ACM World Final部分题目解题报告

南京外国语学校　王悦同

Contents

1 【World Finals 1998 B Flight Planning】

一架飞机要飞连续N（⩽100）段路程。每段路程都需要指定一个飞行高度，这个高度必须在20000到40000之间，并且是1000的整数倍。此外，飞行的自身速度、飞行单位时间的燃料成本、每一段的风速是给定的，每一段风速是一个线性函数。飞机实际速度就是自身速度减去风速（也有可能是顺风，那就是加上）。飞机在相邻两段间可以改变高度，每改变一单位高度需要花费一定代价。你需要决定每一段的飞行高度，最小化总代价（改变高度+飞行时花费燃料）。

算法分析：

这题很明显的具有阶段性。不难发现，有效的状态就是：飞机当前在哪一段；飞机当前高度。用f[i][j]表示飞机在第i段高度为j（j=20到40，j*1000就是实际高度）此时最小代价。枚举下一段飞行的高度，则改变飞行高度的代价易知道；根据风速可以算出飞行时间，也就容易算出燃料成本。这样就可以很方便的完成状态转移。

2 【World Finals 1998 C Lead or Gold】

给出N（⩽100）种物品，每种物品都是A、B、C三种元素按一定比例的混合物。现在我们希望配制出一直新的物品，新的物品中A、B、C三种元素含量满足一定比例。你可以用N种物品中每一个任意多份（可以不必是整数份）。现在你需要判断是否可能配制出那种新的物品呢？

算法分析：

首先，由于每个物品用了任意多份，所以三种元素的具体比例数值是没有意义的，只要知道他们分别占了整个物品的百分之多少。例如4:5:9是没意义的，只要知道2/9,5/18,1/2这三个“所占比例”就行了。同时，三个比例和是1，所以只要知道两个其实就足够了。

现在，N个物品，还有最后我们要的那个物品，都可以用一个二元组表示出来了。我们观察那N个二元组能配出哪些物品呢？由于是二元组，所以不难想到把它们对应到平面上的点。这个时候观察这N个点，我们有一种直观的感觉－－如果某个点对应的物品多用了，就能把最后答案“往那个方向拉”，同时每个物品都有能力往它的方向拉任意的距离这就相当于一些矢量求和。不难发现－－无论如何，都是不可能“拉”出N个点的凸包的！同时对应观察也不难发现，凸包中的点总是能被“拉”到！这个说明并不严谨，但很直观，并且当我发现这样的性质时，我可以确信他就是对的，至少在OI比赛中这样的感觉是很必要的。严格的证明我没有想出，不过这种做法的直观和优美仍然很有欣赏价值。这个算法自然也就很快的通过了本题，复杂度为凸包的复杂度O(NlogN)。

3 【World Finals 1998 D Page Selection by Keyword Matching】

给出Q个操作（Q具体大小题目没说，但非常小，据其他数据范围估计在50以内），每个操作是下面两种形式之一：

1.插入一个网页，一个网页可能有最多8个关键字

2.给出若干关键字，询问和这些关键字最匹配的前5个网页

网页匹配定义如下：一组关键字和一个网页的匹配程度，等同于$\sum$ (9-i)*(9-j)（输入的第i个关键字=待匹配网页的第j个关键字）。最终，匹配程度前5大的网页应该被输出。

算法分析：

题目数据范围非常小，容易看出就是一个简单的模拟。直接记录当前所有的网页和他们对应的关键字。每次询问时，暴力计算所有网页的关键字后排序输出即可。

4 【World Finals 1998 E Petri Net Simulation】

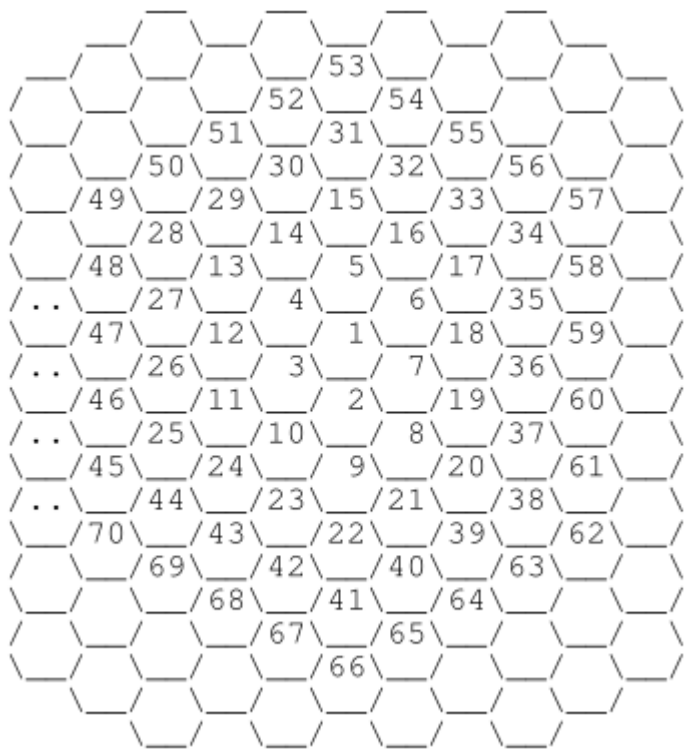
给出一个N（$\leqslant$100）个点M（$\leqslant$1000）的有向图，每个点写着一个数。每次对于一个节点的操作定义为：它的所有前驱都减1，然后所有后继都加1（如果有重边，一个前驱可能被减多次）。一个节点可以被操作，当且仅当它的前驱减完后没有负数（必须是减完后没有负数，而不是减完再加完没有负数）。现在你需要对其操作K（$\leqslant$1000）次，每次如果有多个点可以被操作，随意选择一个。问K次后每个点写的数是多少，或者指出第多少次后就无法继续操作了。保证每次任意选择点操作答案都一样。

算法分析：

和上题一样，这也是简单的模拟。每次暴力判断一个点能否被操作，这个可以通过枚举所有点得到。时间复杂度为O(K*M)。

5 【World Finals 1999 A Bee Breeding】

N（⩽10000）个数按下列方式排列：



请求出某两个数之间的距离。距离定义为：图中相邻的格子对应的数距离为1，然后求最短路。

算法分析：

容易看出这题就是一个最短路问题，关键在于如何构建这个图。

直接往里面填数将面临很多细节。为此我采用了一种牺牲时间而易于实现的做法：设1位于(0,0)，我直接计算出所有点的坐标。易知，在第N层，从开始的那个数（比如第三层的8），先向左下走N-2格，然后向左上走N-1格，然后向上N-1格，然后向右上N-1格，然后向右下N-1格，然后向下N格（进入下一层）。我们不难计算出向六个方向走的矢量，然后O(N)求出所有点位置。接下来连边——两个点如果距离是1，则连边，否则不连。这样时间复杂度是O(N^2)的，不过根据平面图的性质，连出来的边数是O(N)的，所以不用担心存不下。这样以后就是直接求最短路了。时间复杂度在于建图，为O(N^2)，可以通过。

6 【World Finals 1999 D The Fortified Forest】

给定N（⩽15）棵树，每棵树有一个坐标，还有一个高度。你需要砍掉其中一些树，用这些木料作为栅栏，围住剩下的树。高度为X米的树可以做成长度为X米的栅栏。现在需要你决

定砍掉哪些树，才能用这些树作为木材围住剩下的树，并最小化砍掉树的总高度。给出一种方案。

算法分析：

由于容易注意到N很小，因此可以枚举每棵树是否被砍掉。这一步是O(2^N)的，此时可以得到每棵树是否被砍掉的状态。对于那些没被砍掉的树，只需要用尽可能少的总长度将它们围起来－－这是一个经典的问题，这个总长度应当等于剩余点的凸包的周长。因此，通过O(NlogN)的时间可以求出这个凸包，并在O(1)时间内进行判断。总复杂度为O(2^N*NlogN)。

7 【World Finals 1999 E Trade on Verweggistan】

现在有N（⩽50）堆物品，每堆物品有A_i（⩽20）个。所有的物品都有一个价值，这个价值可正可负。你需要从中选若干个物品－－但是，如果你选了这个物品，你必须也选所有那一堆中在它上方的物品。现在你需要从中选一些物品，使得总价值最大。如果有多种方法使得总价值最大，记分别选出了F_1,F_2,F_3...个物品，你需要去重后输出前10 小的F_i。

算法分析：

本题看上去确实十分类似一个背包，不过需要稍加变化。容易发现，对于第i堆物品，由于只能选出其中的顶部若干个，则只有A_i+1种方案，每种方案也对应了一个价值。这很类似背包，区别在于：你不能同时选出某一堆的前i个和同一堆的前j个（i≠j）。这个问题即对应了分组背包。对于集训队同学来说，分组背包应该是一个十分基础的问题，这里简单给出有关代码：

```
for (int i=1;i<=N;i++) /*N组物品*/
    for (int j=totsize;j>=0;j--)
        for (int k=1;k<=number[i];k++)
            if (j>=size[i][k]) dp[j]=max(dp[j],dp[j-size[i][k]]+value[i][k]);
```

本题最终复杂度就等于背包复杂度O(N*A_i*Size)。

8 【World Finals 1999 G Flooded!】

给出一个N*M（均⩽30）的网格，每个位置有一个数$A_{i,j}$表示该位置的海拔。现在出现了积水，所有积水的格子海拔都一样、而海拔高于这个水平面的格子则显然没有积水（不考虑连通性等问题）。现知道所有位置积水高度之和（未积水格子显然记为0而非负数），希望你求出所有积水格子的那个水面高度。这个高度可以是小数。

算法分析：

本题有一个比较显而易见的做法：二分答案，然后计算一下所有格子的积水高度之和。这个做法正确性也很明显，可是这题数据似乎过于强以至于带来一些精度和范围问题（比如二分答案范围好像10^{22}都不够）。当时我这种做法只得了5分……于是只好开发新的做法。

我们将所有高度排序（这里二维没有任何意义，不妨直接转为一维考虑），设排好序后数组为F。首先，我们枚举答案落在F的哪个区间，比如$F_i < ans < F_{i+1}$。此时我们去解方程求这个答案：积水高度之和应该等于k*ans-$\sum F_k$(k⩽i)。解完方程后判断答案是否落在区间即可。通过维护前缀和，时间复杂度容易做到O(M*N)。

9 【World Finals 2000 B According to Bartjens】

本题是我分到的第一阶段作业题，有关翻译和详细题解在第一阶段作业里有。

10 【World Finals 2000 E Internet Bandwidth】

给出一个N（⩽100）个点的无向图，要求计算两个点间的最大流值。

算法分析：

模板题，直接使用任何一种最大流算法即可。

11 【World Finals 2000 F Page Hopping】

给出一个N（⩽100）个点的强连通有向图，求出所有点对间最短路的平均值。

算法分析：

模板题，直接使用任何一种最短路算法即可。比较tricky的地方是这题节点没有保证从1开始编号，所以需要对节点的编号进行一次离散化。其它都很简单；由于N很小，直接Floyd都是可行的。

12 【World Finals 2001 A Airport Configuration】

有一个2*N（N⩽25）个长条的格子。其中若干对格子间有一些人流。两个格子的距离等

同于它们间的曼哈顿距离。现在请你计算∑每对格子间人流量*该对格子距离。

算法分析：

很明显这就是一道模拟题。加上数据范围极小，任何暴力的统计方法都是可行的。

13 【World Finals 2001 H Professor Monotonic's Network】

给出一个比较网络，其中包括M（⩽150）个比较器和N（⩽12）个输入输出装置。一个比较网络是排序网络当且仅当：对于任意输入的N个数，依次经过M个比较器（每个比较器由两个输入X，Y组成，并且将A_x和A_y中较小的放到A_x而较大的放到A_y），输出总是一个有序的序列。现在给出一个比较网络，你需要判断它是否是排序网络；如果是的话，你还需要输出，如果比较器可以并行工作（并行工作的两个比较器不能同时处理某一个元素），那么最少需要多少单位时间才能完成所有比较器的比较。一个单位时间，可以有若干个比较器同时工作；一个比较器仅当它的所有输入值已经被算出来后才能开始工作。

算法分析：

这题的第二问其实是非常容易的。把比较器视作节点，则如果比较器B需要用到比较器A的数据（即B在A后面且B和A共同处理某一位置的数据），则A向B连有向边。最后求这个（拓扑）图的最长路即可。这也是“关键路径”的体现。

