带了剪枝的功能，实际的工作量远达不到上界。

2 Volume II

2.1 2012 A: Asteroid Rangers

题目大意

在空间中有 n 个点，每个点都以一定的速度作匀速直线运动。任意两点之间有边相连，边权为它们的欧几里得距离，边的权值随时间发生改变。试求出这个动态图的最小生存树改变的次数。数据保证当最小生成树在 t 时刻发生改变时，在任何时刻 $s \in (t, t + 10^{-6})$ ，最小生成树不会发生改变。

算法分析

任意两条边长的平方差是一个次数不超过 2 的关于时间的多项式，在多项式的零点处，两条边长的序关系会发生改变。枚举任意两条边，通过解二次方程（或线性方程）求出它们的序关系改变的時刻，将所有的時刻加上 5×10^{-7} 并从小到大排序。依次检查这些時刻，当时间轴到达某个時刻 t ，若有一条不在最小生成树中的边的长度比一条在最小生成树中的边更小时，检查最小生成树是否发生改变。统计改变的次数即可。

最小生成树可能发生改变的次数上界为 $O(n^4)$ ，求一次最小生成树的复杂度为 $O(n^2)$ ，故此算法的时间复杂度为 $O(n^6)$ ，由于最小生成树可能改变的次数远达不到上界，该算法还是可以在规定时限内出解。

2.2 2012 B: Curvy Little Bottles

题目大意

有一个没有刻度的容量瓶，由多项式函数 $P(x)$ $x \in [x_{\text{low}}, x_{\text{high}}]$ 的图像绕 x 轴旋转一圈而成，现要在容量瓶上标记刻度（瓶底为 0 刻度），相邻刻度之间的容积差为 δ ，求出前 8 个刻度与瓶底的距离。

算法分析

对于任意的 $x_1 < x_2$ ($x_1, x_2 \in [x_{\text{low}}, x_{\text{high}}]$)， x_1 到 x_2 这一段的容积为

$$\pi \int_{x_1}^{x_2} [P(x)]^2 dx$$

寻找第 i 个标记 x_i 的位置时，在 $[x_{i-1}, x_{\text{high}}]$ 内二分找出

$$\pi \int_{x_{i-1}}^x [P(x)]^2 dx = \delta$$

的根 x ，令 $x_i = x$ 即可。

设 $P(x)$ 次数为 n ，二分迭代次数为 C ，则时间复杂度为 $O(nC)$ 。

2.3 2012 C: Bus Tour

题目大意

有 n 个旅馆（编号 $1 \sim n$ ），车辆从始发站（编号 0 ）出发，每到达一个旅馆就搭上住在该旅馆的游客，将 n 个旅馆的游客都搭上后到达旅游景点（编号 $n+1$ ），返程时每到达一个旅馆就放下住在该旅馆的乘客，将所有乘客放下后回到始发站。要求前半程上车的旅客必须在返程的前半程下车。车辆允许经过一个旅馆不停靠。给出这些地点之间的线路图，求出一条路径，使得车辆旅行的路线尽可能短。

算法分析

设 $f[i, S]$ 表示从 0 出发，终点为 i ，经过的位置集合为 S 时的最短路径长度， $g[i, S]$ 表示从 $n+1$ 出发，终点为 i ，经过的位置集合为 S 时的最短路径长度，那么答案显然为

$$\min \{f[i_1, S_1] + w[i_1, i_2] + g[i_2, S_2] + g[j_1, S_1] + w[j_1, j_2] + f[j_2, S_2]\}$$

其中， $w[x, y]$ 为 x 到 y 最短路的长度， $S_1 \cap S_2 = \emptyset, S_1 \cup S_2 = V, |S_1| \leq \lfloor n/2 \rfloor$ 。

通过动态规划预处理 f 和 g ，枚举所有可能的 S_1 ，分别枚举 i_1, i_2 和 j_1, j_2 ，将两部分的最优解累加。总时间复杂度为 $O(2^n n^2)$ ，隐含的常数较小。

2.4 2012 D: Fibonacci Words

题目大意

斐波那契串的定义如下：

$$F_n = \begin{cases} "0" & n = 0 \\ "1" & n = 1 \\ F_{n-1} + F_{n-2} & n \geq 2 \end{cases}$$

给定模式串 p 和 $n (n \leq 100)$ ，求 p 在 F_n 中的匹配次数。

算法分析

显然我们不能直接求出 F_n 。

设 $cnt[i]$ 表示 p 在 F_i 中出现的次数。我们考虑将 F_n 分解成左右两个串 F_{n-1}, F_{n-2} 。显然，我们只需要计算出 $cnt[n-1], cnt[n-2]$ 以及起始端在左串，结束段在右串的匹配的次数 $a[n]$ ，那么 $cnt[n] = cnt[n-1] + cnt[n-2] + a[n]$ 即为答案。 $a[n]$ 的计算是算法的瓶颈，设 F_k 是第一个长度不小于 p 的长度的斐波那契串，不妨构造两个串：

$$\begin{aligned} T_0 &= F_k + F_k \\ T_1 &= F_{k+1} + F_k \end{aligned}$$

设 p 在文本串 S 匹配的次数为 $m(S)$ ，经过分析不难得出：

$$a[n] = \begin{cases} 0 & n - 2 < k \\ m(T_0) - m(F_k) - m(F_k) & n - 1 \text{ 与 } k \text{ 同奇偶} \\ m(T_1) - m(F_{k+1}) - m(F_k) & \text{otherwise} \end{cases}$$

我们只需用 KMP 在 $O(|p|)$ 时间 ($|p|$ 为 p 的长度) 计算 $m(T_0), m(T_1), m(F_k), m(F_{k+1})$, 然后 $O(n)$ 递推计算 $cnt[n]$ 即可。时间复杂度 $O(|p|)$ 。

2.5 2012 E: Infiltration

题目大意

有一个 $n(n \leq 75)$ 个点的有向图, 对于任意两点 u, v , 要么 $u \rightarrow v$, 要么 $v \rightarrow u$ 。选取最少数目的点将图进行覆盖, 即使得对于每个点 v , 要么 v 被选中, 要么存在 $u(u \rightarrow v)$, u 被选中。

算法分析

该图的边数为 $\frac{n(n-1)}{2}$, 必然存在一个点 u 的出度不小于 $\frac{n(n-1)}{2n} = \frac{n-1}{2}$ 。选取 u , 将 u 及所有 $v(u \rightarrow v)$ 从原图中删除, 剩余图的性质与原图类似, 且节点数不超过原图的一半。反复选取这样的 u 点, 即可覆盖原图。因为每次都要删除超过一半的点, 故我们只要选取不超过 $\log_2 n$ 个点即可覆盖原图。

由上述分析, 答案肯定不超过 7。故我们可以采取搜索算法。将所有点按照出度排序, 优先考虑选取出度较大的点。加上最简单最优性剪枝, 即可通过全部数据。

2.6 2012 G: Minimum Cost Flow

题目大意

有一个运水系统由接口器和连接接口器的管道组成, 其中有一个接口器是入水口, 还有一个接口器是出水口。开始运水时, 水泵将在入水口施压, 使水可以到达一定的高度, 压力大小任意。如果水到达某个接口器, 水将充满接口器。如果在有水的接口器处有管道横向或向下延伸, 那么水将自动从那些管道流过。水也会沿着管道向上, 直到到达压力的高度限制。

运水系统原有的管道被拆除了一些, 在接口器上留下了洞, 如果水流入有洞的接口器, 将会发生漏水。现在要改造这个运水系统, 对于每一个洞, 可以购置新的管道将其与另一个洞连接, 代价为两个洞的欧几里得距离, 或者购置塞子将洞堵上, 代价为 0.5。试用最少的代价改造运水系统, 使在一定压力下, 水流可以到达出水口, 且不会发生漏水。

算法分析

题目中所有接口器的坐标都是整数, 因此它们之间的距离至少为 1, 由此可以推断, 洞应尽可能用塞子堵上, 连接的管道应尽可能少。

首先枚举压力令水到达的最大高度。忽略高于这个高度的管道和接口器，剩下的设备中，如果两个点有管道相连，将它们放在同一个集合。同一个集合里的点两两连边，边权为 0，不同集合的点也连边，边权为两个点的距离减去 1。看起来我们只需要用 dijkstra 求出源到汇的最短路，再加上洞的数目乘以 0.5 即可。但注意到每个接口器的洞的数目是有限的，有的接口器不一定能接两条管道。我们需要将每个点拆成入点和出点。运行 dijkstra 算法的时候，入点只能松弛同一集合的所有出点，出点只能松弛不同集合的入点。特别需要注意的是，如果某个点洞的数目小于 2，那么该点对应的入点不能松弛对应的出点。通过以上修补才能得到正确的解。最后将不同水压下的最优解取最小值即可。

运行一次 dijkstra 算法的复杂度为 $O(n^2)$ ，枚举量为 $O(n)$ ，总时间复杂度为 $O(n^3)$ 。

2.7 2012 L: Takeover Wars

题目大意

有两家公司 A 和 B，A 和 B 分别有 n 和 m 间子公司，每个子公司都有自己的市场价值。每次两个大公司可以按照 A 先 B 后的顺序，选择自己旗下的一个子公司 u ，并选择一个市场价值比 u 小的公司 v 进行吞并，若 v 在自己的旗下，那么 u 和 v 合并成一个新公司 w ， w 的价值为 u, v 价值之和；若 v 属于敌对势力，那么 v 公司将消失， u 公司的价值不会改变。当某个大公司旗下没有子公司时，该公司落败。问 A 与 B 的商业竞争中哪方会取得胜利。

算法分析

显然每次可行的策略只有两种：将自己旗下最大的两个子公司合并；吃掉对方最大的子公司。如果每一步无法吃掉对方最大的子公司，那么就必须将自己最大的两个子公司合并，以防被对方吃掉。一旦自己旗下最大的子公司被对方吃掉，且该子公司是合并而成的，那么后面己方无论怎么合并，对方只需不断吃掉最大的子公司即可，己方必败。反过来，若当且对方最大的公司是合并而成，且己方有实力吃掉它，那么将其吃掉，己方必胜。我们可以枚举 A 公司的第一步决策是合并自己的子公司还是吃对方的子公司。容易发现，后面的所有决策都是唯一的，只要模拟即可。

算法瓶颈在于排序，故时间复杂度为排序复杂度，即 $O(n \log n)$ 。

3 Volume III

3.1 2011 A: To Add or to Multiply

题目大意

有一种处理器只有两种指令：

- A 数值加 a
- M 数值乘 m

试给出一个最短的指令序列（如果存在多种方案，要求字典序最小），使得在区间 $[p, q]$ 内的所有整数经过指令序列修改后，均落在区间 $[r, s]$ 内。

算法分析

当 $m > 1$ 时，枚举 M 指令的数目 k ，数 x 经过指令序列修改后，必然能表示成 $an + xm^k$ ($0 \leq n < m^{k+1}$)。指令序列的长度为 k 加上 n 在 m 进制下的数位和。设 $u = \lceil \frac{r-pm^k}{a} \rceil, v = \lfloor \frac{s-qm^k}{a} \rfloor$ ，显然有 $u \leq n \leq v$ 。设 $u = (u_k u_{k-1} \dots u_0)_m, v = (v_k v_{k-1} \dots v_0)_m$ ， i 为 u 和 v 的 m 进制表示从高位往低位数第一个不相同的位，即 $u_k = v_k, u_{k-1} = v_{k-1}, \dots, u_{i+1} = v_{i+1}, u_i \neq v_i$ 。若 $u_{i-1} = u_{i-2} = \dots = u_0 = 0$ ，则令 $n = u$ ，否则令 n 的 m 进制的第 $i+1$ 位到第 k 位与 u 的第 $i+1$ 位到第 k 位相同，第 i 位为 $u_i + 1$ ，第 0 位到第 $i-1$ 位均为 0，用 n 的数位和加上 k 来更新答案。

