

# Hedong 解题表格



题目来源	题目名称	题目大意
ZOJ2604 ASC7	Little Brackets	输入整数 $n$ 和 $k$ ，问最深嵌套恰好为 $k$ 层且长度为 $2n$ 的匹配括号的序列个数。
算法讨论	动态规划。用 $f[n][k]$ 表示长度为 $2n$ 深度不超过 $k$ 的括号序列个数。注意到任意一个括号序列都可以由 $(A)B$ 形式唯一表示，则得到状态转移方程： $f[i][j] = \sum \{f[i-k][j] * f[k-1][j-1] \mid 0 < k \leq i\}$ ；边界条件： $f[0][j] = 1$ 需要使用大数	
其它	时间复杂度： $O(N^3)$ 。	
ZOJ2605 ASC7	Under Control	简单 BFS 题目。
算法讨论	数据规模较小，直接宽搜得到答案。	
其它		
ZOJ2606 ASC7	Holidays	按发生顺序输入一定范围年间的 $n$ 条假期新增和取消记录，但记录发生的具体年份未知。问这个范围时间内最多可能有多少天假期，并输出对应方案。
算法讨论	动态规划。用 $f[i][j]$ 表示第 $i$ 条记录发生在第 $j$ 年时到当前时间最多可能有多少天假期，则 $f[i][j] = f[i-1][k] + T$ , $T$ 为第 $i$ 个记录和之前一个记录之间的假期。	
其它	时间复杂度： $O(NM^2)$	
ZOJ2607 ASC7	Laboratory	将一个长方形用一条水平线和一条垂直线分成四块区域，要求所得的面积分别大于等于 $A$ 、 $B$ 、 $C$ 、 $D$ ，问原长方形的最小面积。
算法讨论	最优解必然是或者有三个格子取到等号，或者对角线上的两个格子取到等号。	

其它		
ZOJ2608 ASC7	Maps	给定不同比例尺的三张相同区域的地图，小地图在大地图的内部，要求给出一种中地图的摆放方案，使得中地图在大地图内，小地图在中地图内，且存在小地图内一个点，三张地图在该点地理坐标是相同的。
算法讨论	不知该如何实现。	
其它		
ZOJ 2609 ASC7	Crazy Painter	输入目标染色结果，要求给 $m \times n$ 的格子染色，给连续一段水平格子上色的花费为 $h$ ，给一段竖直格子上色的花费为 $v$ ，给一个格子单独上色的花费为 $s$ 。不能给同一个格子先后上两种不同颜色，要求输出一个花费最小的染色策略。
算法讨论	最小割。把连续相同颜色的水平带或垂直带作为顶点，而由他们相交唯一确定的格子作为连接两点的边，流网络模型：对水平带 $i$ 和垂直带 $j$ 确定的格子 $(i,j)$ ，加三条边 $(S, i, h), (i, j, s), (j, T, v)$ ，最小割就对应最优方案。	
其它		
ZOJ2610 ASC7	Puzzle	$n^2$ 个小正三角形组成了一个边长为 $n$ 的大的正三角形的两个部分。问是否能够通过平移将两部分分离。
算法讨论	分别判断是否能沿每条边的方向分离即可。	
其它		
ZOJ2611 ASC7	Quest	模拟题