第一问比较复杂一点。首先很容易想到，虽然N个数的输入有无数种情况，但本质不同的只有N!种（即将输入离散化）。可惜的是，每次判断是O(M)的，所以O(M*N!)的总复杂度无法接受。这个时候我们需要一个重要结论：只要假设输入只可能是0或1，如果比较网络能正确处理这2^N种情况，则它就是排序网络。这个结论能直接把复杂度优化到O(M*2^N)，就可以过了。这个结论我不会证明，可以参加算法导论。

不过还有一个虽然慢一点但是更加易于理解的做法，这个做法由xhr提出。我们可以分别判断每个位置最后是否能到达目标位置。具体做法是这个位置放1，其余位置枚举放置0或2。所有的0都比1小，它们之间的大小关系任意；所有2也是。这样，如果最后排出的序列是唯一的1夹在所有0和2之间，则说明对于这个位置，能正确识别出哪些位置比它大哪些比它小。如果所有位置都能被正确识别，则这个网络显然就是符合要求的排序网络了。这个做法的正确性就明显多了，时间复杂度是O(M*N*2^{N-1})，也通过了本题。

14 【World Finals 2002 A Ballons in a Box】

你要模拟在长方体的盒子里放置球形的气球，你已知一个长方体的盒子和一个点集。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。你不能使用一个在盒子外面或者在一个之前已经放置好的气球里面的点。但是，你可以按你喜欢的任意顺序使用这些点，而且你不需要每个点都用。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。你要做的是计算盒子里没被气球占据的体积。点集大小不超过6。

算法分析：

唯一需要决策的就是放置的顺序了；由于最多6个气球，所以不妨枚举这个顺序。剩下来要做的事情就是模拟这些气球的膨胀——这个气球i的半径等于min(它到6个面中最短的那个距离，dist($point_i$,$point_k$)-radius(k))其中k<i。最后考虑几个特殊情况：点在盒子外面时不能膨胀；点已经被别的气球包含时不能膨胀；计算出的膨胀半径不能是负数等。

15 【World Finals 2002 C Crossing the Desert】

平面上有N（⩽20）个点，你现在在1号点，打算走到N号点。在任意两点间移动，若这两点欧几里得距离为k，则你需要花费k单位的水和k单位的食物（如果食物和水之一不够就不能这么走）。一开始你可以决定携带一些食物和一些水；一旦出发以后，食物就没有补给了。水则是每到一个点就可以立即补满。任何时候，你身上携带的食物和水的总和不能超过S。不过，你可以在某些点临时存储一些食物，过一段时间再来拿。请你计算为了从1走到N，出发时至少需要带多少单位的食物？

算法分析：

首先注意到水是不值钱的，如果不是容量上限，我们应该带无限多的水。但同时我们也注意到：如果我们能带x单位的食物，则我们如果某一时刻缺水了，就应该之前少带一些食物而改为带水的；但如果是缺食物了就不能这样对应解决。所以我们可以认为：任何时候，我们都应该只考虑带食物，剩下来的空间都是水；什么时候水不够了，食物也能当水用，反之则不行。

接下来我们考虑一个有些诡异的情况：我们是不是到每个点最多一次？乍一看显然，但其实有一个有趣的情况：考虑现在我们只能随身带100单位食物，不过1号点和2号点非常近。这时候，我们带1000单位食物出发，并把900单位临时存在1，100单位带到2；然后带一点回来（仅供回来路上使用）；然后再带100单位去2，再回来……直观上可以看出，如果两个点很近，我们可以往返于两个点之间运输物资，而巧妙躲开了携带食物上限的要求。

不过应该注意到，除了这样“往返运输”外，走一个长度超过2的环后回到出发点是没有意义的。因为每次运输必然只能是两个点间的，所以决定了运输方向后应该直线走过去。这样，我们可以设计一个函数，f(i,j,k)表示从i向j运输，一开始在i有k个单位食物，运到j时剩下

多少。

有了这些分析后问题就好办了。首先二分答案，然后dp[i]表示当前在i号点最多剩下多少物资。枚举所有的点j，用f(i,j,dp[i])去更新dp[j]，以此不断宽搜，最后就可以判断出是否可行了。时间复杂度取决于如何实现f函数，并且宽搜复杂度也是难以估算的，不过N=20显然还是太小了，毫无压力。

16 【World Finals 2002 D Island Hopping】

给出一个N（⩽50）个点的无向图，你需要求出它的最小生成树。接着你需要计算，每个点到1号点的路径上最大值的平均值是多少。

算法分析：

这题也基本上和模拟题无异。有一点可能会有疑问，如果有多棵最小生成树呢？其实根据最小生成树的性质稍加分析就会发现，这种情况下答案是一样的，所以就根据题目的意思求出任意一棵就行了。后面的路径到根最大值就直接从根开始dfs一遍就好了，这个复杂度就是球最小生成树的复杂度O(N+MlogN)。

17 【World Finals 2002 F Silly Sort】

给出N（⩽1000）个互不相同的数。每次，你可以交换任意两个数，代价是这两个数的和。问你最少需要多少代价，才能把整个序列排好序？

算法分析：

这题属于一类很经典的置换环问题。根据每个数最后的位置，可以找出若干个置换环。对于每个置换环，有两种选择：找环内最小的数和所有数换一圈；或者找全局最小的数和环内所有数换一圈。对于每个环独立考虑，从两种策略中选择较优的即可。整个算法复杂度为O(N^2)。

18 【World Finals 2003 B Light Bulbs】

有N（⩽400）个灯泡，排成一排，每个灯泡一开始有一个状态。每次你按下某个灯泡对应的开关，它和它相邻的两个灯泡（一共三个；在边界就是两个）状态取反。问最少需要按多少次，才能把N个灯泡改为目标状态。并给出一种方案。

算法分析：

显然每个灯泡对应的开关最多按一次，并且按开关的顺序没有影响－－这是这类“异或”问题的必然性质。由于每个灯泡控制的范围很小，所以我们可以注意到，如果我们从左向右决定每个灯泡是否按，那么如果当前待决策i，但是1至i-2中有灯泡没有达到目标状态，则一定不合法了；如果i-1没达到，则i必须按一下。有了这一点，我们可以枚举一号开关按不按。此时，决策2号开关的时候，根据1号灯是否达到了目标状态，2号开关选择唯一。以此类推，3号、4号、后面所有开关选择都唯一。全部决定后，判断N号灯是否满足要求（因为没有N+1号开关“补救”，所以如果N号灯不合法就是不可行方案）。这题比较麻烦的地方在于输入输出用10进制数，转成二进制后代表灯泡状态。其实只是多了一个高精度计算而已，代码变得复杂了但是算法上没有什么难度。

19 【World Finals 2003 D Eurodiffusion】

在10*10的网格上有c（⩽20）个国家，每个国家都是一个Hi*Ci的矩形，表示这个国家有Hi*Ci个城市。国家之间的矩形没有重叠，并且所有涉及到的城市是四连通的（不属于任何一个国家的网格不算城市）。一开始，每个城市都只有一百万个本国的硬币，各个国家货币都不一样。然后每一天，每个城市都按照它在这一天开始时的余额，将一定量的硬币送给它的所有邻接城市。这“一定量的硬币”指的是对于这个城市目前所拥有的每种图形的硬币，每满1000个就要拿出来一个。当某个城市中，每种图形的硬币都至少出现了一个，就称这个城市“已经完成”。当一个国家的所有城市都已经完成的时候，就称这个国家已经完成。你的程序需要得出每个国家的完成时间。

算法分析：

这题很明显是一个模拟题，即模拟所有硬币的扩散过程。不过这个算法会超时么？我们不妨进行一些分析。每个国家的硬币种类不同，在扩散的时候可以彼此独立的处理。对于一个国家的货币，一开始该国每个城市有1000000。由于相邻的两个城市是相互扩散的，所以他们之间存在一个“平衡状态”，即两城市该硬币数量基本接近。达到这个状态需要最多500天，而该平衡一旦达到后就不会被打破。所以只要500*(10*9+9*10)天就可以使所有城市对平衡。上面是一个粗略的估计，例如平衡的城市也可能因为相邻的城市数不同而每天相差最多4000枚硬币，但因为刚刚已经说明天数不会很多，所以其实这样的差距代入对天数的影响只是常数差异。有了这个分析后，直接对于每种硬币模拟扩散过程后输出即可。

20 【World Finals 2003 F Combining Images】

给出两棵四分树的编码。每棵四分树对应一幅图像。图像对应四分树编码的方法如下（图像边长必须是2的幂）：

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。

这两幅图像的大小一样。求出它们的交。交定义为：交的每个位置是1当且仅当两个原图这个位置都是1，否则就是0。输出交的四分树编码。

算法分析：

这题明显是一个模拟。构建出图像是不现实的，因为太大了。但是不难发现四分树大小和输入的编码长度一个数量级，所以我们建出四分树。这个过程并不复杂：访问一个点，如果是0，那么递归处理四个子树；如果是1，就记下当前节点颜色，然后返回父节点。两棵树得到后求交：如果一个点，两棵树中有一个这个点是0（全0，即没有孩子），则交的这里全是0；如果全1，那么另一棵树的子树完整的拷贝过去；否则递归合并两棵树的这个节点。求出交后，有一个需要注意的：如果一个点本来有孩子，现在4个孩子全都是全0了，那么这个点就应该全0，并且删掉那些孩子。这样处理后树就合法了直接输出，复杂度和输入的长度线性相关，肯定不会超时。

21 【World Finals 2003 H A Spy in the Metro】

地铁有N（$\leqslant$50）站，从1到N连续编号。列车双向运行：从首站到末站和从末站到首站。列车在相邻站台间所需的运行时间是固定的，因为所有列车的运行速度是相同的。列车停站的时间非常短，为了简化问题可以忽略。可以在同一个站里，同时停车的列车间换乘。现在你0时刻在1号站，你希望恰好在T（$\leqslant$200）时刻到达N号站（注意不能早到，早到必须在站台上等到T；更不能迟到）。你需要最小化在站台上等候的时间。两个方向列车班数均不超过50。

算法分析：

注意到最后见面的时刻T并不大，我们可以考虑将其计入状态，则问题被大大化简了。f[i][j]表示T时刻在站台j的最少等待时间，则f[i][j]可以加1后转移给f[i+1][j]，或者如果有一班车经过，f[i][j]也可以直接通过枚举乘这班车在哪一站下车，转移给f[i'][j']其中i'是下车的时刻，j'是下车的位置。这样就直接进行转移即可，可以按时间轴扫描，转移则是O(C*N)的

其中C是列车班数、然后枚举下车站。故总复杂度是O(N^2*T*C)的，实际上远达不到该复杂度。

22 【World Finals 2003 J Toll】

给出一个N（⩽52）个点的有向图，你要从1号点向N号点运至少K个物品。在图中的某些节点，你如果进过它就必须缴纳一定税款－－每20个物品交1个。现在求出你至少需要带多少物品出发，才能保证最优路径下，到N号点时有至少K个物品。

算法分析：

这题还是比较好想的。由于比较明显的是：出发时带的物品越多，到达时物品至少不会变少。我们二分答案：设出发时带了C个物品。则用F_i表示到达第i个结点时，最多剩余多少物品。一开始显然有F_1=C，然后使用一个类似宽搜的过程：每次枚举队列首的点，去试图刷新它所有后继的F值，并把被刷新的点放入队列里。这个过程很近似SPFA——最后没有新的点被更新时就完成了。算法的复杂度不是很显然，但是通过一些粗略的分析，肯定不到O(N^3)。实际上这个算法跑的非常快。

23 【World Finals 2004 H Tree-Lined Streets】

给出N（⩽100）条直线。现在你需要在直线上的一些位置点若干个点，这些点满足如下条件：

同一条直线上的两个点，距离不小于50。

每一个点距离每个“线段交点”距离不小于25。

你需要求出最多能点多少个点。

算法分析：

两个条件中，25和50正好成两倍关系，使得这个看上去复杂的条件有了特殊的性质。我们考虑这么处理——给每个线段两端都加长25，接着把每条线段按所有和别的线段的交点划分成若干段。这个时候，不难发现——每有一段长度为50的线段，我们就一定可以选择其中点；反之就一定不可以。这个转化非常重要，使得本来一个决策性问题变成了一个以模拟为主的计算几何题。计算线段的交点无疑是计算几何中最容易的部分了，直接用叉积计算即可，再根据刚刚所说的，每一段长度为k的子线段（被交点划分后的线段）都贡献了k/50 个点。因此直接计算几何统计即可。

24 【World Finals 2005 C The Traveling Judges Problem】

给出一个N（⩽20）个点的无向图，请求出连接给定K（⩽10）个点的最小生成树，并且输出这棵树的具体形态。

算法分析：

这题和最小生成树的本质区别是——不需要连接所有点，只需要连接指定点，可以为了最小化代价而连接一些额外的点。这个问题被称为“最小斯坦纳生成树”，有一个O(N*2^{2*K})的状压dp做法。具体可以参加POJ3123，不过我不知道如何通过这样的状压dp输出方案，所以这题没有采纳。