当 $m = 1$ 时，显然指令序列中只有 A 指令，特判即可。

时间复杂度为 $O(\log^2 s)$ 。

3.2 2011 C: Ancient Messages

题目大意

早期文明有一种语言，该语言仅由 6 个字母组成，字母的形状如下图。






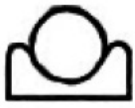
					
Ankh	Wedjat	Djed	Scarab	Was	Akeht

图1：六个象形文字

由于各种原因，字母会发生扭曲，但拓扑结构仍与上图等价。给定一个位图，试在其中找出所有的字母，保证字母不会被字母包含，且位图中的每一个黑色点必然属于一个字母。

算法分析

注意到, 6 个字母内部的洞的数目均不相同。我们可以使用 floodfill 算法找出每个黑色块内部的白洞数, 通过洞的数目来判断字母类型, 最后统计输出即可。

设位图大小为 $w \times h$, 则时间复杂度为 $O(wh)$ 。

3.3 2011 F: Machine Works

题目大意

有一家公司进入了持续 D 天的转型期, 在该时期有 n 台机器可供该公司购买, 第 i 台机器买入价格为 P_i , 卖出价格为 R_i , 每天能创造的利润为 G_i , 其仅在第 D_i 天出售。公司初始资金为 C , 在任意时刻只能拥有一台机器。求出一个方案, 使得公司在转型期内获得的利润最多。

算法分析

首先将机器按照出售时间进行排序。令 $P_0 = R_0 = G_0 = D_0 = 0, D_{n+1} = D + 1$ 。设 $f[i]$ 表示在第 D_i 天前 (含第 D_i 天) 公司所能获得的最大收益, 不难推出状态转移方程:

$$f[i] = \max \{f[j] + R_j - P_j + G_j(D_i - D_j - 1) \mid f[j] \geq P_j\}$$

边界条件 $f[0] = C, f[n+1]$ 为所求。

由此我们得出了一个效率为 $O(n^2)$ 的动态规划, 但是在本题的数据规模下, 该算法是不能达到要求的。

考虑将转移方程进行变形, 令 $k_i = G_i, b_i = f[i] + R_i - P_i - G_i(D_i + 1)$, 那么 $f[i] = \max \{k_j D_i + b_j \mid f[j] \geq P_j\}$ 。每个决策可被视作一条斜率为 k_i , 截距为 b_i 的直线。所有的决策形成一个直线簇, 我们需要快速找出当 $x = D_i$ 时, 哪条直线的 $kx + b$ 最大。事实上, 各个位置的最优决策组成了直线上方的半平面的交的轮廓线, 我们只需动态地维护这个半平面交。这个问题看似复杂, 但由于我们只需每次插入一个半平面以及查询轮廓线上某个横坐标已知的点的纵坐标, 故我们可以用线段树进行维护 (区间修改, 单点询问)。

在线段树上作一次修改和查询的复杂度均为 $O(\log n)$, 操作次数为 $O(n)$, 故总时间复杂度为 $O(n \log n)$ 。

事实上将 (k_i, b_i) 看作平面上的点, 用凸壳优化的动态规划同样也能在 $O(n \log n)$ 的时间复杂度通过此题, 且常数更小, 凸壳的维护可以用增量法 (平衡树) 或分治法。线段树解法的优势在于代码较短。

3.4 2011 H: Mining Your Own Business

题目大意

现要将一个无向图中的一些顶点染色, 使得无论删除哪个点, 对于剩下的每个点, 均有被染色的点与其连通。要求被染色的点数最少, 同时求出方案总数。

算法分析

使用 Tarjan 算法找出该图的所有点双连通分量，将这些分量缩点后形成一棵无根树，最少需要染色的点数为该无根树叶子（度数为 1 的点）的数目。将这些叶子代表的点的数目减去 1（割顶）后相乘即为方案总数。

设该无向图点的数目为 n ，边的数目为 m ，Tarjan 算法求点双连通分量的时间复杂度为 $O(n + m)$ ，寻找缩点后的树的叶子的复杂度也为 $O(n + m)$ ，故总时间复杂度为 $O(n + m)$ 。

3.5 2011 I: Mummy Madness

题目大意

在一个无限大的网格上有 n 个木乃伊和一个人，每次人可以往周围的八个方向移动一步，人移动之后，每个木乃伊将会各自选择一个方向移动一步，使得它与人的欧式距离尽量小。当人与木乃伊在同一格时，人将被抓住。给出人的初始位置和 n 个木乃伊的位置，试求出人最长可以坚持多长时间才被木乃伊抓住。

算法分析

在时间 t 内，人或木乃伊可达的区域是一个边长 $2t + 1$ 的正方形。若木乃伊的所有可达区域的并覆盖了人所可达的区域，那么此时人一定会被木乃伊抓住，同时在 t 及以后的所有时刻，木乃伊可达区域的并仍然能覆盖人的可达区域。若木乃伊可达区域的并未覆盖人的可达区域，可以证明存在一条路径使得人在 t 及以前的时刻不会被抓住。

根据以上分析，我们可以二分人被抓住的时刻，求出在这个时刻木乃伊可达区域的并，检测其是否完全覆盖人的可达区域。

因为可达区域为矩形，我们可以用线段树配合扫描线来求区域的并，在求并的同时进行检测。注意到所有矩形都是等大的正方形，我们可以将线段树用树状数组替代以优化常数，如果进一步挖掘问题的性质，还可以用单调队列替代树状数组，从而将复杂度降阶，本人采取的是树状数组的做法。

设答案上界为 C ，单次求并的时间复杂度为 $O(n \log n)$ ，总时间复杂度为 $O(n \log n \log C)$ ，如果采用单调队列优化，单次求并时间复杂度为 $O(n)$ ，总时间复杂度为 $O(n \log C)$ ，两种做法均能通过测试。

3.6 2011 J: Pyramids

题目大意

有 m ($m \leq 320$) 种体积不同的背包，试选取最少的背包，使它们的体积和为 n ($n \leq 10^6$)。

算法分析

首先将背包按照体积从小到大排序。设在前 i 个背包中最少需要选择 $f[i, j]$ 个背包，使得它们的体积和为 j 。那么显然有 $f[i, j] = \min \{f[i - 1, j - w[i]] + 1, f[i - 1, j]\}$ 。但直接动

态规划无论是时间还是空间复杂度都无法承受。需要进行一些优化。

注意到实际的状态是达不到 mn 的，因此我们可以用 `hash` 表配合记忆化搜索，以减少空间的使用。将递推改为搜索后，显然优先选取较大的背包，能快速缩小答案的范围，故每次状态转移面对两个决策的时候，我们优先采取选择当前背包的决策。最后增加一个最优性剪枝：当搜索到状态 (i, j) 的时候，如果当前搜索的深度 dep 加上估价函数 $w(i, j) = \lceil \frac{j}{w[i]} \rceil$ 的值超过了已搜索出来的最优解，那么之后的搜索必然是无意义的。

加入以上优化后，程序的时空消耗大幅度减少，可以轻松通过所有测试数据。

3.7 2011 K: Trash Removal

题目大意

有一块钢板，其形状为 n 个顶点的简单多边形，现要将这块钢板经过旋转丢进管道里，求管道的最小直径。

算法分析

求出多边形的凸包，枚举哪一条边贴着管道壁进入管道，求出距离该边最远的点到该边的距离，并取所有情形的最小值即可。求凸包的复杂度为 $O(n \log n)$ ，利用旋转卡壳，可以将每次找最远点的复杂度从 $O(n)$ 降为均摊 $O(1)$ 。故总时间复杂度为 $O(n \log n)$ 。

4 Volume IV

4.1 2010 B: Barcodes

题目大意

给定一段条形码，试将其解码为对应的原字符串。原字符串仅包含 $0 \sim 9$ 、 $-$ ，每个字符的编码为条码中五个黑白相间的相邻区域，区域有宽区域和窄区域之分（宽区域的宽度是窄区域的两倍），相同字符的编码相同，不同字符的编码不同。将原字符串编码为条形码时，先原串的末端添加两个检验字符 C 和 K ， C 由原串唯一确定， K 由原串和 C 唯一确定，然后在串的开始和结束分别添加一个特殊字符，最后将串进行编码，相邻两个字符编码之间用一个窄区域隔开，得到条形码。题目输入依次给出条码各个区域的宽度（与真实宽度有 5% 的误差），若该条码不能成功解码，或检验字符的值有误，输出指定的错误信息，否则输出解码得到的原串。

算法分析

首先需要确定每个区域是宽区域还是窄区域。容易证明，一个区域只可能是宽区域或窄区域的一种。设第 i 个区域的测量宽度为 wid_i ，令 $\delta = \frac{2(1-5\%)}{1+5\%}$ ，若 $\exists i \leq n(wid_i \geq wid_1\delta)$ ，则第一个区域为窄区域，否则为宽区域，随后我们可以确定第 2 到第 n 个区域，对于第 i 个区域，若 $wid_i \geq wid_1\delta$ ，则该区域为宽区域，若 $wid_1 \geq wid_i\delta$ ，则该区域为窄区域，若上述两条均不成立，则该区域与第 1 个区域同宽。确定每个区域的宽窄后，可以通过解 n 组线性不等式解出窄区域宽度的解集，若其为空集，则该条码无法识别，否则我们可以进行解码。只要判断开始和结束的 5 个区域是否对应特殊字符，中间的编码是否能对应非特殊字符，字符编码之间是否用窄区域隔离开，以及两个检验字符的值是否正确即可。

时间复杂度 $O(n)$ 。

4.2 2010 C: Tracking Bio-bots

题目大意

在 $n \times m (n, m \leq 10^6)$ 的地图上有 $w (w \leq 1000)$ 个障碍，每个障碍为一个 $a \times 1$ 的矩形。有一个机器人要从起始位置移动到地图右上角，移动的过程中不可以超出地图范围，且只能向右走或向上走。由于障碍的存在，从一些位置出发，无论如何都无法到达地图右上角，将这些位置称为死格子。求死格子的数目。

算法分析

注意到 w 不大，可以将地图离散化成 $O(w^2)$ 个矩形。从地图右上角 BFS，如果一个被访问的矩形左边（下边）的矩形未被障碍占据，则将其插入队列。由此，除去被访问的矩形和被占据的矩形，剩余矩形的面积和即为死格子的总数。时间复杂度 $O(w^2)$ 。