算法讨论		
其它		
ZOJ2612 ASC7	Stable Sets	考虑 $\mathbb{Z}_r$ 上的整数多项式，问有多少 $\{0, 1, \dots, r-1\}$ 的子集 $A$ ，对多项式函数 $p$ 和 $q$ ，其值域也为 $A$ 。
算法讨论	<p>考虑只有一个多项式 <math>p</math> 的情形。假设 <math>x</math> 是 <math>A</math> 中的一个元素，那么显然 <math>p(x)</math> 也是 <math>A</math> 中的一个元素。若对所有的 <math>0, 1, \dots, r-1</math> 我们都连一条 <math>i</math> 到 <math>p(x)</math> 的边，则得到一个有向图且每个点的出度都恰好为 1。为保持封闭性，则若 <math>x</math> 属于 <math>A</math>，那么 <math>p(x)</math> 也属于 <math>A</math>；为保证值域与定义域相等，则对于 <math>A</math> 中的任何元素 <math>x</math>，其入度必须大于 0。所以每个稳定的 <math>A</math> 和若干个不相交的简单圈的并是一一对应的。求出这样圈的个数 <math>k</math>，那么最终答案就是 <math>2^k</math>。两个多项式时同理。</p>	
其它		
ZOJ2670 ASC8	Nonoptimal Assignments	构造一个 $n \times n$ 的矩阵，使得按题目所给的“贪心”算法，不能得到正确解。
算法讨论	简单构造。	
其它		
ZOJ2671 ASC8	Cryptography	有 $n$ 个 $2 \times 2$ 的矩阵， $m$ 次查询，每次查询需要快速回答区间 $[i, j]$ 内矩阵的积(mod $r$ )。 $n, m < 30000$
算法讨论	注意到矩阵乘法满足结合律。建立线段树，查询区间积。	
其它	时间复杂度： $O(M \log N)$	
ZOJ2672 ASC8	Fibonacci Sequence	给定一个长度为 $n$ 的序列，求满足 $c[i]=c[i-1]+c[i-2]$ 的最长的子序列。 $n < 3000$

算法讨论	动态规划，用 $f[i][j]$ 表示 $c[1]=a[i], c[2]=a[j]$ 的序列的最大长度，从后往前 F，则状态转移方程为 $f[i][j]=f[j][k]+1$ ，此时要求 $a[k] == a[i] + a[j]$ 。需要使用 <code>unordered_map</code> 来找到 $k$ 继而转移。	
其它	时间复杂度： $O(N^2)$ 。	
ZOJ2673 ASC8	Hexagon and Rhombic Dominoes	问用两个正三角形组成的菱形覆盖边长为 $n$ 的正六边形三角网格的方案数。 $n \leq 7$
算法讨论	按行二进制状态压缩动态规划。考虑六边形的上半部分，状态可只记录头朝下的三角形，若它与上一行的三角形构成菱形的话，记为 1，否则就是与同一行的三角形构成菱形，记为 0。 $f[i][j]$ 记录第 $i$ 行状态为 $j$ 的方案数，那么有 $f[i][j]=\sum\{f[i-1][k]\}$ ，其中 $j$ 是由 $k$ 状态生成的合法状态。因为中间两行的状态必须相同，所以答案就是 $\sum\{f[n][i]^2\}$ ，要使用 64 位整数。	
其它		
ZOJ2674 ASC8	Strange Limit	求 $a^{(a^{(a^{(\dots)})})} \bmod m!$ 的极限
算法讨论	利用欧拉定理：若 $a$ 和 $n$ 互质，那么 $a\varphi(n)=1(\bmod n)$ ；对于任意 $a, n$ 和较大的 $b$ ，有 $a^b=a^{(\varphi(n)+b \bmod \varphi(n))}(\bmod n)$ 。极限记为 $x=f(a, b=m!)$ 。那么假设 $y=f(a, \varphi(b))$ ，就有 $x=a^{(\varphi(b)+y \bmod b)}$ ，而若 $b=1$ ，显然 $x=0$ 。利用 $\varphi(b)<b$ ，递归求解子问题，即可快速得到结果。	
其它		
ZOJ2675 ASC8	Little Mammoth	计算圆和矩形的面积交。
算法讨论	求出所有线的交点，然后用圆心出发的少描线来求交集。由两个交点确定两条扫描线，进而确定一个三角形或一个扇形，然后可以求出这个三角形或扇形的有向面积。所有有向面积的和的绝对值就是答案。	
其它		
ZOJ2676 ASC8	Network Wars	0-1 分数规划
算法讨论	二分答案再用最小割判定。	

