IOI2014 集训队
Homework I Part II
ACM/ICPC World Finals
题目泛做

俞鼎力
绍兴市第一中学

2014 年 1 月 2 日

Contents

Chapter 1

ACM/ICPC World Finals 2013

Abstract

ACM/ICPC World Finals 2013 共 11 题。其中简单题 4 道，分别是：A、D、F、J；中等题 4 道：C、E、H、I；难题 3 道：B、G、K。

简单题中涉及了图的建模、简单数论、简单搜索、贪心、二分、基础几何等知识点。中等题中有网络流及其建模、搜索及剪枝、动态规划及其优化、堆栈、统计优化等考点。难题主要考到了复杂的数学知识、复杂的几何以及复杂的记忆化搜索。

总体来说，难度安排合理，知识点覆盖全面，给我们带来了很多启发。

以下是本人做题之后的一些理解与思路。

1.1 Problem A Self-Assembly

1.1.1 题目大意

给出 $n(n \leq 40000)$ 种正方形，正方形的四条边上分别有 2 个字符（一个大写字母加‘+’或‘–’），两个正方形能拼接起来的条件是：相接的两条边的字母相同，符号相反。

正方形可以旋转、翻转。问是否能拼出一个无限大的图形。

1.1.2 简化

首先，如果存在无限大的图形，那么，它一定可以是某一个图形头尾相接，循环无限次构成的。

其次，一定存在一个无限图形不包含 2×2 的矩形，因为我们的目标是将其头尾相接，只需要保留头到尾的路径即可。

再次，不必担心前后两个相同的图形接起来之后重合的问题，因为可以翻转，所以我们可以保证这条路径始终是从左上到右下的。

1.1.3 算法

有了上述简化，算法也比较显然了。

我们将每种边的标识 x 看成点，它能通过与它对应的边 x' 走到另一个正方形内，然后走到该正方形除进入的边以外的其他边内。

也就是说，如果一个正方形中有两条边的标识为 x, y，那么就连一条 x' 到 y 的有向边，和一条 y' 到 x 的有向边。

最后我们只需判断这个图是否有环即可。如果有环，则说明能拼出无限大的图形。

判断是否有环可以用代码较短的 Floyd，也可以用复杂度较低的拓扑序来判断。

1.1.4 时空复杂度

时间复杂度：$O(n + C^2)$，C 表示字符集的大小。

空间复杂度：$O(C^2)$。

1.2 Problem B Hey, Better Bettor

1.2.1 题目大意

一个变量，初始为 0，可以进行多次操作一：这个变量有 $p(p \leq 0.4999)$ 的概率 +1、$1-p$ 的概率 –1；只能进行一次操作二：把这个变量乘上 $\lambda(0 \leq \lambda \leq 99.99)$。可以随时停止操作。要使这个变量的期望值尽可能大，问最大是多少。

1.2.2 算法

第一步，可以发现肯定是最后进行操作二，也就是说可以直接看作只有操作一，最后结束之前如果是负数就进行操作二。

第二步，下一步的决策只和当前变量的值有关，与之前的一系列操作无关，因此，肯定存在一个下限 $-a(a \geq 0)$ 和一个上限 $b(b \geq 0)$，当变量的值等于 $-a$ 或 b 时停止操作，当然如果 $a = 0$ 或 $b = 0$，那么期望为0，所以以下我们默认 $a, b > 0$。

第三步，设 $f(x)$ 表示当前变量的值为 x 时，其先到达 b 的概率，显然有以下式子：

$$f(-a) = 0 \tag{1.1}$$

$$f(b) = 1 \tag{1.2}$$

同时有

$$f(n) = pf(n+1) + (1-p)f(n-1) \qquad (-a < n < b)$$

即

$$f(n) = \frac{1}{p}f(n-1) - \frac{1-p}{p}f(n-2) \qquad (-a < n < b) \tag{1.3}$$

这个二阶递推数列可用特殊根法求得通项，其特征方程为 $x^2 = \frac{1}{p}x - \frac{1-p}{p}$，解方程得 $x_1 = 1, x_2 = \frac{1-p}{p}$，所以

$$f(n) = c_1 + c_2\left(\frac{1-p}{p}\right)^n \qquad c_1, c_2 \text{ 是待定常数} \tag{1.4}$$

令 $\alpha = \frac{1-p}{p}$，并将(1.1)和(1.2)代入(1.4)，可以得到 $c_1 = \frac{1}{\alpha^b - \alpha^{-a}}, c_2 = \frac{\alpha^{-a}}{\alpha^{-a} - \alpha^b}$。

因此设

$$\begin{aligned} g(a,b) &= b \cdot f(0) - a\lambda(1 - f(0)) \\ &= (a\lambda + b)f(0) - a\lambda \\ &= (a\lambda + b)(c_1 + c_2) - a\lambda \\ &= \frac{(a\lambda + b)(\alpha^{-a} - 1)}{\alpha^{-a} - \alpha^b} - a\lambda \end{aligned}$$

答案即为 $g(a,b)$ 的最小值。

打表可发现 $g(a,b)$ 只有一个极值，因此可以两次三分来解决这个问题。

可以证明：极值点 (a_0, b_0) 满足 $a_0 + b_0 \leq \log_\alpha 10^{10} = 10/\log_{10}\alpha$，因此三分时上界定为 $10/\log_{10}\alpha$ 即可。

1.2.3 时空复杂度

时间复杂度：$O(\log^2 C)(C = \log_{10}\alpha)$。

空间复杂度：$O(C)$。

1.3 Problem C Surely You Congest

1.3.1 题目大意

给出 $n(n \leq 25000)$ 个点 $m(m \leq 50000)$ 条边的无向图，有 $c(c \leq 1000)$ 辆车想要从某些点走最短路到达 1 号点，但是不能有两辆车并排行驶。问最多能满足多少辆车的要求。

1.3.2 算法

首先，显而易见，如果两辆车到达 1 号点的距离不同，那么他们一定不可能并排行驶。所以我们可以对每种距离的车分开计算。

现在问题变成某些点上有车，它们要到 1 号点，每条路（如果在最短路径 DAG[1] 上）有 1 的流量。那么，我们运用简单的网络流知识就能知道这是一个最大流问题。

1.3.3 时空复杂度

时间复杂度：$O(c \cdot m)$。

[1] 也叫最短路径树

空间复杂度：$O(m+c)$。

1.4 Problem D Factors

1.4.1 题目大意

$f(k)$ 表示 k 的质因子排列方案数，如 $f(10)=2(10=2\times5=5\times2)$，$f(20)=3(20=5\times2\times2=2\times5\times2=2\times2\times5)$。

给一个正整数 $n(n<2^{63})$，求最小的 k 使得 $f(k)=n$。数据保证 $k<2^{63}$。

1.4.2 算法

假设我们已知 $k=\prod_{i=1}^{m}p_i^{q_i}$，那么一定有 $p_1=2,p_2=3,p_3=5,\cdots$ 且 $q_i\geq q_{i+1}$。

而这样的 k 在 2^{63} 内并不多。所以我们只要将所有这样的 k 搜出来就行了。计算时：$f(k)=\frac{(\sum_{i=1}^{m}q_i)!}{\prod_{i=1}^{m}q_i!}$。

1.4.3 时空复杂度

时间复杂度：$O(2^{63})$。

空间复杂度：$O(2^{63})$。

1.5 Problem E Harvard

1.5.1 题目大意

$b(b\leq13)$ 个内存库，每个内存库有 $s(\leq13)$ 变量，有三种指令：

1. 访问 0 号内存库中的某个变量。
2. 访问 BSR 选择的内存库中的某个变量。
3. 设定 BSR 值。

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 `Vi`，`i` 是一个正整数($\texttt{i}\leq\min\{b\cdot s,13\}$)。
- 一个循环操作，写作 `Rn` ⟨`program`⟩ `E`，`n` 是一个正整数($\texttt{n}\leq10^6$)，⟨`program`⟩ 是一个任意的程序。这个操作等价于依次执行 `n` 遍 ⟨`program`⟩。

给出要执行的程序（最多有 1000 个元素），你需要给每个变量分配到各个内存库中，使得要执行的指令数最少，输出最少指令数。答案在 `long long` 范围内。

1.5.2 算法

搜索。

首先，变量的分配方案不超过第 13 个贝尔数($B_{13} = 27644437$)，但是我们仍然需要一个快速的方法来得到答案。

所以，我们先枚举 0 号内存库中存放的变量，然后计算给出的程序中除去 0 号内存库后 i 和 j 相邻的次数，也就是说，如果 i 与 j 不在同一个内存库中，会增加的指令数。这样我们就可以边搜索边计算需要的指令数了。

除此之后，还有一些剪枝：

- 最优性剪枝。
- 可行性剪枝，判断是否有足够的空位。
- 0 号内存库一定放满。
- 不会有两个内存库中的元素加起来小于 s。

1.5.3 时空复杂度

时间复杂度：$O(2^m \cdot l + B_m)$(m 为涉及的变量总数)。

空间复杂度：$O(l + m^2)$(l 为程序的长度)。

1.6 Problem F Low Power

1.6.1 题目大意

有 n 个机器，每个机器有两个芯片，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。

1.6.2 算法

二分，贪心。

首先，如果我们将 $2nk$ 个电池按能量从小到大排列，那么同一机器中两个芯片的最小能量的电池一定是相邻的，如果不相邻，那么将靠后的向前移一定不会变差。

然后，由于要求最大值最小，所以二分答案，现在只需要判断是否有一个方案满足所有能量差都不超过 mid。

由于同一机器中的代表电池一定相邻，所以我们可以将机器按其代表电池从 1 到 n 标号。那么，只需要满足第 i 个机器的代表电池的编号不超过 $2ki+2$。

这样，我们就可以贪心地尽可能取编号小的电池作为代表电池。也就顺利解决了这个问题。

1.6.3 时空复杂度

时间复杂度：$O(2nk(\log n + \log k))$。

空间复杂度：$O(2nk)$。

1.7 Problem G Map Tiles

1.7.1 题目大意

给出一个 $n(3 \le n \le 50)$ 个点的多边形，只可以平移，问最少需要多少 $x_s \times y_s$ 的矩形砖才能放下这个多边形。$1 \le x_s, y_s \le 100, 0 \le x_i \le 10x_s, 0 \le y_i \le 10y_s$

1.7.2 算法

首先，如果我们将所有 x_i 都除以 x_s，y_i 除以 y_s，那么就变成多边形最少要占据平面上多少 1×1 的网格的问题了。

然后，此类几何题大多是卡边界，然后暴力解决的。对于这题，我们可以找出某一个格点，然后计算答案。关于找出格点，有以下三种情况：

1. 两个多边形的点分别卡在网格的一条竖直边界和一条水平边界上；
2. 一个多边形的点卡在网格的边界上，一条多边形的边卡在格点上；
3. 两条多边形的边卡在格点上。

第一种情况容易解决，假设这两个多边形的点分别是 A, B，那么将 $(A.x, B.y)$作为格点来统计更新答案即可。

第二种情况分两种，一种情况是多边形的点 A 卡在竖直边界上，那么我们可以枚举这条竖直边界与格点的水平距离 Δx，那么格点的 x 坐标就是 $A.x + \Delta x$，另一种情况是多边形的点 A 卡在水平边界上，也是类似的方法。

第三种情况中，我们需要枚举两个被卡住的格点的相对坐标 $(\Delta x, \Delta y)$，如果我们将第二条边平移 $(-\Delta x, -\Delta y)$，那么它与第一条边的交点就是一个格点。注意数据范围保证 $|\Delta x|, |\Delta y| \leq 10$。

最后就是计算，在已知网格的情况下，这个多边形需要占据的格子数。如果我们枚举每个格子，那就变成判断一个格子是否被占据的问题。我们可以这样判断：如果有边穿过，那么一定被占据，否则，用射线法判断是否在多边形内部。

1.7.3 时空复杂度

时间复杂度：$O(n^2XY \cdot nXY)$。X, Y 分别表示 x, y 坐标的范围，根据题意有 $X, Y \leq 10$。

空间复杂度：$O(n)$。

1.8 Problem H Матрёшка

1.8.1 题目大意

有 $n(n \leq 500)$ 个大小已知的还没有嵌套的套娃排成一排，你可以将相邻的套娃合并，花费的代价为需要打开的套娃的个数。

问把这一排套娃变成若干个嵌套完全的套娃需要的最少代价。嵌套完全是指一个大小为 s 套娃，其内部的所有套娃的大小分别为 1 到 $s-1$。

1.8.2 算法

动态规划。

用 $f(l, r)$ 表示将 $[l, r]$ 这个区间内的套娃合并成一个所需要的代价。

通过将区间 $[l, r]$ 分成两个部分 $[l, m]$ 和 $[m+1, r]$ 可以进行转移，主要的问题是计算最后一次合并的代价。

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

假设我们已经将 $[l,r]$ 内的套娃从小到大排序，那么我们假设前 i 个套娃都在左边(右边)，找出这前 i 个套娃中编号最大(小)的，设为 $major(minor)$，那么对于所有 $m \geq major(m < minor)$ 都可以满足。于是我们预处理一系列前缀(后缀)信息即可。