事实上，通过离散化加上扫描线法，可以得到一个 $O(w \log w)$ 的优秀算法，考虑到篇幅有限，在此略去。

4.3 2010 D: Castles

题目大意

现要用军队夺取并控制一个地区，该地区由 n 个城堡组成，任意两个城堡之间有且仅有一条路径。夺取第 i 个城堡，至少需要 a_i 个士兵，进攻的过程会有 m_i 个士兵死亡，占领第 i 个城堡后，需要留下 g_i 个士兵看守城堡。军队可以任意选取一个城堡作为第一个被进攻的城堡，若占领一个城堡后，仍有城堡未被占领，则军队留下规定的人数看守目前所在的城堡，其余的人继续前进夺取未夺取的城堡。军队可以安全经过已被占领的城堡，但每条路在一个方向上至多只能被经过一次。求在上述条件下，控制所有城堡需要的最少兵力。

算法分析

显然该地区可以抽象为一棵 n 个节点的树（一个节点代表一个城堡），由于树上每条边在一个方向上至多只能经过一次，故军队经过城堡的顺序必然为该树的某个 DFS 序列的前面一部分（攻占完最后一个城堡后，军队不会回溯到第一个被攻占的城堡，故 DFS 序列残缺了后面的一小部分）。为了方便，我们可以让军队在占领最后一个城堡后，剩余的兵力（可以为 0 人）回到第一个被占领的城堡，如此可以将 DFS 序列补全，显然这样做对答案不会造成影响。

若第一个被攻击的城堡 r 确定，则原问题转化为在有根树上的情形。设 $f_r(i)$ 表示军队攻占以 i 为根的子树上的所有城堡后最少的剩余人数， $g_r(i)$ 表示占领和看守以 i 为根的子树上的所有城堡需要牺牲的兵力总和， i 的儿子为 v_1, v_2, \dots, v_k 。显然，攻占以 i 为根的子树上的所有城堡需要的最少兵力为 $f_r(i) + g_r(i)$ 。我们可以递归计算出 $f_r(v_1), f_r(v_2), \dots, f_r(v_k)$ ，再试图将这些结果合并，以计算出 $f_r(i)$ 。通过调整法可以证明，最优的策略必然是按照 $v_{j_1}, v_{j_2}, \dots, v_{j_k} (f_r(v_{j_1}) \geq f_r(v_{j_2}) \geq \dots \geq f_r(v_{j_k}))$ 的顺序发动进攻（即先占领以 v_{j_1} 为根的子树上的所有城堡，再占领以 v_{j_2} 为根的子树上的所有城堡，以此类推）。那么我们将 v_1, v_2, \dots, v_k 按照 $f_r(v_i)$ 的值从大到小排序得到 $v_{j_1}, v_{j_2}, \dots, v_{j_k}$ ，并依次发动进攻：当攻占完以 v_{j_i} 为根的子树上的所有城堡后，将剩余的兵力用于进攻以 $v_{j_{i+1}}$ 为根的子树上的城堡，若兵力不足，就将兵力增加至 $f_r(v_{j_{i+1}}) + g_r(v_{j_{i+1}})$ 。最终攻占完子树 v_{j_k} 后剩余的兵力即为 $f_r(i)$ 。

根据上述分析，在主算法中我们只要枚举 r ，通过递归贪心计算出 $f_r(r) + g_r(r)$ ，并取最小值即可。

在每次计算 $f_r(r)$ 时，算法时间复杂度的瓶颈在于排序，每个节点至多参与一次排序，故一次计算的时间复杂度为 $O(n \log n)$ 。因为总共要计算 n 次，故总时间复杂度为 $O(n^2 \log n)$ 。

4.4 2010 E: Channel

题目大意

在一个 $n \times m (n \leq 20, m \leq 9)$ 的网格上找一条最长的路径，该路径以左上角为起点，右下角为终点，且路径不能自交或经过网格上的障碍。下面两幅图为路径自交的例子。

算法分析

注意到 m 不大，可以采用经典的连通性状态压缩动态规划（插头 dp）解决本题，该方法的思想详见陈丹琦《基于连通性的状态压缩动态规划问题》，此处仅讨论状态的表示方法。

若不考虑路径自交的情形，对于轮廓线的每一个位置，我们只需要用 0 表示没有插头，1 表示左括号插头，2 表示右括号插头，3 表示独立插头。考虑路径自交这一约束条件时，则需要记录轮廓线上方的 m 个格子以及轮廓线转角的左上角的格子是否在路径中。我们可以增加一个 $m+1$ 位的二进制数码以表示这 $m+1$ 个格子是否在路径中，但注意到，如果轮廓线的某个位置上有插头，那么该位置的上（左）方的格子必定在路径中，我们只要在原来状态表示的基础上增加一个号码 4，表示轮廓线的某个位置没有插头，但该位置的上（左）方的格子在路径中。轮廓线转角的左上角的格子不在轮廓线任何位置的上（左）方，故需将状态增加一维来表示该格子是否在路径中。

为了简化问题，我们可以将网格图进行如下改造，从而避免独立插头的出现。如此我们将编码 4 改成 3，状态就可以用一个 $m+1$ 位的 4 进制数表示了。

修改前	修改后
	00000
0x0	0xxx0
000	0x0x0
0x0	000x0
	0x0x0
	xx000

最终我们用 $f[i, j, S, b]$ 表示轮廓线的状态编码为 S (S 为 $m+1$ 位的 4 进制数)，转角为 (i, j) ，转角左上角的格子的状态为 b ($b=0$ 表示该格子不在路径中， $b=1$ 则相反) 时，轮廓线以下最多能有多少格子被路径覆盖。状态的数目为 $O(nm4^m)$ ，转移复杂度为 $O(m)$ ，总时间复杂度为 $O(nm^24^m)$ 。考虑到状态极其稀疏，实现时可以用记忆化搜索。

4.5 2010 G: The Islands

题目大意

一片海区内有 n 个岛屿，一次巡航从最西边的岛屿开始不断往东走，直到到达最东边的岛屿，途中会到达一些岛屿。然后从最东边的岛屿不断往西，直至回到起点，途中会到达剩下未到达的所有岛屿。有两个岛屿（不是最西边或最东边的）上有加油站，由西往东的途中需要经过一次加油站，而由东往西的途中也要经过一次。试求一条路径使得在满足以上所有要求的前提下，路径的长度最短。

算法分析

该问题为经典的双调欧几里得旅行商问题的变体。可采用双进程动态规划解决。

为了方便，我们将路径分成两条，第一条为原路径由西向东的部分，第二条为原路径由东向西的部分，两条路径均视为以最西边的岛屿 s 为起点，以最东边的岛屿 e 为终点。设 $f[a, b, p_a, p_b]$ 表示第一条路径以 a 为起点，第二条路径以 b 为起点， p_a 表示从 a 到终点的路径中是否要经过加油站（ $p_a = 0$ 表示必须经过加油站， $p_a = 1$ 表示不能经过加油站， $p_a > 1$ 时该状态非法）， p_b 类似。设 c 为在 a 和 b 东边的岛屿中最靠西的一个岛屿。很容易得出状态转移方程：

$$f[a, b, p_a, p_b] = \begin{cases} \infty & p_a > 1 \vee p_b > 1 \\ \min \{ f[c, b, p_a + 1, p_b] + \text{dis}(a, c), f[a, c, p_a + 1, p_b] + \text{dis}(b, c) \} & c \text{ 是油站} \\ \min \{ f[c, b, p_a, p_b] + \text{dis}(a, c), f[a, c, p_a, p_b] + \text{dis}(b, c) \} & c \text{ 不是油站} \end{cases}$$

其中 $\text{dis}(u, v)$ 表示 u 到 v 的欧几里得距离。

边界条件

$$f[a, e, p_a, p_b] = \begin{cases} \infty & p_a = 0 \vee p_b = 0 \\ \text{dis}(a, e) & p_a = p_b = 1 \end{cases}$$

$$f[e, b, p_a, p_b] = \begin{cases} \infty & p_a = 0 \vee p_b = 0 \\ \text{dis}(b, e) & p_a = p_b = 1 \end{cases}$$

那么 $f[s, s, 0, 0]$ 即为所求。状态总数为 $O(n^2)$ ，单步转移的复杂度为 $O(1)$ ，故总时间复杂度为 $O(n^2)$ 。

4.6 2010 I: Robots on Ice

题目大意

有一个 $n \times m$ ($n, m \leq 8$) 的棋盘和一个机器人，机器人每次可以从一个格子移动到相邻的格子。现在机器人要从左下角 $(0, 0)$ 出发，经过每个格子恰好一次，最后到达 $(0, 1)$ ，且规定机器人第 $\lfloor \frac{nm}{4} \rfloor$ ($i = 1, 2, 3$) 步必须在 (r_i, c_i) 。求出满足条件的路径数。

算法分析

n 和 m 的规模较小，同时路径的限制条件非常多，可以考虑采用搜索。搜索过程中可加入如下剪枝：

(1) 机器人在第 $\lfloor \frac{nm}{4} \rfloor$ ($i = 1, 2, 3, 4$) 步未到达格子 (r_i, c_i) ，这里当 $i = 4$ 时， $(r_i, c_i) = (0, 1)$ 。

(2) 机器人在其他不正确的时刻到达 (r_i, c_i) 。

(3) 搜索的过程中，网格被路径分成了多个连通块，这里可以利用一些拓扑性质，将判断的复杂度优化至 $O(1)$ 。

加入以上剪枝后即可通过全部测试数据。

4.7 2010 J: Sharing Chocolate

题目大意

有一块 $a \times b$ 的矩形巧克力，每次可以将一块矩形巧克力沿横向或纵向切割成两块矩形巧克力，现有 n 个人需要吃巧克力，第 i 个人希望能吃到大小为 $size_i$ 的矩形巧克力。问是否可以经过一系列操作将巧克力恰好分成 n 块，以满足所有人的需求。

算法分析

设布尔函数 $f(x, y, S)$ 表示将 $x \times y$ 的巧克力恰好分给集合 S 里的所有人是否可行， $sum(S)$ 表示集合 S 内所有人希望吃到的巧克力大小总和。显然若 $sum(S) \neq x \times y$ ， $f(x, y, S) = \text{false}$ 。为了方便，我们规定第一次切割只能沿纵向切割（横向切割的情形可以将 x 和 y 交换，判断 $f(y, x, S)$ 的值）。注意到，当 x 和 S 确定时， y 被唯一地确定，故可以将 y 舍去，简记为 $f(x, S)$ 。不难发现 $f(x, S)$ 为真的充要条件为 $|S| = 1$ 或者存在 S 的一个分割 $S = S_1 \cup S_2 (S_1 \cap S_2 = \phi, S_1, S_2 \neq \phi)$ ，使得

$$(f(x, S_1) \vee f(\frac{sum(S_1)}{x}, S_1)) \wedge (f(x, S_2) \vee f(\frac{sum(S_2)}{x}, S_2)) = \text{true} \quad (x \mid sum(S_1))$$

我们可以枚举所有集合 S 并计算 $sum(S)$ 作为预处理，通过记忆化搜索判断 $f(a, U) \vee f(b, U)$ 是否为真。计算 $f(x, S)$ 的真值时，枚举 S 的所有真子集 S_1 并递归计算 $f(x, S_1)$ ， $f(\frac{sum(S_1)}{x}, S_1)$ ， $f(x, S \setminus S_1)$ ， $f(\frac{sum(S \setminus S_1)}{x}, S \setminus S_1)$ ，若发现 $S_1, S \setminus S_1$ 是可行的分割，则令 $f(x, S) = \text{true}$ 并中途退出。不妨设 a 与 b 同阶，状态数为 $O(a2^n)$ ，计算单个状态 $f(x, S)$ 的转移复杂度为 $O(2^{|S|})$ 。时间复杂度为 $O(a3^n)$ （根据二项式定理可得）。本题中 $a \leq 100, n \leq 15$ ， $a3^n$ 的最大值约为 10^9 。但由于状态较为稀疏，上述方法仍可以在规定时限内通过全部测试数据。