实际上，这题不光K小，N也非常小。可以不妨直接枚举最终我们连接了哪些点，然后求一下最小生成树即可——两点边权就是他们的最短路，用Floyd预存。诚然这样得到的生成树不一定符合要求，因为可能floyd带入了额外的一些点，不过我们只要说明：最优方案合法，就行了（这是一种十分常见的手段）。而实际上最优方案必然合法，因为Floyd带入的点总有一次会被我们枚举到。

所以直接枚举求解即可，输出方案也十分容易。O(2^N*N*N)。不过注意到，这N个点的枚举中，至少选出给定的K个点，而额外的点选出个数必然不超过K-1，所以实际上复杂度远达不到这个数量级。

25 【World Finals 2005 H The Great Wall Game】

一个n*n（⩽15）的网格和n颗石子。这些石子随机地放在网格的方格之中，一个格子中最多放一颗石子。每一次移动，可以将任意一颗石子移动到上下左右相邻的空方格之中。游戏的目标是用最少的移动步数，使得n颗石子构成“一堵墙”——排成一条水平、竖直或斜的直线。给出一个初始的棋盘状态，求出这个最少需要的步数。

算法分析：

不难看出，最后的状态只有n*2+2种——n条竖直线，n条水平线，或者两条对角线。如果最后是竖直线的话，则每个棋子显然还在原来那一行，所以枚举最后是哪一列就可以了；最后是水平线的情况同理。只有对角线的情况比较麻烦，但不难发现由于N很小，所以直接状压dp求出这个最优的匹配就行了；费用流求最优匹配也同样可行。复杂度是O(N*2^N) 或者O(maxflow(N,N*N))，都能通过本题。

26 【World Finals 2005 I Workshops】

现在有N（⩽1000）个会议，还有M（⩽1000）个房间。每个会议有些人数。某些会议可以在某些房间里进行（M*N个关系），不过最后每个房间里最多容纳一个会议，每个会议也只能最多分配到一个房间里。你需要让尽可能多的会议能在房间里进行，在此基础上你希望尽可能多的人能参加房间里的会议。

算法分析：

这题很明显是一个二分图匹配的模型，如果没有那个“在此基础上你希望尽可能多的人能参加房间里的会议”的要求，那么就是完完全全的最大匹配模板题。加上这个条件以后，就需要进行修改了。观察匈牙利算法求二分图匹配的过程：每次找到一条交错轨，然后在此基础上增广：虚边比实边多一条，所以将实边和虚边反过来，就能完成增广。这个过程有一个重要的性质：一个左边的节点一旦在某次被匹配上了，它就再也不会失配！因此，我们显然只要让人多的会议先增广就行了。将会议按照人数重新排序，此时问题仍然是一个匈牙利算法，在求出最大匹配数的同时，也选出了尽可能多的人。时间复杂度O(N*Q)，Q是二分图的边数（也就是“会议可以在房间里进行”的关系对数）。匈牙利算法复杂度很松，虽然理论上是三方，但实际上很快。

另一个做法是费用流，直接建立二分图，并且在源到左边的点（左边会议右边房间）的边加上费用——人数。求最大费用最大流即可。注意到N=1000对于费用流来说还是有点大了；我没有尝试，不知道是否会超时。不过它就显然没有之前利用匈牙利的性质做优美了。

27 【World Finals 2005 J Zones】

给你N（⩽20）个可以选择的信号塔，你需要从中选择K（⩽N）个修建。每个信号塔如果修建了，就可以覆盖一些服务区。现在给出每个服务区的信息：它被哪些信号塔覆盖，该区有多少人。对于每个服务区，所有覆盖它的信号塔中如果有至少一个修建了，那么所有区内人都可以享受信号。现在求出最多能有多少人享受信号？服务区数目不超过30。

算法分析：

这题N过于小，以至于完全没有思维难度。从N个中选择K的方案是C(N,K)，不超过2^N。所以可能的方案数十分有限，一一枚举出来判断即可。

28 【World Finals 2006 A Low Cost Air Travel】

有N（⩽20）种机票，每个机票有一定花费，并且会依次访问Ai个（不超过20个）城市。

买了一张机票后，你可以选择使用它的任意一个前缀。你需要花尽可能少的钱买一些机票，依次访问K(⩽20）个城市。中途可以访问一些额外的城市。

算法分析：

这题最tricky的地方就在于最后一句话。由于这句话的存在，所以我们不光要记录我们访问了需要的K个中的前多少个，还有记录当前在哪个城市。用f[i][j]表示当前访问了K个城市中的前i个，目前在城市j，的最小花费。每次枚举N种机票中的一种，再枚举在哪个城市下车，就可以完成转移。不过，由于一张机票使用不意味着i要至少加1，可能只是转移一个城市为以后准备，所以这些状态之间有环。简单的处理方法就是，把状态都表示为节点，机票都表示为Ai条不同的边，然后直接求单源最短路即可。点数V=N*N*Ai，边数E=N*N*Ai*Ai（这里涉及到均摊，因为一张机票必须在起始点开始使用）。时间复杂度为O((V+E)log(V+E))，不会超时。

29 【World Finals 2006 B Remember the A La Mode!】

有N（⩽50）种冰淇淋和M（⩽50）种糕点，每种搭配的时候有一个美味程度F_{ij}，也有可能不能搭配。此外数量也有限制，第i种冰淇淋有A_i个，第i种糕点有B_i 个。你需要用完所有的冰淇淋和糕点，组合出若干个“套餐”，现在求出这些套餐的美味程度之和最大是多少？最小是多少？数据保证能恰好用完所有冰淇淋和糕点，也保证A_i和B_i都不超过100。

算法分析：

其实这题的模型还是十分明显的。冰淇淋放在左边，糕点放在右边，则就是一个二分图模型了。源点到左边冰淇淋连边，费用为0，容量为冰淇淋数量；右边糕点往汇连边，容量为糕点数量，费用为0。中间边容量无穷，费用为搭配的美味程度（不能搭配就不连边）。这个时候，题目就变成了——最小费用最大流和最大费用最大流。于是这就和模板题无异了。一般情况下，这种“两种物品”的搭配和决策往往可以和二分图网络流有关系，所以这题不难往这方面想然后直接解决。注意由于数据组数多，spfa费用流会超时；对于二分图来说zkw费用流明显要快得多，可以通过本题。

30 【World Finals 2006 D BipartiteNumbers】

二段数是这样的正整数:恰好包含两种不同的十进制数字s和t，s不是0，并且s的所有出现均排列在所有的t的前面。例如23333333333。现在给出一个数N（⩽100000），请求出最小的二段数，使得它是N的倍数。200组数据。

算法分析：

看到这题不难想到，从小到大枚举二段数，但这样显然会超时，原因是枚举二段数的时候要枚举四个参数——两段分别长度；两段分别是哪个数字。后两个参数很小（10*10）大概可以忽略，但即便这样也有两个O(N)的参数要枚举。

一般情况下对于两个参数的枚举，我们应该考虑只枚举一个而通过Hash等方式查找另一个从而优化复杂度。对于这题我们做一些观察，如果我们定义对应的“一段数”（所有数码都一样且非0），那么任意一个二段数都可以表示为两个长度不同的一段数的和或差。例如我们考虑，现在假设我们枚举其中一个一段数是6666666666，那么我们就要找所有长度小于10的一段数，要么数码是1,2,3且加上去是N的倍数，要么数码是1,2,3,4,5,6并且减去后是N的倍数。所以，问题就是查找，长度小于某个值的某个数码构成的一段数中，有没有mod N是某个指定数的数；如果有，我们需要存最大（减去的时候用）和最小（加上的时候用）。所以，我们先预存N*9个数在Hash表里，就可以快速查找了。有个技巧：因为每次查找的数长度必须小于当前枚举的长度，所以我们应该每次先枚举长度K，求出此时是否有答案，如果没有，再把长度为K的一段数加入Hash表供后面查询。这样总时间复杂度就是O(N)的，可以通过。

31 【World Finals 2006 H Pocket】

本题是我分到的第一阶段作业题，有关翻译和详细题解在第一阶段作业里有。

32 【World Finals 2006 I Degrees of Separation】

给出一个N（$\leqslant$50）个点的无向图，求出所有点对间最短路的最大值，或者指出这个图不连通。

算法分析：

模板题，直接使用任何一种最短路算法即可。Floyd等算法均可，并且这些最短路算法同时也是可以求出连通性的O(N^3)。

33 【World Finals 2007 A Consanguine Calculations】

给出父母和孩子三个人中两个人的血型（A,B,O）和RH阴阳性，判断第三个人的所有可

能血型和RH阴阳性的组合。符合生物遗传规律。

算法分析：

首先这题因为数据范围很小，我们不妨预存出所有的三人表现型组合。

从学生物知识的时候就应该有这样的处理经验：一个基因型恰好对应一个表现型，但反过来不一定。所以，虽然我们要求表现型，但我们应该枚举父母的基因型，这样才容易模拟遗传的过程。很容易得到孩子的所有基因型；这里数据范围太小，随便怎么模拟。最后，算出三个人对应的表现型，并且去重。总复杂度不估计了，因为是预存的，本地存好没超时就肯定可以AC了。

34 【World Finals 2007 E Network】

有N（⩽5）条信息被拆分成了M（⩽1000）个数据包。每个数据包是属于一条信息，记录了这条信息的某一部分（起始字节、终止字节）。这些数据包依次进入缓冲区，在任意时刻，你可以取出缓冲区里的任何一个数据包输出到终端。最终，你需要保证，N条信息的所有数据包都是被连续的接受的，且字节顺序正确。为此你的缓冲区至少需要多少字节呢？

算法分析：

容易发现这题N非常小。对于每条消息，数据包的接受顺序是一定的，所以我们需要决定N条消息的接受顺序。这并不难——N!的枚举是可行的。那么现在我们就发现，M个数据包的要求的接受顺序就知道了。因此，逐一读入M个数据包，放入缓存区；下一个需要的数据包如果不在缓存区里，就继续读取后面的数据包，如果在就取出来输出。模拟一遍即可知道这个缓存区任意时刻的字节数，也就知道至少需要多大了。时间复杂度O(N!*M)，不会超时。

35 【World Finals 2007 H Tunnels】

给出一个N（⩽50）个点T（⩽1000）条边的无向图。有个人希望从1号点走到2号点，而你需要阻止他这么做。你可以在任意时候瞬间毁灭一条路。注意，这意味着你可以等他做出决策，跑到某个点的时候，突然毁掉一条路，而不是你毁完后对方移动。你需要计算出最坏情况下最少需要毁掉多少条路？

算法分析：

这题我虽然在清澄上通过了，不过所用的做法正确性是猜测的，不太清楚证明。欢迎对

证明有研究的同学和我交流……

我的做法是，先删掉一些路，使得那个人只能到达某些点；然后看他到了哪个点，此时立即求最小割把它和2割开。具体实现就是，最后能到达的点肯定是最小割值最小的连续一段。我们枚举最后能到哪些点，那么我们需要用最小的代价把1和其它点割开，这个最小割值、加上能到达的点到2的最小割值的最大值，更新答案即可。算法复杂度为N次网络流复杂度。

36 【World Finals 2008 B Always an Integer】

给出一个关于x的N（⩽100）次多项式，和一个正整数D（longint以内）。判断是否对于任意整数x，这个多项式的值都是D的倍数？

算法分析：

这题一个显而易见的做法是：不断地代入数，计算多项式是否是D的倍数（也就是关于D的余数）。实际上这个做法是可以通过本题的，如果从1开始在时间允许的范围内逐个枚举数x，一旦发现不是D的倍数就跳出，否则就认为是的。这个做法的理论依据来源于一个结论：如果当x代入从1至N的所有整数时，一个N次多项式都是D的倍数，那么x代入任意整数时仍然成立。关于这个定理的证明可以参见Per Austrin的题解，我也是先用枚举通过了本题，才从他的题解中学习如何证明这个结论的。

其实这题更大的难点在于输入的处理。我们需要处理一个多项式，也就是处理若干种特殊情况。包括：首位正负号；次数为1的项；系数为1或-1时的省略；某一项系数为0而不写；常数项（或者没有）。一旦处理好多项式以后，就是一个枚举和模拟求值的过程了。时间复杂度为O(N^2)。

