IOI2014中国国家集训队第一次作业

浙江省镇海中学 岑若虚

Contents

1 ACM/ICPC World Finals 1998 **4**
1.1 B. Flight Planning 4
1.2 C. Lead or Gold 4
1.3 G. Spatial Structures 5

2 ACM/ICPC World Finals 1999 **6**
2.1 A. Bee Breeding 6
2.2 C. A Dicey Problem 6
2.3 D. The Fortified Forest 7
2.4 E. Trade on Verweggistan 7
2.5 H. Flooded! 8

3 ACM/ICPC World Finals 2000 **8**
3.1 A. Abbott's Revenge 8
3.2 B. According to Bartjens 9
3.3 C. Cutting Chains 9
3.4 E. Internet Bandwidth 9
3.5 F. Page Hopping 10
3.6 H. Stopper Stumper 10

4 ACM/ICPC World Finals 2001 **11**
4.1 A. Airport Configuration 11
4.2 B. Say Cheese 11
4.3 F. A Major Problem 11
4.4 H. Professor Monotonic's Network 12

5 ACM/ICPC World Finals 2002 **12**
5.1 A. Balloons in a Box 12
5.2 E. Island Hopping 13
5.3 G. Partitions 13
5.4 H. Silly Sort 14

6 ACM/ICPC World Finals 2003 **15**
6.1 A. Building Bridges . . . 15
6.2 B. Light Bulbs . . . 15
6.3 F. Combining Images . . . 15
6.4 H. A Spy in the Metro . . . 16
6.5 J. Toll . . . 17

7 ACM/ICPC World Finals 2004 **17**
7.1 C. Image is everything . . . 17
7.2 E. Intersecting Dates . . . 18
7.3 F. Merging Maps . . . 18
7.4 H. Tree-Lined Streets . . . 19

8 ACM/ICPC World Finals 2005 **19**
8.1 C. Lots of Sunlight . . . 19
8.2 I. Workshops . . . 20
8.3 J. Zones . . . 20

9 ACM/ICPC World Finals 2006 **21**
9.1 A. Low Cost Air Travel . . . 21
9.2 B. Remember the A La Mode! . . . 21
9.3 D. Bipartite Numbers . . . 22
9.4 E. Bit Compressor . . . 23
9.5 G. Pilgrimage . . . 23
9.6 I. Degrees of Separation . . . 24
9.7 J. Routing . . . 24

10 ACM/ICPC World Finals 2007 **25**
10.1 A. Consanguine Calculations . . . 25
10.2 B. Containers . . . 26
10.3 C. Grand Prix . . . 26
10.4 J. Tunnels . . . 27

11 ACM/ICPC World Finals 2008 **28**
11.1 A. Air Conditioning Machinery . . . 28
11.2 B. Always an Integer . . . 28
11.3 E. Huffman Codes . . . 29
11.4 F. Glenbow Museum . . . 30
11.5 I. Password Suspects . . . 31
11.6 J. The Sky is the Limit . . . 31
11.7 K. Steam Roller . . . 32

12 ACM/ICPC World Finals 2009 **32**
12.1 A. A Careful Approach . . . 32
12.2 B. My Bad . . . 33
12.3 F. Deer-Proof Fence . . . 33
12.4 H. The Ministers' Major Mess . . . 34

13 ACM/ICPC World Finals 2010 **34**
13.1 B. Barcodes . . . 34
13.2 C. Tracking Bio-bots . . . 35
13.3 D. Castles . . . 35
13.4 H. Rain . . . 36
13.5 J. Sharing Chocolate . . . 37

14 ACM/ICPC World Finals 2011 **37**
14.1 A. To Add or to Multiply . . . 37
14.2 B. Affine Mess . . . 38
14.3 C. Ancient Messages . . . 38
14.4 E. Coffee Central . . . 39
14.5 F. Machine Works . . . 39
14.6 H. Mining Your Own Business . . . 40
14.7 J. Pyramids . . . 41
14.8 K. Trash Removal . . . 41

15 ACM/ICPC World Finals 2012 **42**
15.1 B. Curvy Little Bottles . . . 42
15.2 C. Bus Tour . . . 42
15.3 D. Fibonacci Words . . . 43
15.4 L. Takeover Wars . . . 43

16 ACM/ICPC World Finals 2013 **44**
16.1 A. Self-Assembly . . . 44
16.2 B. Hey, Better Bettor . . . 45
16.3 D. Factors . . . 46
16.4 E. Harvard . . . 46
16.5 F. Low Power . . . 47
16.6 H. Матрёшка . . . 48
16.7 I. Pirate Chest . . . 48
16.8 J. Pollution Solution . . . 49

1 ACM/ICPC World Finals 1998

1.1 B. Flight Planning

Description

要进行一次有n个航段的飞行。飞机可以在高度20000到40000英尺飞行，在30000英尺飞行时每小时消耗燃料2000加仑，每低于或高于30000英尺1000英尺，每小时多消耗燃料10加仑。飞机起飞前和降落后的高度均为0。飞机的飞行高度可以瞬间变化，每升高1000英尺消耗燃料50加仑，下降不消耗燃料。飞机的空速是每小时400海里，实际速度还要加上风速，每个航段的风速是高度的线性函数。已知每个航段的长度，20000英尺处的风速和40000英尺处的风速。飞行高度必须是1000英尺的整数倍。求消耗的最少燃料以及每个航段的飞行高度。

Analysis

可以用dp解决该问题。用$f[i,j]$表示飞了前i个航段，第i个航段飞行高度为$1000j$英尺消耗的最少燃料。枚举上一个航段的飞行高度，按照题意计算改变高度消耗的燃料、飞行时间以及飞行消耗的燃料，取总燃料的最小值，并记录最优决策。最后取$f[n,j](j=20,21,\ldots,40)$的最小值作为答案，并按照记录的最优决策求出每个航段的飞行高度。

时间复杂度：$O(nm^2)$（$m=21$表示高度的个数）

空间复杂度：$O(nm)$

1.2 C. Lead or Gold

Description

有n中合金，第i种合金中A, B, C三种金属的比例是$a_i:b_i:c_i$，每种合金有足够多。要用这些合金混合成比例为$a:b:c$的新合金，问是否可能。

Analysis

设使用第i种合金的质量为x_i，新合金质量为k，则有

$$\begin{cases}\sum_{i=1}^{n}\dfrac{a_i}{a_i+b_i+c_i}x_i=\dfrac{a}{a+b+c}k\\ \sum_{i=1}^{n}\dfrac{b_i}{a_i+b_i+c_i}x_i=\dfrac{b}{a+b+c}k\\ \sum_{i=1}^{n}\dfrac{c_i}{a_i+b_i+c_i}x_i=\dfrac{c}{a+b+c}k\end{cases}$$

令$y_i = \frac{a+b+c}{(a_i+b_i+c_i)k}x_i$，则问题转化为以下方程组是否有解。

$$\begin{cases} \sum_{i=1}^{n} a_i y_i = a \\ \sum_{i=1}^{n} b_i y_i = b \\ \sum_{i=1}^{n} c_i y_i = c \end{cases}$$

将上式看做以y_i为未知数，0为目标函数的线性规划，用单纯形法判断是否有解即可。

时间复杂度：指数级

空间复杂度：$O(n^2)$

1.3 G. Spatial Structures

Description

一张黑白图片对应的四分树是通过将图片不断地分割成四个大小相等的象限来构建的。如果一个象限内的所有像素点颜色都相同，则这个象限内不再继续分割。否则该象限继续被分割成四个大小相等的子象限，直到象限内的像素点全都是黑色或白色的。四分树的根节点表示整张图片。每个非叶子节点有四个儿子，对应这个节点的四个子象限。叶子节点表示一个颜色相同的区域，因此没有被继续分割。

将非叶子节点到四个子象限的边分别编号为1,2,3,4，一个叶子到根的路径上的边的编号序列构成一个五进制数，将它转化为十进制作为这个叶子的标号。给你一个用01矩阵表示的图片，从小到大输出所有黑色叶子节点的标号，或给你所有黑色叶子节点的标号，求原图片。图片边长n是2的幂次。

Analysis

对于第一问，在假想的四分树上dfs，每次碰到黑色叶子节点，即当前子矩形是全黑的，则将当前的标号放入答案队列中。

对于第二问，将每个黑色叶子节点的标号转化为子矩形的位置，将它涂黑。

时间复杂度：$O(n^2 \log n)$

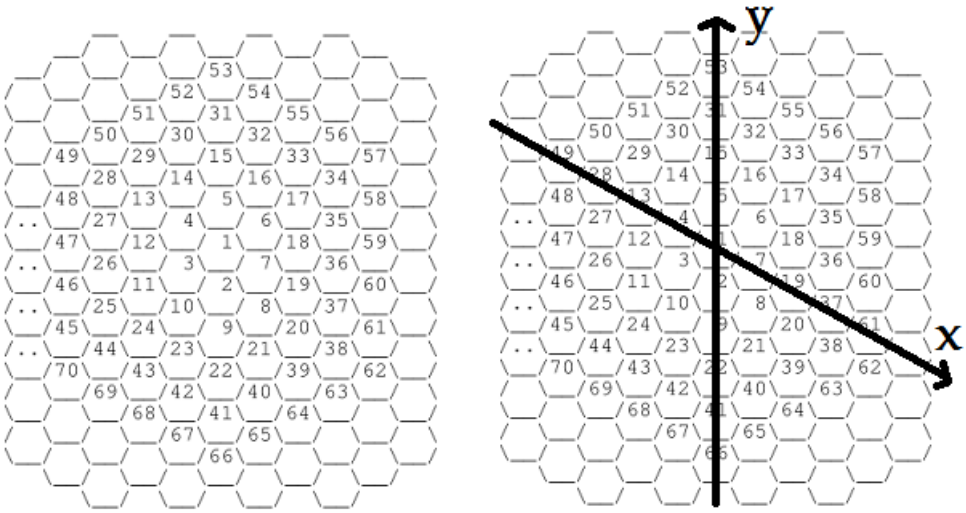
空间复杂度：$O(n^2)$

2 ACM/ICPC World Finals 1999

2.1 A. Bee Breeding

Description

用正六边形平铺平面，如图(a)螺旋给每个正六边形编号。每步只能走到有公共边的正六边形，给你两个正六边形A和B的编号，问从A走到B最少要走几步。



(a) (b)

Analysis

如图(b)建立斜坐标系。由于编号较小，可以直接模拟求出每个编号对应的坐标。现在只需求两点坐标差对应的点到原点的距离。每步只能将x或y坐标增减1，或将x和y坐标同时增减1。如果该点的x和y坐标同号则答案为$\max\{|x|,|y|\}$，否则答案为$|x|+|y|$。

时间复杂度：$O(n+q)$

空间复杂度：$O(n)$

2.2 C. A Dicey Problem

Description

给你一个带权值的$n\times m$网格。有一个底面为一个格子大小的骰子，初始位于某一个格子上，状态已知。如果当前骰子顶面的值和相邻格子的权

值相同或相邻格子的权值为-1，则骰子可以以公共边为轴滚动到相邻格子上。求一条最短的路径，使得骰子从起点出发，最后又回到初始格子。保证没有多解。

Analysis

由于骰子相对两面的数字和为7，只要记录底面和前面的数字，再结合预处理的信息就可以确定骰子的状态。将每个点拆成24个点表示骰子以不同状态走到这个点上，从初始状态开始bfs并记录方案，最后取起点对应的点中距离最短的，到它的路径即为答案。

时间复杂度：$O(6^2nm)$

空间复杂度：$O(6^2nm)$

2.3 D. The Fortified Forest

Description

有n棵树，每棵树看做一个点，已知位置、高度和价值。要删掉一些树，使得这些树能将其他树围起来，即这些树的总高度不小于其它树的凸包的周长。要使砍掉的树的总价值最小，求砍树方案。

Analysis

由于n较小，可以直接枚举删掉哪些树。每次求出未删去的树的凸包，判断周长是否合法，取总价值最小的方案作为答案。枚举时如果砍掉的树总价值比当前的答案还大就不用计算凸包了。