5 Volume V

5.1 2009 A: A Careful Approach

题目大意

有 $n(n \leq 8)$ 架飞机, 第 i 架飞机可以在时间区间 $[a_i, b_i]$ 内的任意时刻降落。问如何安排各架飞机的降落时间使得连续两次降落的间隔时间的最小值最大。

算法分析

二分答案并枚举飞机降落的顺序, 使最优性问题转化为可行性问题。设当前设置的最小间隔为 x , 飞机降落的顺序为 (r_1, r_2, \dots, r_n) , 显然第一架飞机降落时间为 $t_1 = a_{r_1}$, 第 $i+1(1 \leq i < n)$ 架飞机降落的时间为 $t_{i+1} = \max\{a_{r_{i+1}}, t_i + x\}$, 如果 $\exists 1 \leq i \leq n, t_i > b_{r_i}$, 那么这个方案是不可行的。

设答案上界为 C , 则时间复杂度为 $O(n!n \log C)$ 。

5.2 2009 B: My Bad

题目大意

给定一个逻辑电路, 若干组测试输入及其产生的输出, 试判断该逻辑电路是否发生故障, 并判断故障可能的类型是否唯一。

算法分析

将逻辑电路中的逻辑门拓扑排序后, 枚举所有可能的故障并模拟电路的运行即可。

设测试输入的数量为 t , 逻辑门和输入输出端的总数为 n , 则时间复杂度为 $O(n^2t)$ 。

5.3 2009 D: Conduit Packing

题目大意

有 4 个直径分别为 a, b, c, d 的圆, 现要用一个大圆将 4 个圆完全包含, 要求 4 个圆不能有重叠部分。求大圆的最小直径。

算法分析

二分直径的最小值, 问题转化为, 判断一个直径为 D 的圆是否能包含给定的 4 个圆。

首先放一个圆 Γ_1 , 令其与大圆内切, 再放两个圆 Γ_2, Γ_3 , 令它们与大圆内切, 且与 Γ_1 外切。最后再将剩下的一个圆 Γ_4 放进大圆内, Γ_4 的圆心应尽可能远离 Γ_1 的圆心, 如果圆与圆有重叠, 则此种放置方法是不可行的。枚举放置的顺序, 如果有一种方案可行, 那么答案必然不超过当前大圆的直径, 反之亦然。

设答案上界为 C , 则时间复杂度为 $O(\log C)$, 本题中 C 可取 10^5 。

5.4 2009 E: Fare and Balanced

题目大意

给定一个带权的有向无环图 $G = (V, E)$ ，现要将一些边的权值进行修改，使得从节点 1 出发的任意一条到达节点 n 的路径的权（路径的权为路径上所有边的权和）均等于某个值 k ，且经过被修改的边的次数不能超过 1，要求 k 的值最小。

算法分析

显然 k 的最小值为从 1 到 n 的最长路。

通过递推找出两个点集 S 和 T ：对于 S 中任意一点 u ，从 1 到 u 所有路径的长度相同；对于 T 中任意一点 v ，从 v 到 n 所有路径的长度相同。

若 $S \cup T \neq V$ ，那么该问题无解。否则找出所有的边 (u, v) ， $u \in S, v \notin S$ ，将该边的边权增加 $k - d(1, u) - d(u, n) - w(u, v)$ ，其中 $d(u, v)$ 为 u 到 v 的最短距离。

设 $|V| = n, |E| = m$ ，以上所有步骤的时间复杂度均为 $O(n + m)$ ，故总时间复杂度为 $O(n + m)$ 。

5.5 2009 F: Deer-Proof Fence

题目大意

在笛卡尔坐标系上有 $n(n < 10)$ 个点，现要画一些封闭的曲线，使得每个点都被其中一条封闭曲线包围，且距离封闭曲线上任意一点的距离至少为 R 。求这些曲线长度和的最小值。

算法分析

如果只画一条封闭曲线，容易发现，最短的封闭曲线的长度必然为这些点的凸包周长加上半径为 R 的圆的周长。

设 $f[S]$ 表示用一段封闭曲线包围点集 S 的所有点，封闭曲线的最小长度。 $g[S]$ 表示用任意数目的封闭曲线包围点集 S 的所有点，所有封闭曲线长度和的最小值。那么显然有

$$g[S] = \min \{f[S], \min \{f[S'] + g[S \setminus S'] \mid S' \neq \phi, S' \subset S\}\}$$

边界条件 $g[\phi] = 0$ ，设 n 个点组成的点集为 S_0 ，那么 $g[S_0]$ 即为所求。

在计算 $g[S]$ 时需要枚举所有子集，若一个集合有 k 个元素，那么该集合有 2^k 个子集。故枚举总量为

$$\sum_{i=1}^n \binom{n}{i} 2^i = (1 + 2)^n = 3^n$$

预处理 $f[S]$ 的时间复杂度为 $O(n2^n)$ ，计算 $g[S]$ 的时间复杂度为 $O(3^n)$ ，故总时间复杂度为 $O(n2^n + 3^n) = O(3^n)$ 。

5.6 2009 G: House of Cards

题目大意

给定一个规则较为复杂的不平等博弈游戏，假设双方都采取最优策略。试求先手与后手的最大分数差。

算法分析

博弈搜索。每步搜索玩家所有可能的策略，实现时可以使用类似 Dancing-Links 的链表加速搜索。加入博弈搜索常用的 $\alpha - \beta$ 剪枝技术，即可通过所有数据。

5.7 2009 H: The Ministers' Major Mess

题目大意

有 n 个议案和 m 个议员，每个议员只能对最多 4 个不相同的议案投票（支持或反对）。试指出哪些议案必须被通过或否决，使得每个议员有超过一半的建议得到满足。

算法分析

注意到，若一个议员提出的建议数不超过 2，那么他提出的建议必须被满足，否则他提出的建议至多有一个不被满足。

将每个议案抽象成两个点，一个点代表该议案被满足，另一个点代表该议案被否决。将建议数不超过 2 的议员的所有建议代表的点染色（表示该事件为真）。对于一个建议数超过 2 的议员 i ，设其提的建议数为 p_i ，每个建议对应的点为 $T_{i_j} (1 \leq j \leq p_i)$ ，我们在图中将所有的 $\neg T_{i_j}$ 与 $T_{i_k} (k \neq j)$ 连边。至此问题转化为经典的 2-SAT 模型。设 x_i 表示议案是否被通过，若 $x_i \Rightarrow \neg x_i$ ，则该议案必须被否决，若 $\neg x_i \Rightarrow x_i$ ，则该议案必须被通过。判断 $x_i \Rightarrow \neg x_i$ 以及 $x_i \Rightarrow x_i$ 是否成立，需要对图进行一次遍历。

图的点数为 $O(n)$ ，边数为 $O(m)$ ，一次遍历的复杂度为 $O(n + m)$ ，至多进行 n 次遍历，故总时间复杂度为 $O(n^2 + nm)$ 。

5.8 2009 J: Subway Timing

题目大意

有一棵 $n (n \leq 100)$ 个节点的树，树上的每条边有一个正整数权值。现要将所有边的权值进行修改，对于一条权值为 w 的边，可以将其权值修改为 $\gamma_1(w) = w \bmod 60$ 或 $\gamma_2(w) = w \bmod 60 - 60$ 。问如何修改，使得权的绝对值最大的一条路径，其绝对值最小。

算法分析

首先为了方便，不妨令 1 号点为树的根。

二分答案 W ，设 $f[u, T_1, T_2](T_1 + T_2 \leq W)$ 表示是否存在一种变换方案，使得从 u 到其任一后裔的路径的权在 $[-T_1, T_2]$ 之间，如果存在一个长为 $|child(u)|$ ($child(u)$ 为 u 的所有儿子的集合) 的三元组序列 $\{(t_{i_1}, t_{i_2}, p_i)\}$ ，使得对于任意的 $v_i, v_j \in child(u)$ ，有

$$\begin{cases} f[v_i, t_{i_1}, t_{i_2}] \wedge f[v_j, t_{j_1}, t_{j_2}] = \text{True} \\ t_{i_1} + t_{j_1} - \gamma_{p_i}(w(u, v_i)) - \gamma_{p_j}(w(u, v_j)) \leq T_1 \\ t_{i_2} + t_{j_2} + \gamma_{p_i}(w(u, v_i)) + \gamma_{p_j}(w(u, v_j)) \leq T_2 \end{cases}$$

那么 $f[u, T_1, T_2]$ 为真，否则为假。 f 可以通过经典的树形动态规划计算，若

$$\exists T_1 + T_2 \leq W (f[1, T_1, T_2] = \text{True})$$

那么 W 可以作为答案的上界。

设 W 的上界为 w ，一次计算时间复杂度为 $O(nw^4)$ ，总时间复杂度为 $O(nw^4 \log w)$ ，效率较低。

事实上，若 $f[u, T_1, T_2] = \text{True}$ ， $\forall T_2 < T_3 < W - T_1$ ， $f[u, T_1, T_3] = \text{True}$ 。设 $g[u, T]$ 表示满足 $f[u, T, T_2] = \text{True}$ 最小的 T_2 ，类似地采用树形动态规划来计算 g ，若

$$\exists 1 \leq T \leq W (g[1, T] + T \leq W)$$

则 W 可作为答案的上界。

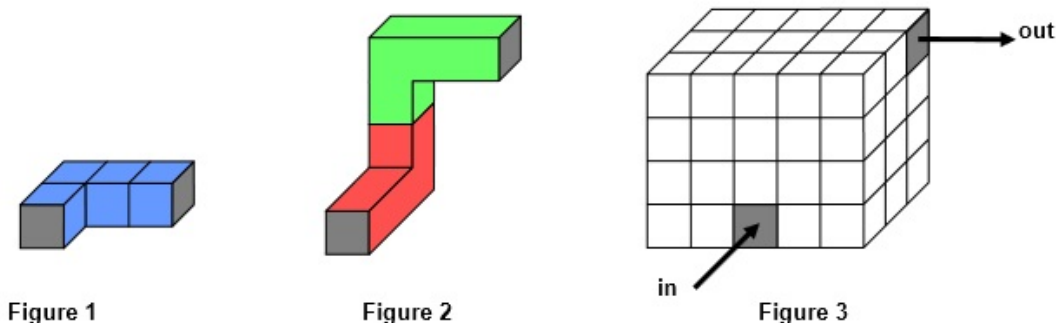
一次动态规划的时间复杂度为 $O(nw^2)$ ，总时间复杂度为 $O(nw^2 \log w)$ 。通过构造法可以证明 $w \leq 240$ ，故该算法可以在规定时限内出解。

6 Volume VI

6.1 2008 A: Air Conditioning Machinery

题目大意

现有 6 个同一型号的 L 形弯管（图 1），每个弯管恰好占用 4 个单位正方体的体积，且只有两个口（在图 1 中以灰色显示），可以将 2 个弯管接成一根长的管子（图 2 为一种对接方式）。给定一个长方体空间（长方体各边长为单位长的整数倍），该空间的外表面上有一个入口和一个出口，试求一种方案，在所有弯管不超出给定空间的要求下，用最少的弯管接通入口和出口。