论		
其它		
ZOJ2677 ASC8	Oil Deal	给定一个由带权无向边组成的联通图和 $s$ ，要求在保持连通性的前提下，删掉最多条边，所删边的权值和不能超过 $s$ 。
算法讨论	先求最大生成树，那么为了保持连通性，这些边都不能删了。然后对能删的那些边按权重排个序，从权值小的开始一条条删，权值和超过 $s$ 就停止。	
其它		
ZOJ2678 ASC8	Bishops on a Toral Board	问要覆盖一个 $m*n$ 的棋盘，最少需要多少象。
算法讨论	首先覆盖整个棋盘等价于覆盖整个第 0 行。若第 0 行的第 $b$ 列被覆盖了，那么所有形如 $(i \% m, (i+b) \% n)$ 的格子，在第 0 行就是所有形如 $(km+b) \% n$ 的列都被覆盖了，这种数一共有 $n/\gcd(m,n)$ 个。取 $b=0..\gcd(m,n)-1$ 就能把第 0 行完全覆盖。	
其它		
ZOJ2696 ASC9	Chip Designing	给你个芯片，要用水平或垂直的线路连接两排输入输出节点，要求线路不相交。
算法讨论	没想出可行的构造方法。	
其它		
ZOJ2697 ASC9	Electricity	给定一个有向无环图，要求每次改变一条边的方向，使得最后一些边的方向恰好与原来相反，要求中间过程始终没有环。

算 法 讨 论	<p>对于一个 DAG，它可以进行拓扑排序。假设拓扑排序结果中有相邻的两个点 a 在 b 的前面，若 ab 间没有边，那么交换它们的位置后，DAG 还是原来的 DAG；否则一定是有 a 到 b 的边，那么交换位置后，DAG 里 a-&gt;b 的边变成了 b-&gt;a 的边，而其他的边不受影响。所以方案便是不断交换拓扑序中相邻元素，使原始拓扑序变成目标拓扑序。当且仅当 ab 间有重边且要交换这些边的方向时，无解。</p>	
其它		
ZOJ2698 ASC9	Numbers to Numbers	把一篇文章中英语描述的数字全部转为阿拉伯数字（小自然），要求转化的单词数尽量多，若相同，则要求前面的数尽量大。
算 法 讨 论	码农题，没有写出来。	
其它		
ZOJ2699 ASC9	Police Cities	n 个城市由有向的道路连接，要选出 k 个城市设立警察局，要求每个城市都能走到某个警察局，也能被某个警察局走到。求方案数。
算 法 讨 论	<p>先求强联通分量得到的一个 DAG。每个缩点后入度或出度为 0 的联通分量都应该至少有个警察局。接着动态规划求解 <math>f[i][j]</math> 表示在前 i 个入或出度为 0 的联通分量里安排了 j 个警察的方案数，那么 <math>f[i+1][j]=\sum\{f[i][j-y]*c[x][y]\}</math>，x 是这个联通分量里城市的个数。最后答案是 <math>\sum\{f[m][k-y]*c[z][y]\}</math>，z 是所有入出度大于 0 的联通分量里城市的个数之和。需要使用大数。</p>	
其它		
ZOJ2700 ASC9	Quadratic Equation	求系数只能为 0 或 1 的多项式二次方程 $a(t)x^2(t)+b(t)x(t)+c(t)=0$ 的任意一个解,或判断无解。
算 法 讨 论	<p>注意到 <math>x(t)=\sum\{x_i t^i\}</math>，那么 <math>x^2(t)=\sum\{x_i t^{2i}\}</math>，因为系数全部对 2 取模，且 <math>X_i^2 = X_i</math>。待定 x(t) 的系数，令 <math>a(t)x^2(t)+b(t)x(t)+c(t)</math> 的系数全为 0，得到一个异或线性方程组，用高斯消元法解之。</p>	



