

2006 ACM/ICPC

Xi'an Site

Judge's Comments

(Chinese Version)

Rujia Liu



Credits

- Problem Authors
 - Problem A~I: **Rujia Liu**
 - Problem J: **Shahriar Manzoor**
- Alternate Solutions
 - Problem A~J: **Derek Kisman**
 - Problem J: **Rujia Liu**
- **You may use this material if you like, but do NOT modify anything**

On-site Competition (5 hours)

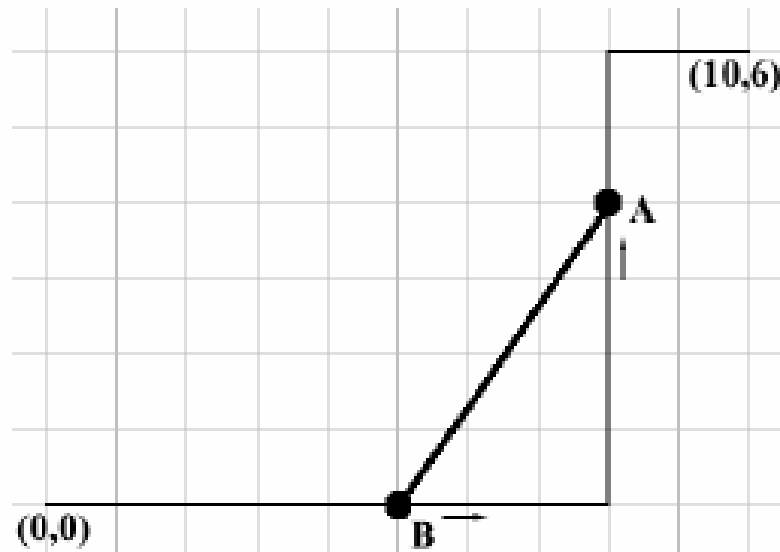
- Time limits: 20s, 20s, 20s, 30s, 2s, 30s, 60s, 2s, 30s, 10s (10~30 times larger than the judge solutions' runtime)
- Nobody attempted A, I
- Somebody attempted C, G but failed. Their algorithms were proved wrong.
- Three problems (B, E, J) are solved within the first 40 minutes, but at most one problem for each team

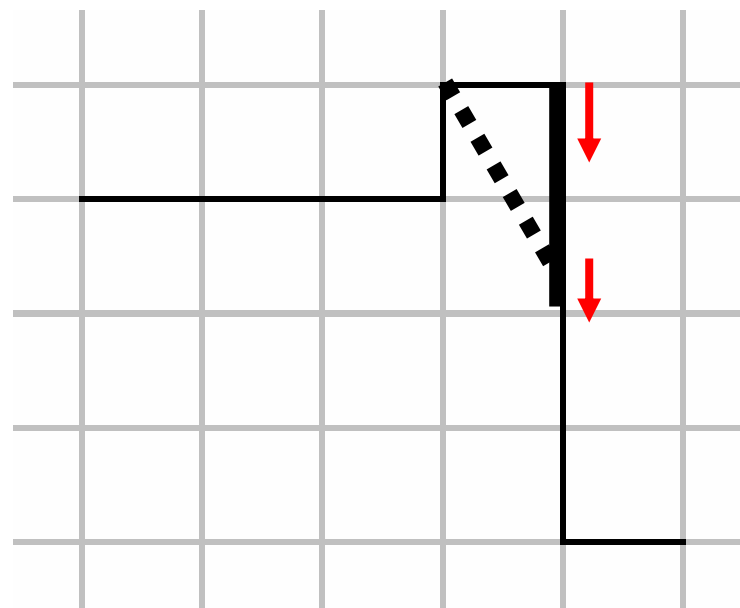
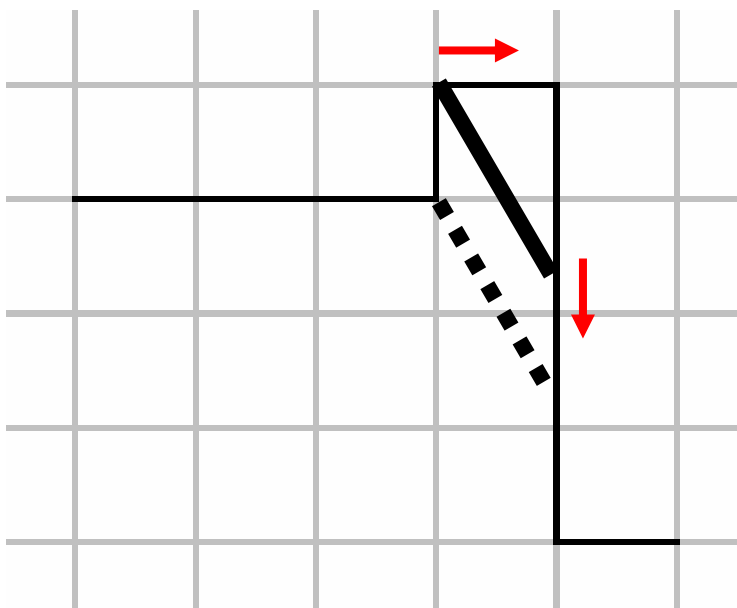
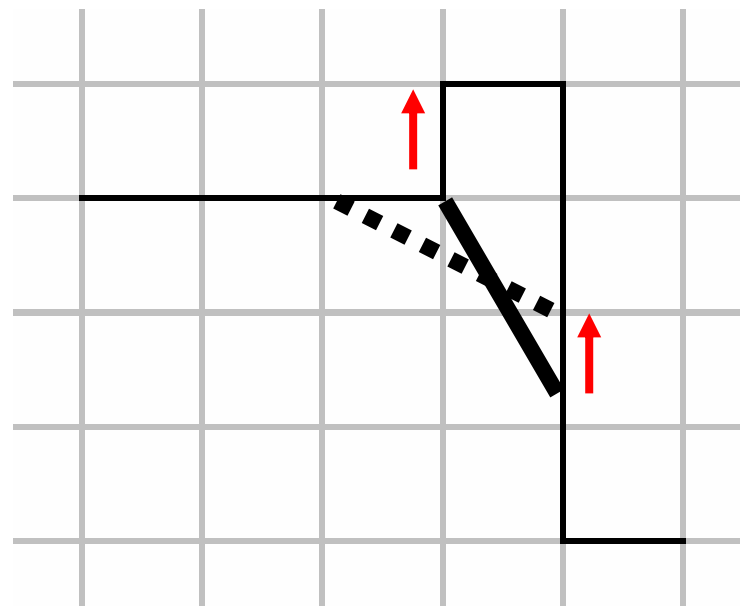
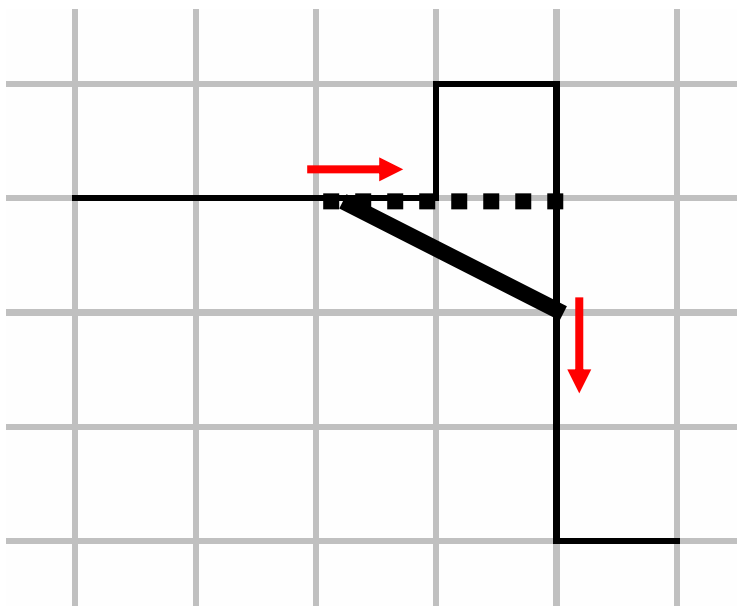
Online Competition (24 hours)