算法分析

注意到可用弯管只有 6 条，我们可以考虑深度优先搜索。搜索过程中，记录哪些方格被占用，当前管道的终点和管道口的方向。每次在当前管道的终点尝试添加一个新管道并进行下一层的搜索。如果当前使用的管道数已超出之前搜索到的最优解则剪枝。

每次添加新管道的选择只有 4 种，故时间复杂度为 $O(4^{depth})$ ，本题中 $depth = 6$ 。

6.2 2008 B: Always an Integer

题目大意

一个多项式 $F(x)$ 可以表示为 $\frac{P(x)}{d}$ 的形式，其中 $P(x)$ 为整系数多项式， d 为正整数。问当 x 取任意正整数时， $F(x)$ 是否恒为整数。

算法分析

显然问题等价于求证 $P(x)$ 是否恒为 d 的倍数。

设 $P(x)$ 为 n 次多项式，下面证明， $d|P(x)$ 恒成立的充要条件是 $d|P(k), k = 0, 1, \dots, n$ 。

必要性显然。若 $d|P(k), k = 0, 1, \dots, n$ 成立，那么 $P(x)$ 的 n 阶差分多项式 $P^{(n)}(x)$ 必然满足 $d|P^{(n)}(0)$ 。注意到 $P^{(n)}(x)$ 为常数项，故 $d|P^{(n)}(x)$ 恒成立 $\Rightarrow d|P^{(n-1)}(x)$ 恒成立 $\Rightarrow \dots \Rightarrow d|P(x)$ 恒成立。充分性得证。

故我们只需依次求出 $P(0), P(1), \dots, P(n)$ ，并检测它们是否为 d 的倍数即可。

一次多项式求值的时间复杂度为 $O(n)$ ，故总时间复杂度为 $O(n^2)$ 。

6.3 2008 E: Huffman Codes

题目大意

文本中有 $n(n \leq 20)$ 个字母，每个字母在文本中出现频率的百分数都是正整数。已知每个字母的哈夫曼编码，问所有字母的出现频率有多少种可能。

算法分析

首先构建哈夫曼树。在哈夫曼树上，定义一片叶子的权为该叶子对应字母的出现频率；非叶节点的权为其两个儿子的权和。观察哈夫曼树的构造过程，可以得出一个非常有用的性质：

若在哈夫曼树上进行 BFS，且每次优先将右儿子插入队列，则得到的 BFS 序等价于其所有节点按权值从大到小排序的序列。

根据这个性质，我们可以设计一种回溯算法，该算法使用一个队列作为辅助数据结构，具体方法如下所述：

首先将树根 r 加入队列， r 的权值设为 100。

在每一层搜索中，如果队列为空，则将所求的方案数增加 1 并回溯。否则取出队首元素 v ，若 v 有儿子，则将 v 的权值完全分配给两个儿子，左儿子的权值不超过右儿子的权值，右儿子的权值不超过队尾节点的权值。每次分配完权值之后，将右儿子和左儿子依次加入队尾，并删除队首，进行下一层的搜索。最后将队列恢复到初始状态并回溯。

该算法的理论时间复杂度为指数级，但实际效率很高。

6.4 2009 F: Glenbow Museum

题目大意

一个直角多边形为内角均为 90° 或 270° 的多边形。从多边形任意一个内角出发，沿逆时针方向绕一圈，经过 90° 角记下 R，经过 270° 角记下 O，我们就可以得到一个“角序列”。一个角序列可以粗略地描述一个多边形。求出长度为 n 的角序列的数目，这些角序列描述的多边形满足：在多边形内存在一点，其到多边形内任意一点的连线不与多边形的边相交。

算法分析

容易发现，当角序列存在连续两个 O（或者最后一个字母与第一个字母同时为 O）时，无论站在多边形内哪个点都不可能看到完整的多边形，若角序列中不存在连续的 O，其描述的多边形是否能满足条件呢？答案是肯定的。由于本人水平有限，具体的证明不在此阐述。

设 $f[i, j, k]$ 表示长度为 i 且首字母为 R 的角序列中，R 的个数比 O 的个数多 j ，最后一个

字母为 k 时可能的方案数。则有

$$\begin{cases} f[i, j, \mathbf{R}] = f[i-1, j-1, \mathbf{R}] + f[i-1, j-1, \mathbf{O}] \\ f[i, j, \mathbf{O}] = f[i-1, j+1, \mathbf{R}] \end{cases}$$

边界条件为 $f[1, 1, \mathbf{R}] = 1$ ，所求为 $f[n, 4, \mathbf{R}] + f[n, 4, \mathbf{O}] \times 2$ 。

注意到 f 的第二维只能在 0 到 5 之间，第二维超出这个范围的状态都是没有意义的。故总状态数为 $O(n)$ ，转移的复杂度为 $O(1)$ ，总时间复杂度为 $O(n)$ 。

6.5 2008 G: Net Loss

题目大意

给定一个 n 次多项式 $p(x)$ ，试找出一个定义在 $[-1, 1]$ 上的函数 $g(x)$ ，使得

$$d(p, g) = \int_{-1}^1 (p(x) - g(x))^2 dx$$

最小。 $g(x)$ 为一个分段函数，其图像由两条线段组成，所在直线解析式分别为 $y = a_1x + a_0$ 和 $y = b_1x + b_0$ ，两线段有一个共同的端点，其横坐标 $c \in [-1, 1]$ ，在输入中给定。

算法分析

$d(p, g)$ 为关于 a_0, a_1, b_0, b_1 的二次式，设两线段交点的纵坐标为 y_0 ，则 a_0 可以用 a_1 和 y_0 替代， b_0 也可以用 b_1 和 y_0 替代， $d(p, g)$ 化为关于 a_1, b_1, y_0 的二次式。当 $d(p, g)$ 取最小值时，其关于三个变量的偏导数均为 0，由此我们可以列出线性方程组，求根并代入 $d(p, g)$ 的表达式即可。

算法中需要求几个函数的定积分，时间复杂度为 $O(n)$ 。其余计算均为常数时间。故总时间复杂度为 $O(n)$ 。

6.6 2008 I: Password Suspects

题目大意

求出所有的长度为 n ($n \leq 25$) 的包含指定子串的字母串 T 的数目。子串数 $m \leq 10$ ，题目保证答案在 10^{15} 之内。

算法分析

建立 m 个串的 AC 自动机，设 $f[i, j, S]$ 表示 T 已被自动机识别了 i 个字符，到达状态为 j ，已匹配的子串集合为 S 的情况下，子串 $T[i+1..n]$ 有多少种可能。边界条件为 $f[n, i, U] = 1$ ， i 为自动机的任意状态， U 为 m 个子串组成的全集。目标为 $f[0, 0, \phi]$ 。找到状态表示后，剩下的问题就是一个简单的动态规划。

设每个子串最大长度为 l ，字母表大小为 Σ ，则总时间复杂度为 $O(nml\Sigma 2^m)$ 。

6.7 2008 J: The Sky is the Limit

题目大意

有 n 座山，每座山可以看作一个等腰三角形，如下图所示。试求出群峰形成的山脉的轮廓线的长度（下图中的黑色加粗部分）。



算法分析

每座山由两条线段组成，求出所有的线段与线段之交点的横坐标，将这些坐标及山脚和山顶的横坐标一并排序。这些坐标将数轴分成了若干个区间，每段区间最高的线段是唯一的。分别找出这些线段，并将长度累加。

线段交点数为 $O(n^2)$ ，排序的复杂度为 $O(n^2 \log n)$ ，在每段区间内找最高的线段，时间复杂度为 $O(n)$ ，区间数与交点数同阶，为 $O(n^2)$ 。故总时间复杂度为 $O(n^2 \log n + n^3) = O(n^3)$ 。

6.8 2008 K: Steam Roller

题目大意

有一个网格地图，网格上每个格点与其上下左右四个方向的相邻格点（如果存在的话）相连，形成若干条道路，每条道路都有一个理想通过时间。有一个人要驱车从一个格点到另一个格点，如果经过某条道路时不需要变速，那么通过该道路的时间为其理想通过时间，否则通过该道路的时间为其理想通过时间的两倍。车辆拐弯时，要在拐弯前通过的道路减速，在拐弯后进入的道路加速。车辆在起点出发要加速，到达终点时要减速。给出每条道路的理想通过时间（有些道路可能不允许经过），试求此人从起点到达终点的最少耗时。

算法分析

如果没有“两倍时间”这一条件，那么本题就是一个非常简单的最短路问题。事实上，本题仍可以用最短路算法解决：将一个格点拆成上点，下点，左点和右点，每个点再进行拆分，拆成 a 点和 b 点， a 点表示不需要变速， b 点表示需要变速。如此改造之后，稍作分类讨论就能构造出最短路模型，用 `dijkstra` 算法求最短路即可。具体的建图方法不再阐述。

本题中边数与点数是同阶的，故我们采取二叉堆优化的 `dijkstra` 算法，设网格大小为 $m \times n$ ，则时间复杂度为 $O(mn \log(mn))$ 。

7 Volume VII

7.1 2007 A: Consanguine Calculations

题目描述

给出父亲、母亲、孩子其中两人的血型，求出另一人可能的血型。

算法分析

枚举所有的血型，判断是否可能成立。时间复杂度 $O(1)$ 。

7.2 2007 H: Marble Game

题目大意

在一个 $n \times n$ 的棋盘上有 m 个球和 m 个洞，棋盘上有一些障碍。每次可以将整个棋盘沿 4 个方向的其中一个倾斜，球会向倾斜的方向滚动，直到碰到障碍或不能继续滚动的球，或掉进洞里。每个球必须落到对应编号的洞中。求最少的倾斜次数。

算法分析

用 m 个球的坐标表示一个状态，由初始状态 BFS 即可。时间复杂度为 $O(n^{2m})$ ，但不合法状态很多，程序仍然能在规定时限内出解。

7.3 2007 I: Water Tanks

题目大意

有 n 个圆柱体水箱排成一排，所有水箱的底面积均为 1m^2 ，第 i 个水箱高为 h_i 。第 $i(1 \leq i < n)$ 个水箱与第 $i+1$ 个水箱之间有一根横向的细管相连，高度为 l_i ，保证 $l_1 < l_2 < \dots < l_{n-1}$ 。1 号水箱顶部与大气相连通。现从 1 号水箱的顶部缓缓地注水，问最多能注多少水。

算法分析

显然我们应按照从 2 号水箱到 n 号水箱的顺序，逐一确定水在各水箱内的水位。

设水在每个水箱的水位为 m_i ，我们可以设计如下的算法框架来计算答案：

- (1) 令 $k = 2$ 。
- (2) 若利用理想气体状态方程能得出 $m_k < l_k$ 的结论，求出 m_k ，令 $i = k$ ，转 (5)。
- (3) 若利用理想气体状态方程能得出 $m_{k+1} < l_k$ 的结论，求出 m_{k+1} ，令 $i = k+1$ ，转 (5)。
- (4) 求出 m_k ，令 $k = k+1$ ，转 (2)。