时间复杂度：$O(2^n n \log n)$

空间复杂度：$O(n)$

2.4 E. Trade on Verweggistan

Description

有w列数，你可以从每列数的第一个开始选择连续的一段数，选择一个数x的收益是$10-x$。问最大收益，以及达到最大收益需要选几个数。若方案多于10种只需输出最小的10种方案。

Analysis

第一问贪心即可，对于每列数，求出收益的前缀和，前缀和取到最大值时显然是最优的。w个最大值之和就是第一问的答案。

第二问可以用背包解决。我们在求前缀和的同时求出这列数取几个时前缀和取到最大值，设第i列数可以取$x_{i1}, x_{i2}, \ldots, x_{im}$个。为达到最大收益

至少要选x_{i1}个。剩下的方案可以看做体积分别为$x_{i2}-x_{i1}, x_{i3}-x_{i1}, \ldots, x_{im}-x_{i1}$的物品，且最多取一个。用$f(i,j)$表示前$i$列数取$j$个是否可行，可以通过$f[i-1, j-(x_{ik}-x_{i1})]$转移得到，初始值为$f[0, \sum_i x_{i1}] = 1$。这样就能求出所有方案。

时间复杂度：$O(n^2)$（n表示数的总数）

空间复杂度：$O(n)$

2.5 H. Flooded!

Description

给你$n \times m$网格中每个方格的高度，每个方格的底面积是$100\mathrm{m}^2$。这个网格上有水，除了比水面高的方格外每个方格上的水面高度都相同。已知总积水量为V，求水面高度以及在水面以下的面积占网格面积的百分比。

Analysis

将方格高度排序，设高度分别为a_i。假设前k低的方格在水面下，则$V = \sum_{i \le k} 100(h - a_i)$，即

$$h = \frac{V + 100 \sum_{i \le k} a_i}{100k}$$

枚举k，若求出的h满足$a_i \le h < a_{i+1}$（令$a_{nm+1} = \infty$）则这个h就是答案。

时间复杂度：$O(nm \log(nm))$

空间复杂度：$O(nm)$

3 ACM/ICPC World Finals 2000

3.1 A. Abbott's Revenge

Description

有一个9×9的网格，已知起点和初始方向，要走到终点。走的方式有一定限制。每个点的每个方向有一些特定的标记，如果你从这个方向进入这个点，那么下一步就只能走标记的方向。问从起点走到终点的最短路径，要输出方案，或判断无解。

Analysis

将每个点拆成四个点，分别表示从四个方向进入这个点的情况。设表示从方向d进入点u的新点为(u,d)，点u向方向d上走的下一个点为$trans(u,d)$。

对于点(u,d)，下一步能走的方向就可以通过u在方向d上的标记确定，对于能走的方向d'，将(u,d)向$(trans(u,d'),d')$连边。另外，新建点S,T。设起点为s，初始方向为d_0，将S向$(trans(s,d_0),d_0)$连边，将终点拆出的四个点向T连边。求S到T的最短路即为答案。

时间复杂度：$O(n^2 \log n)$（n为网格边长9）

空间复杂度：$O(n^2)$

3.2 B. According to Bartjens

Description

有n个数字，你可以在两个数字之间插入$+,-$或$\times$号，构成一个算式，中间没有运算符的数字被连成一个数，但不能有前导0。至少要插入一个运算符。求所有计算结果为2000的算式，按字典序输出，或输出无解。

Analysis

直接枚举每两个数字之间的运算符。由于符号的字典序都比数字小，按$\times,+,-$,空白的顺序枚举即可按字典序求解。枚举出一个算式后判断有无前导0和有无运算符，然后用表达式求值的方法，维护运算数栈和运算符栈，求出算式的运算结果。若结果为2000则输出。

时间复杂度：$O(4^n n)$

空间复杂度：$O(n)$

3.3 C. Cutting Chains

Description

给你一个无向图，你可以选择一个点，删掉它所连的所有边，然后给它连任意多的边。问至少要选择多少点才能将图变成一条链。

Analysis

由于图较小，可以枚举选择哪些点。若删掉选择的点后，图中所有的连通块都是链，且链数不超过选择的点数+1，则这种方案合法。

时间复杂度：$O(2^n n^2)$

空间复杂度：$O(n^2)$

3.4 E. Internet Bandwidth

Description

给你一个n个点m条边的无向图，求S到T的最大流。

Analysis

直接使用最大流算法即可。我用的是SAP。
时间复杂度：$O(n^2m)$
空间复杂度：$O(n)$

3.5 F. Page Hopping

Description

给你一个有向图，求任意两个节点之间的最短距离的平均值，不包括起点和终点相同的情况。输入中提到的点是强联通的，但有些未提到的点不用考虑。

Analysis

用floyd算法求出任意两点之间的最短路。判断每个点是否和其它点连通，将连通的点之间的最短路取平均值即可。
时间复杂度：$O(n^3)$
空间复杂度：$O(n^2)$

3.6 H. Stopper Stumper

Description

有一个底面为三角形，高为3cm的柱体，底面的三边长已知。要在里面放下三个不同的瓶塞，瓶塞是由两个高度都为1.5cm 的同心圆柱拼成的，两个圆柱的底面直径已知。圆柱底面必须与三棱柱底面平行，不过可以将瓶塞倒过来放。当两个瓶塞的大圆柱一个在上面，一个在下面时，这两个大圆柱可以重叠。只考虑3个瓶塞都接触三角形的两条边且没有两个不同的瓶塞接触到相同的两条边的情况。问能否将塞子放进三棱柱。

Analysis

首先求出三角形的内接圆半径，判断每个圆的半径是否超过内接圆半径，若超过则无解。现在每个瓶塞都可以放进三角形中，只要考虑瓶塞之间会不会重叠。枚举每个瓶塞接触的是哪个角的两条边。用余弦定理求出这个角，再根据大圆的半径即可求出圆心的位置。这样可以求出圆心之间的距离。然后枚举每个瓶塞的大圆柱在上面还是下面，对上面和下面分别判断三个圆是否都外离或外切，如果都是则有解。如果每种情况下都无解则无解。
时间复杂度：$O(1)$
空间复杂度：$O(1)$

4 ACM/ICPC World Finals 2001

4.1 A. Airport Configuration

Description

有n个起点和n个终点，告诉你每个起点和每个终点之间的客流量。每次给出两个$1 \sim n$的排列A, B，将起点分别放在$(A_i, 0)$，终点分别放在$(B_j, 1)$，求每个起点和每个终点之间的客流量乘以曼哈顿距离的总和。将所有询问的答案从小到大输出。

Analysis

由于数据规模较小，直接模拟即可。
时间复杂度：$O(qn^2)$
空间复杂度：$O(n^2)$

4.2 B. Say Cheese

Description

空间中有一些球，可能相交。小虫在球外的移动速度是0.1，在球内移动不用花费时间。问小虫从空间中一点到另一点的最短时间。

Analysis

起点和终点可以看做半径为0的球。要从一个球移动到另一个球，显然沿着两个球心的连线运动是最快的。以所有球（包括起点和终点）为顶点建图，两点间的边权为两球心距离减去两球半径（若两球相交则边权为0），用朴素的dijkstra算法求起点到终点的最短路即可。
时间复杂度：$O(n^2)$
空间复杂度：$O(n)$

4.3 F. A Major Problem

Description

12个音符依次如下循环排列：C/B$^\sharp$ C$^\sharp$/D$^\flat$ D D$^\sharp$/E$^\flat$ E/F$^\flat$ F/E$^\sharp$ F$^\sharp$/G$^\flat$ G G$^\sharp$/A$^\flat$ A A$^\sharp$/B$^\flat$ B/C$^\flat$。斜杠表示相同音符的两种表示方式。一个大调音阶是从某个音符开始，依次向右走2, 2, 1, 2, 2, 2步构成的长度为7的音符序列。另外，要求一个大调音阶中字母A到G都恰好出现一次，不能同时出现$\sharp$号和$\flat$号。因此大调音阶中每个符号都是确定的。给你两个大调音阶开头的音符，你要将第一个大调音阶的一些音符转化到第二个大调音阶的对

应位置。要判断这两个大调音阶是否合法，判断音符是否在第一个大调音阶中。

Analysis

直接模拟即可。比较方便的做法是给每种音符编码，预处理出每种音符开头的音阶和每个音符在每个音阶的哪个位置。由于每个字母对应的音符都是连续三个，每次只走1或2步，A到G又要恰好出现一次，所以音阶序列的字母序列一定和ABCDEFG循环同构。这样只需枚举出现♯号还是出现♭号就可以了推出整个音阶了。

时间复杂度：$O(n^2+q)$（n表示符号数）

空间复杂度：$O(n^2)$

4.4 H. Professor Monotonic's Network

Description

有一个对n个数操作的比较网络，它由k个比较器构成。一个比较器可以看做有序数对(u,v)，它将第u个数和第v个数比较，较小数放入第u个数，较大数放入第v个数。比较器依次进行操作，一个比较器仅当它要比较的两个数确定后才能开始操作，每次操作花费1单位时间。如果一个比较网络能将任意顺序的数排序，就称为排序网络。问这个比较网络是否为排序网络，以及整个比较网络的操作时间。

Analysis

根据0-1原理，我们只要枚举每个数是0或1，然后进行模拟，检验比较网络能否将这些数正确排序。若都能正确排序则比较网络是排序网络。对于第二问，记录每个数最后被处理的时间。依次处理每个比较器，这次处理的时间就是这两个数最后处理时间的较大值加1。

时间复杂度：$O(2^n k)$

空间复杂度：$O(n+k)$

5 ACM/ICPC World Finals 2002

5.1 A. Balloons in a Box

Description

给你一个长方体的盒子和一个点集。你可以以任意顺序选择这些点（也可以不选一些点），每次以选择的点为球心作一个最大的不与以前放的球或盒子相交的球。求球的最大总体积。

Analysis

除非一个点在其它的球中，否则选择它一定比不选优。由于点数较少，只需枚举选择的顺序然后模拟。第i个球的半径即为球心和前面的球或盒子的距离的最小值。若它已经在其它球中则忽略。

时间复杂度：$O(n!n^2)$

空间复杂度：$O(n)$

5.2 E. Island Hopping

Description

有n座岛屿，第i座岛屿的坐标是(x_i, y_i)，居民数是m_i。要在岛屿间架设电缆使得每座岛屿都与1号岛屿相连。所有电缆都是同时开始匀速建造的，当某个岛屿与1号岛屿连通的时候，这座岛屿上的所有居民就接入了互联网。在电缆总长度最小的前提下，求所有居民接入互联网的平均时间。

Analysis

用prim算法求出最小生成树。点u上的居民接入互联网的时间即为1到u的路径上的最大边。从1号点开始dfs即可求出所有居民接入互联网的平均时间。

事实上，在不同的最小生成树中1到u的最大边都是相同的。假设两棵最小生成树T_1, T_2中1到u的最大边分别为e_1和$e_2(e_1 < e_2)$，我们将T_2中的e_2去掉，将与1相连的点染红色，与u相连的点染蓝色。考虑T_1中1到u的路径，由于1和u颜色不同，一定存在相邻两个点颜色不同。在T_2中将这两个点相连，由于这条边一定比e_2短，我们就得到了一个更小的生成树，产生矛盾。因此在不同的最小生成树中，每个点的瓶颈边都是相同的，答案也是相同的。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

5.3 G. Partitions

Description

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。给你两个矩形的划分A和B。求1. 通过对A或B进行进一步的划分都能够得到的最粗糙的划分；2. 再进行进一步的划分既能够得到A又能够得到B的最精细的划分。划分用网格的形式表示。

Analysis

将矩形的划分看做网格中单位边的集合。考虑第一问的答案X，由题意A和B都是X的子集，于是$A \cup B$也是X的子集。而$A \cup B$就是合法的划分，它又是最粗糙的，所以X即为$A \cup B$。