最后我们在用一个简单的 $O(n^2)$ 动态规划就可以解决这个问题了。

1.8.3 时空复杂度

时间复杂度：$O(n^3)$。

空间复杂度：$O(n^2)$。

1.9 Problem I Pirate Chest

1.9.1 题目大意

给出一个 $n \times m(n,m \leq 500)$ 的池塘，每个格子的深度已知，问最大能放入多大的宝箱，使得宝箱顶面严格低于水面。要求宝箱的长宽高均为整数，且底面一边尺寸不能超过 a，另一边的尺寸不能超过 b。请注意宝箱排开的水。

1.9.2 算法

首先，$O(n^4)$ 暴力做法显然。

然后，我们需要优化成 $O(n^3)$。如果我们先枚举左上角，似乎陷入了僵局。于是我们尝试枚举上下边界 u,d，设 $b(i) = \min_{u \leq j \leq d} a(j,i)$。现在假设我们的左右边界是 l,r，而 i 是 $l \sim r$ 中 b 值最小的一个，那么我们可以很快得到结论 $b(l-1) < b(i)$ 且 $b(r+1) < b(i)$。这样，我们只要对于每个 i，分别找到其左边和右边第一个 b 值比它小的就能解决这个问题了。而找左边(右边)第一个小于它的，可以用栈来维护。

但是此题略卡常数，需要优化。注意到，其实不必每次更新答案，可以用一个数组 $h(i,j)$ 记录 $i \times j$ 的矩形最大深度，最后 $O(n^2)$ 统计答案。速度快一倍。

1.9.3 时空复杂度

时间复杂度：$O(n^3)$。

空间复杂度：$O(n^2)$。

1.10 Problem J Pollution Solution

1.10.1 题目大意

给出一个以原点为圆心的半径为 r 的圆和一个在 x 轴上方的 $n(n \leq 100)$ 边形(不一定凸)。求它们的交的面积。

1.10.2 算法

按两者交点和多边形的顶点的弧度划分成若干块，每块内重叠部分一定是一个三角形或者扇形。

将面积赝标量化，即如果是从小角度走到大角度则面积为正，否则为负。

1.10.3 时空复杂度

时间复杂度：$O(n)$。

空间复杂度：$O(n)$。

1.11 Problem K Up a Tree

1.11.1 题目大意

前序遍历、中序遍历、后序遍历的过程中的 6 个递归调用语句被随机地交换了。现在给出其生成的前序、中序、后序，要求复原每种情况下的原树。如果有多种情况，输出前序+中序字典序最小的。树的节点数 n 满足 $4 \leq n \leq 26$。

1.11.2 算法

记忆化搜索。

枚举左子树的大小，判断是否可行，然后递归搜索。

直接搜索是不能过的，于是加上记忆化。由于如果已知中序遍历和前序(后序)遍历，只可能有一种情况，所以最坏情况是已知前序和后序，这样的话状态数是 $O(n^4)$ 的，能顺利通过。

可以用 *Hash* 记录状态。

1.11.3 时空复杂度

时间复杂度：$O(n^5)$。

空间复杂度：$O(n^4)$。

Chapter 2

ACM/ICPC World Finals 2012

2.1 Problem A Asteroid Rangers

2.1.1 题目大意

给出三维空间内 $n(n \leq 50)$ 个点 0 时刻的坐标和其速度向量。问其构成的最小生成树将会变换的次数。

2.1.2 算法

首先，从最小生成树算法中可以看出，只有当两条边的大小关系发生改变时才有可能导致整棵树发生改变，因此可以将某一条边的长度超过另一条边的时刻作为事件点，那么事件点个数为 $O(n^4)$。

但是，如果对于每个事件点都重新计算一遍最小生成树的话，复杂度无法承受。于是考虑优化判断的过程，如果边 e 在 t 时刻之前长度大于边 f，在 t 之后小于 f，且 e 不在最小生成树中、f 在最小生成树中，且删掉 f 添上 e 后仍然是一棵树的话，就会发生一次改变。

因此问题的关键在于如何快速地判断 f 能否被 e 替换。通过观察，可以发现，f 能被 e 替换的充要条件是：f 的两个端点都是 e 其中一个端点的祖先，且 f 的两个端点不都是 e 另外一个端点的祖先。

对于询问祖先，我们可以用 DFS 序在 $O(n) - O(1)$ 复杂度内解决。而由于很难构造出 $O(n^3)$ 的答案，所以预处理的次数也很难达到 $O(n^3)$。不严谨地，我们可以说整体复杂度是 $O(n^4 \log n)$。

2.1.3 时空复杂度

时间复杂度：$O(n^4 \log n)$。
空间复杂度：$O(n^4)$。

2.2 Problem B Curvy Little Bottles

2.2.1 题目大意

要求在一个瓶子上做一些标记来指示各个体积，相邻标记间体积为 inc，假设瓶子是由一条从 $x = Xlow$ 到 $x = Xhigh$ 的 n 次多项式曲线绕 x 轴旋转一周构成。输出整个瓶子的体积、从瓶底开始的连续标记距离瓶底的距离。标记最多 8 个。

2.2.2 算法

设该多项式为 $f(x)$，则：

$$V = \int_{Xlow}^{Xhigh} \pi f^2(x)$$

求得不定积分 $F(x) = \int f^2(x)$。则 $V = \pi(F(Xhigh) - F(Xlow))$；第一个标记 x 满足 $\pi(F(x) - F(Xlow)) = inc$，用二分就可以解出 x。第二个标记同理。

而牛顿迭代法由于精度的原因，无法得到满分。

2.2.3 时空复杂度

时间复杂度：$O(n^2 \log P)$。P 表示精度要求。

空间复杂度：$O(n^2)$。

2.3 Problem C Bus Tour

2.3.1 题目大意

给出 $n(n \leq 20)$ 个点的一张无向图，一辆车需要从 0 号点出发，到 $1 \sim n-2$ 号点接客人到 $n-1$ 号点，然后再把客人送回 $1 \sim n-2$ 号点，最后回到 0 号点。其中前 $\lfloor \frac{n-2}{2} \rfloor$ 名上车的客人，要求也是前 $\lfloor \frac{n-2}{2} \rfloor$ 名下车。要求满足要求的最短路程。

2.3.2 算法

动态规划。

由于 n 很小，我们可以直接 $\binom{n-2}{\lfloor \frac{n-2}{2} \rfloor}$ 枚举先上车的客人集合 S，再枚举第 $\lfloor \frac{n-2}{2} \rfloor$ 上车的客人 x。假设我们已经求得了从 0 出发、途径 $S(|S| \leq 10)$、最后到达 x 的最短路 $f(S,x)$，以及从 1 出发、途径 $S(|S| \leq 10)$、最后到达 x 的最短路 $g(S,x)$。那么要求的最短路程就是

$$\min_{S \in U, |S| = \lfloor \frac{n-2}{2} \rfloor} \left\{ \begin{array}{rl} & \min_{x \in S} \{f(S,x) + g((U \setminus S) \cup \{x\}, x)\} \\ + & \min_{x \in S} \{g(S,x) + f((U \setminus S) \cup \{x\}, x)\} \end{array} \right\}$$

f, g 都可以通过动态规划预处理求得，当然，在此之前需要进行一次 Floyed。

2.3.3 时空复杂度

时间复杂度：$O(2^n \cdot n^2)$。
空间复杂度：$O(2^n \cdot n)$。

2.4 Problem D Fibonacci Words

2.4.1 题目大意

设 $F(0) =$ "0", $F(1) =$ "1"，$F(n) = F(n-1) + F(n-2) \quad (n \geq 2)$，给出一个长度为 $m(m \leq 100000)$ 的 01 串 p，问 p 在 $F(n)$ 中出现的次数($n \leq 100$)。

2.4.2 算法

用 $f(n)$ 表示 p 在 $F(n)$ 中出现的次数，$g(i,j)$ 表示 p 在 $F(i) + F(j)$ 中出现且穿越分割线的次数，则 $f(n) = f(n-1) + f(n-2) + g(n-1, n-2)$。

显然的，如果 $fib(n-2) \geq m$，则 $g(n,i) = g(n-2,i)$；同理如果 $fib(n-1) \geq m$，则 $g(i,n) = g(i,n-1)$。如果 i,j 足够小，就可以使用 Hash 或者其他算法在 $\min\{fib(i), fib(j)\}$ 的复杂度内得到 $g(i,j)$。

这样，我们的复杂度是 $O(nm)$ 的。但是，我们会发现，如果使用记忆化搜索得到 g，那么复杂度能降为 $O(n+m)$。

2.4.3 时空复杂度

时间复杂度：$O(n+m)$。
空间复杂度：$O(n+m)$。

2.5 Problem E Infiltration

2.5.1 题目大意

给出一张 $n(n \leq 75)$ 个点的有向图，保证若 i 到 j 有边，则 j 到 i 没有边，若 i 到 j 没有边，则 j 到 i 有边。如果选择了 i 号点，那么就能覆盖 i 号及其连出去的点。要求选择最少的点，覆盖所有点，输出任意一种方案。

2.5.2 算法

搜索，构造较优解。

直接搜索是不能过的。我们可以加些优化，比方说最优性剪枝、用 bitset 来记录点的覆盖情况。这样理论上能快 30 倍，但是还是无法通过的。我们需要更强力的优化。

注意到最优性剪枝，如果我们可以一开始就构造一个比较优可行解，那么就可以剪去许多。注意到整张图的边非常多，也就是说，如果每次取能覆盖最多节点的点，那么我可以将点数变成原来的一半，这样构造出来的解最大是 6，而对于不超过 5 的搜索是可以轻松地搜出来的，这样就完美解决了此题。

2.5.3 时空复杂度

时间复杂度：$O\left(n^5\right)$。
空间复杂度：$O(n^2)$。

2.6 Problem G Minimum Cost Flow

2.6.1 题目大意

给出 n 个三维节点，m 条管道连接两个节点使得这两个节点之间水可以通过。一些节点上有若干洞，如果水到达这个节点，那么你必须花费每个 0.5 的代价把一个洞堵住。你也可以用新添一个管道堵住两个洞并连接这两个洞所在的节点，代价为它们之间的欧几里得距离。你还可以控制水位的高低使得水维持在某一高度。只能在点 1 注水，请你用最少的代价使得水到达点 n。

2.6.2 算法

首先，由于输入都是整数，所以两点之间最小距离至少为 1，因此除去必要的管道，我们不会添加其他管道以减少代价。

然后，我们可以枚举最后的高度 h，总共有 n 种可能。我们只需要保留高度不大于 h 的点，然后做一次最短路即可。

做最短路时，我们可以缩点，然后做；也可以不缩点。无论怎么做，都需要注意一点：虽然不会在一个节点新建两个管道，但是忽视这一点可能会多算被管道堵住的洞的数量。

2.6.3 时空复杂度

时间复杂度：$O(n^3)$。
空间复杂度：$O(n^2)$。

2.7 Problem I A Safe Bet

2.7.1 题目大意

给出一个 $R \times C$ 的网格($R, C \leq 1000000$)，在某些格子上有 ／ (n 个)或 ＼ (m 个)的镜子($n, m \leq 200000$)。问在哪些格子上放一面镜子能满足：激光从矩形网格的最上方一行的左侧水平射入，从矩形网格底部一行的右侧水平射出。如果不用放，输出 0，如果不可能，输出 impossible，否则输出满足条件的格子数以及字典序最小的格子。

2.7.2 算法

根据光具有可逆性，我们可以知道，满足条件的格子一定是光从起点射入的路线和从终点射入的路线的交点。

因为光的性质，我们不必担心重合或者说多次在同一点相交的问题，所以我们只需要把路线分成横的和竖的，第一条路线的横的与第二条的竖的、第二条路线的横的与第一条的竖的分别做一次交即可。

然后，就可以用扫描线+树状数组来求得交点个数了，注意答案会超过 int。

2.7.3 时空复杂度

时间复杂度：$O(n \log n)$。

空间复杂度：$O(n)$。

2.8 Problem K Stacking Plates

2.8.1 题目大意

一个盘堆是将一些盘子从上到下从小到大堆成的，现在给你 $n(n \leq 50)$ 个盘堆，第 i 个盘堆有 $h_i(h_i \leq 50)$ 个盘子。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；也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。问最少需要多少次操作才能把所有盘堆合并成一个。

2.8.2 算法

动态规划。

首先，如果知道最后的盘堆中盘子一开始所在的盘堆，那么我们就可以知道操作次数了。事实上，只需要知道，相邻的盘子有多少是从同一个盘子中来的，亦即分割的次数。

这样我们就可以用动态规划来完成：$f(i,j)$ 表示从小到大做到大小为 i 的盘堆，这类盘子的最上面一个盘子必须是 j 的最小分割数。转移显然。

2.8.3 时空复杂度

时间复杂度：$O(n\sum_i h_i)$。
空间复杂度：$O(n\sum_i h_i)$。

2.9 Problem L Takeover Wars

2.9.1 题目大意

有 X、Y 两个公司，它们分别有 n,m 个子公司，轮流操作，X 先来，每次可以合并自己的两个子公司得到一个实力为它们之和的一个子公司，或者，干掉对方一个子公司，但是要求其实力比自己其中一个子公司弱。问谁能赢。