(5) 输出 $\sum_{k=1}^i m_k$ 。

这里令 $l_n = +\infty$ 。

上面关于 m_k 的求解，均利用了理想气体状态方程。具体实现的细节较多，由于不是算法的核心思想，故在此略。

算法时间复杂度为 $O(n)$ 。

7.4 2006 A: Low Cost Air Travel

题目大意

有 n 种机票，第 i 种机票的票价为 c_i ，飞行路线为 $(a[i, 1], a[i, 2], \dots, a[i, n_i])$ ，只有在城市 $a[i, 1]$ 出示一张该种机票才能登机，乘客可以任选一个城市 $a[i, j] (1 < j \leq n_i)$ 下飞机。一张机票只能使用一次。问如何购买机票，使得从 b_1 出发，依次经过 b_2, b_3, \dots, b_{m-1} ，最后到达 b_m 的花费最小。

算法分析

设 $f[i, j, k]$ 表示当前在第 i 种机票规定的旅行路线上的第 j 个城市，已经过旅行计划的前 k 个城市时的最小花费。 $f[i, j, k]$ 可以更新 $f[i', 1, k] (a[i', 1] = a[i, j])$ 的值， $f[i, j, k] + w(a[i, j], a[i, j+1])$ 可以更新 $f[i, j+1, k']$ (若 $a[i, j+1] = b_{k+1}$ ，则 $k' = k+1$ ，否则 $k' = k$) 的值。

由于状态与状态之间没有拓扑关系，故我们不能直接进行动态规划。我们可以在 $f[i, j, k]$ 与 $f[i', 1, k]$ 之间连一条权为 0 的有向边，在 $f[i, j, k]$ 与 $f[i, j, k']$ 之间连一条权为 $w(a[i, j], a[i, j+1])$ 的有向边。这里所有变量的含义同上一段。

将所有的 $f[i, 1, k]$ (若 $a[i, 1] = b_1$ ，则 $k = 1$ ，否则 $k = 0$) 合并为一个点 s 。用 dijkstra 算法或 spfa 算法求以 s 为源点的单源最短路径。最后找到最小的 $f[i, j, m]$ 即为所求。

设总状态数为 K ，则时间复杂度为 $O(K \log K)$ 或 $O(K)$ ，取决于最短路的实现方式。

7.5 2006 B: Remember the A La Mode!

题目大意

有一家甜品店出售冰淇淋和饼片，一份冰淇淋和一份饼片搭配在一起出售，给出不同种类的饼片与冰淇淋搭配出售的利润（一些饼片与冰淇淋不能搭配），以及每种饼片和冰淇淋的份数，试给出甜品店能获得的利润范围。

算法分析

将饼片和冰淇淋抽象成图论中的点，按照如下方法建立一个网络：

1) 若饼片 u 与冰淇淋 v 可以搭配，搭配在一起出售的利润为 $w(u, v)$ ，那么从 u 连一条容量为 $+\infty$ ，费用为 $w(u, v)$ 的边。

- 2) 增加一个源点 s ，设饼片 i 的份数为 a_i ，从 s 往 i 连一条容量为 a_i ，费用为 0 的边。
- 3) 增加一个源点 t ，设冰淇淋 i 的份数为 b_i ，从 i 往 t 连一条容量为 b_i ，费用为 0 的边。

该网络的最小费用最大流即为最小收益，最大费用最大流即为最大收益。我们只需要运行两次费用流增广路算法即可。

设网络节点总数为 n ，网络最大流流量为 C ，则总时间复杂度为 $O(n^2C)$ ，该上界较为宽松，算法的实际效率非常高。

7.6 2006 D: Bipartite Numbers

题目大意

给定一个正整数 $k(k \leq 10^5)$ ，试找出比 k 大且为 k 的倍数的最小“二段数”。二段数各个数位由且仅由两种数字 $s(s \neq 0)$ 和 t 组成，且所有的 s 均在所有 t 的前面。

算法分析

设 $f[r, p](r = 0, 1, \dots, 9, 0 \leq p < k)$ 为数列 $a_0 = p, a_i = 10a_{i-1} + r(i > 0)$ 中为 k 的倍数的项的最小下标，那么显然有

$$f[r, p] = \begin{cases} 0 & p = 0 \\ f[r, (10p + r) \bmod k] + 1 & p \neq 0 \end{cases}$$

用记忆化搜索对 f 进行预处理。然后枚举 s 和 t 的值，并枚举二段数的前半部分 s 的个数 c ，设前半部分除以 k 的余数为 p ，需要添加 $g(s, t, p)$ 个 t 才能让二段数为 k 的倍数。如果 $p = 0$ ，则 $g(s, t, p) = [t, t \bmod k] + 1$ ，否则 $g(s, t, p) = f[t, p]$ 。二段数可以表示为四元组 $(s, c, t, g(s, t, p))$ ，用这个四元组来更新最优解。这样做有一个漏洞，就是若 k 本身为二段数，那么得到的答案必然为 k ，不满足“比 k 大”这一要求。具体实现时，我们要保留一个最优解和一个次优解，用 $(s, c, t, g(s, t, p))$ 更新最优解和次优解的同时用 $(s, c, t, g(s, t, p) + f[t, t \bmod k] + 1)$ 更新次优解。最后若最优解不为 k ，则输出最优解，否则输出次优解。

二段数每种数字的个数不超过 k ，令 Σ 为可选数字的个数 (10)，则时间复杂度为 $O(k\Sigma^2)$ ，不能在规定时间内出解。注意到枚举 c 的时候，若 c 已超过次优解的位数时，可以停止枚举。因为答案的位数不会特别大，这种优化的效果极其显著。事实上可以通过预处理将时间复杂度降为 $O(k\Sigma)$ ，但该做法常数过大，实际效率比之前优化过的算法劣，也不能在规定时间内出解。

7.7 2006 E: Bit Compressor

题目描述

有一种压缩 01 串的方法：将连续的 k 个 1 压缩成 k 的二进制表示，要求这 k 个 1 的左右必须是 0 或串的开头结尾。给定一个长度为 $m(m \leq 40)$ 的经过压缩后的 01 串 S ，问有多少个长度为 $L(L \leq 16000)$ ，1 的数目为 n 的 01 串经过压缩后为 S 。

算法分析

将不是由压缩得到的 \emptyset 字符视为分隔符。枚举压缩的串中 \emptyset 有哪些作为分隔符，将分隔符之间的子串解压，判断解压后是否满足条件。时间复杂度为 $O(2^m)$ 。注意到，分隔符之间的子串若非空，必须以 1 开头，且紧跟在分隔符之后的 \emptyset 也必然为分隔符。因此我们可以将 $\emptyset 1$ 视为分隔符。 $\emptyset 1$ 在 S 中出现的次数不超过 $\frac{m}{2}$ ，故时间复杂度可修正为 $O(2^{\frac{m}{2}})$ ，与 L 无关。

事实上本题还有 $O(m^2L)$ 的动态规划算法，但实现细节较多，且在 m 较小， L 较大的情况下，运行效率不及之前提到的指数级算法。

7.8 2006 G: Pilgrimage

题目大意

有一个组织，组织有一些公共资金及管理公共资金的账本。账本上会记录如下信息：

- pay k : 支出 k 元的公共资金
- collect k : 每人缴纳 k 元作为公用
- in k : 有 k 个人加入组织
- out k : 有 k 个人离开组织

当有 k 个人离开组织时，如果他们离开前有 n 个人，公共资金必然为 $a \cdot n$ (a 为正整数)，离开的 k 个人每人可以拿走 a 元。当有 k 个人加入组织时，如果他们加入前有 n 个人，公共资金也必然为 $a \cdot n$ (a 为正整数)，每个加入的人都需要支付 a 元作为公用。

给定账本的某一页，问在该页开始记录前，组织里可能有多少人。询问有多组，组数 $T \leq 30000$ 。

算法分析

首先可以发现，collect 操作对答案没有任何影响，因此可以忽略此类信息。

设组织在账本该页记录前有 x 人，那么总资金可以表示为 $px + q$ 的形式。

第一次 in/out 操作时，设之前支出总量为 q' ，则总资金为 $px + (q - q')$ 。我们只要令 $q = q'$ ，即可保证 in/out 操作的实施。故第一次 in/out 操作之前的 pay 操作对答案也没有影响。

设第 i 次 in/out 操作后有 $x + \delta_i$ 个人，总资金为 $p_i(x + \delta_i)$ 。第 i 次 in/out 操作到第 $i + 1$ 次 in/out 操作之间的 pay 操作总共支出了 q_i 元，那么 $(x + \delta_i) | q_i$ 必然成立，满足条件的 x 形成一个解集。假设 in/out 操作共有 n 次，那么我们将得到 $n - 1$ 个解集，求出这些解集的交即可。

设 $\max q_i = Q$ ，如果每次以枚举法找出所有 q_i 的因子，那么处理一个询问的时间复杂度为 $O(n\sqrt{Q})$ ，总时间复杂度为 $O(Tn\sqrt{Q})$ 。本题中 $Q \leq 10^5, n \leq 50, T \leq 3 \times 10^4$ ，这样的复杂度无法通过测试。我们可以预处理 1 到 10^5 范围内所有数的因子表，这样每次寻找 q_i 的因子

时可以减少大量无用的枚举。最坏情况下时间复杂度为 $O(Q \log Q + Tn\sqrt{Q})$ ，但常数比前一种枚举法小很多。若数据为随机生成，期望复杂度为 $O(Q \log Q + Tn \log Q)$ 。

7.9 2006 I: Degrees of Separation

题目大意

给定一个 p 个点的无向图，图中任意两点间的分离度为两点间的最短路长，求该图分离度的最大值。

算法分析

用 Floyd 算法求出任意两点的最短路并取最大值即可。时间复杂度 $O(p^3)$ 。

7.10 2006 J: Routing

题目大意

给一个 n 个点， m 条边的有向图。试求出一条从 1 号点到 2 号点，再返回 1 号点的回路，使其经过的节点最少（重复经过的点只算经过一次）。

算法分析

设 $f[i, j]$ 表示从 1 出发，顺次经过 i 和 j ，最终返回 1，经过的最少节点数。可以得到状态转移方程

$$f[i, j] = \min \{f[x, y] + w[y, i] + w[i, j] + w[j, x] - 1\}$$

其中 $w[i, j]$ 表示从 i 到 j 的最短路的长度。

边界条件 $f[1, 1] = 1$ ，目标为 $f[2, 2]$ 。

由于状态与状态之间不存在拓扑关系，我们不能直接动态规划。将每个状态视作一个点，对于两个状态 $(x_1, y_1), (x_2, y_2)$ ，从 (x_1, y_1) 出发，向 (x_2, y_2) 连一条权值为 $w[y_1, x_2] + w[x_2, y_2] + w[y_2, x_1] - 1$ 的边。 $f[2, 2]$ 的值即为状态 $(1, 1)$ 到 $(2, 2)$ 的最短路的长度加上 1。该图为完全图，故我们采取 Dijkstra 算法计算 $(1, 1)$ 到 $(2, 2)$ 的最短路。

由上述分析我们可以得到如下的算法：先用 Floyd 算法求出任意两个点之间的最短路，再用 Dijkstra 算法计算 $f[2, 2]$ 的值。

Floyd 算法的效率为 $O(n^3)$ ，Dijkstra 算法在节点数为 $O(n^2)$ 个点的完全图上的效率为 $O(n^4)$ 。故总时间复杂度为 $O(n^4)$ 。