考虑第二问的答案Y，由题意Y是A和B的子集，于是Y也是$A \cap B$的子集。但是$A \cap B$不一定合法，可能会出现悬在连通块内的划分边（删掉即可）或不是矩形的连通块。由于每次划分不一定是将某个子矩形一分为二，所以不能枚举产生Y的划分。我们需要将不是矩形的连通块合并成矩形的连通块。对于每个连通块，求出它横纵坐标的范围，在该范围内的所有方格所在的连通块都要和该连通块合并。合并后坐标范围可能会变化，再和新的范围内的连通块合并。不断迭代直到坐标范围不变，此时就得到了一个矩形的连通块。然后将该矩形内的单位边都删掉，这样如果后面的连通块要和它合并仍然能保证前面连通块的正确性。处理完后得到的就是答案。

时间复杂度：$O(n^5)$

空间复杂度：$O(n^2)$

5.4 H. Silly Sort

Description

有一些互不相同的数，每次可以交换两个数，代价是两个数的和。求将所有数排序的最小代价。

Analysis

将排序过程看做原数列到有序数列的一个置换，并将这个置换分解成几个循环。考虑某一个循环，设其中有m个数，和为s，最小值x。如果这个循环不与其它循环交换，则每次用循环中最小的数和正确的数交换是最优的，总代价为$s + (m-1)x$。若循环中某个数与另一个循环中的某个数交换，则两个循环被合并。不妨设另一个循环中有m'个数，最小值$x' < x$。同两个循环分别处理相比，增加的代价为这次交换的代价，减少的代价为$(x - x')(m-1)$。显然用循环中的最小值和全局最小值交换能使增加的代价最小，减少的代价最大。对于每个循环取直接处理的代价和用最小值与全局最小值交换后的代价的较小值，总和即为答案。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

6 ACM/ICPC World Finals 2003

6.1 A. Building Bridges

Description

$n \times m$的网格上有一些格子被建筑物占据，另一些格子是空地。格子是八连通的，每个建筑物占着一些连通的格子。你可以在网格的边上建立道路，道路只能直线连接两个建筑。问通过修建道路能使不连通的建筑群最少有几个，在不连通的建筑群最少的前提下最短需要修建多长的道路。

Analysis

先通过bfs找出所有的建筑。以建筑物为点建图。对于每条横向或竖向的网格线，找到线上面和下面一行所有有建筑物的格子，这些格子按线的方向排序后，给相邻格子对应的建筑物连边。这样边最多只有$2nm$条。在建出的图中求最小生成树即可求出答案。

时间复杂度：$O(nm\log(nm))$

空间复杂度：$O(nm)$

6.2 B. Light Bulbs

Description

给你两个二进制数A和B，求一个二进制数X使得经过如下操作后A变成B：如果X的第i位为1，则将A的第i位、第$i-1$位和第$i+1$位（如果存在）取反。多解输出1的位数最少的X，还有多解输出数值最小的X。

Analysis

A的第i位受到X的第i位、第$i-1$位和第$i+1$位（如果存在）的影响。如果A的第i位与B的第i为相同，X的第i位、第$i-1$位和第$i+1$位（如果存在）异或和为0，否则异或和为1。枚举X的第一位，接下来就可以不断根据A的第i位和X的前i位推出X的第$i+1$位，从而推出整个X。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

6.3 F. Combining Images

Description

用如下方式递归给边长为2的幂次的01方阵编码：若方阵中所有元素都相同，编码以1开始，接下来是一个元素的值；否则将方阵分为四个大小相

同的方阵，编码以0开始，接下来依次是左上、右上、左下、右下方阵的编码。之后，在编码开头插入1，再在开头用补0使得编码的位数是4的倍数，并将得到的二进制编码转化为十六进制。给你两个大小相同的01方阵的十六进制编码，求它们对应位置进行与运算后的方阵的十六进制编码。

Analysis

仿照编码规则递归处理。考虑当前要处理的两个方阵的编码。如果有一个方阵的编码是10，即该方阵全0，则进行与运算后结果方阵也全0，结果编码也为10。如果有一个方阵的编码是11，即该方阵全1，则进行与运算后结果方阵与另一个方阵相同，结果编码即为另一个方阵的编码。如果两个方阵的编码都是0开头，则可以分别对四部分的编码递归进行处理，结果编码即为0与四部分的结果编码依次连接。需要注意四部分的结果编码可能都是10，此时需将它们合并，结果编码为10。

时间复杂度：$O(L)$（L表示输入编码长度）

空间复杂度：$O(L)$

6.4 H. A Spy in the Metro

Description

一条直线上有n个车站，1号车站和n号车站会分别在一定的时间向对面发车。经过同样路程的列车的运行时间都是一样的。已知相邻车站之间列车的运行时间和1号与n号车站所有的发车时间。你要在时刻0从1号车站开始乘车，在时刻T到达n号车站，可以任意换乘在同一车站的车，换车时间不计。要使没有乘车而在车站等待的时间最少，求最少等待时间。

Analysis

按时间从后往前dp。$f(i,j)$表示在时刻j从i号车站出发的最小等待时间。转移有三种：一是在原地等1单位时间；二是在正好有向右开的车到i的情况下向右坐一站；三是在正好有向左开的车到i的情况下向左坐一站。预处理出每个车站到起点的距离，以及每个时间是否发车，就可以$O(1)$完成转移。

时间复杂度：$O(nT)$

空间复杂度：$O(nT)$

6.5 J. Toll

Description

给你一个n个点，m条边的无向图，点分为城镇和村庄。你要带着一些货物从起点走到终点。设走某一步之前有x件货物，则走到一个村庄要支付1件货物，走到一个城镇要支付$\lceil \frac{x}{20} \rceil$件货物。要将$p$件货物带到终点，问出发时最少需要带多少货物。

Analysis

从后往前逆推，将减少货物变为增加货物，且到城镇增加的货物可以通过不断迭代计算出来。于是转化为求终点到起点的最短路，且逆推过程中货物只会增加不会减少，可以用dijkstra算法，每次选择一个当前费用最小的点向前更新。

时间复杂度：$O(m \log n)$

空间复杂度：$O(m \log n)$

7 ACM/ICPC World Finals 2004

7.1 C. Image is everything

Description

有一个$n \times n \times n$的大立方体，由单位立方体构成，其中一些单位立方体是缺失的。一个单位立方体的六个表面都会涂上同一种颜色，颜色用大写字母表示。从六个表面的方向分别扫描大立方体，得到大立方体的六个面的视图，点表示该位置可以被看穿。求最多有几个单位立方体。保证有解。

Analysis

先假设所有位置都有单位立方体，那么可以确定边上的面的颜色。如果有一个单位立方体的两个面的颜色不同，那么这个立方体不能存在。将它删掉之后，它表面的颜色要落到这个方向的下一个立方体上，这可能会引发新的矛盾。当删到没有矛盾时，剩下的立方体就是最优解。可以用类似bfs的方法模拟这个过程。虽然一个立方体出队后可能再次进队，但由于每个立方体最多被删一次，而bfs的队列只有在有立方体被删掉时才会增加$O(1)$个元素，因此队列总长为$O(n^3)$。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^3)$

7.2 E. Intersecting Dates

Description

已知一些时间段内的信息，要询问另一些时间段内的信息。为减少询问代价，我们只关心未知的信息。也就是求询问的时间中不属于已知时间的时间段。时间段都用日期区间的形式表示，输出要合并为最简形式。

Analysis

显然只有在已知或询问区间开始或结束的关键时间答案才会变化。将输入的区间分成已知区间开始、已知区间结束、询问区间开始、询问区间结束四种事件，然后按时间顺序处理，就可以得到每个关键时间是否在已知时间或询问时间内，从而得到每个关键时间是否在答案内。这样就可以得到答案每次开始和结束的日期，从而求出答案的时间段。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

7.3 F. Merging Maps

Description

有n个矩形地图，编号$1 \sim n$，地图用二维字符数组表示，尺寸不超过$m \times m$，大写字母表示可识别的区域，‘-’表示不可识别的区域。两个地图可以重叠其中一部分并合并成一个大矩形地图，未被小地图覆盖的部分用‘-’填充。一次合并的得分为相同大写字母重叠的位置数。进行合并要求得分大于0，并且不能有两个不同的大写字母相重叠。但字母可以与‘-’重叠，合并后得到字母。一次合并的偏移量(r, c)表示将第一张地图的左上角放在第0行第0列，第二张地图的左上角放在第r行第c列进行重叠。

每次合并得分最大的一对地图（优先处理拥有最小序列编号的地图对），然后删去这一对地图，并将新地图插入序列最后，编号为原来的最大编号加1。如果两个地图可以以多种方式合并，并且取得同样的最高分数，那么使用偏移量中r最小的方案，如果r最小的情况下仍有多种方案，则使用c最小的方案。输出不能进行合并时序列中剩下的所有地图。

Analysis

按题意模拟即可。注意每对地图的最大得分是不变的，可以把得分和偏移量先预处理出来。插入新地图时也计算它和前面地图的最大得分。这样可以加速每次求最大得分的地图。

时间复杂度：$O(n^6 m^2)$

空间复杂度：$O(n^3 m^2)$

7.4 H. Tree-Lined Streets

Description

给你一些街道，用线段表示。要在街道上种树（放点），需满足如下条件：同一条街道上相邻两棵树距离不小于50，树到它所在街道和其它街道的交点（不包括端点）的距离不小于25。求最多总共能种几棵树。相邻两交点的距离不会是25的整数倍，街道不会在端点处相交。

Analysis

每条线段可以单独考虑。由于交点两侧的树都要和交点相距25以上不会相互影响，线段上被交点分开的每个小段也可以单独考虑。对于每条线段，求出其它线段和它的所有交点并求出每小段的长度。若位于两个交点之间的一小段长度为x，可以种$\lfloor \frac{x}{50} \rfloor$ 棵树。若位于端点和交点之间的一小段长度为x，可以种$\lfloor \frac{x+25}{50} \rfloor$棵树。若线段上没有交点，长度为$x$，可以种$\lfloor \frac{x+50}{50} \rfloor$ 棵树。总共的树数即为答案。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

8 ACM/ICPC World Finals 2005

8.1 C. Lots of Sunlight

Description

东西方向的地平线上有n栋公寓楼，公寓楼是一边在地平线上的矩形，由一列房间构成，每个房间都是大小相同的矩形。已知每栋公寓楼的层数和相邻两栋公寓楼之间的距离。每次对于一个房间，判断是否存在，若存在求它能被太阳直射的时间段。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。日出和日落的时间已知，阳光的角度均匀变化。

Analysis

考虑某个房间，正午的时候它一定受到太阳直射。考虑上午的情况，将阳光绕着房间左下角逆时针旋转，直到碰到其它公寓楼之前，这个房间都受到直射。但碰到其它公寓楼之后就不受到直射了。下午也是类似的情况。由于数据规模较小，只需枚举即可。先列出每栋公寓楼顶部的两个点。对于每个询问，求位于房间左侧的点与房间左下角的连线与地平线所成的最大角度，将这个角度占平角的比例换算成秒数，即可求出直射开始

的时间。求位于房间右侧的点与房间右下角的连线与地平线所成的最大角度，即可求出直射结束的时间。

时间复杂度：$O(nq)$

空间复杂度：$O(n)$

8.2 I. Workshops

Description

有w个会议和r个房间。会议有一定的人数和时间，房间有一定的人数和时间限制，当会议的人数和时间都不超过房间的限制时，这个会议可以在这个房间内举行。一个房间只能举行一个会议。不能在房间内举行的会议只能在帐篷中举行。问最少有多少这样的会议，以及在这样的会议最少的前提下，最少有多少人要在帐篷中参加会议。

Analysis

将会议和房间看做横坐标为人数，纵坐标为时间的点，则一个房间点可以和一个在它左下方的会议点匹配。我们按人数从大到小处理。处理到一个房间是将它加入备用列表。处理到一个会议(x_i, y_i)时，横坐标满足要求的房间都在备用列表里面。如果有纵坐标满足要求的房间，就直接将这个会议和纵坐标大于y_i的房间中纵坐标最小的一个匹配。这可以用线段树或平衡树找到。

这样贪心得出的答案为什么是最优的呢？把会议和房间看做二分图的两个部分。如果某一次的将会议u和房间v的匹配不是最优的，那么之后一定有另一个会议（设为p）与v匹配，然后u又与另一个房间（设为q）匹配，形成一个匈牙利算法中的增广路。但是由于v是能与u匹配的房间中y最小的，p一定能与q匹配，u和v的匹配依然是最优的。另外由于每次尽量给人数大的会议匹配，所有第二问的答案也是最优的。

时间复杂度：$O(w+r)\log(w+r)$

空间复杂度：$O(w+r)$

8.3 J. Zones

Description

有n个区域，已知每个区域的面积。还已知一些区域的公共部分的面积，给出的公共部分是只属于某些区域，不属于其它区域的。公共部分有m个。问在这n个区域中选k个的并的面积最大是多少。

Analysis

由于n较小，可以直接枚举选择哪些区域。设选择的区域集合为S。先求出这些区域的面积之和，然后减去每个公共部分重复算的面积，就是这些区域的并的面积。对于只属于区域集合T，面积为x的公共部分，重复算了$|S \cap T| - 1$次，要减去$(|S \cap T| - 1)x$。求每种方案的总面积的最大值即为答案.