37 【World Finals 2008 E Huffman Codes】

现在有N（⩽20）个数，现在要把它们按如下规则建成哈夫曼树：每次选择两个最小的数，合并起来，小数放左大数放右，如果一样就任意；其它规则和建立哈夫曼树一样。现在已知N个数和为100，并且给出建立的最终的哈夫曼树形态（形态仅仅是：每个点有没有左孩子、有没有右孩子）。问最早的N个数有多少种分布，使得建出的哈夫曼树形态和给出的一致？排序后相同的N个数算一种方案。

算法分析：

这题比较无聊，我一开始觉得是一道好题想了好久没有想法，看了一下题解是直接搜

索。一开始根肯定写着100；每次，枚举任何一个节点，把它拆成两个数，分别给左儿子和右儿子，然后继续递归搜索。树都填满时就是一种合法的解。直接这么搜即可。

38 【World Finals 2008 F Glenbow Museum】

现在你有一个多边形，多边形每条边都水平或垂直于坐标轴。你按照某个点开始的顺时针顺序记录下其每个角的大小：90度的角记为R，270度的记为O。不过你没有记录每条边的长度。此外你还知道，在这个多边形中存在一个点，能看到这个多边形的其它任意位置。现在已知你记录序列的长度L（⩽1000），问你有多少个不同的序列，使得存在一个满足要求的对应的多边形？

算法分析：

首先，我们明确一点：一个长度为L的序列对应了一个L条边的多边形。这一点很容易说明。因此，我们也可以得到这个多边形的内角和——由此简单推理可以得到，序列中的R一定比O多4个，不然从内角和看来就不合要求了。另外，连续两个O一定不能有，否则就肯定是“凹”进去一块儿，不能被同时看到。如果没有两个O呢？由于边长可以调整，所以这样的多边形一定存在——因为基本就是R和O交错，画出来是若干个“阶梯型”的边缘，而整个图的中心肯定是可以看到所有位置的。所以直接动态规划就可以了：f[i][j][k]表示前i个位置，填了j个R，上一个是R还是O（k=0/1），每次枚举下一个是哪个就行了。时间复杂度O(N^2)。

39 【World Finals 2008 G Net Loss】

给出一个N（⩽10）次多项式和一个实数c（介于-1和1之间）。你需要用两个交于x=c的一次函数来拟合这个N次多项式。评估函数f记为两个函数在-1至1区间上的差函数的平方的定积分。你需要恰当的选择这两个一次函数，使得这个评估函数的值尽可能小。这个值最小是多少呢？

算法分析：

这是一道完全考察数学能力的题目。由于这两个函数在x=c时y值也相等，不妨设这个y值为t。现在关于两个一次函数的唯一未知数就是他们分别的斜率，我们不妨设斜率为k （两个函数应该独立处理；前者要最小化(-1,c)区间的定积分值而后者要最小化(c,1)的定积分值）。此时把评估函数展开并平方然后求定积分，可以发现：由于定积分是求这个函数的不定积分在c处和在-1处的差值（后半段类似），所以多项式的大多数项都是可以直接计算的，含有

未知数的项一开始是一次，平方再积分后是三次（其它高次项实际上代入定积分后都是常数）。我们要求这个三次函数的极值——直接求导后解一个二次方程，并把解代入原函数，可以得到t和最后答案的关系。所以t和最后答案关系仍然是一个多项式函数的极值的形式；继续求导然后代入即可。时间复杂度为O(N)。

40 【World Finals 2008 I Password Suspects】

给出N（⩽10）个字符串，每个长度不超过10，全部由小写字母组成。你需要构造一个长度为M（⩽25）的小写字母字符串，使得N个串都是其子串（可以重叠），问你有多少种不同的方案？不需要高精度。如果方案数不超过42，则输出全部方案。

算法分析：

首先考虑方案数不超过42，此时肯定不能有某一位是随意的（不然就26种，加上可以平移一下在另一个位置随意填，至少52种），所以一定是原来N个串某种排列。枚举全部N!种情况即可。接下来的统计中，我们将不考虑输出方案的事情。

考虑把这N个串建成AC自动机，由于串的个数不多，可以根据每个串是否出现过进行状态压缩。f[i][j][k]表示当前字符串长度为i，在AC自动机上的节点j，并且N个字符串出现情况是k（一个bitmask）。在传统动态规划中，我们需要依次记录每个串出现到哪一位了，但这样显然十分低效；AC自动机则是精简了这一点使得状态的存储十分高效。每次对于一个f[i][j][k]，我们枚举下一个字符是哪一个，设j通过这个字符走到了j'，那么f[i][j][k]就应该转移给f[i+1][j'][k']其中k'表示如果j'是某个串的结尾那么需要在k的基础上修改一下。需要注意的是，AC自动机中，一个点是某个串结尾，和trie里的定义不一样——如果一个点，或者一个点沿着失败指针走到的某个点是串结尾，则这个点也是。bitmask里被重复计算是没有关系的，这样做可以确保不遗漏。其它和正常的动态规划一样，枚举下一个字符，完成转移。算法复杂度为O(N*len*M*2^N*char)，char是字符集大小，也就是26。

41 【World Finals 2008 J The Sky is the Limit】

给出N（⩽100）个等腰三角形：它们底边都重合于x轴。请你求出它们的并的轮廓线长度，注意不计算x轴部分。

算法分析：

由于N非常小，所以可以使用比较暴力的方法解析几何。考虑到轮廓线就是每段y坐标最高的，而“关键点”数量很有限——要么是输入的等腰三角形底边的端点，要么是不同等腰

三角形的边的交点。计算出关键点后，对于每一段，可以再暴力扫描每个等腰三角形，找出谁在这一段的y坐标最大，这样一段段统计就可以了。一共有O(N^2)段，而N=100，故整个算法O(N^3)是足以通过的。

42 【World Finals 2008 K Steam Roller】

给出一个R*C的网格（R,C⩽100），每条边上有个权（是网格的边，也就是R*(C-1)+(R-1)*C条），你需要从(0,0)点走到(R,C)点。在每条边的时候，如果上一条边是转弯过来的，或者下一条边要转弯，那么这条边需要减速行驶——花费双倍时间，否则通过时间就等于长度。问你最少的需要时间是多少？有些边是不能通过的，输入会给出。

算法分析：

这题并不难，主要就是各种细节。大体思路就是f[i][j][flag][k]表示现在在(i,j)，刚刚来的方向是k（1,2,3,4之一，表示上下左右），上一段行驶是全速还是慢速（flag）。每次枚举下一步的方向和速度，就不难计算出需要的时间（如果当前是全速，那么下一步方向就不能变，否则可以变）。这也可以看做把每个状态作为节点，就是求单源最短路了。时间复杂度是O(R*C*log(R*C))。一个细节：一开始我k只记录了两个方向，以为“掉头加速”这种情况肯定不优。实际上不然。如果前一段是1，后一段100，我们在100的路上必须花双倍时间；如果我们在1加速，然后立即掉头，在1那段用了三倍时间，而100则是全速——这是非法的行为，因为掉头也是转弯，所以必须存储4个方向判断。

43 【World Finals 2009 A A Careful Approach】

给出N（⩽8）个区间[L_i,R_i]，你需要为每个区间选择一个K_i($L_i \leqslant K_i \leqslant R_i$)，使得所有$K_i$的两两差的最小值尽可能大。

算法分析：

首先二分答案。容易发现，如果每个区间的K_i是有序的，我们就可以很容易判断了。但是N只有8，这明显告诉我们搜索的可行性——搜索这N个区间中K_i的顺序，每次显然应该K_i较大的应该在区间范围和二分的答案要求内尽可能小。这样时间复杂度就是O(N!*N*log(ans))。这题数据多测，时间略紧，需要搜索时加最优性剪枝而不能直接全排列。

44 【World Finals 2009 E Fare and Balanced】

给出一个N（⩽50000）个点M（⩽50000）条边的有向无环图，现在你需要增加某些边的权值，使得从1到N的所有路径长度都一样。在此基础上，你需要保证——对于任何一条从1到N的路径，都不能有超过一条边被增加了权值。输出增加的总权值，或者指出这办不到。

算法分析：

我们先假设输入永远合法，即存在这样一个方案。那么我们考虑求出1–>N的最长路，并据此构出最长路图。那么，对于那些不在最长路图上的边(u,v)，它的权值应该被修改为maxdis(1,N)-maxdis(1,u)-maxdis(v,N)。如果某条边不修改，则方案一定不合法；但也易知，如果这些边都被修改成对应的权值，那么合法的方案就得到了，同时“增加的总权值”也就得到了。我们发现——答案已经得到了！所以其实，本题主要在判断是否存在这样的方案，即判断无解情况。我们考虑对于一个点u，如果某个点u使得1–>u的最长路和u–>N的最长路都可能经过修改的边，则肯定无解了，因为某条过u的路径一定包含两条被修改的边。但如果这样的点不存在呢？这个时候我们发现，如果我们经过了一条修改了权值的边(u,v)，则1–>u最长路肯定没经过修改的边、v–>N的最长路也肯定没经过修改的边，不然和刚刚假设矛盾。所以这样的点不存在就一定合法。综上所述，我们只要构建最长路图，并对于每个点求出1到它和它到N分别是否可能经过修改的边，即可解决本题。时间复杂度O(M+N)。

45 【World Finals 2009 F Deer-Proof Fence】

给出N（⩽9）个点。你需要修建一些栅栏围住它们。要求：栅栏不必连通，但每个点都必须被围住；且每个点距离栅栏任意位置距离必须不小于R。求出栅栏的最短总长度。

算法分析：

我们考虑，如果栅栏必须连通呢？那么这就是一个经典问题，求出凸包，然后凸包向外面扩一圈（这里只要求周长，那就是凸包长度+半径R的圆周长）。这题不同的是栅栏不必连通。不过这个问题并不难办，因为N非常小，所以我们不妨枚举所有的组合情况，取最优值就行了。

这些凸包可能会相交，这个怎么判断呢？其实不用在意。因为相交意味着不优，那么我们必然会在枚举到某次相交的两个凸包并在一起时，得到更优的答案。所以最后得到的最优解一定是合法的。算法复杂度约为O(N!*NlogN)，不过实际上达不到。

46 【World Finals 2009 H The Ministers' Major Mess】

有M（⩽500）个大臣对N（⩽100）个议案进行投票表决。每个大臣只对其中K（⩽4）个议案投票。每个投票只能是同意或者反对。现在，你需要决定这N 个议案是否通过，使得对于每个大臣，他超过一半的投票都和结果一致。同时判断：是否可能满足这个条件；满足这个条件的情况下，哪些议案必须通过，哪些必须不通过，哪些随意？

算法分析：

这题比较诡异的条件就是那个K⩽4了。由于每个大臣必须有超过一半的投票和结果一致，所以至少要三个和结果一致。换句话说，一旦某个和结果不一致了，其它都必须和结果一致！

这可以让我们联想到2-SAT问题。对于每个议案，拆成两个点，通过和不通过；如果一个点不通过导致了某些议案必须通过，就可以连上对应的边，反之亦然。对于那些投票数小于等于2 的人，他们的所有议案必须被采纳，所以，比如要求议案2不通过，那么议案2通过那个点连向不通过那个点，代表这个点不能被选择。

接下来就是求解2-SAT了。对于第一问，只要判断是否有某个点，既不能通过（通过会导致不通过），也不能不通过（不通过会导致通过），即这两个在同一个强连通分量里。对于接下来那一问，只要对于每个议案，分别要求它通过、不通过，判断剩下的2-SAT是否有解，即可对应得到答案了。

47 【World Finals 2009 J Subway Timing】

给出一棵N（⩽100）个节点的树，每条边边权不超过300。因此，每对顶点之间都有一个距离。现在，你需要把每条边的权值改成60的倍数（并且只能是原来数最接近的两个60的倍数之一）。新的树中，每对顶点之间有一个新的距离，和原来的距离有一个误差。我们希望这个误差的最大值尽可能小。你需要决策每条边改成哪个60的倍数，最小化这个误差的最大值。

算法分析：

这题由于是一棵树，所以不难想到树上有关的做法。比较明显的是需要二分答案——因为我们是要“最大值最小”。另一个也不难想到的是树形dp。不过我单纯的顺着这个思路并没有啥想法。后来题解给出了一个重要结论：最后的误差一定不会超过118！（就是(60-1)*2）这个结论为啥如此重要呢？我据此考虑，决定将误差这个值计入状态中。

二分答案的部分仍然不变。然后对于二分后的答案K，我们进行动态规划：f[i][j]表示当前在点i，子树内部全部符合要求，从子树到当前点i的所有误差中，区间是[j,f[i][j]]——这