8 Volume VIII

8.1 2005 C: The Traveling Judges Problem

题目大意

有 k 个人要租车到达同一个城市 t ，但他们的出发位置不同。为了节约成本，如果有任意两个人的行程路线经过同一个城市，那么这两个人会在这个城市租同一辆车向目的地进发。一辆车可以搭载任意数量的乘客，租车的费用与车上的人数无关，仅与行驶距离成正比。给出城市与城市之间的交通图（可以抽象为 n 个点 m 条边的无向图），试给出一个方案，使得这 k 个人租车的总费用最小。

算法分析

如果 k 个人途经的城市集合 V' 确定，我们找出原图的子图

$$G' = (V', E') \quad (V' \subset V, E' \subset E, \forall (u, v) \in E', u, v \in V')$$

最优方案必然为 G' 的最小生成树（如果生成树不存在，则在这种情况下不存在合法方案）。

本题 n 的范围较小，我们可以枚举所有的 V' ，求出对应子图 G' 的最小生成树，这些最小生成树中边权和最小的就是最优解。

枚举集合的复杂度为 $O(2^n)$ ，每次求最小生成树的复杂度为 $O(m)$ ，故总时间复杂度为 $O(m2^n)$ 。

8.2 2005 H: The Great Wall Game

题目大意

在一个 $n \times n$ 的棋盘上放了 n 颗棋子，每次可以将一颗棋子移动到上下左右相邻的空方格之中，求出最少的移动步数，使得 n 颗棋子排成一条直线。

算法分析

枚举所有可能的直线（共 $2n + 2$ 条），将直线上任意一个格子与任意一颗棋子连边，边权为这两个位置的曼哈顿距离。由此形成一个带权的二分图，用 KM 算法求出其最小权匹配，并求出 $2n + 2$ 次匹配的最优值即可。

运行一次 KM 算法的时间复杂度为 $O(n^3)$ ，故算法总时间复杂度为 $O(n^4)$ 。

8.3 2005 I: Workshops

题目大意

有 n 个会议和 m 个会议室，第 i 个会议有 a_i 个人参与，第 k 个会议室最多能容下 b_k 个人。所有会议在同一时间举行，每个会议室有一个清场的时间。会议 p 能安排在会议室 q

的充要条件是, p 结束的时间在 q 清场的时间之前, 且 $a_p \leq b_q$ 。试计算最多有多少个会议可以被安排在会议室中, 如果有多种方案, 求出在会议室中开会人数最多的一种。

算法分析

如果会议 p 能在会议室 q 举办, 那么将 p 与 q 连一条容量为 1, 费用为 a_p 的边。新增一个源点 s , 一个汇点 t 。从 s 出发, 向每一个会议代表的点连一条容量为 1, 费用为 0 的边, 从每个会议室代表的点向 t 连一条容量为 1, 费用为 0 的边。那么该网络的最大费用最大流即为所求。但这个网络过于庞大, 需要进行优化。

该网络存在一定的特殊性, 我们可以设计如下的贪心策略。

每次优先安排人数较多的会议, 在剩下的可以容下该会议的会议室中, 找一个清场时间最早但在会议结束之后的会议室, 将会议安排在该会议室中。如果找不到这样的会议室, 则该会议安排在外面。该算法的正确性可以由费用流的增广路算法在网络上的运行过程证明。

最终的算法是, 将会议按照人数从大到小排序, 将会议室按照容量从大到小排序。然后进行扫描, 每次将容量不小于当前会议人数的会议室放进待选集合中, 从待选集合中选一个“时间最紧”的会议。这个待选集合可以用平衡树 (C++ STL 中的 `set`) 维护。算法时间复杂度为 $O(n \log n + m \log m)$ 。

8.4 2005 J: Zones

题目大意

某公司计划建立 n ($n \leq 20$) 个信号塔, 每个信号塔可以为一定数目的客户服务。一些区域可能被多个信号塔发出的信号覆盖。由于一些原因, 最终只能建造其中的 m 座塔。给出每个区域的客户数以及其被哪些待建信号塔覆盖, 问选择哪几座塔建设, 被覆盖到的客户最多。

算法分析

n 的规模较小, 可以采取枚举法。枚举所有方案并计算每个方案覆盖到的客户数, 取最优解即可。设区域数为 k , 则时间复杂度为 $O(C_n^m k)$ 。

8.5 2004 H: Tree-Lined Streets

题目大意

有 n 条线段, 线段与线段之间可能有交点, 称这些点为交叉点。现要在线段上放点, 在同一条线段上的两点距离至少为 50 单位, 点与同一条线段上的交叉点距离至少为 25 单位。试求在这些线段上最多能放多少个点。

算法分析

首先注意到, 我们对于每条线段放点的决策是相互独立的, 即在一条线段上放的点不会影响到另一条线段放点的位置。因此我们可以找到每条线段上能放的点数的最大值, 最后再

进行累加。

对于一条线段，设其长度为 L ，其上有 m 个交叉点。若 $m = 0$ ，那么显然在该线段上可以放 $\lceil \frac{L}{50} \rceil$ 个点。否则将其分割为 $m + 1$ 条线段（分割点为 m 个交叉点），可以发现在这 $m + 1$ 条线段上放点的决策也是相互独立的。我们将这些线段顺次编号为 $1, 2, \dots, m + 1$ ，设它们的长度分别为 l_1, l_2, \dots, l_{m+1} 。这些线段中， $2, 3, \dots, m$ 号线段的两个端点都是交叉点，在其上最多能放 $\lceil \frac{l_i - 50}{50} \rceil$ 个点， $1, m + 1$ 号线段的其中一端是原线段的端点，在这 2 条线段上最多能放 $\lceil \frac{l_i - 25}{50} \rceil$ 个点。

根据以上分析，我们只需要将所有线段以交叉点为分割点进行分割，然后对分割出的线段分别统计答案并累加即可。

设有 n 条线段，分割一条线段的复杂度为 $O(n^2)$ ，故将所有线段分割的复杂度为 $O(n^3)$ 。经过分割后，最多有 $O(n^2)$ 条线段，对每条线段进行计算的复杂度为 $O(1)$ 。因此总时间复杂度为 $O(n^3 + n^2) = O(n^3)$ 。事实上，如果使用快速排序加速分割，时间复杂度可以优化到 $O(n^2 \log n)$ 。

8.6 2003 A: Building Bridges

题目大意

有一个城市，其地图呈 $r \times c$ 的网格，地图上相邻的建筑连通，不相邻的建筑不连通。可以建造平行或垂直于地图横轴的道路将不连通的建筑相连，道路不能拐弯。求使所有建筑连通需要的道路长度和的最小值。

算法分析

将每个建筑与其下方和右方的相邻建筑连边，求最小生成树。

时间复杂度 $O(rc \log(rc))$ 。

8.7 2003 B: Light Bulbs

题目大意

有 n 个灯泡和 n 个开关，第 i 个开关控制第 $i - 1, i, i + 1$ 个灯泡（如果存在的话）。问最少需要转换多少开关能使这 n 个灯泡由初始状态转化为目标状态，并求出方案。

算法分析

枚举第一个开关转换与否，后面的开关是否转换将被唯一地确定。最后判断第 n 个灯泡是否被转换到了目标状态。

算法时间复杂度为 $O(n)$ ，但由于本题的输入将数据进行压缩，解压所需的时间复杂度为 $O(n^2)$ ，故总时间复杂度为 $O(n^2)$ 。

8.8 2003 J: Toll

题目大意

有一个 n 个点的无向图，有一些点代表乡村，有一些点代表城镇。现要从 s 运 p 个银匙到 t ，途经乡村需要缴纳一个银匙作为过路费，途经城镇时，每运 20 个银匙需缴纳一个银匙。试求出将 p 个银匙从 s 运到 t 至少需要带多少银匙。

算法分析

经过城镇时，若运了 k 个银匙，那么需要缴纳的银匙数为 $g(k) = \lfloor \frac{k-1}{20} \rfloor + 1$ 。设 $f(u)$ 为将 p 个银匙从 u 运到 t 需要携带的银匙数。显然 $f(t) = p$ ，目标是最小化 $f(s)$ 。以 t 为源点运行一次 dijkstra 算法，通过边 (u, v) 进行松弛操作时，如果 v 是乡村，用 $f(u) + 1$ 更新 $f(v)$ ，否则解出 $x - g(x) = f(u)$ 的根 x ，用 x 来更新 $f(v)$ 。通过简单的数学推导，可以在 $O(1)$ 时间内求出 x 的值。

算法总时间复杂度为 $O(n^2)$ 。

9 Volume IX

9.1 2002 A: Balloons in a Box

题目大意

空间中有一个各棱平行于坐标轴的长方体和 n ($n \leq 6$) 个点，每次可以在 n 个点中任选一个点，以这个点为气球的球心（假设气球是球体）让气球不断膨胀，直至碰到长方体的边界或与之前已被放置好的气球接触，要求被选点在长方体内且不在之前已被放置好的气球内部，选点的顺序任意，且有些点可以不被选择。试求出选点的顺序使得气球在长方体中占据的体积最大。

算法分析

注意到 n 非常小，我们可以暴力枚举选点的顺序，并模拟放置气球的过程，在点 O 放置一个气球时，先通过长方体来确定该气球半径 r 的上界，枚举已被放置的所有气球 (O_i, r_i) ，显然新的气球半径 $r \leq |OO_i| - r_i$ ，用 $|OO_i| - r_i$ 更新 r 的上界即可。最后计算出所有气球的体积和并更新答案。时间复杂度 $O(n^2n!)$ 。

9.2 2002 E: Island Hopping

题目大意

在太平洋上有 n 个岛，每个岛上有一定数量的居民。其中一个岛（称为主岛）已与互联网相连，现要在岛与岛之间连接海底电缆，使得每座岛都通过一条电缆路径与主岛相连。要

求总的电缆长度最小。当方案确定后，所有的电缆将同时开工，每条电缆的工期与其长度成正比。试求每个岛与互联网连通的时间，并求出所有居民联网时间的平均值。

算法分析

显然本题的模型是最小生成树。一个岛与互联网接通的时间为其沿最小生成树的边到主岛的路径中的最大边权。容易证明，无论选取哪个最小生成树作为连接方案，每个岛与互联网连通的时间为定值。故我们只需用 Prim 算法求出一个最小生成树并计算答案即可。

时间复杂度 $O(n^2)$ 。

9.3 2002 G: Partitions

题目大意

将一个矩形分成若干较小的不重叠的子矩形，称为对矩形的划分。如果划分 A 是划分 B 通过继续划分它的一个或多个子矩形得来，那么称 A 比 B 精细， B 比 A 粗糙。给出对相同矩形的两个划分 A 和 B ，求出比 A 和 B 都粗糙的最精细的划分 C ，以及比 A 和 B 都精细的最粗糙的划分 D 。

算法分析

将 A 和 B 用 01 矩阵（位图）表示，设 $A \cap B$ 表示将 A, B 的每一位取与（and）得出的位图。求 C 的方法是，在 $A \cap B$ 中找出一条将矩形一分为二的分割线，如果找不到分割线，那么比 A 和 B 都粗糙的划分中，这块矩形不可能被更精细地划分。否则沿着分割线将矩形一分为二，并递归处理两个子矩形。在每层递归中找到的分割线组合在一起就是划分 C 。