时间复杂度：$O(\binom{n}{k}m)$

空间复杂度：$O(n + m)$

9 ACM/ICPC World Finals 2006

9.1 A. Low Cost Air Travel

Description

有Nt张机票，一张机票会依次标有最多m个城市，且有一定的价格。使用时必须从这张机票的第一个城市出发，依次经过机票上所标的城市，但可以在中途停止使用这张机票改用其它机票。停止使用一张机票后这张机票就作废了。机票可以重复购买。现在有Ni个旅行路线，一个旅行路线要依次经过最多m个城市。求每个旅行路线的最小花费。

Analysis

先将城市编号离散化，最多可能有$Nt \cdot m$个城市。由于旅行路线上的城市要依次经过，考虑使用动态规划。用$f(i, j)$表示已经依次经过旅行路线的前i个城市，现在在城市j的最小花费。由于可以走到路线之外的城市，转移可能形成环，可以用最短路算法求解。对于每种状态，枚举从j出发的机票，再枚举使用这张机票走几步进行转移。

时间复杂度：$O(NiNt^2m^3)$

空间复杂度：$O(Nt^2m^2)$

9.2 B. Remember the A La Mode!

Description

有P种饼片和I种冰激凌，每种有一定数量，饼片和冰激凌的总数量相同。已知每种饼片和每种冰激凌搭配的价格。要将所有饼片和冰激凌配对，求最大总价格和最小总价格。

Analysis

建立源点和汇点，从源点向每种饼片连容量为饼片数量，费用为0的边。从每种冰激凌向汇点连容量为冰激凌数量，费用为0的边。每种饼片向每种冰激凌连容量为无穷大，费用为它们搭配的价格的边。求最小费用最大流即为最小总价格。再将所有边的费用取负，求最小费用最大流即为最大总价格。

时间复杂度：$O(MinCostFlow(P+I, PI))$

空间复杂度：$O(PI)$

9.3 D. Bipartite Numbers

Description

定义“二段数”为十进制表示中有且仅有两个数字a和b（$a \neq 0$），且所有a都出现在所有b前面的数。求n的倍数中最小的二段数，不包括n本身。

Analysis

令$f(a,i) = \underbrace{aaa\ldots a}_{i个a}$，则要求的二段数可以看做

$$f(x,l_1) + f(y-x,l_2)(1 \leq x < y \leq 9, y \neq x, 1 \leq l_2 < l_1)$$
$$或f(x,l_1) - f(x-y,l_2)(0 \leq y < x \leq 9, y \neq x, 1 \leq l_2 < l_1)$$

它是n的倍数。由于$f(a,i)$模n有不超过n的循环节，答案长度不会超过$2n$。

我们枚举答案的长度l_1和第一个字母x。为使答案最小，先按l_1 从小到大，再按x从小到大。为使答案至少为$2n$，要有l_1比n的位数大，或位数相同且x比n的首位大。先判定这种情况下是否有解，也就是要判定是否存在

$$f(d,l_2) \equiv n - f(x,l_1) \pmod{n} (1 \leq d \leq 9-x, l_2 < l_1)$$
$$或f(d,l_2) \equiv f(x,l_1) \pmod{n} (1 \leq d \leq x, l_2 < l_1)$$

等号右边是定值，我们只要预处理出$g(i,j)$表示只含一个1到i之间的数字，模n余j的最小的数是几位（如果存在），就可以$O(1)$判定。

求出l_1和x后再枚举l_2和y即可。

时间复杂度：$O(10n)$

空间复杂度：$O(10n)$

9.4 E. Bit Compressor

Description

用如下方法压缩一个二进制串：将连续的x个1替换为x的二进制表示，但当替换后长度不缩短时不进行替换。已知压缩后的二进制串，以及原串的长度L和1的个数n。你要判断能否确定原串。输出有多解、有唯一解或无解。压缩后串长m不超过40，原串长L不超过16000。

Analysis

将原串分成交替的一段1、一段0、一段1、一段0，那么压缩后会变成交替的一个数、一段0、一个数、一段0。这里的一个数一定是1开始的。我们将一段0看做一些由0分开的空串，那么整个串就是由0分开的一些二进制数，要使它们的和为n，且0的个数为$L-n$。需要注意的是2个1如果替换为10长度不会缩短，所以还会保留11的形态，而3个1压缩后也是11。如果有一段是11则看做2或3都可以。基于同样的理由，分出的一段不能是10。

我们可以从后往前搜索，每次枚举这一段数和上一段数分界的0，保证这一段数以1开始，且剩下的1的个数和0的个数非负。找到两个解就可以直接输出有多解。状态数最多为m^2L，且大部分是无效的，总的搜索量可以接受。

时间复杂度：$O(m^2L)$

空间复杂度：$O(m)$

9.5 G. Pilgrimage

Description

有一些人进行长途旅行，Jack管理公共资产。有四种操作：

1. COLLECT x：表示每位成员捐出x元钱作为公用；
2. IN x：表示新加入x位成员，新加入的成员每人需要捐出与当前人均公共资产相等的钱；
3. OUT x：表示有x位成员离开，离开的成员每人可以得到与当前人均公共资产相等的钱；
4. PAY x：表示Jack使用公共资产中的x元进行支付。

已知n个操作，要求任何时刻至少有一个成员，并且每次IN或OUT时捐出或得到的钱都是整数。求一开始的人数。判断无解、有有限解还是有无限解。若有有限解输出所有解，若有无限解输出最小解。

Analysis

假设一开始有x人，我们模拟人数的变化可以得到最小人数是$x+d$，根据$x+d\geq 1$可以得到x的最小值$Minans$。只要开始人数不小于$Minans$就可以满足任何时刻至少有一个成员。

另一个要求是每次IN或OUT之前人均公共资产都是整数。注意到除了PAY操作以外，其它三个操作都不会改变人均公共资产。假设一次IN或OUT后人数为x，此时人均公共资产是整数。经过几次PAY操作，总共支付s元后又碰到了IN或OUT操作，为使现在的人均公共资产也是整数，x必须是s的约数。也就是说，相邻两次IN或OUT操作之间PAY的总钱数必须是当前人数的倍数。第一次IN或OUT之前的PAY操作和最后一次IN或OUT之后的PAY操作则不用考虑。如果没有位于IN或OUT之间的PAY，则有无限解，最小解为$Minans$。如果有相邻两个IN或OUT之间PAY的总钱数是s，可以$O(\sqrt{s})$枚举s的约数，推出初始人数，模拟操作过程进行判定。符合要求的约数就是所有的解。

时间复杂度：$O(n\sqrt{nm})$（m表示一次PAY的最大钱数）

空间复杂度：$O(n)$

9.6 I. Degrees of Separation

Description

给你一个n个点的无向图，点用一个字符串表示，求距离最远的两个点的距离。

Analysis

用map或trie给点标号。用floyd算法求出任意点对距离，取最大值即为答案。

时间复杂度：$O(L+n^3)$（L表示总长度）

空间复杂度：$O(L+n^2)$

9.7 J. Routing

Description

给你一个n个点的有向图，要从1走到2再走回1，问经过的不同的点最少有几个，包括1和2。

Analysis

先用floyd算法求出任意点对最短路，记为$d(u,v)$。由于1到2的路径和2到1的路径可能共用若干段路，难以将路径分成独立的小段。但是，共用的

路径一定是按相同顺序排列的。将2到1的路径看做反图中1到2的路径。假设原图中的路径是$1 \to A \to B \to \cdots \to C \to D \to 2$，$A \to B$和$C \to D$这两段是共用的，那么反图中的路径一定是先经过$B \to A$，再经过$D \to C$，否则中间一段也共用是更优的。因此，两条路径可以看做交替分开走一段路和共用一段路。

用$f(u,v)$表示原图中从1走到u，反图中从1走到v经过的不同点的最少数目。转移有三种：

1. u到w有边的情况下从u走到w。若$w \neq v$用$f(u,v)+1$更新$f(w,v)$，否则用$f(u,v)$更新。
2. w到v有边的情况下从v走到w。若$w \neq u$用$f(u,v)+1$更新$f(u,w)$，否则用$f(u,v)$更新。
3. u和v不同的情况下共用从u到v的一段路。用$f(u,v)+d(u,v)-1$更新$f(v,u)$。

这样可以用最短路算法求出所有$f(u,v)$。答案即为$f(2,2)$。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

10 ACM/ICPC World Finals 2007

10.1 A. Consanguine Calculations

Description

ABO血型系统由两个等位基因决定，每个等位基因可以为A、B、O三种之一。父母会各自随机遗传一个等位基因给孩子。对应关系如下：

基因组合	AA	AB	AO	BB	BO	OO
血型	A	AB	A	B	B	O

Rh血型系统也由两个等位基因决定，每个等位基因可以为$+$、$-$两种之一。父母会各自随机遗传一个等位基因给孩子。只要有一个基因为$+$，血型即为$+$，否则为$-$。

将一个人的血型用ABO血型系统和Rh血型系统连起来表示。给你父亲、母亲、孩子三个人中两个的血型，求另一个人所有可能的血型。

Analysis

枚举未知血型，判断孩子的每个等位基因能否从父母遗传得到。

需要注意当父母血型未知时，孩子的基因组合可能有多种情况，如A+血型的基因组合可能为AA++、AA+−、AO++或AO+−。只要有一种基因组合是可能的，这种血型就是可能的。

时间复杂度：$O(1)$

空间复杂度：$O(1)$

10.2 B. Containers

Description

给你一个由大写字母组成的字符串。有许多栈，你要依次将字符串中的每个字母压入栈中，使得每一个栈中的字母从栈底到栈顶字典序递减。问最少需要几个栈。

Analysis

贪心，每次选取栈顶不小于当前字母的栈顶最小的栈压入，如果所有栈的栈顶都比当前字母大则需要新开一个栈。

这样做显然是正确的。一个栈对以后字母的影响只和栈顶有关。假设当前字母为i，有两个栈的栈顶分别为j和k，$i \leq j < k$。考虑将i分别压入这两个栈后的状态，其它的栈都相同，这两个栈的栈顶分别为i, k和i, j。之后j上面能压入的字母k上面也能压入，但k上面能压入的字母j上面不一定能压入。因此选择栈顶较小的栈压入是更优的。

实现时可以记录栈顶为某个字母的栈现在有多少个。

时间复杂度：$O(n|\Sigma|)$（Σ表示字符集）

空间复杂度：$O(n)$

10.3 C. Grand Prix

Description

要在斜坡上滑雪，斜坡看做与水平面夹角为θ的平面，赛道看做n条线段构成的折线，问最少要将折线顺时针或逆时针旋转多大的角度，使得每一段赛道都不是下降的，或者怎么旋转都不能做到。