- Time limits: 3s, 5s, 3s, 5s, 1s, 3s, 5s, 1s, 5s, 1s, for UVa's slow machine (800MHz)
- Nobody solved C and F (due to tight time limit), but A, G, I are solved by 1, 1 and 4 people respectively (A, G, I are not solved on-site)
- C is the only problem that remained unsolved after the on-site and online competitions

A Rod in a Path

- 有一根无法压缩或伸展的长度为 L 为杆AB，初始时B在 $(0,0)$ ，A在 $(L,0)$ 。路径为水平与垂直线段交替的 n 段折线， n 为奇数(即最后一段为水平)。水平段自左向右，垂直线段可以向上，也可以向下。求A到达路径最右端时经过的距离





情况介绍

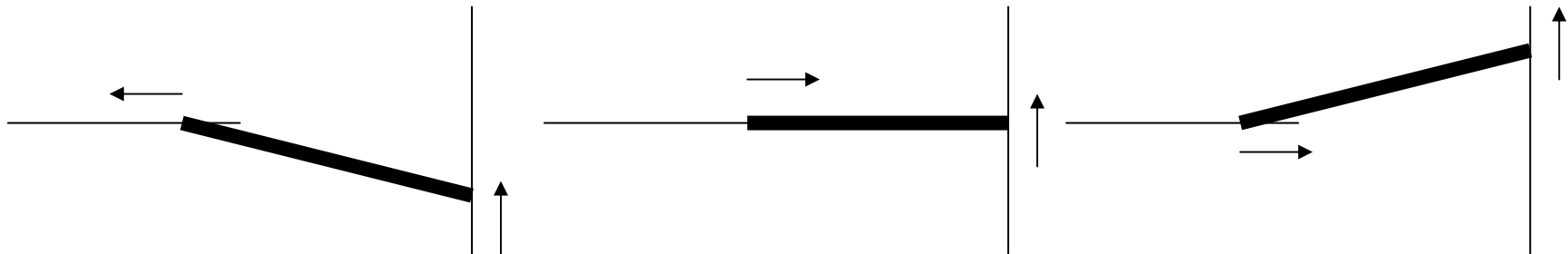
- On-site比赛无人提交
- Online比赛20份提交, 其中1个Yes

分析

- 用 P_A 与 P_B 表示棍的两个端点离起点的位置, 则 (P_A, P_B) 共同描述了棍的状态
- **引理:** 最优运动过程中, 不会存在两个时刻 t_1 和 t_2 , 使得 $(P_A(t_1), P_B(t_1)) = (P_A(t_2), P_B(t_2))$
- 注意, 此引理并不代表需要考虑的路径总是惟一的 (留给读者思考), 需要建立图论模型
- 问题: 状态是连续的, 因而是不可数无穷多的! 我们需要把状态数减少到有限多个

关键状态

- 在任何时刻, 两个端点要么往前, 要么往后, 我们把端点运动方向可能改变的状态的称为**关键状态**. 特别的, 初始与终止状态也是关键状态.
- 不难得出: 关键状态只有以下两种:
 - A或B处于路径的拐点
 - A或B处于路径某两条垂直线段的延长线交点上