求 D 的方法与求 C 的类似，将与改为或即可。

设矩形大小为 $w \times h$ ，对于每个子矩形，找分割线的复杂度为 $O(wh)$ ，矩形最多被划分为 $O(wh)$ 个子矩形。故时间复杂度为 $O(w^2h^2)$ 。

9.4 2002 H: Silly Sort

题目大意

给定一个序列，序列中元素互不相同。现要将其按升序排序，每次可以交换任意两个数，交换的代价是两个数的和。求出将该序列按升序排序的最小总代价。

算法分析

将排序后的序列视为原序列的置换，那么序列可以被拆解成若干个置换圈。对于一个长度为 m 的置换圈，可以用其最小元 a 进行 $m - 1$ 次交换，将圈中所有元素调整到对应的位置，也可以用整个序列的最小元 b 与 a 交换，再用 b 通过 $m - 1$ 次交换将圈中其它元素调整到对应位置，最后再交换 a 和 b 。对每个置换圈，取上述两种策略中较优的一种，每个圈的最少代价的和即为所求。

元素的值的数量级为 $O(n)$ ，故排序的复杂度为 $O(n)$ ，找到所有置换圈并处理的复杂度也是线性，故总时间复杂度为 $O(n)$ 。

9.5 2001 A: Airport Configuration

题目大意

有机场布局方案若干，每种方案有一个客流指数。客流指数可以根据一些繁琐的定义计算。试将这些方案按照客流指数从小到大排序并输出。

算法分析

简单的模拟排序题。时间复杂度 $O(mn^2 + m \log m)$ ， m 为布局方案数， n 为机场的规模。

9.6 2001 B: Say Cheese

题目大意

在一块巨大的奶酪中有两只小虫，其中一只能动，另一只不会动。奶酪中有一些气泡，小虫在气泡中移动花费时间忽略不计。小虫能在 10s 咬穿 1 毫米的奶酪，求会动的小虫找到另一只小虫的最短时间。

算法分析

从气泡 (P_1, r_1) 到气泡 (P_2, r_2) 的时间为 $\max\{dis(P_1, P_2) - r_1 - r_2, 0\}$ 。在气泡与气泡间连边，权值为从其中一个气泡到另一个的时间。两只小虫所在位置可以视为两个半径为 0 的气泡。至此问题已转化为单源最短路径问题，用 dijkstra 算法求出起点与终点间的最短路。

设气泡数为 n ，则总时间复杂度为 $O(n^2)$ 。

9.7 2000 B: According to Bartjens

题目大意

有一个值为 2000 的表达式，其符号被丢失了，只剩下一个长度为 n ($n \leq 9$) 的数字串。试在数字之间添加 $+$, $-$, \times 号，使其成为一个合法的表达式，且值为 2000。

算法分析

注意到 n 非常小，枚举所有方案并计算即可。

时间复杂度 $O(n3^n)$ 。

9.8 2000 E: Internet Bandwidth

题目大意

有一个网络，网络中有一些连接，已知这些连接所连接的机器及其带宽，给定两个节点 S 和 T ，求 S 到 T 的带宽。

算法分析

建立网络流模型，对于每条连接 (u, v, c) ，从 u 到 v 连一条容量为 c 的弧，且从 v 到 u 连一条容量为 c 的反向弧。 S 到 T 的带宽即为该网络从 S 到 T 的最大流。

用 dinic 算法求出最大流即可，时间复杂度 $O(n^2m)$ 。

9.9 2000 F: Page Hopping

题目大意

给定一个有向图 $G = (V, E)$ ，求 G 中各点的最短平均距离。

算法分析

直接运用 Floyd 算法即可。设节点数为 n ，则时间复杂度 $O(n^3)$ 。

10 Volume X

10.1 1999 H: flooded!

题目大意

一个地段可被分为 $m \times n$ 个 $10\text{m} \times 10\text{m}$ 的区域，每个区域内的所有位置的海拔都是相同的。当该地段发生积水时，积水首先积在海拔较低的区域，随着积水量的增大，海拔较高的区域也会发生积水。为了方便，假定较高区域的积水能完全排放到较低的区域中（即使其被更高的区域所包围）。给出该地段的总积水量及每个小区域的高度，求出该地段积水的高度，以及积水区域的百分比。

算法分析

将所有区域的高度从小到大排序，设这个序列为 $\{h_1, h_2, \dots, h_{mn}\}$ 。这个序列将答案可能的范围分成了 mn 个区间 $[h_1, h_2), [h_2, h_3), \dots, [h_{mn}, +\infty)$ 。答案必然落在其中的一个区间内，当答案所在区间确定时，积水区域的总面积也被确定。我们可以枚举所有的区间，通过解一元一次方程判断答案是否在该区间内，如果不在则继续枚举，直到找到答案为止。

时间复杂度 $O(mn \log(mn))$ 。

10.2 1998 A: Crystal Clear

题目大意

有一种材料由晶体及其赖以生长的网格组成，网格相邻两个格点相距为 1，每个格点都会生长出一个直径为 1 的晶体。现在要对材料进行切割，切割的过程中，若某个晶体被切割，且割线不经过圆心，则该晶体会被破坏。沿切割线切割会割出一个简单多边形，该多边形材料块的有效面积为在该材料块内未被破坏的晶体（或晶体的一部分）的总面积。以顺时针顺序给出多边形 n ($n \leq 25$) 个顶点的坐标，试求该多边形的有效面积，坐标绝对值不超过 250。

算法分析

由于多边形坐标绝对值不超过 250，我们可以枚举坐标范围在 $[-250, 250]$ 内的所有格点。

对于每个格点 P 及其对应的晶体 Γ ，依次做如下的步骤：

(1) 判断 Γ 是否未被破坏，该问题等价于判断 P 到多边形各边的距离是否均不小于晶体半径。若 Γ 被破坏则退出，否则转 (2)。

(2) 判断 P 是否为多边形顶点，如是则将答案加上 Γ 在多边形内的部分的体积并退出，否则转 (3)。

(3) 判断 P 是否在多边形边上，如是则将答案加上单个晶体面积的一半，否则转 (4)。

(4) 判断 P 是否在多边形内部，该问题可用经典的射线法解决。若 P 在多边形内部，由 (1) (2) (3) 的判断结果可以得出， Γ 被多边形完整包含，将答案加上单个晶体的面积。

枚举完所有格点后得到的答案即为所求。上述 4 个步骤复杂度均为 $O(n)$ ，设枚举的格点数为 m ，则总时间复杂度为 $O(mn)$ 。

10.3 1998 B: Flight Planning

题目大意

现已知飞机的空速 V_{cruise} ，飞行效果最佳的飞行高度 A_{opt} ，在海拔为 A_{opt} 的高度飞行时每小时消耗的燃料 G_{opt} ，飞机每低于或高于 A_{opt} 1,000 英尺每小时消耗的额外燃料 G_{extra} ，飞机飞行高度每提升 1000 英尺消耗的燃料 C （飞行高度下降不需要消耗燃料，飞机飞行的高度变化可在瞬间完成）。飞机的航程中要经过 n 个航段，给定每个航段的长度 s_i 、风速关于海拔高度的函数（用两个数 a_i, b_i 确定， a_i 表示在海拔 20,000 英尺的风速， b_i 表示海拔在 40,000 英尺的风速，风速关于海拔线性增长），试确定飞机在每一段的飞行高度（必须在 20,000 英尺到 40,000 英尺之间，且为 1,000 英尺的整数倍），使飞机的总油耗最小，当有多种飞行方案使得总油耗最小时，输出字典序最小的方案。

算法分析

设 $f(i, j)$ 表示当第 i 段的飞行高度为 $j \times 1000$ 英尺时，第 i 段到第 n 段的最小总油耗。则有

$$f(i, j) = \min \left\{ f(i+1, k) + \max \{k - j, 0\} \times C + \frac{s_i}{v(i, j)} (G_{\text{opt}} + G_{\text{extra}} |j - A_{\text{opt}}|) \right\}$$

边界 $f(n+1, i) = 0$ 。

其中 $v(i, j)$ 表示在第 i 段海拔为 j 时的风速， $v(i, j)$ 可由 a_i, b_i 及 V_{cruise} 在 $O(1)$ 时间内确定。

最终的目标值为 $\min \{Ch + f(1, h)\}$ 。

设 m 为飞机可能所处的高度的数目 ($m = 21$)，则状态的数目为 $O(nm)$ ，单个状态转移复杂度为 $O(m)$ ，总时间复杂度为 $O(nm^2)$ 。

10.4 1998 D: Page Selection by Keyword Matching

题目大意

现需要维护一个搜索引擎。每次操作可以定义一个网页或者进行查询。定义网页时，会给出该网页的若干关键字（不超过 n 个），按照与该网页联系的紧密程度降序排列。查询操作则会给出若干关键字（不超过 n 个），按照重要程度降序排列。网页与询问之间定义了一个相关强度，若有一个关键字在网页和询问中同时出现，且该关键字为网页第 i 重要的关键字、询问第 j 重要的关键字，那么该关键字对相关强度的贡献为 $(n - i + 1)(n - j + 1)$ 。对于每个询问，试求出关联强度最大的前 5 个网页。

算法分析

对于每个询问，枚举每个网页，计算该网页与询问的关联强度并从大到小排序，取关联强度最大的前 5 个即可。计算每个网页的关联强度，可以枚举询问中的每个关键字，用 C++ 关联容器快速查找出该关键词在网页中的重要程度，将该关键字的贡献值累加到答案。

设询问数为 Q ，网页数为 P ，时间复杂度 $O(QPn \log n)$ ，这里假设比较两个字符串是否相等的时间复杂度为 $O(1)$ 。

10.5 1998 E: Petri Net Simulation

题目大意

有一个网络由 n 个库所与 m 个变迁构成，初始时每个库所都有一定数量的令牌。每个变迁的发生，需从若干个指定的输入库所（可以重复）各获取一个令牌（要求输入库所中有足够数目的令牌），变迁发生后，变迁的若干个输出库所（可以重复）各补充一个令牌。如果网络没有可以发生的变迁，该网络将死去。给定一个网络及规定的变迁的次数 p ，试判断该网络能否完成规定数目的变迁，并输出该网络在完成规定数目的变迁或死去后，每个库所剩下的令牌数。保证答案唯一。

算法分析

注意到数据保证最终的答案唯一，我们只需模拟变迁的全过程：每次找一个可行的变迁令其发生，直到网络死掉或者达到了规定的变迁次数。时间复杂度为 $O(nmp)$ 。

10.6 1998 G: Spatial Structures

题目大意

本题有两问，第一问给定一个位图，构造该位图的四分树，并输出颜色为黑色的叶节点编号。第二问给出一棵四分树的黑色叶节点的编号和位图的边长，求出整个位图。

算法分析

第一问非常简单，直接按照四分树的定义构造四分树，找到所有黑色的叶节点，按照编号排序输出即可。时间复杂度 $O(n^2)$ 。

第二问可以设计一个递归函数，参数为当前处理区域的左上角和右下角，以及在这个区域中的黑色叶节点的集合。对于每个在该区域中的叶节点，根据其模 5 的值判定其属于哪个子区域（或者覆盖整个区域），若该区域没有黑色叶节点或者有一个覆盖整个区域的黑色叶节点则将该区域填满并回溯，否则将当前区域四分，对于每个黑色节点，将其编号整除以 5，投放到对应的区域中。之后对 4 个区域递归分治即可。时间复杂度仍然为 $O(n^2)$ 。