

国家集训队第一次作业第二部分解题报告

(共 108 题)

广东省中山市中山纪念中学 孙伟峻

目录.....	1
1. 1998 D Page Selection by Keyword Matching.....	4
2. 1998 E Petri Net Simulation.....	5
3. 1998 G Spatial Structures.....	6
4. 1999 A Bee Breeding.....	8
5. 1999 C A Dicey Problem.....	9
6. 1999 D The Fortified Forest.....	10
7. 1999 E Trade on Verwegistan.....	11
8. 1999 H flooded!.....	12
9. 2000 A Abbott's Revenge.....	13
10. 2000 B According to Bartjens.....	14
11. 2000 C Cutting Chains.....	15
12. 2000 E Internet Bandwidth.....	16
13. 2000 F Page Hopping.....	17
14. 2001 A Airport Configuration.....	18
15. 2001 B Say Cheese.....	19
16. 2001 F A Major Problem.....	20
17. 2002 A Ballons in a Box.....	21
18. 2002 C Crossing the Desert.....	22
19. 2002 E Island Hopping.....	23
20. 2002 G Partitions.....	24
21. 2002 H Silly Sort.....	25
22. 2003 A Building Bridges.....	26
23. 2003 B Light Bulbs.....	27
24. 2003 F Combining Images.....	28
25. 2003 G A Linking Loader.....	29
26. 2003 H A Spy in the Metro.....	30
27. 2003 I The Solar System.....	31
28. 2003 J Toll.....	33
29. 2004 E Intersecting Dates.....	34
30. 2004 G Navigation.....	35
31. 2004 H Tree-Lined Streets.....	36
32. 2004 I Suspense.....	37
33. 2005 B Simplified GSM Network.....	39
34. 2005 C The Traveling Judges Problem.....	40
35. 2005 E Lots of Sunlight.....	41
36. 2005 F Crossing Streets.....	42
37. 2005 G Tiling the Plane.....	43

38.	2005 H The Great Wall Game.....	44
39.	2005 I Workshops.....	45
40.	2005 J Zones.....	46
41.	2006 A Low Cost Air Travel.....	47
42.	2006 B Remember the A La Mode!.....	48
43.	2006 D BipartiteNumbers.....	49
44.	2006 E Bit Compressor.....	50
45.	2006 G Pilgrimage.....	51
46.	2006 I Degrees of Separation.....	52
47.	2006 J Routing.....	53
48.	2007 A Consanguine Calculations.....	54
49.	2007 E Collecting Luggage.....	56
50.	2007 F Marble Game.....	57
51.	2007 G Network.....	59
52.	2007 I Water Tanks.....	60
53.	2007 J Tunnels.....	61
54.	2008 A Air Conditioning Machinery.....	62
55.	2008 B Always an Integer.....	63
56.	2008 D The Hare and the Hounds.....	64
57.	2008 E Huffman Codes.....	65
58.	2008 F Glenbow Museum.....	66
59.	2008 G Net Loss.....	67
60.	2008 H Painter.....	68
61.	2008 I Password Suspects.....	69
62.	2008 J The Sky is the Limit.....	70
63.	2008 K Steam Roller.....	71
64.	2009 A A Careful Approach.....	72
65.	2009 B My Bad.....	73
66.	2009 D Conduit Packing.....	74
67.	2009 E Fare and Balanced.....	75
68.	2009 F Deer-Proof Fence.....	76
69.	2009 G House of Cards.....	77
70.	2009 H The Ministers' Major Mess.....	79
71.	2009 I Struts and Springs.....	80
72.	2009 J Subway Timing.....	81
73.	2009 K Suffix-Replacement Grammars.....	82
74.	2010 B Barcodes.....	83
75.	2010 C Tracking Bio-bots.....	85
76.	2010 D Castles.....	86
77.	2010 F Contour Mapping.....	87
78.	2010 G The Islands.....	88
79.	2010H Rain.....	89
80.	2010 I Robots on Ice.....	91
81.	2010 J Sharing Chocolate.....	92

82.	2010K Paperweight.....	93
83.	2011 A To Add or to Multiply.....	94
84.	2011 C Ancient Messages.....	95
85.	2011 E Coffee Central.....	96
86.	2011 F Machine Works.....	97
87.	2011 H Mining Your Own Business.....	98
88.	2011 I Mummy Madness.....	99
89.	2011 J Pyramids.....	100
90.	2011 K Trash Removal.....	101
91.	2012 A Asteroid Rangers.....	102
92.	2012 B Curvy Little Bottles.....	103
93.	2012 C Bus Tour.....	104
94.	2012 D Fibonacci Words.....	105
95.	2012 E infiltration.....	106
96.	2012 I A Safe Bet.....	107
97.	2012 K Stacking Plates.....	108
98.	2012 L Takeover Wars.....	109
99.	2013 A Self-Assembly.....	110
100.	2013 B Hey, Better Bettor.....	111
101.	2013 C Surely You Congest.....	112
102.	2013 D Factors.....	113
103.	2013 E Harvard.....	114
104.	2013 F Low Power.....	115
105.	2013 H М а т р ё ш к а	116
106.	2013 I Pirate Chest.....	117
107.	2013 J Pollution Solution.....	118
108.	2013 K Up a Tree.....	119

【题目编号】

1998 D Page Selection by Keyword Matching

【题目名称】

Page Selection by Keyword Matching

【题目大意】

给出若干组字符串，每组字符串开头之前是一个大写字符 P 或 Q，代表这一组字符串是网页或询问。接下来有不超过 8 个字符串，由 26 个字母组成。两组字符串的匹配程度定义如下：若第一组字符串的第 i 个字符串等于第二组字符串的第 j 个字符串，它们的匹配程度增加 $(8-i+1)*(8-j+1)$ 。现在对于每组询问的字符串，要求在它之前出现的网页中，和它匹配程度最大的五个。如果两个网页和它匹配程度相当，保留出现较早的一个。

输入以大写字符 E 结束。所有字符串长度不差过 20，网页数不超过 25，每组字符串包含的字符串数量不超过 8 个。

【题目关键字】

模拟，枚举

【算法讨论】

因为题目数据范围较小，所以直接按题意模拟即可。为了方便查询一个字符串是否在某个字符串中出现，可以用一个 map。

【时空复杂度】

时间复杂度： $O(n\log n)$, n 为读入字符串总数。

空间复杂度： $O(n)$

【题目编号】

1998 E Petri Net Simulation

【题目名称】

Petri Net Simulation

【题目大意】

给出 NP 个库所，和 NT 个事件。开始的时候每个库所有一些令牌。每个事件在一些库所减少一定数量的令牌和在另一些库所增加一定数量的令牌。一个事件会发生当且仅当需要减少令牌的库所有足够数量的令牌。每个单位时间最多发生一个事件。如果有同一时刻多个事件有多个事件能够发生，那么发生输入中最早给出的事件。问 NF 个事件后每个库所的令牌数。

NP,NT \leq 100,NF \leq 1000, 所有事件涉及的库所总数 \leq 20000

【题目关键字】

模拟

【算法讨论】

枚举时间，然后按照题意找到这个时间应该发生的事件并处理令牌数量的变化即可。

【时空复杂度】

时间复杂度: $O(NF*NP*NT)$

空间复杂度: $O(NF*NT)$

【题目编号】

1998 G Spatial Structures

【题目名称】

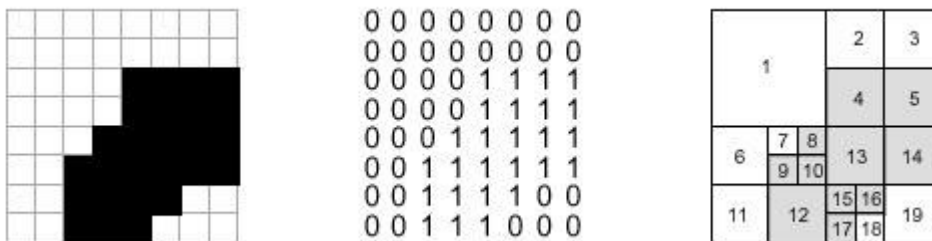
Spatial Structures

【题目大意】

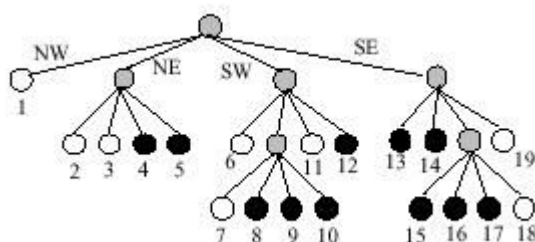
在计算机绘图，图像处理和 GIS（geographic information system，地理信息系统）等领域都用到了一种叫做四分树的数据结构。四分树能够高效地表示一个区域内的或一个块的信息，并且能够为各种操作提供高效的算法，例如图片的合并与分割。

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（都是黑色或白色），则这个象限内不再继续分割。同时包含黑色和白色像素点的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。一个象限完全有可能只包含一个像素点。

举个例子，用 0 代表白色，1 代表黑色，下图中左边的区块被表示成中间的 01 矩阵。如右图所示，矩阵被分割成若干子象限，其中灰色区域表示只包含黑色像素点的象限。



一棵四分树是由图片的块状结构构建起来的。树的根节点表示整个存储像素点的数组。每个非叶子节点有四个子节点，对应这个这个节点表示的象限的四个子象限。叶子节点表示一个只包含同种颜色的像素点的区域，因此没有被继续分割。举个例子，上图中右边的块状结构被表示成下图中的四分树。



如果一个叶子节点只包含白色像素点，则这个叶子节点是白色的，否则是黑色的。树中的每个节点都有一个标号，对应第一张图中右边的图示。一个非叶子节点的四个子节点从左到右依次表示西北、东北、西南、东南四个象限（或者说左上、右上、左下、右下）。

一棵四分树可以通过一个表示根到所有黑色叶子节点的路径的数列来表示。每条路径用一个五进制数表示，若将每个节点的分支从左到右标号为 1,2,3,4，则这个五进制数最低位表示路径中根节点的分支的标号，以此类推。举个例子，根到 4 号点经过的分支为 NE、SW，这条路径表示为 32₅（五进制）或 17₁₀（十进制）；根到 12 号点的路径经过 SW、SE，表示为 43₅或 23₁₀。整棵树可以用下面这个数列来表示（十进制下）：

9 14 17 22 23 44 63 69 88 94 113

请写一个程序来实现将图片转成根到叶子路径序列，或将根到叶子路径序列转成图片。

给出的图片边长 n 均为 2 的幂，且 $|n| \leq 64$ 。

对于图片转成根到叶子路径序列的询问， $n > 0$ ，随后给出 n 行字符串，每行 n 个字符 0 或 1，表示每个像素是黑色或白色。

对于根到叶子路径序列转成图片的询问， $n < 0$ ，随后给出若干个整数，即根到每个叶子的路径序列转化成的五进制数的十进制形式。

【题目关键字】

模拟

【算法讨论】

按题意直接模拟即可。

对于图片转成序列的询问，预处理图片上左上角到 (i, j) 的子矩形和，即可 $O(1)$ 计算每个子矩形内的黑色像素个数。然后从整张图形开始，递归处理左上，右上，左下，右下四个区域，通过黑色像素个数即可 $O(1)$ 判断整个子区域全黑或全白。

对于序列转成图片的询问，同样递归处理即可。从整个矩形开始，根据给出的代表路径的整数的末尾递归到左上，右上，左下，右下四个区域，然后当前代表的整数除以 5。如果该整数为 0，那么将整片区域染黑后退出递归即可。

两种询问处理的最坏时间复杂度均为 $O(n^2 \log n)$ 。因此总的时间复杂度为 $O(n^2 \log n)$ 。

【时空复杂度】

时间复杂度： $O(n^2 \log n)$

空间复杂度： $O(n^2)$

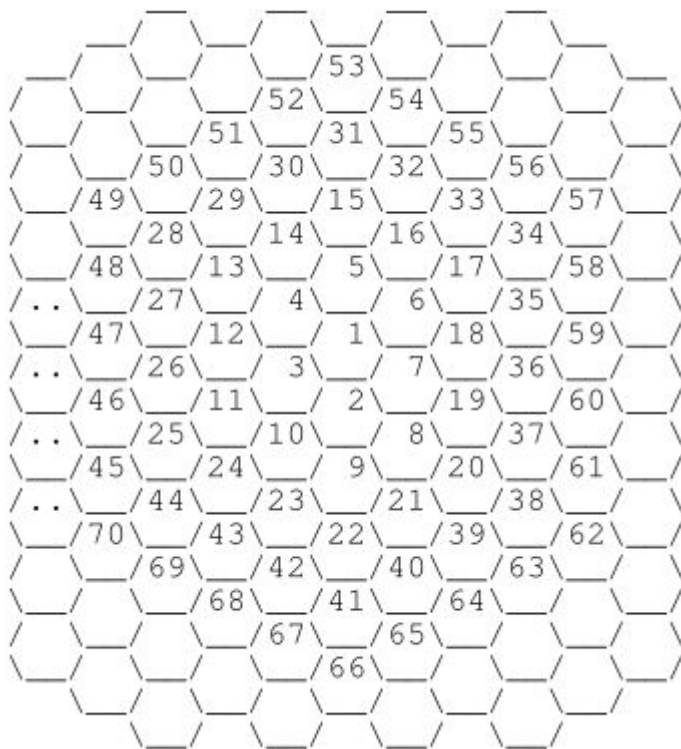
【题目编号】

1999 A Bee Breeding

【题目名称】

Bee Breeding

【题目大意】



给出如图所示的一个蜂巢，从点 1 开始给每个点编号，求图中某两个点 a,b 之间的距离。a,b 的距离定义为从 a 出发最少经过几个六边形能到达 b。多组数据,a,b<=10000,数据组数<=100

【题目关键字】

模拟

【算法讨论】

首先，记由 1 到 2 的方向记为 2，1 到 3 的方向记为 3……1 到 7 的方向记为 7，他们分别是：(0,1),(-1,1),(-1,0),(0,-1),(1,-1),(1,0);这些规律不仅对于 1 的周围六个方向有效，对于所有的点都是有效的。然后记 1 所在为圈 1，2..7 为圈 1，8..19 为圈 2……，所以，很容易可以得到第 n 圈有蜂窝 6n 个(n>0)，对于这个等差数列求和， $S[1..n]=3n^2+3n$ ，包括第 0 圈的 1，则 $S[0..n]=3n^2+3n+1$ 。

然后对于一个节点 a，解方程 $3n^2+3n+1=a$ ，解出来 $n=[\sqrt{12a-3}]/6$ 如果 n 为整数，则圈数 $p=n$ ，否则 $p=\text{trunc}(n)+1$ ，又可以通过公式 $t=a-3*\text{sqr}(p)+3*p-1$ 求出 t (a 是第 n 圈的第 t 个)。然后根据 p,t，就可以方便的算出 a,b 两点的坐标。接下来分别算出 a,b 的 x 坐标差 dx 和 y 坐标差 dy，若 dx,dy 符号相同，那么答案为 dx,dy 的绝对值之和，否则答案为 dx,dy 的绝对值的较大值。

【时空复杂度】

时间复杂度：O(T)，T 为数据组数。

空间复杂度：O(1)

【题目编号】

1999 C A Dicey Problem

【题目名称】

A Dicey Problem

【题目大意】

给出一个 $R \times C$ 大小的棋盘，每个格子上有一个数字或星星。给出一个位于某个格子上的骰子，骰子个每个面有一个数字。给出骰子开始的状态。每次骰子可以滚到相邻的一个有和它顶面上的数字相同数字的格子上，或是滚到一个有星星的格子上，要求找一条骰子从初始位置出发后又回到起点的路径。 $R, C \leq 10$

【题目关键字】

搜索

【算法讨论】

因为骰子的不同状态只有 24 种。令 $f[i][j][k]$ 表示骰子滚到 (i, j) 这个格子，骰子状态为 c 的路径是否存在。搜索所有状态并记录路径，然后判断是否存在环即可。

【时空复杂度】

时间复杂度: $O(R \times C)$

空间复杂度: $O(R \times C)$

【题目编号】

1999 D The Fortified Forest

【题目名称】

The Fortified Forest

【题目大意】

平面上有 n 棵树，给出每个树的坐标，价值，和砍伐这棵树能做成的栅栏长度。假设树的直径为 0，要求砍倒其中一些树木，做成栅栏围住剩下的树。要求砍伐树木的总价值最小，如果有多种最小价值的方案，输出其中砍伐树木数量最小的。输出方案，数据保证解唯一。

$n \leq 15$ ，多组数据。

【题目关键字】

枚举，凸包

【算法讨论】

因为 $n \leq 15$ ，所以可以枚举所有砍伐树木的方案，然后对剩下的树木求凸包，即可求出围住剩下的树所需的栅栏长度。如果被砍伐的树木能提供的栅栏长度不小于这个长度即为合法解。求出所有合法方案后取最优即可。

【时空复杂度】

时间复杂度： $O(2^n \cdot n \log n)$

空间复杂度： $O(n)$

【题目编号】

1999 E Trade on Verwegistan

【题目名称】

Trade on Verwegistan

【题目大意】

给出 w 堆物品，每堆物品有 b 个堆成 1 列。每购买一个物品可以获得 10 元的利润，但每个物品有不同的价格。同一堆物品只有先购买堆在上面的所有物品才能购买下面的物品。例如要购买某一堆的第五个物品，就必须先购买它上面 4 个物品。

要求最大获利，以及要获得最大获利需要购买的物品数量。如果有多个解，那么输出最小的十个不同数量，不足十个则全部输出。多组数据，数据组数 $\leq 10, 1 \leq w \leq 50, 0 \leq b \leq 20$ 。

【题目关键字】

贪心，排序

【算法讨论】

很明显每堆物品互不影响。因此我们可以分开每堆物品能获得的最大利润，以及获得最大利润在该堆物品中需要购买哪几种数量的物品。

然后是统计答案。最大利润只要相加即可。购买数量方案的话，每次合并两组物品需要购买数量，合并方法为分别枚举在两堆中购买的数量，然后相加。合并后排序并保留最小的 10 个即可。时间复杂度为最坏为 $O(T * w * (100 \log 100))$ ，可以通过本题。

【时空复杂度】

时间复杂度: $O(T * w * (b^2 \log b^2))$

空间复杂度: $O(b^2)$

【题目编号】

1999 H flooded!

【题目名称】

flooded!

【题目大意】

有一块由 $N \times M$ 个正方形构成的区域，每一块正方形的边长为 10m，每一块都有自己的海拔。在较高的区域中的积水（即使完全被更高的区域所包围）能完全排放到较低的区域中，并且水不会被地面吸收。给出该区域的积水量的总体积，问积水的高度和该地段完全被淹没的区域的百分比（指该地段中高度严格低于积水高度的区域的百分比）。

多组数据， $n, m \leq 30$

【题目关键字】

排序，贪心

【算法讨论】

因为水一定会流到海拔较低的地区，所以很明显应该从低到高枚举哪些格子被水面覆盖。

首先将所有区域按海拔高度排序，从低到高枚举哪些格子被水面覆盖，把水均匀的铺在当前的水面上，并不断更新当前水面面积和剩余水量，若剩余水量为 0，输出当前水面高度和水面覆盖率。

枚举被水覆盖的格子 and 计算水面高度的总时间复杂度为 $O(n \times m)$ 。因此算法的主要复杂度在于排序的 $O(nm \log(n \times m))$ 。对于本题 $n, m \leq 30$ 可以轻松通过本题。

【时空复杂度】

时间复杂度： $O(nm \log(n \times m))$

空间复杂度： $O(nm)$

【题目编号】

2000 A Abbott's Revenge

【题目名称】

Abbott's Revenge

【题目大意】

给出一个大小不超过 $9*9$ 的迷宫，给出起点和终点。对于每个点，给出从每个不同的进入方向有一个限制，接下来往前、往左、往右三个方向中的可以选择哪些继续沿着选择的方向前进，问从起点到终点的最短路，输出路径。多组数据，数据组数 ≤ 1000

【题目关键字】

BFS

【算法讨论】

令 $vis[x][y][d]$ 表示从方向 d 进入路口 (x,y) 的状态是否存在， $fa[x][y][d]$ 记录这个节点的父节点。从起点开始 BFS，BFS 中第一条到达终点的路径即为答案。最后根据 fa 数据把路径还原并输出即可。

【时空复杂度】

时间复杂度： $O(T*R*C)$ ， T 为数据组数， R,C 为迷宫的长和宽。

空间复杂度： $O(R*C)$

【题目编号】

2000 B According to Bartjens

【题目名称】

According to Bartjens

【题目大意】

给出一个由 0~9 组成的字符串，末尾是一个等号。要求在其中插入一个或多个运算符，使得最后的结果为 2000。

对最后的算式有如下要求：

- 1.写的数字没有前导零。例如 $2*10*0100=$ 就是不可行的。
- 2.没有多个 0。例如 $2*1000+000=$ 就是不可行的。
- 3.二元运算符，不用取负。所以 $2*-100*-10+0=$ 也不合法。
- 4.+、-、*，不用/和括号。
- 5.这些算式按照正常的优先级顺序计算。

要求输出所有合法的方案。字符串长度 $n \leq 9$

【题目关键字】

枚举，模拟

【算法讨论】

很明显，只有在两个数字中间才有可能插入运算符。枚举所有可能的方案，共 $4^{(n-1)}=4^8=65536$ 种，然后暴力判断是否合法即可。时间复杂度为 $O(4^n*n)$ ，可以轻松通过。

【时空复杂度】

时间复杂度： $O(4^n*n)$

空间复杂度： $O(n)$

【题目编号】

2000 C Cutting Chains

【题目名称】

Cutting Chains

【题目大意】

给出 n 个链环，其中有一些链环互相连接在一起。现在你可以将一些链环打开和关上。问最少打开几个环可以将所有链环连成连续的一段链。 $n \leq 15$ ，多组数据。

【题目关键字】

搜索，枚举

【算法讨论】

因为 n 很小，所以我们可以考虑暴力枚举打开哪些链环，然后判断是否合法。

把链环看成点，把连接关系看成一条无向边（注意删除重边）。然后把要打开的链环对应的节点和该节点连接的边删除。在剩下的图中，如果有环，或者存在度数大于 3 的点，那么显然不合法。

接下来图中应该剩下一些长度不等的链，一个打开的链环可以连接两条链，统计链的个数，链的数量大于打开的链环数，那么不合法，否则一定存在合法解。

打开链环的不同方案数为 2^n ，而判断的复杂度是 $O(n^2)$ 的，总的复杂度为 $O(n^2 \cdot 2^n)$ ，可以通过本题。

【时空复杂度】

时间复杂度： $O(n^2 \cdot 2^n)$

空间复杂度： $O(n^2)$

【题目编号】

2000 E Internet Bandwidth

【题目名称】

Internet Bandwidth

【题目大意】

给出一个 n 个点， m 条边的无向图。给出每条边单位时间内传送信息量的最大值。问节点 S 和 T 之间单位时间内能传输信息量的最大值。 $n \leq 100, m \leq n(n-1)/2$

【题目关键字】

网络流

【算法讨论】

很明显的最大流模型。单位时间内传送信息量的最大值就是每条边的容量，答案即为节点 S 到 T 的最大流。

【时空复杂度】

时间复杂度: $O(n^2 \cdot m)$

空间复杂度: $O(m)$

【题目编号】

2000 F Page Hopping

【题目名称】

Page Hopping

【题目大意】

给出一个 n 个点的有向图，询问任意两点间的最短路的平均值。 $n \leq 100$

【题目关键字】

最短路，floyd

【算法讨论】

因为 n 很小，所以直接用 floyd 算出任两点的最短路然后求平均数并输出即可。

【时空复杂度】

时间复杂度： $O(n^3)$

空间复杂度： $O(n^2)$

【题目编号】

2001 A Airport Configuration

【题目名称】

Airport Configuration

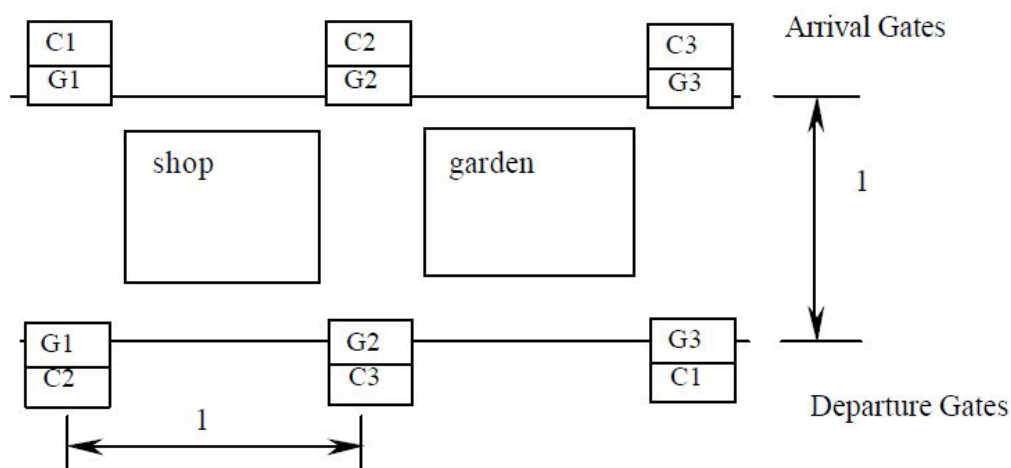
【题目大意】

ACM 机场是一个本地机场，对于大多数人来说，机场不是他们的终点或起点，而是中转站。机场有一个规划图。到达的大门在机场的北边（相当于空格）。出发的大门在机场的南边（也相当于空格）。两个正对着的大门距离相当于大门间的距离。每一个到达的大门只对应一个城市。每一个出发的大门也是这样。乘客到达的大门对应他们的起始城市，而出发大门对应他们的目标城市。因为这个问题，我们只需考虑转机的乘客。

转机的乘客会产生机场的交通堵塞。我们已经知道某两个城市之间的平均客流量。用这些信息，有可能降低交通堵塞。例如，Cx 城到 Cy 城的客流量大，就可以将他们安排得很近，甚至是对位。

因为花园和商店无法穿越，所以到达门 G1 和出发们 G3（见图）的距离为 $1+2=3$ 。

你需要计算几个方案的客流指数。两个大门间的客流指数等于人数乘以距离。而总的客流指数就是所有门之间的客流指数之和。



$$n \leq 25$$

【题目关键字】

模拟

【算法讨论】

用一个二维数组存下任意两个城市之间的客流量。然后对于每个方案，因此枚举每对城市，计算人数乘以距离，统计答案。然后排序输出即可。

【时间复杂度】

时间复杂度： $O(20 \cdot n^2)$

空间复杂度： $O(n^2)$

【试题编号】

2001 B Say Cheese

【题目名称】

Say Cheese

【题目大意】

给出三维空间中的两个点和若干个中空的球体。在空间中移动 1 个单位长度需要花费 10 个单位时间，但在球体内部移动不花时间。问从两点中一点到另一点的最少时间

【试题关键字】

最短路

【算法分析】

把两个点看做半径为 0 的球体。两个球体之间距离为球心距离-两球半径和，若小于 0 则设为 0。用最短路算法求出两点间最短路后乘以 10 输出即可。

【时空复杂度】

时间复杂度: $O(n^2)$

空间复杂度: $O(n)$

【题目编号】

2001 F A Major Problem

【题目名称】

A Major Problem

【题目大意】

在西方音乐中，用大写字母 A 到 G 来表示在乐谱中被使用的 12 个音符，它们后面可能连有升调符号“#”和降调符号“b”，而且如下面展示的那样循环排列。斜杠用来描述相同音符。

C/B# C#/Db D D#/Eb E/Fb F/E# F#/Gb G G#/Ab A A#/Bb B/Cb C/B# ...