2.9.2 算法

我们需要明确以下三点：

- 如果选择干掉对方的子公司，那么就一定要干其最大的子公司。如果不能干掉其最大的子公司，那一定不干，否则，下一回合对方就会干掉你的最大的子公司，而你只能合并而陷入被动。可以证明，无论之后情况如何，一定不如不干掉对方的非最大公司而合并自己的公司来的划算。
- 如果选择合并，那么一定合并最大的子公司和次大的子公司。这由上面一点可以得到。
- 如果对方上一步是合并，那么你的决策唯一。因为如果你能干掉对方最大的子公司，那么无论对方之后如何合并，你都能干掉他，也就是说你获得了比赛的胜利。

根据以上三点，可以知道，只有第一回合能做出决策，后面的操作都是确定的：如果合并，那么你可能被干掉，或者对方合并，而对方合并会导致你合并或获得胜利……；而如果第一步干掉对方，对方下一步就会合并，而又陷入

这个循环。也就是说：一旦有人合并，那么两个人就会一直执行合并操作，直到某人确定他一定能获得胜利。

2.9.3 时空复杂度

时间复杂度：$O(n+m)$。
空间复杂度：$O(n+m)$。

Chapter 3

ACM/ICPC World Finals 2011

Abstract

ACM/ICPC World Finals 2011 共 11 题。其中简单题 4 道，分别是：C、E、J、K；中等题 5 道：A、B、F、H、I；难题 2 道：D、G。

简单题中涉及了简单搜索、简单数据结构、坐标转化、变种背包和简单几何等知识点。中等题中有问题简化及贪心、较简单的解方程、动态规划及其优化、割点、二分套数据结构等考点。难题主要考到了复杂的网络流及其线性规划建模、较难的几何数学问题。

总体来说，思维复杂度强，知识点覆盖全面，给我们带来了很多启发。

以下是本人做题之后的一些理解与思路。

3.1 Problem A To Add or to Multiply

3.1.1 题目大意

对于一个数 x，有两种操作：数值加 a 或数值乘 m。

给出 $a, m, p, q, r, s(\leq 10^9)$，要求构造一个最短的操作的序列，使得 $\forall x, p \leq x \leq q$，执行该操作后得到的 y 满足 $r \leq y \leq s$。

如果有多解，输出字典序最小的。

3.1.2 算法

首先，因为操作是一个增函数，所以我们只要使得 p 和 q 可行即可。

然后，因为 $s \leq 10^9$，所以可以枚举 m 乘的次数 n。设每次乘之前加的次数分别为 $k_n, k_{n-1}, \ldots, k_1, k_0$。那么，需要满足 $(\overline{k_n, k_{n-1}, \ldots, k_1, k_0})_m \in \left[\lceil \frac{r-p\cdot m^n}{a} \rceil, \lfloor \frac{s-q\cdot m^n}{a} \rfloor\right]$。

设 $u = \lceil \frac{r-p\cdot m^n}{a} \rceil, v = \lfloor \frac{s-q\cdot m^n}{a} \rfloor$，由于要最小化 $\sum_{i=0}^{n} k_i$，所以可以贪心地从 k_n 开始，如果加上一之后未超过 v，那么就加，直到不小于 u。

最后，只需要更新维护答案即可。

3.1.3 时空复杂度

时间复杂度：$O(\log^2 s)$。

空间复杂度：$O(\log s)$。

3.2 Problem B Affine Mess

3.2.1 题目大意

给出三个格点，以及它们经过旋转、缩放、平移之后得到的三个格点。旋转是这样的：在以原点为中心，边长为20的正方形上任选一点，旋转坐标轴使x轴通过该点，并保持原先的单位长度不变，旋转之后，图上的所有点四舍五入跳转到了距离最近的整点。注意形如 $-n + 0.5(n > 0)$ 的坐标跳转到了 $-n$。缩放是指 x 坐标和 y 坐标分别缩放，且缩放系数为非零整数。平移时，平移量为整数。

旋转一定最先，缩放和平移顺序未知。问是否存在一组或多组操作序列，如果存在，问它们是否本质相同。

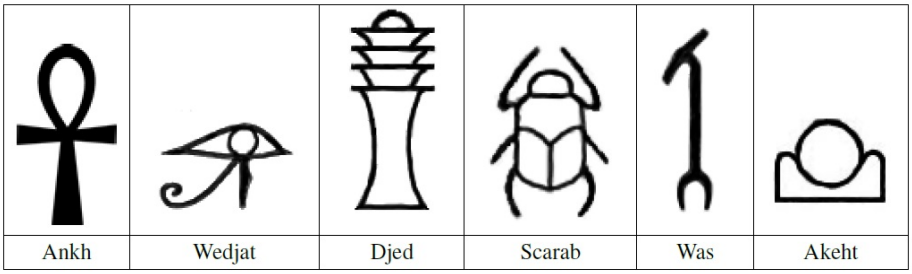

Figure 3.1: 六个象形文字

3.2.2 算法

首先，我们需要枚举每个点对应的点，总共有 6 种情况。然后，可以枚举旋转的量，由于旋转 α 和 $\alpha + \pi$ 可以通过之后缩放比例负数来得到本质相同的方案，所以我们枚举的旋转量的取值只有 40 种。

其次，我们需要分别对 x 和 y 坐标判断是否存在一种或多种本质不同的缩放和平移方法。设缩放系数为 k，平移量为 b，那么就得到了三个二元一次方程，判断其是否有解、是否多解、以及 k 的值是否为整数即可。

3.2.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

3.3 Problem C Ancient Messages

3.3.1 题目大意

给定如下六种标准象形字符，在一张 $H \times W (H, W \leq 200)$ 的图上有若干象形字符(可能被拉伸变形)，你需要写一个程序来识别。

3.3.2 算法

观察上面的图 3.1 可以发现，每个图形内部的洞都不一样，因此我们只需要找出其内部洞的个数就可以了。

3.3.3 时空复杂度

时间复杂度：$O(W \cdot H)$。
空间复杂度：$O(W \cdot H)$。

3.4 Problem D Chips Challenge

3.4.1 题目大意

一张 $n \times n (n \leq 40)$ 的芯片上，有些格子已经焊接有零件，有些格子禁止焊接零件。问，在满足下面约束的前提下，至多可以再焊接多少个零件。

- 每个格子至多焊接一个零件；
- 第 i 行上的零件总数与第 i 列一致；
- 任意一行的零件数不得超过总零件数的 $\frac{A}{B}$。

3.4.2 算法

此题在 WC2013 曹钦翔的讲课上有讲。首先，我们设变量 x_{ij} 表示 (i, j) 这个格子放的零件数，变量 $D_i = \sum_j x_{ij} = \sum_j x_{ji}$。假设已知 $X = ans \times \frac{A}{B}$，那么我们得到了以下约束：

$$
\begin{aligned}
x_{ij} &\geq 0 \\
x_{ij} &\geq 1 \qquad \text{(slot } (i,j) \text{ is already occupied)} \\
x_{ij} &\leq 1 \\
x_{ij} &\leq 0 \qquad \text{(slot } (i,j) \text{ is disabled)} \\
D_i &\leq X \\
D_i - \sum_j x_{ij} &= 0 \\
\sum_j x_{ji} - D_i &= 0
\end{aligned}
$$

然后，我们的目标函数是

$$\max \sum_i D_i$$

经过观察可以发现，约束的等式中 x_{ij} 和 D_i 都被减了一次、加了一次，符合网络流的格式。同时修改目标函数为 $-\min\sum_i -D_i$。也就是说，以等式为点，以变量为边，我们可以构出这样一张图：

总共 $2n$ 个点，左边 n 个设为 $1, 2, \ldots, n$，右边 n 个设为 $1', 2', \ldots, n'$。i 到 j' 连 x_{ij} 这条边(有上下限，费用为 0)，i' 到 i 连一条流量为 X，费用为 -1 的边。

于是，我们要求一个最小费用流。也就是说，在这张没有源和汇的图中，我们需要增广一些循环流，使得费用最小。出于负权边和下限，我们需要将一些边强制增广，并且删去一些边，这样流量就会不平衡，也就是点会有盈余。设点 i 的盈余 $e_i = \sum_j f_{ji} - \sum_j f_{ij}$，那么若 $e_i > 0$，则建源 s，向 i 连流量为 e_i、费用为 0 的边；若 $e_i < 0$，则建汇 t，i 向 t 连流量为 $-e_i$、费用为 0 的边。然后做最小费用最大流，如果 s 出去所有边都满流，那么说明这个网络流有解。

然后，我们得到目标函数最优解为 M，如果 $\frac{M \cdot A}{B} \geq X$，那么说明 M 是一个符合要求的解，也就是说，答案大于等于 M；否则，只能说明 M 不是一个可行的解。

这也是此题不能二分 X 的原因，我们只需要从大到小枚举 X，找到一个最大的符合要求的解即可。

此题用 zkw 费用流速度飞快。

3.4.3 时空复杂度

时间复杂度：$O(n \cdot \text{MinCostFlow}(n, n^2))$。

空间复杂度：$O(n^2)$。

3.5 Problem E Coffee Central

3.5.1 题目大意

给出一张 $dx \times dy (dx, dy \leq 1000)$ 的地图，其中有 $n(\leq 500000)$ 个咖啡店，已知它们的坐标。给出 $q(\leq 20)$ 个询问，每次询问曼哈顿距离在 m 以内可以抵达咖啡店的最大数目，以及该位置的坐标，如果有多个，以 y 为第一关键字，x 为第二关键字，输出最小的那个。

3.5.2 算法

首先，旋转坐标，控制范围就变成了一个正方形。

接着，维护前缀和，每次询问对每个位置询问其能到达的正方形内的咖啡店的个数，找出最大的即可。

3.5.3 时空复杂度

时间复杂度：$O(q \cdot dx \cdot dy)$。
空间复杂度：$O(dx \cdot dy)$。

3.6 Problem F Machine Works

3.6.1 题目大意

给出 $n(\leq 10^5)$ 台机器，对于第 i 台机器，你可以在第 d_i 天买进，其价格为 p_i，卖出价格为 r_i，其每天能带来的利润 g_i。现在你在第 0 天拥有 C 美元，但是不能在同一天拥有两台机器，买进或卖出机器的那一天不能生产获得利润。到第 $d+1$ 天，你将卖掉现有的机器，问当时最大拥有的钱为多少。除 n 外所有整数范围均为 10^9。

3.6.2 算法

动态规划，斜率优化。

首先，我们一定是按照当前机器数来动态规划的。亦即 $f(i)$ 表示买入 i 时最大数量的钱。

可以对机器按照 d 值排序，得到方程：

$$f(i) = \max\{f(j) + (d_i - d_j - 1)g_j + r_j - p_i\}$$

亦即

$$f(i) = \max\{g_j \cdot d_i + f(j) + r_j - (d_j + 1)g_j\} - p_i$$

然后我们就可以用斜率优化来做，但是由于斜率 $k = g_j$ 不一定是单调的，所以需要平衡树来维护。

但是，其实我们可以做到 k 递增，因为购买的一系列机器一定是 g 递增的，否则一定不如不换，于是我们可以将机器按 g 排序，从而使得斜率递增。这样，我们只需要用栈维护直线就行了。

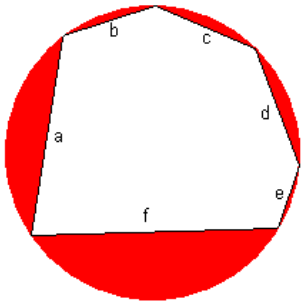

Figure 3.2

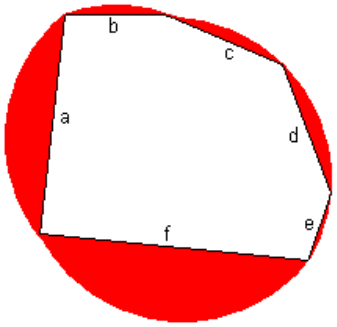

Figure 3.3

3.6.3 时空复杂度

时间复杂度：$O(n \log n)$。
空间复杂度：$O(n)$。

3.7 Problem G Magic Sticks

3.7.1 题目大意

给出 $n(n \leq 500)$ 根连在一起的木棍，第 i 根木棍的长度为 a_i，现在可以把木棍分成好多份，分别围成多边形，要求这些多边形的面积的最大值。

3.7.2 算法

数学。

首先，此题最需要解决的一个问题是如果不分组，那么构成的多边形面积最大是多少。

Conclusion 1. *多边形的所有点都在同一个圆上。*

Proof. 首先将这个多边形经过一系列调整，使得其所有点都在同一个圆上，如图 3.2 ，并将每条边与它上方的红色弓形部分“粘合”。

然后，如果多边形的形状发生任意改变(每条边上方的弓形部分与这条边一起移动)，如图 3.3 ，整个图形的周长并未发生变化，红色部分面积也没有发生变化，唯一改变的是多边形的面积。而众所周知，若干图形边长一定，那么圆形的面积最大，也就是说图 3.2 的总面积大于图 3.3 ，而红色部分面积不变，所以多边形面积图 3.2 大于图 3.3 。 □