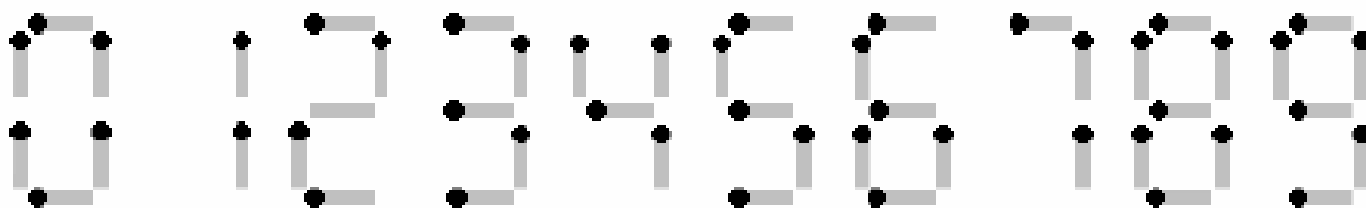


算法

- **算法1:** 把关键状态看成点, 状态之间的转移看作边, 建立有向带权图, 则问题转化为经典的**s-t**最短路问题, 可以用**Dijkstra**求解. 关键是细心设计状态转移过程.
- **算法2:** 本题的 l_i 很小, 且所有长度均为整数, 因此可以直接考虑所有整状态, 即某端点处于整点的状态, 大大降低编程复杂度

Bigger is Better

- 用不超过 n 根火柴组成 m 的非负整数倍. 这个数应该尽量大



Sample Input	Sample Output
6 3 5 6 0	Case 1: 111 Case 2: -1

$1 \leq n \leq 100, 1 \leq m \leq 3000$

情况介绍

- On-site比赛约40个Yes
- Online比赛231份提交, 11个Yes, 76个WA, 109个TLE(原因见后), 11个MLE

分析

- 很容易得到这样的动态规划算法: 设 $d_{i,j}$ 为 i 根火柴拼成除以 m 余数为 j 的最大整数, 则状态转移时可以枚举首位, 也可以枚举末位. 枚举首位的问题很容易得到错误算法! (想一想, 为什么) 比较稳妥的方法是枚举末位. 这样, 状态总数为 nm , 决策量为10, 但由于 $d_{i,j}$ 可能很大, 需要用高精度存储, 实际的运行时间比较长. **On-site**时限很长, 此法可以通过数据, 但**Online**比赛缩短时限, 此法将超时

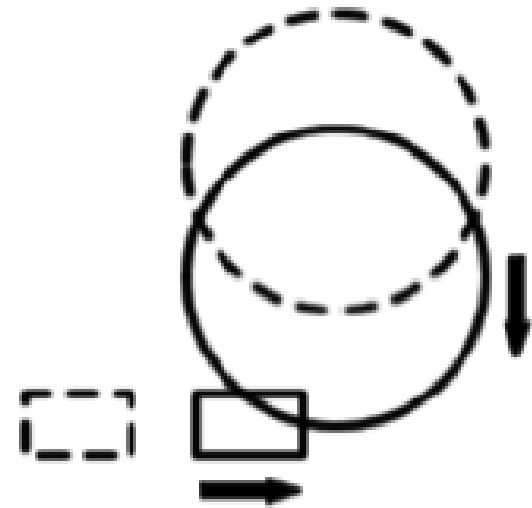
分析

- 换一个思路: 设 $d_{i,j}$ 表示用拼一个除以 m 余数为 j 的 i 位数至少需要多少根火柴, 则很容易用 $O(nm)$ 时间递推出所有 $d_{i,j}$. 找到让 $d_{i,0} \leq n$ 的最大 i , 也就求出了答案的位数. 接下来只需要根据 $d_{i,j}$ 贪心的一位一位取即可, 总时间复杂度为 $O(nm)$
- 也可计算 $d_{i,j}$: 用 i 根火柴拼一个除以 m 余数为 j 的最大位数, 但容易出错(想一想, 为什么)

Collapse!!!

- 你坐着一个 $a*b$ 的矩形飞船，左下角位于 $(0,0)$ 。你的任务是在 T 时刻前让它的左下角到达 $(L,0)$ 。你可以选择一个时刻 t_0 让飞船往右以你选择的速度 v_0 运动，途中不能与下落的圆形石头相撞，但可以相切
 - 所有石头速度均为 V
 - 石头 i 半径 r_i ，下落时间 t_i
 - 石头 i 下落时中心为 (X_i, H)
 - 石头顶端落到 $y=0$ 时消失

最小化



Sample Input	Sample Output
1 2 1 20 12 3 20 5 2 5 0 <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <div style="border: 1px solid black; padding: 5px;">$n \ a \ b \ L \ H \ V \ T$</div> <div style="border: 1px solid black; padding: 5px;">$x_i \ r_i \ t_i$</div> </div>	Case 1: 1.00