个f[i][j]最小是多少。这个j是-118到118之间的，不是很大。对于一个点i，一开始，f[i][0]=0。以后依次考虑每一棵子树cs，然后更新f[i]数组。

f[i][min(j,k)]=max(f[s][j],f[cs][k])其中j+k>0且f[s][j]+f[cs][k]<K。这样自底向上更新上来就可以得到最后的答案了。整个算法复杂度为O(log(Q)*Q*Q*N)，其中Q=118*2，为误差的可能范围。

48 【World Finals 2009 K Suffix-Replacement Grammars】

给出两个等长字符串S和T，长度均不超过20。再给出N（⩽100）个替换规则，每个替换规则由两个等长的字符串S1,T1组成，表示如果当前字符串的后缀包含S1，那么可以把这段后缀替换成T1。问你把S替换成T最少需要几步，或者指出这不可能。

算法分析：

这题还是比较难想的。和一般字符串替换问题不同，这里有两个限制：替换只能是后缀，替换前后长度不变。我们考虑，如果把字符串S替换成了T，假设S和T长度都是K，那么肯定替换的每一步长度都是K。每一步替换，无非两种情况：直接用某个长度为K的替换做一下；或者将其后缀做一些替换，而前面字符保持不变。

另外我们也发现，有意义的状态并不多。如果我们目前存在一个字符串X，它不是S、T、Si、Ti的任何一个后缀，那么它肯定是不可能在一次合法的变换中，作为中间过程的后缀出现。所以，我们只需要考虑那些是输入的某个字符串的某个后缀的字符串，对于这些字符串，我们全部存储下来（显然不多，N*20个）。从小到大来计算替换的最少步数。

接下来所说的字符串，都是我们存储好的有意义的字符串。我们首先考虑长度为1的字符串之间替换：这十分显然，构成图论模型的话，能变换的字符串间连边，长度为1，Floyd一下即可。然后考虑长度为K（>1）的字符串。这时候，我们已经求出了K-1的所有字符串间的变化最少步数。继续构图，每个字符串作为一个节点，两个字符串如果能直接变换，连边长度为1；如果首位相同，且去掉首位以后可以在T步之内变换（这个刚刚已经求出了），就连边长度为T。这时再进行Floyd，如此执行若干次，直到S和T的最短路被计算出来。整个算法复杂度为O(N^3*Len)，还有些额外的复杂度（比如查找字符串之类的），但可以用Hash等技术使得其复杂度小而可以忽略。注意这题答案需要long long存储。

49 【World Finals 2010 B Barcodes】

这题是模拟题，题意较为复杂而且不好简述，故直接复制原题：

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

因此，字符1被编码后就会表示成为深色的宽区域，浅色的窄区域，深色的窄区域，浅色的窄区域，深色的宽区域这五个区域。并且，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一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。

Code-11这种编码方式可以简化我们的编码和解码过程，因为他仅需要区分窄区域和宽区域，并且提高了对低打印水平的容忍度。

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c1、c2.....cn，那么检验字符C的值就应该是$sigma(((n-i) \bmod 10+1)*w(c[i])) \bmod 11(1<=i<=n)$。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。（注意mod的运算优先级比+高）

与此对应的检验字符K的值是$sigma(((n-i+1) \bmod 9+1)*w(c[i])) \bmod 11(1<=i<=n+1)$，其中c[n+1]即为检验字符C。举个例子，假设需要被编码的字符串是123-45，那么我们有C=5和K=2，所以最终需要被编码的字符串是123-4552，并且有一个开始标志和结束标志分别在头部和尾部。

一般的简单的条形码读取器会包含数百个小型的CCD传感器，这些CCD传感器会识别出深色和浅色区域以及每一块的宽度。利用这些信息，解码器就会将解码得到被编码之前的信息。由于条形码的方向不是固定的，所以无论读取器是正着扫描的条形码还是倒着扫描的，解码器都需要能够解码。

你的任务是解码一个被Code-11编码过的条形码。你能够知道每一块区域的宽度，并且宽的区域的宽度是窄的区域的宽度的两倍。但是由于打印设备的不完美，所以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。

算法分析：

这题最有算法含量的地方就是那个识别5%了。我采用的办法是：设基准为x，则求出x和2x能容纳的范围，判断是不是每一个数都落在这个范围内——这也就是相当于关于x的若干一次不等式，判断有没有解就行了。注意枚举x是很可能错的，因为题目未保证基准是整数。

接下来就完全是模拟了。正着和反着各解码一次。注意无解的情况有很多都要注意，比如如果N除以6必须余5；每段解码都要有意义；校验值要对等等。不过实现起来没有什么技术，只要完全模拟即可。

50 【World Finals 2010 C Tracking Bio-bots】

给出一个R*C（R,C≤1000000）的网格，其中有W（≤1000）堵墙。每堵墙都是连续若干个横着的或竖着的格子（也就是一定是宽为1的长方形）。现在你每次只能从(x,y)走到(x+1,y)或者(x,y+1)，并且不能撞到墙。问你存在多少个格子，使得你从它出发能够走到(R,C)。坐标从1开始标号。墙壁保证不互交。

算法分析：

由于每个点只能走到右边或下边，所以存在明显的序，可以考虑动态规划。f[i][j]表

示(i,j)能否走到(R,C)，则不难写出转移：f[i][j]=f[i+1][j] or f[i][j+1]。边界情况是f[R][C]=1，所有不在矩形内的格子都是0，墙也是0。

观察上面的算法，正确性比较显然，不幸的是复杂度是O(R*C)的，而R和C很大，无法直接用该方法解决本题。注意到墙的个数则是非常少——1000，和R、C比小多了。直观的考虑，由于每堵墙宽度只能是1，所以很容易感受到，整个图绝大多数地方都是空地，而一大片空地如果右下角能到，则整个就都能到了。这启示我们进行离散化——把整片的空地和墙合并到一起，形成一个新的网格，每个网格有个权值（表示它实际上是多少个网格）。容易发现，这个新的网格边长最多是W*2！

至此问题就解决了。对于新的2W*2W网格，刚刚所述的动态规划仍然有效。在这个新的网格上dp并求值就很方便了。时间复杂度为O(W^2)。

51 【World Finals 2010 D Castles】

给出一个N（⩽100）个节点组成的树，每个节点有两个值X_i和Y_i，表示到达这个点的时候至少有X_i个人，并且会死掉Y_i个。你需要决定一个访问所有点的顺序，并且一旦进入一棵子树你就必须访问完才能出来（也就是一条边最多经过两次，一进一出）。在最优情况下，最少需要一开始有多少个人，才能完成遍历呢？

算法分析：

由于题目是一棵树，并且每条边最多走两次，因此很明显的有树形动态规划的暗示：要走完一个点的子树，无非就是决定一个访问它直接孩子的顺序，然后依次对每个孩子递归处理。

接着我们发现，对于一棵子树，如果要遍历它，无论采用什么样的策略，总的死亡人数是固定的。可能随策略改变而改变的是最少需要人数。显然，对于每个子树，我们都希望这个最少需要人数尽可能少。我们记need[i]表示访问i的子树最少需要人数，dead[i]表示访问i的子树死亡人数。显然dead可以直接求出来，现在我们要做的是，根据子树的need[]和dead[]，求出当前点的need[]，最后need[1]就是答案。

因此，这个问题其实不是树上的，而是线性的。问题就是：

给出need[1],need[2]……,need[N],dead[1],dead[2],……,dead[N],求出一个1至N的排列，最小化需要人数。

对于“求出排列”这类问题，似乎很难有动态规划之类的方法，因为动态规划必须基于一个序。我们自然想到了贪心。直观来看，need[]大的要先访问（因为后面可能人死了），dead[]大的要后访问（为了避免人死的太多后面访问不下去），所以猜测：need[i]-dead[i]大的先访问！（事实上根据以往情况，如果一题是贪心的话，往往把几个制约因素根据他们的影响方向加减乘除试一试，很可能是对的）写完这个方法AC了……不过我后来研究

了一下怎么证明。

考虑试图交换相邻两个位置的顺序。不妨设为i和i+1。另外记k表示，访问i+2到最后N至少需要多少人。

此时，最少需要人数为max(need[i],max(need[i+1],dead[i+1]+k)+dead[i])。

也就是max(need[i],need[i+1]+dead[i],dead[i+1]+dead[i]+k)。

如果交换i和i+1呢？最少需要人数为max(need[i+1],max(need[i],dead[i]+k)+dead[i+1])。

也就是max(need[i+1],need[i]+dead[i+1],dead[i]+dead[i+1]+k)。

最后一项都是一样的，而第一项分别小于对方第二项，故比较的关键就在第二项上。如果i和i+1交换，意味着need[i]+dead[i+1]<need[i+1]+dead[i]，也就是need[i]-dead[i]<need[i+1]-dead[i+1]。故看出，need[]-dead[]小的放后面，也就符合了之前的猜测。

52 【World Finals 2010 G The Islands】

给出平面上N（⩽100）个点，其中有两个点特殊（保证不是1和N）。在两个点之间移动的代价是他们俩的欧几里得距离。你现在需要从1出发，经过一些编号递增的点，到达N，然后经过一些编号递减的点，回到1。除了1经过恰好两次，其它点都必须经过恰好1次，同时两个特殊点必须一个在1–>N时经过而另一个在N–>1时经过。求出最少需要的总代价完成这一要求。

算法分析：

问题就是找两条1至N的路，每条编号递增，每个点恰好经过一次（除了1和N）。由于每条编号递增，所以不难想到明显的“序”，我们考虑动态规划。先不管那两个特殊点，我们设计这样一个状态：f[i][j]表示当前访问了前i个点，一条路径最后必须是i，还有一条路径最后是j。

转移并不难。每次枚举下一个i+1点是i走到还是j走到，就可以O(1)实现转移了。具体是：

f[i+1][i]=min(f[i+1][i],f[i][j]+dist(j,i+1)); f[i+1][j]=min(f[i+1][j],f[i][j]+dist(i,i+1));

最后处理一下那两个点，其实只需要两个flag表示两条路径分别是否经过特殊点即可。每当i+1是特殊点的时候，转移时同时考虑flag，状态和转移非常显然，和刚刚基本一样。总复杂度仅为O(N^2)。

53 【World Finals 2010 I Robots on Ice】

给出一个M*N（M,N⩽8）的网格，下标从0开始标号，你需要从(0,0)出发，经过所有格子

恰好一次后回到(0,1)，并且：有三个特殊的点，你需要在整个行程的第m*n/4、m*n/2、m*n/4*3步经过这三个格子。问你有多少种可行路线？

算法分析：

由于N和M很小，很容易想到两种做法：搜索；连通性状压dp。先考虑后者比较正常的想法，但我们会发现：原来连通性dp依赖于插头的合并，然而插头合并方式不同决定了访问顺序不同，则不能满足这题的要求！如果要正确的记录全部需要信息，那么需要难以估计的空间。不如我们来进行搜索。

搜索就是直接做，关键在于效率。几个剪枝：

1.不能在不对的时候到达标记点；

2.如果当前位置距离下一个标记点曼哈顿距离太远以至于不可能赶到，则剪枝；

3.如果当前没走过的位置被分为了不止一个联通块，那么剪枝。这个剪枝是这种网格里搜索的至关重要、但也是非常常见和传统的剪枝。

经过这三个优化本题就可以AC了。不知道有没有同学能用类似状压dp的方法通过，欢迎和我交流。

54 【World Finals 2010 J Sharing Chocolate】

你有一块R*C（R,C⩽100）的巧克力，你要把它分给N（⩽15）个朋友。每次，你拿出巧克力，沿着某一条线把它分成两块（必须沿着整数的线），然后再拿出一块分成两块，以此类推。N个人都有一个想要的大小，你必须恰好把原巧克力分成N块（即分N-1次），然后给N个人，每个人得到的大小必须和想要的严格一致。问你能否做到？

算法分析：

N非常小启示我们可以状态压缩。我们把这个过程改为：N个人，每个人决策一个自己要的巧克力的长和宽，然后我们看能不能每次选两个人的拼起来，最后拼成R*C。另外，如果我们确定了某一块巧克力是哪些人拼起来的，则大小就已知了——我们只需要记录某一条边长，剩下来一条也就知道了。故我们用f[i][j]表示i这个bitmask的人拼起来能否边长为j。一开始，对于所有i只含一个人的情况，如果j是其大小约数则f[i][j]=true,否则f[i][j]=false；对于其它f[i][j]，通过枚举i的一个子集，用两个较小的状态来更新大状态。例如如果i'是i的子集，那么f[i'][j]和f[i-i'][j]可以更新f[i][j]，而计算出f[i][j]对应的另一条边长p后，用f[i'][p]和f[i-i'][p]也可以更新f[i][j]。所以总复杂度就是O(3^N*R)。不过因为约数等各种限制，所以复杂度松的多。