这样我们就可以二分大圆的半径，看看是否能放下所以的边，从而得到多边形的最大面积。**注意**圆心在多边形外部的情况。

然后我们设计动态规划，$f(i)$ 表示用了前 i 条边总面积的最大值。转移显然。

但是这样是 $O(n^3 \log P)$的(P表示精度要求)。无法通过。

于是我们经过仔细地观察发现这样一个结论。

Conclusion 2. *如果存在分开更优的情况，那么一定是将最大边删除，并且要求圆心在多边形外部。*

我还无法严谨地证明这个结论，但是应该是不乏巧妙证明的。

于是，我们只需要每次找出最大边，将它删除之后再递归左右分别做就行了。

3.7.3 时空复杂度

时间复杂度：$O(n^2 \log P)$。

空间复杂度：$O(n^2 \log P)$。

3.8 Problem H
Mining Your Own Business

3.8.1 题目大意

n 个矿点，$m(m \leq 50000)$ 个通道分别连接某两个矿，一旦一个矿点塌方，另外每个矿至少要有一个与它相连的矿有竖井。问最少需要的竖井数和该情况下的方案数。

3.8.2 算法

首先，显然是关于割点。先把割点拎出，剩下来的联通块都缩成点，将形成一棵树（注意与块和桥的区别）。作为叶子的联通块（只有一个割点与它相邻）需要建竖井。

注意如果没有割点，那么答案需要特判。

3.8.3 时空复杂度

时间复杂度：$O(m)$。

空间复杂度：$O(m)$。

3.9 Problem I Mummy Madness

3.9.1 题目大意

有 n 个木乃伊，坐标为 $x_i, y_i(-10^6 \leq x_i, y_i \leq 10^6)$。你现在在原点，每次你可以向你相邻的八个方向移动一步，而木乃伊也会同样向八个方向移动，当然他们会选择一个最靠近你的方向。问你最多能走几步。

3.9.2 算法

二分。

首先，我们有一个直观的猜想：二分走的步数 L，如果你能走到的格子都能被木乃伊走到，那么就会被抓住，否则不会。这个非常显然，证明也比较容易，只需要证明两点：

- 你不可能用飘逸的步伐把本来能追上你的木乃伊甩开。
- 如果 L 步可行，那么 L 步能走的格子一定与 $L-1$ 步能走的格子相邻。

然后，我们可以确定二分的上界 $r = \max\{abs(x_i), abs(y_i)\}$ ，这个也是容易证明的，只需要证明：如果 r 步可行，那么一定存在一个在角落上的格子(即 $(r,r),(r,-r),(-r,r),(-r,-r)$)是安全的。

最后，我们要处理一个等边长正方形覆盖的问题，显然是可以用扫描线加数据结构做的，但是由于正方形等边长，所以也可以 $O(r)$ 维护四个角被覆盖的信息，然后 $O(r)$ 扫一遍判断。

3.9.3 时空复杂度

时间复杂度：$O(r \log r)$。

空间复杂度：$O(r)$。

3.10 Problem J Pyramids

3.10.1 题目大意

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，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”。

现在你有 $n(\leq 10^6)$ 个石块，需要把它们搭成若干座金字塔，并且满足以下条件：

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……

输出方案。

3.10.2 算法

首先，可以发现，金字塔的个数是 $O(\sqrt[3]{n})$，具体来讲是 320 个，也就是说我们做的是有 320 个物品的背包，似乎难以完成这个任务。

但是，我们打表发现，如果有解，那么答案不大于 6。如果数据是单组，那么我们直接做 6 次动态规划，每次找到能放进去的最大的物品，然后就能过了。

如果数据是多组，我们有两种做法，第一种做法是在动态规划时记录其方案，最后直接输出即可。第二种方法是预处理一个 DP 数组 $f[i][j]$ 表示用 j 个物品填满 i 的空间，最大物品的最小值是多少，询问的时候，从大到小枚举放的物品 i，如果 $f[n-v_i][k-1] < i$ 那么说明 i 可行。

最后，搜索也是实际运行较快的算法。

3.10.3 时空复杂度

时间复杂度：$O(n^{\frac{4}{3}}k) - O(\sqrt[3]{n}+k)$。$k$ 表示答案大小。

空间复杂度：$O(nk)$。

3.11 Problem K Trash Removal

3.11.1 题目大意

给出一个 $n(n \leq 100)$ 个点的多边形，要求计算让该多边形通过的最小管道宽度。

3.11.2 算法

首先，管道两边至少分别有一个点。

其次，如果我们卡住了这两个点，并且没有被某条边卡住，那么我们一定能将多边形朝某个方向旋转使得管道变窄。所以，一定有一条边和一个点分别被卡在管道两边。

所以，我只需要枚举这条边，找到距离它最远的点即可。

当然这条边一定是在凸包上的，所以我们可以使用旋转卡壳来做到 $O(n)$。

3.11.3 时空复杂度

时间复杂度：$O(n)$。

空间复杂度：$O(n)$。

Chapter 4

ACM/ICPC World Finals 2010

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

Table 4.1

4.1 Problem B Barcodes

4.1.1 题目大意

条形码是将一个串的末尾加上两个关于这个串的检验字符后，再在头尾分别加上与串无关的开始和结束标志，将串中字符逐个编码成 01 串（对应方式如表4.1），并用 0 隔开。其中 0 代表窄区域，1 表示宽区域，宽区域的宽度是窄区域的两倍。

给出 $n(n \leq 150)$ 个区域的条形码，但是一些区域可能与实际宽度有 5% 的误差。

问该条形码能否被成功解码，检验字符是否正确。

4.1.2 算法

首先，对于所有区域宽度，排序后枚举哪些是窄区域，得到双倍宽度的上下界，判断是否满足误差。

然后，模拟即可，不要遗漏条件。

4.1.3 时空复杂度

时间复杂度：$O(n^2)$。
空间复杂度：$O(n)$。

4.2 Problem C Tracking Bio-bots

4.2.1 题目大意

给出一个 $n \times m$ 的网格$(n, m \leq 10^6)$，中间有 $w(\leq 1000)$ 个横着的墙。机器人只能向左下方向走，出口在最左下角。问有多少位置的机器人是无法到达出口的。

4.2.2 算法

对横坐标、纵坐标分别离散，然后从出口向右上 DFS 找出所有能到的格子即可。

4.2.3 时空复杂度

时间复杂度：$O(w^2)$。
空间复杂度：$O(w^2)$。

4.3 Problem D Castles

4.3.1 题目大意

给出一棵 $n(n \leq 100)$ 个点的树，每个节点是一个城堡，已知攻克这个城堡需要的士兵数量、会损失的士兵数量和需要镇守的士兵数量。每条路一个方向只能走一次。问如果第一个城堡任选，最少需要多少士兵才能攻克和镇守所有城堡。

4.3.2 算法

贪心。

首先枚举第一个城堡，把它作为根，设 $f(i)$ 为攻克和镇守以 i 为根的子树所需的士兵数，$g(i)$ 为攻克和镇守以 i 为根的子树所损失的士兵数。

那么计算时我们需要对 i 的所有子树定一个顺序。结论是按 $f(x)-g(x)$ 从大到小排序，以下证明：

Proof. 如果要满足答案尽量小，那么对于第 x 个被攻克的子树，设其 f 值为 a_1、g 值为 b_1，第 $x+1$ 个被攻克的子树 f 值为 a_2、g 值为 b_2，第 $1\sim x-1$ 个被攻克的子树的 g 值的和为 s，那么有：

$$\max\{a_1+s, a_2+b_1+s\} \le \max\{a_2+s, a_1+b_2+s\}$$

否则交换 x 与 $x+1$ 答案不会更劣。

若 $a_2 > a_1+b_2$，那么 $a_2 \ge a_2+b_1$，矛盾。

$\therefore a_1+b_2 \ge a_2, \therefore a_1+b_2 \ge a_2+b_1$，即 $a_1-b_1 \ge a_2-b_2$。 □

4.3.3 时空复杂度

时间复杂度：$O(n^2 \log n)$。

空间复杂度：$O(n)$。

4.4 Problem E Channel

4.4.1 题目大意

给出一个 $n \times m$ 的格子矩阵($n \le 20, m \le 9$)，其中有一些障碍。现在要从左上角开始挖渠道直到右下角，渠道宽度为一格，并且渠道不能自交，即使是在角上碰到也不行。问使得渠道最长的方案。

4.4.2 算法

连通性状态压缩动态规划。

每个格子有四种状态：左插头、右插头、没插头但是有渠道、空着。逐格转移。而且要多记一个当前格子左上角是否有渠道。

具体细节请看代码。

4.4.3 时空复杂度

时间复杂度：$O(nm \cdot 4^m)$。

空间复杂度：$O(nm \cdot 4^m)$。

4.5 Problem F Contour Mapping

4.5.1 题目大意

将一个地图划分成若干边长为 d 的等边三角形，三角形顶点的海拔已知，将每个三角形视为一个平面，并把所有海拔为 h 倍数的地方画上等高线（当一整块区域水平时，只在边界处有等高线）。

问等高线总长度。

4.5.2 算法

首先，我们对于一个三角形内部的等高线，可以用等差数列来做，举例来说：

设 ΔABC 中 A 点海拔最低，那么端点在 AB 和 AC 上的等高线长度为：

$$\sum_{i=h_A/h}^{\min\{h_B,h_C\}/h} \sqrt{\left(\frac{ih-h_A}{2(h_B-h_A)}+\frac{ih-h_A}{2(h_C-h_A)}\right)^2+\left(\frac{\sqrt{3}(ih-h_A)}{2(h_B-h_A)}-\frac{\sqrt{3}(ih-h_A)}{2(h_C-h_A)}\right)^2}$$

将 $(ih-h_A)$ 提出，之后都是常数。当然请注意边界。

然后对于三角形边界的等高线，设 ΔABC 和 ΔABD 中 $h_A=h_B$ 且是 h 的倍数，如果 $h_A=h_B=h_C=h_D$ 那么不算是等高线，否则是等高线。注意整个图的边界。

4.5.3 时空复杂度

时间复杂度：$O(s\cdot p)$。

空间复杂度：$O(s\cdot p)$。

4.6 Problem G The Islands

4.6.1 题目大意

给出二维平面上横坐标递增的 $n(\leq 100)$ 个点，要求从 0 号点出发一直朝 x 正方向到达 $n-1$ 号点，再从 $n-1$ 号点一直朝 x 反方向回到 0 号点，中途需要经过所有点，且有两个特殊岛屿必须在不同的路径中被访问到。输出总路程最小的方案。

4.6.2 算法

动态规划。

设 $f(i,j)$ 表示 $[0,\max\{i,j\}]$ 中的点都被访问过且第一条路径在 i 号点，第二条路径在 j 号点的最小距离。

转移时枚举哪一条路径到达 $\max\{i,j\}+1$ 这个点即可。

4.6.3 时空复杂度

时间复杂度：$O(n^2)$。

空间复杂度：$O(n^2)$。

4.7 Problem H Rain

4.7.1 题目大意

给定某个地区的三角形划分模型，即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。

该区域的三角形划分的点数 $n \leq 2,704$。

4.7.2 算法

首先，一个湖泊是由很多相邻的且水位相等的点组成的，因此，如果我们已知了所有点的水位高度，那么只需要 FloodFill 就能找出湖泊了。

然后，有一个明显的结论：某个点的水位高度一定是它到边界的所有路径上的最大值中的最小值，亦即：在这一段不断地加水，直到水会溢出到边界，此时有水的路径中的最大值就是当前的水位。因此我们记 $d(i)$ 为 i 点水位，那么有 $d(i) \leq \max\{d(j),h(i)\}$。

于是，只要找出边界，然后做一遍最短路就行了。找边界的话，可以对每条边判断它是否是边界，即它连能构出的三角形都在它的同一侧。

4.7.3 时空复杂度

时间复杂度：$O(m \log n)$。

空间复杂度：$O(m)$。

4.8 Problem I Robots on Ice

4.8.1 题目大意

给出一个 $n \times m$ 的棋盘，需要找一条从 $(0,0)$ 到 $0,1$ 的哈密尔顿路，其中有 3 个点特殊点，即在 $[i \cdot n \cdot m/4]$ 时刻必须到达点 (r_i, c_i)。问方案数。

4.8.2 算法

搜索加剪枝。

首先，如果当前到特殊点的距离大于时间差，那么就不可能了。

其次，如果当前连通块个数大于一，那么也不可能。

再次，如果某个格子的度数小于 2 且它不是终点，也就不可能了。

4.8.3 时空复杂度

时间复杂度：$O(3^{nm})$。

空间复杂度：$O(nm)$。

4.9 Problem J Sharing Chocolate

4.9.1 题目大意

$x \times y (x, y \leq 100)$ 大小的巧克力，需要分成 $n (n \leq 15)$ 份，每份的大小已知。每次只能将一块巧克力掰成两半，问是否可行。

4.9.2 算法

动态规划。

设 $f(S, x)$ 为大小为 $x \times \frac{sum(S)}{x}$ 的巧克力能否分成 S 集合的目标巧克力。

转移时只需枚举枚举 S 的子集 S'，并判断 $sum(S')$ 是不是 x 或 $\frac{sum(S)}{x}$ 的倍数，然后分别通过 $f(S', x)$ and $f(S \setminus S', x)$ 和 $f(S', \frac{sum(S)}{x})$ and $f(S \setminus S', \frac{sum(S)}{x})$ 转移。

4.9.3 时空复杂度

时间复杂度：$O(3^n \cdot X)$。

空间复杂度：$O(2^n \cdot X)$。

4.10 Problem K Paperweight

4.10.1 题目大意

给出一个六面体和其内部一个点 F，问其在放置足够稳定的情况下，F 距离桌面的最大和最小距离。如果在一种放置方案中，纸镇的重心向任意方向移动 0.2 个单位后，纸镇仍不会发生移动，则这种放置方案被认为是“足够稳定”的。