Analysis

若$\theta = 0$则不用旋转。若$\theta > 0$，旋转后每条线段在前进的方向上x坐标都是不降的，也就是将每条线段的起点平移到原点，终点都在y轴及其右侧。我们先将所有的向量极角排序，如果有相邻两条向量的角度差不小

于π，那么所有向量都位于某条直线的一侧，将这条直线旋转到y轴就是一种可行的方案。如果不是一开始就符合要求，边上两条向量其中一条所在的直线一定是最优的。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

10.4 J. Tunnels

Description

给你一个无向图，点数为n，边数为m。有个间谍要从1号点走到0号点，你可以任何时候瞬间去掉图中若干条边。问在最坏情况下，要使间谍无法到达0号点，最少需要去掉几条边。

Analysis

如果我们要在间谍在某个点u中时封住他，显然可以删u和0之间的最小割集。既然机智的间谍会选择最优路径，我们是否可以直接等间谍走到最小割最小的点时删掉最小割集呢？问题在于我们可以多次删边，前面删掉的边也会影响间谍的走的路径。通过删掉少量的边迫使间谍走到一个最小割较小的点，这也是完全可能的。

我们可以用迭代的方法求出最优解。用$f[u]$表示间谍从点u出发，要删掉的最少的边数。先对于每个i令$f[i]$为i和0之间的最小割。之后，如果删掉x条边后，u到0的路径一定存在点v使得$f[v] \leq y$，则可以用$x+y$更新$f[u]$。这等价于删掉所有满足$f[v] \leq y$的点v后，u和0之间的最小割为x。由于f值最小的点不会被更新，可以每次用未被删掉的点的最小的f值作为y，删掉所有满足$f[v] \leq y$的点v后对剩下的点求最小割并更新。这样，f值会不断减小，当所有点都被删掉后达到最小值。由于每次更新都是合法的，最终求出的$f[u]$也是可行的，但它是否是最优的呢？例如，我们第一次删掉了x_1条边，第二次删掉了x_2条边使间谍走到了某些不利的点，这x_1条边和x_2条边中会不会有重复的边呢？

不加证明地给出这样一个结论：在图G和G删掉一条边构成的图G'中分别进行上述迭代，最后对每个点都有$f'[i] = f[i]$或$f'[i] = f[i] - 1$。从u出发，如果只删掉$f[u]-1$条边，可以证明间谍一定能逃跑。设$p = f[u]$。首先，一定存在一条从u到0的f值都不小于p的路径，否则$f[u]$可以更小。间谍会选择这样一条最优路径，假设当间谍在点v是删一条边，新的$f[v]$不小于$p-1$，而删边的机会也少了一次，不超过$f[v]-1$次。不断重复这个过程，当删边机会用完时，当前所在点f值至少为1，这意味着间谍仍可以到达0号点。因此$f[u]$是最优解。

时间复杂度：$O(n^2 maxflow(n,m))$，使用SAP算法最坏为$O(n^4 m)$

空间复杂度：$O(n+m)$

11 ACM/ICPC World Finals 2008

11.1 A. Air Conditioning Machinery

Description

一个弯管是L形排列的四个单位立方体，其中与其它单位立方体无公共点的两个单位正方形（即弯管两端的两个面）是接口。现在要用最多6个弯管在一个由单位立方体构成的长方体空间内接通入口和出口。相邻两个弯管的接口必须重合。入口和出口是空间外表面上的一个单位正方形，接通入口的弯管的接口要与入口重合，出口亦然。两个弯管不能占据同一个单位立方体。问最少需要几个弯管。

Analysis

暴力搜索即可。由于每一步搜索中弯管的一个接口的方向已经确定，可能的摆放方法只有8种，搜索量只有$8^6 = 262144$。注意到两个弯管不能占据同一个立方体，所以在搜索过程中要记录当前有哪些方格被占据，不能用双向广搜减少搜索量。并且到达出口时也要验证当前方向与出口方向一致。

时间复杂度：$O(8^6)$

空间复杂度：$O(m^3)$（m表示坐标范围）

11.2 B. Always an Integer

Description

给你一个形如$\frac{P(x)}{D}$的多项式，其中$P(x)$是次数不超过100 的整系数多项式，D是正整数。问x取任意正整数时该多项式是否都为整数。

Analysis

题目等价于判断命题$\forall x \in \mathbf{N}^+, P(x) \equiv 0 \pmod{D}$是否为真。因式分解等数学方法是难以做到的，但是我们可以轻松地对某个x_0计算出$P(x_0) \bmod D$。那么，要验证多少个数才能确保该式对所有x都成立呢？设$P(x)$的次数为d，则只需要验证$1, 2, \ldots, d+1$这$d+1$个数即可。归纳证明如下。当$d = 0$时，只要验证一个数1就可以了，因为此时结果都是相同的。当$d > 0$ 时，命题等价于$P(1) \equiv 0 \pmod{D}$且$\forall x \in \mathbf{N}^+, P(x+1) - P(x) \equiv 0 \pmod{D}$。令$Q(x) = P(x+1) - P(x)$，注意到$Q(x)$ 是一个$d-1$次的多项式，由归纳假设需要验证$1, 2, \ldots, d$这d 个数。我们验证了$P(1) \equiv 0$后，再验证$P(x) \equiv 0, x = 2, 3, \ldots, d+1$，就等价于验证了$Q(x) \equiv 0, x = 1, 2, \ldots, d$，从而验证了命题是否为真。

时间复杂度：$O(n^2)$（n表示次数）

空间复杂度：$O(n)$

11.3 E. Huffman Codes

Description

若已知文本中每个字母的出现频率，可按如下步骤构造哈夫曼编码：

1. 每个字母建立一个节点，权值为该字母的出现频率，构成一片森林。
2. 新建一个节点，选取森林中根节点权值最小的树作为新节点的左子树，根节点权值次小的树作为右子树（若最小的两棵树权值相等可任意排列）。新节点的权值为两个儿子的权值和。删去原来的两棵树。
3. 重复第2步直到森林中只有一棵树。
4. 对树上每条向左的边标0，向右的边标1。从根走到某个叶子，将经过的边上的标记连起来，就得到了这个叶子代表的字母的哈夫曼编码。

假设每个字母的出现频率都是0.01的正整数倍，和为1。告诉你每个字母的哈夫曼编码，问有多少种字母出现频率的分布情况能得到这样的哈夫曼编码。

Analysis

先根据哈夫曼编码建立01字母树，这就是构造过程中最后的那棵树，称为哈夫曼树。问题转化为有多少种给哈夫曼树上的节点赋权值的方法。

我们发现合法的赋值方法具有如下性质：

1. 深度相同的节点中，左边节点的权值不大于右边节点。
2. 深度较大的节点的权值不大于深度较小的节点。

也就是说，将哈夫曼树进行每次先访问右儿子的BFS，BFS序即为按权值降序排序的结果。

下面简要证明性质1。如果两个节点的父亲相同，由于构造过程中将权值较小的节点作为左儿子，结论成立。如果两个节点的父亲不同，设左边节点为u，其兄弟为u'，父亲为fu，右边节点为v，其兄弟为v'，父亲为fv。假设u的权值较大。那么在合并v和v'的时候，由于它们两个是权值最小和次小的节点，所以v和v'的权值都比u和u'小。由于父亲的权值等于儿子的权值之和，fv的权值也比fu小。fu仍然是fv左边的节点，这样可以不断往上推，最后总会出现两个节点的父亲相同的情况。这时左边节点的权值依然较大，这与前面的讨论矛盾。因此假设不成立，结论成立。性质2可以用类似的方法证明。

我们还可以证明，满足这两个性质的赋值方法就是合法的赋值方法。因此，按照从上到下，从右到左的顺序dfs，可以将权值范围缩减到最小。由于字母较少，合法方案不会很多（最多3万左右），dfs 过程中在合法权值范围内枚举即可。

时间复杂度：上限$O(50^n)$，实际远远不到。

空间复杂度：$O(n)$

11.4 F. Glenbow Museum

Description

对于一个内角均为90°或270°的多边形，将90°角记为R，270°角记为O，按逆时针顺序将每个角的字母连起来，得到一个R和O构成的字符串，称为角序列。一个多边形可能可以用多个角序列表示。如果存在一个点能看到多边形内的所有点（连线上没有边阻挡），称这个多边形是合法的。如果一个多边形是合法的，它的所有角序列也是合法的。求长度为n的合法的角序列有多少种。

Analysis

首先角序列要能构成多边形。设有x个R，y个O，则有

$$\begin{cases} x+y=n \\ 90x+270y=180(n-2) \end{cases} \text{解得} \begin{cases} x=\frac{n}{2}+2 \\ y=\frac{n}{2}-2 \end{cases}$$

显然n为奇数时无解。

考虑某个能看到多边形内所有点的点，它一定在所有边内侧半平面的交里。如果角序列里有两个连续的O，显然这样的半平面交为空，该角序列不合法。如果角序列里没有两个连续的O，可以把ROR构成的形状中的拐角缩得足够小，就不会影响合法性，效果和一个R相同。最后可以简化为RRRR即矩形，这样的角序列是合法的。

因此问题转化为将$\frac{n}{2}-2$个O放入$\frac{n}{2}+2$个R产生的$\frac{n}{2}+3$个空当中。由于角序列是循环的，首尾两个空当不能都放。答案即为

$$\binom{\frac{n}{2}+3}{\frac{n}{2}-2}-\binom{\frac{n}{2}+1}{\frac{n}{2}-4}=\binom{\frac{n}{2}+2}{4}+\binom{\frac{n}{2}+1}{4}$$

直接计算即可。

时间复杂度：$O(1)$

空间复杂度：$O(1)$

11.5 I. Password Suspects

Description

有一个长度为n的未知字符串S，已知m个字符串$a_1, a_2, \ldots, a_n$都是S的字串。问有多少种可能的S，若不超过42种按字典序输出所有可能的S。

Analysis

先建立所有子串的AC自动机，并预处理出每个状态加入每个字母会转移到哪个状态。问题转化为从起点开始在AC自动机上走n步，到达所有结束状态，有多少种走法。用$f[i, j, k]$表示前i步，位于状态j，结束状态的到达情况用二进制表示是k。由于转移过程中i一定会增大，可以dp求解，答案为$\sum_u f[n, u, 2^m - 1]$，其中u取AC自动机所有状态。

下面考虑要求方案的情况，由于此时答案很小，可以进行枚举。先倒过来再进行一次dp，求出哪些dp状态对答案有贡献。然后从初始状态开始枚举，每次按字典序枚举下一个字符，只走对答案有贡献的dp状态，$i = n$时就求出了一种方案。

时间复杂度：$O(nl2^m)$（l表示子串总长度）

空间复杂度：$O(nl2^m)$

11.6 J. The Sky is the Limit

Description

有n个等腰三角形，底边位于x轴上，腰在x轴上方。求它们的并的上边界的长度。上边界中间可能断开。

Analysis

取每个三角形的两腰，从左到右扫描，不断取当前最上面的点（如果存在），得到的就是要求的上边界。我们求出线段两两交点，按这些交点将横坐标分段，则每一段里任意两条线段一定有一条完全在另一条上方。对于每一段，求出最上面的线段，这一段的上边界都在这条线段上。计算它在这一段中的长度。每一段长度之和即为答案。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

11.7 K. Steam Roller

Description

给你一个网格图，经过一条边需要一定的时间，有些边不能经过。如果你在走某条边之前或之后改变方向，那么经过这条边的时间会变成两倍。这同样适用于一开始和最后的边。求从一个格点走到另一个格点的最短时间。

Analysis

我们把点拆成两类：一类是能自由转向的慢速点，一类是不能转向的快速点。那么只有在快速点之间的边不用翻倍，和慢速点相连的边都要翻倍。同时为了保证快速点不转向，要将快速点再拆成四个点，表示只能向某个方向走的快速点。原图中的一条边对应为连接两个慢速点的边，连接慢速点和该方向快速点的边以及连接两个该方向快速点的边。建完图后求起点的慢速点到终点的慢速点的最短路即可。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