上表中任意两个相邻音符构成一个半音。恰被一个音符隔开的两个音符构成一个全音。一个大调音阶由八个音符组成。它由上述之一的音符开始并且紧跟着连续的“全音-全音-半音-全音-全音-全音-半音”。分别由 C 和 Db 开始的两个大调音阶，将由下面的音符组成：

C D E F G A B C

Db Eb F Gb Ab Bb C Db

下面的规则同样适用于大调音阶：

1. A 到 G 的每个字母将在音阶中出现恰好一次，同时第一个字母将例外地在音阶最后重复出现一次。

2. 音阶当中不允许同时出现升调或降调记号。

大调音阶的首个音符被认为是这个音阶的曲调。比方说，上面的两个音阶分别是 C 和 Db 大调音阶。更换两个音阶的音符就是交换对应位置的音符那么简单。举个例子，C 大调音阶中的 F 会和 Db 大调音阶中的 Gb 交换因为它们在各自音阶中的位置相同。

现在给出一个源曲调和一个目标曲调，以及若干个音符。要求源曲调是否有效，目标曲调是否有效，以及若两个曲调都有效，给出的音符在是否源曲调中，以及在源曲调中的音符在目标曲调对应位置上的音符。

多组数据，数据组数 ≤ 100

【题目关键字】

枚举，模拟

【算法讨论】

第一步搜索出所有合法的曲调的音阶。然后对于每组输入数据，暴力枚举两种曲调是否存在，然后在源曲调中通过枚举找到它们的位置，并输出目标曲调该位置上的音符即可。

【时间复杂度】

时间复杂度： $O(T*S)$ ，T 为输入数据组数，S 为合法曲调数量

空间复杂度： $O(S)$

【试题编号】

2002 A Ballons in a Box

【题目名称】

Ballons in a Box

【题目大意】

给出一个长方体，和若干个点。你可以以任意的顺序选择其中若干个点，作为气球的中心。然后你可以依次将气球充气直到碰到长方体的边缘。选择的点不能在长方体外或某个气球中。问最后气球的最大总体积。点数 $n \leq 6$ ，坐标绝对值不超过 1000

【试题关键字】

枚举

【算法分析】

因为题目范围 n 很小，所以暴力枚举选择的气球和气球膨胀的顺序，然后令气球膨胀直到碰到长方体边缘或其他气球，并统计答案即可。

【时空复杂度】

时间复杂度: $O(n!)$

空间复杂度: $O(n)$

【试题编号】

2002 C Crossing the Desert

【题目名称】

Crossing the Desert

【题目大意】

沙漠中有 n 个点，节点 1 为商店，也就是你的出发点，节点 n 为终点。2~ $n-1$ 节点为绿洲。在沙漠中每移动距离 1 就要消耗 1 个单位的食物和水。你最多可以携带总共 tot 单位的食物和水，在商店可以购买食物和收集水，在绿洲可以储存食物和收集水。问最后抵达终点最少需要在商店购买多少食物。 $n \leq 20, tot \leq 100$

【试题关键字】

最短路，二分

【算法分析】

令 $f[i]$ 表示从节点 i 出发，到达节点 n ，初始最少需要在节点 i 储存多少食物。然后从节点 n 开始倒着求答案。假设我们已知节点 t 到 n 的答案 $f[s]$ ，现在要求从节点 s 到 t 再到 n 的答案 $f[t]$ ，因为 tot 的限制我们就需要在 s 到 t 间往返多次来运送总共 $f[s]$ 的食物，往返的次数可以二分或可以直接计算。往返次数确定后，往返路程加上 $f[s]$ 即为 $f[t]$ 的一个解。求解的过程类似最短路。最后求出 $f[1]$ 即为答案。

【时空复杂度】

时间复杂度： $O(n^2)$ 或 $O(n^2 \log n)$

空间复杂度： $O(n)$

【题目编号】

2002 E Island Hopping

【题目名称】

Island Hopping

【题目大意】

给出平面 n 个点的坐标，第 i 个点上有 m_i 个居民，节点 1 已经连入了互联网。现在要修建若干条电缆，使得所有点与节点 1 之间或间接相连。在两点间修电缆的长度即为两点的欧几里得距离。确定要修建哪些电缆后，所有电缆同时开始施工。每个单位时间每条电缆能修建 1 个长度单位。问在电缆总长度最短的方案中所有居民接入互联网的品均时间。

$n \leq 100$ ，其它数字绝对值 $\leq 2^{31}-1$ ，多组数据。

【题目关键字】

最小生成树，最短路

【算法讨论】

显然，电缆总长度最短的方案中即为原图的最小生成树。一个点上的居民接入互联网的时间即为这个点在最小生成树中到节点 1 的路径上最长的边的长度。

因为最小生成树上任两点间树上路径上的最长边是这两点所有路径中最长边最小的，因此不必求出最小生成树，直接用最短路求节点 1 到达每个点路径上的最长边最短为多少，即可求出每个点的居民接入互联网的时间。这一步的时间复杂度是 $O(n^2)$ 的。然后用所有居民接入互联网的总时间除以居民总数即为答案。

【时空复杂度】

时间复杂度： $O(n^2)$

空间复杂度： $O(n)$

【试题编号】

2002 G Partitions

【题目名称】

Partitions

【题目大意】

一个矩形的划分是指把一个矩形分成若干个较小的、不重叠的子矩形。设 A, B 为对同一个矩形不同的两种划分。如果 B 是 A 通过划分它的一个或多个子矩形得来，那么我们说 B 比 A 更精细，或者说 A 比 B 更粗糙。将 A, B 都精细的划分中最粗糙的成为下确界，将比 A, B 都粗糙的划分中最精细的上确界。现在给出大小为 $w \times h$ ($0 < w, h < 21$) 的矩形的两种不同的划分，要求它们的下确界和上确界。

【试题关键字】

floodfill

【算法分析】

下确界的求法很简单，直接对两种划分求并即可。

上确界的话，先对两种划分求交。两种划分的角不一定合法。那么把不合法的线段删除。

不合法的线段有两种：没有闭合的|或_，以及突出的角。

处理方法是对矩形做一次 floodfill，然后如果|或_的两端位于同一个连通块，那么删除。如果一个角外的三个格子属于同一块，那么也删除。然后重复上述过程，直到找不到不合法的线段位置。

floodfill 一个的复杂度是 $w \times h$ 的，每次最少删除一条线段，因此总的复杂度最坏是 $O((wh)^2)$ 的，可以解决本题。

【时空复杂度】

时间复杂度： $O((wh)^2)$

空间复杂度： $O(wh)$

【题目编号】

2002 H Silly Sort

【题目名称】

Silly Sort

【题目大意】

给出由 n 个不同的数组成的序列，每次可以交换序列中两个数的位置。每次交换的代价是交换的两个数的和，问把这个序列按升序排序的最小代价。 $n \leq 1000$ ，每个数 ≤ 1000 ，多组数据。

【题目关键字】

贪心，排序

【算法讨论】

因为每个数都不相等，所以排序后每个数所在的位置是唯一确定的。把每个位置看成一个节点，然后将每个位置上这个位置上的数排序后所在的位置连一条有向边。然后整个图会被分成若干个环。

对于每个环，只有两种最优解：用环内最小值作为中介，每次把环内一个数交换到正确位置；把环外的最小值和环内最小值交换，用这个最小值作为中介，每次环内把一个数交换到正确位置，最后再把环外的最小值和环内最小值交换。两种情况的代价都分别计算，然后取较小值即可。这一步的复杂度是 $O(n)$ 。因此总的复杂度是前面排序的复杂度 $O(n \log n)$ ，可以轻松通过本题。

【时空复杂度】

时间复杂度： $O(n \log n)$

空间复杂度： $O(n)$

【题目编号】

2003 A Building Bridges

【题目名称】

Building Bridges

【题目大意】

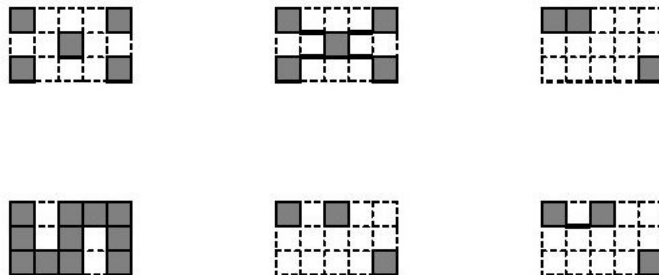
新奥德卫莱城市委员会计划建造一个连接系统来连接所有市中心的建筑,这样人们从一个建筑去另一个建筑时就不用走到外面了。你要写一个程序来帮忙确定建造方案。

新奥德卫莱市是正方形网状布局,每个建筑物占着一些连通的格子,两个有建筑物的格子如果有角接触就算相连,不需要连接道路。道路只能在正方形的边上建造,并且只能是直线且连接两个建筑。

给你一个建筑示意图,你要算出连接所有建筑的最小道路数。如果这是不可能的,找出最少的无法连接的建筑群数。在可连接所有建筑物的情况下,如果多种方案道路数都是最小的,选择在网格中道路总长度最小的方案。道路可以互相穿过,但在这种情况下道路被判定互不相干,之间无法连接。

下图说明了 4 种可能的城市配置。城市 1 包含 5 个建筑,被 4 条道路连接起来,总长度为 4。城市 2 中,因没有建筑共享网格线所以无法建立道路。在城市 3 中,因为只有 1 个建筑所以我们不需要道路。在城市 4 中,最好方案是用 1 条长度为 1 的道路连接两个城市,剩下两个不连通的建筑群(一个是两个建筑,一个是一个建筑)。

左上和城市 1, 中上是建路后的城市 1, 右上是不能建路的城市 2, 左下是不需要建路的城市 3, 中下是城市 4, 右下是建路后的城市 4。



城市对应的正方形网格大小为 $R \times C, R, C \leq 50$, 多组数据。

【题目关键字】

floodfill, mst

【算法讨论】

首先,用一次 floodfill 把所有连通的建筑群找出来,这一步的复杂度是 $O(R \times C)$ 的。

然后,找出所有连通的建筑群,考虑道路的情况。道路只有水平和垂直两种。分开考虑两种道路所在的 $R+1$ 条和 $C+1$ 条直线。很明显只有在同一条水平线上有公共点、且在该水平线上的公共点之间没有建筑物阻挡的两个建筑群直接可以建立一条道路。把所有这样的道路抽象出来看成边,建筑群看成点,做一次最小生成树即为答案。

最后,因为边权一定不超过 $\max(R, C)$, 所以用 kruskal 算法的话,排序可以在 $O(R \times C)$ 的时间内完成,可以在时限内通过本题。

【时空复杂度】

时间复杂度: $O(R \times C \times T)$, T 为数据组数

空间复杂度: $O(R \times C)$

【题目编号】

2003 B Light Bulbs

【题目名称】

Light Bulbs

【题目大意】

有 n 个灯泡排成一行, 按下每个灯泡上的按钮会改变该灯泡和相邻灯泡的状态(明变灭, 灭变明), 给出灯泡的起始状态和目标状态。要求从起始状态到目标状态的最短操作序列, 如果有多种方案, 输出字典序最小的一个。 $n \leq 100$

【题目关键字】

构造

【算法讨论】

显然, 每个开关最多被按下一次。

假设我们已经确定了是否按下第一个开关的, 那么之后能改变第一个灯泡状态的操作只有按下第二个开关。如果第一个灯泡与目标状态相同, 那么一定不能按下第二个灯泡, 否则必须按下。

依次类推, 确定了前 i 个开关的状态以后, 如果第 i 个灯泡与目标状态相同, 那么一定不能按下第 $i+1$ 个开关, 否则必须按下。

因此枚举第 1 个开关的状态之后, 按下开关的方案也就唯一确定。接下来只要判断方案是否合法(第 n 个灯泡是否达到目标状态), 并将最优的方案输出即可。

【时空复杂度】

时间复杂度: $O(n)$

空间复杂度: $O(n)$

【题目编号】

2003 F Combining Images

【题目名称】

Combining Images

【题目大意】

一种图像压缩算法是基于四分树编码的。进行四分树编码的图片必须是一个二进制像素方阵（每个像素的值都是 0 或 1，代表颜色），方阵的边长必须是 2 的幂次。

如果一个图像中所有像素的颜色都相同，这个图像的四分树编码以 1 开始，然后是每个像素的颜色。例如，一个每个像素都是 1 的图像的四分树编码是 11，不管这个图像有多大。

如果一个图像中含有不同颜色的像素，这个图像的四分树编码以 0 开始，然后依次是它的左上象限、右上象限、左下象限、右下象限的四分树编码。

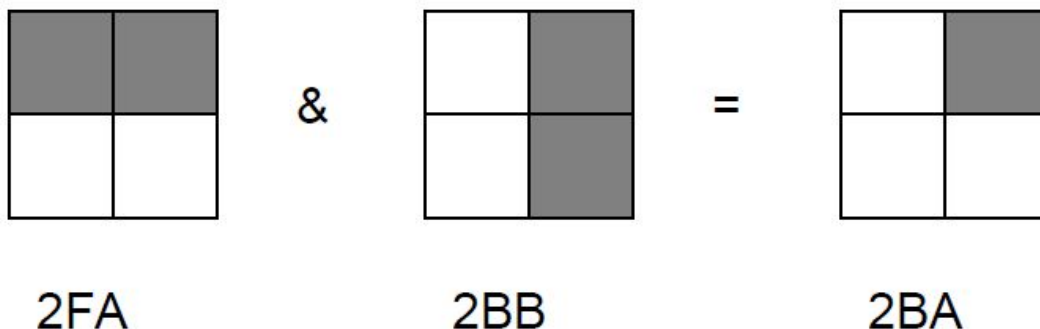
这样得到的四分树编码是一个二进制串。为便于打印，可按如下步骤将其转换为十六进制四分树编码。

- 在四分树编码的开头插入一个 1 作为分隔符；
- 再在编码开头不断插入 0，直到编码的位数是 4 的倍数。
- 将每 4 位二进制编码转化为一位十六进制编码。用数字 0 到 9 和大写字母 A 到 F 表示二进制串 0000 到 1111。

例如，一个每个像素都是 1 的图像的十六进制四分树编码是 7，它对应二进制串 0111。

你需要写一个程序，读入两幅图像的十六进制四分树编码，计算两幅图像的交，并输出它的十六进制四分树编码。假设两幅图像都是正方形的且含有一样多的像素（虽然编码的长度不一定相同）。如果两幅图像的大小和形状相同，它们的交（用 $A \& B$ 表示）也会有同样的大小和形状。根据定义， $A \& B$ 中一个像素是 1 当且仅当 A 和 B 中对应的像素都是 1。

下图举例说明了两幅输入图像和它们的交，并给出了每幅图像的十六进制四分树编码。图中阴影方格表示颜色为 1 的像素。



给出的两幅输入图像的四分树编码长度不超过 100。

【题目关键字】

模拟

【算法讨论】

直接按题意根据给出的两幅输入图像的四分树编码把图形的每个像素还原出来的话空间太大。但我们可以通过递归直接通过四分树编码求两个图形的交的四分树编码即可，按照题意直接模拟即可，时间复杂等于两个图形的四分树编码长度和，可以轻松通过本题。

【时空复杂度】

$O(L)$, L 为四分树编码长度

$O(L)$

【题目编号】

2003 G A Linking Loader

【题目名称】

A Linking Loader

【题目大意】

给出一个程序，程序由许多模块组成。一个对象模块有序地包含 0 个或多个外部符号定义、0 个或多个外部符号引用、0 个或多个字节的代码和数据（可能包含对外部符号值的引用）以及一个模块结尾标志。在这个问题中，一个对象模块表示为一个文本行序列，每行以一个大写字母开头，描述了剩余文本的含义。它们的格式如下所示。每行文本间存在空白分隔符（一些空格或制表符），每行的末尾也可能存在多余的空白分隔符。

D symbol offset

“D”语句是一个外部符号定义。它定义了 symbol 的地址为当前模块的代码和数据的第一个字节地址向后偏移 offset 字节。symbol 是一个长度小于等于 8 的大写字母字符串。offset 是一个至多 4 位的十六进制数（使用大写字母 A-F）。单组数据中“D”语句数量不超过 100。

E symbol

“E”语句是一个外部符号引用。它表明 symbol 的值可能会在当前模块的代码或数据中被调用（可能被定义在另一个对象模块中，并且可能会在当前模块之后）。每一个模块中，“E”语句从 0 开始连续编号，使它们能够在“C”语句中被引用。

C n byte1 byte2 ... byten

“C”语句指定了当前模块的代码和数据的第一个或后 n 个字节的地址。数值 n 是一位或两位的十六进制数，并且不超过十六进制的 10。每一个 byte 是一位或两位的十六进制数，或者是一个“\$”符号。一个“\$”符号后方跟随一个字节，表示引用当前模块中一个从 0 开始编号的外部符号。编译器将这个符号的值（即它的地址）插入当前链接程序所指的位置，十六进制数的高位放在“\$”所指示的位置。其余的字节将会被放入连续的内存地址中，起始位置为第一个未使用的内存地址。如果这个被引用的符号从未被定义，那么将其地址视为(0000)16。

一行一个字母“Z”代表了当前模块的结束。

你可以假设不存在超过四位十六进制数的地址。每行的格式都按照以上所述并且不存在语法错误。要求输出内存的 16 进制校验和以及每个外部符号的地址。

十六位校验和的计算方法如下：首先将其值设为 0，然后以升序遍历所有地址，每次循环左移一位，然后加上该内存地址所存储的值，并无视溢出。

循环左移的定义为，将十六位校验和的二进制的最高位移至最低位。比如 FF00 循环左移一位之后得到 FE01。

【题目关键字】

模拟，字符串

【算法讨论】

按照题意直接模拟即可。首先将全部操作读入，然后就可以计算出每个模块的开始地址。通过每个模块的开始地址，就可以计算出每个外部符号的地址。最后在根据 C 操作完每个位置写入数字。最后输出答案即可。

【时空复杂度】

时间复杂度：O(nlogn)

空间复杂度：O(n)

【题目编号】**2003 H A Spy in the Metro****【题目名称】****A Spy in the Metro****【题目大意】**

地铁有 N 站，从 $1 \sim N$ 连续编号，地铁在相邻两站间的运行时间是固定的，从第 i 站往第 $i+1$ 和从第 $i+1$ 站到第 i 站都需要运行 t_i 的时间。从首站有 m_1 列地铁开往末站，出发时间分别为 d_1, d_2, \dots, d_{m_1} ($0 \leq d_i \leq 250$ 且 $d_i < d_{i+1}$)，从末站到首站有 m_2 列地铁，出发时间分别为 e_1, e_2, \dots, e_{m_2} ($0 \leq e_i \leq 250$ 且 $e_i < e_{i+1}$)，地铁停靠和人上下地铁都在瞬间完成，现在有个人 0 时刻从首站出发，要在 T 时刻准时到达末站，问过程中这个人在站台上等待的最短总时间。多组数据，数据组数 $Q \leq 1500, N \leq 50, T \leq 200, t_i \leq 20$ 。

【题目关键字】

最短路

【算法讨论】

最短路，令 $f[i][j]$ 为人在时刻 j 到达站台 i 的在站台上的最短等待时间。从状态 (i, j) 出发有两种转移，一种是在站台上等待，转移到状态 $f(i, j+1)$ ，等待时间+1。另外一种为在 j 时刻上车到达相邻站台。处理这种转移需要预处理到首站每个站台所需时间 S_i 和末站到每个站台的时间 T_i 。然后要到 $i-1$ 号站台需要末站在 $j-T_i$ 这个时刻发一班列车，到 $i+1$ 号站台则需要首站在 $j-S_i$ 这个时刻发一班列车。可以用两个布尔数组分别记录首站和末站发车的时刻。

把每个状态看成一个节点，两个状态能够转移则连一条带权的有向边。那么我们需要在一个点数为 $n*T$ 的，边数与点数同阶的有向图上求最短路。用 Dijkstra+heap 的话单次最短路为 $O(n*T \log(n*T))$ ，总共有 1500 组数据，会 TLE。但注意到本题因为在 T 时刻要赶到终点，最短路长度不会超过 T ，因此我们可以用一个桶来代替堆完成取最小权值节点的操作，单次最短路复杂度下降为 $O(n*T)$ ，可以通过本题。

【时空复杂度】时间复杂度： $O(Q*n*T)$ 空间复杂度： $O(n*T)$

【题目编号】

2003 I The Solar System

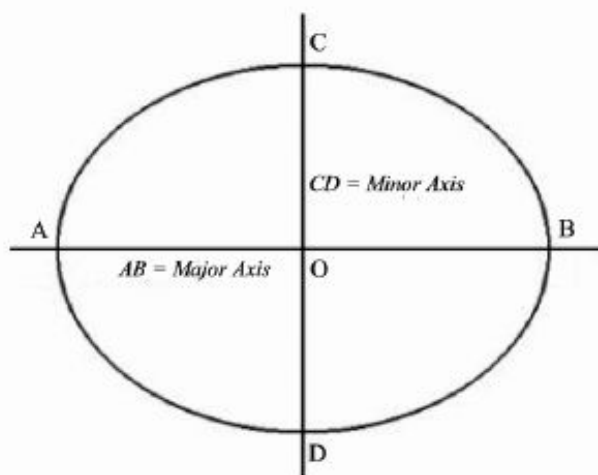
【题目名称】

The Solar System

【题目大意】

给出开普勒第一、第二和第三定律。

1. 行星的轨道是椭圆，太阳是椭圆的其中一个焦点。（回想椭圆的两个焦点是椭圆上所有点到它们的距离和都相等的点）
2. 当行星在椭圆轨道上运行时，连接行星和太阳的线在相等的时间内扫过相等的面积。
3. 两颗行星的环绕周期平方比等于它们的半长轴立方比。



在这个问题中，你要使用开普勒定律计算行星的位置。给出太阳系中一个行星的描述，就是说行星轨道的半长轴长度 a_1 ，半短轴长度 b_1 ，和它的环绕周期 t_1 ，以及第二个行星的描述，它的长轴 a_2 和短轴 b_2 。假设第二个行星的轨道关于坐标轴对称（就像上面的图像中一样），它沿逆时针方向运动，并且太阳位于 x 坐标非负的焦点处。你要计算第二个行星从轨道上的 x 坐标最大处（上图中的点 B）开始，在经过时间 t 后到达的位置。

多组数据，数据组数 $T \leq 100$ ，出现的所有数字在 C++ 的 `int` 类型范围内。

【题目关键字】

数学、椭圆、二分

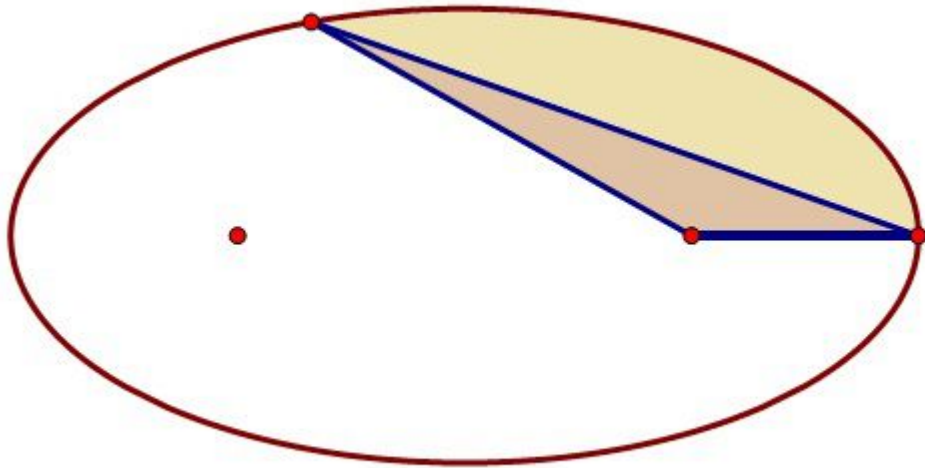
【算法讨论】

显然，根据开普勒第二、第三定律，可以算出第二行星的环绕周期为 $t_2 = \sqrt{t_1^2 \cdot \left(\frac{a_2^3}{a_1^3}\right)}$ ，

行星和太阳的射线在时间 t 内扫过的面积为 $a_2 b_2 \pi \cdot \frac{t}{t_2}$ 。

令 $a = |AB|/2$ ， $b = |CD|/2$ ，那么椭圆被定义为 $x/a + y/b = 1$ 。二分向量 $(x/a, y/b)$ 与 x 正半轴所成夹角的大小，然后算出行星运动到 (x, y) 时，连接行星和太阳的射线扫过的椭圆面积。

如下图椭圆被扫过的面积可以被分成一个三角形和一个弓形。三角形的面积很容易计算，弓形的面积可以在单位圆上算出二分向量 $(x/a, y/b)$ 与 x 正半轴所成夹角对应的弦与周长所成的弓形面积，然后乘以 $a_2 b_2$ 即可。



【时间复杂度】

时间复杂度: $O(T \log n)$

空间复杂度: $O(1)$

【题目编号】

2003 J Toll

【题目名称】

Toll

【题目大意】

有 a 到 z 共 26 个村庄，A 到 Z 共 26 个城镇。一些村庄和城镇之间有一些双向路径。当你身上携带货物时，进入一座村庄要缴纳 1 件货物，而进入城镇时每携带 20 件货物就要缴纳一件货物，例如，携带 70 件货物进入城镇需要交纳 4 件货物。

共有 n 条双向路径，给出起点和终点，问要从起点出发，把 p 件货物送达终点，在起点至少要携带几件货物。

$0 \leq n, 0 < p \leq 1000$ ，数据没有重边。

【题目关键字】

二分，最短路

【算法讨论】

答案显然具有二分性。二分答案，然后计算最多能运送多少货物到达终点。

令 $F[i]$ 表示到达节点 i，最多剩下的货物数目。转移可以之间使用 $O(n^2)$ 的 dijkstra 完成。因为这题边数最坏情况是 $O(n^2)$ 的，所以没有用堆优化的必要。

最后总的时间复杂度为 $O(n^2 \log n)$ 。

【时间复杂度】

时间复杂度： $O(n^2 \log n)$

空间复杂度： $O(n^2)$

【题目编号】

2004 E Intersecting Dates

【题目名称】

Intersecting Dates

【题目大意】

服务器上存储着 1700~2100 年每一天的股票信息。以前已经在服务器上访问过不同的 NX 个日期区间中每一天的股票信息。现在要知道 NY 个日期区间中每一天的股票信心。问 NY 个日期区间中哪些天的股票信息需要再次到服务器上访问。把需要再次上服务器上访问的日期以区间形式输出。

$NX, NX \leq 100$, 多组数据数据组数 ≤ 100 。

【题目关键字】

排序，枚举

【算法讨论】

显然， NX 个日期区间中互相有交集（包括边界）的区间可以合并， NY 个日期区间中互相有交集的同理。先各用一次排序然后扫描即可完成合并。

接下来枚举 NY 个日期区间，然后枚举 NX 个日期中每一个，把这些区间与枚举的询问的区间的交集删除，剩下的即为需要再次到服务器上访问的信息。因为数据范围较小，所以这一步暴力即可。

最后总的时间复杂度为 $O(NX * NY * T)$, T 为数据组数，可以通过本题。

【时空复杂度】

时间复杂度： $O(NX * NY * T)$ ， T 为数据组数

空间复杂度： $O(NX + NY)$

【题目编号】

2004 G Navigation

【题目名称】

Navigation

【题目大意】

给出 n 个信号源，给出一个终点。每个信号源以 100m/s 的速度向不同的方向直线运动，给出每个信号源的坐标(x_i, y_i)、移动方向 D ，发送信号的时间 T 以及你接收到信号的时间 t ，信号以 350m/s 的速度传播。问你相对终点的位置，多解输出 Inconclusive，无解输出 Inconsistent。 $n \leq 10$, 多组数据，数据组数 ≤ 5

【题目关键字】

数学

【算法讨论】

根据信号源的移动方向和发送信号的时间，可以求出信号源发送信号时的坐标。

给出接收到信号的时间，可以知道你的位置的范围，是以为发送信号时的坐标中心的一个半径为 $(t-T)*350m$ 的圆的周长上。枚举两个圆求交点，然后分别判断两个交点是否在所有圆的周长上即可。