55 【World Finals 2011 A To Add or to Multiply】

给出两个数a和m。你需要写一段代码，代码由字母A和字母M组成。一开始有个输入的数，然后依次扫描这段代码，遇到一个A就把那个数加上a，遇到一个M就把那个数乘上m，最后输出那个数。你这段代码要保证，对于任意[p,q]间的输入，输出必须在[r,s]间。在此基础上选择最短的代码；最短的中选择字典序最小的。输出这段代码。代码可能很长，你输出的时候，输出先多少个A，然后多少个M，然后多少A，以此类推。a,m,p,q,r,s均在$[1,10^9]$之间。

算法分析：

首先，m=1时我们只能不断加法，这个很容易得到；m>1的时候我们不会用很多次乘法（最多29次，不然就超过10^9了）。所以我们首先枚举使用了多少次乘法，设为G 次。于是，现在我们要决策的就是：相邻两次乘法间用了多少次加法。通过观察可以发现：第i次乘法和第i+1次乘法之间，如果加了一次，那么就对答案贡献是a*m^{G-i}。所以，在靠前的地方用一次加法，等于在后面的位置用m次加法，所以我们应该尽可能在靠前的地方用加法。所以我们贪心的在靠前的地方使用加法，直到什么时候超过s了就不能使用，转到后面使用——以此操作知道什么时候大于等于r了结束。不难说明，类似m进制数，这样的贪心长度最小；同时，因为长度固定，而a尽可能的在前面用了，所以字典序也最小。因此复杂度是关于s和m范围的对数级算法，具体复杂度不再分析因为显然这样的做法不会在时间上有任何问题。

56 【World Finals 2011 C Ancient Messages】

你需要写一个程序，识别出一幅黑白图像中的所有字符。六个需要识别的字符如下：

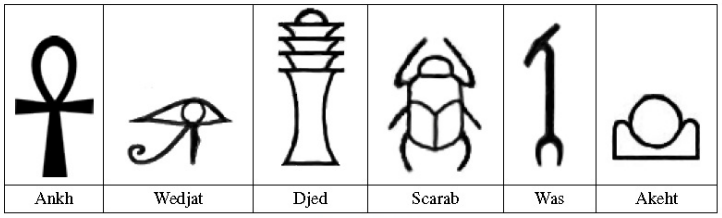


图1：六个象形文字

图像可以被扭曲或伸缩，但拓扑结构不会改变。保证任何有字符的地方都属于某个需要识别的字符。

算法分析：

这题看上去很难。不过我们考虑到拓扑结构不变，而拓扑结构最直接相关的就是“洞”的个数，于是我们来数一下每个图案的洞的个数。结果我们惊奇的发现：6个图案洞个数各不

相同！因此，这题的关键就是找出所有联通块内部有多少个洞。我们用Floodfill找出所有黑联通块标号；然后找出所有白联通块和哪些黑联通块相邻。如果白联通块只和一个黑联通块相邻，那么就是这个黑联通块的一个洞。统计每个联通块的洞数，就能数出所有的字符了。复杂度和图像的像素数一样，显然不会超时。

57 【World Finals 2011 D Chips Challenge】

一位著名的微处理器公司邀请您帮忙在他们的电脑芯片上安排一些组件，每个芯片被设计成拥有N（⩽40）×N个插槽的正方形。一个插槽可以存放一个组件，你要尽可能多地在芯片上安装组件。满足以下的限制：一些插槽是不可用的；一些插槽已经被其他的组件所占据，无法用于固定额外的组件；第i行和第i列的组件数目必须一样多；对于给定的一组A，B，任何一行/一列的组件数都不能超过总组件数的A/B。你希望计算出最多可以在芯片上再安装多少个组件。

算法分析：

这题明显是一个网络流模型，难在如何建模。首先，对于一个插槽(i,j)，如果它放了一个组件，则第i行和第j列各记为多了一个。因此，可以想到用用二分图模型：左边i个点表示i行，右边j个点表示j列，对于一个插槽如果它能放，就在左i和右j间连边。

然后我们逐渐修改建图来把题目中其它元素都融合进去。首先一些插槽不可用——很简单，不连边即可。然后一些插槽必须用，这里可以用下界，但更方便的做法是和“第i行和第i列的组件数目必须一样多”一起考虑——加入费用。左i和右i连边，容量很大（比如100000，记为K），费用0；而插槽的那些边费用为1。此时只要求出最大费用最大流即可。由于原图最大流肯定是K*N，所以最大费用最大流求出来后，第i行和第i列自然就一样多了，否则肯定达不到K*N的流量。必须用的插槽呢？费用足够大即可，比如100000。最后处理局部过热的问题，我们可以枚举一行（列）最多多少个，此时我们发现总组件个数仍然是越多越好（越多越有希望不局部过热；题目要求的也是尽可能多，一举两得）。所以枚举上界后，建图，球最大费用最大流，据费用计算实际零件个数，更新答案。这样可能会超时；不过不难看到，每次行列上界加1，可以在原图基础上修改边权继续增广，所以实际上就是求了一次最大流的复杂度。O(maxflow(2*N,N*N))，不会超时。

58 【World Finals 2011 E Coffee Central】

给出一个R*C（R,C⩽1000）的矩形，其中有N（⩽500000）个格子有标记。一共有Q（⩽20）个询问，每个询问给出一个距离P，你需要找出一个格子(x,y) 使得离(x,y)距离不超过P的标

记格子尽可能多，输出这个(x,y)和有多少个格子。距离定义为曼哈顿距离。

算法分析：

由于Q非常小，所以我们不妨认为只有1个询问，然后暴力做Q次就行了。主要是考虑如何处理曼哈顿距离。对于一个给定的点，距离其曼哈顿距离⩽P的点构成了一个转了45度的正方形。45度旋转的正方形可以用旋转坐标轴的方法处理：对于一个点(x,y)，映射到(x+y,x-y)，此时所有正方形都变成边平行于坐标轴的了（这一点在同年的2011I 中还会看到）。此时，由于坐标范围不大，所以可以用部分和数组预存出所有矩形的标记点有多少个。接下来直接枚举那个(x,y)，O(1)查询部分和，这题就解决了。时间复杂度O(Q*R*C+N)。

59 【World Finals 2011 F MachineWorks】

有N（⩽100000）台机器，你一开始有C元钱，在接下来的D天内，你需要花钱购买机器获得更多的收益。对于每台机器M_i，你已经知道了其价格P_i和可以买入的日期D_i。注意，如果不在第D_i 天买入机器M_i，那么别的人也会买走这一台机器，也就是说，以后你将没有机会购买这台机器了。如果手上的钱低于一台机器的价格，那么你显然不可能买到这一台机器。如果你在第D_i 天买入了机器M_i, 那么可以从第D_i+1 天开始使用这一台机器。每使用这台机器一天，就可以为公司创造出G_i美元的收益。你可以决定要在买入之后的某一天，以一定的折扣价卖出这一台机器。收购市场对于每一台机器，都有一个折扣价R_i。你不能在卖出的那一天使用机器，但是你可以在卖出的那一天再买入一台新的。最后一天，你会卖掉当前所拥有的机器。你的任务就是最大化总收入。

算法分析：

由于任何时刻只能拥有1台机器，而每台机器只能在指定的时间购买，因此将机器按时间排序后，明显问题就有了阶段性。考虑动态规划：f[i]表示第i天最多有多少钱（这里只考虑有机器的天才发生决策），那么转移就是f[i]更新f[i+1]，或者f[i]在第i天买机器，然后枚举在第j天卖掉，f[i]+(i至j间收益)去更新f[j]。

这个动态规划是O(N^2)的，还需要优化。我们观察后发现，时间瓶颈在于第二种转移。观察第二种转移：f[j]=max(f[j],f[i]-(P_i-R_i)+(j-i)*G_i)，我们发现，随着天数增加，收益是一个线性函数，这让我们想到了可以把它表示为一条直线！因此刚刚的动态规划就变成了：插入一条直线，询问x坐标处y最大值是多少。这时候不难看出维护一个半凸壳就行了，而这个半凸壳可以用平衡树维护。插入线段时暴力更新凸壳：因为每条线段被插入删除至多各一次，所以均摊复杂度是O(NlogN)。

60 【World Finals 2011 G magicsticks】

给出N（⩽500）个数，每个数代表了一条线段的长度。你需要把这个序列划分成若干段（不能改变原来顺序），然后把每段的若干条线段拼成一个多边形（多边形也必须按原序拼，即边的顺时针方向顺序必须和原数列中序一致）。如果某一段的若干线段拼不成多边形，则面积视为0。你需要合理的划分序列、合理的拼多边形，最大化所有拼出的多边形的总面积。这个总面积是多少呢？

算法分析：

关于序列上的问题，很容易想到动态规划。f[i]表示前i个数拼出的最大总面积，则我们显然只需要枚举j，然后f[i]=max(f[i],f[j]+maxarea[j+1,i])即可。现在问题是求若干个数能拼出的最大多边形的面积。

对于一段N个线段，我们要拼出一个N边形。当N=3时显然是一个三角形；N=4时，由高中数学知识也不难说明，一定是拼成四点共圆。如果是更大的N呢？其实很容易猜到是N点共圆，具体证明可以用调整法，每次随意找4个点，如果不共圆，则调整使得它们共圆并说明面积一定不会变小。

这一步我们计算是O(NlogC)的，因为求出面积的时候，我们要二分圆的半径，然后根据圆的半径和每一段的长度解这个圆心角总和是不是360度。不幸的是，和绝大多数动态规划不一样，对于一段区间[i,j]如果我们计算出了它拼出的“圆”半径是多少，我们无法利用这个信息快速计算[i,j+1]的答案，只能每次重新计算。因此刚刚说的dp复杂度是O(N^3*logC)的，会超时。我反复试图优化常数，没有结果。

后来我参考了一份题解。题解中表示：应该尽可能让所有的拼成一个多边形而不是拆开；如果要拆，一定是拆最大的边！这个形象的理解就是：如果共圆情况下面积不够大，那么一定是因为圆心在多边形外，所以需要拆掉最大的边。这样一来，每次求一下当前区间多边形的面积，然后拆成两个子区间，这样调用刚刚的方法求共圆时的面积，总复杂度就是O(N^2*logC)了。可惜那个结论不会证明，也没有地方给出有关证明。

61 【World Finals 2011 H Mining Your Own Business】

给出一个N（⩽50000）个点的无向图，你需要选择尽可能少的点，使得：我们任意删掉一个点（和其对应的边；这个点可能是被选择的点），剩下的N-1个点在剩下的图中，都和至少一个被选的点连通。并且你需要计算出，在选的点最少的情况下，不同的方案数。

算法分析：

这题要求“删掉一个点后连通”，只删一个、判连通性，很容易让我们想到点双连通分

量和割点。我们不妨尝试缩点双连通分量，求出所有的割点，然后观察剩下来的图有什么性质。

首先，我们只需要考虑删掉割点的情况，不然整个图都连通，随便选一个就行了；正是因此，我们选的点绝对不可能是割点，另外每个联通块里选谁都是等价的，所以我们只考虑选择哪些联通块。

一个联通块如果只和一个割点相连，那么它必须选一个点——不然唯一的割点被删了就麻烦了。如果一个联通块和不止一个割点相连呢？这时我们发现，由于只被删一个点，所以它一定能沿着割点，最终到达一个“叶子”联通块，也就是那些只和一个割点相连的联通块（这就好比树上删掉任何一个点，剩下来的每个点也一定和一个叶子相连）。所以，对于“叶子”联通块，我们必须选；其它的都不用选。所以最少的点数就是叶子联通块的个数，方案数就是他们的大小之积。

考虑一个特殊情况：如果整个图都连通呢？那么此时要选两个点，否则唯一的点被删了就不行了。其它没有什么特殊情况，实现主要就在于双连通分量缩点。复杂度为O(N^2)。

62 【World Finals 2011 I Mummy Madness】

把沙漠看成一个正方形的网格，共有N（⩽100000）个木乃伊占据了其中的某些位置（坐标点），而你在(0,0)。你与木乃伊轮流移动（你走出第一步）。轮到你时，你可以移动到相邻的8个格子之一，或者站着不动。轮到木乃伊时，每个木乃伊会移动到其相邻的格子之一，使得他与你的欧几里得距离尽量小。允许多个木乃伊同时占据同一个格子。在每个单位时间内，你先做出移动，然后木乃伊做出移动。如果你与任何一个木乃伊站在同一位置，你会被抓住。最优策略下求出经过多少单位时间你会被抓住，或者判断永远不被抓住。