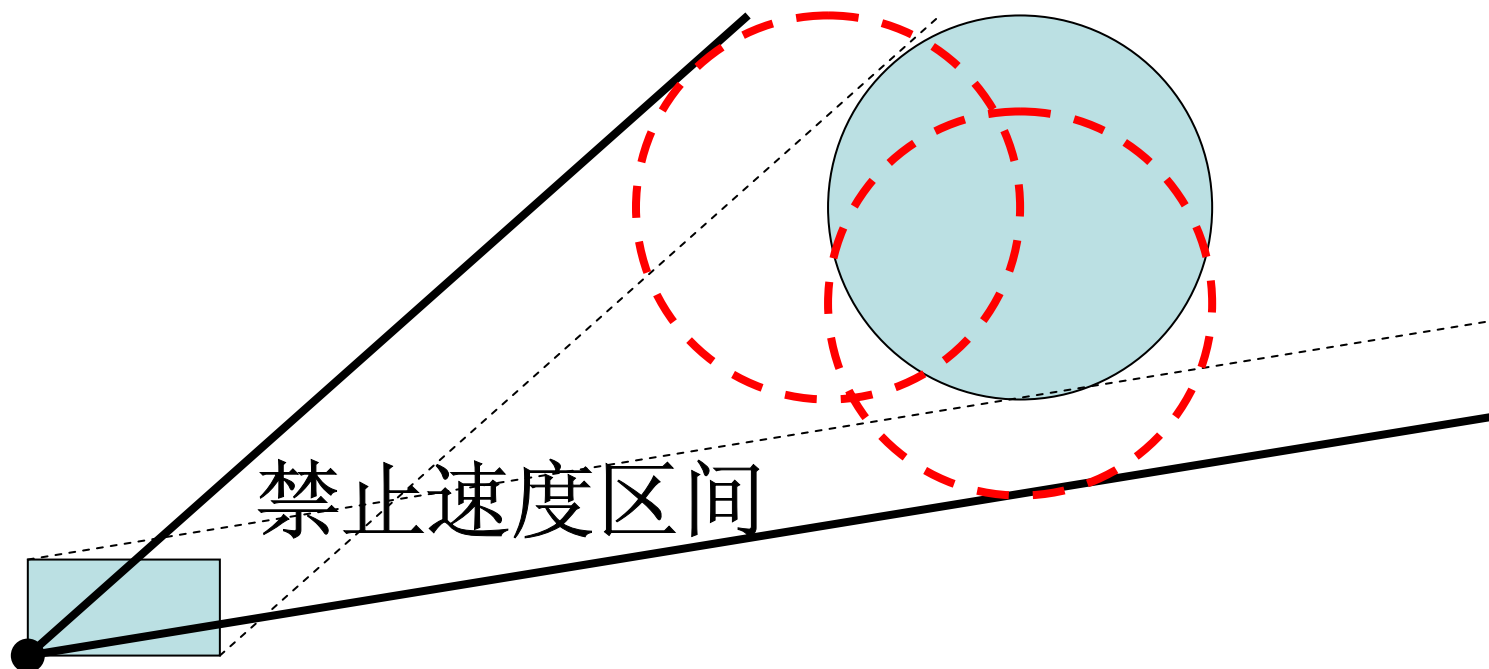
- $1 \leq n \leq 50$ 石头个数
- $1 \leq a, b \leq 10$
- $1 \leq L, H, V, T \leq 10000$ 刚下落时不会砸到飞船
- $1 \leq t_i \leq T, 1 \leq r_i \leq 200, a + r_i \leq x_i \leq L - r_i, b < H - r_i$

情况介绍

- **On-site**有约**5**支队伍提交, 均没有通过. 赛后询问后得知均是错误算法或近似算法(按较小步长枚举出发时间后计算最小速度). 由于数据比较难故而未通过
- **Onsite**有**22**份提交, 均未通过

相对运动 + 拆圆

- 给出发时间 t , 哪些速度不会与石头 i 相撞?
 - **相对运动**: 石头不动, 飞船往右上方移动
 - **拆圆**: 飞船变成一个点



极限法

- 最优直线有以下几种情况
 - 与两圆相切
 - 经过 $(0,0)$ 并与一个圆相切
 - 经过 (L, V_t) 并与一个圆相切
 - 经过 $(0,0)$ 与 (L, V_t)
- 枚举 $O(N^2)$ 条直线后判断是否合法即可. 注意不能只判断出发与到达时刻是否合法, 以及直线是否和某圆相交(想一想, 为什么)

Digit Puzzle

- 给出形如 $a*b=c$ 的数子谜, 修改尽量少的数字使得解惟一. a, b, c 分别不超过2, 2, 4位

$$7 \times \square \square = 8 \square$$

$$\square \square \times \square \square = 1 \square 1$$

Sample Input	Sample Output
7 ** 8* ** ** *** 0	Case 1: 7 ** 8* Case 2: ** ** 1*1

情况介绍

- **On-site**有约**15**支队伍提交, 约**5**个**Yes**, 其中一个使用在线搜索, 其他均为离线搜索
- **Online**有**49**份提交, **4**个**Yes**, **24**个**WA**, **19**个**TLE**. 注意**Online**比赛时限比较紧

思路一

- 离线搜索, 先保存所有解惟一的数字谜, 然后每次遍历所有元素, 选一个最接近的
- 显然, 如果枚举所有游戏一一判断是否解惟一, 还不如在线搜索. 注意到合法等式不超过 $100^2=10000$ 种, 因此可以从等式出发进行修改, 即枚举方格位置集合(最多有 $2^8=256$ 种), 让该数字谜对应的解加1, 最后删除解不惟一的, 得到**770589**个数字谜

思路二

- 虽然看上去搜索量很大, 但本题确实可以在线搜索.
- **迭代加深搜索:** 枚举花费 c , 枚举与输入游戏相距为 c 的所有游戏并逐个判断.

Extraordinarily-Tired Students

- 每个学生的睡觉方式用三元组(a, b, c)表示, 即先听课a分钟, 然后睡觉b分钟. 第一分钟时处于周期的第c分钟, 准备睡觉时如果发现至少有 half 的学生在听课, 就不再睡觉
- 求出所有学生都开始听课的时间

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
		😴	😴	😴	😴			😴	😴	😴	😴						
😴	😴	😴	😴	😴		😴	😴	😴	😴	😴		😴	😴	😴	😴	😴	
😴	😴	😴		😴	😴	😴	😴		😴	😴	😴	😴					

(2,4,1), (1,5,2), (1,4,3)

Sample Input	Sample Output
<div> <div> <div>3</div> <div>2 4 1</div> <div>1 5 2</div> <div>1 4 3</div> <div>3</div> <div>1 2 1</div> <div>1 2 2</div> <div>1 2 3</div> <div>0</div> </div> <div> <div>1 ≤ n ≤ 10</div> <div>1 ≤ a, b ≤ 5</div> <div>1 ≤ c ≤ a + b</div> </div> </div>	<div>Case 1: 18</div> <div>Case 2: -1</div>