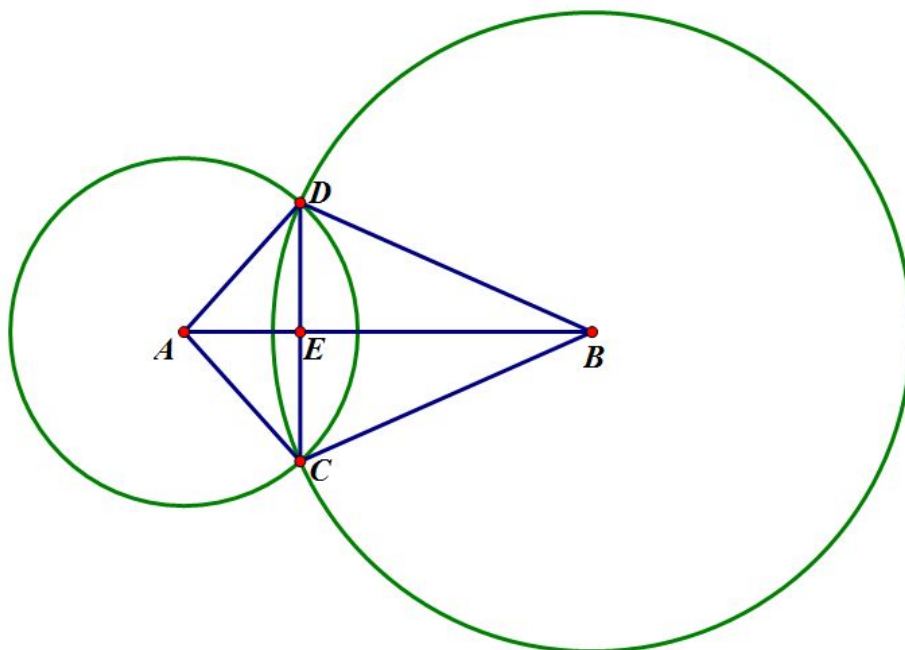
圆的交点的求法：如图，设圆 A 和圆 B 交于点 C,D，连接 AB，CD，AB,CD 交于点 E。

$$AE^2 + DE^2 = AD^2$$

显然，AB 垂直 CD。 $BE^2 + DE^2 = BD^2$ ，其中 AD,BD,AB 已知，解这个方程组然后通过

$$AE + BE = AB$$

向量运算即可求出两个交点。



【时空复杂度】

时间复杂度： $O(n)$

空间复杂度: $O(n)$

【题目编号】

2004 H Tree-Lined Streets

【题目名称】

Tree-Lined Streets

【题目大意】

给出 n 条线段, 任意两条线段最多只有一个交点。要求你在线段上种树。同一条线段上两棵树的位置距离不能少于 50, 且树的位置距离交点不少于 25。问最多种几棵树。

【题目关键字】

计算几何, 排序

【算法讨论】

不同线段上的树互不影响。因此可以每条线段分开处理。

对于一条线段, 计算其它线段和它的交点, 然后把交点按照到线段一段的距离从小到大排序。这些交点把线段分成若干段, 对于每一段可以分别求答案。若长度 < 50 , 不能种树, 否则可以种上最多 $(\text{长度} - 50) / 50 + 1$ 向下取整这么多棵。注意首尾两段只有一个交点, 需要特判。还有特判无交点的情况即可。

【时空复杂度】

时间复杂度: $O(n^2 \log n)$

空间复杂度: $O(n)$

【题目编号】

2004 I Suspense

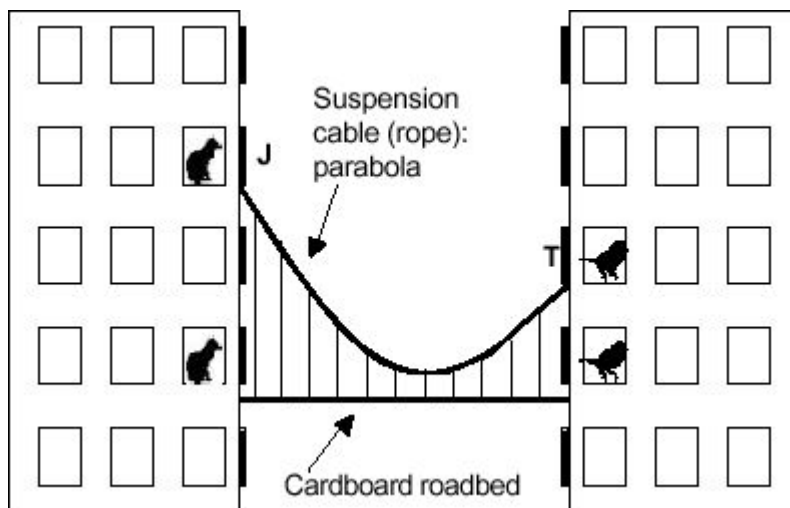
【题目名称】

Suspense

【题目大意】

给出两栋距离为 D 的公寓，要求用绳子和硬纸板建造一座微型的吊桥。两条一样长的支缆组成了主缆。这两根支缆的两端被系在第一栋公寓第 J 层的窗台上和第二栋公寓第 T 层的窗台上。用硬纸板做的“桥面”是被大量的绳子系在主缆上的。水平的桥面正好比主缆的最低点的低 1 米。由于一些审美原因，桥面至少比两个窗户的最低点低 2 米。根据物理常识，每条吊索都会形成一个抛物线。

公寓的一些房客饲养了宠物猫，要求保证这座纸桥不会提供一条猫抓到鸟的路。一只猫不能往上跳 0.5 米，也不能往下跳 3 米。所以只要桥面比猫的窗户的最低端高 0.5 米及以上，或比猫的窗户的最低端低 3 米及以上，猫就不能跳上桥面了。同样地，如果鸟的窗户的最低端比桥面高 0.5 米及以上，或比桥面低 3 米及以上，猫就算到了桥面上，也没法抓住这只鸟。猫只关心抓鸟，它们不关心怎么回家。下图给出了一种搭桥的方案



请你写一个程序，要求在不伤害任何鸟的情况下，他们做主缆的一根支缆需要多长的绳子。

给出两幢建筑之间的距离 D ，绳子两端所在层数 J 和 T ；第一栋公寓每层所养的宠物类型，第二栋公寓每层所养的宠物类型。你的程序需要输出最长的能建造支缆的绳子长度，使得没有一只猫能通过这座桥抓到一只鸟。桥面必须至少比地面高一米，且必须正好比主缆的最低点低 1 米，以及至少比主缆的最低点都低 2 米。这两座公寓的每个房间都是 3 米高，所有窗户都是 1.5 米高，且所有窗户的最低端都比所在的房间的地面高 1 米。

$D \leq 25, 2 \leq J, T \leq 25$, 数据组数 ≤ 100000

【题目关键字】

数学

【算法讨论】

根据所给出的宠物的信息，可以很容易的算出合法的桥面高度区间。显然，要使得主缆最长，桥面的高度必须最低。

知道了桥面高度后，关键是如何求主缆的最低点和抛物线的长度。

令主缆的最低点到第一栋公寓距离为 x ，主缆两端到桥面的垂直距离为 h_J, h_T ，抛物线方

程为 $y=kx^2$ ，列出方程： $\frac{kx^2}{k(l-x)^2} = \frac{h_j}{h_t}$ ，变形可得 $x^2 \bullet h_t = h_j \bullet (l-x)$ 。继续变形可以得到

到一元二次方程，解之可得 x ，然后把 x 代入 $kx^2 = h_j$ ，即可得到 k 。

解出最低点的位置和抛物线方程后，即可得到抛物线长度。最后求抛物线长度是 $O(1)$ 的，找到合法的最低桥面高度是 $O(J)$ 的，总时间复杂度为 $O(\text{数据组数} \cdot J)$ 。

【时空复杂度】

时间复杂度： $O(Q \cdot J)$ ， Q 为数据组数

空间复杂度： $O(J)$

【题目编号】

2005 B Simplified GSM Network

【题目名称】

Simplified GSM Network

【题目大意】

平面上有 B 座 BTS 塔，平面上每个点都会连接到它最近的 BTS 塔。一个 BTS 塔覆盖的区域被称为一个 cell。有 C 座城市， R 条道路。某些城市之间有道路连接。道路是一条连接两个城市的直线段。有 Q 次询问，每次询问从一座城市到另一座城市最少要经过几次 cell 的边缘。

$B, C \leq 50, R \leq 250, Q \leq 10$

【题目关键字】

计算几何，最短路

【算法讨论】

如果求出了每条道路会穿过几次 cell 的边缘，那么这题就转化成一个简单的最短路问题。直接用 $O(B^3)$ floyd 求出任两点间最短路并输出即可。

主要问题在于求出每条道路会穿过几次 cell 的边缘。每个 cell 的边缘肯定是某两个 BTS 的中垂线的一部分，因此我们可以枚举所有的中垂线与道路求交点，然后判断有没有其他 BTS 距离这个交点比这两个 BTS 更近，如果没有，那么这条道路就会穿过这两个 BTS 所对应的 cell 的边缘。这样的时间复杂度是 $O(R*B*B)$ 的，可以轻松通过本题。

【时空复杂度】

时间复杂度: $O(R*B^2+B^3)$

空间复杂度: $O(B^2)$

【题目编号】

2005 C The Traveling Judges Problem

【题目名称】

The Traveling Judges Problem

【题目大意】

给出一个 nc 个点， nr 条边的无向图，每条边有不同的权值。给出一个终点 dc 和 nj 个起点。要求从图中选出若干条边，使得从 nj 个起点出发都可以通过这些边到达终点 dc 。输出边权和最小的方案，如果有多种，输出所有边涉及到的点的集合元素最少的一种。仍然存在多种方案时，选择集合元素升序排列后字典序最小的一种。同时输出每个起点通过选择的边到的终点的任意一条路径。 $nc \leq 20, nj \leq 10$ ，图中任两点最多只有一条边直接相连， $1 \leq \text{边权} \leq 100$ 。

【题目关键字】

枚举，最小生成树

【算法讨论】

因为每个起点都能到达终点，所以显然涉及的点都互相连通。涉及点的集合确定，要选择若干条边使得这些点互相连通且权值最小，这是一个经典的最小生成树问题。因此可以暴力枚举涉及的点的集合，然后求最小生成树即可。最后要求每个点到终点的路径，等价于求该点和终点的树上路径。枚举点集的复杂度为 $O(2^{nc})$ ，求最小生成树的复杂度为 $O(nr \log nr)$ ，但排序可以预处理，复杂度下降为 $O(nr)$ ，因此总的复杂度为 $O(2^{nd} * nr)$ 。对于 $nd=20$ 的数据似乎会超时，但显然终点和起点必定在点集中，因此实际有效的点集并没有 2^{nd} 个，可以通过本题。

【时空复杂度】

时间复杂度： $O(2^{nd} * nr)$

空间复杂度： $O(nr)$

【题目编号】

2005 E Lots of Sunlight

【题目名称】

Lots of Sunlight

【题目大意】

自东向西给出 n 栋公寓楼排成一条直线，每栋公寓宽度均为 w ，每层公寓高度均为 h 。第 i 栋公寓有 $m(i)$ 层，与第 $i+1$ 栋公寓楼距离为 $d(i)$ 。

太阳从东方升起，以恒定角速度划过天空，然后从西方落下。阴影仅由楼房投射出（也即，每栋楼可以投影到一栋或者多栋其他的楼房上）。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，我们就认为公寓受到太阳直射。

给出最多 Q 个询问，每次询问某一间公寓每天送到太阳直射的时间。本题中认为上午 5:37 日出，下午 6:17 日落。多组数据，数组组数 $T \leq 10$ ， $n, m(i), d(i), w, h \leq 100, Q \leq 1000$

【题目关键字】

模拟，三角函数

【算法讨论】

按照题意模拟太阳照射情况。每间公寓的墙壁太阳直射开始的时间，也就是公寓东侧外墙根被照射的时间，每间公寓太阳直射结束的时间，也就是公寓西侧外墙根不再被照射。因此我们枚举太阳照射其它楼房的影子投射在这两个位置的情况，用三角函数计算光线与地面的角度从而算出时间即可计算答案。

【时空复杂度】

时间复杂度： $O(T \cdot n \cdot Q)$

空间复杂度： $O(n)$

【题目编号】

2005 F Crossing Streets

【题目名称】

Crossing Streets

【题目大意】

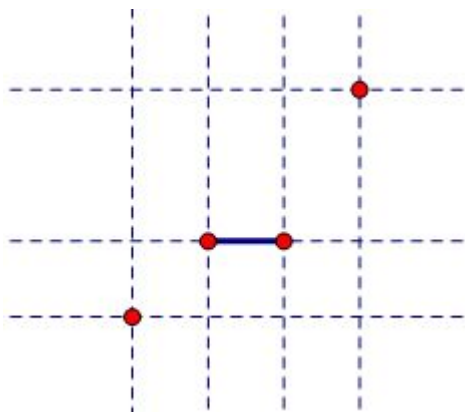
平面上有 n 条线段，每条线段都与 x 轴或 y 轴平行。Peter（可以看成一点）可以在平面上任意行走，可以穿过线段，但不能穿过两条线段的交点，也不能沿着线段走。给出起点和终点，问 Peter 从一点到另一点最少需要穿过几条线段。保证起点和终点均不在任意一条线段上。

$n \leq 500$ ，给出的所有坐标均为整数且小于 2×10^9 。

【题目关键字】

离散化，最短路

【算法讨论】



如图，所有线段的端点最多有 $2 \times n$ 个不同的点。过每个点分别作 x 轴和 y 轴的垂线，把平面分成约 $4 \times n^2$ 个不同的区域。把每个区域看成一个节点。对于两个区域，如果它们之间有一条线段，在它们之间连一条权值为 1 的无向边，否则连一条权值为 0 的无向边。最后从起点所在区域到终点所在区域的最短路即为答案。

这样隔出来的图点数和边数均为 $O(n^2)$ 的级别。因为边权均为 0 或 1，所以可以用 bfs 来找最短路。或者在 Dijkstra 中用一个桶代替堆取最小值。无论采用哪一种算法时间复杂度均为 $O(n^2)$ ，同样可以通过本题。

【时空复杂度】

时间复杂度： $O(n^2)$

空间复杂度： $O(n^2)$

【题目编号】

2005 G Tiling the Plane

【题目名称】

Tiling the Plane

【题目大意】

给出一个简单多边形，多边形的每个角都是直角且每条边长度都是整数，问这个多边形是否能够通过平移和复制来覆盖整个平面。

以下是一些可能有用的信息：

只有两种本质不同的铺满平面的情况：使用正四边形铺满平面（棋盘覆盖），或使用正六边形铺满平面（蜂巢覆盖）。一个多边形当且仅当满足以下两个条件中至少一个时可以铺满平面：

1. 在多边形边界上顺次存在四个点 A,B,C,D（不一定要是多边形的顶点），使得 A 到 B 的边界与 D 到 C 的边界重合，B 到 C 的边界与 A 到 D 的边界重合。这表明这个多边形可以用棋盘覆盖的方式铺满平面。

2. 在多边形边界上顺次存在六个点 A,B,C,D,E,F（不一定要是多边形的顶点），使得 A 到 B 的边界与 E 到 D 的边界重合，B 到 C 的边界与 F 到 E 的边界重合，C 到 D 的边界与 A 到 F 的边界重合。这表明这个多边形可以用蜂巢覆盖的方式铺满平面。

多组数据，测试数据组数 ≤ 50 ，多边形周长 ≤ 50 。

【题目关键字】

枚举，搜索

【算法讨论】

首先，A,B,C,D,E,F 这些点只会在整点上，否则我们可以把它们移动到最近的正点上，答案依然合法。

对于第一种覆盖方式，可以发现很容易发现 ABC 与 CDA 长度相等，对于覆盖方式，可以发现 ABCD 与 DEFA 长度相等。也就是多边形的轮廓划分成了长度相等的两个部分。

并且可以发现四边形覆盖前半中 AB,BC 和 CD,DA 长度分别相等，蜂巢覆盖中 AB,BC,CD 和 DE,EF,FA 长度分别相等。因此在四边形覆盖中，只要枚举 A,B，即可确定 C,D，在蜂巢覆盖中，只要枚举一半的点，就可以确定另一半。所有点确定后，暴力验证即可。

枚举的复杂度最坏是 $O(n^3)$ ，验证的复杂度是 $O(n)$ ，因此总的复杂度最多为 $O(n^4)$ 。 n 很小，并且枚举一半的点的过程中，有大量不合法的状态，可以在时限内通过全部数据。

【时空复杂度】

时间复杂度： $O(n^4)$

空间复杂度： $O(n)$

【题目编号】

2005 H The Great Wall Game

【题目名称】

The Great Wall Game

【题目大意】

给出一个 $n \times n$ 的棋盘，给出棋盘上 n 颗石子的坐标。现在每次操作可以把一颗石子移动到上下左右的任意一个位置。一个格子最多有一颗石子。要求最少的步数，使得所有石子在棋盘上排成一条直线（同一行，同一列或对角线均可）。 $n \leq 15$ ，多组数据组数 ≤ 500

【题目关键字】

贪心，二分图最大权完备匹配

【算法讨论】

因为石子有 n 颗，棋盘大小为 $n \times n$ ，因此石子排成 1 行有 n 种情况，排成 1 列有 n 种情况，排成一条对角线最多有 2 种情况。枚举这 $n+n+2$ 种情况，然后最终哪些位置上有石子也就确定了。把石子从棋盘上一个位置移动到另一个位置，所需步数即为两个位置的曼哈顿距离。这样问题就转化成一个二分图最大权完备匹配问题。因为数据组数较多，同时一组数据要解决 $O(n)$ 次二分图最大权完备匹配问题，只能使用 KM 算法。

但事实上这题要简单的多。计算两点曼哈顿距离可以分别统计两点 x 上的距离和 y 上的距离。在石子排成一列，每个石子上 x 方向的移动距离是确定的， y 方向上的移动转化为一个一维的问题，可以用贪心解决。石子排成一行的情况类似。因此每组数据只要做 2 次二分图最大权完备匹配即可。

【时空复杂度】

时间复杂度： $O(T \cdot n^4)$, T 为数据组数。

空间复杂度： $O(n^2)$

【题目编号】

2005 I Workshops

【题目名称】

Workshops

【题目大意】

有 r 个会议要同时召开，第 r 个会议有 p_i 个人参加，持续 d_i 分钟。有 w 个房间，第 j 个房间可以容纳 s_j 个人，最多使用 t_j 个时间。每个房间安排最多一个会议，安排参加人数不超过房间容量且持续时间不超过房间允许使用时间。问最多能安排几场实验，如果有多种方案，问最少有几人不能参加会议。 $r, w \leq 1000$, $p_i, s_j \leq 100$, $d_i, t_j \leq 300$

【题目关键字】

贪心

【算法讨论】

把会议和房间按照时间分别从小到大排序。按使用时间从小到大枚举每个房间。对于当前房间，给它安排能安排在这个房间的会议中参加人数最多的会议。总的时间复杂度为 $O(n^2)$ 。正确性可以通过反证法证明。

【时空复杂度】

时间复杂度: $O(n^2)$

空间复杂度: $O(n)$

【题目编号】

2005 J Zones

【题目名称】

Zones

【题目大意】

给出 n 座服务塔各自服务范围内的用户数量，它们的服务范围有 m 个公共服务区，给出每个公共服务区的用户数量和给该区提供服务的服务塔。现在要求从其中选择 $n1$ 座服务塔，使得能服务的用户数量最多。公共服务区如果有多个服务塔能提供服务，服务区内的用户数量只计算一次。输出能服务的用户数量最多的方案，如果有多种建造方案能为相同数量的客户提供服务，则优先选择包含 1 号塔的方案。如果在这样的条件下仍有多于一种方案，则优先选择包含 2 号塔的方案，以此类推。 $1 \leq n1 \leq n \leq 20$, $1 \leq m \leq 10$

【题目关键字】

枚举，容斥

【算法讨论】

因为 n, m 很小，枚举即可。枚举所有选择服务塔的方案，并计算这些服务塔内的用户总数，然后枚举公共服务区，减去重复计算的部分即可。

【时空复杂度】

时间复杂度: $O(2^n * m)$

空间复杂度: $O(2^n)$

【题目编号】

2006 A Low Cost Air Travel

【题目名称】

Low Cost Air Travel

【题目大意】

航空公司给出 nt 种机票。对于每种机票，要求旅客必须按次序完成一张票的旅行，而且不允许中途加入其他线路。例如，如果你有一张从城市 1 到城市 2 再到城市 3 的机票，那么你就不能仅使用从城市 2 至城市 3 的这一部分。你必须总是在从机票上的第一个城市出发。除此之外，你也不能从城市 1 飞至城市 2，然后飞到别的地方并返回，最后继续你从城市 2 到城市 3 的旅行。

现在给出 ni 种旅行路线，要求按照先后顺序依次访问路线上的城市，求最小花费和方案。

$nt, ni \leq 20$, 每种旅行路线和机票访问的城市数目不超过 10。多组数据。

【题目关键字】

最短路

【算法讨论】

显然，对于不同的旅行路线可以分开处理。

令 $f[i][j][k]$ 表示已经访问过旅行路线上的前 i 座城市，现在位于机票 j 路线上的第 j 座城市。

初始状态有 $f[1][p][1] = co[p]$ ，若旅行路线上的第一座城市等于机票 p 路线上的第一座城市。 $co[p]$ 为第 p 种机票的价格。转移有两种，一种是向机票路线的下一站转移（若当前部位机票的终点站），另一种是枚举一张从这座城市的出发的机票然后换用这张机票。对于第二种状态，需要另外支付新机票的费用。

这样的转移下，某些状态的依赖关系会形成环，不容易划分明显的阶段。但因为题目求的是最小值，同时机票价格均为正，所以可以用 **dijkstra** 来进行转移。

这样总的复杂度 $O(T * nt * S \log S)$ ， T 为数据组数， S 为 dp 状态总数，对于本题的数据范围 $S \leq 2000$ 。

【时空复杂度】

时间复杂度： $O(T * nt * S \log S)$ ， T 为数据组数， S 为 dp 状态总数，对于本题的数据范围 $S \leq 2000$ 。

空间复杂度： $O(nt * ni * L)$

【题目编号】

2006 B Remember the A La Mode!

【题目名称】

Remember the A La Mode!

【题目大意】

给出 P 种冰淇淋和 I 种饼片。一份甜点由一份冰淇淋和一份饼片组成。已知第 i 种冰淇淋和第 j 种饼片组成的甜品能带来 $F(i,j)$ 的收益 ($F(i,j)=-1$ 则表示不能使用这种组合)。给出每种冰淇淋和饼片的数量, 要求使用所有冰淇淋和饼片组成甜品的前提下, 能获得的最大和最小收益分别是多少。

$P, I \leq 50$, 多组数据。每种冰淇淋和饼片的数量均不超过 100。

【题目关键字】

二分图最大权完备匹配, 费用流

【算法讨论】

这题是一个很明显的二分图带权匹配模型。可以使用费用流来解决。

P 个节点表示冰淇淋, I 个节点表示饼片。新增源点 S 和汇点 T , 从 S 往每个代表冰淇淋的节点连一条容量为冰淇淋个数, 费用为 0 的边。从每个代表饼片的节点往 T 连一条容量为饼片个数, 费用为 0 的边。从第 i 个冰淇淋往第 j 个饼片连一条容量为无穷大, 费用为 $F(i,j)$ 的边。

在这个图上求最小费用最大流, 即为最小收益; 做最大费用最大流, 即为最大收益。因为图的节点数量很少, 所以朴素的费用流即可通过本题。

【时空复杂度】

时间复杂度: $O(n^4)$

空间复杂度: $O(n^2)$

【题目编号】

2006 D BipartiteNumbers

【题目名称】

BipartiteNumbers

【题目大意】

形如 $a \cdots ab \cdots b$ 的数叫做 bipartite number, 例如 44444411, 41,10000000 和 5555556 等都是。给出一个数字 p , 求出 p 的最小的倍数 $n=kp$ ($k>1$), 使得 n 是 bipartite number。

$P<100000$ 多组数据, 数组组数 ≤ 200 , 时限 4s。

【题目关键字】

枚举, 二分

【算法讨论】

首先, 长度为 L 的 bipartite number 最多有 $L*9*9$ 种。预处理形如 1,11,111 的数对 p 取模的结果, 然后枚举 n 的前一部分的数字, n 后一部分的数字, 以及前一部分的数字长度, 然后就可以 $O(1)$ 算出这个数字对 p 取模的结果。

上述的枚举算法大部分数据都会很快出解, 只有少数数据跑的非常慢。经过观察发现跑的慢的数据满足 $p \bmod 10=0$ 。因此可以对这一部分单独枚举。

因为 $p \bmod 10=0$, 所以 $n \bmod 10=0$ 。也就是 n 的后半部分由若干个 0 组成。对于一个数字 n , 假如 $n \bmod p=0$ 的话, 显然 $n*10 \bmod p=0$ 。那么我们枚举组成 n 前一部分的数字和长度, 就可以二分后面 0 的个数。显然 n 前一部分数字长度最多为 p (否则一定出现循环), 因此对于 $p \bmod 10=0$ 的数据, 可以在 $O(p \log p)$ 的时间内解决。数据的最坏情况下所有 $p \bmod 10=0$, 所以时间复杂度最坏为 $O(Tp \log p)$ 。因为时限为 4s, 所以只要程序常数不太大即可通过本题。

【时空复杂度】

时间复杂度: $O(Tp \log p)$, p 为数据组数。

空间复杂度: $O(p)$

【题目编号】

2006 E Bit Compressor

【题目名称】

Bit Compressor

【题目大意】

给出一个一种数据压缩方式。将连续的 n 个 1 替换为 n 的二进制表示（注：替换发生当且仅当这种替换减少了二进制串的总长度）（译者注：连续的 n 个 1 的左右必须是 0 或者是串的开头、结尾）比如：11111111001001111111111111110011 会被压缩成 10000010011110011。原串长为 32,被压缩后串长为 17。

这种方法的弊端在于，有时候解压缩算法会得到不止一个可能的原串，使得我们无法确定原串究竟是什么。请你写一个程序来判定我们能否利用压缩后的信息来确定原串。给出原串长 L ，原串中 1 的个数 N ，以及压缩后的串。 $L \leq 16$ Kbytes，压缩后的串长度 ≤ 40 bits。

【题目关键字】

动态规划，记忆化搜索

【算法讨论】

令 $f[i][j][k]$ 表示把压缩串的前 i 位解码，解码后原串出现了 j 个 0，原串长度为 k 的不同解码的方案数。

转移的时候每次枚举下一个未被解码的位置开始的连续一段进行解码。要注意的是解码的一段前后都必须是 0，同时 10 不能解码成 11（替换发生当且仅当这种替换减少了二进制串的总长度），因此 11 有两种解码方式，不解码或解码成 111。

这样总的状态数是 $O(n^2 * L)$ ，转移 $O(n)$ ，总的时间复杂度为 $O(n^3 * L)$ 。考虑到题目的数据范围，比较危险。但因为实际有效的状态很少，所以可以写成记忆化搜索，实际的时间和空间复杂度大大降低，可以通过本题。

【时空复杂度】

时间复杂度: $O(n^3 * L)$

空间复杂度: $O(n^2 * L)$

【题目编号】

2006 G Pilgrimage

【题目名称】

Pilgrimage

【题目大意】

Jack 带了一帮人去朝圣。Jack 的小册子上记录了四种事: PAY k , 从公用资金中扣除 k 元, IN k , 队伍增加 k 人, OUT k , 队伍减少 k 人, COLLECT K , 队伍中每个人交 k 元给公共资金。当队伍人数变动时, 加入或离开要上交或带走一定数量的钱, 使得每个人拥有的平均公用资金不变。例如有 7 个成员, Jack 有 140 元现金, 即每人 20 元。如果 2 个成员离开了, 每人会得到 20 元, 并且 Jack 会记下: "OUT 2"。在相同的情况下, 如果 3 个新成员加入了, 他们每个人需要支付 20 元, 而 Jack 会记下: "IN 3"。巧合的是, 整个旅程中, 每次每个人上交或带走的钱数都是整数。现在给出 Jack 的小册子的某一部分, 问开始可能有多少成员。无解输出 Impossible, 无限解输出最小的解, 有限解则全部输出。给出事件数 $n \leq 50$, 对于 COLLECT k , $k \leq 200$, 对于 PAY k , $k \leq 2000$, IN k 和 OUT k 中 $k \leq 20$ 。多组数据, 数组组数 $T \leq 30000$ 。

【题目关键字】

数论, 枚举

【算法讨论】

不难发现会出问题的只是 in 和 out 这两种操作。假设第一次 in 或 out 之后的, 有 $n+p$ 个人, 平均每个人的钱是 k , 不妨假设接下来两个操作分别是 pay q 和 in/out t , 那么 pay q

后有 $n+p$ 个人，每个人的钱是 $k-q/(n+p)$ 。这必须是整数，因为下一个操作是 in/out。所以 $n+p$ 必须是 q 的约数，从而 n 只可能有有限个。

从上面的分析不难得出结论，那么问题有有限个解当且仅当至少存在一个 pay 操作夹在两个 in 或 out 操作之间。

删除 collect 操作，把所有连续的 in 或 out 操作之间的 pay 合并，然后得出一系列整除式，每个都是形如 $(n+a)|b$ 。不需要去求解整个系统， n 一定是 $b-a$ 的约数，所以对一个式子求出所有约数，然后代入其它式子检验即可。