4.10.2 算法

首先，求出六面体的重心。由于四面体的重心（中心）即为四个顶点坐标的算术平均值，那么六面体的重心即为两个四面体重心关于其体积的加权平均值。

然后，枚举一个面（不一定是某个表面），判断所有点是不是都在这个面的正方向。对于所有在面上的向量，判断它是否是底面的边界（判断是否所有在面上的点都在其正方向），然后计算重心到面上的投影到它的距离。如果所有距离都不小于 0.2，那么更新答案。

当然以上计算需要对三维向量有充分的认识。

4.10.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

Chapter 5

ACM/ICPC World Finals 2009

Abstract

ACM/ICPC World Finals 2009 共 11 题。其中简单题 4 道，分别是：A、B、F、I；中等题 4 道：C、D、H、K；难题 3 道：E、G、J。

简单题中涉及了二分、贪心、枚举、模拟、凸包、动态规划以及树上的模拟等知识点。中等题中有几何、2-SAT、最短路以及其模型转化等考点。难题主要考到了较难的模型建立、复杂的 min-max 搜索以及 alpha-beta 剪枝和较难动态规划。

总体来说，思维复杂度强，知识点覆盖全面，给我们带来了很多启发。

以下是本人做题之后的一些理解与思路。

5.1 Problem A A Careful Approach

5.1.1 题目大意

有 $n(\leq 8)$ 个未知数 x_i，已知 a_i, b_i，要求在 $x_i \in [a_i, b_i]$ 的约束下，使得 $\min_{i,j}\{|x_i - x_j|\}$ 最大。

5.1.2 算法

首先，此类最小最大问题可以用二分答案来解决。

如果已知答案以及 x_i 从大到小排好之后的顺序，那么我们就可以贪心地将 x_i 取到最小。

由于 n 较小，可以枚举所有可能的排列，当然也可以用动态规划来加速：$f(S)$ 表示已经将 S 集合中的未知数放好，其中最大的未知数的最小值，转移显然。

5.1.3 时空复杂度

时间复杂度：$O(n \cdot 2^n \log T)$。T 表示 b_i 的最大值。

空间复杂度：$O(2^n)$。

5.2 Problem B My Bad

5.2.1 题目大意

给出一个逻辑电路，其中有 $n(\leq 8)$ 个输入、$g(\leq 19)$ 个门、$u(\leq 19)$个输出。其中某个门可能会出故障，即与正确答案相反或总是给出 0 或总是给出 1。

现在对这个逻辑电路做了 b 个实验，得到了一些输入对应的输出，问是哪个门出了哪个故障。

5.2.2 算法

枚举所有可能的情况，模拟判断即可。

5.2.3 时空复杂度

时间复杂度：$O(gb(n+g+u))$。

空间复杂度：$O(b(n+u)+g)$。

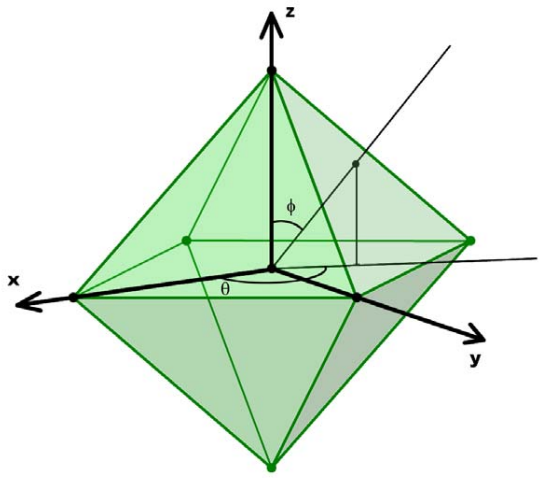

Figure 5.1: 正八面体

5.3 Problem C The return of Carl

5.3.1 题目大意

给出一个所有边长均为 10 的由八个互为全等的正三角形组成的正八面体，如图 5.1 。

给出八面体表面上两个点，问如果只能在其表面行走，最短的距离是多少。

5.3.2 算法

首先，可以将问题简化为起点在 xy 坐标第一象限上方，按逆时针方向走到终点，且终点不会在 xy 坐标第四象限。

这样，如果终点在 xy 坐标平面上方，那么直接将上方三个面摊平即可。

如果终点 xy 坐标第一象限下方，那么也只需将上下两个面摊平即可。

如果终点 xy 坐标第二象限下方，那么有两种情况，不赘述。

如果终点 xy 坐标第三象限下方，那么有三种情况，同不赘述。

5.3.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

5.4 Problem D Conduit Packing

5.4.1 题目大意

给出 4 个已知半径的圆，要求能将这 4 个圆包含的大圆的最小半径。

5.4.2 算法

首先，二分大圆的半径 R。然后，我们将圆放入这个大圆，每个圆尽量贴近之前放入的圆并与大圆相切。

假设当前放入的圆的半径为 r_1、与它贴住的圆的半径为 r_2。那么两个圆的圆心和大圆的圆心构成了一个边长分别为 $R-r_1, R-r_2, r_1+r_2$ 三角形，这样我们就可以确定当前放入的圆的位置，即其弧度。

这样，我们就可以知道先枚举放入的顺序，然后判断即可。

5.4.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

5.5 Problem E Fare and Balanced

5.5.1 题目大意

给出 $n(\leq 50000)$ 个点 $m(\leq 50000)$ 条边的带权拓扑图，并且保证其拓扑序的第一个一定是 1，最后一个一定是 n。要求给一些边增加权值，使得从 1 到 n 的所有路径的权值和相等。并且使得不存在一条路径，含有超过一条增加了权值的边。最后，需要最小化 1 到 n 的距离。输出方案。

5.5.2 算法

此题非常容易出错，我们需要建立一个完善的模型。

设 $U(u,v)$ 表示 u 到 v 的所有路径是否有唯一的权值，如果 $U(1,n)$ 是 true，那么就不用更改。如果 $U(1,v)=U(v,n)=$ false，那么就一定无解。因此，之后我们默认，如果 $U(1,v)=$ false，那么 $U(v,n)=$ true。

由于 $U(1,1)=\text{true}, U(1,n)=\text{false}$，并且，如果存在一条边 (u,v,c)，且 $U(1,u)=$ false，则 $U(1,v)=$ false。这是一个割的模型，即将 $U(1,v)=$ true

的 v 划为 S 部，$U(1,v) = \text{false}$ 的 v 划为 T 部，割集就是所有 $U(1,u) = \text{true}, U(1,v) = \text{false}$ 的边 (u,v,c)。

再设 $C(u,v)$ 表示 u 到 v 的所有路径中的最大值，我们将所有割边 (u,v,c) 都加上一个权值 $C(1,n)-C(1,u)-C(v,n)-c$，由于 $U(1,u) = \text{true}$ 且 $U(v,n) = \text{true}$，所以所有经过这条割边的路径的长度都为 $C(1,n)$，并且根据割的性质，所有 1 到 n 的路径都经过且只经过一条割边。这样我们就构造出解了。

5.5.3 时空复杂度

时间复杂度：$O(n+m)$。

空间复杂度：$O(n+m)$。

5.6 Problem F Deer-Proof Fence

5.6.1 题目大意

给出 $n(\leq 9)$ 个半径为 $m(\leq 200)$ 的圆，要求用篱笆将所有圆围住，可以用多个独立的围栏。问围栏的最小长度。

5.6.2 算法

动态规划。

首先，如果要将所有圆用一个围栏围住，则围栏长度为这些圆圆心构成凸包的周长加上一个圆的周长。

然后我们就可以用 $f(S)$ 表示已经围住了 S 集合的圆的围栏长度，然后枚举它的一个子集来转移。

5.6.3 时空复杂度

时间复杂度：$O(2^n \cdot n)$。

空间复杂度：$O(2^n)$。

5.7 Problem G House of Cards

5.7.1 题目大意

Axel 和 Birgit 喜欢玩这样的一种纸牌游戏：他们建造一个由纸牌组成的房子，当他们添加纸牌到房子的时候，会获得（或失去）游戏的分数。Axel

和 Birgit 只使用 2 种花色，1 红 1 黑。每种花色有 13 个等级。我们使用记号 1R,2R,...,13R,1B,2B,...,13B 来表示等级和颜色。

开始前，玩家要选择半副标准纸牌的一个子集，子集中所有纸牌的最大等级是 M。洗完选出的纸牌后，他们从牌堆的最上面拿出 8 张，从左到右连续地放置它们形成4个“山峰”。举个例子，如果 $M = 13$ 而且前 10 张纸牌（26 张的前 10 张）是：

6B 3R 5B 2B 1B 5R 13R 7B 11R 1R ...

那么这个游戏开始的时候就像图 5.2 所展示的那样。

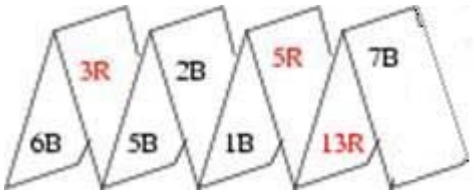

Figure 5.2: 由牌堆的最上面 8 张牌组成的山峰和山谷

剩下的纸牌正面朝上被放置成一排。

每个玩家被认定一种颜色，红色或黑色。Birgit 总被认定是黑色，Axel 总被认定是红色。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。图 5.2 的那个例子，Birgit 先开始，因为第一张纸牌是 6B。

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。

图 5.2 的那个例子，如果 Birgit 放置她的纸牌（11R）到中间的山谷上，她将获得 14 分。如果她放置她的纸牌到左边的山谷上，Axel 获得 19 分。如果她放置她的纸牌到右边的山谷上，Axel 获得 29 分。

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）。

当这个游戏结束后，分数低的玩家将要支付一定数量的瑞典克朗给另一个玩家，数量等同于两个玩家的分数差。如果是平局，就不用支付。

你必须写一个程序读入一副被洗过的牌堆和一个玩家的名字，然后找出这个玩家最多能赢多少（或者最少能输多少），假设另一个玩家总是采取最优策略。

5.7.2 算法

min-max 搜索，alpha-beta 剪枝。

首先，这是一个标准的 min-max 搜索，然后直接套用加上 alpha-beta 剪枝就能够通过了。

5.7.3 时空复杂度

时间复杂度：$O(16^M)$。

空间复杂度：$O(M)$。

5.8 Problem H
The Ministers' Major Mess

5.8.1 题目大意

$n(n \leq 100)$ 个布尔变量，$m(m \leq 500)$ 个条件，每个条件中最多有 4 个类似第 i 个变量值是 `true/false` 的命题，要满足每个条件中有超过一半是真命题。求每个布尔变量是否可能是 `true/false`。

5.8.2 算法

根据题目要求，如果条件中命题数不超过 2，那么这些命题必须满足；否则，由于最多只有 4 个命题，所以最多只有一个假命题，也就是说，这些命题之间满足‘或’关系。显而易见，这些都是满足 2–SAT 的，所以直接套用 2–SAT 即可。

由于数据范围不大，我们只要对每个 2–SAT 中的点，DFS 出它的所有后继节点，判断这些点之间是否有冲突即可。

5.8.3 时空复杂度

时间复杂度：$O(nm)$。

空间复杂度：$O(n+m)$。

5.9 Problem I Struts and Springs

5.9.1 题目大意

平面上有 n 个窗户，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。

5.9.2 算法

首先对每个窗户找出直接包含它的窗户，形成一棵树。

每次对根进行操作，然后会影响其儿子的大小，递归处理即可。

5.9.3 时空复杂度

时间复杂度：$O(n \cdot q)$。

空间复杂度：$O(n)$。

5.10 Problem J Subway Timing

5.10.1 题目大意

给出一个 $n(\leq 100)$ 个地铁站的树形地铁网络，已知相邻两个地铁站之间的运行时间（以秒为单位的整数），现在要将其标识在图上，但是需要表示成以分为单位的整数，也就说可以向上或者向下取整。这样为造成一些误差，问如果从任意一点走到另一点，造成最大误差的最小值。

5.10.2 算法

动态规划。

此题有个结论但是很难证明：答案不大于 118。当然没有这个结论也是可做的。

首先，我们二分答案，这样，我们就不用记录子树内部的路径的值。所以，设 $f(i,j)$ 表示以点 i 为根的子树中，终点为 i 的所有路径的最小值**不小于** j 时，最大值的最小值，也就是说，点 i 为根的子树中，终点为 i 的所有路径误差可以在 $[j, f(i,j)]$ 这个区间中。

假设 i 有 m 个儿子，将他们分别标为 $1 \ldots m$。设 $g(k,j)$ 表示以点 i 为根的子树中，经过点 k 且终点为 i 的所有路径的最小值**不小于** j 时，最大值的最小值，g 值好求。当我们需要求 $f(i,j)$ 的值时，首先根据定义 $f(i,j) \leq f(i,j+1)$，然后可以枚举其中一个点 k，它带来的路径的区间是 $[j, g(k,j)]$（如果没有这样的点，那么这种方案一定包含在 $f(i,j+1)$ 中），然后，由于有了这样一个点，所以另外点带来的路径区间的下限应该不小于 $l = max(j, -ans - j)$，并且他们