情况介绍

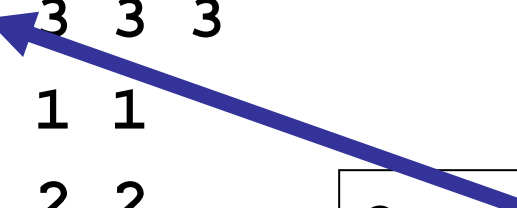
- 本次比赛最简单的题目
- **On-site**约90支队伍通过, 但第一份**Yes**在30分钟后才出现
- **Online**有172分提交, 47个**Yes**, 71个**WA**, 31个**TLE**

分析

- 每人周期 $c \leq 10$, 因此 $\text{lcm}(1, 2, \dots, 10) = 2520$ 个时间单位后一定出现循环
- **算法:** 模拟3000个周期, 如果仍有人睡觉, 则输出-1

Fairies' Defence

- 在长方体 $(0,0,0)-(a,b,c)$ 内有 n 只天使, 另有一只生物将此长方体中以均匀概率密度出现, 并攻击最近的天使. 求每只天使被攻击的概率.

Sample Input	Sample Output
 <pre> 2 3 3 3 1 1 1 2 2 2 ----- 2 7 2 10 1 1 6 3 1 6 0 </pre>	<pre> Case 1: 0.500 0.500 Case 2: 0.286 0.714 </pre>

$$2 \leq n \leq 20$$

$$0 \leq a, b, c \leq 1000$$

$$0 \leq x \leq a, 0 \leq y \leq b, 0 \leq z \leq c$$

情况介绍

- 考虑到整场比赛难度太大, **On-site**的时限故意设得很宽, 优化的**Monte-Carlo**算法可以通过本题. 但**Online**时限紧, 近似算法无法通过
- **Online**比赛11分提交, 5个WA, 6个TLE

分析

- **模型:** 计算三维Voronoi图, 计算每个点所在Voronoi多面体与长方体相交部分的体积
- 本题的规模很小, 因此不需要十分高效的三维Voronoi图算法. 事实上, 本题不需要任何三维Voronoi图的知识
- 假设只有两个天使P和天使Q, 怪兽出现在哪些位置将攻击P而不是Q呢? 作线段PQ的中垂面, 则处于半空间Q-P时攻击P

Voronoi多面体

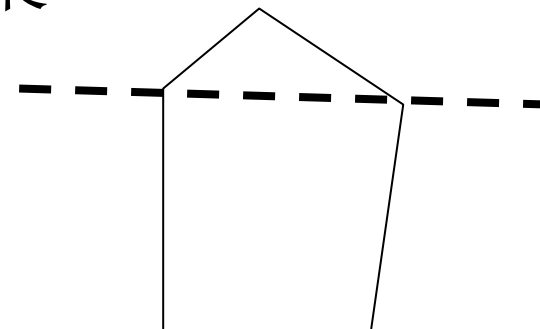
- 对于天使 P , 单独考虑所有天使 Q , 则所有半空间 $Q-P$ 的交再与 $(0,0,0)-(a,b,c)$ 求交后得到一个体积有限的凸多面体
- 连接 P 和多面体的各个顶点, 则多面体被切割成了若干个棱锥, 每个棱锥的体积等于 $V=Sh/3$, 其中 S 为底面积, h 为 P 到该底面的距离. 下面考虑如何求出Voronoi多面体

切割多面体

- **增量法:** 依次考虑各个半空间, 把当前多面体中非法的部分切掉. 初始多面体是长方体 $(0,0,0)-(a,b,c)$
- 半空间可用点法式: (P, Q) , 其中 P 为半平面上一点, Q 为单位法向量.
- 先考虑顶点 V_i , 记 $d_i = (V_i - P) \cdot Q$, 则
 - $d_i < 0$, V_i 不在 (P, Q) 半空间中
 - $d_i \geq 0$, V_i 仍然是顶点

图论模型

- 考虑原来的每条边 (V_i, V_j)
 - $d_i < 0$, $d_j \leq 0$, 或 $d_i \leq 0$, $d_j < 0$, 删除该边
 - $d_i \geq 0$, $d_j \geq 0$, 保留该边
 - $d_i d_j < 0$, 则用 (V_i, V_j) 与半空间的公共部分代替 (V_i, V_j)
- 最后缺少的一部分边: 把所有满足 $d_i = 0$ 的顶点取出, 求出二维凸包, 加入边集
- 求出图表示法后, 计算体积



Gargoyle

- 城堡顶层有 n 个怪兽状滴水嘴, 还有一个 m 个连接点和 k 个水管的水流系统. 从滴水嘴流出的水直接进入蓄水池, 通过水管后重新由滴水嘴流出. 假设水量无损失, 每个连接点处的总入水速度应等于总出水速度. 水管中水流的速度有上下界, 单位水速有固定费用

设计各水管的水速,
用尽量少的总**费用**
让各滴水嘴的出水**速度相同**



Sample Input	Sample Output
3 1 4 0 4 8 15 5 4 1 2 5 2 4 2 1 6 1 4 3 3 7 2 0	Case 1: 60.00

- $1 \leq n \leq 25$, $1 \leq m \leq 50$, $1 \leq k \leq 1000$,
- $0 \leq a, b \leq n+m$, $0 \leq l \leq u \leq 100$, $1 \leq c \leq 100$
- 无自环和重边, 水管入口不会是滴水嘴, 出口不会是蓄水池. 蓄水池编号为0, 滴水嘴编号为1~n, 连接点编号为n+1~n+m

情况介绍

- **On-site**有三份提交, 均未通过, 但答案已比较接近. 事后得知该队错误的认为答案一定可以写成分母为 n 的有理数.
- **Online**有23份提交, 1个Yes, 9个WA, 13个TLE