其它		
ZOJ2701 ASC9	Restore the Tree	要求构造一棵树，满足树的所有叶子之间的距离恰好为输入的。
算法讨论	<p>只考虑那些叶子节点，而认为叶子间的边是带权的。我们先求出解，再检验解，从而判断是否无解。求解方法：首先任意取一个叶子节点 C，这个叶子节点一定连到了一个父亲节点 D，而 CD 间的距离<math> CD </math>就等于 <math>\min\{ AC + BC - AB \}/2</math>。若已有点 A 使得<math> AC = CD </math>，那么 AC 连边，边长<math> CD </math>，删去点 C；否则新增点 D，（注意 D 不一定是叶子）连边 CD，边长<math> CD </math>，删去点 C，而 D 到各点的距离为<math> AD = AC - CD </math>。这样最后生成一棵树，树的节点数不超过 <math>2*n-1</math>。</p>	
其它		
ZOJ2702 ASC9	Unrhymable Rhymes	有一些诗句，现在要求选取一个子序列构成最长的诗歌，诗歌应该由 AABB, ABAB, ABBA 或 AAAA 这四种形式的小节组成。
算法讨论	<p>实际上合法小节的充要条件就是由两组两句相同的诗句组成。那么假设处理完了前 i 句，我们会找最小的满足存在 <math>i &lt; j' &lt; j \&amp;\&amp; a[j'] = a[j]</math> 的 j，再找最小的满足存在 <math>i &lt; k' &lt; k \&amp;\&amp; k' \neq j' \&amp;\&amp; a[k'] = a[k]</math> 的 k，这样 j', j, k', k 排序后可组成一个小节，而且这种贪心是最优的。通过预处理和维护一个数组 c，c[i]表示在可选诗句里，i 之前的最后一句与 i 相同的诗句的序号，扫描一遍可得解。</p>	
其它		
ZOJ2703 ASC9	Selling Tickets	一辆车有 9 个 4 人座的隔间，现在要安排座位，使得总满意度最大。乘客分成了不相交的 1~4 人小组，若有 i 个来自同一小组的人分在同一个隔间，他们的满意度为 $i*(i-1)*c$ ，c 是一个小组相关的系数。求最大满意度。
算法讨论	<p>1 个人的满意度总是 0 的，所以别人都安排好后，最后随便塞就好了。4 个人的隔间有价值的安排方案无非是 2+2/3/4。3 个人的小组要么在一起，要么拆成 2+1 或 1+1+1；4 个人的小组要么在一起，要么拆成 2+1+1 或 1+1+1+1，3+1 和 2+2 都不会是最优解；且 3/4 个人的小组里都最多只有一个拆出了 2；而且把系数高的安排在一起，系数低的拆开，满意度更大。所以只要枚举 3 和 4 有多少组在一起，然后贪心就可以了。</p>	

其它	
----	--

题目来源	题目名称	题目大意
ZOJ2704 ASC10	Brackets	给定一个只含0[]的字符串，问其中最长的合法字符串是什么。
算法讨论	用一个 sub[]记录匹配成功的序列，从中查找最长的序列，然后输出对应的字符串序列即可。这样可以较好的处理形如[OO][]的序列。	
其它		
ZOJ2705 ASC10	Dividing a Chocolate	给定一个 $m*n$ 的矩形，初始可以沿任意的整数位置分成两个矩形。然后始终让大矩形减小矩形，直到两个矩形相等，目标是最后的矩形面积最小。中间不能出现一个矩形大面积大于另一个的两倍的情况。 $m,n < 1e9$
算法讨论	若最后是两个 $a*b$ 的矩形，那么我们可以反过来推出之前较大的那个矩形依次是 $2a*b, 3a*b, 5a*b, 8a*b \dots$ 系数为 Fibonacci 数列。则我们要找最大的 Fibonacci 数 $f$ ，使得 $m\%f==0 \parallel n\%f==0$ ，枚举 $f$ 就可以了，答案为 $m*n-m*n/f$ 。	
其它		
ZOJ2706 ASC10	Thermal Death of the Universe	给定一个长度为 $n$ 的整数序列，进行 $m$ 次区间操作，每次操作就是使得区间内的数等于它们的平均数取整，上/下取整根据当前总和与初始总和的关系。
算法讨论	线段树。	
其它		
ZOJ2707 ASC10	Equation System	给你一个系数在模 2 运算下的多项式构成的二元一次线性方程组，试求解。