的上限也应该和不超过 ans。也就是说，我们需要维护一些前缀和后缀信息来得到除去 k 这个点之后，剩下的点 $k' \in [1, k-1] \cup [k+1, m]$的 $g(k', l)$的最大值，以及这些 $g(k', l)$ 中会不会有两个加起来超过 ans。这样就完成了转移的过程。

5.10.3 时空复杂度

时间复杂度：$O(n \cdot U \log U)$。U 表示答案的范围，如果已知结论，那么 $U = 118$，否则 $U = 59n$。

空间复杂度：$O(n \cdot U)$。

5.11 Problem K
Suffix-Replacement Grammars

5.11.1 题目大意

给出两个长度相等且不大于 20 的串 s, t，再给出 $m(\leq 100)$ 个规则 (a, b)，a, b 是两个长度相等的字符串，如果 a 是一个串的后缀，那么就可以花费 1 的代价用 b 替换它。问将 s 变成 t 所需的最小代价。

5.11.2 算法

首先，我们有这样一个想法，假设当前字符串的长度为 l，那么如果已知它先后使用了某些长度为 l 的规则，我们就可以将问题变为许多子问题（可能只有一个），并且这些子问题中字符串长度为 $l-1$。

这样，我们发现在所有这些子问题中涉及到的串一定是读入的所有串的某个后缀，然后，我们以长度从小到大来完成它们之间的最短距离。也就是说，设 $d_l(i, j)$ 表示从 i 这个串变成 j 这个串需要的最小花费（i, j 长度均为 l），那么有 $d_{a.length()}(a, b) \leq 1$。然后，从小到大枚举 l，如果 $i[0] = j[0]$，则$d_l(i, j) \leq d_{l-1}(i[1, l-1], j[1, l-1])$，接着用 Floyd 得到 i 到 j 的距离 $d_l(i, j)$。

还要注意：答案可能会超过 32bit、常数需要优化。

5.11.3 时空复杂度

时间复杂度：$O(l \cdot n^3)$。

空间复杂度：$O(l \cdot n^3)$。

Chapter 6

ACM/ICPC World Finals 2008

6.1 Problem A
Air Conditioning Machinery

6.1.1 题目大意

一个三维的 $x_{max} \times y_{max} \times z_{max} (x_{max}, y_{max}, z_{max} \leq 20)$ 的长方体空间，要求用一些（但不超过六个）“肘状物”（如图 6.1 ）构成从入口到出口的管道。“肘状物”不能重叠。

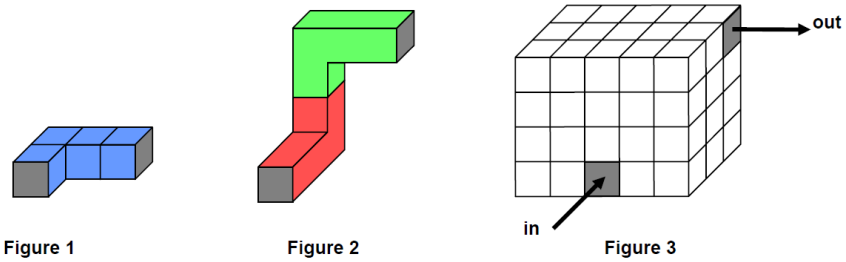

Figure 6.1

问最少需要多少“肘状物”。

6.1.2 算法

搜索，剪枝。

由于数据范围不大，我们可以直接采用搜索。

首先，最优性剪枝必不可少。

其次，我们可以设计一个估价函数：$d[x][y][z][dir]$ 表示目前在 (x, y, z) 这一格、方向是 dir，在不考虑重叠的情况下到达出口所需要的最少步数。这样我们在搜索过程中，如果已经走了的步数加上当前这个位置和方向的估价超过 6，那么可以直接退出。这个剪枝非常强，加上后搜索次数非常少，只是预处理稍微慢了一些。数据增大的话，其优势会更加明显。

6.1.3 时空复杂度

时间复杂度：预处理 $O(x_{max} \cdot y_{max} \cdot z_{max})$，搜索复杂度无法估计，但是运行时间远远小于预处理复杂度。

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

6.2 Problem B Always an Integer

6.2.1 题目大意

给出一个 $n(\leq 100)$ 阶多项式 $P = \frac{Q}{D}$，其中多项式 Q 的系数均为整数，正整数 $D \leq 2^{31} - 1$。问是否对于任意正整数 n 都有 $P(n)$ 是整数。

6.2.2 算法

问题简化为整数多项式 Q 是否能被 D 整除。

Conclusion 3. 对于一个 n 阶多项式 P，如果 $P(1), P(2), \ldots, P(n+1)$ 都能被 D 整除，则对于任意正整数 n 都有 $P(n)$ 能被 D 整除。

Proof. 数学归纳法证明。

对于 0 阶多项式 $P = c$，显然成立。

假设 $n-1$ 阶多项式成立，则对于 n 阶多项式 P（$n > 0$），如果对于 $x \in \mathbb{Z} \cup [1, n+1]$ 有 $P(x)$ 能被 D 整除，设 $Q(n) = P(n+1) - P(n)$，则对于 $x \in \mathbb{Z} \cup [1, n]$ 有 $Q(x) = P(x+1) - P(x)$ 能被 D 整除，且由于 Q 是 $n-1$ 阶多项式，所以对于任意正整数 n 都有 $Q(n)$ 能被 D 整除，而对于 $n > 1$ 有 $P(n) = Q(n-1) + P(n-1)$，所以对于任意正整数 n 都有 $P(n)$ 能被 D 整除。

□

6.2.3 时空复杂度

时间复杂度：$O(n^2)$。

空间复杂度：$O(n)$。

6.3 Problem D The Hare and the Hounds

6.3.1 题目大意

猎狗和兔子在进行公路拉力赛。参赛者（猎狗）必须找到组织者（兔子）选择的一条或多条路。他们在路上走的时候会遇到各种各样的十字路口。进入任何十字路口时（除了一些特殊的情况），他们都应该使用主要的道路规则。主要的道路规则总是“尽量直”，意思是转最小的角度。如果两条路转的角度相同，这时主要的道路规则要求选择右边那条。

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，他使用主要道路规则的方式略有不同，第一道路的选择是与原来一样的。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。

对于这个问题，你会得到一个路线图(一个配置的道路和路口)，选择点的列表，确认点的位置，$maxdis$ 的值，起始点与终点（不是选择点），离开起始点时的方向。有了这些信息，你就可以模拟猎狗追兔子时的路线，你可以认为猎狗使用这些策略总可以找到兔子的路线。

6.3.2 算法

按照题目模拟，注意起点等于终点的情况。

6.3.3 时空复杂度

时间复杂度：$O(ncp \cdot nroad^2)$。
空间复杂度：$O(nroad)$。

6.4 Problem F Glenbow Museum

6.4.1 题目大意

一个角序列是一个由 R 和 O 组成的序列，其中 R 表示内角为 90°，O 表示内角为 270°。那么一个只包含 R 和 O 的字符串能够粗略的表示一个直角多边形，如果存在一个直角多边形，且能从某一点监视到整个多边形内部，那么这个角序列合法。

问长度为 $L(L \leq 1000)$ 的合法角序列个数。

6.4.2 算法

首先，L 是偶数且不小于 4，否则答案为 0。

然后，我们发现，如果有两个 O 相邻，那么显然不可能。如果没有 O 相邻，那么一定可行。

也就是说，这个角序列不能有 O 相邻，且 R 的个数为 $\frac{L}{2}+2$，O 的个数为 $\frac{L}{2}-2$，这样就可以设计 $O(n^2)$ DP。

但是，其实有更好的做法，假设先前我们已经有了 ORORO...ORO，我们可以再插入 5 个 R，这样答案就是 $\binom{\frac{L}{2}+3}{5}$。当然，还需要减去头尾都是 O 的情况，也就是 $\binom{\frac{L}{2}+1}{5}$。

最终答案为 $\binom{\frac{L}{2}+3}{5}-\binom{\frac{L}{2}+1}{5}$。

6.4.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

6.5 Problem G Net Loss

6.5.1 题目大意

给出一个 $n(n \leq 10)$ 次多项式 $p(x)$ 和 c，求一个函数

$$g(x)=\begin{cases} k_1x+b_1 & -1 \leq x \leq c \\ k_0x+b_0 & c < x \leq 1 \end{cases} \qquad \text{其中 } k_1c+b_1=k_0c+b_0$$

最小化

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

6.5.2 算法

Lemma 1. 设 $b = k_1c + b_1$，假设 b 已知，那么 $\int_c^1 (p(x) - g(x))^2 \,\mathrm{d}x$ 是关于 k_0 的一个二次函数，$\int_{-1}^c (p(x) - g(x))^2 \,\mathrm{d}x$ 是关于 k_1 的一个二次函数。

Proof.

$$\begin{aligned}
&\int_c^1 (p(x) - g(x))^2 \,\mathrm{d}x \\
=&\int_c^1 (p(x) - k_0(x-c) - b)^2 \,\mathrm{d}x \\
=&\int_c^1 p^2(x) + k_0^2(x-c)^2 + b^2 - 2k_0p(x)(x-c) - 2bp(x) + 2k_0b(x-c) \,\mathrm{d}x \\
=&k_0^2 \cdot \int_c^1 (x-c)^2 \,\mathrm{d}x \\
&+ k_0 \cdot \left(-2\int_c^1 p(x)(x-c) \,\mathrm{d}x + 2b\int_c^1 (x-c) \,\mathrm{d}x\right) \\
&+ b^2\int_c^1 1 \,\mathrm{d}x - 2b\int_c^1 p(x) \,\mathrm{d}x + \int_c^1 p^2(x) \,\mathrm{d}x
\end{aligned}$$

同理

$$\begin{aligned}
&\int_{-1}^c (p(x) - g(x))^2 \,\mathrm{d}x \\
=&k_1^2 \cdot \int_{-1}^c (x-c)^2 \,\mathrm{d}x \\
&+ k_1 \cdot \left(-2\int_{-1}^c p(x)(x-c) \,\mathrm{d}x + 2b\int_{-1}^c (x-c) \,\mathrm{d}x\right) \\
&+ b^2\int_{-1}^c 1 \,\mathrm{d}x - 2b\int_{-1}^c p(x) \,\mathrm{d}x + \int_{-1}^c p^2(x) \,\mathrm{d}x
\end{aligned}$$

□

Theorem 6.5.1. 答案关于 b 是一个二次函数。

Proof. 设关于 k_0 的二次函数为

$$Ak_0^2 + (Bb + C)k_0 + (Db^2 + Eb + F)$$

那么最小值为

$$\frac{4A(Db^2 + Eb + F) - (Bb + C)^2}{4A}$$

关于 k_1 的同理。 □

当然我们也可以不用具体计算关于 b 的二次函数是什么，取三个点计算，然后用待定系数法求出系数即可。

6.5.3 时空复杂度

时间复杂度：$O(n^2)$。
空间复杂度：$O(n)$。

6.6 Problem H Painter

6.6.1 题目大意

给出平面上 n 个三角形，问是否有相交，如果没有，输出三角形最多嵌套多少层。

6.6.2 算法

如果只判断相交，那就是经典的线段相交问题：扫描线过去，用平衡树维护线段的上下关系，时刻保持相邻的线段不相交。

现在要求某个三角形嵌套的层数，那么就在插入这个三角形的第一条边的时候，询问其相邻的线段对应的三角形是否包含它，如果包含，那就是相邻的线段对应的三角形层数 $+1$，否则不用 $+1$。

6.6.3 时空复杂度

时间复杂度：$O(n \log n)$。
空间复杂度：$O(n)$。

6.7 Problem J The Sky is the Limit

6.7.1 题目大意

给出 n 个底在 x 轴上的等边三角形，问 x 轴上方的周长长度。

6.7.2 算法

求出所有交点的 x 坐标，那么两个每一小段对应的周长一定是一条线段，然后求哪条线段在最上面即可。

6.7.3 时空复杂度

时间复杂度：$O(n^3)$。
空间复杂度：$O(n^2)$。

Chapter 7

ACM/ICPC World Finals 2007

7.1 Problem A Consanguine Calculations

7.1.1 题目大意

给出父母孩子其中两个人的血型，求另一个人的可能的血型。

7.1.2 算法

首先，根据题意，两个标记无关，所以可以分开来算。
然后分别枚举所有可能的组合，判断是否符合即可。

7.1.3 时空复杂度

时间复杂度：$O(1)$。
空间复杂度：$O(1)$。

7.2 Problem I Water Tanks

7.2.1 题目大意

给出一排 n 个水箱，每个水箱的高度 h_i 已知，相邻的水箱之间有一根高度为 k_i 的管子连接。问当第一个水箱被水填满时，总共灌入了多少水。

保证 k_i 递增，你可以认为在水面以下 d 米的水压等于在表面气压加上 $0.097d$ 个大气压

7.2.2 算法

事件点，模拟。

把水箱被水分隔开的时间作为事件点，由于数据保证 k_i 递增，所以，一定是前 i 个水箱被单独隔开，后 $n-i$ 个水箱为一个空间。

我们将要做以下的工作：判断是否能到达下一个事件点，如果能，计算第 i 个水箱最后的水位，如果不能，计算后 $n-i$ 个水箱中总共进入多少水。总结以下，就是求当前水位为 h、大小为 v、压强为 p 的空间，当第一个水箱被水填满时，水位的上升的高度 x。可以通过解二次方程 $(0.097(h_1-h-x)+1)(v-x)=pv$ 求得。