【时空复杂度】

时间复杂度: $O(n + \sqrt{k})$

空间复杂度: $O(Tn\sqrt{k})$

【题目编号】

2006 I Degrees of Separation

【题目名称】

Degrees of Separation

【题目大意】

给出一个 P 个点， R 条边的无向图，判断无向图是否连通。若无向图连通，则输出所有两个点之间最短路的最大值。

多组数据， $P \leq 50$ 。

【题目关键字】

最短路，floyd

【算法讨论】

因为数据范围很小，所以直接用 floyd 计算任两点之间最短路，然后取最大值即可。连通性只要在 floyd 之后判断任两点之间最短路的最大值是否为正无穷即可。

【时空复杂度】

时间复杂度: $O(P^3)$

空间复杂度: $O(P^2)$

【题目编号】

2006 J Routing

【题目名称】

Routing

【题目大意】

给出一个 n 个点， m 条边的有向图。要求从找到一条经过节点 1 和节点 2 的环路（允许经过重点或重边），使得环上经过的不同节点尽量少。

$n \leq 100, m \leq 1000$ ，多组数据，数据组数 ≤ 5 。

【题目关键字】

动态规划,最短路

【算法讨论】

首先用 floyd 算法计算出任意一对节点之间的最短路，记为 $f[i][j]$ 。

最优解中，节点 1 到节点 2 的路径和从节点 2 到节点 1 的路径重复的部分把这条路径分成了若干个环，且重复部分的点的遍历顺序是一样的。把节点 2 到节点 1 的路径看成反图中节点 1 到节点 2 的路径，问题转化成在两个图中选取两条节点 1 到节点 2 的路径，经过的不同点数尽量少。

令 $T[i][j]$ 为从节点 1 沿着原图的边走到节点 i ，沿着反图的路径走到节点 j 经过的最少不

同点数。转移分为三种情况：

1，原图中存在有向边(i,k)，若 $j=k$ ，用 $T[i][j]$ 向 $T[k][j]$ 转移，否则用 $T[i][j]+1$ 向 $T[k][j]$ 转移。

2，反图中存在有向边(j,k)，若 $i=k$ ，用 $T[i][j]$ 向 $f[i][k]$ 转移，否则用 $T[i][j]+1$ 向 $f[i][k]$ 转移。

3， i 不等于 j ，且原图中存在路径一条从 i 到 j 的路径。此时两条路径可以共用从 i 到 j 的这一部分，用 $T[i][j]+f[i][j]-1$ 向 $T[j][i]$ 转移。

$T[i][j]$ 难以明确的划分阶段，为保证复杂度，可以用最短路算法计算出所有的 $T[i][j]$ 值。最后答案即为 $T[2][2]$ 。

因为最后答案的范围不超过 n ，用 `dijkstra` 算法的话，可以用一个桶代替堆来完成取最小值的操作，这样整个算法的时间复杂度即为 $O(n^3)$ ，可以通过本题。

【时空复杂度】

时间复杂度： $O(n^3)$

空间复杂度： $O(n^2)$

【题目编号】

2007 A Consanguine Calculations

【题目名称】

Consanguine Calculations

【题目大意】

每个人的血型都有两个标记，这两个标记被叫做 ABO 血型系统的等位基因，每个标记都代表了 ABO 三个字母之一，因此我们有 6 种可能的基因组合，而每一种组合都表示了一个特定的 ABO 血型。

组合	血型
AA	A
AB	AB
AO	A
BO	B
BB	B
OO	O

与之相对应的，每个人同样有两个 Rh 血型系统的等位基因，而这两个等位基因的标记是+或-。某个人如果是 Rh+血型则至少含有一个+基因，而如果是 Rh-血型则一定含有两个-

基因。

所以一个人的血型是 ABO 血型系统和 Rh 血型系统的组合，血型的书写方式是先写 ABO 血型后写 Rh 血型。例如 A+、AB-、O-。

血型是可以继承的：父母分别贡献一个 ABO 血型基因和一个 Rh 血型基因（随机从两个中选择）给他们的孩子。因此父母的 2 个 ABO 血型基因和 Rh 血型基因决定了孩子的血型。比如父母都是 A-，则他们的孩子会是 A-或 O-；而父母是 A+和 B+时，孩子可以是任意血型。

在本题中，你会得到父母和孩子三者中两者的血型，你需要确定剩下的那个人所有可能的血型。多组数据，数据组数 ≤ 10

【题目关键字】

枚举，模拟

【算法讨论】

算法很明显，枚举剩下的人的血型组合，然后判断是否合法即可。很明显，ABO 血型和 Rh 血型互不影响，我们可以分开判断。

首先是 ABO 血型的判断。

1. 对于 O 型血的后代，要求父母都拥有 O 型血基因。
2. 对于 A 型血后代，要求父母一人有 A 型血基因。
3. 对于 B 型血后代，要求父母一人有 B 型血基因。
4. 对于 AB 型血，要求父母一人有 A 型血基因，另一人有 B 型血基因

接下来是 Rh 血型判断，比较简单。

1. 对于 Rh+型血后代，要求父母一人有 Rh+型血基因。
2. 对于 Rh-型血后代，要求父母两人有 Rh-型血基因。因为两种 Rh 血型都可能有 Rh-型血基因，所以这一条事实上不用判断。

然后将枚举过程中所有合法方案输出即可。不同的血型共有 8 种，因此时间复杂度是常数级的。

【时空复杂度】

时间复杂度：O(1)

空间复杂度：O(1)

【题目编号】

2007 E Collecting Luggage

【题目名称】

Collecting Luggage

【题目大意】

已知一简单 N ($3 \leq N \leq 100$) 边形的各顶点坐标, 行李箱从第一个顶点按逆时针顺序以速度 V_l 沿多边形边界移动, 人从点 (px, py) 出发, 速度为 V_p ($0 < V_l < V_p \leq 10000$, 单位为米每分), 所有坐标均为绝对值不大于 10000 的整数 (单位为米), 求人最快取得行李的时间 (四舍五入精确到秒)。

【试题关键字】

最短路算法, 计算几何

【算法分析】

因为行李的速度 V_l 小于人的速度 V_p , 所以答案具有二分性。

首先第一步二分答案。答案 t 确定后, 行李 t 时刻所在位置也就确定了, 接下来只要求出点 (px, py) 到 t 时刻所在位置的最短路, 若最短路长度不超过 $v_p * t$, 则答案合法。

接下来的问题如何求人到行李的最短路。显然, 最短路一定由线段组成, 且每一段线段的两个端点都是 (px, py) , 行李所在位置, 多边形端点三者之一。枚举这 $n+2$ 个点组成的每一

对线段，对于每一条不经过多边形内部的线段，我们都在两个端点间连一条对应长度的边，然后求一边最短路即可。

对于判断线段是否经过多边形内部，分为两种情况讨论。一种是线段一部分在多边形内，一部分在多边形外，另一种是线段完全被多边形包含。对于第一种情况，线段一定与多边形边界相交，对于第二种情况，线段的中点一定被包含在多边形内。这里要用到判线段相交和判点在简单多边形的计算几何基础知识。时间复杂度均为 $O(n)$ 。每次二分后，有 $O(n^2)$ 对线段要进行判断，每次重新构图的时间复杂度为 $O(n^3)$ ，最短路可以用 `dijkstra` 实现，每次求最短路的时间复杂度为 $O(n^2)$ 。二分的次数是 $\log(t)$ 别的， t 为答案范围，因此总的时间复杂度为 $O(n^3 \log t)$ ，可以解决问题。

【时空复杂度】

时间复杂度: $O(n^3 \log t)$

空间复杂度: $O(n^2)$

【题目编号】

2007 F Marble Game

【题目名称】

Marble Game

【题目大意】

小球游戏使用 M 个小球玩的游戏。一个正方形木板被分割为 $N \times N$ 单位的小正方形格子，其中 M 个小正方形格子被标识为洞，小球和洞用 1 到 M 的数字标记。小球游戏的目标就是把每个小球滚进与之标有相同编号的洞里。

游戏使用的模板可能会有些分隔板。每一面分隔板的长度都是一个单位，将两个相邻的单位小正方形格子隔开当且仅当两个小正方形格子有公共边时，他们是相邻的。

在游戏开始的时候，所有的小球都被放置在木板上，每个小球在各不相同的小正方形格子中。一次“移动”指的是轻轻提起木板的一边，然后木板上所有的小球都会朝着相反方向往下滚。直到碰到了分隔板或者掉进一个空的洞里，或者下一个小正方形格子已经被另一个小球所占据，才停止滚动。小球滚动的时候遵循下面的规则：

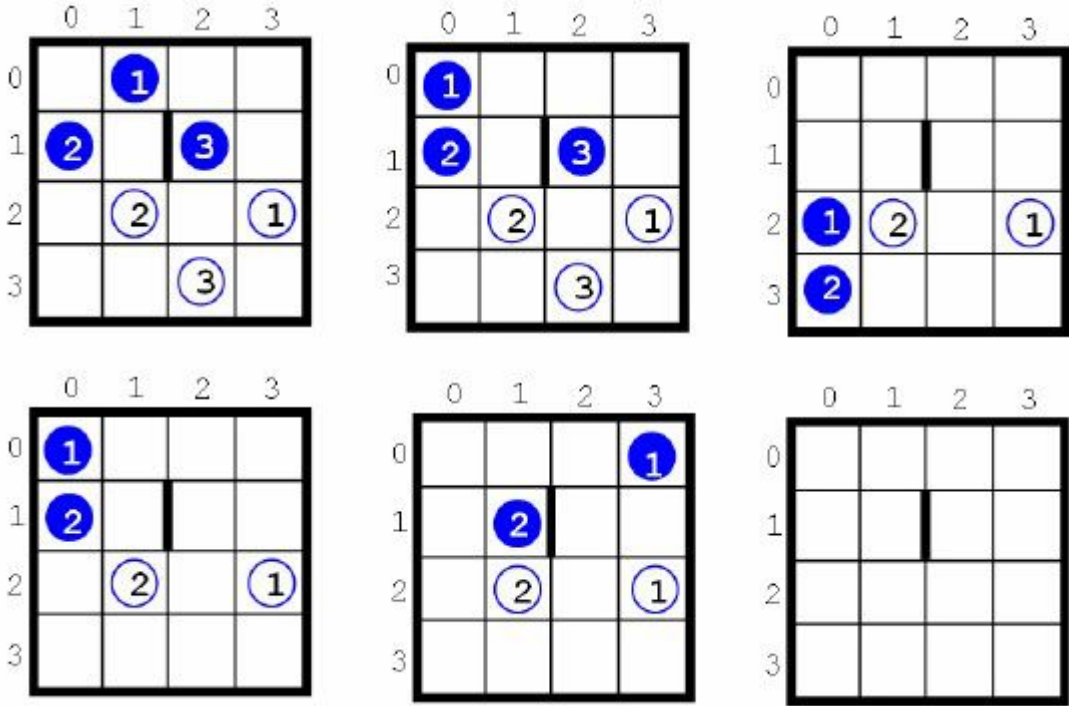
1. 小球不能越过分隔板、其他小球或空洞。
2. 小球不能离开木板（木板的边缘是分隔板）。
3. 一个单位小正方形格子在任何时候都最多只能容纳一个小球。

4.当一个小球移动到一个游动的小正方形格子时，这个小球会掉进洞里。然后这个洞就会被填充了，随后其他小球就可以从这个洞上面滚过。在洞里的小球永远不能离开洞。

5.当所有的小球都掉进相应标号的洞里，游戏就结束。

如图所示，是在一个 4*4 的模板上滚 3 个小球的一个解决方案，木板上有 3 个洞和一个分隔板。这个解决方案有 5 次移动：右边，上边，下边，左边，上边。

要编写的程序需要确定，如果存在解决方案的话，把所有的小球滚进对应的洞里的最小移动次数。



棋盘边长 $2 \leq n \leq 4$ 。多组数据

【试题关键字】

广度优先搜索，哈希

【算法分析】

因为求最少步数，所以显然可以使用广度优先搜索。

但直接搜索显然时间和空间都不够，不过因为有大量重复状态可以使用 hash 去重。

因为空洞和隔板位置不变，所以对于一个状态，只需要记录全部小球所在的位置即可。

因为一个小球对应一个洞，所以最多有 8 个小球（否则必然无解）。一个小球的位置有 $n \times n$ 种状态，还有一种掉进洞里的状态。因此可以用一个 8 位的 17 进制数存下一个状态。

因为状态数量是指数级别的，结束的步数可能很大。因此时空复杂度不好估计。但在清澄和 uva 上都能很快出解。

【时空复杂度】

时间复杂度: $O(?)$

空间复杂度: $O(?)$

【题目编号】

2007 G Network

【题目名称】

Network

【题目大意】

给出 n 条信息， n 条信息被分成 m 个信息包。每个信息包都是其中某一条信息中连续的一段。现在已知你接收得到信息包的顺序，对于每条接收的信息包，你可以选择放入缓存或直接输出，任意时刻你可以输出缓存中的任意一个信息包。但要求来自同一条信息的信息包顺序并连续地输出（不要求同时输出）。问缓存区最小的大小。

$n \leq 5, m \leq 1000$ ，多组数据。

【试题关键字】

搜索，模拟

【算法分析】

因为要求来自同一条信息的信息包顺序并连续地输出，所以枚举每条信息的输出顺序，那么每个信息包的输出顺序也就唯一确定了。然后按题意模拟，依次处理每个信息包，如果下一个要输出的信息包在缓存中或将被输入则输出这个包，否则把这个包放入缓存。这个过程中放在缓存信息包的总大小的最大值即为答案。时间复杂度为 $O(n!m)$

【时空复杂度】

时间复杂度: $O(n!m)$

空间复杂度: $O(nm)$

【题目编号】

2007 I Water Tanks

【题目名称】

Water Tanks

【题目大意】

给出 n 个水箱排成一行, 除第一个水箱外其余水箱都不与大气连通, 每个水箱底面积均为 1。相邻两个水箱间有一条水平的管道连接, 管道的高度递增 (就是说, 连接水箱 i 和水箱 $i+1$ 的管道高于连接水箱 i 和水箱 $i-1$ 的管道), 管道可以通过空气和水, 管道尺寸可以忽略。

下列物理原理会帮助你解决这些问题 (有些是根据题目意图允许的近似):

- 水会向下流。
- 在与外界联通的区域, 气压等于一个大气压。
- 空气可以被压缩 (一定量的空气所占体积取决于压强)。水不可以被压缩 (一定量的水所占的体积是恒定的, 与压强无关)。
- 在一个封闭空间中, 气压处处相同。如果一个封闭空间的体积缩小, 体积和气压的乘积恒定。举个例子, 一个封闭空间有一个初始体积 V_1 和气压 P_1 。如果体积变成 V_2 , 那么新的气压 P_2 满足 $P_1V_1=P_2V_2$ 。

- 在一个空气层下的水体，在水面以下 D 米的水压等于在表面气压加上 $0.097 \cdot D$ 个大气压。无论空间是打开的或是封闭的，这都是正确的。

- 在一个连通的水体中（举个例子，当两个或更多水在水面下每管子连通），水的压强在每个高度上都相同。

给出 n 个水箱的高度和管道的高度。现在向水箱 1 中灌水，问水面到达水箱 1 顶部前一共灌了多少单位的水。 $2 \leq n \leq 10$

【题目关键字】

模拟

【算法讨论】

如果水在第 i 个水箱没有达到连接第 i 个水箱和第 $i+1$ 个水箱的管道高度的话，那么第 $i+1 \sim n$ 个水箱都不会有水。因为连通的水体处处水压相等，所以第 i 个水箱底部的水压等于第一个水箱底部的水压。同时根据当前气体的体积和气压，可以列出关于底部水压与该水箱水面高度的二次方程，解之可得该水箱的水面高度。

因为水管高度递增，若第 i 个水箱不是水流停止的水箱，那么当水面高度超过连接第 i 个和第 $i+1$ 个水箱的管道高度后，留在第 i 个水箱的气体质量就不再变化。此时可以列出关于底部水压与该水箱水面高度的二次方程，解之可得该水箱的水面高度。

那么按照 $2 \sim n$ 个顺序枚举每个水箱。同时记录当前与第 n 个水箱连通的大气部分的体积和气压，以及当前灌入水的总体积。然后对于第 i 个水箱，首先假设水流并未停滞，将水面管道与第 $i+1$ 个水箱连通的管道高度并计算地面水压。若超过水箱 1 的底面水压，那么水流在这个水箱停滞。直接算出该水箱的水面高度然后计入答案并退出即可。否则现将水升高至与 $i+1$ 个水箱连通的管道高度，更新录当前与第 n 个水箱连通的大气部分的体积和气压，然后计算该水箱的水面高度并计入答案，然后继续处理下一个水箱即可。

处理的过程中需要解二次方程，复杂度 $O(1)$ ，最后总的复杂度为 $O(n)$ 。

【时空复杂度】

时间复杂度： $O(n)$

空间复杂度： $O(n)$

【题目编号】

2007 J Tunnels

【题目名称】

Tunnels

【题目大意】

给出一个由 $0 \sim n$ 这 $n+1$ 个节点和 m 条边组成的无向图。有一个人从节点 1 出发，试图抵达节点 0。当人在边上移动时，你不能做任何操作。但当人到达某个节点，你可以删除图上任意数量的任意边。问最少删除几条边能使这个人不能到达节点 0。 $n \leq 50, m \leq 1000$

【试题关键字】

网络流，最短路

【算法分析】

设 $\text{mincut}(i,j)$ 为节点 i,j 之间的最小割大小。设 $L[i]$ 为从节点 i 出发的情况下，最少要删掉多少条边，才能使人不能到达节点 0。假设我们已经算出除了 $L[1]$ 之外的所有 $L[i]$ ，那么 $L[1]$ 的取值只有两种情况。一种是在节点 1 把节点 1,0 之间最小割上的边删除，代价为 $\text{mincut}(0,1)$ 。另一种为枚举一个节点 i ，把所有满足 $L[j] \leq L[i]$ 的节点 j 删除后形成一个新图，先新图上删除节点 1 和 0 最小割上的边，然后再删除原图节点 i 和节点 0 之间最小割上的边，这一种代价为 $L[i] + \min(i,T)$ 。第一种方案很好理解，第二种方案的本质是先删除若干条边，

迫使人到达一个较容易被阻断的地方，然后切断他和节点 0 的通路。

接下来为求 $L[i]$ 的顺序问题。首先从每个点出发，求一边最小割，作为初始的 $L[i]$ 值。显然最小割 $L[i]$ 值不可能被改变，即为 M 。然后将所有满足 $L[i]=M$ 的节点从图中删除，对于剩下的节点 j ，用 $M+\text{mincut}(0,j)$ 的值去更新它的 $L[j]$ 值即可。不停重复上述过程直到整个图被删除为止。因为每次至少删掉一个节点，最多删 n 次，每次做 n 边网络流，总的时间复杂度为 $O(n^2 \times \text{网络的时间复杂度})$ 。因为题目 n 较小，网络流上界较松，可以通过本题。

【时空复杂度】

时间复杂度： $O(n^6)$

空间复杂度： $O(n^2)$

【题目编号】

2008 A Air Conditioning Machinery

【题目名称】

Air Conditioning Machinery

【题目大意】

你必须在指定的空间内装一个管道。空间是一个 $x_{\max} \times y_{\max} \times z_{\max}$ 的长方体长方体，所有的边的都是单位长度的整倍数，可以想象为一个空间堆满了单位正方体。每个弯管占用恰好 4 个单位的正方体，如下图 1 所示。两个弯管不能重合在同一个单位正方体上。每个弯管只有 2 个口，它们在图形 1 中以灰色显示。你可以把 2 个弯管接成一根长的管子，但是它们不得超过给定的空间。图 2 表示了其中一种对接方式。你的任务是接通入口和出口。入口和出口在给定空间的外表面上，和单位正方体对齐，如图 3 所示。输出接通所需的最少弯管数量。如果答案大于 6 或无解，输出 Impossible。 $1 \leq x_{\max}, y_{\max}, z_{\max} \leq 20$

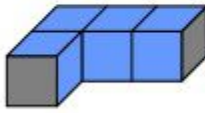


Figure 1

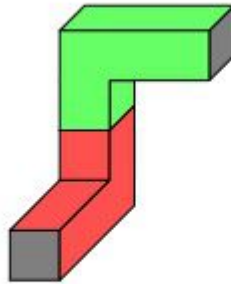


Figure 2

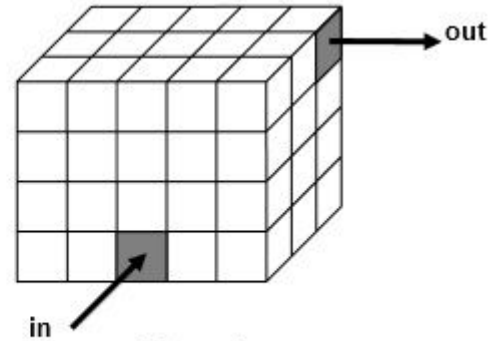


Figure 3

【题目关键字】

搜索，枚举

【算法讨论】

做法很明显。在接口确定的情况下，能接在一个面上的弯管摆放方式最多只有 8 种。因此我们枚举 6 个弯管每个的摆放方式，看是否能接通即可，时间复杂度为 $O(8^6)$ 。

【时空复杂度】

时间复杂度: $O(8^6)$

空间复杂度: $O(x_{\max} * y_{\max} * z_{\max})$

【题目编号】

2008 B Always an Integer

【题目名称】

Always an Integer

【题目大意】

给定自变量为 n 的多项式，问当 n 取任意正整数时这个多项式的值是否恒为整数。

多项式的格式一定是 $(P)/D$ ，其中 P 是一个整系数多项式， D 是一个正整数分母。

P 是由若干单项式 Cn^E 组成的，单项式的系数 C 和指数 E 满足下列条件：

1. E 是一个整数，且 $0 \leq E \leq 100$ 。如果 E 为 0，则 Cn^E 会用 C 表示；如果 E 为 1，则 Cn^E 会用 Cn 表示，除非 C 为 1 或 -1，会用 n 或 $-n$ 表示。
2. C 是一个整数，如果 C 为 1 或 -1 且 E 不为 0 和 1， Cn^E 会用 n^E 或 $-n^E$ 表示。
3. 如果 C 的值非负， Cn^E 前面会用 + 号连接。

4.多项式降幂排列。

5.C 和 D 都可以用 32 位有符号整数表示。

多组数据。

【题目关键字】

递归，数学

【算法讨论】

证明 n 为任意正整数时，多项式 $(P)/D$ 恒为整数，等价于证明 n 为任意正整数时，多项式 P 对 D 取模等于 0。

证明这一点，可以分成两个部分。令 $F(x)$ 为 $n=x$ 时多项式 P 的值。1，当 $n=0$ 是， $P \bmod D=0$ 。2，当 n 为任意正整数时， $(F(n+1)-F(n)) \bmod D=0$ 。

第一部分可以 $O(1)$ 证明，第二部分等价于证明一个次数-1 的多项式对 D 取模结果恒为 0。是一个规模更小的子问题，可以递归结果。时间复杂度为 $O(E^2)$ ， E 为多项式次数。

【时空复杂度】

时间复杂度： $O(E^2)$, E 为多项式次数。

空间复杂度： $O(E^2)$

【题目编号】

2008 D The Hare and the Hounds

【题目名称】

The Hare and the Hounds

【题目大意】

猎狗和兔子在进行公路拉力赛。当猎狗和兔子进入十字路口时，它们会按照主要道路规则选择道路。即它们会选择与原来方向所成角最小的一条路线进入，如果有多条，它们会选择右边的那条。

在一些路口，兔子可能违反的主要道路规则（但不会去来的那条路），随机选一条路走，这些十字路口被兔子标为选择点。猎狗遇到选择点后，必须尝试每一条道路，直到到达一个确认标记。下列都表示一个不正确的路线选择：

1.猎狗走了 $maxdis$ 后还没有碰到确认点。

2.猎狗在碰到确认点前遇到了死胡同（只有一条路的十字路口）

3.猎狗在碰到确认点前遇到了其他的选择点（兔子总是会在到达下一个选择点的路上放确认点，且兔子不会再回到之前那个选择点）

在遇到一个不正确的路线后，猎狗必须沿原路返回，然后选其他路，即使中间他到达了终点，他也会忽略。

当猎狗在选择点时，选择进入的路口的规则和之前不一样，第一道路使用主要道路规则。但是如果猎狗必须返回这个选择点时（例如遇到一个不正确的路线），猎狗会从回来的方向使用主要道路规则选路（忽略所有他已经尝试过的路与一开始来这个点时走的路），这个过程会不断的循环直到他找到了正确的路。这种不同的主要道路规则的方式只会用在选择点上，对于本问题，猎狗不会记得它走向一条路的结果，即使他在探索选择点的时候不断的经过这条路。

给出 n_{cp} 个选择点， n_{road} 条道路， n_{cm} 个确认点， $maxdis$ ， $startisect$ （出发十字路口的标号）， $endisect$ （结束十字路口的标号）， $startdir$ （初始方向），十字路口从 1-100 编号。要求出兔子走过的路径，以及兔子和猎狗各自走过路径的总长度， $n_{cp} \leq 100, n_{road} \leq 150, n_{cm} \leq 100, maxdis \leq 2000$ ，路口编号 1~100。

【题目关键字】

模拟

【算法讨论】

按照题意模拟即可。记录当前猎狗所在路口 r 和面向方向 dir ，接下来按照当前路口是否选择点分类讨论。

若当前路口不是选择点，那么找到与方向 dir 所成角最小的一条路线进入。

如果当前路口是选择点，那么首先选择一条方向 dir 所成角度最小的路线进入，进入后按照主要道路规则直到找到确认点或发现路线不正确未知。若找到确认点，则进入。否则将 dir 设置为这条边回来的方向，然后重复这个过程。

不停重复以上两个过程，直到走到终点为止。过程中分别记录兔子和猎狗的路径并输出即可。

【时空复杂度】

时间复杂度： $O(n_{road} * n_{cp} * n_{cp})$

空间复杂度： $O(n_{road})$

【题目编号】

2008 E Huffman Codes

【题目名称】

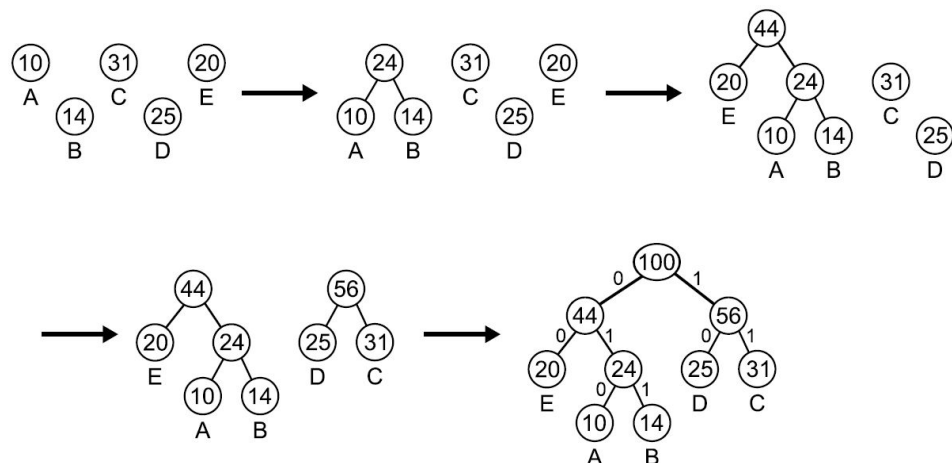
Huffman Codes

【题目大意】

给出一棵有 n 个叶节点的 **huffman tree**。每个叶节点的权值和为 100，且每个节点的权值均为不超过 100 的正整数。问叶节点有多少种分配权值的方案，能构造出给定形态的 **huffman tree**。

huffman tree 的构造过程如下：一开始建立一片森林，其中每棵树都只有一个节点。每次建立一个新的根节点，选择两棵权值最小的树（若相同则任取一个），将它们分别作为新的根节点的左右儿子，将它们的根节点的权值和作为新的根节点的权值，然后将这两棵树替换为新树。下图显示了一个例子，这里有 A、B、C、D、E 五个节点，权值分别为 10、14、