算法讨论	<p>对于求解</p> $\begin{cases} a_1(t)x(t)+b_1(t)y(t) = c_1(t) \\ a_2(t)x(t)+b_2(t)y(t) = c_2(t) \end{cases}$ <p>记 <math>D=a_1b_2-a_2b_1</math>, <math>X=a_1b_2-a_2b_1</math>, <math>Y=a_1b_2-a_2b_1</math>。</p> <p><math>D</math> 不是零（多项式），这种情况最简单，答案就是 <math>x(t)=X/D</math>, <math>y(t)=Y/D</math>，若不整除，那么无解，其他情况；</p> <p>若 <math>D</math> 为零，但 <math>X</math> 和 <math>Y</math> 不全为零，这时候也无解；</p> <p>若有 <math>0+0 \neq 0</math> 的情况显然也是无解的；</p> <p>若都是 <math>0+0=0</math>，那么任意 <math>x(t), y(t)</math> 都是解；</p> <p>最后，就是方程组是冗余的情况，问题转为一个方程 <math>a(t)x(t)+b(t)y(t)=c(t)</math> 的情况。和解 <math>ax+bt=c \pmod m</math> 一样，我们用扩展欧几里德算法来求解。</p> <p>利用 <code>extGcd</code> 可以得到 <math>x'(t)a(t)+y'(t)b(t)=\gcd(x(t), y(t))=g(t)</math>：</p> <p><math>c(t)</math> 不整除 <math>g(t)</math>，那么无解；反之有解，令 <math>c'(t)=c(t)/g(t)</math>，则 <math>x(t)=x'(t)*c'(t)</math>, <math>y(t)=y'(t)*c'(t)</math>。</p>	
其它		
ZOJ2708 ASC10	Fool's Game	<p>开始 A 打出数字一样的一些牌，然后 B 总要大过这些牌。牌 <math>x</math> 比牌 <math>y</math> 的大，当且仅当它们花色相同且 <math>x</math> 大，或者 <math>x</math> 是给定的 <b>trump</b>，而 <math>y</math> 不是。</p> <p>此后 A 继续出牌，但有一个限制，A 打出的牌的数字必须已经被打出过。问谁必胜。</p>
算法讨论	<p>假设胜负已分，那么 B 胜的情况就是打出的牌中 B 都大过 A 了，而却无牌可打了，即没有和打出的牌数字相同的牌。若牌局不可能发展到这样一个状态，那么 B 必败无疑，否则 B 必胜。则必胜策略为：首先 B 确定好最后打出牌的数字的集合，以 A，B 牌该数字的子集为二分图的点，<b>cover</b> 关系为边，利用匈牙利算法匹配，若 A 都被覆盖了，则 B 必胜。</p>	
其它		
ZOJ2709 ASC10	Lottery	<p>给定一个数字 <math>n</math> 和一个字符串 <math>S</math>，要求构造一个包含 <math>n</math> 个字符的集合，使得从中选出 <math>S</math> 的概率最大或最小。概率不能为 0，字符都要来自 <math>S</math>，无序。</p>