12 ACM/ICPC World Finals 2009

12.1 A. A Careful Approach

Description

有n个实数，每个数位于一个特定区间$[l_i, r_i]$中。要求将数排序后，相邻两个数的差的最小值最大。求最大的最小值。

Analysis

先二分答案为x。由于n较小，可以直接枚举n个数的排列顺序。然后可以贪心确定每个数为前一个数加x和l_i的较大值。这样确定的是最小的取值，如果还是大于r_i则这种顺序下不存在差不小于x的方案。如果有一种排列顺序能求出符合要求的取值则答案大于x，否则小于x。

时间复杂度：$O(2^n n \frac{ans}{eps})$

空间复杂度：$O(n)$

12.2 B. My Bad

Description

有一个逻辑电路，有N个输入接口，G个逻辑门和U个输出接口。逻辑门可能是与门、或门、异或门、非门。每个逻辑门从一个或两个特定的输入接口或逻辑门接受输入，并产生一个运算结果。每个输出接口输出一个特定的逻辑门的运算结果。逻辑门之间不会构成环。逻辑门可能会出三种故障：总是输出0，总是输出1，总是输出相反的结果。最多只有一个门会出一种故障。进行了B次实验，已知每次实验的输入和输出。判断电路运行是否正确，若不正确判断可能是哪个门出了那种故障。

Analysis

将逻辑门拓扑排序，模拟正常情况下每次实验的输出，若都与实际输出相同则运行正确。否则枚举出故障的门和故障种类，再模拟每次实验的输出，判断是否与实际输出相同。

时间复杂度：$O(BG(N+G+U))$

空间复杂度：$O(B(N+G+U))$

12.3 F. Deer-Proof Fence

Description

有n个点，要画若干条封闭曲线将点围起来，使得每个点都在一个曲线中，且这个点到曲线上的任意一点的距离不小于m。所以，最优的曲线将会是一些圆角多边形。求栅栏的最小总长度。

Analysis

考虑将一个点集围起来的最短曲线，当$m=0$时就是这个点集的凸包。当$m>0$时可以直观地看出，将凸包沿与边垂直的方向向外移动m，转角处用圆弧连接起来是最优的。这些圆弧的圆心角之和就是凸包的外角和2π。所以，将一个点集围起来的曲线最小长度为凸包周长加$2\pi m$，凸包只有一条线段时要算两次。

由于将两个点分开是否更优难以确定，我们可以直接使用状压dp求解。先对于每个点集S，预处理出用一条曲线围的最小长度$c[S]$。用$f[S]$表示点集S用若干条曲线围的最小长度，则

$$f(S)=\min_{u\subseteq S,u\neq\emptyset}\{f[S\setminus u]+c[u]\}$$

按集合大小顺序dp，答案即为$f[U]$，其中U是所有点的集合。

时间复杂度：$O(2^n n+3^n)$

空间复杂度：$O(2^n)$

12.4 H. The Ministers' Major Mess

Description

有n个议案和m个大臣，每个大臣会对1到4个议案投票表示赞成或反对。你要决定每个议案是否通过，问能否使得每个大臣有大于一般的投票被满足。若能，输出每个议案一定要通过，一定要否决还是通不通过都可以。

Analysis

若一个大臣只投了1或2票，则他的每个投票都必须满足。若一个大臣投了3或4票，则他的投票最多只能有一个不被满足，也就是说若他的某一个投票不被满足，则另三个都必须满足。这是一个2-sat问题，对每个议案建两个点分别表示赞成通过和否决，按2-sat建图。对于必须满足的投票，新建两个点S和T，将S向这些投票的对应点连边，将这些投票的相反点向T 连边。若2-sat无解或S能走到T（也就是一个必须满足的投票不能满足）则无解。有解的话，S能走到的点要满足，另外如果一个点能走到他的相反点则它不能取，也就是要取它的相反点。不存在以上情况的议案通不通过都可以。判断一个点能否走到它的相反点可以暴力dfs。

时间复杂度：$O(nm)$

空间复杂度：$O(n+m)$

13 ACM/ICPC World Finals 2010

13.1 B. Barcodes

Description

一种条形码的编码方式是用不同的5位二进制数来表示$0 \sim 9$，'-'号和开始结束符号这12种字符。二进制位为1表示宽条，0表示窄条。条形黑白相间，两个字符之间用一个窄条隔开。对于一个要编码的字符串，要以一定规则在后面加上两个校验字符，再在首尾各加上一个开始结束符号。告诉你每个条形的宽度。宽条的宽度是窄条的两倍，但是宽度可能会有不超过5%的误差。输入顺序有可能是反的。判断条形码无法识别、可以识别但校验字符错误还是可以正确识别。若可以正确识别输出原字符串。

Analysis

先以最窄条形宽度的1.5倍为分界线将条形分为宽条和窄条。设窄条的标准宽度为x，则窄条宽度p要满足$0.95x \le p \le 1.05x$，宽条宽度q要满足$1.9x \le q \le 2.1x$。若x解集为空则无法识别。然后条形转化为字母，注意

判断是否有无法识别的5位二进制数，开始结束符号是否仅出现在首尾，两个字符之间的条形是否为窄条。若成功转化为字母，按题目要求验证校验字母是否正确。由于开始结束符号倒过来后就不是开始结束符号了，可以直接输出答案。若不能成功转化，将输入倒过来重新识别一遍，仍然不能成功转化才能确定无法识别。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

13.2 C. Tracking Bio-bots

Description

在$n \times m$的网格中有w个平行于x轴，宽度为1的墙。你只能向上或向右走。那么从某些不是墙的格子出发就不能走到右上角。问这样的格子有多少个。

Analysis

由于坐标范围较大，先进行离散化，将没有墙的端点的一段坐标并成一个大格，并记录每个大格的面积。然后用一次dp求出哪些大格能到达右上角。不能到达右上角又不是墙的大格的总面积即为答案。

时间复杂度：$O(w^2)$

空间复杂度：$O(w^2)$

13.3 D. Castles

Description

你要攻占一些城堡，连接这些城堡的道路构成了一棵树。每个城堡有三个参数a, b, c分别表示进攻该城堡需要的最少士兵数量，在进攻中会死去的士兵数量和必须留守的士兵数量。你可以带一些士兵从任意一个城堡开始沿道路移动，每到达一个城堡就要将它攻占，最后攻占所有城堡。同一条道路不能从同一个方向经过两次。问一开始最少需要多少士兵才能攻占所有城堡。

Analysis

在进攻中死去的士兵和留守的士兵没有什么区别，它们都是在这次进攻中失去的士兵。假设现在有一个攻占城堡的序列，需要的士兵依次为x_i，失去的士兵依次为y_i。若$x_i < y_i$则可将x_i改为y_i。一开始现有的士兵$t = 0$。每处理一个城堡，需要增加$\max\{x_i, t\} - t$的士兵，并失去y_i的士

兵。每次增加士兵之和即为总共需要的士兵，每次失去士兵之和即为总共失去的士兵。

对于以u为根的子树，先递归求出每个子树需要的士兵x_i和失去的士兵y_i。由于一条边只能来回一次，只能分别进攻每棵子树。进攻这些子树的顺序可以贪心确定。考虑两棵子树(x_1, y_1)和(x_2, y_2)，现有士兵为t，按不同顺序进攻它们失去的士兵总数都是$y_1 + y_2$，剩余的士兵数分别为$\max\{x_2 + y_1, x_1, t\} - y_1 - y_2$和$\max\{x_1 + y_2, x_2, t\} - y_1 - y_2$。若先访问第一棵子树更优则有$x_2 + y_1 <= x_1 + y_2$。按$x_i - y_i$递减的顺序进攻子树是最优的。我们先枚举根，每次进行树形dp即可求出总共需要的士兵数。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n)$

13.4 H. Rain

Description

给你一个平面图和每个点的高度表示地形。图中每个区域都是三角形。想象有足够多的水倒下来，一些水会流到边界外，另一些水会在图中形成湖。求会形成几个湖和每个湖的水面高度。两个只有一些深度为0的公共点的湖泊会被当作不同的湖泊。

Analysis

考虑能否求出所有点的水位。对于点u，假设与u相连的点的水位都已求出。如果其中有一个点的水位比u的高度低，则u上不积水，水位即为u的高度。如果与u相连的所有点水位都比u的高度高，则u的水位为相邻点水位的最小值。注意到如果将水位的初值都赋为无穷大，两种转移都是使水位减小的，已确定水位最小的点不会被更新，可以用类似dijkstra的方法，每次用水位最小的点去更新相邻点。而位于图的边界上的点是不能积水的，水位可确定为点的高度。

先求图的边界。找出纵坐标最大的点和从它出发极角最小的边，这条边一定在边界上。然后每次找到上一条边的反向边，并将上一个点出发的边极角排序，找到反向边的下一条边，这条边也一定在边界上。不断重复这个过程，回到开始的边时边界就找全了。将边界上点的水位设为点的高度，每次用没用过的水位最小的点更新相邻的点，所有点都用过后就求出了所有点的水位。找到所有水位与高度相等的点，这些点上没有积水。将这些点删去后，图中的每个连通块都表示一个湖，且湖中每个点的水位相同，再bfs一边即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

13.5 J. Sharing Chocolate

Description

有一个由$W \times L$个单位正方形构成的矩形。每次可以将一个矩形沿着行或列的分界线分成两个小矩形。给你n个小矩形的面积，问是否可能将原矩形恰好分为这样n个小矩形。

Analysis

我们使用状态压缩的记忆化搜索。将取哪些小矩形压缩成2^n种状态。用$f(i,j)$表示状态为i的小矩形是否能拼成一边长为j的矩形。设这些小矩形总面积为s，s必须是j的倍数。可以通过枚举i的子集，判断分成的两个子集是否都能拼成一边长为j或$\frac{s}{j}$的矩形，来进行转移。

时间复杂度：$O(3^nW)$

空间复杂度：$O(2^nW)$

14 ACM/ICPC World Finals 2011

14.1 A. To Add or to Multiply

Description

每次操作可以将原数加a或乘m，分别称为操作A和操作M。求一个最短的操作序列，使得区间$[p,q]$内的所有整数经过操作后都位于区间$[r,s]$内。若有多解输出字典序最小的操作序列，一段相同的操作用nA或nM表示，n是连续操作的次数。

Analysis

一开始最大数与最小数之差$d=q-p$。操作A不改变d，操作M会将d乘上m。由于必须满足$d \le s-r$，操作M的次数只有$O(\log s)$，可以枚举进行了几次操作M。若进行了k次操作M，则题中的要求等价于p经过操作后位于区间$[r, s-(q-p)m^k]$内。

现在我们要让x经过k次M操作和一些A操作后位于区间$[L,R]$内。假设第i次M和第$i+1$次M之间有p_i次A，第1次M之前有p_0次A，则操作后

$$x' = (\dots((x+p_0a)m+p_1a)m+\dots)m+p_ka = xm^k + a\sum_{i=0}^{k} p_{k-i}m^i$$

也就是要求一个位于$[\frac{L-xm^k}{a}, \frac{R-xm^k}{a}]$的“$p$进制数”，首先各位数之和最小，其次这个数最大。令$L' = \left\lceil \frac{L-xm^k}{a} \right\rceil$，$R' = \left\lfloor \frac{R-xm^k}{a} \right\rfloor$ $(L' \le R')$。如果它

们的第k位不同，则$[L', R']$存在一个数，它的后$k-1$位都是0。考虑第k位最小的这样的数，显然$[L', R']$中的每个数各位数之和都不小于它，且比它大的数各位数之和都大于它，它就是所求的“p进制数”。如果L'和R'的第k位相同，则第k位已经确定，将第k位去掉之后继续处理。这样就可以求出最优的“p进制数”，从而求出当前k下最优的操作序列。