分析

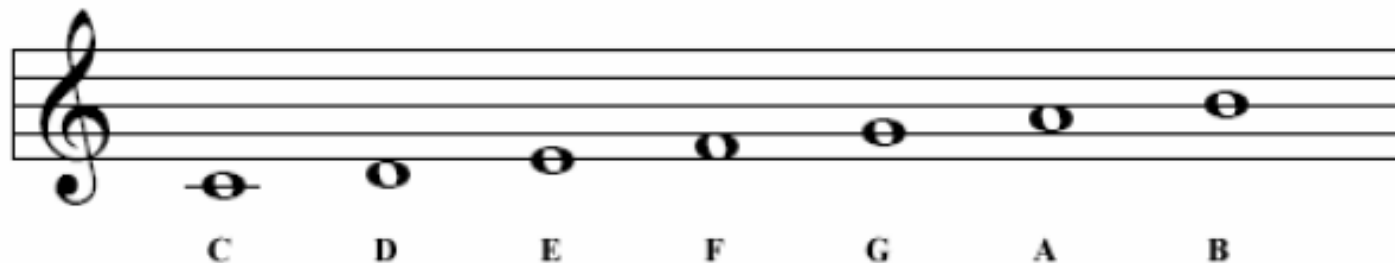
- **模型:** 求一个最小费用流使得某边集上流量相同. 假设该边集的流量均为 f , 则当 f 确定下来以后问题转化为普通的带上下界最小费用流. 假设该问题的解为 $c(f)$, 我们有:
- **引理1:** f 的可行域(让 $c(f)$ 存在的取值范围)为连续区间 $[l, r]$
- **引理2:** 在可行区域内, $c(f)$ 是 f 的下凸函数
- 证明略. 因此可以通过二分/三分法求解

另一个结论

- 本题还有一个结论, 但并不一定要用:
- **引理3:** 最优值对应的 f 一定是有理数, 且分母为不超过 n 的整数, 其中 n 为滴水嘴的个数
- 证明略. 有了这个定理, 可以进一步缩小二分/三分的范围

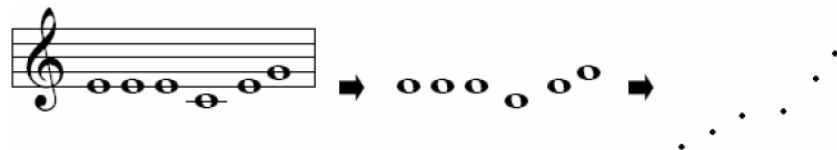
Hidden Music Score

- 五线谱的相邻两线的距离称为一个标准距离(sd), 为1.0cm到5.0cm. 相邻音符的水平距离保证不小于1sd, 不大于5sd
- 擦掉五线谱后旋转 a 度($-60 \leq a \leq 60$, a 为整数)后给出所有音符的坐标和第一个、最后一个音符的名称(保证不相同), 求原始音符. 解保证惟一, 且对浮点误差不敏感



Sample Input	Sample Output
<p>6 E G</p> <p>0.00000000 1.00000000</p> <p>1.00000000 1.00000000</p> <p>2.00000000 1.00000000</p> <p>4.00000000 0.00000000</p> <p>5.00000000 1.00000000</p> <p>8.00000000 2.00000000</p> <p>-----</p> <p>4 A C</p> <p>0.00000000 15.62499286</p> <p>11.28111236 5.80618831</p> <p>20.63744497 6.54957842</p> <p>37.94846083 0.00000000</p> <p>-----</p> <p>7 B F</p> <p>0.00000000 0.00000000</p> <p>15.14798698 18.22443643</p> <p>25.04608611 30.65582149</p> <p>19.58478851 24.56084570</p> <p>23.09216832 27.86533768</p> <p>11.29672536 9.31513384</p> <p>8.65999632 3.84492903</p> <p>0</p>	<p>Case 1: EEECEG</p> <p>Case 2: ADEC</p> <p>Case 3: BDEGGFF</p>

3<=n<=20



情况介绍

- 出题者的本意是送分，但**On-site**只有不超过**5**支队伍通过，且第一份提交是**3**小时以后才出现的. 很多队伍错误的原因是看题不仔细
- **Online**有**177**份提交，**14**个**Yes**, **87**个**WA**, **17**个**TLE**

分析

- 关键点：
 - 旋转角度为整数
 - 第一个音符和最后一个音符不同
 - $1 \leq sd \leq 5$, 相邻水平距离为 $1sd$ 到 $5sd$ 之间
- 因此只需枚举旋转角度, 排序后就可以得到第一个和最后一个音符, 并计算出 sd 并且算出其他音符的音名. 注意判断 sd 和水平距离是否符合要求.
- 累积每个字符垂直位置的误差, 输出误差最小的, 而不要想当然的设置 ϵ

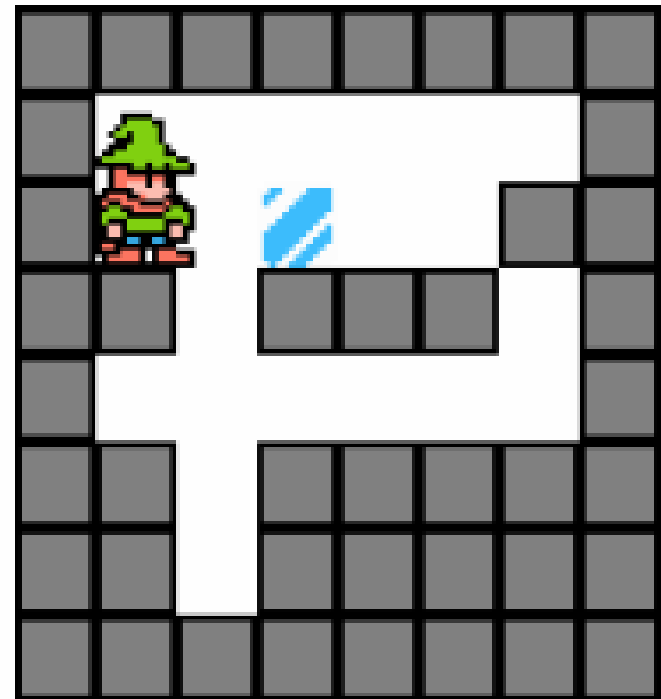
Iceman

- 在寒冰王国的一个房间里有一个冰人，你的任务是用最少的步数让他到达一个给定的目的地。这个房间是一个 $n*m$ 网格，各行从上到下编号为 $1\sim m$ ，各列从左到右编号为 $1\sim n$ 。每格要么为空，要么是冰，要么是石头。石头始终不变，而冰可以由魔法创造或者销毁。房间的第一行、最后一行、第一列和最后一列都是石头。冰人的初始位置总是一个空格，目的地也是一个空格，它的正下方总是一块石头。冰人看上去比较大，但它总是恰好占据一个空格