算法讨论	假设字符集大小为 $m$ ， $S$ 里有 $x_i$ 个字符 $c_i$ ，集合里有 $y_i$ 个字符 $i$ ，那么 $x_i \leq y_i$ ， $\sum y_i = n$ ，概率 $p = \prod C_{y_i} x_i / C_n  S $ 。要求 $p_{\max}$ 和 $p_{\min}$ ，先令所有的 $y_i = x_i$ ，然后选择 $y_k$ 增加使得 $\sum y_i = n$ 。每次给 $y_k$ 加 1，就等于对 $p$ 乘 $p' = (1+y_k)/(y_k+1-x_k)$ ，分析其性质知道， $y_k$ 越大值越小； $x_k$ 越大，值随 $y_k$ 减小得越慢。贪心求解即可。	
其它	时间复杂度为 $O(N^3)$ 。	
ZOJ2710 ASC10	Two Pipelines	每个城市可以连到一条线，有关于距离的费用。 要求在分别连接两条线的城市数相差不超过 $c$ 的前提下，总花费最小，输出方案。
算法讨论	先计算点到直线的距离从而求出每个点到每条线的费用。然后动态规划，记 $f[i][j]$ 表示前 $i$ 个城市有 $j$ 个连到了第一条线，则 $f[i][j] = \min(f[i-1][j-1] + \text{cost1}[i], f[i-1][j] + \text{cost2}[i])$ 。答案取 $f[n][(n-c+1)/2..(n+c)/2]$ 中的最小值。	
其它		
ZOJ2711 ASC10	Regular Words	求由 A, B, C 构成长度为 $n$ 的序列的个数，要求每个 B 前面的 A 的个数不少于当前 B 的个数，每个 C 前面的 B 的个数不少于当前 C 的个数。
算法讨论	动态规划。 $f[a][b][c] = f[a-1][b][c] + f[a][b-1][c] + f[a][b][c-1]$ ; 其中 $f[X][Y][Z]$ 满足 $X \geq Y \geq Z$ 。注意使用大数。	
其它		
ZOJ3362 ASC11	Beer Problem	源点城市 1 可以生产任意数量的啤酒，销往城市 2~ $n$ ，每个城市都有不同的售价，但销量无限制，连接城市的道路只能运输有限数量的啤酒，并且运啤酒需要运费。求最大收益。
算法讨论	最小费用最大流。在原图的基础上增加一个汇点，每个节点到汇的容量无穷大，费用为售价的相反数。费用为正的时候已亏本，则退出。	
其它		

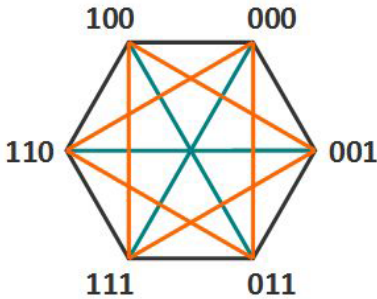
ZOJ3363 ASC11	Another Brick in the Wall	给出砖块稳定的四种条件，问有多少种第一层为 $m$ 个连续砖块高为 $n$ 层的砖墙？
算法讨论	裸状态压缩动态规划。	
其它		
ZOJ3364 ASC11	Black and White	问一个无限国际象棋棋盘内的简单多边形内有多少黑/白格子。
算法讨论	遍历一下这些点，用前一个点与原点围成的矩形的面积减去后一个点与原点围成的矩形的面积，遍历到最后就可以得出这个几何图形的总面积，再求出黑色方块的个数即可。	
其它		
ZOJ3365 ASC11	Integer Numbers	给定一个数列，要修改最少的数，使得结果序列是公差为一的等差数列。
算法讨论	注意到公差为一的等差数列每项减去其下标为常数，所以只要对所有 $a[i]-i$ 计数，取计数最多的上述差值，改变数列，得到答案。	
其它		
ZOJ3366 ASC11	Islands	一个六角网格中，依次加入一些六边形，若加入后会产生大于 $s$ 的联通块的话，则该次加入操作被忽略。问执行所有操作后，各个联通块的大小。
算法讨论	用并查集来保存一个集合的关系。每当读到一个新的单元，扫描它周围 6 个单元，看他们是不是陆地，若是海洋的话那么直接跳过，若是陆地的，那么就要处理了。利用并查集维护并回滚操作。	
其它		
ZOJ3367 ASC11	Counterfeit Money	求两个矩阵的最大公共矩形区域。