算法分析：

这道题初看还是很难下手的。但对于这种问题，由于多个木乃伊独立行动，因此可以看做一种“最大值最小”类的问题。想到这儿我们可以尝试二分答案。

如果我们二分了一个可能的时间t，那么我们接下来要做的全部事情就是判断能否活到这个时间。我们考虑一种奇特的思维：不考虑怎么走，只考虑我最后走到了哪儿。

首先有一个比较显然（但是很重要）的结论：如果t时间后我走到了(x,y)，而某个木乃伊在t时间内也能走到(x,y)，那么我一定会被它抓住。在此基础上，考虑我t时间后可以到达的区域是一个正方形（转了45度角的正方形），而每个木乃伊到达的区域也是一个转了45度的正方形。我和木乃伊都能到达的区域就是无效区域。那么，我们可以直接求出这N 个无效区域，最后判断无效区域的并是否等于我能到达的区域。如果是的，那么显然我会被至少一个木乃伊抓住；否则我找一个“有效区域”内的点，尽快走到那儿，待着不动——就是安全的。45度旋转的正方形可以用旋转坐标轴的方法处理：对于一个点(x,y)，映射到(x+y,x-y)，

此时所有正方形都变成边平行于坐标轴的了。在这种处理下，正方形求交、若干正方形求面积并都不难，前者直接分情况讨论即可，后者则是直接扫描线处理，并用一棵线段树维护，这也是经典问题。因此整个算法复杂度是二分答案+扫描线，O(Nlog^2N)。

63 【World Finals 2011 J Pyramids】

本题是我分到的第一阶段作业题，有关翻译和详细题解在第一阶段作业里有。

64 【World Finals 2011 K Trash Removal】

给出一个N（⩽100）个点的简单多边形（不一定是凸的）。现在要求你适当旋转它，然后将其滑过一个管道（这里管道视为一个二维平面的长方形）。现在求出管道宽度至少是多少才有可能让这个简单多边形滑过去？

算法分析：

这应该属于一类常见的问题了。看到“适当旋转”，我们应该有一些直觉：肯定要旋转到一条边贴着管道边为止。这个并不难说明——因为如果一条边都不贴着，我们总可以适当旋转，答案肯定不会变劣。当我们确定哪条边贴着的时候，此时答案就是所有点中离这个边最远的了。枚举边，再枚举所有点，要求最远距离尽可能近即可。O(N^2)即可解决。其实这题能做的更优——离某个边最远的点，这个正是熟悉的旋转卡壳！求出原多边形的凸包（其实就是点集的凸包），然后旋转卡壳，代替之前的双重循环枚举即可，O(NlogN)。

65 【World Finals 2012 B Curvy Little Bottles】

给出一个N（⩽10）次多项式函数和一个[l,r]区间，保证f(x)在[l,r]中都大于0。你把这个函数在[l,r]的图像绕x轴转一圈，形成了一个弧柱。你需要求出它的体积，并将它自底向上每25单位体积做个标记。输出标记后的瓶子（即在哪些x的位置标记了）。

算法分析：

这题虽然有各种细节，但关键就在一点：如果给出区间[l,r]，我们如何求这一段内瓶子的体积？如果这个能求了，那么标记容量的事情只要二分答案就可以了。我们考虑，如果给出一个多项式函数，要求它下方到x轴的部分面积是多少，那么我们会对原函数求[l,r]区间内的定积分。这里求定积分的方法很简单，因为是多项式函数，所以求出它的不定积分函

数g(x)，然后g(r)-g(l)就是所求。那么体积呢？类比刚刚的方法，我们发现，如果将函数平方后积分再乘上π，那么得到的函数就是“体积函数”的原函数！而多项式积分是非常容易的，对于每一项k*x^n积分成k/(n+1)*x^{n+1}就行了。然后用这个函数在r处的值减去在l处的值即为所求。有了这个后，所有的问题都可以转化为区间求体积。因为N非常小所以不用在意复杂度，大约是O(N^3)的。

66 【World Finals 2012 C Bus Tour】

给出一个有N（⩽20）个点M条边的无向图，其中1号点是总部，N号点是景点，其余2至N-1个点每个点有个人。你需要驾驶一辆车，从1出发，按某个次序前往接N-2个人，到达N，再按某个次序（和来时不一定一样）送回，最后回到1。这个接送人的次序必须满足：来的时候前N/2个上车的人，回来的时候也必须是前N/2个下车的。在此基础上，你需要最小化总时间。注意，可以路过一个点但是不接送这个点的人，这可能仅仅是为了满足之前的要求。

算法分析：

题目从描述看来就十分像一个哈密尔顿回路问题，再加之N很小，因此状压的求解思路还是十分显然的。不同的是，我们还需要决定每个人是在前半段上车还是在后半段。经过计算，当N=20的时候也只有48620种不同的分配方案决定每个人在哪一半。对于这一步，我们不妨暴力枚举。

接下来，既然有了这个顺序，我们就可以开始执行状压dp了。f[i][j]表示经过点的集合为i，最后到达了j号点，此时的最小费用。由于任何时刻，i⩽512，所以其实复杂度并不高，仅仅是O($2^{N/2-1}$*N)。不过这样常数有点大会超时。一个优化如下：由于每次的前半段状压dp其实是一样的，所以可以先预存一个数组，前半段就直接调用计算好的值，后半段再进行计算，这样能快一倍左右，就可以通过了。

67 【World Finals 2012 D Fibonacci Words】

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。

给定一个模式串p和一个数n，p在F(n)中出现了多少次？n⩽100，len(p)⩽100000

算法分析：

由于fibonacci数列定义时就是通过递推的，所以我们考虑递推求解。dp[i]表示p在f(i)中出现的次数，则我们直观的写出：dp[i]=dp[i-1]+dp[i-2]。这时我们发现我们还漏算了一点东西：如果p出现时跨越两个f(i-1)和f(i-2)则我们需要额外计算。如果能有l[i][j]表示f(i)的前j位和p的后j位是否一致、r[i][j]表示f(i)的后j位和p的前j位是否一致，那么每次转移时O(len(p))就可以维护当前的答案了。关键是l和r数组如何维护。这并不难。对于j比较小的情况（不超过f(i-1)），那么l[i][j]就等于l[i-1][j]；如果j比较大能超过f(i-1)，那么一定是对前几个i才可能（因为f(i)长度指数级增长），所以暴力判断一下就行了。时间复杂度O(len(p)*N)。

68 【World Finals 2012 E infiltration】

给出一个N（⩽75）的竞赛图（即完全有向图），请你选出尽可能少的点，使得N个点中的每一个要么被选出，要么是某个被选出的点的直接后继。

算法分析：

一眼看上去就知道，题意就是求一个有向图的最小支配集。不过同样我们应该知道，求有向图的最小支配集是一个NP问题。即便这题是竞赛图，但也没什么好性质。

不过N非常小，这告诉我们很可能是指数级算法。接着，我们尝试几个例子后就容易发现：因为是竞赛图，所以往往选出很少的点就可以足够“支配”所有的点了。因此我们尝试直接搜索。一个显然需要用的技巧：压位，不过可以用bitset实现。写完这个搜索以后它直接就过了……实际上看了一下官方题解，有一个结论：最多选出6个点就可以了。

69 【World Finals 2012 F Minimum Cost Flow】

一栋楼里有N（⩽400）个水箱，每个水箱有一个三维的坐标（其中z坐标表示海拔）坐标都是整数；第i个水箱上有A_i个孔，并且有一些现存的管道连接这些水箱。水箱体积忽略不计。你现在要从1号水箱注水，目标是使得在不漏水的情况下N号水箱有水。水依照物理原理流过这个系统：如果压力足够让一个节点灌满水，那这些节点将始终充满了水；如果管道横向延伸或者从一个节点向下延伸，那么水也会从那些管道流过；水也会沿着管道向上，直到

达到压力高度的限制。

现在你要做的是：修补一些孔，连接一些新的管道。一条新的管道连接两个水箱，花费为两个水箱的距离，同时占用（也堵住）了这两个水箱的各一个孔。或者你可以直接塞住一个孔。花费为0.5。你需要计算，最少花费多少才能满足上述条件呢（N号水箱有水且不漏水）。注意，这里不漏水并不意味着你要堵住每一个孔；你可以有一些孔没堵住，但你需要同时保证它们所在的水箱你不必经过。

算法分析：

虽然标题是【最小费用流】，不过不难看出这题和网络流没啥关系。我们需要首先分析题目中的一些性质。

首先，我们看到，堵住一个孔的花费是给定的：0.5。这个数字很有趣；我们考虑新的管道的花费——这时可以发现，因为坐标都是整数，所以花费至少为1（也就是堵住两个孔的花费）。这告诉我们，用建管道的方式堵孔是不划算的。我们应该建管道的目的仅仅是为了送水。

不过接下来还有一个问题：水是满足物理条件的，并不是有管道相连的水箱就会有水运送。解决这个问题的想法是枚举一个全局水压——水压就是一个“高度”，在低于这个高度的水箱间，水可以互达，反之则不可以。这个“高度”需要O(N)的枚举，并且不难发现必须高于1号和N号水箱。有了这个枚举之后，我们就是需要求出建一些管道连接1和N了。

这个问题十分像最短路！不过仍然不全一样。由于枚举高度后，有些水箱间可以互达（也就是无向边！），所以我们可以找出一个一个的联通块。此时联通块就是新的节点了。两个联通块之间的距离，等同于他们俩中最近的那对点的距离，这个不妨暴力计算。另外，每访问一个联通块，就得堵上它所有的点（除了两个出入口）。因此，一个新的无向图建出来了，最短路的模型也就明显了。

最后注意一个细节：“两个联通块之间的距离，等同于他们俩中最近的那对点的距离”,前提是这两个选中的点都必须有至少一个孔。不用担心两个孔自己出入的情况，因为这样就没必要经过这个联通块了——这得益于“费用为距离”的性质。至此本题变为最短路问题，加上之前的枚举和建图，时间复杂度为O(N^3)。

70 【World Finals 2012 G A Safe Bet】

给出一个M*N的盒子（M,N⩽1000000），里面有K（⩽400000）面镜子：每一面镜子都放在某个格子中，倾向45度角（也就是有两种方法，输入会告诉你是哪一种）。现在你需要判断以下两个问题：

1.从(1,1)的格子左方水平向右射入一道光线，它能否从(M,N)的格子水平向右射出？

2.如果上一问不能，那么我们可以考虑给某一个格子加一面镜子使得光线从(M,N)水平向

右射出。有多少格子可以满足要求？另外输出字典序最小的那个格子（如果一个格子加两种镜子之一可行，它就是可行的；两种镜子都可行也只算一次）。

算法分析：

首先考虑第一问，第一问显然只要模拟这个光线的行踪就行了。为了模拟这一点，我们需要快速询问：一面镜子上方、下方、左方、右方第一面镜子分别是哪一个。这个显然只要将所有镜子排序（按两种不同的关键字分别排序）就可以了。注意光线肯定只会打到每面镜子的两面最多各一次（因为光路可逆，所以不可能循环，不然就不会从(1,1)进入了）。这样的话，一个O(NlogN)的排序预处理和一个O(N)的模拟就可以解决第一问。

第二问要适当麻烦一点。继续考虑光路可逆：由于目标的光线必须从(M,N)水平向右射出，所以不妨认为这束光是从(M,N)右边水平向左射入的（这两束光等价）。这时，如果我们模拟出这束光的路程，就可以发现——这和原来那束光相交的地方，如果我们放一面镜子，就一定可以把两束光“接”起来，让从(1,1)射入的光最终从(M,N)射出。所以，我们需要求出它们有多少交点。这就是若干个矩形的并求交。矩形分为横着的和竖着的，易知只有横矩形和竖矩形才会相交，所以分别处理，每种处理就是扫描线扫一遍就行了，并用线段树维护区间和。输出字典序最小的方案也非常简单，扫描线的时候顺便记录一下即可。

71 【World Finals 2012 K Stacking Plates】

有N（⩽50）堆盘子，每堆不超过50个。每堆一开始都是从上到下有序的（小的在上大的在下，一样大的顺序随意）。每次可以进行两种操作之一——拆分：任意一堆一分为二，上面一部分下面一部分；或者合并：直接把一堆完全的堆在另一堆上方（要求仍然合法，即上面一堆的最大的不超过下面一堆最小的）。问你最少多少次操作，能把它合并为合法的一堆？