31、25 和 20。



多组数据，数据组数 ≤ 20 , $n \leq 20$ 。

【题目关键字】

搜索，剪枝

【算法讨论】

因为 n 较小，而且根据小规模数据的暴力可以发现答案不会很多，因此可以采用搜索。

因为有大量无用节点，因此搜索要进行剪枝和优化才能通过本题。

1，从根向叶子搜索，每次搜索一个节点左右子树的和。

2，每次搜索当前未分配的、权值最大的子树分配给它的左右子树的权值。

3，因为要保证树的形态，每次搜索分配的较大子树的权值不会大于之前任何一次分配分配给较小子树的权值（主要优化）。

加上这三条优化，就可以轻松通过所有测试数据了。

【时空复杂度】

时间复杂度： $O(T \cdot C(99, n-1))$ ， T 为数据组数

空间复杂度： $O(n)$

【题目编号】

2008 F Glenbow Museum

【题目名称】

Glenbow Museum

【题目大意】

从一个所有边都平行于坐标系的多边形的任一顶点出发，逆时针遍历，记录每次经过的顶点处的转角，90 度为 R，270 度为 O。组成的字符串叫做 **angle string**。求指定长度 L 的 **angle string** 中，能表示至少一个星形多边形的串个数。 $L \leq 1000$

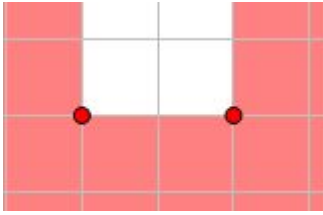
【题目关键字】

组合数

【算法讨论】

当 L 为奇数时，答案显然为 0。

由于多边形外交和为 360 度，得 R 的个数比 O 的个数恰好多 4 个。
如图，若字符串中有两个连续的 OO，那么显然无解。



【时空复杂度】

时间复杂度: $O(L)$

空间复杂度: $O(L)$

【题目编号】

2008 G Net Loss

【题目名称】

Net Loss

【题目大意】

给出一个多项式 $p(x)$ 。令 $g(x)$ 为 $y = a_1x + a_0$ 以及 $y = b_1x + b_0$ 两条线段组成函数。两线段交于一点，其横坐标 c 介于 -1 和 +1。也就是令 $g(x) = \begin{cases} a_0x + a_1 & x \leq c \\ b_0x + b_1 & x > c \end{cases}$ 。定义函数

$d(p, g) = \int_{-1}^1 (p(x) - g(x))^2$ 。给出 $p(x), c$ ，要求 a_0, a_1, b_0, b_1 ，使得函数 $d(p, g)$ 最小。 $p(x)$ 次数不超过 10

【题目关键字】

数学，二次函数，定积分

【算法讨论】

设两条线段交与点(c,d)，可以 $a_0=d-a_1c$ 。

函数 $d(p, g) = \int_{-1}^1 (p(x) - g(x))^2 = \int_{-1}^c (p(x) - g(x))^2 + \int_c^1 (p(x) - g(x))^2$ ，将 $\int_{-1}^c (p(x) - g(x))^2$ 展开：

$$\begin{aligned} & \int_{-1}^c (p(x) - g(x))^2 \\ &= \int_{-1}^c (p(x) - (a_1x + d - a_1c))^2 \\ &= \int_{-1}^c [p(x)^2 - 2p(x)(a_1x + d - a_1c) + (a_1x + d - a_1c)^2] \\ &= \int_{-1}^c p(x)^2 - 2a_1 \int_{-1}^c (x \bullet p(x)) - 2d \int_{-1}^c p(x) + 2a_1 \int_{-1}^c c - 2a_1 \int_{-1}^c (p(x) \bullet c) \\ &+ a_1^2 \int_{-1}^c x^2 + d^2 \int_{-1}^c 1 + 2a_1d \int_{-1}^c 1 - 2a_1^2 \int_{-1}^c xc - 2da_1 \int_{-1}^c c + a_1^2 \int_{-1}^c c^2 \end{aligned}$$

将 d 看成常数，那么可以将原函数整理成一个关于 a_0 的二次函数。整理后可以发现这个二次函数的最小值只与 d 的取值有关。同上，我们可以将 $\int_c^1 (p(x) - g(x))^2$ 展开，将 d 看

做常数，将这个函数整理成一个关于 b_0 的二次函数，同样它的最小值只与 d 的取值相关。将两个最小值和展开，可以得到一个关于 d 的二次函数，那么就可以在 $O(1)$ 的时间内算出 d 的取值。算出 d 的取值后就可以反过来退出 a_0, b_0, a_1, b_1 的取值。因为在这个过程中变量的系数要用定积分计算，因此总的时间复杂度为 $O(n^2)$ 。

【时空复杂度】

时间复杂度： $O(n^2)$

空间复杂度： $O(n)$

【题目编号】

2008 H Painter

【题目名称】

Painter

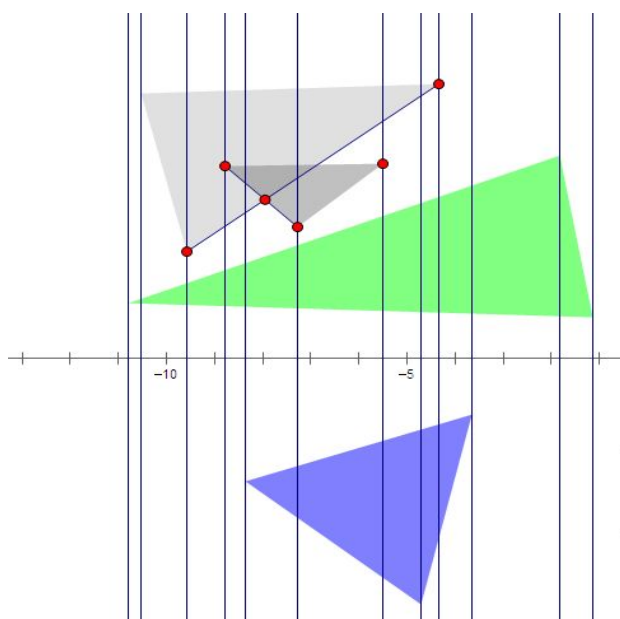
【题目大意】

平面上给出 n 个三角形。问三角形是否两两不相交。如果存在两个三角形的边至少有一个交点，输出 ERROR。否则输出三角形中最多的嵌套层数。 $n \leq 100000$

【题目关键字】

计算几何、扫描线、平衡树

【算法讨论】



如图，用垂直于 X 轴的扫描线，按 Y 值从小到大依次扫描整个平面。若三角形没有相交，两条线段在不同两条扫描线上交点的 Y 的相对大小关系是一定的。否则一定会有两条大小关系相邻的线段在跨过某一个区域后大小关系改变。这种改变一定会先从 Y 坐标大小相邻的两个节点开始。因此我们用扫描线扫描整个平面的过程中，用平衡树以与扫描线交点的 Y 坐标为关键字，维护当前与扫描线有交点的线段。只有在线段插入，删除的时候，线段的相邻关系才会改变。过程中维护新增加的相邻的线段是否有交点即可。

接下来是求嵌套层数。把三角形之间直接包含的关系看做父子关系。一个三角形和一条扫描线会有两个交点。按照交点 Y 坐标大小分，Y 坐标较小的为左括号，较大的为右括号。然后在插入一个三角形，找到它的左括号在平衡树上的前驱的线段。若该线段是左括号，那么该线段对应的三角形是插入的三角形的父亲，否则就是插入的三角形的兄弟。然后就可以求出插入三角形的父亲了。扫描结束后若三角形两两没有交点，那么就可以根据三角形之间的父子关系求出每个三角形的嵌套层数了。

每个三角形都有三条边，每条边插入一次，删除一次，平衡树单次插入/删除的复杂度是 $O(\log n)$ 的，因此总的时间复杂度为 $O(n \log n)$ ，可以解决本题。实现的过程中有一个小问题，三角形的线段可能与扫描线平行，不好处理。这时我们可以通过将所有三角形随机旋转一个角度来解决问题。

【时空复杂度】

时间复杂度： $O(n \log n)$

空间复杂度： $O(n)$

【题目编号】

2008 I Password Suspects

【题目名称】

Password Suspects

【题目大意】

给出 m 个字符串，要求一个长度为 n 个字符串，使得这 m 个串都是这个字符串的子串。要求输出方案数，如果方案数不超过 42，输出所有方案。

$n \leq 25, m \leq 10$, 字符串长度 ≤ 10

【题目关键字】

动态规划，状态压缩，AC 自动机

【算法讨论】

首先，对输入的 m 个字符串建立一个 AC 自动机。令 $f(i,j,S)$ 表示，字符串的前 i 位位于 AC 自动机的节点 j 上，已经出现过的字符串集合为 S ，字符串剩下 $n-i$ 位的方案数。

转移枚举字符串下一位的字符 p 即可转移，总的时间复杂度为 $O(n*L*2^m)$ ，可以通过本题。

【时空复杂度】

时间复杂度： $O(n*L*2^m)$ ， L 为所有字符串总长度。

空间复杂度： $O(n*L*2^m)$

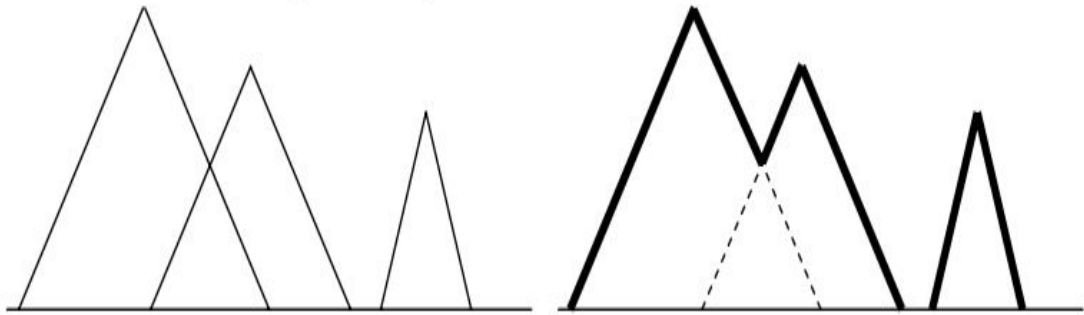
【题目编号】

2008 J The Sky is the Limit

【题目名称】

The Sky is the Limit

【题目大意】

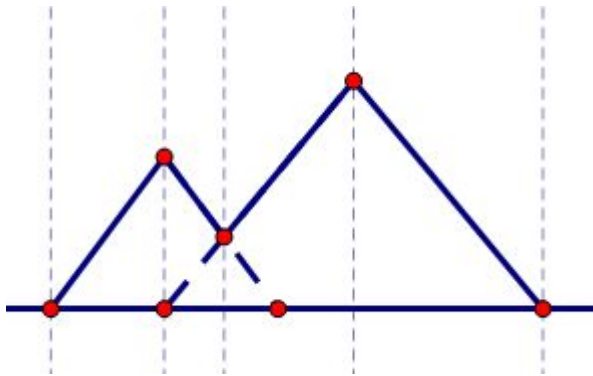


如图，给出 n 个底边为 X 轴，且有一个点位于 X 轴上方的等腰三角形，求这些三角形组成的图形的周长位于 X 轴上方的部分的总长度。

【题目关键字】

计算几何

【算法讨论】



如图，过每个三角形的顶点和三角形边的交点做关于 x 轴的垂线，然后对于相邻的两条垂线之间，找到跨过这两条垂线的线段中最高的一条，并计入答案即可。

【时空复杂度】

时间复杂度: $O(n^3)$

空间复杂度: $O(n)$

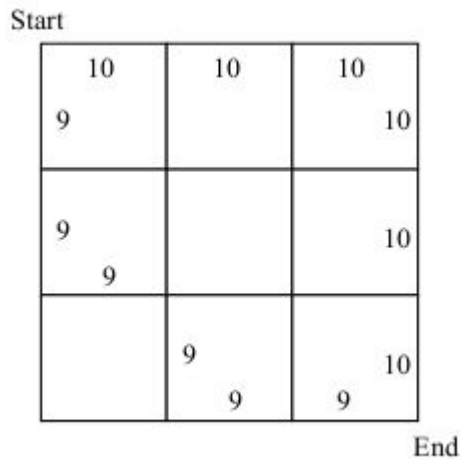
【题目编号】

2008 K Steam Roller

【题目名称】

Steam Roller

【题目大意】



如图，给出一个 $R \times C$ 的网格图，其中某些边不可通过。

要求从一点到另一点的最短路径。

给出通过网格上每条可以通过的边的通过时间。通过一条边之前或之后改变运动方向，通过时间即为原本时间的两倍。同时路径上的第一条边和最后一条边也要花费原本时间的两倍来通过。

$R, C \leq 100$ ，多组数据，数组组数 ≤ 20

【题目关键字】

最短路

【算法讨论】

令 $f[i][j][k]$, $0 \leq k \leq 3$, 表示到达节点 (i, j) 之前的运动方向 k 的最短时间，接下来可以用 1 倍时间或 2 倍时间从 (i, j) 的 k 方向离开。 $f[i][j][4]$ 表示从上一个节点花费 2 倍时间到达 (i, j) 的最短时间，接下来可以从任何方向离开，但必须花费 2 倍时间。直接使用 *dijkstra* 求出最短路径即可。时间复杂度为 $O(T \cdot n^2 \log n)$ 。

【时空复杂度】

时间复杂度: $O(T \cdot n^2 \log n)$

空间复杂度: $O(n^2)$

【题目编号】

2009 A A Careful Approach

【题目名称】

A Careful Approach

【题目大意】

给出 n 个事件，第 i 个事件只能在 $[a_i, b_i]$ 内的时间发生。事件的发生是在瞬间完成的。问任意两个事件之间的时间间隔的最小值最大是多少。 $n \leq 8, a_i, b_i \leq 1440$

【题目关键字】

二分，搜索

【算法讨论】

对于这类最大化最小值的问题，一个很自然的想法是二分答案。

假设当前二分的答案为 t ，那么任意两个时间之间发生的间隔不小于 t 。我们可以枚举事件发生的先后顺序。在顺序确定的情况下，每个事件越早发生，对后面的事件影响越有利。所以我们可以用贪心判断是否有解。也就是让每个事件在上一个事件发生时间 $+t$ 以后，第一个被区间 $[a_i, b_i]$ 包含的时刻发生。贪心复杂度为 $O(n)$ ，发生顺序的方案有 $n!$ 种，因此总的时间复杂度为 $O(n * n! \log 1440)$

【时空复杂度】

时间复杂度： $O(n * n! \log 1440)$

空间复杂度： $O(n)$

【题目编号】

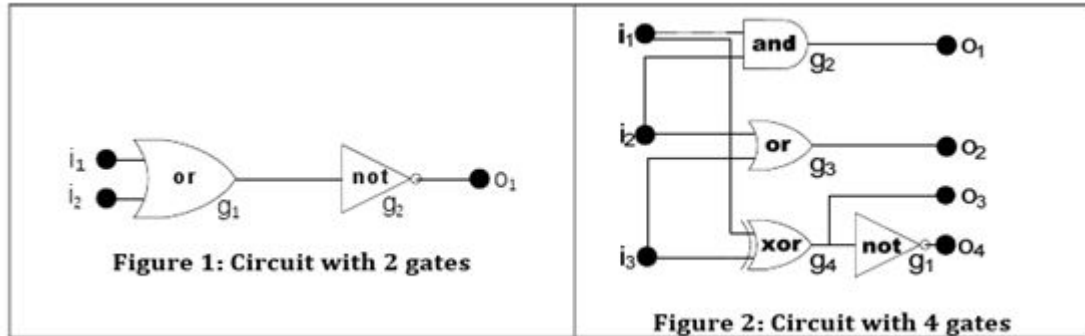
2009 B My Bad

【题目名称】

My Bad

【题目大意】

一个逻辑电路将其输入通过不同的门映射到输出，在电路中没有回路。输入和输出是一个逻辑值的有序集合，逻辑值被表示为 1 和 0。我们所考虑的电路由与门（and gate，只有在两个输入都是 1 的时候，输出才为 1）、或门（or gate，只要两个输入中有一个是 1，输出就是 1）、异或门（exclusive or (xor) gate，在两个输入中仅有一个是 1，输出才是 1）和非门（not gate，单值输入，输出是输入的补）组成。下图给出两个电路。



不幸的是，在实际中，门有时会出故障。虽然故障会以多种不同的方式发生，但本题将门会出现的故障限于如下三种形式之一：

- 1) 总是与正确的输出相反；
- 2) 总是产生 0；
- 3) 总是产生 1；

在本题给出的电路中，最多只有一个门出故障。

请编写一个程序，对一个电路进行分析，对多组输入和输出进行实验，看电路运行是正确的还是不正确的。如果至少有一组输入产生了错误的输出，程序要确定唯一的出故障的门，以及这个门出故障的方式。但这也可能是无法判断的。

电路 n 个输入， G 个门， U 个输出。给出每个门的种类和门的输入。给出 B 组测试用的输入输出，要求判断 B 组输出是否均为正确。如果不正确，输出唯一的出故障的门，以及这个门出故障的方式。如果出故障的门或出故障的方式不能唯一确定，输出“Unable to totally classify the failure”。

$n \leq 8, G, U \leq 19$ ，多组数据

【题目关键字】

模拟

【算法讨论】

因为数据范围较小，所以直接枚举出故障的门和出故障的方式，然后按题意计算出每组输入对应的输出，并判断是否合法即可。

【时空复杂度】

时间复杂度： $O(n * B * (G + U))$

空间复杂度： $O(B * (G + U))$

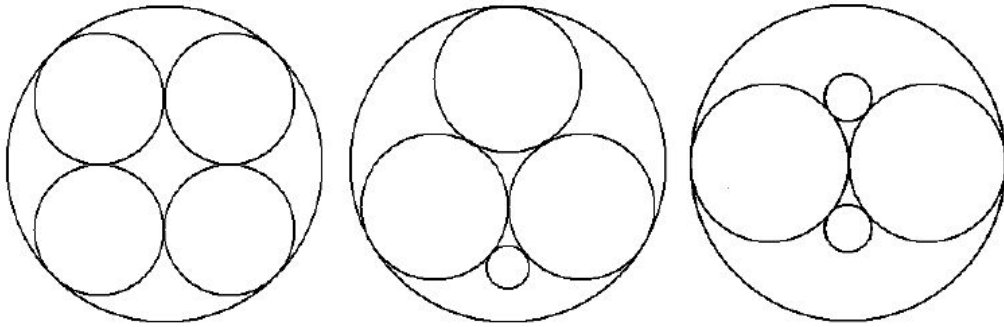
【题目编号】

2009 D Conduit Packing

【题目名称】

Conduit Packing

【题目大意】



如图，给出四个圆的直径，四个圆的位置可以任意移动。要求找到一个大圆把 4 个圆包含在内，且 4 个圆两两之间不重叠，求大圆最小直径。

四个圆的半径均为整数且 ≤ 20000 ，多组数据，数组组数 ≤ 100 。

【题目关键字】

二分，搜索，计算几何

【算法讨论】

显然，答案具有二分性。因此第一步二分答案。

二分答案后，枚举放入每个圆的顺序。最优方案中，第一个圆显然只要与大圆相切即可。在最优方案中从放入的第二个圆开始，每个放入的圆至少与之前的两个圆（包括大圆）相切。枚举与它相切的两个圆，求该圆圆心所在位置就转化成了求两圆交点的问题。然后解方程即可。枚举这个圆放入的位置，判断该位置是否合法，然后搜索判断即可。

因为圆的个数只有 4 个，所以枚举两个圆的方案不会很多，因此可以在时限内通过本题。

【时空复杂度】

时间复杂度： $O(T \cdot (C(4,2)^4 \cdot 2^4) \log V)$ ， T 为数据组数

空间复杂度： $O(1)$

【题目编号】

2009 E Fare and Balanced

【题目名称】

Fare and Balanced

【题目大意】

给出一个 n 个点, m 条边的有向无环带权图。图中每条边都在某条节点 1 到节点 n 的路径上。现在你可以增加其中某些边的权值, 要求:

- 1, 每条节点 1 到节点 n 的路径长度相等。
- 2, 没有任何一条节点 1 到节点 n 的路径经过了两条权值被增大的边。
- 3, 满足上述条件的情况下, 最后节点 1 到 n 的路径长度最短。

要求判断是否有解, 如果有解, 输出方案。多组数据, 数据组数 ≤ 100 , $n, m \leq 50000$, 边权为正整数且不超过 1000。

【题目关键字】

最短路, 动态规划, 拓扑排序

【算法讨论】

显然, 因为边权只能增大, 所以最后的路径长度不会小于原图中节点 1 到节点 n 的最长路。因此在有解的情况下, 只有构造任意节点 1 到节点 n 的路径长度均为原图中节点 1 到节点 n 的最长路长度的路径, 就一定是最优解。

首先对原图进行一次拓扑排序。拓扑排序后, 对原图进行两次 DP。

第一次 dp 从前往后。令 $f(i)$ 表示节点 1 到节点 i 的路径长度是否唯一, $F(i)$ 表示节点 1 到节点 i 的最长路。

第二次 dp 从后往前。令 $g(i)$ 表示节点 i 到节点 n 的路径长度是否唯一, $G(i)$ 表示节点 i 到节点 n 的最长路。

若存在 $f(i)$ 为 false 且 $g(i)$ 为 false, 那么显然无解。否则, 能增加权值的边 (u, v) , 所有经过它的路径长度都相等。这样的边一定满足以下条件: $f(u)$ 为 true, $f(v)$ 为 false 且 $g(v)$ 为 true。

那么对于每条满足上述条件的边, 假设它的权值为 c , 因为 $f(u)$ 和 $g(v)$ 均为 true, 所以经过它的路径长度都相等。给它的权值加上 $F(n) - F(u) - G(v) - c$, 就可以令通过这条边的所有路径长度变为目标长度。并且可以发现, 每条从起点到终点长度小于目标长度的路径, 一定存在路径上的某个点 $f(i)$ 为 false, 也就是一定会经过一条被修改过的边。这样我们就构造出了相当于理论下界的最优解。

DP 和拓扑排序的复杂度均为 $O(n+m)$, 因此算法的总复杂度也为 $O(n+m)$ 。

【时空复杂度】

时间复杂度: $O((n+m)*T)$, T 为数据组数。

空间复杂度: $O(n+m)$

【题目编号】

2009 F Deer-Proof Fence

【题目名称】

Deer-Proof Fence

【题目大意】

给出平面上 n 个点的坐标，要求你用围栏将这些点围起来。你可以用一个围栏围住所有的点，也可以用多个围栏，每个围栏围住不同的点。要求每个点距离围栏的距离至少为 m 。问围栏最小的总长度。

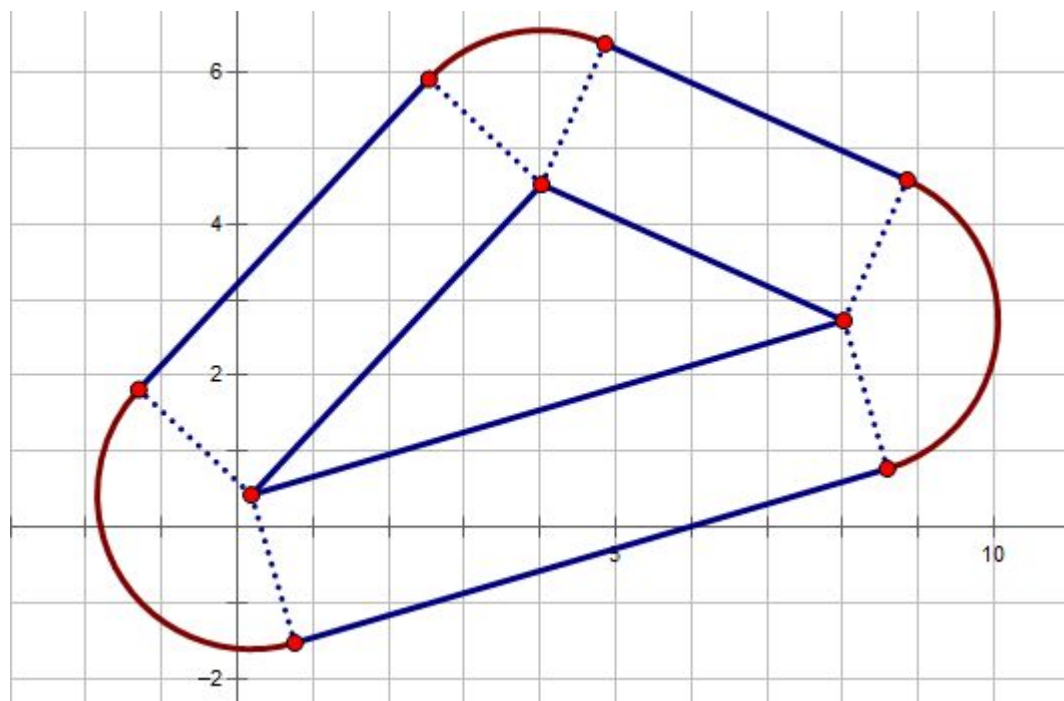
多组数据，数据组数 $T \leq 100$, $n \leq 9, m \leq 100$, 每个点的坐标绝对值 ≤ 100

【题目关键字】

状态压缩 DP，凸包

【算法讨论】

如图，将某几个点围起来的围栏的最小长度，等于这几个点所成的凸包周长加上半径为 m 的圆的周长。



令 $\text{cost}(S)$ 为用一个围栏围住集合 S 中的所有点最小围栏长度。每次求 $\text{cost}(S)$ 的复杂度为 $O(n \log n)$ ，求 2^n 次，总复杂度为 $O(2^n \cdot n \log n)$

接下来令 $f(S)$ 为用一个或多个围栏围住集合 S 中的所有点的最小长度， $f(S)$ 的转移如下：

$$f(S) = \min(f(S - S') + \text{cost}(S')) \mid S' \subseteq S$$

转移的总时间复杂度为 $O(3^n)$ ，因为 n 较小，所以可以轻松通过本题。

【时空复杂度】

时间复杂度： $O(T \cdot 3^n)$

空间复杂度： $O(2^n)$

【题目编号】

2009 G House of Cards

【题目名称】

House of Cards

【题目大意】

给出一个纸牌游戏，规则如下：纸牌有 $2M$ 张，只有红与黑 2 种花色，用 R 和 B 表示，分别为 $1R, 2R, 3R, \dots, MR, 1B, 2B, 3B, 4B$ 。洗牌后抽出前 8 张纸牌，组成 4 座山峰，如图所示：

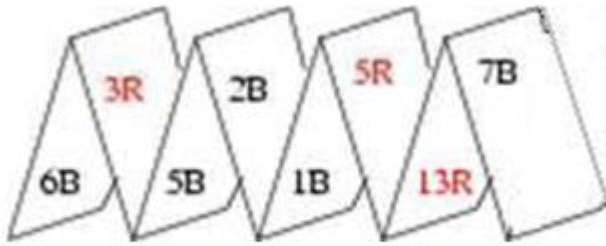


图 7：由牌堆的最上面 8 张牌组成的山峰和山谷

剩下的纸牌正面朝上被放置成一行。

每个玩家被认定一种颜色，红色或黑色。第一张用于组成山峰和山谷的纸牌的颜色决定了哪个玩家先开始。玩家交替进行操作。一步操作包括从一排纸牌的最前面抽取一张纸牌然后进行下列的一条：

1. 持有这张纸牌直到下次操作（这是一张“被持有的纸牌”）。
2. 用刚抽取的纸牌或被持有的纸牌覆盖在两个山峰之间的山谷，形成一个“基底”。如果还剩下一张牌，那么这张牌就被持有。
3. 把 2 张纸牌放在基底上面，形成一个山峰（其中一张纸牌一定是一张被“持有”的纸牌）。

不是所有的选择总是可行的。任何时候最多持有 1 张纸牌，所以第一个选择只有当这个玩家没有持有纸牌时才可行。

因为排成一排的纸牌是正面朝上，所以两个玩家在纸牌被抽取前就事先知道纸牌的顺序。

如果玩家通过添加了一个基底组成了一个向下的三角形，或者通过添加了一个山峰组成了一个向上的三角形，那么玩家的分数就会像下面描述的那样更新。组成三角形的 3 张纸牌的等级之和将被增加到那个颜色与 3 张纸牌的多数颜色相等的那个玩家的分数上。如果在游戏中没有组成三角形，两个玩家的分数保持不变。

如果在某步操作结束后没有纸牌等待被抽取，这个游戏就结束了。如果某个玩家在这个时候持有纸牌，那个玩家的分数将会增加（或减少）这张纸牌的等级如果这张纸牌的颜色与玩家颜色相同（不同）。

给出纸牌的顺序，问两个玩家都采取最优策略的情况下，最后的两名玩家的分差是多少， $M \leq 13$ 。

【题目关键字】

博弈树搜索，alpha-beta 剪枝

【算法讨论】

这是一个很明显的博弈游戏，所以可以使用博弈树搜索。因为不同策略产生的状态基本不同，所以记忆化搜索的策略在这里并不适用。但因为 M 很小且可供选择的策略非常少，加上 alpha-beta 剪枝后很快就出解了，博弈树搜索+alpha-beta 剪枝即可通过本题。alpha-beta 剪枝详情请见英文 wiki: http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