7.2.3 时空复杂度

时间复杂度：$O(n)$。
空间复杂度：$O(n)$。

7.3 Problem J Tunnels

7.3.1 题目大意

n 个点 m 条边的图，一个人要从 1 号点走到 0 号点，你可以随时看到他的行动，并删掉一些边使得他无法到达 0 号点。输出最少需要删除的边数目。

7.3.2 算法

最小割。

设 d_i 为从 i 点出发需要删掉的边数，那么我们可以这样得到 d_i 的值：从小到大枚举 L，把所有 $d_j \leq L$ 的 j 从图中删除得到子图 G，然后用 $L + \mathrm{MinCut}(G, i, 0)$ 更新 d_i。

可行性显然。下面用归纳法证明其最优性：设 δ_i 为从 i 点出发需要删除的最少边数。假设我们已知所有 $d_i < I$ 的点 i 满足 $d_i \leq \delta_i$，那么只需证明对所有 $d_i = I$ 的 i 满足 $d_i \leq \delta_i$，就可以证明其最优性。我们可以使用反证法：

Proof. 假设在对方离开 i 点之前，你删掉了 l 条边，又因为反证时默认 $\delta_i < I$，所以对方将不能进入 δ 值 $\leq I - l - 1$ 的点(注意：当前删除的 l 条边将不会和之后的重复)。因此，可以将所有 $\delta_j \leq I - l - 1$ 的 j 从图中删除得到子图 G'，再根据 d_i 的计算方法，有 $I - l - 1 + \mathrm{MinCut}(G', i, 0) \geq I$，即 $l < \mathrm{MinCut}(G', i, 0)$，所以从 i 点将能到达 0，与题意不符，因此有 $I \leq \delta_i$。 □

7.3.3 时空复杂度

时间复杂度：$O(n \cdot \mathrm{MaxFlow}(n, m))$。
空间复杂度：$O(m)$。

Chapter 8

ACM/ICPC World Finals 2006

8.1 Problem F Building a Clock

8.1.1 题目大意

给出一个转速为 R 的主动轴和 n 个齿轮，要求组装出时针和分针的轴，输出转轴最少的方案，如果有多种，输出齿轮最少的，如果还有多种，输出字典序最小的。

8.1.2 算法

搜索。

枚举轴的个数，主动轴到分界轴的距离，分界轴到分针轴的距离，可以得到分界轴到时针轴的距离，然后搜索连接相邻两个轴之间的齿轮即可。

8.1.3 时空复杂度

时间复杂度：$O(n^{2n+3})$。

空间复杂度：$O(n^2)$。

8.2 Problem G Pilgrimage

8.2.1 题目大意

给出一个转速为 R 的主动轴和 n 个齿轮，要求组装出时针和分针的轴，输出转轴最少的方案，如果有多种，输出齿轮最少的，如果还有多种，输出字典序最小的。

8.2.2 算法

搜索。

枚举轴的个数，主动轴到分界轴的距离，分界轴到分针轴的距离，可以得到分界轴到时针轴的距离，然后搜索连接相邻两个轴之间的齿轮即可。

8.2.3 时空复杂度

时间复杂度：$O(n^{2n+3})$。

空间复杂度：$O(n^2)$。

Chapter 9

ACM/ICPC World Finals 2005

9.1 Problem C
The Traveling Judges Problem

9.1.1 题目大意

$n(\leq 20)$ 个点 m 条边的无向带权图，现有 k 个人在一些城市，他们要到达同一个目标城市，如果多个人一起行走，那么只花费一个人的边权。问最小代价，如果有多种方案，输出需要访问到的节点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。

9.1.2 算法

这是一个斯坦纳树的模型，但是由于需要判断集合元素等繁杂的信息，我们可以直接枚举一些点将其删除，然后做最小生成树。

如果使用边做 Prim 边搜索，那么复杂度能够去掉一个 n。

9.1.3 时空复杂度

时间复杂度：$O(2^n \cdot n)$。

空间复杂度：$O(n^2)$。

9.2 Problem E Lots of Sunlight

9.2.1 题目大意

给出 $n(n < 100)$ 栋楼，$q(q \leq 1000)$ 个询问，询问某一间公寓的日照时间段。假设 5:37 am 日出，6:17 pm 日落，且太阳运行的角速度不变。

9.2.2 算法

开始时间与结束时间肯定是公寓最下面的点被前面的公寓的顶遮住的时刻。

所以暴力枚举，取极值即可。

注意在之前的楼房比当前公寓的地板还低的时候，如果用 `atan2()` 函数会返回一个负值，取极值时会出现错误，致使我答案错误。

9.2.3 时空复杂度

时间复杂度：$O(nq)$。
空间复杂度：$O(n)$。

9.3 Problem G Tiling the Plane

9.3.1 题目大意

对于一个多边形，如果可以通过它本身复制多次来不重不漏地覆盖一个无限的二维平面，我们就称这个多边形能铺满平面。

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。

现在给出一个 $n(n \leq 50)$ 条边的直角多边形，问它是否能铺满平面。
保证多边形的周长 $l \leq 50$。

9.3.2 算法

首先，题目描述中说边界的端点不一定是多边形的顶点，但是我们可以很快发现，至少有一个点会是多边形的顶点，那么我们可以枚举这个点，然后将整个多边形割成长度相等的两段。假设第一段是 $AB + CD + EF$，那么将第二段反过来(整体以及方向取反)就是 $EF + CD + AB$。

那么，我们可以枚举 A 在第二段中的位置，C 在第二段中的位置，然后检验就可以了。这样复杂度是 $O(nl^3)$。

如果题目没有 $l \leq 50$ 的条件，也是可做的。我们只需要明确一点：假设 A 之后第一个转折点是 G，我们发现，并不需要枚举 A 的具体位置，只需要枚举 G 在第二段中对应的转折点的编号，对应 C 也同理。但是如果 AB 之间没有转

折点，那么我们可以证明一定有一段会全部取完，如果不是，我们可以通过选取其他的分割点，或者将整个串反过来再做一遍来得到可行的分割方法。

因此如果暴力检验，复杂度应该是 $O(n^4)$。如果不怕麻烦，预处理一些子串的匹配以达到 $O(1)$检验的话，复杂度能做到 $O(n^3)$。

9.3.3 时空复杂度

时间复杂度：$O(n^3)$。
空间复杂度：$O(n^2)$。

9.4 Problem I Workshops

9.4.1 题目大意

给出 $n(n \leq 1000)$ 个会议的人数和持续时间，现在有 $m(\leq 1000)$ 个房间分给这些会议，房间的容纳人数和使用时间也已知。问不能在房间里开会的最少人数。

9.4.2 算法

贪心。

把房间按时间的大小排序，从小到大分配给会议，并且满足得到该房间的会议人数尽量大即可。

9.4.3 时空复杂度

时间复杂度：$O(nm)$。
空间复杂度：$O(n+m)$。

9.5 Problem J Zones

9.5.1 题目大意

给出一张有 $n(\leq 20)$ 个圆 $n+m(m \leq 10)$ 个区域的韦恩图，问取其中 k 个圆，能包含的元素个数最大为多少，输出方案。

9.5.2 算法

$\binom{n}{k}$ 枚举即可。

9.5.3 时空复杂度

时间复杂度：$O\left((n+m)\binom{n}{k}\right)$。
空间复杂度：$O(n+m)$。

Chapter 10

ACM/ICPC World Finals 2004

10.1 Problem G Navigation

10.1.1 题目大意

给出 $n(\leq 10)$ 个圆，问它们的交，允许 0.1 的误差。

10.1.2 算法

理论上应该求 n 个宽度为 0.2 的圆环的交，然后判断是否可行的解之间距离小于 0.1。

但事实上，只需要先找两个圆求交，判断求出的交点是否可行就能通过所有点了。

10.1.3 时空复杂度

时间复杂度：$O(n)$。

空间复杂度：$O(n)$。

10.2 Problem H Tree-Lined Streets

10.2.1 题目大意

二维平面上有 $n(n \leq 100)$ 条线段，在每条线段上取若干特殊点，要求同一条线段上的特殊点之间距离不小于 50，且到这条线段和其他线段的交点距离不小于 25。

求特殊点最多个数。数据满足不会有三条线段交于同一点。

10.2.2 算法

暴力几何。

我们求出每条线段与其他线段的交点，并将其排序，就可以将每条线段割开。然后根据小线段的长度就可以计算出其最多能放的特殊点个数。由于每个小线段之间互不相关，所以答案就是它们的和。

10.2.3 时空复杂度

时间复杂度：$O(n^2 \log n)$。

空间复杂度：$O(n)$。

10.3 Problem I Suspense!

10.3.1 题目大意

分别住在两幢楼的 $n(\leq 25)$ 层和 $m(\leq 25)$ 层的两人从其窗户下端挂出绳子，绳子悬在空中组成了抛物线，他们还在绳子最低点下方 $1m$ 处搭了纸质桥面组成吊桥。但他们要保证两幢楼内的猫无法通过桥面吃到其他楼内的鸟。已知所有猫只能向上跳 $< 0.5m$ 或向下跳 $< 3m$，每间房间高度均为 $3m$，窗户下端离房间地面 $1m$。要求桥面至少比底面高 $1m$，至少比他们的窗户下端低 $2m$。给出他们楼下所有住户内的宠物情况，以及两幢楼之间的距离 d，求能悬挂的最长的绳子长度。

10.3.2 算法

首先找出可行的最低桥面高度 H，设绳子的抛物线为 $y = Ax^2 + Bx + C$，将 $(0, h1), (d, h2)$ 代入：

$$\begin{aligned} C &= h1 \\ d^2A + d \cdot B + C &= h2 \\ 4C \cdot A - B^2 &= 4H \cdot A \end{aligned}$$

得到 $A = \frac{B^2}{4C-4H}$，带入第二个式子得到 B 的二次式，就能求得抛物线的方程了。

紧接着，我们需要做一个路径积分：

$$\begin{aligned} L &= \int_0^d \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}\,\mathrm{d}x = \int_0^d \sqrt{1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2}\,\mathrm{d}x = \int_0^d \sqrt{1 + f'(x)^2}\,\mathrm{d}x \\ &= \int_0^d \sqrt{1 + (2Ax + B)^2}\,\mathrm{d}x \end{aligned}$$

令 $t = 2Ax + B$，则 $\mathrm{d}t = 2A\mathrm{d}x$，得到:

$$L = \int_B^{2Ad+B} \frac{1}{2A}\sqrt{1 + t^2}\,\mathrm{d}t \tag{10.1}$$

求不定积分得到原函数:

$$F(x) = \frac{1}{4A}\left(x\sqrt{x^2+1} + \ln\left|x + \sqrt{x^2+1}\right|\right) \tag{10.2}$$

答案即为 $F(2Ad + B) - F(B)$。

10.3.3 时空复杂度

时间复杂度：$O(n+m)$。
空间复杂度：$O(n+m)$。

10.4 Problem J Air Traffic Control

10.4.1 题目大意

给定平面上 $n(\leq 100)$ 个点，这 n 个点按照 y 坐标大的优先级高，y 坐标相同时 x 坐标大的优先级高的顺序排序。

一个控制范围是指的一个圆心 P 和其所能控制的点的个数 val，控制范围控制的点集是按照到 P 的距离从小到大，距离相同按照以上优先级选择 val 个点。控制范围的半径是 P 到控制的点的最远距离。控制范围的边界是以 P 为圆心，r 为半径的圆。

给定 $m(\leq 10)$ 个控制范围所能控制的点数和控制范围边界上的两个点。请确定其所控制的点集。若有多个点集满足题意，依照优先级考虑所有点，所某点属于某控制范围而不属于另一个，那么包含该点的控制范围较优。

请对于所有 i，计算出被 i 个控制范围控制的点的个数。

10.4.2 算法

首先，将所有点按照优先级排序。

对于每个控制范围，我们枚举其最后控制的点，从而得到其控制的点的点集，并在所有点集中找一个字典序最小的即可。

10.4.3 时空复杂度

时间复杂度：$O(n^2m)$。
空间复杂度：$O(n+m)$。

Chapter 11

ACM/ICPC World Finals 2003

11.1 Problem B Light Bulbs

11.1.1 题目大意

一行有 n 个灯，n 个开关，每个开关控制所有与其位置的差≤ 1 的灯。求所需按的开关数量最少的一种方案使灯从初始状态变为目标状态，在此前提下使开关状态压缩为 10 进制后最小。

灯的初始和结束状态都用 10 进制压缩后的形式给出，保证最高位至少有一个是 1，且都不超过 100 个数字。

11.1.2 算法

只需枚举 1 号开关是否转换即可。如果 1 号开关已知，那么 $[2, n]$ 号开关的状态就确定了：因为 1 和 2 号开关决定 1 号灯的状态，从而得出 2 号开关，同理，通过 2 号灯得出 3 号开关，最后通过 $n-1$ 号灯得出 n 号开关，当然，我们需要最后检验第 n 号灯是否满足。

最后，我们取一个最优方案即可。

11.1.3 时空复杂度

时间复杂度：$O(n^2)$。

空间复杂度：$O(n)$。

11.2 Problem G A Linking Loader

11.2.1 题目大意

一组数据包含若干模块，每个模块以 `Z` 结束。支持以下操作：

1. `D symbol offset` 定义 *symbol* 变量的值为当前内存指针 $+$*offset*，初始内存指针在 $(0100)_{16}$ 位置。如果该变量已被定义，则该次定义无效，且标记 *symbol* 为重复定义。