算法分析：

经过简单观察，发现：最优策略一定是先把该拆分的都拆开，最后一鼓作气合并成为一堆。这个很好理解——因为除非最后要构造答案了，否则分开总比合并后灵活。我们采用这样的做法：我们先假设彻底拆散了后再合并，再通过动态规划计算哪些是本不必拆开的。首先，答案的那一堆形态是确定的。对于答案中一段相同的数，它们肯定是来自若干堆，把这些堆都写下来。举个例子，[1,2,6,9][3,8,6][3,4,5,7][1,2,4]表示：最小的数（比如1）来自1、2、6、9堆；次小的（比如2）来自3、6、8堆，然后比如3来自3、4、5、7 堆，最后（比如4）来自1、2、4堆。我们看到，我们要做的仅仅是把每一段中的顺序调整一下，比如1,2,6,9可以换一个顺序；但是段之间顺序不能改，不然最后就不合法了。段内顺序可以任意交换，并且我们发现：如果一段的最后一个和下一段的第一个相等，则我们就可以省掉一

次！所以我们就通过动态规划来决定这个顺序。f[i][j]表示当前处理第i段，最后一个是j；下一段如果有j肯定要放前面（能省显然尽可能省），不然随意，然后枚举谁放最后（中间顺序无意义），进行转移。因此，复杂度就是O(N*D*D)，其中D是总共有多少个不同的大小。不过实际上复杂度完全撑不满，不用担心。

72 【World Finals 2012 L Takeover Wars】

有两个人，第一个人有N（⩽100000）个数，第二个人有M（⩽100000）个数。两个人轮流操作。每次每个人可以进行如下两种操作：选择自己的两个数，删掉他们俩，并加入他们俩的和；或者选择自己一个数，删掉对方某一个比自己小的数。数据保证无论如何操作两个人不可能有相等的数。谁先没有数谁输，问最优策略下先手获胜还是后手获胜呢？

算法分析：

直观上感觉，胜负应该取决于最大的那个是否足够大，而不是取决于剩余数的多少。因此也可以直观的总结出几个策略：

1.如果要删对方的，肯定要删尽可能大的。

2.如果删不掉对方最大的，就不要删。

3.如果要合并自己的两个，那么一定是合并两个最大的。

如果我们枚举第一步先手是合并了还是删了（我们只需要决策：合并还是删，因为这个决定后策略唯一），那么我们接下来发现：如果对方删了我，我下次就只能合并；如果对方合并了，那么分两种情况：如果合并完我还是能删（它最大的，也就是刚合并的），那么它直接输了；如果我删不掉，我就必须合并。因此发现，除了第一步枚举，后面策略都是固定的，所以只要模拟一遍就行了，复杂度为O(NlogN)因为要对输入数据排序。不过这题中，几个策略都是“直观”猜测的——基于一个原则“有用的只是最大的那个”，我没有详细去证明它们。

73 【World Finals 2013 A Self-Assembly】

有N（⩽40000）种“分子”，每种都有无限个。每种分子有四个连接标识来分别表示每条边能与另外分子的哪种边相连。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’）。

每个分子都可以旋转和翻转。当分子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。请判断它们能否组成一个无限大的结构体？这里分子都是大小一样的正方形，并且结构体必须是二维的。

算法分析：

首先我们不考虑“必须是二维的”这一要求。那么，我们可以构建这样一个图：

把A+,B+,...,Z+,A-,...,Z-共52个标识对应52个节点。如果两个标识同时出现在某个分子中，比如A+和B-，那么我们在A+和B+、A-和B-之间连边，表示以A+开始的分子可以接以B+开始的分子（只要通过那个A+和B-共存的分子中转即可）。这个时候，我们发现：“是否能构成无限大的结构体”等价于图中是否存在一个环。这个问题就十分容易了，直接Floyd即可。

然后考虑必须是二维的。由于分子可以旋转或翻转，所以我们可以规定：这个结构体每次只能向右、向下“扩展”。如果需要向左或者向上，那么就翻转一下那个位置（后面的随之翻转），这个正确性非常显然——因此无视这个条件，判断是否有环即可。

74 【World Finals 2013 B Hey, Better Bettor】

你抛一枚硬币，每次抛一下要花费1元，抛完如果是正面你可以得到2元，否则不得到钱。有p的概率会是正面（0⩽p⩽0.5）。你可以进行任意多次，结束时如果你赚了，那么赚的钱都是你的；否则你只需要支付亏的钱的x%。（0⩽x⩽100）。问你期望的最大收益是多少？

算法分析：

观察此题，每次我们做出的决定仅仅是——离开，还是继续玩。这个决定应当仅仅和目前我还剩下多少钱（或者亏了多少钱）有关。也就是，我的策略无非是两个数W和L，如果我手上有W元，或者-L元，我就离开。对于一个二元组(W,L)，我们需要得到期望收益；也就是，我们要计算，“赚W元离开”的概率是多少，剩下来的概率就是亏(1-x)*L元。

设f(i)表示你现在有i元的情况下，最终赢钱离开的概率是多少。显然，f(W)=1,f(L)=0,并且不难列出一个类似方程的东西：f(x)=p*f(x+1)+(1-p)*f(x-1)。手动化简这个式子，可以得到：f(x)=u+v*$(\frac{1-p}{p})^x$。同时，根据f(W)=1,f(L)=0，可以解出u和v，然后代入求出f(0)。最终得到f(0)=$\frac{1-r^L}{r^W-r^L}$。最后我们就枚举所有的二元组(W,L)。通过实验可以发现，W=21000，L=2500就可以通过本题了（这个实验可以在本机进行，因为输入范围有限）。

75 【World Finals 2013 C Surely You Congest】

给出一个N（⩽25000）个点M（⩽50000）条边的无向图，每条边容量为1。在其中一些点有一些车，车辆数不超过1000。每辆车目的地都是1号点。你需要给每辆车安排一个路径使其到达一号点，并且经过的路程是最短路。现在，你需要让其中一些车沿指定的路线（最短路之一）行驶，并且中途他们不能停顿；剩下的车则不允许行驶。任何时刻，每条路的车流量都不能超过容量（容易发现只有一个方向会有车流）。请求出最多能有多少车行驶。

算法分析：

由于是最短路中选择一条，因此可以联想到最短路图。虽然起点有多个，但是终点只有一个，所以从终点出发建最短路图即可。

接着我们发现，因为每辆车都走最短路且中途不能停下，所以如果两辆车到终点的距离不同，他们是不可能互相干扰的，可分别计算。而对于到终点距离相同的所有车，它们到达任何一条边时间都是同样的（比如如果三辆车路径经过3号边，则必然同时经过，不然反推出有车停顿了）。所以，任何一条边只能供一辆车使用。因此这个问题显然就是最大流了。对于每种距离的车，在最短路图上求最大流，最后累加即可。数据范围看似很大，但是因为边容量都是1，所以其实EK比SAP效果好——SAP大量时间花在了标号上。这个算法就可以通过本题了，由于涉及了网络流，复杂度略不好估算，可记为O(1000*maxflow(N,M))。

76 【World Finals 2013 D Factors】

给出一个N（在long long范围内）。你需要找出最小的k，使得k的所有质因子的不同排列方式有恰好N个。保证k在long long范围内能找到。多组数据。

算法分析：

这题就是直接搜索。从小到大用质因数，并且观察到：小的质因数用的次数不会少于大的。所以当前状态dfs(k,prime,times,maxtime)表示当前k是多少，当前用哪个质数，这个质数已经用了多少次（为了计算不同排列数），最多能用多少次（不能超过前面的）。每次枚举下一个质数用prime还是用prime的下一个就行了。注意到即便是prime=2的时候，maxtime 也就63，所以这个搜索非常快就能出解。需要加一个简单的最优性剪枝（就是当前k不能大于已知最优答案），不需要其他剪枝就能通过本题了。

77 【World Finals 2013 F Low Power】

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。2*n*k⩽1000000。

算法分析：

由于是“最大值差最小”，所以自然可以想到二分答案。接着我们发现，我们只需要决定2*n个“关键电池”，剩下来的电池都没有意义——只要每个关键电池都能找到k-1块比它大的即可。如果把输入的所有2*n*k 个电池排序，则容易发现，每个机器的两块“关键电池”肯定是数列中相邻的两块，不然把它换成相邻的两块答案不会更劣。接下来，我们要为每一块关键电池选k-1个比它大的：这就意味着，之前的关键电池肯定要尽可能小。所以我们选最小的n对相邻的数使得差都符合二分答案的要求。然后再为每个关键电池选k-1个比它大的，这个就是直接往后找k-1个还没被选的就可以了，因为也是要尽可能先选小的。所以这一步扫描一遍，就可以判断是否有解，据此二分答案就可以解决本题。时间复杂度O(n*k*log(n*k))。

78 【World Finals 2013 H Матрёшка】

有N（⩽500）个玩偶排成一行，每个都有一个整数的大小，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。

算法分析：

根据题意，合并的玩偶不能被拆开，所以一个区间的玩偶一旦并到了一起，它们就永远在一起了。因此，很明显，我们处理的对象是“连续的一段”。这让我们联想到区间动态

规划：如果f[i][j]表示将i到j这一段合起来最少代价，那么就可以考虑转移了；根据常规的思路，枚举断点k，让[i,k]和[k+1,j]分别合并，最后再合并这两段。

关键在于合并这两段。我们应该计算出最少打开多少个套娃。观察一个例子：[1,2,4,6,10]和[5,9,12]合并，此时5,6,9,10,12都要打开，而1,2,4则可以直接整体移动。很直观的感觉到了：设最小套娃较大的那一段中，最小的体积为C，那么另一段中所有<C的就都不用打开，否则就必须打开！因此，就是求区间内比某个数小的数的个数。然而很容易发现可以离线存储，类似部分和——sum[i][j]表示前i个数中比j小的个数，这样询问就是O(1)的了。因此，整个动态规划就是O(N^3)的。

f数组做完后，因为可以最终合并成不止一个套娃，所以外面还需要一个dp：g[i]表示前i个合并成若干段最少需要几次。这个dp就十分基础了。另外需要注意，如果[i,j]区间中有重复体积的套娃，则f[i][j]代价为无穷大；最后判断一段是否能合并成“合法的套娃”时也需要注意。其它就没什么要注意的了，整个算法复杂度为O(N^3)，属于比较传统的区间动态规划题。

79 【World Finals 2013 I Pirate Chest】

给出一个M*N（M,N⩽500）的池塘，每个位置有个深度。你需要往里放置一个长方体，长不超过a，宽不超过b，体积尽可能大，且放进去以后（可能使水面升高了一些）顶面不能被水面看到。这个长方体最大是多少呢？

算法分析：

经过简单分析可以发现：虽然我们存在一个“放入长方体后水面上升”的问题，但是实际上我们枚举了一条边后，还是最大化他能占有水下的体积最大是多少（另外另一条边显然越长越好；另一条边最长的情况下，水面上升的比例一定，不必考虑）。所以我们考虑枚举一条边。不妨用两重循环枚举最终的子矩形从哪一行到哪一行，这是O(M^2)的。此时，我们相当于是要取出连续的若干列。所以其实这就是一个一维的情况了——因为每一列的权值就是这若干行中这一列的深度最小值。然后我们枚举最后“瓶颈”是哪一列，那么就从这一列向两端尽可能的扩展即可，这一步用单调队列很容易做到O(N)。最后这个底面可能超过a*b，但显然是越大越好，因为多了可以削减成我们需要的限制。所以总复杂度就是O(M^2*N)了。

80 【World Finals 2013 J Pollution Solution】

给出一个N（⩽100）个点的简单多边形（按逆时针方向），求出这个简单多边形与一个半径为R（⩽1000）的半圆（这个半圆的直边平行于x轴）的面积交。

算法分析：

本题属于一个比较明显的计算几何。一个容易想到的做法——因为它们的交每一段仍然是二次曲线，所以可以考虑直接用simpson积分。不过可能有精度问题，我没有尝试直接这么做。

考虑适当的改进。回想我们如何求一个简单多边形的面积：我们随便找一个点，然后逆时针方向将其划分为N-1个三角形，三角形的面积可正可负——由叉积得到。只要我们能求出这些三角形的面积，那么求和就可以了。回到这个问题，我们发现这个思想仍然适用：对于每个三角形，我们求出它和半圆的面积交（这个面积可正可负，用叉积判断一下就行了），最后求和就可以了！三角形和半圆面积求交容易多了，分情况讨论计算即可，simpson积分也可以：因为这种情况下几乎不存在精度问题了。至此本题就被解决了，复杂度为N次simpson。simpson由于运行时间和数据有关（需要处理精度），不方便估计复杂度，但肯定对于本题不会超时。