- **左魔法.** 作用于他的左下方
 - 如果该格子是石头，则魔法无效；
 - 如果格子是空格，则变为冰；
 - 如果是冰，则变为空格。
- **魔法把空格变成冰时**，如果它的左(右)边相邻格是冰或者石头，那么这块新创造出来的冰会和左(右)边的相邻冰或者石头**粘在一起**。“粘”的作用是相互的，同一行中的多块冰或石头可以粘成一个整体。连接在一起且最左/右端都不是石头的整体称为冰棒。冰棒中每个格子的正下方均为空时该冰棒将整体下落。但直接或间接与石头粘在一起的冰块不会自由下落的。**魔法把冰变成空格时**，它左右的连接（如果有的话）都将随之被销毁

- **左移.** 假设冰人在 (r,c)
 - $(r,c-1)$ 为空, 则冰人移动到此方格。如果此时他脚下的方格 $(r+1,c-1)$ 为空, 则冰人垂直下落, 直到脚下是冰或者石头. 下落过程中不能进行操作
 - $(r,c-1)$ 是石头, 则试着往上爬。如果 $(r-1,c-1)$ 和 $(r-1,c)$ 均为空, 则爬到 $(r-1,c-1)$, 否则留在 (r,c) ;
 - $(r,c-1)$ 是冰, 则当且仅当 $(r,c-2)$ 也为空时, 该冰被推向左方。它将一直向左运动, 直到被冰或石头挡住, 或者它正下方为空 (此时冰块按照前面叙述的规则下落, 且落地后不再继续往左移动)。
- 冰块被推走后, 它上方的冰棒将会落下。每次操作后, 冰人总是等所有冰块都不再移动后才能继续操作。如果 $(r,c-1)$ 是冰但不能被推走, 则把它当作石头一样处理 (尝试往上爬上)

- 只有水平连接, 没有垂直连接(冰块下落后不会和下方的冰块/石头粘在一起), 且只有魔法操作可以创建或销毁这些连接. 根据冰块的与左右两边格子的连接情况, 用四种不同的字符表示
 - 双端自由“O”, 双端连接“=”
 - 左端自由“[”, 右端自由“]”
 - 右魔法“>”与左魔法“<”对称
 - 右移“R”与左移“L”对称
 - 初始位置用“@”表示
 - 目标用“#”表示
 - 空格用“.”表示
 - 石头用“X”表示



Sample Input	Sample Output
<pre> 5 5 XXXXX X@.#X XX.XX X...X xxxxx ----- 7 7 xxxxxxxx X.....X X@[=].X xxx.xxx xxx.xxx xxx#xxx xxxxxxxx ----- 6 6 xxxxxxx X@...X XXXX=X X..O.X X.#O.X xxxxxxx 0 </pre>	<pre> Case 1: >RR Case 2: R>R Case 3: RR>RLLLLL>R </pre>

地图不超过 $10*10$
 一定有解且不超过15步
 步数最少的解惟一

情况介绍

- On-site无人提交
- Online有32份提交, 4个Yes, 14个WA, 4个TLE

分析

- 本题的背景是FC游戏《所罗门之匙2》，日文名为《Solomon no Kagi 2 – Coolmin Tou Kyuushutsu Sakusen》
- 实现规则时注意细心
- **On-site**时限很长，只要实现正确均可通过数据.即使使用**DFS**作为主框架, 只要使用了简单的**h**函数，也不会超时

布局的表示

- 有多种复杂的表示，不过这里采取**最简单**的方式：自上而下自左向右各个方格连接而成的字符串

```
char map[20][20];  
for(int i = 0; i < n; i++) scanf("%s", map[i]);  
string s = "";  
for(int i = 0; i < n; i++)  
    for(int j = 0; j < m; j++){  
        if(map[i][j] == '#'){ target = i*m + j; map[i][j] = '.'; }  
        s += map[i][j];  
    }
```

算法框架

- 有多种复杂的框架，这里采取BFS

```
q.push(s);
```

```
sol.clear();
```

```
sol[s] = "";
```

```
while(!q.empty()){
```

```
    string s = q.front();
```

```
    q.pop();
```

```
    if(expand(s, '<')) break; if(expand(s, '>')) break;
```

```
    if(expand(s, 'L')) break; if(expand(s, 'R')) break;
```

```
}
```