2. `E symbol` 声明变量 *symbol*（每个模块声明的变量分别从 0 开始编号，可能会用到之后定义的变量，但如果该变量一直未定义则值视为 0）

3. `C n byte`$_1$ `byte`$_2$ `...` `byte`$_n$ 给内存指针开始的 n 个内存赋值，并移动内存指针，其中会引用到该模块中声明的变量的编号，变量的值占用两个内存。

然后输出 16 位校验和 sum，计算方法如下：首先 $sum = 0$，然后按顺序遍历所有地址，每次 sum 循环左移一个二进制位（即最高位变为最低位，其他位左移），然后加上该内存地址所存储的值，并保留最后 16 位。

然后按变量名的字典序输出变量名和变量的值（ 16 进制），若该变量未被定义，输出 `????`，若该变量被重复定义则多输出一个 `M`。

11.2.2 算法

直接模拟即可，可以用 map 或者 hash 来记录变量。

11.2.3 时空复杂度

时间复杂度：$O(M)$。M 为内存大小，本题中是 16^4。

空间复杂度：$O(M)$。

11.3 Problem H A Spy in the Metro

11.3.1 题目大意

$n(\leq 50)$ 个站台的地铁中，有 $m1(\leq 50)$ 趟正向列车， $m2(\leq 50)$ 趟反向列车，问从 1 号站台出发且 $T(\leq 200)$ 时刻到达 n 号站台所需要的最少的在站台上的等待时间（只能在站台换车，可以同一时间从正向列车切换到反向列车，反之亦可。）

11.3.2 算法

$f(i, j)$ 表示第 i 个时刻在 j 站台最少需要等待的时间。

转移有三种：在 j 站台等到下一个时刻；如果有向左去的火车，那就可以向左乘一站；如果有向右去的火车，那就可以向右乘一站。

11.3.3 时空复杂度

时间复杂度：$O(n(m1 + m2 + T))$。

空间复杂度：$O(nT)$。

11.4 Problem I The Solar System

11.4.1 题目大意

给出第一个行星的轨道的半长轴，半短轴和周期，以及第二个行星的半长轴，半短轴，求第二个行星在第 t 天的坐标。

11.4.2 算法

先用开普勒第三定律求得第二个行星的周期，然后二分角度，求面积。

设椭圆方程为 $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，其上半部分方程为 $y=\frac{b}{a}\sqrt{a^2-x^2}$，与半径为 a 的圆的上半部分方程 $y=\sqrt{a^2-x^2}$ 相比，积分时求得的原方程之比为 $\frac{b}{a}$。这样，我们求面积时只需要求出对应的圆上的面积就能求出椭圆上的面积。

当然，我们也可以直接对 $\frac{b}{a}\sqrt{a^2-x^2}$ 积分：令 $\alpha=\cos^{-1}\frac{x}{a}$，即 $x=a\cos\alpha$，则 $\sqrt{a^2-x^2}=a\sin\alpha, \mathrm{d}x=-a\sin\alpha\,\mathrm{d}\alpha$，解得：

$$\begin{aligned}\int\frac{b}{a}\sqrt{a^2-x^2}\,\mathrm{d}x &= \frac{b}{a}\int -a^2\sin^2\alpha\,\mathrm{d}\alpha\\ &= \frac{ab}{2}\int(\cos 2\alpha-1)\,\mathrm{d}\alpha\\ &= \frac{ab}{2}(\frac{1}{2}\sin 2\alpha-\alpha)+C\end{aligned}$$

当然我们也可以求出公式后直接用牛顿迭代法解方程。

11.4.3 时空复杂度

时间复杂度：$O(\log P)$。P 是精度要求。

空间复杂度：$O(1)$。

Chapter 12

ACM/ICPC World Finals 2002

12.1 Problem A Balloons in a Box

12.1.1 题目大意

已知一个长方体的盒子和一个大小为 $n(\leq 6)$ 的点集。每一个点代表一个可以放置气球的位置。在一个点上放置一个气球，就是以这个点为球心，然后让这个球膨胀，直到触及盒子的边缘或者一个之前已经被放置好的气球。你的目标是按照某种顺序在盒子里放置气球，使得气球占据的总体积最大。

12.1.2 算法

直接枚举放气球的顺序，求半径、统计体积即可。

12.1.3 时空复杂度

时间复杂度：$O(n! \cdot n)$。
空间复杂度：$O(n)$。

12.2 Problem C Crossing the Desert

12.2.1 题目大意

给出二维平面上 $n(\leq 20)$ 个点，你需要从某个点走到另一个点。你每走一英里，你就需要消耗一单位食物和一单位水。所以在起始点上你可以在商店里购买食物而且你可以收集到无限的免费的水，在每个绿洲你同样可以收集无限的水和储存食物，但是你没法再绿洲购买新的食物，并且你能携带的单位数量上限为 tot。你需要确定到达终点所需要购买的最少食物量。由于商店只能整单位售卖食物且只有一百万单位的存货，你的答案必须是大于0小于等于一百万的整数。

12.2.2 算法

首先，我们可以明确一点，我们一定是找到一条路径，把食物一个点一个点地搬运过去。

设 $f(x)$ 表示从 x 这个点到达终点所需的最少食物量。转移时，设 x 之后一个点是 y，如果 $tot <= 3dist$，那么只能走一次，无法回来，所以要满足 $f(y) >= tot - 2dist$ 才能转移；如果 $tot > 3dist$，那么就可以来回走，总距离为 $dist\left(2\left\lceil\frac{f(y)-(V-2dist)}{V-3dist}\right\rceil+1\right)$。

这样我们就可以用最短路算法来解决整个问题了。

12.2.3 时空复杂度

时间复杂度：$O(n^2)$。
空间复杂度：$O(n)$。

Chapter 13

ACM/ICPC World Finals 2001

13.1 Problem B Say Cheese

13.1.1 题目大意

给出三维空间上两个点 s 和 t，需要从 s 走到 t，正常情况下速度为 0.1，但是有 $n(\leq 100)$ 个小孔，给出小孔中心坐标 (x_i, y_i, z_i) 和半径 r_i，在这些小孔中速度为 ∞。问最短时间。

13.1.2 算法

首先，可以把 s 和 t 看作半径为 0 的两个小孔。

然后定义小孔之间的距离为 $\max\{0, dist(x, y) - r_x - r_y\}$，做最短路即可。

13.1.3 时空复杂度

时间复杂度：$O(n^2)$。

空间复杂度：$O(n)$。

13.2 Problem F A Major Problem

13.2.1 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。

你需要写一个程序，在不同音阶中交换音符。

13.2.2 算法

找出所有可能的大调，模拟即可。

13.2.3 时空复杂度

时间复杂度：$O(1)$。

空间复杂度：$O(1)$。

13.3 Problem I A Vexing Problem

13.3.1 题目大意

给出一个竖直放置的 $n \times m(n, m \leq 9)$ 的棋盘，一些格子是障碍，不会动，一些格子是石块，如果左边（右边）不是石块或障碍，那么就可以向左（向右）移动，如果一个石块悬空，那么它就会掉下来。紧接着，如果某个石块上下左右有与它颜色相同的石块，那么它就会被消掉。消完了以后可能还有石块要掉落，之后相同的再消掉. . . . . 问最少需要多少步才能把所有石块消光。答案保证不大于 11，求棋盘最底下一行和左右两列保证都是石块。

13.3.2 算法

此题用 BFS 可以在 LA 上通过，但是由于出题人的数据非常强，在清澄上无法通过。考虑使用 DFS。

首先，我对于一个局面，优先考虑移动在下面的石块，因为下面的石块移动可能迫使更多的石块的移动。

然后，对于回溯这一步，我记录了石块移动、消失、下坠的信息，每次只需要逆操作即可。

接着，我对棋盘进行 Hash，如果发现上次走到这个局面时走的步数不比现在的大，那么剪枝。

当然，如果某种颜色只剩一颗石块，那么剪枝；如果只剩两颗或三颗石块，那么剩下的步数至少为它们的距离减一。

13.3.3 时空复杂度

时间复杂度：$O((2nm)^{11})$。
空间复杂度：$O((2nm)^{11})$。

Chapter 14

ACM/ICPC World Finals 2000

14.1 Problem A Abbott's Revenge

14.1.1 题目大意

你需要在一个 $n \times m (n, m \leq 9)$ 的箭头迷宫中从起点出发到达终点。在每个路口处，如果你从某个方向进入了该路口，那么路口的地面上在靠近你的方向会画有一组箭头，它们相对于你的方向可以是向左，向前，向右，或者是它们的任意组合，表示你能走的方向。输出最短的方案。

14.1.2 算法

用 $f(x, y, d)$ 表示从起点走到 (x, y) 这个点，并且方向为 d 的最短距离。
按照题意 BFS 即可。

14.1.3 时空复杂度

时间复杂度：$O(nm)$。
空间复杂度：$O(nm)$。

14.2 Problem E Internet Bandwidth

14.2.1 题目大意

给出一张 n 个点 m 条边的无向流网络，求 s 到 t 的最大流。

14.2.2 算法

直接把无向边当成两条有向边就行了，因为这两条边不可能同时有流。

14.2.3 时空复杂度

时间复杂度：$O(\text{MaxFlow}(n, m))$。
空间复杂度：$O(m)$。

Chapter 15

ACM/ICPC World Finals 1999

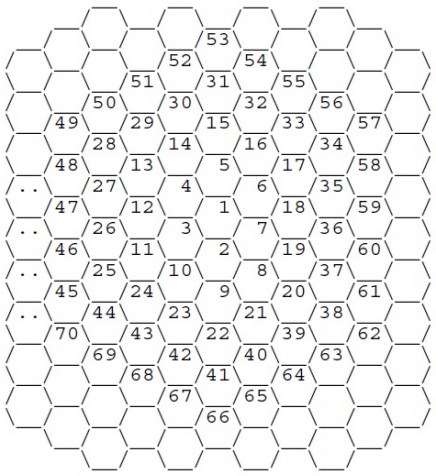

Figure 15.1: 蜂窝

15.1 Problem A Bee Breeding

15.1.1 题目大意

给一些蜂窝编号，如图 15.1 ，问标号为 $a, b(\leq 10000)$ 的蜂窝之间的距离。

15.1.2 算法

首先，设斜上方为 x 轴正方向，正上方 y 轴正方向。

$O(\max\{a, b\})$ 预处理出坐标，然后直接用公式计算出它们之间的距离。即：

设 $\Delta x = x_a - x_b, \Delta y = y_a - y_b$，那么答案为 $\min\{|\Delta x| + |\Delta y|, |\Delta x| + |\Delta y + \Delta x|, |\Delta x + \Delta y| + |\Delta y|\}$。

15.1.3 时空复杂度

时间复杂度：$O(\max\{a, b\})$。

空间复杂度：$O(\max\{a, b\})$。

15.2 Problem C A Dicey Problem

15.2.1 题目大意

给出一个 $n \times m (n, m \leq 10)$ 的骰子地图。一个标准的六面骰子需要在地图上移动。每张地图有一个规定的初始位置和一个初始的骰子放置方式。

你可以通过沿着骰子的一条边转动来将它移动到地图上水平或是竖直方向上相邻的格子上。假设当前骰子顶面的数字为 x，那么你只能将它移动到写着 x 的格子上，或者画着星星图案的格子上。我们的最终目标是找到一条路径，使得骰子能从起点出发，最后又回到起点。

求一条最短距离的方案。

15.2.2 算法

$f(i, j, d)$ 表示从起点到 (i, j) 这个点并且顶面是 d 的最短距离。

按照题意 BFS 即可。

15.2.3 时空复杂度

时间复杂度：$O(nm)$。

空间复杂度：$O(nm)$。

15.3 Problem E Trade on Verweggistan

15.3.1 题目大意

给出 $n (\leq 50)$ 个数列，每个数列元素不超过 20 个，如果你要取某个数列中第 i 个，那么必须要取前 $i-1$ 个。你需要最大化你取的数字和减去 10×数字个数，如果有多种方案，升序输出需要取的数字个数，多余 10 个的部分不用输。

15.3.2 算法

首先把每个元素都减去 10，然后对每个数列做前缀和，然后用背包求出可能的取的数字个数。

15.3.3 时空复杂度

时间复杂度：$O(n^2m^2)$。m 为数列元素个数。

空间复杂度：$O(n^2m)$。

Chapter 16

ACM/ICPC World Finals 1998

16.1 Problem E Petri Net Simulation

16.1.1 题目大意

有 $n(\leq 100)$ 个节点，$m(\leq 100)$ 种变换，初始第 i 个节点有 c_i 个令牌。一个变换涉及若干个输入点和输出点，一次变换后所有输入点令牌数 -1（令牌数不能为负），输出点令牌数 $+1$。求 $T(\leq 1000)$ 次变换（或在 T 次变换之前就不能变换了）后的所有点的令牌数。变换可以按任意顺序发生，但保证答案唯一。

16.1.2 算法

由于数据范围较小，并且保证答案唯一，所以直接模拟即可。

16.1.3 时空复杂度

时间复杂度：$O(Tnm)$。
空间复杂度：$O(nm)$。