时间复杂度：$O(\log^2 s)$

空间复杂度：$O(\log s)$

14.2 B. Affine Mess

Description

有三个整点。有人对它们进行了旋转、缩放和平移操作，其中旋转操作是先进行的。旋转操作是在原点为中心，边长为20的正方形的边上选一个整点，旋转坐标轴使x轴通过该点，并保持原先的单位长度不变。旋转之后，图上的所有点四舍五入跳转到了距离最近的整点。缩放操作是将横纵坐标分别乘以一个非零整数。平移量也是整数。现在知道操作前和操作结束后三个点的坐标，但不知道对应关系。判断是否有一个操作序列可以造成这样的变化。如果有，判断是否存在两个不等价的操作序列。两个操作序列等价当且仅当对于任意图形，它们会将该图形变换到同一图形。

Analysis

先枚举正方形边上的整点，模拟旋转操作。缩放和平移操作不管顺序如何都是对横纵坐标分别进行了一次线性变换，枚举点的对应关系，解一个线性方程组即可。只有旋转方向相反，缩放系数是相反数的两个操作序列才是等价的。如果找到了两个不同的方案方程组都有姐则存在两个不等价的操作序列。

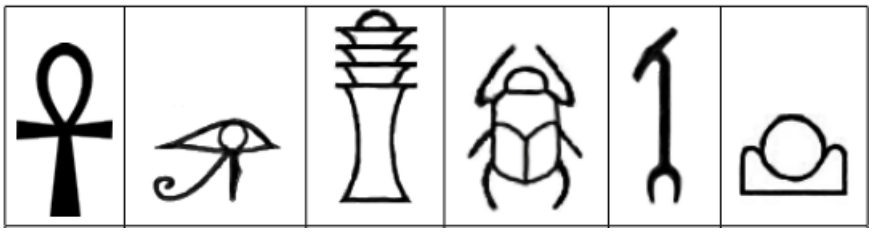
时间复杂度：$O(n^4 m)$（$n = 3$表示点数，$m = 20$表示方向数）

空间复杂度：$O(n)$

14.3 C. Ancient Messages

Description

给你一个用黑白矩阵表示的图像，里面每个黑色的连通块都表示一个象形符号。象形符号有以下六种，且拓扑等价的符号都算作同一种符号。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。黑色和白色都可认为是四连通的。你要识别出图中所有的象形符号，按字典序输出。



Analysis

由于象形符号可以进行二维的拓扑变换，我们只要考虑符号内部空洞的个数，通过观察可以发现6种符号中空洞的个数恰好都不相同，可以以此为依据进行识别。先从边界开始bfs求出白色背景部分。然后将背景以外的部分用一次bfs找出连通块，从而将每个符号分离出来。然后再进行一次bfs找出除背景外的白色连通块，这些连通块是某个符号内部的空洞，这样可以求出每个符号内部的空洞数，从而识别出每个符号。

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

14.4 E. Coffee Central

Description

在$dx \times dy$的网格中，有n个咖啡店。有q个询问，每个询问中，一个点能走到曼哈顿距离不超过m的咖啡店，求一个点最多能走到几个咖啡店，并求能走到咖啡店最多的点中先y最小，再x最小的一个。

Analysis

考虑一个询问。对于每个咖啡店，将与咖啡店距离不超过m的点权值加1，求权值最大的点即为答案。注意到与一个点距离不超过m的点构成了一个倾斜45°的正方形，将坐标轴旋转45°后二维差分，即可将修改优化到$O(1)$。修改完后再求前缀和还原成每个点的权值，直接找权值最大的点即可。

时间复杂度：$O(qdxdy + n)$

空间复杂度：$O(dxdy + n)$

14.5 F. Machine Works

Description

有n台机器，对于第i台机器，你可以在第D_i天以P_i的价格买入（如果钱足够），并在之后的任意一天以R_i的较低价格卖出。除买入和卖出的那

天，这台机器可以给你每天G_i的收益。你一开始有C块钱，同一时刻只能有一台机器，不过可以在同一天卖出机器后再买一台。在$D+1$天机器会被卖出。问这D天内的最大收益。

Analysis

显然，在要买新机器时才卖出是较优的。将初始状态看做售出价格为C 的第0台机器，终止状态看做限定第$D+1$天买入的第$n+1$台机器。将机器按买入时间排序，用$f[i]$表示买下第i台机器后剩余的最多钱数，则答案即为$f[n+1]$。状态转移方程如下：

$$f(i) = \max_{j<i, f[j]\geq 0} \{f[j] + G_j(D_i - D_j - 1) + R_j - P_i\}$$

注意到如果求出$f[i] < 0$说明买不起第i台机器，因此在转移中加入了$f[j] \geq 0$的限制，不过这可以通过给$f[i]$赋负无穷大解决。这样状态转移方程可以转化为：

$$f(i) = \max_{j<i}\{\underline{f[j] - G_j(D_j + 1) + R_j} + D_i G_j\} - P_i$$

划线部分只与j有关，令$Y[j] = f[j] - G_j(D_j + 1) + R_j$。对于决策$j < k$，$j$比$k$优等价于$\frac{Y[k]-Y[j]}{G_k-G_j} < -D_i$。用点$(G_i, Y[i])$表示决策，则可能的最优决策一定在点集的上凸壳上。由于询问的斜率$-D_i$是单调的，可以用cdq分治优化dp。每次先递归处理前半部分，然后求出前半部分的上凸壳，用前半部分更新后半部分，再递归处理后半部分。如果使用归并排序，每次的复杂度就是$O(n)$的。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

14.6 H. Mining Your Own Business

Description

给你一个联通无向图，你要给点黑白染色，要求删掉任意一个点后剩余的所有点都和黑点连通。问最少有几个黑点和黑点最少的前提下染色的方案数。

Analysis

将割点定义为删掉后使得图不连通的点（只与一个点相连的点不算）。若删掉的不是割点则只要有一个黑点即可，若删掉的是割点则分成的每个连通块内都必须有一个黑点。如果一个团中只有一个割点，那么把该割点删掉后这个块就孤立开来了，最少要有一个黑点，方案数为团中除

割点外的点数。如果一个团中有多个割点，删掉一个割点后它还和其它的团连通。由于团与团之间的相连关系不会形成环，这个连通块内必然有只有一个割点的团，由于那里已经有黑点了，不用再添加新的黑点。

先用tarjan算法求出割点，然后对于每个团求出其中的割点数。总的黑点数为割点数为1的团的个数。总的方案数为每个团的方案数相乘。值得注意的是如果整个图就是一个团，那么需要两个黑点，方案数为$\binom{n}{2}$。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

14.7 J. Pyramids

Description

你有n个石子，可以用$\sum_{i=1}^{k} i^2$个石子搭一个边长为k的高金字塔，或用$\sum_{i=1}^{k}(2i)^2$个石子搭一个边长为$2k$的矮金字塔，或用$\sum_{i=1}^{k}(2i-1)^2$个石子搭一个边长为$2k-1$的矮金字塔。要用完所有的石子造尽可能少的互不相同的金字塔，求造塔的方案，多解时取最大的金字塔最大的，还有多解取第二大的金字塔最大的，以此类推。

Analysis

可能的金字塔大约有320种。很容易把问题转化为背包，但由于要求方案，如果直接存转移就不能使用滚动数组，320*1000000的空间难以接受。好在最少的塔数不会很多，打表可得最多为6。因此我们可以将方案压位，用9位表示一座塔，就可以用long long存下。对于塔数超过6的转移可以直接忽略。为了保证方案的字典序最大，按石子数从小到大处理金字塔，在塔数相同时选择新的方案，这样每次都保证最大的金字塔是最大的。

时间复杂度：$O(n^{\frac{4}{3}})$

空间复杂度：$O(n)$

14.8 K. Trash Removal

Description

给你一个多边形，要把它用两条平行线夹在中间，求平行线的距离最小是多少。

Analysis

先求出多边形的凸包，这两条平行线显然也会把凸包夹在中间。为了使距离最小，一定有一条线经过凸包的一条边，另一条线经过离这条边最

远的点。使用旋转卡壳算法找到离每条边最远的点，取距离的最小值即为答案。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

15 ACM/ICPC World Finals 2012

15.1 B. Curvy Little Bottles

Description

有一个瓶子是由一条从$x = Xlow$到$x = Xhigh$的多项式曲线绕x轴旋转一周构成的。求瓶子的体积。还要从$Xlow$开始给瓶子标刻度，已知相邻两条刻度的体积差，求刻度应该标在横坐标为多少的地方。若刻度过多只需输出前8个。

Analysis

$Xlow$到$x0$的体积为$\int_{Xlow}^{x0} \pi(\sum_k a_k x^k)^2 dx$。由于多项式次数很低，可以直接求出原多项式的平方的原函数，就可以对任意坐标快速求出体积。然后由于精度要求较低，二分刻度的横坐标即可。

时间复杂度：$O(n^2 + n \log \frac{ans}{eps})$

空间复杂度：$O(n)$

15.2 C. Bus Tour

Description

给你一个n个点的带权无向图，你要从0号点出发，经过$1 \sim n-2$的所有点到达$n-1$。设这$n-2$个点中首先经过的$\lfloor \frac{n-2}{2} \rfloor$个点是集合$S$。再经过一次$1 \sim n-2$的所有点到达0，并且首先经过的$\lfloor \frac{n-2}{2} \rfloor$个点也要是集合$S$。求最短路程。

Analysis

由于数据规模较小，可以用状压dp解决。先用floyd算法求出任意两点之间的最短路。用$f(X,u)$表示从0出发，经过的点用二进制表示为X，最后到u的最短路程，$g(X,u)$表示从$n-1$出发，经过点用二进制表示为X，最后到u的最短路程。通过枚举最后一步求出f和g的所有值。枚举先经过的$\lfloor \frac{n-2}{2} \rfloor$个点表示为$X$，后经过的$\lceil \frac{n-2}{2} \rceil$个点表示为$Y$。由于无向图路径反过来也是一样的，答案可以表示为$(f(X,u)+dis(u,v)+g(Y,v))+(g(X,u')+$

$dis(u',v')+f(Y,v'))(u,u' \in X,\ v,v' \in Y)$。两部分可以分开考虑，先枚举$u,v$再枚举$u',v'$即可。

时间复杂度：$O(2^n n^2)$

空间复杂度：$O(2^n n)$

15.3 D. Fibonacci Words

Description

定义斐波那契字符串如下：$F(0)=0,\ F(1)=1,\ F(n)=F(n-1)+F(n-2)(n \geq 2)$，其中+是字符串的连接。求长度为$L$的字符串$p$在$F(n)$中出现了多少次。

Analysis

先暴力推出前几个斐波那契字符串，直到$F(k)$和$F(k-1)$都比p长为止。若$n<k-1$则答案为0。否则从$k-1$开始dp，求出$F(n)$中有几个$F(k)$和几个$F(k-1)$，以及前面和后面分别是$F(k)$和$F(k-1)$的四种连接方式各出现了几次。

将$F(n)$看做一段段$F(k)$和$F(k-1)$构成的，由于它们比p长，$F(n)$中的p要么全部位于一段$F(k)$或$F(k-1)$里面，要么位于连续的两段$F(k)$或$F(k-1)$里面。第一种情况的次数用KMP求出$F(k)$和$F(k-1)$分别出现了几次p，再乘以$F(n)$中它们的出现次数即可求出。第二种情况，以$F(k)$连接$F(k-1)$，求经过中间的p有几个为例。先以$F(k)$为模式串，对p反向做KMP，再以$F(k-1)$为模式串，对p正向做KMP。这样求出p的哪些前缀能与$F(k)$的后缀匹配，哪些后缀能与$F(k-1)$的前缀匹配，计数排序后找有几对加起来等于L的就是答案。其它连接也可以用同样的方法求出，再乘以$F(n)$中这种连接的出现次数即可求出总的出现次数。

时间复杂度：$O(n+L)$

空间复杂度：$O(n+L)$

15.4 L. Takeover Wars

Description

A有n个数，B有m个数，他们轮流操作，A先操作。每次可以将自己的两个数并在一起变成它们的和，或者去掉对方的一个数，要去掉的数必须比自己的一个数小。能操作必须操作。没有数的人就输了。两人的数不会出现相等的情况。两人都使用最优策略，问谁能获胜。