难点在于搜索的具体实现，我们可以用两个数组分别记录“山峰”和“基底”。搜索的过程中不具体计算每一步对分差造成的影响，到最后一步在计算双方分值并返回即可。这样虽然会牺牲一点速度，但代码会更加简洁和方便检查。

【时空复杂度】

时间复杂度： $O(6^{(2M-8)})$ ，实际复杂度远小于估计值

空间复杂度: $O(M^2)$

【题目编号】

2009 H The Ministers' Major Mess

【题目名称】

The Ministers' Major Mess

【题目大意】

给出 n 个议案，和 m 个大臣，每个大臣会对某些议案投票表示赞成或反对。由于票数统计等一系列技术方案在设计上局限性，每个大臣只能对最多四个不相同的议案投票（但这几乎不成问题，因为大多数大臣只对少数议题感兴趣）。得到了这些投票之后，议案将被决定是否通过，使得每个大臣有大于一半的建议被满足。要求你输出在满足所有大臣大于一半的建议的方案中，哪些议案一定要通过，哪些一定不能通过，哪些既可以通过又可以不通过。无解输出 impossible。 $n \leq 100, m \leq 500$

【试题关键字】

2-sat, floyd

【算法讨论】

对于关心议案少于 3 个的大臣，他的建议都必须被满足，它的反对建议必须不被满足。

对于关心议案为 3 个或 4 个的大臣，他的某个建议如果不被满足，那么他其它的建议都必须被满足。这是一个经典的 2-sat 问题，建立 $1, 1', 2, 2', \dots, n, n'$ 共 $2n$ 个节点。分别代表每个议案通过或不通过。然后对于每个关心议案为 3 个或 4 个大臣，枚举它的一个不满足的建议所对应节点，向其它满足建议的节点连一条有向边。然后对于一个节点，如果能通过有向边到达 i 和 i' ，或是能到达某些必须不被满足的节点，那么该节点不可选。

若节点 i, i' 均不可选，那么输出无解。否则根据 i 和 i' 的可选情况即可判断该议案是一定要通过，一定不能通过，或既可以通过又可以不通过。

对于判断每个节点可达的节点，可以用 floyd 算法实现，时间复杂度为 $O(n^3)$ 。

【时空复杂度】

时间复杂度: $O(n^3)$

空间复杂度: $O(n^2)$

【题目编号】

2009 I Struts and Springs

【题目名称】

Struts and Springs

【题目大意】

平面上有一些窗户，当外层窗户的大小变化时，里面的窗户也会变化，而支杆和弹簧是决定里面的窗户怎样变化大小或改变位置的装置。每扇窗户覆盖了平面上的一个矩形区域，窗户可以包含别的窗户以产生层次。当最外面的窗户改变形状时，每一个被它直接包含的窗户会改变位置或大小（基于支杆和弹簧的布置）；这些变化可能又会导致更里面的窗户改变位置、大小。

支杆定义为一个长度固定的杆，被放置在窗户的水平边或竖直边之间，或被放置在窗户的一条边与直接包含它的窗户的相对的边之间。如果一个支杆连接一个窗户相对的水平边或竖直边，那么这个窗户的宽度和高度就固定了。同样地，如果一个支杆连接了窗户的一条边与包含它的窗户的一条边，那么这两条边之间的距离就固定了。但是弹簧可以被拉伸或压缩，而且可以替换支杆。

除了最外面的窗户，每一个窗户连着六根支杆或弹簧。一根连接窗户的两条垂直边，另一根连接窗户的两条水平边。另外四根连接这个窗户的一条边与直接包含它的窗户的相对的边。垂直放置的三根支杆或弹簧的长度和等于直接包含它的窗户的高度；类似地，水平放置的三根支杆或弹簧的长度和等于直接包含它的窗户的宽度。直接包含它的窗户的宽度改变时，所有水平放置的弹簧会按照相同的比例压缩或伸长，新的水平支杆和弹簧的总长等于这个新的宽度。类似地，包含它的窗户的高度改变时，垂直放置的弹簧也会产生同样的变化。如果三根垂直的或水平的装置都是支杆，那么最上面的支杆或最右边的支杆会分别被一根弹簧代替。你需要写一个程序完成一下工作：输入初始窗户的大小和位置（保证一个窗户直接或间接包含其他窗户），撑杆和弹簧的位置，以及改变最外面的窗户的操作。请你对每次操作输出当前窗户的大小和位置。

【题目关键字】

模拟

【算法讨论】

首先找出直接包含每个窗户的窗户。方法为把窗户按照宽度从大到小排序。然后直接包含某个窗户的窗户就是包含它的窗户中宽度最小的一个。

接下来对于每个窗户，要处理操作对它的影响，就要先处理操作对直接包含它的窗户的影响。因此我们按照排序的结果按宽度从大到小的顺序处理操作队每个窗户的影响即可。

【时空复杂度】

时间复杂度： $O(nm)$

空间复杂度： $O(n)$

【题目编号】

2009 J Subway Timing

【题目名称】

Subway Timing

【题目大意】

给出一棵 n 个节点树，给出通过树上每条边的时间，单位为秒。现在要求将通过树上每条边的时间以分钟写出，精度保留到整数。你可以自由选择每条边的时间是向上取整或是向下取整。要求给出一个方案，使得取整后任意两点间距离与原来树上任意两点间距离的误差值的最大值最小。要求求出最小的最大误差值。

$n \leq 100$, 多组数据

【题目关键字】

二分，动态规划

【算法讨论】

首先，答案具有很明显的二分性。因此第一步二分答案。

设当前二分的答案为 L ，令 $f[r][v]$ 表示以 r 为根的子树中， r 的后代中任意两点距离都在 $[-L, L]$ 中，距离 r 最远的后代距离为 v 的方案中，距离 r 最近的后代距离最远为 $f[r][v]$ 。显然每棵子节点对父亲有影响的只有它的后代到自己距离的最大值和最小值。显然，对于距离最大值相同的方案，距离最小值越大越好，因此只要保留最小值最大的一个即可。

假设已经求出 r 的子树 x 的所有状态，将 x 的状态合并到 r 的状态时，分别枚举双方的距离最大值 i 和 j ，然后枚举它们之间边是向上或向下取整即可。总的时间复杂度为 $O(n * L^2)$ 。 L 为二分的答案。再乘以二分的复杂度，总的时间复杂度为 $O(n * L^2 \log L)$

答案一般不会太大，一般只有几百。因此把 L 看做与 n 同阶，所以总的复杂度为 $O(n^3 \log n)$ 。可以轻松通过本题。

【时空复杂度】

时间复杂度： $O(n^3 \log n)$

空间复杂度： $O(n^2)$

【题目编号】

2009 K Suffix-Replacement Grammars

【题目名称】

Suffix-Replacement Grammars

【题目大意】

一个后缀替换法由起始字符串 S 和一些后缀替换规则组成。每个规则是由 $X \rightarrow Y$ 的形式给出，其中 X 和 Y 是等长的由字母构成的字符串。这个规则表示如果你现在的字符串后缀（最右的字符）是 X ，你就可以用 Y 替代它。这个规则可以无限次使用。

举个例子，假如有 4 个规则 $A \rightarrow B, AB \rightarrow BA, AA \rightarrow CC, CC \rightarrow BB$ ，你就可以通过三步把 AA 变成 BB ： $AA \rightarrow AB$ （用 $A \rightarrow B$ 规则）， $AB \rightarrow BA$ （用 $AB \rightarrow BA$ 规则）， $BA \rightarrow BB$ （用 $A \rightarrow B$ 规则），但你也可以用两步更快的完成： $AA \rightarrow CC$ （用 $AA \rightarrow CC$ 规则）， $CC \rightarrow BB$ （用 $CC \rightarrow BB$ 规则）。

给出 n 条规则和两个等长的字符串 S, T ，要求判断通过后缀替换规则字符串 S 能否变成 T ，如果可能，求最小步数。 $0 \leq NR \leq 100$ ，所有字符串长度 ≤ 20 ，多组数据。

【题目关键字】

DP, 最短路

【算法讨论】

按长度划分 DP 阶段处理。一个字符串 x 能转化成 y 有两种情况, 一种是通过某些转化规则使得 x 整个能转化成 y , 一种是 x 的首字母等于 y 的首字母, 通过某些规则能使 x 删除首字符后所成的字符串 x' 转化成 y 删去首字符后所成的字符串 y' 。

不难发现上述过程中的中间状态一定是输入的某个字符串的后缀。因此我们把所有字符串的所有后缀按照长度分层。对于第一种情况, 直接判断首字母用直接调用上一层的结果即可。对于第二种, 同一层的节点之间互相转化的最小代价可以转化成无向图两两之间的最短路, 使用 $O(n^3)$ 的 floyd 即可。

最后总的时间复杂度为 $O(20 * n^3)$ 。因为实际状态比较稀疏, 所以可以轻松通过本题。

【时空复杂度】

时间复杂度: $O(20 * n^3)$

空间复杂度: $O(20 * n^2)$

【题目编号】

2010 B Barcodes

【题目名称】

Barcodes

【题目大意】

Code-11 是一种主要用来将标签编码为条形码的编码方式。被编码的字符被限制为 0-9 和 '-' 号, 以及一种特殊的符号: 开始和结束标志 (开始标志出现在 Code-11 编码的最前面而结束标志出现在最后面)。

Code-11 的编码会独立的编码每一个字符。一个字符会由 5 个相邻的区域来编码。每个区域可能是深色和浅色中的一种, 相邻的两个区域的颜色一定不同, 每次编码的第一个区域的颜色一定是深色的。每个区域的宽度也不是一定的, 我们将宽度总共分为两种, 用 0 表示一个窄的区域, 用 1 表示一个宽的区域。

字符	编码
0	00001
1	10001
2	01001
3	11000
4	00101
5	10100
6	01100
7	00011
8	10010
9	10000
-	00100
开始/结束标志	00110

在编码后两个相邻的字符各自编码出来的五个区域之间需要用个浅色的窄区域将它们隔开。这个浅色的窄区域的唯一功能就是分割两个字符编码出来的区域。

为了能够检验编码中出现的错误，Code-11 在编码时会用到两个检验字符 C 和 K，并且会把 C 和 K 插入到原串的末尾再进行编码（在结束标志之前）。假设需要被编码的串有 n 个字符，分别是 c_1, c_2, \dots, c_n ，那么检验字符 C 的值就应该是 $\text{sigma}((n - a) \bmod 10 + 1) * w(c[i]) \bmod 11 (1 \leq i \leq n)$ 。其中， $w(c[i])$ 是字符 $c[i]$ 的权重。如果 $c[i]$ 是 0、1、2、……9 这些字符，那么 $c[i]$ 的权重就是 0、1、2、……9。如果 $c[i]$ 是 '-'，那么 $c[i]$ 的权重就是 10。

与此对应的检验字符 K 的值是 $\text{sigma}((n - a + 1) \bmod 9 + 1) * w(c[i]) \bmod 11 (1 \leq i \leq n+1)$ ，其中 $c[n+1]$ 即为检验字符 C。

你的任务是解码一个被 Code-11 编码过的条形码。你知道每一块区域的宽度，并且宽的区域宽度是窄的区域的宽度的两倍。条形码可能倒序输入。读取的区域宽度可能跟实际的宽度有 5% 的误差。在数据当中不会有编码前长度为 0 的字符串。

如果该条形码无法被成功解码，输出 "bad code"。

如果该条形码能被成功解码但是检验字符 C 的值不正确，输出 "bad C"。

如果该条形码能被成功解码且检验字符 C 的值正确但检验字符 K 的值不正确，输出 "bad K"。

如果该条形码能被成功解码且两个检验字符均是正确的，输出不包含检验字符和起始结束标志的原字符串。N ≤ 150。

【题目关键字】

模拟

【算法讨论】

首先枚举是否翻转条形码。接下来找到最窄的区域，它一定是一个窄区域。把它作为基准，判断是否所有区域和它的误差都在 5% 以内。然后判断其它区域是否宽窄区域只需要用该区域的长度除以最窄区域的长度并四舍五入即可。分出观察区域后剩下的只要按照题意模拟解码出每个字符，然后判断 C, K 是否合法即可。

【时空复杂度】

时间复杂度：O(n)

空间复杂度：O(n)

【题目编号】

2010 C Tracking Bio-bots

【题目名称】

Tracking Bio-bots

【题目大意】

给出一个 $m \times n$ 大小的网格，左下角为 $(0,0)$ ，右上角为 $(n-1,m-1)$ 。网格中有 w 个宽度为 1 个网格的、与 x 轴平行的墙壁，墙壁互相不会重叠。机器人每个可以从所在网格向上或向右移动，但不能穿越墙壁。问有多少个格子机器人不能移动到 $(n-1,m-1)$ 。

$0 \leq n, m \leq 10^6, 0 \leq w \leq 1000$

【题目关键字】

枚举，模拟，DP。

【算法讨论】

原问题很明显等价于，从(n-1,m-1)这个格子出发，可以向下或向右移动，但不能穿越墙壁，有多少个格子不能到达。

这个问题很明显可以用一个 $n*m$ 的 dp 解决。 $f[x][y]$ 表示(x,y)这个格子是否可达。若(x,y)未被墙壁覆盖， $f[x][y]=f[x+1][y]$ or $f[x][y+1]$ ，否则 $f[x][y]=false$ 。

这题 n,m 很大，但 w 很小，因此将墙壁坐标离散化后，再用 dp 解决，时间复杂度为 $O(w^2)$

【时空复杂度】

时间复杂度： $O(w^2)$

空间复杂度： $O(w^2)$

【题目编号】

2010 D Castles

【题目名称】

Castles

【题目大意】

你要攻占一棵树。你可以从任意一个节点开始攻击这棵树，攻击第 i 个节点至少需要 a_i 的兵力，攻击过程中会损失 m_i 的兵力，攻击完成后需要 g_i 的兵力来看守它。攻击一个节点后结束后，可以派人沿着树边去攻击其它节点。军队可以安全到达已被夺取的节点的相邻节点。但由于被进攻的潜在可能，军队在同一方向上经过同一条路最多一次。问攻占这个数的最少总兵力。 $n \leq 100, a_i, g_i \leq 1000, m_i \leq a_i$

【题目关键字】

树形 DP，贪心

【算法讨论】

定义 $\text{cost}[i]=m[i]+g[i]$ 为攻占节点 i 的兵力消耗, $\text{need}[i]=\max(a[i], \text{cost}[i])$ 为攻占节点 i 的最少所需总兵力。首先, 对于一个完全图, 很容易想到贪心的解法, 按照 $\text{need}[i]-\text{cost}[i]$ 的值从大到小攻占图中的每个节点即可。

接下来对于一颗树, 枚举根作为第一个攻占的节点。我们可以算出根的每个儿子攻占所需的最少所需总兵力和兵力消耗, 然后参考之前的贪心方法, 按照差值从大到小攻占每棵子树即可。而计算每棵子树的最少所需总兵力和兵力消耗是一个和原问题类似、但规模更小的子问题, 递归处理即可。

对于每次枚举的根, 状态总数是 $O(n)$ 的, 转移总复杂度是 $O(n \log n)$, 因此总的复杂度为 $O(n^2 \log n)$ 。

【时空复杂度】

时间复杂度: $O(n^2 \log n)$

空间复杂度: $O(n^2)$

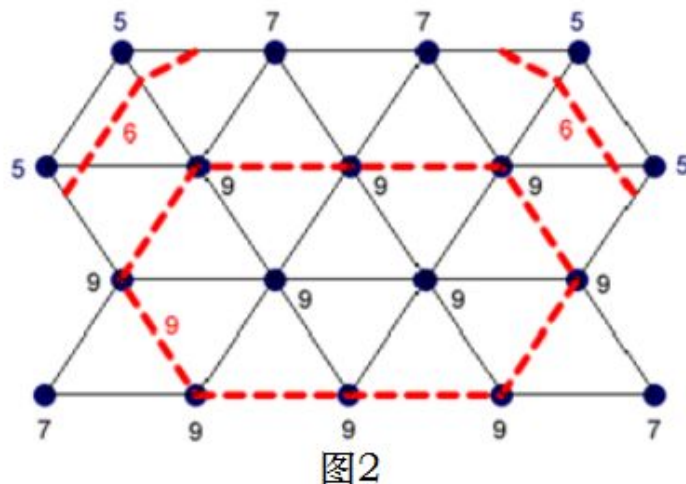
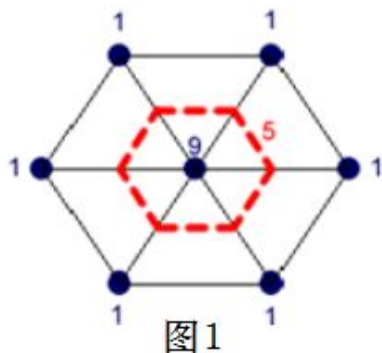
【题目编号】

2010 F Contour Mapping

【题目名称】

Contour Mapping

【题目大意】



如图，在平面上选取若干个等高点。所有不在边缘的每个测高点都有 6 个其他的测高点离它最近，而且相互等距（这里忽视海拔高度）。测高点共 p 行，第奇数行有 s 个测高点，第偶数行有 $s+1$ 个测高点。三角形边长为 d ，所有海拔为 h 整数倍的地方会有一条等高线，问等高线的总长度。

$p, s, d \leq 1000$ ，测高点高度 ≤ 1000000

【题目关键字】

模拟，数学

【算法讨论】

把每个三角形取出来，单独求三角形上等高线的长度。根据相似三角形定理，三角形上等高线的长度组成了两个等差数列，可以 $O(1)$ 计算。

接下来是判重。对于图上两个相邻的高度相等且为 h 整数倍的测高点，若包含它的两个相邻三角形的另外两个测高点和这两个点的高度相等，那么这条边被重复计算了两次，否则被重复计算了一次，从答案中减去即可。

【时空复杂度】

时间复杂度： $O(p*s)$

空间复杂度： $O(p*s)$

【题目编号】

2010 G The Islands

【题目名称】

The Islands

【题目大意】

按照 x 坐标递增给出平面上的 n 个点，编号为 $0 \sim n-1$ 两点间距离即为它们的欧几里得距

离。要求找到一条路径，从 x 坐标最小的点出发，一直向东，到达 x 坐标最大的点。然后一直向西直到到 x 坐标最小的点，除了起点外不重复经过任何点，且恰好经过除起点外每个点正好一次。同时给出两个点 b_1, b_2 ，要求从从起点到终点的路径上经过其中一个点，从终点回起点的路径上经过另一个点，求最短路，并输出方案，要求方案中起点出发的路径首先经过节点 1。多组数据, $n \leq 100$

【题目关键字】

动态规划

【算法讨论】

因为处理环很麻烦，所以我们可以把问题转化成从起点出发，找两条不相交的路径经过所有点到达终点，求最短路。

令 $f[i][j]$ 表示第一条从起点出发的路径当前的终点为 i ，第二条为 j 时的最短路。然后从左到右枚举每个节点在哪一条路径上即可转移，时间复杂度为 $O(n^2)$ 。

对于 b_1, b_2 ，只要转移过程中强制要求第一条路径经过 b_1 ，第二条经过 b_2 即可。同时记录方案即可。

这样求出来的方案不能保证方案中起点出发的路径首先经过节点 1，对于这种情况只要交换两条路径然后再输出即可。

【时空复杂度】

时间复杂度： $O(n^2)$, T 为数据组数

空间复杂度： $O(n^2)$

【题目编号】

2010H Rain

【题目名称】

Rain

【题目大意】

给出一个地区的三角划分模型。这种模型使用一些三角形来近似表示地表。相邻三角形

共享整条邻边。除了边界上的边，每条边都被两个相邻的三角形完整地共享。

下雨的时候，一些雨水流向了海洋，另一些则被困在这块地区，形成了湖泊。你需要写一个程序去确定形成了多少个湖泊以及每个湖泊的水面的海拔。你可以假设雨足够大以使每个湖泊的水面达到它最大可能的海拔高度。

对于每个湖泊，你都可以驾驶小船在它表面上任意两点间穿梭。船的体积可以尽可能的小，但不能为 0。这就意味着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。

给出该三角划分的点数 p 和边数 s ，给出每个点的高度。你可以认为所有在这个区域外的点的高度都比与其相邻的边界低。因此当雨水流到区域边界的时候可以自由流出。

$p \leq 2756$, $0 \leq \text{高度} \leq 8848$

【题目关键字】

计算几何、最短路、floodfill

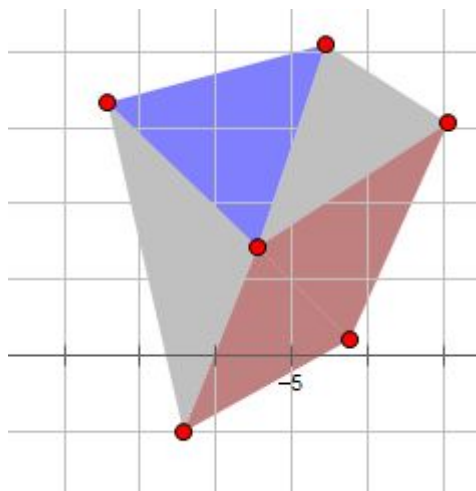
【算法讨论】

因为这个图是个平面图，所以边数 s 与点数 p 同阶。

令 $h[i]$ 为每个点的高度， $dis[i]$ 为节点 i 通过图上的边到达区域外的路径上，高度最高的点高度的最小值。显然 $dis[i]$ 即为雨后节点 i 的水面高度，所有 $h[i] < dis[i]$ 的点上都有积水。

那么相邻的有积水的点组成的连通块即为湖泊。可以用 floodfill 从每个未被遍历过的，有积水的节点，遍历所有相邻的有积水的节点。这样遍历出来的有积水的节点组成的块即为湖泊。floodfill 总复杂度为 $O(p+s)$ ，也就是 $O(p)$ 。

求出每个节点的 $dis[i]$ 值，可以从所有在区域边界上的点开始，用 dijkstra 在 $O(p^2)$ 的时间内求出，关键是求出哪些点在区域边界上。



如图，可以发现一个在区域内的点，它相邻的点组成了一个把它包含在内的简单多边形。因此判定一个点是否在区域上时，可以把所有和它相邻的点按逆时针排序，然后判断相邻点是否组成了包含该点的多边形。这一步的时间复杂度是 $O(s \log s)$ ，也就是 $O(p \log p)$ 。

最后算法的复杂度瓶颈在于 dijkstra 的 $O(p^2)$ ，如果用 heap 优化 dijkstra，可以做到 $O(p \log p)$ ，能够处理更大规模的数据。不过对于本题 $p \leq 2756$ ， $O(p^2)$ 的 dijkstra 已经可以轻松通过本题，就不需要进一步的优化了。

【时空复杂度】

时间复杂度: $O(p^2)$

空间复杂度: $O(p^2)$

【题目编号】

2010 I Robots on Ice

【题目名称】

Robots on Ice

【题目大意】

给出一个 $n*m$ 大小的网格，左下角为(0,0),右上角为(n-1,m-1)。给出三个坐标

$(r1,c1),(r2,c2),(r3,c3)$ ，要求从 $(0,0)$ 出发，每次可以走到上下左右四个相邻格子，不重复的遍历 $n*m$ 个格子并在 $(0,1)$ 结束，要求在第 $(n*m*i)/4$ 步是正好位于坐标 (ri,ci) 的方格上，求方案数。

$n,m \leq 8$ ，数据组数 ≤ 10

【题目关键字】

搜索，剪枝

【算法讨论】

因为没有想到多项式的算法，所以我采用搜索来一一枚举所有合法方案。

因为限制条件很强，合法方案数不会太多，因此要注意剪枝。

主要有以下三条剪枝：

- 1，对于每个 (ri,ci) ，如果当前位置与 (ri,ci) 的曼哈顿距离大于到达这个位置的剩余时间，那么退出。
- 2，对于每个 (ri,ci) ，如果提前到达了位置，那么退出。
- 3，如果搜索过程中，路径把棋盘分成两个不相交的部分，那么退出。

加上这三条剪枝后，就可以在时限内通过本题了。

【时空复杂度】

时间复杂度： $O(3^{(n*m)*T})$ ， T 为数据组数

空间复杂度： $O(n*m)$

【题目编号】

2010 J Sharing Chocolate

【题目名称】

Sharing Chocolate

【题目大意】

给出一个由 x 行 y 列的网格组成矩形，每次你可以沿着行或者列的分割线将一个矩形切

成两块。问是否能将矩形切割成大小分别为 a_1, a_2, \dots, a_n 的 n 块。 $x, y \leq 100, n \leq 15$ 。

【题目关键字】

状态压缩，DP

【算法讨论】

设 $dp(a, b, S)$ 表示将 $a \times b$ 的矩形，分成集合 S 中的大小的若干块，是否存在合法方案。 S 是 $\{a_1, a_2, \dots, a_n\}$ 的子集，用一个二进制数表示。

令 $sum(S)$ 表示集合 S 中的元素之和，转移过程则枚举 S 的一个真子集 s_0 ， s_1 为 s_0 的补集，然后枚举按行线或列线分割成两块。

按行线分割成两个子集： $dp(a, sum(s_0)/a, s_0), dp(a, sum(s_1)/a, s_1)$

按列线分割成两个子集： $dp(b, sum(s_0)/b, s_0), dp(b, sum(s_1)/b, s_1)$

这样的状态最多有 $x \times y \times 2^n$ 个，时间和空间都不够。但我们注意到当 S 和 x 确定时，合法的 y 值也唯一确定。因此只要保存 y 即可。这样状态只有 $x \times 2^n$ 个，转移的总复杂度则是 $x \times 3^n$ ，可以通过本题。

【时空复杂度】

时间复杂度： $O(x \times 3^n)$

空间复杂度： $O(x \times 2^n)$

【题目编号】

2010K Paperweight

【题目名称】

Paperweight

【题目大意】

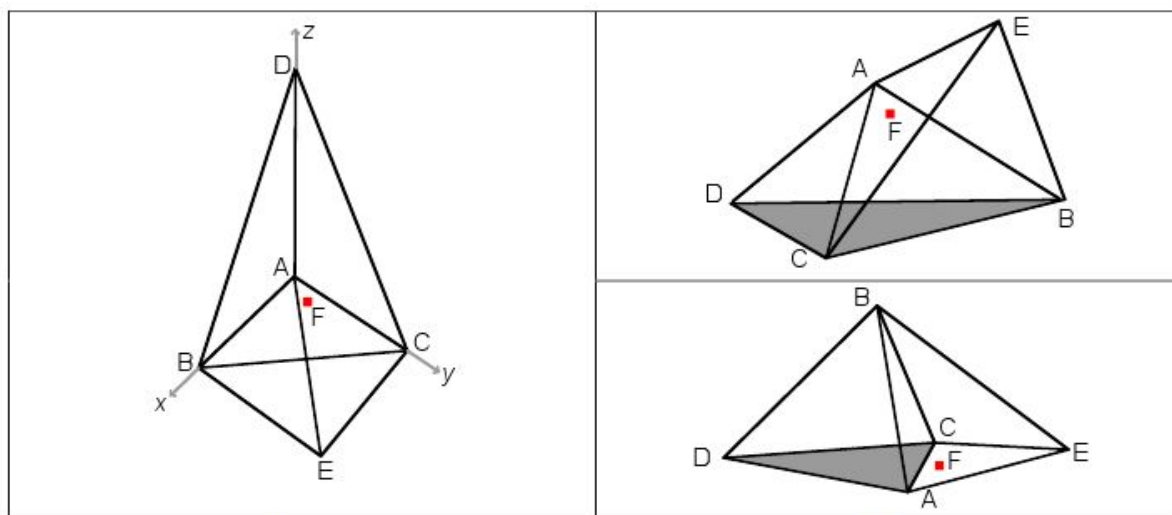


图 1

图 2

给出六个点 A,B,C,D,E,F，多面体 ABCDE（如图）是一个由两个四面体合起来组成的多边形，点 F 被多面体严格包含。现在可以选择把多面体的某一个面放在地板上，但要求放在地板上后，多面体的重心无论往哪个方向移动，多面体都不会发生移动。问点 F 距离地面的最小和最大距离。点坐标的绝对值 ≤ 1000