初始状态入队，操作序列为空

核心过程：扩展状态并返回是否找到解

```
bool expand(string s, char cmd){  
    string seq = sol[s] + cmd;  
    int x = s.find('@'); 获取人的位置并删除  
    s[x] = '.';          否则可能会把人当障碍  
    if(cmd == '<' || cmd == '>'){  
        ...  
    }else{  
        ...  
    }  
    s = fall(s); 物体自由下落          利用h函数剪枝  
    if(h(s) + seq.length() > 15) return false;  
    if(s.find('@') == target){ printans(seq); return true; }  
    if(!sol.count(s)){ sol[s] = seq; q.push(s); }  
    return false; 新结点放入已扩展结点集中并入队  
}
```

```
s[x] = '@'; 人的位置不变
int p = (cmd == '<' ? x+m-1 : x+m+1); 被操作位置
if(s[p] == 'X') return false;
else if(s[p] == '.'){
    s[p] = 'O'; 创建冰与相关连接
    if(icy[s[p-1]]) s[p-1] = link_r[s[p-1]];
    if(s[p-1] != '.') s[p] = link_l[s[p]];
    if(icy[s[p+1]]) s[p+1] = link_l[s[p+1]];
    if(s[p+1] != '.') s[p] = link_r[s[p]];
}else{
    s[p] = '.'; 消除冰与相关连接
    if(icy[s[p-1]]) s[p-1] = clear_r[s[p-1]];
    if(icy[s[p+1]]) s[p+1] = clear_l[s[p+1]];
}
```

魔法的处理

```
int k, p = (cmd == 'L' ? x-1 : x+1);
if(s[p] == '.') s[p] = '@'; 成功移动
else{
    if(s[p] == 'O' && cmd == 'L' && s[p-1] == '.'){ 左推冰
        for(k = p-1; k > 0; k--) if(s[k-1] != '.' || s[k+m] == '.') break;
        s[p] = '.'; s[k] = 'O'; s[x] = '@';
    }
    else if(s[p] == 'O' && cmd == 'R' && s[p+1] == '.'){ 右推冰
        for(k = p+1; k < n*m; k++) if(s[k+1] != '.' || s[k+m] == '.') break;
        s[p] = '.'; s[k] = 'O'; s[x] = '@';
    }
    else{ 往上爬
        if(s[p-m] == '.' && s[x-m] == '.') s[p-m] = '@'; else s[x] = '@';
    }
}
```

移动的处理

自由下落过程

- 从下到上处理每一行，每行自左向右扫描，处理人和冰
- 人和O：单独处理
- 遇到[则接下来的几个方格为====]或===X
 - 以]终止：判断是否下落
 - 以X终止：不用判断，一定不会下落

思考：

如何减少不必要的判断？

如果需要，可以附加信息或者修改状态表示

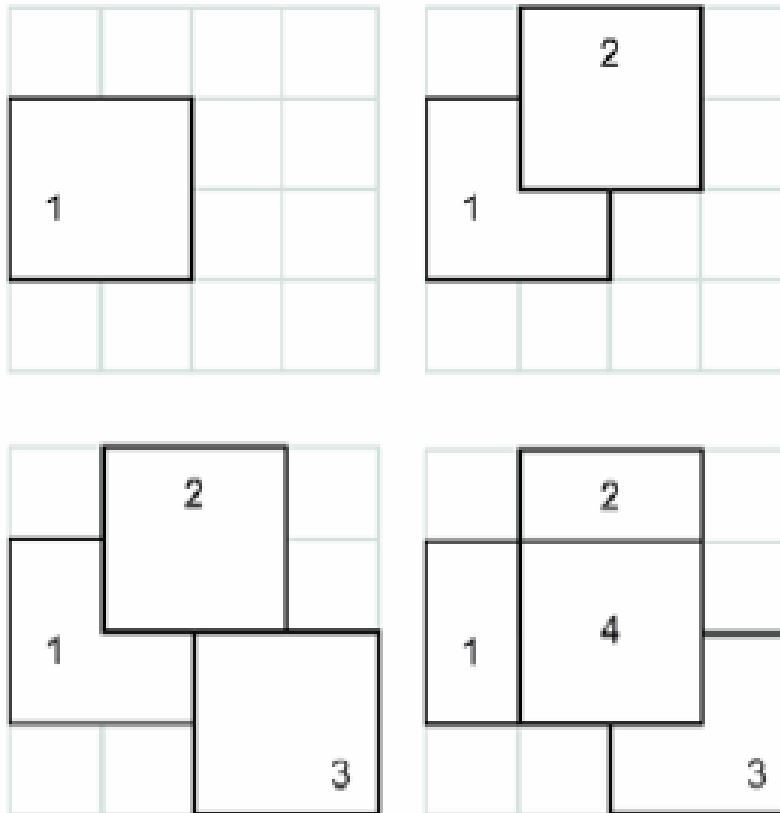
启发函数h

- $h(s)$ 表示从局面 s 出发到达目标状态至少还需要多少步.可以用最短路算法代替简单的 h 函数,但是不能考虑冰块

```
int h(string s){  
    int a, b, x = s.find('@');  
    a = x%m - target%m; if(a < 0) a = -a; 水平移动  
    if(x/m > target/m) b = x/m - target/m; 竖直移动  
    else b = (x/m < target/m ? 1 : 0);  
    return a > b ? a : b;  
}
```

Overlapping Squares

- 把1到6个 2×2 正方形放到 4×4 的格子中. 给出图案, 问是否可能



Sample Input

```
          #
    _ _ _  #
| | _ _ |  #
| _ |   |  #
    | _ _ |  #
```

#

```
      _ _  #
    |   |  #
    | _ _ |  #
```

#

```
    _ _ _ _  #
| _ | _ | _ |  #
| _ | _ | _ |  #
| _ | _ | _ |  #
| _ | _ | _ |  #
```

#

```
      _ _  #
    _ |   | _  #
| | _ _ | |  #
| _ |   | _ |  #
    | _ _ | _ |  #
```

0

Sample Output

Case 1: Yes

Case 2: Yes

Case 3: No

Case 4: Yes

情况介绍

- On-site约有40支队伍通过
- Online有159份提交， 34个Yes, 60个WA, 55个TLE

分析

- **思路一：** 预处理算出所有图案，然后每次询问时进行简单查找
- **思路二：** 在线搜索，注意从“上往下”搜，即先搜最顶上的（它必须完全露在外面而且中间不能有其他线条），然后搜压在底下的。这样利于及时排除不可能的情况