Analysis

如果进行去数操作，显然是去掉能去掉的最大的数最优。如果进行合并操作，可以证明合并自己最大的两个数最优。如果自己的最大数比对方的最大数小，此时去数会使得自己的最大数也被去掉，局势更为不利，所以必须进行合并。

我们先枚举A第一步是合并还是去数（如果可以）。那么接下来B会怎么做呢？如果B的最大数比A大那么B就赢了，因为此时A的最大值一定是之前最大的两个数合并得到的，去掉它之后A再合并出的数都无法撼动B的最大数。如果B的最大数比A小则B必须合并。考虑接下来A的情况，其实与刚才B的情况非常相似，因为B的最大数也是合并得到的。如果A的最大数比B大A就赢了，否则A必须合并。这样双方不断合并直到自己的最大数比对方大，之后的操作都可以确定。

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$

16 ACM/ICPC World Finals 2013

16.1 A. Self-Assembly

Description

有n种正方形，四条边上都有2个字符的标记。对于任意大写字母X，标有$X+$和$X-$的两条边可以拼在一起。特别地，标有00的边不能和任何边拼在一起。每种正方形都有无限个，可以任意旋转和翻转，问能否拼成一个无限的结构。

Analysis

我们以正方形为点建图，能拼在一起的正方形之间连边。如果图中没有环，即使将所有能拼的正方形都拼起来也不可能拼成无限的结构。如果图中有环，由于正方形可以任意旋转和翻转，一定可以用环上的正方形无限拼接下去而不使其它边重合。

但是正方形数较多，这样做复杂度不能接受。注意到一个正方形相当于将四个边的标记连了起来，原图中的环等价于标记的环，这个环中的边是正方形带来的边和同种字母带来的边交替出现的。所以我们可以建立一个二分图，每种标记在左右两边各有一个对应点。同一个正方形内的标记从左边的点向右边的点连边，形如$X+$和$X-$的点从右边向左边连边。然后用floyd算法找环即可。由于点数只有$m=104$复杂度就可以接受了。

时间复杂度：$O(m^3)$

空间复杂度：$O(m^2)$

16.2 B. Hey, Better Bettor

Description

有一个赌局，赢了能获得1元，输了会失去1元。你的胜率是$p\%(0 \leq p < 50)$，如果赌完之后你的资金减少了，赌场会将$x\%$的损失还给你。x和p最多包含两位小数。赌局没有时间限制和金钱限制，但是你只能赎回一次。求最优策略下的最大期望收益。

Analysis

由于没有金钱限制，假设一开始钱数为0。游戏中的局面只与当前钱数有关，与之前的操作无关。所以，最优策略总是这样的形式：如果获利到R或亏损到$L(L \leq 0 \leq R)$则赎回。若$p = 0$则一局都不玩最优，下面假设$p \neq 0$。

用$f(i)$表示在有i元钱的情况下，因获利到R而不是亏损到L赎回的概率。有

$$
\begin{aligned}
&f(L) = 0 \\
&f(R) = 1 \\
&f(x) = p\%f(x+1) + (1-p\%)f(x-1)(L < x < R)
\end{aligned}
$$

转化为递推式

$$f(x) = \frac{1}{p\%}f(x-1) - \frac{1-p\%}{p\%}f(x-2)$$

用特征方程解得

$$f(x) = A + B\left(\frac{1-p\%}{p\%}\right)^x$$

将$f(L) = 0,\ f(R) = 1$代入，令$q = \frac{1-p\%}{p\%} \in (1, +\infty)$，得到

$$f(x) = \frac{q^x - q^L}{q^R - q^L}$$

所以期望收益

$$E(L, R) = f(0) = \frac{1 - q^L}{q^R - q^L}$$

对L，R分别求偏导，得到的偏导数都是单调的。预处理q的幂，三分套三分即可求出最优解。实际上q的范围是$[\frac{5001}{4999}, 9999]$，为在$q$较小时保证精度，在$q$较大时防止溢出，要根据$q$的大小调整三分上限。

时间复杂度：$O(\log^2 M)$（M表示三分上限）

空间复杂度：$O(M)$

16.3 D. Factors

Description

用$f(k)$表示将k分解质因数后所有质因子的可重排列数，相同的质因子算多个。求满足$f(k)=n$的最小的k。n和k都不超过2^{63}。

Analysis

设$k=\prod_{i\le l}p_i^{a_i}$，则

$$f(k)=\binom{a_1+a_2+\cdots+a_l}{a_1,a_2,\ldots,a_l}=\frac{(a_1+a_2+\cdots+a_l)!}{a_1!a_2!\ldots a_l!}$$

这和p_i以及排列方式都无关，所以确定所有a_i后，将它们从大到小分配给最小的几个质数是最优的。由于n和k较小，可以直接枚举所有a_i。枚举过程中要保证a_i递减，且$\prod p_i^{a_i}\le 2^{63}$（这里$p_i$表示第$i$小的质数）。通过先用实数类型估计大小，并不断约分的方法计算出的$f(k)$，若$f(k)$也不超过2^{63}则将这个k存入表中。事实上符合条件的k不到40000个。将表按$f(k)$排序，询问时只要在表中二分查找即可。注意要事先把1排除。

时间复杂度：$O(m\log m)$（m表示合法的k的个数）

空间复杂度：$O(m)$

16.4 E. Harvard

Description

要完成一段n个语句的程序，语句有三种，一种是访问某个变量，另两种是循环的开始和结束标志，表示将中间这段语句循环若干次。变量最多有13个，要分到编号为$0\sim b-1$的b个内存库中，每个内存库最多k个变量。你可以进行进行三种操作：

1. 访问0号内存库中的某个变量；
2. 改变指示器的值；
3. 访问编号为指示器的值的内存库中的某个变量。

一开始指示器的值未定义。已知程序，求最少操作数。

Analysis

朴素的做法是先枚举每个变量位于哪个内存库中。然后，考虑将所有循环都展开的程序，显然等到要访问的内存库不是指示器也不是0的时候再

改变指示器的值是最优的。于是可以递归处理一段程序，内部的循环处理一次之后，第二次开始指示器的初始值都是一样的，可以一起处理。这样可以在$O(n)$的时间内求出操作数。但是枚举量不能接受。

注意到将一个变量从其它内存库移到0号内存库一定不会变差。我们先枚举哪些变量放入0号内存库，要求将0号内存库放满。然后可以用与上面类似的方法，求出每两个非0号内存库变量相邻的次数，以及访问操作的次数。然后再搜索非0号内存库变量位于哪个内存库。由于当且仅当相邻非0号内存库变量位于不同内存库时才需要改变指示器的值，可以快速求出总操作数。搜索过程中还可以加入最优性剪枝，以及针对内存库的对称性的剪枝。这样搜索量就可以接受了。

时间复杂度：$O(2^m n + m!)$（$m = 13$表示变量数）

空间复杂度：$O(n + m^2)$

16.5 F. Low Power

Description

有$2nk$个数，要将它们分成n组，每组$2k$个数。每组再分成两小组，每小组k个数，得到两小组的最小值之差。求n个差的最大值最小是多少。

Analysis

先将$2nk$个数排序，设为$a_1, a_2, \ldots, a_{2nk}$。首先注意到答案只与每小组的最小值有关。假设我们已经找出$2n$个最小值，要把它们分成n组作差，为使差的最大值最小，应该怎么分呢？不难发现，排序后将相邻两个数分在一起是最优的。不然的话，必然存在$a \le b \le c \le d$分成了a, c和b, d或a, d和b, c，调整成a, b和c, d显然不会变差。

那么，这$2n$个最小值能不能随便找呢？由于每个最小值后面都要放$k-1$个数，第i个最小值必须在前$(i-1)k+1$个数内。另外，如果a_i和$a_j (j > i+1)$是同一组的两个最小值，由于它们之间不会有其它最小值，调整成a_i和a_{i+1}显然不会变差，而且一定合法。

首先二分答案。假设当前差的最大值为x，则从a_1到a_{2nk}进行一次扫描，如果$a_{i+1} - a_i \le x$，就将它们作为同一组的两个最小值。如果当前只找到t个最小值但已经扫到第$tk+2$个数则构造失败，答案大于x。如果找到了n个最小值则构造成功，答案不大于x。

时间复杂度：$O(nk \log nk)$

空间复杂度：$O(nk)$

16.6 H. Матрёшка

Description

有一列数集，一开始都只有一个数。每次可以将相邻两个不含有相同元素的数集合并，若这两个数集的最小值中较大的是x，则这次合并的代价为这两个数集中不小于x的数的个数。要使所有数集都包含从1开始的连续自然数，求最小代价。

Analysis

进行区间DP，用$f(i,\ j)$表示将第i个数到第j个数合并的最小代价（如果这些数都不相同）。枚举最后一次合并，假设是将第i到k个数构成的数集和第$k+1$到j个数构成的数集合并，两个数集的最小值分别是x和y，$x < y$。则这次合并的代价就是x所在的一段中大于y的数的个数加上y所在区间的个数。如果事先用$O(n^2)$的时间预处理出每个区间的最小值，以及每个前缀中比每个数大的数的个数，就可以在$O(1)$的时间内算出一次合并的代价，从而将转移优化到$O(n)$。

然后再进行一次DP，用$g(i)$表示把前i个数合并成一些形如$\{1, 2, \ldots, m\}$的数集的代价，枚举最后一个数集从第几个数开始进行转移。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

16.7 I. Pirate Chest

Description

已知矩形池塘每个点的水深，用一个矩阵给出。要在矩阵中选择一个尺寸不超过$a \times b$（或不超过$b \times a$）的子矩形作为底面，放下一个箱子，箱子底面的深度是子矩形中的最小值，高度是任意整数。但是，箱子表面必须严格低于水面，且箱子排开的水会上升到池塘表面。求箱子的最大体积。

Analysis

设池塘总面积为S，选取的子矩形尺寸为$w \times l$，其中最小深度为d。则箱子高度h需满足$Sh < Sd + wlh$，由于h为整数且越大越好，得到$h = \lfloor \frac{Sd-1}{S-wl} \rfloor$。于是

$$V = \left\lfloor \frac{Sd-1}{S-wl} \right\rfloor wl$$

从上式可以看出，如果w和d一定，V随着l的增大而增大。为使V最大，需要在符合$a \times b$限制的范围内找到最大的l。

先枚举子矩形的上下边界u和$u+w$，求出每一列在上下边界之间的数的最小值，这可以利用边界为u和$u+w-1$时的结果在$O(n)$的时间内做到。设第i列的最小值是x_i，问题转化为选择一个区间。枚举区间的最小值x_i，由于需要使区间长度尽可能大，只需找到x_i左侧和右侧第一个比它小的数即可。这些结果可以利用单调栈在$O(n)$的时间求出。具体做法是：将x_1到x_n依次压入栈，若压入时比栈顶元素小则先弹栈，这样保证栈中元素单调不降。当一个元素被弹栈时，当前要加的数即为右侧比它小的第一个数，栈中前一个数即为左侧比它小的第一个数。这样就在枚举$w,\ d$ 的情况下求出了最大的l，从而可以求出V的最大值。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

16.8 J. Pollution Solution

Description

给你一个在y轴上方的简单多边形，和一个圆心为原点，半径为r的圆在y轴上方部分和y轴构成的半圆，求它们的交的面积。没有顶点会位于圆弧上。

Analysis

对于多边形的每一条线段，计算它的顶点和原点相连构成的三角形和半圆的交的有向面积，所有有向面积的和即为总面积。

有向面积的符号可以用叉积确定，下面只考虑大小。若线段的两个端点都在圆内面积即为三角形的面积。若线段有一个端点在园内，一个端点在圆外，求出线段与圆的交点，相交部分可分为一个三角形和一个扇形。若两个端点都在园内，也要求出线段与圆的交点。若没有交点，相交部分是一个扇形。若有2个交点，相交部分可分为两个扇形和一个三角形。求交点可以直接写出一般式解方程，再判断解是否在线段范围内。

时间复杂度：$O(n)$

空间复杂度：$O(n)$