【题目关键字】

计算几何、枚举

【算法讨论】

首先第一步是求多面体的重心。这个多面体的重心其实就是两个四面体重心的加权平均数。

接下来枚举多面体的每个面，把它作为底面。若多面体的顶点都在这个面的同一侧，那么这个面可以作为多面体的底面。然后对在底面上的点求一个凸包，再把重心投影到底面上，若重心的投影在凸包外，多面体显然会移动。若重心的投影在凸包内但距离凸包边界边的距离小于 0.2，那么重心移动后多面体也会移动。如果多面体的顶点都在这个面的同一侧且重心的投影在凸包内但距离凸包边界边的距离大于 0.2，F 点到地面的距离就是一个可行解。多个解中将最大、最小值输出即为答案。

【时空复杂度】

时间复杂度：O(1)

空间复杂度：O(1)

【题目编号】

2011 A To Add or to Multiply

【题目名称】

To Add or to Multiply

【题目大意】

给出两个数 a, m 。有两种操作 A 和 M，A 是将一个数加上 a ，M 是将一个数乘以 m 。现在给出两个区间 $[p, q]$ 和 $[r, s]$ ，要求求出一个最短的由 A 和 M 组成的操作序列，使得区间 $[p, q]$ 中的每个数经过这个序列中的操作之后的结果在区间 $[r, s]$ ，如果有多个解，输出字典序最小的一个。 $1 \leq a, m, p, q, r, s \leq 10^9$

【题目关键字】

贪心，枚举

【算法讨论】

对于 $m=1$ 的情况可以特殊处理

$m \geq 2$ 的时候，显然 M 的个数不可能太多，于是枚举 M 的个数，假设操作序列中有 n 个 M

假设序列中第 i 个 M 后面紧跟了 A_i 个 A ($A_i \geq 0$)，那么对某个数 x 的处理结果就是

$$(((x + A_0 \cdot a) \cdot m + A_1) \cdot m + A_2) \dots + A_{n-1}) \cdot m + A_n$$

$$= x \cdot m^n + a \cdot (A_0 \cdot m^n + A_1 \cdot m^{n-1} + \dots + A_n)$$

因此 $p \cdot m^n + a \cdot (A_0 \cdot m^n + A_1 \cdot m^{n-1} + \dots + A_n) \geq r$ 且 $q \cdot m^n + a \cdot (A_0 \cdot m^n + A_1 \cdot m^{n-1} + \dots + A_n) \leq s$

稍作处理可得 $(r - p \cdot m^n) / a = r' \leq A_0 \cdot m^n + A_1 \cdot m^{n-1} + \dots + A_n \leq s' = \dots$

然后显然对于 $i \geq 1$ ， $A_i \geq m$ 显然不是最优的，因此对于 $i \geq 1$ 有 $0 \leq A_i < m$

这可以认为是一个 m 进制数（除了最高位），找出范围内的最小数 A_{\min} 和最大数 A_{\max}

那么从 A_{\min} 的最高位（也就是 A_0 ）开始从高位到低位，每一位要么取值当前的 A_i ，要么 $A_i + 1$ （如果没有超过 A_{\max} ）且后面的低位都为 0，如此找到一个数位和最小的数，这个数就对应了最优的那个操作序列

【时空复杂度】

时间复杂度： $O(\log s)$

空间复杂度： $O(\log s)$

【题目编号】

2011 C Ancient Messages

【题目名称】

Ancient Messages

【题目大意】




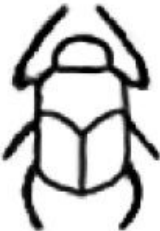

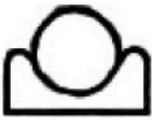
					
Ankh	Wedjat	Djed	Scarab	Was	Akeht

图1：六个象形文字

给出一个 $H*W$ 大小的由黑白像素组成的图像。图像满足以下规则

1. 图像仅包含图 1 所示的象形文字。
2. 每个图像至少有一个有效的象形文字。
3. 每个黑色像素都是一个有效象形文字的一部分。
4. 每个象形文字由一个联通的黑色像素块组成，对于每个黑色像素，至少在其上下左右至少有一块黑色像素块。
5. 象形文字互不接触，而且不存在一个象形文字在另一个的内部。
6. 如果有两块黑色像素对角线相接处，则必然存在一块公共接触的黑色像素。
7. 象形文字可能会扭曲，但是它的拓扑结构必然会和图 1 中所示的一个等价。(如果一个图像可以通过拉伸但不被毁坏的方式转化成另一个，则它们是拓扑等价的)

要求输出图像中包含哪些象形文字, $H,W \leq 200$ 。

【题目关键字】

floodfill

【算法讨论】

经过观察可以发现，每个象形文字内部的白色像素组成的空洞数目都不一样。因此我们对于每个联通的黑色像素块用 floodfill 统计它内部白色像素组成的空洞数目即可判断这个像素块是哪个象形文字，时间复杂度为 $O(H*W)$ 。

【时空复杂度】

时间复杂度： $O(H*W*T)$ ， T 为数据组数。

空间复杂度： $O(H*W)$

【题目编号】

2011 E Coffee Central

【题目名称】

Coffee Central

【题目大意】

给出一个 $dx*dy$ 的方格网,给出 n 个咖啡馆。第 i 个咖啡馆坐标为 (x_i, y_i) 。网格中 $(a, b), (c, d)$ 两点距离为 $|a-c|+|b-d|$ 。现在给出 q 个询问,每次询问给出一个数 m ,当两个点距离不超过 m 时,认为这两个点互相可达。问网格内在距离 m 内能到达咖啡馆数量最多的点,输出它能到达的咖啡馆数目和它的坐标,如果有多个点,输出 y 坐标最小的点,如果仍有多,输出 x 坐标最小的一个。 $1 \leq dx, dy \leq 1000, n \leq 500000, q \leq 10, m \leq 1000000$ 。

【题目关键字】

切比雪夫距离, 矩形覆盖

【算法讨论】

容易发现,两个点之间的距离就是他们的曼哈顿距离。把每个咖啡馆在平面上 m 步内可达的点画出来,是一个菱形。 n 个菱形重叠层数最多的区域就是答案。

菱形的情况比较复杂。但我们可以把原来的每个点坐标 (x_i, y_i) 转化为 $(x_i + y_i, x_i - y_i)$ 。新图后两个点的距离就是他们的切比雪夫距离, $(a, b), (c, d)$ 两点的距离为 $\max(|a-c|, |b-d|)$ 。可以发现原图的两个点之间的曼哈顿距离和在新图中它对应的两个点之间的切比雪夫距离相等。这样一来,每个点在新图中的覆盖区域就是一个矩形。

新图中网格的大小为 $(dx+dy)*(dx+dy)$, 在一个 $X*Y$ 的平面内每个点被矩形覆盖的次数可以离线后用区域和在 $O(X*Y)$ 的时间内求出来。因此我们可以用 $O((dx+dy)^2)$ 的时间在新图处理每次询问。这样总的复杂度为 $O(Q*((dx+dy)^2+n))$, 因为 Q 较小, 可以在时限内通过本题。

【时空复杂度】

时间复杂度: $O(Q*((dx+dy)^2+n))$

空间复杂度: $O((dx+dy)^2+n)$

【题目编号】

2011 F Machine Works

【题目名称】

Machine Works

【题目大意】

你要经营公司 D 天，初始有 M 元。 D 天中市场上有 N 台机器出售。第 i 台机器买入为 P_i ，卖出价格为 R_i 运转一天收益为 G_i ，在市场上出现的时间为第 D_i 天。公司最多同时拥有 1 台机器。对于第 i 台机器，只能在它出现在市场上的当天立即购买，而且购买时你持有的资金不能少于 P_i ，且每台机器买入和卖出的一天都不能使用。第 $D+1$ 天时公司会卖出拥有的机器（假如有的话）。问你经营公司的最大收入。

$N \leq 100000$ ，其余数字 $\leq 10^9$

【题目关键字】

动态规划，斜率优化，单调队列，分治/平衡树

【算法讨论】

首先将所有机器按出现时间排序。为了方便，令公司一开始拥有第 0 台机器，它的价格、收益、出现时间均为 0。

令 F_i 表示购入第 i 台机器的当天后公司拥有的最大金额。很明显，公司只有第 $D+1$ 天或要换购另一条机器是才有卖掉手上的机器，否则就会让它一直运转下去。那么枚举 i 之间购买的那台机器，设为 j 。 $F_i = \max(F_j + G_j * (D_i - D_j - 1) + R_j - P_i) \quad 0 \leq j < i$ 且 $F_j > G_j * (D_i - D_j - 1) + R_j$ 。

最后答案为 $\max(F_i + G_i * (D - D_i) + R_i)$ 。

这样 DP 的复杂度是 $O(n^2)$ 的，显然会超时。但这题的决策具有很明显的单调性。把决策 F_j 看成一个节点 $(G_j, F_j - G_j * (D_j + 1))$ ，然后用一条斜率为 D_i 的直线从 x 轴上方无穷远处向下逼近，遇到的第一个点即为最优解（显然）。

那么不在凸壳上的节点永远不可能成为最后解。因此有两种做法，一种是用平衡树维护当前所有决策对应节点组成的凸壳，然后对于每次询问可以在平衡树上二分找到最优解。另一种做法是采用分治法。假设当前要计算区间 $[L, R]$ 的最优解，首先递归计算区间 $[L, (L+R)/2]$ 的最优解，然后算出在 $[L, (L+R)/2]$ 的所有决策形成的凸壳。根据凸壳就可以计算区间 $[(L+R)/2+1, R]$ 的决策落在区间 $[L, (L+R)/2]$ 时的最优解。然后继续递归处理区间 $[(L+R)/2+1, R]$ 即可。

两种算法的时间复杂度均为 $O(n \log n)$ ，分治法代码较短，速度也较快，但两种算法都可以通过本题。

【时空复杂度】

时间复杂度： $O(n \log n)$

空间复杂度： $O(n)$

【题目编号】

2011 H Mining Your Own Business

【题目名称】

Mining Your Own Business

【题目大意】

给出一个 $m(m \leq 50000)$ 条边的无向图，要求你选出尽量少的节点建立安全竖井，使得删除任意一个节点后，每个节点都能至少与一个安全竖井相连通。求最少建立几个安全竖井，以及不同的方案数。

【题目关键字】

深度优先遍历，割点

【算法讨论】

最优解中安全竖井不可能建立在割点上。假设存在最优解把安全竖井建立在割点上，因为删除割点后，被割点分开的至少两个连通块依然与其他安全竖井连通。那么不删除割点时，两个连通块都能通过割点与另一个连通块的割点连通。也就是说割点上无论是否存在安全竖井，都满足条件，那么删除割点上的竖井，就能得到一个更优解，与假设矛盾。

首先遍历整个图，求出所有的割点，然后将所有割点删除。对于剩下来的点组成的每个连通块分开计算。一个连通块如果与其他连通块通过割点相连或连通块只有一个节点，那么只需要在连通块中任意选择一个节点建立安全竖井，答案最小值+1，方案数乘以连通块的节点数 s 。否则连通块中必须选择两个不同的节点，答案最小值+1，方案数乘以 $C(s, 2)$ 。

求割点的复杂度为 $O(m)$ ，求出割点后遍历所有连通块并统计答案的复杂度同样是 $O(m)$ ，因此总的时间复杂度为 $O(m)$ 。

【时空复杂度】

时间复杂度： $O(m)$

空间复杂度： $O(m)$

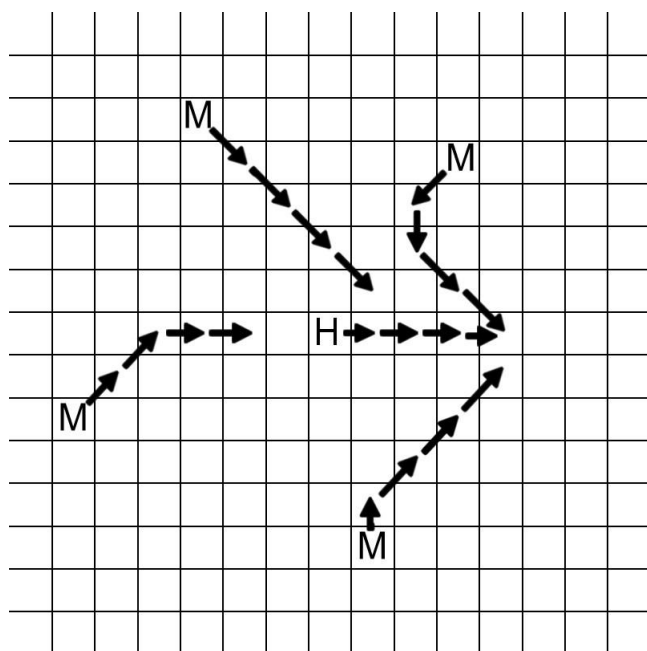
【题目编号】

2011 I Mummy Madness

【题目名称】

Mummy Madness

【题目大意】



在一个无穷大的正方形网格中，人(H)一开始位于点(0,0)，有 n 只木乃伊(M)位于不同的位置。每一步人可以往相邻的 8 个格子移动，每当人移动一步，木乃伊也会往它附近 8 个格子移动使得他与人的欧几里得距离尽量小（假设人与木乃伊都站在格子的中心位置）。问最多几步后人就会被木乃伊抓住。n<=100000,坐标绝对值<=1000000

【关键字】

二分，线段树，扫描线

【算法讨论】

二分答案，假设当前要判断是否能够在 t 时间内不被吃掉。

那么以自己为中心的边长为 $2t$ 的正方形就是人的活动范围，以每个木乃伊为中心的边长为 $2t$ 的正方形就是木乃伊的活动范围，只要人的活动范围没有被木乃伊完全覆盖那么就 t 步内人不会被抓住。

判断一个正方形是否被其它正方形覆盖是一个经典问题。把正方形按 x 坐标排序后用扫描线按 x 从小到大扫描所有正方形，线段树维护当前正方形覆盖区域即可。时间复杂度为 $O(n \log C \log C)$ ，C 为坐标范围。

【时空复杂度】

时间复杂度: $O(n \log C \log C)$, C 为坐标范围。

空间复杂度: $O(n+C)$ 。

【题目编号】

2011 J Pyramids

【题目名称】

Pyramids

【题目大意】

如果你有足够的石块，那么建一座金字塔绝不算难事。举个例子，在一块平地上，我们铺一个 10×10 的矩形，然后在 10×10 的矩形上面铺一个 9×9 的，然后 8×8 的……以此类推，直到顶上 1×1 。这个金字塔有 10 层，我们称这类金字塔为“高金字塔”。

如果你认为这样的金字塔太陡了，那么我们有办法让他看上去坡度平缓一些。比如，在 10×10 的矩形上，我们铺一个 8×8 的矩形，然后是 6×6 的……这样的金字塔只有 5 层了，大约为底座边长的一半。我们称之为“矮金字塔”。

现在给出 n 个石块，建造若干座金字塔，要求如下：

1. 所有石块都必须用上；
2. 金字塔数要尽可能少；
3. 所有金字塔两两不同；
4. 金字塔至少包含两层，即底座为 1 的金字塔和底座为 2 的矮金字塔是不允许的；
5. 满足以上 4 点的基础上，最大的金字塔要尽可能大（大定义为用的石块数多）；
6. 满足以上 5 点的基础上，次大的金字塔要尽可能大；
7. 以此类推。。

$n \leq 1000000$

【题目关键字】

背包，迭代加深，深度优先搜索

【算法讨论】

首先把所有大小在 1000000 以内的金字塔求出来，发现只有 320 座。现在问题转换成有 320 个物品，恰好装满一个大小为 n 的容器。这是一个背包问题。设 $f[i][j]$ 为使用前 i 个物品，恰好装满大小为 j 的空间，最少需要使用几个物品。这样的时间复杂度是 $O(n \cdot m)$ ，会超时。

通过前面求出的 $f[i][j]$ ，可以发现有解的 n 的最优解使用的物品个数不超过 6。解的大小很小，因此首先枚举答案大小，然后搜索方案即可。这样的复杂度最坏是 $O(C(320, 6))$ ，但确定了 5 个物品后，第 6 个物品的大小也就确定了，因此总的复杂度是 $O(C(320, 5))$ ，再加上剪枝，就可以轻松的通过本题。

【时空复杂度】

时间复杂度： $O(C(320, 5))$

空间复杂度： $O(n)$

【题目编号】

2011 K Trash Removal

【题目名称】

Trash Removal

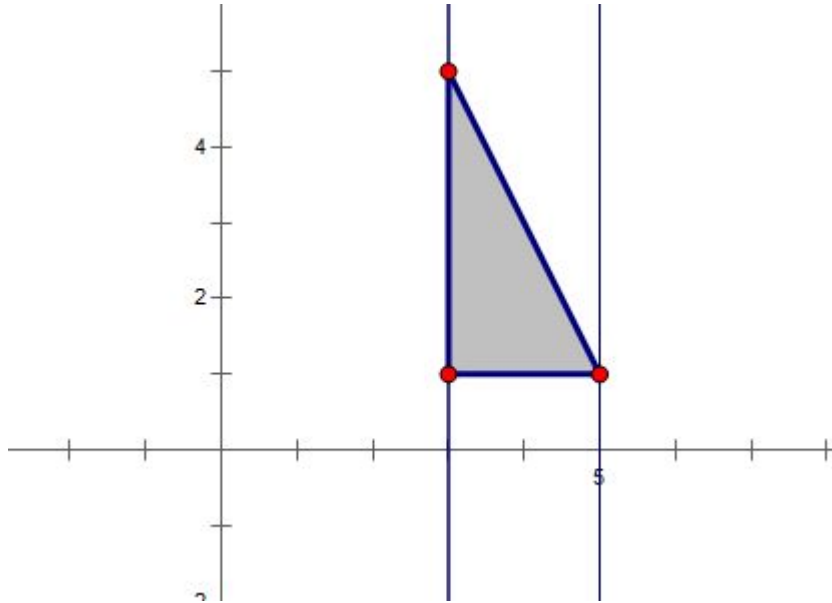
【题目大意】

给出一个 n 个点的多边形，你可以任意旋转它，使得它在 x 轴上的投影长度尽量小。问旋转后它在 x 轴上的最小投影长度。

【题目关键字】

枚举，计算几何。

【算法讨论】



如图，过旋转后多边形顶点中 x 坐标最小/最大的点做两条 x 轴的垂线，至少有一条线上有两个点，否则总可以旋转图形使得投影长度更小。

枚举这两个点，判断其余 $n-2$ 个点是否落在所成直线的同一边。然后将多边形旋转使得所成直线与 x 轴垂直，多边形顶点中与直线垂直距离最大的点即为答案。

【时空复杂度】

时间复杂度: $O(n^3)$

空间复杂度: $O(n)$

【题目编号】

2012 A Asteroid Rangers

【题目名称】

Asteroid Rangers

【题目大意】

空间中有 n 个点，每个点以固定的速度和方向不停的移动。给出每个点开始的位置和每个点在 x,y,z 三维上的速度，问从一开始到最后， n 个点组成的最小生成树总共变化了多少次。任意两个点在任意时刻不重合，且当一棵生成树成为最小生成树后在至少 10^{-6} 的单位时间内最小生成树不会变化。

【题目关键字】

最小生成树，数学

【算法讨论】

很显然，最小生成树发生改变当且仅当一对点 (i,j) 的距离开始变得比另一对点 (x,y) 小为止。枚举 (i,j) 和 (x,y) ，解一个一元二次方程，求出所有可能变化的时刻。这一步的复杂度是 $O(n^4)$ 的。

接下来枚举每个变化的时刻，当且仅当当前时刻 (i,j) 和 (x,y) 这两条边一条为树边，另一条为非树边，且该时刻前树边小于非树边，这个时刻最小生成树可能改变。对于这样的时刻，暴力求一遍最小生成树，判断它是否会改变即可。

求最小生成树用 **prim** 算法实现，时间复杂度为 $O(n^2)$ ，总的复杂度为 $O(n^6)$ 。看起来会超时，但因为最小生成树可能改变的要求很苛刻，实际可能变化的时刻非常少，因此只要常数不太差，即可通过这道题。

【时空复杂度】

时间复杂度： $O(n^6)$

空间复杂度： $O(n^4)$

【题目编号】

2012 B Curvy Little Bottles

【题目名称】

Curvy Little Bottles

【题目大意】

Jill 有一个瓶子,瓶子是由一条从 $X=X_{low}$ 到 $X=X_{high}$ 的多项式曲线 P 绕 X 轴旋转一周构成。因此, X 轴与一条穿过瓶子底面中心的垂线重合。瓶底在 $X=X_{low}$ 由一个实心圆构成,瓶口在 $X=X_{high}$, 瓶口是敞开的。现在 Jill 要从平底开始打上最多不超过 8 个标记, 要求两个标记之间瓶子的体积增量恰好为 inc , 要求每个标记与 X_{low} 的距离, 标记不能打在超过 X_{high} 的距离。

给出多项式 P , X_{low}, X_{high} , inc , 多项式次数不超过 10, 每项系数不超过 100, $-100 \leq X_{low} < X_{high} \leq 100, X_{high} - X_{low} > 0.1, 1 \leq inc \leq 500$ 。

【题目关键字】

数学

【算法讨论】

令多项式 $S = \pi * P * P$, 显然瓶子在区间 $[a, b]$ 上的增量即为 S 在区间 $[a, b]$ 上的定积分。

因为 X_{low} 到第 i 个标记的体积增量为 $i * inc$, 因此我们可以二分每个标记的高度, 然后依次输出即可。

求多项式 S 的时间复杂度为 $O(n^2)$, 二分求单个标记的复杂度为 $O(n \log H)$, 要求得标记数量为 $O(1)$, 因此总的时间复杂度为 $O(n^2 + n \log H)$ 。

【时空复杂度】

时间复杂度: $O(n^2 + n \log H)$, H 为瓶子的高度。

空间复杂度: $O(n)$

【题目编号】

2012 C Bus Tour

【题目名称】

Bus Tour

【题目大意】

给出一个 n 个点， m 条边的无向图。 n 个点编号 $0 \sim n-1$ ，同时每条边长度不一。要求你从节点 0 出发，遍历其它 $n-2$ 个节点后再到终点，然后再从终点出发，遍历其它 $n-2$ 个节点后在回到起点。要求两次遍历的前 h 个节点相同， $h=(n-2)/2$ 向下取整。 $n \leq 20$

【试题关键字】

状态压缩，DP

【算法分析】

首先对原图做一次 floyd 求出任两点最短路。然后进行状态压缩动态规划。

$f[r][s]$ 表示从起点出发，访问过的节点集合为 s ，最后一个访问的节点为 r 的最短路。

$g[r][s]$ 表示从终点出发，访问过的节点集合为 s ，最后一个访问的节点为 r 的最短路。

s 用二进制数表示。状态 (r,s) 的转移为枚举一个不在集合 s 中的节点 k ，然后当前答案加上 r 到 k 的最短路，向状态 $(k,s+2^k)$ 转移。

求出 f,g 后，枚举遍历的前 h 的节点组成的集合 s_0 ，令它的补集为 s_1 ，枚举连接两个集合的边 (j,k) ，即可求出答案。Dp 的时间复杂度为 $O(n \cdot 2^n)$ ，最后合并求解的复杂度为 $O(C(n-2,h) \cdot n^2)$ ，因此总的时间复杂度为 $O(n^2 \cdot 2^n)$ 。时限为 10s，可以解决问题。

【时空复杂度】

时间复杂度： $O(n^2 \cdot 2^n)$

空间复杂度： $O(n \cdot 2^n)$

【题目编号】

2012 D Fibonacci Words

【题目名称】

Fibonacci Words

【题目大意】

斐波那契 01 字符串的定义如下

```
F(n) =  
{  
0    if n = 0  
1    if n = 1  
F(n-1)+F(n-2) if n >= 2  
}
```

这里+的定义是字符串的连接。F(n)的前几个元素如下：

```
F(0)=0  
F(1)=1  
F(2)=10  
F(3)=101  
F(4)=10110  
F(5)=10110101  
F(6)=1011010110110  
F(7)=10110101101101010101  
F(8)=10110101101101010101101010110110  
F(9)=1011010110110101010110101011010110101101010101010101
```

给定一个模式串 p 和一个数 n，问 p 在 F(n)中出现了多少次。n<=100，多组数据，数据组数<=30,len(p)<=100000，答案严格小于 2^{63} 。

【题目关键字】

字符串匹配，KMP

【算法讨论】

令 ans(n)为 p 在 F(n)中出现了多少次。ans(n)由 ans(n-1)，ans(n-2)，F(n-1)的后 len(p)-1 个字符+F(n-2)的前 len(p)-1 个字符组成的字符串中 p 出现的次数这三个部分累加起来。

对于长度小于 len(p)的 F(n),显然有 ans(n)=0。令 x 为最小的满足 len(F(x))>=len(p)。F(x)接下来的几个字符串为 F(x+1),F(x+1)+F(x),F(x+1)+F(x)+F(x+1)，可以发现 x+1 之后的 ans(n)中,F(n-1)的后 len(p)-1 个字符+F(n-2)的前 len(p)-1 个字符组成的字符串最多只有 F(x)+F(x+1)和 F(x+1)+F(x+1)两种，把两种分别求出即可。这一部分可以用 kmp 完成。

递推的复杂度加上模式匹配的复杂度为 $O(n+\text{len}(p))$ ，因此总的复杂度为 $O(T*(n+\text{len}(p)))$ ，T 为数据组数，可以通过本题。

【时空复杂度】

时间复杂度： $O(T*(n+\text{len}(p)))$ ，T 为数据组数。

空间复杂度： $O(n+\text{len}(p))$

【题目编号】

2012 E infiltration

【题目名称】

infiltration

【题目大意】

给出一个 n 个节点的有向图。任两点间都有一条有向边。现在要求你选择尽量少的点，使得任意一个未被选的点都被至少被选择的点的出边指向。 $n \leq 75$

【试题关键字】

迭代加深搜索，状态压缩

【算法分析】

首先我们可以用贪心法证明答案不超过 6。整个图共有 $n*(n-1)/2$ 条有向边，因此出度最大的一个点的度数不小于 $(n-1)/2$ 条。每次选择出度最大的点，将它和它的出边指向的点删除，得到一个大小不超过 $n-(n-1)/2-1$ 大小的子图。然后再对子图重复上述操作。每次图的大小至少减少一半，对于最大数据 $n=75$ ，最优解不会超过 6。

因此我们可以发现答案不会很大。所以我们枚举答案大小然后搜索方案即可。如果不存在不超过 5 的方案，就可以直接调用前面提到的贪心算法。最后为了减少常数，可以用把图的状态二进制数表示来快速判断答案是否合法。

【时空复杂度】

时间复杂度： $O(C(n,5))$

空间复杂度： $O(n^2)$

【题目编号】

2012 I A Safe Bet

【题目名称】

A Safe Bet

【题目大意】

给出一个 $R \times C$ 大小的矩形网格，给出 M 面/镜子和 N 面\镜子的位置。有一束激光从网格第一行左侧水平射入，当激光集中镜子时会被镜子以 45 度角反射。询问激光是否能从最后一行右侧水平射出。以及如果不能，在某个添加一面/或\镜子后能使激光从最后一行右侧水平射出位置有几个以及这些位置中字典序最小的一个。
 $1 \leq R, C \leq 1000000, 1 \leq N, M \leq 100000$ 。多组数据。

【试题关键字】

平衡树,线段树

【算法分析】

首先，从第一行左侧水平射入一束激光，模拟激光的行走路径。方法为每一行、每一列建立一个平衡树，节点为该行或该列的所有镜子，关键字为所在列数或行数。然后对于激光当前照射到的镜子，根据激光的方向，找到同一行或同一列相邻的镜子即为激光下一个照射到的目标，如果没有，就是射出边界，模拟结束。这里的平衡树也可以在把每一行或每一列镜子用线性表存下来后用快速排序预处理+二分查找代替。总的复杂度为 $O((N+M)\log(N+M))$

根据第一次模拟的结果，可以判断激光的射出位置。如果是从最后一行右侧水平射出，那么算法结束。否则从最后一行右侧水平射入一束激光，并用之前的办法模拟激光的行走路径。显然，如果两者的路径没有交点，那么显然无解；如果有交点那么通过在交点放置合适方向的镜子，就能使左上方射入的光束之后的轨迹为右下方射入的光束轨迹重合。所以两者路径的交点数量即为答案。

因为激光只能以 45 度被折射，所以两者路径的水平部分和水平部分，垂直部分和垂直部分之间没有交点。水平部分和垂直部分之间的交点可以用线段树加扫描线在 $O((N+M)\log C)$ 的时间内解决。最后算法总的时间复杂度为 $O((N+M)\log(N+M) + (N+M)\log C)$ 。

【时空复杂度】

时间复杂度: $O((N+M)\log(N+M) + (N+M)\log C)$

空间复杂度: $O(N+M+C)$

【题目编号】

2012 K Stacking Plates

【题目名称】

Stacking Plates

【题目大意】

给出 n 堆盘子，每堆的盘子从下到上大小不递增。现在要求通过下面两种操作把所有盘子合并成从下到上从大到小排好序的一堆。

1, 拆分(Split): 可以将一个盘堆堆顶任意数目的盘子抬起, 并放置在原堆的一侧, 使堆一分为二。

2, 合并(Join): 可以将一个盘堆放置在另一堆的堆顶, 前提是在上方的堆最底层的盘子尺寸不大于在下方的堆最顶层的盘子尺寸。

注意不能将一个盘堆堆顶上的若干盘子直接移动到另一个盘堆的堆顶, 必须先拆分, 后合并。给定盘堆若干, 你要求出将这些盘堆通过两种操作合并成一堆的最少次数。下面的图是对输入输出样例的解释。

$n \leq 50$, 每堆盘子数目 ≤ 50 , 盘子尺寸 ≤ 10000 , 多组数据, 数组组数 ≤ 400 。

【试题关键字】

动态规划

【算法分析】

因为合并 n 堆盘子成为一堆需要 $n-1$ 次合并, 同时每次拆分必定对应一次合并。因此我们可以把问题转化成: 把 n 个递减数列分成若干组, 使得它们可以通过一定顺序组合成一个递减的数列。最后答案为分成的组数 $\times 2 - n - 1$

这个问题可以用一个 dp 解决。把大小相同的盘子看成同一种, $f[i][j]$ 表示前 i 大的盘子, 最后一个来自第 j 堆的最优方案。枚举上一个盘子属于的堆 k , 则如果 k 可以放在第 $i-1$ 种盘子的末尾和第 i 种盘子的开头, 则 $f[i][j] = \min(f[i-1][k] + \text{拥有第 } i \text{ 种盘子的堆数} - 1)$, 否则 $f[i][j] = \min(f[i-1][k] + \text{拥有第 } i \text{ 种盘子的堆数})$ 。而 k 需要满足的条件, 就是第 k 堆既拥有第 i 种盘子, 又拥有第 $i-1$ 种盘子, 且除非第 i 种盘子来源仅有 j , 否则 $j \neq k$ 。答案即为 $\min(f[m][i]) (1 \leq i \leq n)$ m 为不同的盘子尺寸数目。

这样的状态数是 $O(n^3)$, 一次转移 $O(n)$, 复杂度为 $O(n^4)$ 。但因为对于 k 仅有 $j \neq k$ 的约数, 所以可以用两个数一个记录 $k < j$ 的 $f[i-1][k]$ 的最小值, 另一个记录 $k > j$ 的 $f[i-1][k]$ 的最小值。这样就可以将一次转移降低为 $O(1)$, 总的时间复杂度降为 $O(n^3)$, 可以通过本题。

【时空复杂度】

时间复杂度: $O(T \times n^3)$, T 为数据组数。

空间复杂度: $O(n^3)$

【题目编号】

2012 L Takeover Wars

【题目名称】

Takeover Wars

【题目大意】

有 X,Y 两个玩家, X 有 n 个子公司, 第 i 个子公司价值为 A_i ; Y 有 m 个子公司, 第 j 个子公司价值为 B_j 。现在每个玩家轮流操作, 每个玩家每个回合可以选择下面三种操作之一:

1, 合并自己的两个子公司为新的子公司, 新的子公司价值为原来两个子公司价值之和。

2, 用自己的某个子公司取兼并对方一个价值不超过自己这个子公司价值的子公司。兼并后自己的子公司价值不变, 但对方被兼并的子公司会消失。

3, 如果上面两种操作都不能进行, 玩家这个回合就不必进行任何操作。

为了操作方便, 无论游戏如何进行, 游戏过程中不会出现价值相同的子公司。当一个玩家失去所有子公司的时候, 他就输了。现在 X 先手, 问最后 X 能否获胜。 $n, m \leq 10^5$

【试题关键字】

排序, 贪心

【算法分析】

首先有一个很明显的贪心策略, 就是如果兼并对方的子公司, 那么一定会兼并能兼并的子公司中价值最大的一个。

另一个贪心策略是, 如果不能兼并对方价值最大的子公司, 那么兼并行为就没有意义。否则接下来每一回合, 对方就会兼并自己最大的子公司, 直到自己合并出价值更大的子公司为止, 因此最优策略是现在马上进行合并。

结合上面两个贪心策略, 可以发现如果合并的话, 一定是合并自己最大的两个子公司。因为下一回合对方如果进行兼并, 一定会兼并自己最大的子公司。因此自己做的就是使得自己最大的子公司尽量大使得对方不进行兼并。同时, 如果当前自己最大的两个子公司价值和小于对方的最大子公司, 那么显然对方只要每回合不停的兼并自己最大的子公司即可获胜, 此时显然是必败局。

那么我们枚举 X 第一个回合是兼并还是合并, 分情况讨论。

1, 兼并。设 X 最大和次大的子公司为 A_1, A_2, A_3 , Y 最大的三个子公司为 B_1, B_2, B_3 。若 $A_1 > B_1$ 那么第一回合 X 可以进行兼并。Y 此时只能选择合并, 最大子公司为 $B_2 + B_3$ 。若 $A_1 < B_2 + B_3$, 否则接下来每一回合双方都会不停的合并, 直到一方能兼并对方的最大子公司为止, 这时能兼并另一方最大子公司的即为胜者。

2, 合并。经过合并后, 若 $A_1 + A_2 < B_1$, 那么先手必败, 否则 Y 只会选择合并操作。接下来每一回合双方都会不停的合并, 直到一方能兼并对方的最大子公司为止这时能兼并另一方最大子公司的即为胜者。

最后主要复杂度在于每次合并最大的两个子公司。因为到决出胜负前都没有出现兼并, 因此每次合并的子公司一个是上次合并后新产生的公司, 另一个是剩下最大的子公司, 只要排序后便可快速处理。主要复杂度集中在排序上, 因此时间复杂度为 $O(n \log n)$, 可以通过本题。

【时空复杂度】

时间复杂度: $O(n \log n + m \log m)$

空间复杂度: $O(n + m)$

【题目编号】

2013 A Self-Assembly

【题目名称】

Self-Assembly

【题目大意】

给出 $n(n \leq 40000)$ 种正方形，每种正方形四条边上各有一个连接标识。连接标识有两种：

- 一个大写字母(A,...,Z)加上一个‘+’号或一个‘-’号。两条边能并在一起当且仅当两者的字母相等且符号相反。比方说，‘A+’与‘A-’兼容，但与‘A+’或‘B-’不兼容
- 两个零‘00’。这条边将不和任意一条边兼容（包括‘00’）。

假设每种正方形都有无限个，并且每个正方形都可以旋转和翻转。当正方形将自身组成一个结构体时，相互贴合的边必须能够相互兼容，当然，无论边的连接标识是什么，它都可以不与另外边贴合。

现在要求判断给出的 n 种正方形能否组成无限大的结构体。

【题目关键字】

Floyd

【算法讨论】

建立一个 52 个节点的图，节点分别是 A+,A-,B+,B-……Z+,Z-。若两个标识符能兼容，称他们互为兼容标识符。对于每种给出的正方形，从每条边的标识符对应节点往另外三条边的兼容标识符的对应节点连一条有向边。然后用 floyd 判断图是否有环即可。

【时空复杂度】

时间复杂度：O(n)

空间复杂度：O(1)

【题目编号】

2013 B Hey, Better Bettor

【题目名称】

Hey, Better Bettor

【题目大意】

有一个赌场正在提供以下的优惠：你可以在这个赌场里赌很多次。当你赌完之后，如果你的总资金减少了，这个赌场会把你损失的 $x\%$ 退还给你。显然，如果你赚了，你可以留下所有的钱。这个服务没有时间限制和金钱限制，但是你能只能赎回一次。

为简单起见，假定所有的赌局会花费 1 块钱，如果你赢得了赌局，会返还 2 块钱。现在假设 x 等于 20。如果你一共进行了 10 次赌局，且只赢得了其中 3 个，那么你会一共会损失 3.2 块钱。如果你赢得了 6 次赌局，你会赢得 2 块钱。

给定 x 和赢得赌局的概率 $p\%$ ，写一个程序，计算在最优策略下你的最大期望收益。

$$0 \leq x < 100, 0 \leq p < 50$$

【题目关键字】

数学

【算法讨论】

最优策略很明显是选择两个非负整数 a, b ，当输了 a 元或赢了 b 元的时候停止。

令 $f(i)$ 为当前赢了 i 元的情况下最后的期望收益。

显然，有 $f(i) = pf(i+1) + (1-p)f(i-1)$ ，变形得 $f(i+1) = \frac{1}{p}f(i) + \frac{p-1}{p}f(i-1)$ ，

令 $g(i) = f(i-a)$ ，那么有 $g(i) = \frac{1}{p}g(i-1) + \frac{p-1}{p}g(i-2)$ 。这是一个简单的线性其

次递推式，可以尝试解它。特征根方程为 $px^2 - x + 1 - p = 0$ ，解得 $x = \frac{p-1}{p}$ 或 1。令

$c_1\left(\frac{p-1}{p}\right)^i + c_2$ ，将 $g(0), g(a+b)$ 代入即可求出 c_1, c_2 。

推出 $g(i)$ 的通项公式后，答案即为 $g(0)$ 。经过观察发现，当 a 确定是， $g(0)$ 的取值是一个关于 b 的单峰函数。同时 $\max(g(a, b))$ 的也是一个关于 a 的单峰函数。因此可以先三分 a ，然后三分 b ，取最优值即可。

【时空复杂度】

时间复杂度： $O(\log n * \log n)$

空间复杂度： $O(1)$

【题目编号】

2013 C Surely You Congest

【题目名称】

Surely You Congest

【题目大意】

给出一个 N 个点， M 条边的无向图，通过每条边要花费不同的时间。开始的时候有 C 个人在不同的节点同时向节点 1 出发。当两个人在相同的时间，从同一方向，开始沿着相同的边移动时，就会出现堵塞。每个人都要求走最短路到节点 1。要求问你最多能安排多少人在不堵塞的情况下走最短路到达节点 1。 $N \leq 25000, M \leq 50000, C \leq 1000$

【题目关键字】

最短路，网络流

【算法讨论】

首先以节点 1 为原点，求出一个最短路网络。因为每个人都只会走最短路，所以不在最短路网络上的边是没有意义的。

在最短路网络上从任意一个节点出发，走任意一条路径到节点 1 都是最短路。因此我们把节点按到节点 1 的距离分层。在同一层的人通过最短路网络上的边到达同样节点的时间相同，而不在同一层的人，互相之间没有影响。因此我们每次将出发节点到节点 1 距离相等的人取出来在最短路网络上求一次最大流，将每次的答案累加即为答案。

因为网络流上界很宽松，这题时限有 10s，因此暴力用 Dinic 跑 c 次最大流即可通过本题。

【时空复杂度】

时间复杂度：预处理 $O((N+M)\log M)$ ，Dinic $O(C \cdot N^2 \cdot M)$

空间复杂度： $O(M)$

【题目编号】

2013 D Factors

【题目名称】

Factors

【题目大意】

一个最基本的算数法则就是大于 1 的整数都能用 1 个或多个素数相乘的形式表示出来。当然，可以安排出多种的质因子排列方案，例如： $10=2*5=5*2$ $20=5*2*2=2*5*2=2*2*5$

让我们用 $f(k)$ 表示 k 的质因子排列方案数，如 $f(10)=2$ ， $f(20)=3$ 。

给你一个正整数 n ，至少有一个 k 使得 $f(k)=n$ ，我们想知道最小的 k 是多少。 $n<2^{63}$ ，数据保证存在 $k<2^{63}$ 使得 $f(k)=n$ ，多组数据，数据组数不超过 1000。

【题目关键字】

搜索，组合数

【算法讨论】

不难发现，最小的 k 的质因数分解一定是最小的若干个质数，且这些质数的指数随着质数的增大单调不升。

另外因为数据保证 $k<2^{63}$ ，所以不同的质因数不会超过 16 个，否则 k 就会大于 2^{63} 。

令 $k=a_1^{p_1}*a_2^{p_2}*\cdots*a_N^{p_N}$ ，那么有：

$$f(k)=C(p_1+p_2,p_2)*C(p_1+p_2+p_3,p_3)*\cdots*C(p_1+p_2+\cdots+p_N,p_N)$$

所以要计算 $f(k)$ ，要预处理组合数。暴力搜索 16 个质数每个数的指数。搜索过程中记录当前确定指数的质数的幂的乘积，并保证这个乘积是 n 的约数即可。因为这题搜索条件约束非常严格，所以实际解的个数很少，可以在时限内通过测试数据。

【时空复杂度】

时间复杂度：最坏情况 $O(C(77,15))$

空间复杂度： $O(1)$

【题目编号】

2013 E Harvard

【题目名称】

Harvard

【题目大意】

给出 b 个内存库，编号 $0 \sim b-1$ ，每个内存库可以存储 s 个变量，再给出一个选择寄存器 BSR。每次可以用一个指令来直接访问选择 0 号内存库的一个变量或 BSR 所指向内存库的一个变量。如果需要访问的变量既不在 0 号内存库也不在 BSR 所指向内存库，就需要先用一个指令令 BSR 指向该变量所在内存库，再用一个指令访问该变量。

一个程序是一个操作的序列，每个操作是：

- 一个变量访问操作，写作 V_i ， i 是一个正整数。

- 一个循环操作，写作 $R_n \langle \text{program} \rangle E$ ， n 是一个正整数， $\langle \text{program} \rangle$ 是一个任意的程序。这个操作等价于依次执行 n 遍 $\langle \text{program} \rangle$ 。

开始的时候 BSR 的值为 `undefined`，直到一条命令显式的设定了它的值。现在要求你给出一个变量到内存库的映射，使得运行程序总的指令数（包括访问指令和设置 BSR 的指令）最小。变量个数不超过 13， $b, s \leq 13$ ，操作序列长度不超过 1000，循环次数不超过 10^6 ，程序访问变量的次数不超过 10^{12} 。

【题目关键字】

模拟，搜索，模拟栈

【算法讨论】

因为访问次数是固定的，所以只需要使得设置 BSR 操作次数最小即可。

因为 0 号内存库的元素不需要设置 BSR，所以先枚举哪些元素在 0 号内存库中。

接下来把程序中访问 0 号内存库中变量的操作删除，然后模拟程序运行，求出数组 $\text{data}[i][j]$ ，表示程序中有多少次在访问变量 i 后紧接着访问了变量 j 。

然后暴力枚举剩下的变量分配到哪个内存库，然后把所有 i, j 不在同一个内存库的 $\text{data}[i][j]$ 累加，即为设置 BSR 的操作次数。分配内存的方案数 $C(m+b-1, b-1)$ ，乘以统计的复杂度，时间复杂度为 $O(C(m+b-1, b-1) * m * m)$ ， m 为变量个数。

接下来是模拟程序运行的部分，因为程序范围变量的有 10^{12} 次，暴力模拟会超时。对每个子程序 k 求出 $\text{data}_k[i][j]$ 表示有程序中有多少次在访问变量 i 后紧接着访问了变量 j ，以及子程序 k 第一个访问的变量 `first`，最后一个访问的变量 `last`。然后根据 data_k ，`first`，`last`，就可以快速的求出子程序对主程序的 data 数组的影响，时间复杂度 $O(n * m * m)$ 。这里需要用到模拟栈，属于基础内容，不再叙述。

【时空复杂度】

时间复杂度： $O(C(m+b-1, b-1) * m * m)$ ， m 为出现的不同变量个数

空间复杂度： $O(L)$ ， L 为操作序列长度。

【题目编号】

2013 F Low Power

【题目名称】

Low Power

【题目大意】

有 n 个机器，每个机器有 2 个芯片，每个芯片可以放 k 个电池。

每个芯片能量是 k 个电池的能量的最小值。

两个芯片的能量之差越小，这个机器就工作的越好。

现在有 $2nk$ 个电池，已知它们的能量，我们要把它们放在 n 个机器上的芯片上，使得所有机器的能量之差的最大值最小。

$2nk \leq 10^6, 1 \leq \text{芯片能量} \leq 10^9$

【题目关键字】

二分、贪心

【算法讨论】

将所有电池按能量值 p_i 从小到大排序。对于一个机器，它的能量差显然是它所有电池中最小的两块的能量差。同时由调整法可证，排序后，这两块电池一定是相邻的两块 p_i, p_{i+1} 。

那么接下来二分答案 d 。然后按 p_i 从大到小选择每块芯片最小的两块电池。若当前选择 p_i 和 p_{i+1} 作为第 t 个机器最小的电池，那么就等于要将 $2nk-i+1$ 块电池分给 t 个机器。若 $2 \cdot t \cdot k \leq 2nk-i+1$ ，那么就可以选择。这样能选就选，最后判断是否选择了全部 k 个机器即可。

【时空复杂度】

时间复杂度： $O(nk \log P)$

空间复杂度： $O(nk)$

【题目编号】

2013 Н М а т р ё ш к а

【题目名称】

М а т р ё ш к а

【题目大意】

有 $n(n \leq 500)$ 个俄罗斯套娃摆成一排，套娃大小都是整数。你需要重新组装套娃集，你既不知道套娃集的数量，也不知道某个套娃集内玩偶的数量，你只知道一个完好的套娃集内的玩偶大小是从 1 到某个数字 m 。

在组装套娃集时，你必须遵守下列规则：

1. 你只能将一个玩偶或者套娃集放入一个更大的玩偶中
2. 你只能把相邻两个俄罗斯套娃组合在一起
3. 已经被合并的玩偶是不能再重新拆出来的。

唯一需要耗时的部分为打开一个玩偶并马上关上它。所以你要尽可能少的做这种操作。比如说：合并 $[1,2,6]$ 与 $[4]$ ，你需要将大小为 4 和 6 的两个玩偶拆开。合并 $[1,2,5]$ 与 $[3,4]$ 代价为 3。

求将 n 个玩偶重新拼成一些完好的俄罗斯套娃的最小代价。

【题目关键字】

dp

【算法讨论】

首先我们需要快速算出合并两个套娃集的代价。两个套娃集可以合并当且仅当它们没有相同大小的套娃。显然，一个套娃集中的一个套娃要打开的充要条件是它的大小超过另一个套娃集中最小的套娃。求区间最小值和一个区间中在一定范围内的数的个数是经典问题。设 $h[i][j]$ 为区间 $[i,j]$ 中最小的套娃大小， $s[i][j]$ 为区间 $[1,i]$ 中大小不超过 j 的套娃个数。两者可以在 $O(n^2)$ 的时间内求出。然后就可以 $O(1)$ 算出合并代价。

设 $g[i][j]$ 为把第 i 个到第 j 个套娃合并成一个套娃集的最小代价。当第 i 个到第 j 个套娃中有相同大小的套娃是， $g[i][j]$ 赋值为正无穷。否则 $g[i][j] = \min(g[i][k] + g[k+1][j] + \text{合并所需代价})$ 。这一步的时间复杂度是 $O(n^3)$

接下来设 $f[i]$ 为将前 i 个套娃合并成若干个完整套娃集的最小代价。 $f[i] = \min(f[j-1] + g[j][i])$ ，若第 j 个~ i 个套娃的大小恰好为从 1 到某个数字 m 。这一步的时间复杂度为 $O(n^2)$ 。问题解决。

【时空复杂度】

时间复杂度： $O(n^3)$

空间复杂度： $O(n^2)$

【题目编号】

2013 I Pirate Chest

【题目名称】

Pirate Chest

【题目大意】

给出一个 $n*m$ 个网格状的池塘。要在其中沉入一个一边不超过 a ，另一边不超过 b 的宝箱。当宝箱沉入水中时，它会一直下沉直到碰到池底。沉底时，宝箱的顶面会和池塘的表面平行，宝箱的边缘会和网格对齐。宝箱排开了一部分水，这会使池塘的水位上升（即使被宝箱排开的水没有空隙上升也会这样）。四周的悬崖足够高，所以水不会溅出来。当然，由于宝箱不能被别人看到，宝箱的顶面必须严格低于水面，且宝箱的长，宽，高均为正整数。给出池塘每个方格的深度 $d(i,j)$ ，求能沉入的宝箱的最大体积。

【题目关键字】

枚举，贪心，栈优化

【算法讨论】

设宝箱地面为 $w*l$ ，沉入宝箱区域的最小深度为 d ，算出宝箱顶面严格小于水面的充要条件为 $nmd < nmd + wlh$ ，变形可得 $h < \frac{nmd}{nm - wl}$ ，又因为 h 为正整数，所以 h 的值为 $\frac{nmd - 1}{nm - wl}$ 向下取整。

考虑枚举 w 和 d ，不难发现 w 和 d 固定的情况下， l 越大， h 越大。因为目标为最大化 wlh 所以 l 应取最大值。

枚举 w 和选择区域的第一行，这样问题就变成一维的情况：给定 $H_1, H_2, H_3 \dots H_n$ ，选择一个区间。假设区间最小值为 H_i ，显然最大区间的为向左找到第一个小于 H_i 的数，向右找到第一个小于 H_i 的数，这中间的区域即为最小值为 H_i 的最大区间。这样一位的问题可以用单调栈在 $O(n)$ 的时间内解决。

另一个问题是 H_i 如何求出。令 $h(i,j,w)$ 为 $d(i,j), d(i,j+1) \dots d(i,j+w-1)$ 中的最小值，可以发现 $h(i,j,w)$ 与 $h(i,j,w-1)$ 的计算范围相比只是多了一个 $d(i,j+w-1)$ ，因此我们可以在 $O(n^3)$ 的时间内算出所有 h 值，然后要求 H_i 的时候 $O(1)$ 调用即可。

【时空复杂度】

时间复杂度： $O(n^3)$

空间复杂度： $O(n^2)$

【题目编号】

2013 J Pollution Solution

【题目名称】

Pollution Solution

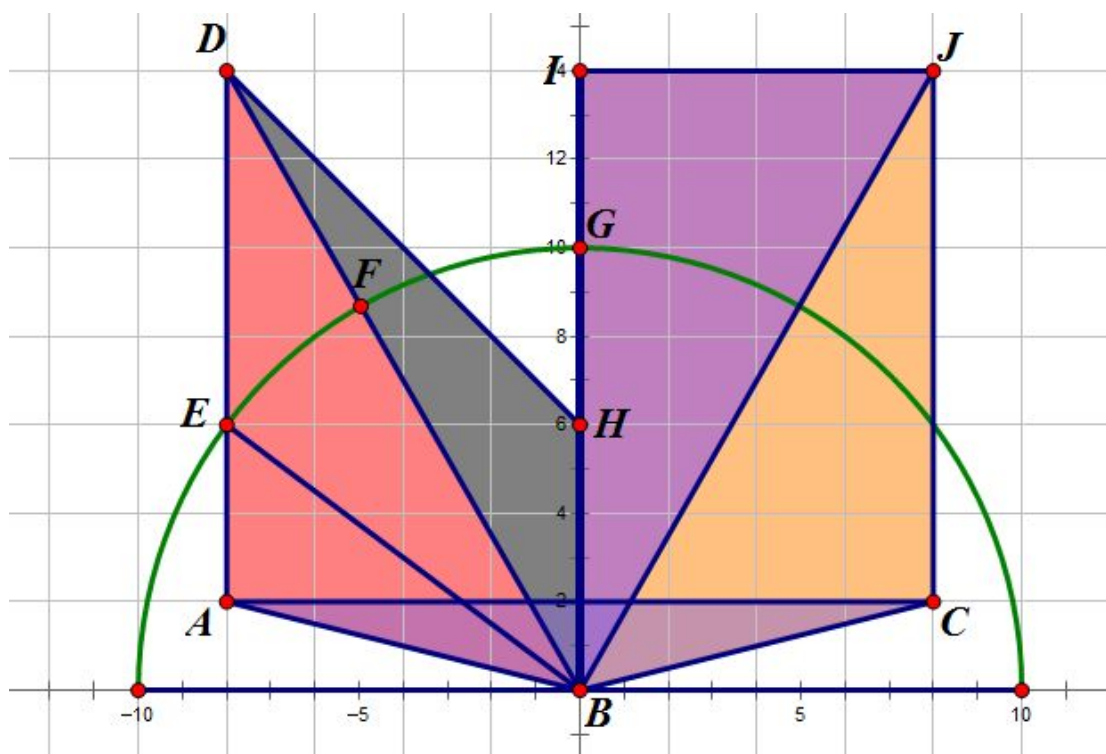
【题目关键字】

计算几何

【题目大意】

按逆时针给出一个 $n(n \leq 100)$ 个点的简单多边形。多边形在 x 轴以上，再给出一个以 $(0,0)$ 为圆心，半径为 r 的圆。求圆在 x 轴以上的部分与多边形的交点。

【算法讨论】



如图，将给定多边形的顶点 $(0,0)$ 连线。然后对于多边形的每条边，将该边与边上两个点和 $(0,0)$ 连线段组成的三角形与半圆求交。然后判断两个点和 $(0,0)$ 连线段的方向，如果是顺时针则减去，逆时针则加上。最后结果即为答案。如上图，最后三角形 ABC 部分就会被减去。

而三角形与半圆的相交部分一定是由三角形和扇形组成。先求出线段与圆弧交点，然后根据交点剖分即可。以上图三角形 ABD 为例，相交部分可以以 BE 为界分为三角形 ABE 和扇形 EBF 两个部分。

至于求圆弧和线段交点的部分可以用二分查找或解方程，求三角形面积可以用叉积。最后总的时间复杂度为 $O(n \log n)$ 或 $O(n)$ 。

【时空复杂度】

时间复杂度: $O(n \log n)$

空间复杂度: $O(n)$

【题目编号】

2013 K Up a Tree

【题目名称】

Up a Tree

【题目大意】

Anatoly Cheng McDougal 写了一段树的先序遍历（如左下），然后把随后，他简单的剪切和粘贴的这段代码，然后将输出语句调整到正确的位置，并将过程（函数）重命名。但是，他忘记重新书写将会在内部被调用的过程（函数）名，从而导致生成了如下有问题的树的先序，中序和后序遍历代码。

前序遍历	中序遍历	后序遍历
<pre>void prePrint(TNode t) { output(t.value); if (t.left != null) prePrint(t.left); if (t.right != null) prePrint(t.right); }</pre>	<pre>void inPrint(TNode t) { if (t.left != null) prePrint(t.left); output(t.value); if (t.right != null) prePrint(t.right); }</pre>	<pre>void postPrint(TNode t) { if (t.left != null) prePrint(t.left); if (t.right != null) prePrint(t.right); output(t.value); }</pre>

当输出结果不正确之后，Anatoly 随机地改变三个过程中的调用，希望这能奏效。显然，情况更糟了。

现在给出 Anatoly 程序的前序、中序和后序遍历，要求重构这个树和他的程序经过修改后 Anatoly 的程序满足以下条件：

1. 输出语句在各个过程（函数）已经在正确的位置。（例如，在中序遍历中，输出语句在两个递归调用中间）。
2. 在涉及到这三个过程（函数）的六次递归调用中，恰好有两个调用为先序遍历 (prePrint)，恰好两次为中序遍历 (inPrint)，还有恰好两次为后序遍历 (postPrint)，虽然有可能在错误的过程（函数）里。

你需要重构出所有可能的 Anatoly 的代码。另外，对于所有这样的重构，请您找到能够产生这种输出的，字典序最小的树。

树的前序、后序、中序遍历为长度 ≤ 26 的由大写字母组成的字符串。

【题目关键字】

搜索，记忆化

【算法讨论】

第一步枚举 prePrint, inPrint, postPrint 三个过程中调用 prePrint, inPrint, postPrint 的先后顺序。然后用给出的三个输出进行搜索，搜索的参数为错误前序、中序、后序遍历程序对应的输出，由于它的递归调用可能错的，不能保证到任何一个子树，都存在每种输出，但至少存在一个输出。分情况讨论：若前序遍历或后不为空遍历，那么根节点就确定了，否则需要枚举根节点。根节点确定后，若中序遍历存在，那么左右子树的大小也就确定了。左右子树的大小确定后，根据前面枚举的三个错误遍历的调用顺序，就可以分别求出左右子树的三种错误遍历的输出（不能保证全部存在）。然后分别递归处理左右子树即可。

搜索过程中有相当多的重复状态，因此可以用一个 map 把搜索过的状态保存下来，再次遇到后直接调用即可。

【时空复杂度】

时间复杂度： $O(n^7)$

空间复杂度： $O(n^6)$