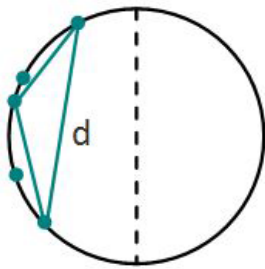
算法讨论	先枚举第二个矩形相对第一个矩形的偏移(x,y),然后再用动态规划求最大公共矩形。	
其它		
ZOJ3368 ASC11	Chinese New Year	问一个边长为整数 a, b, c 的三角形里最多能放多少个单位圆, 若超过 10 个, 则当作 10 个。
算法讨论	迭代加深搜索, 实现起来比较难。	
其它		
ZOJ3369 ASC11	Saving Princess	给定人初始的 h, s, p, m 值, 每一关可以攻击 D 或 睡眠 E 敌人, 使用招式需要一些条件。问能否过关和方案。
算法讨论	动态规划。f[i][j][k]表示搞定前 i 个怪兽, 其中攻击了 j 个怪兽, 睡了 i-j 个, 魔法剩下 k 时剩余的最大血量。	
其它		
ZOJ3370 ASC11	Radio Waves	平面上有 n 个基站, 他们的覆盖半径要相同。每个基站可以使用两个频率中的一个, 但是一个地方不能被两个使用相同频率的基站同时覆盖。求最大覆盖半径。
算法讨论	二分覆盖半径, 然后判断是否可以用至多用两个频率分配方案。	
其它		
ZOJ3371 ASC11	Cheater's Shuffle	给定几种种洗牌规则, 问是否能通过不超过 t 次操作, 把牌组从一个排列洗到另一个排列。

算法讨论	直接爆搜状态太多，所以用双向搜索，状态就在可接受范围内了。所以从初状态出发搜 $t/2$ 步得到集合 S，从末状态出发反搜 $t/2$ 次得到集合 T，有解当且仅当 S 和 T 有交集。	
其它		
ZOJ3372 ASC11	Gone Swimming	一个矩形泳池，分成了一个一个矩形区域，每个区域深度不同，其中一个区域是水源，问水填满池子要多久。满的水会往边界外流，流量和边界长度成正比。
算法讨论	类似狄杰斯特拉那样模拟即可。	
其它		
CEOI2010 自选	All	<div style="display: flex; flex-wrap: wrap;"> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>elves</p> <pre> ...  .X.  ...  ... .OX  .O.  XO.  .O. ...   ...   ...  .X. </pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>humans</p> <pre> .X.  .X.  ...  ... .OX  XO.  XO.  .OX ...   ...  .X.  .X. </pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>dwarves</p> <pre> .X.  .X.  .X.  ... .OX  XOX  XO.  XOX .X.  ...  .X.  .X. </pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>hobbits</p> <pre> .X. XOX .X. </pre> </div> </div> <p>在一个 <math>R \times C</math> 的网格上 (<math>R, C \leq 70</math>)，某些方格中有部落。现在他们要相互结盟。每个部落可以与它上下左右四个方向的部落结盟。上图表示了四种部落每种可能的结盟方式。(O 为部落，X 为部落的盟友)。要求输出任意一种方案或无解。</p>
算法讨论	因为每个部落只能与相邻部落连边，所以我们可以对整个网格黑白染色。增加源点和汇点，源点向黑色部落连一条边，容量为该部落的结盟数；白色部落向汇点连一条边，容量为该部落的结盟数；黑色部落向每个与它相邻的部落（都是白色的）连一条边权为 1 的边。这样子做一次最大流，若满流我们就可以通过残余网络找到答案，否则无解。	

	<p>这样的构图对于 humans 的部落求出来的方案可能不合法。观察 humans 的结盟方案，可以看成在上下两个方向找一个部落结盟，左右两个方向找一个部落结盟。由于两种方式互不影响，我们可以把这种部落拆成两个点，一个点负责与左右部落的结盟，另一个点负责与上下部落的结盟。然后使用最大流算法即可求出答案。</p>	
其它	<p>时间复杂度：<math>O((R*C)^3)</math></p> <p>空间复杂度：<math>O(R*C)</math></p>	
CF418D 自选	Big Problems for Organizers	<p>在 <math>n (n \leq 1e5)</math> 座宾馆里举行一个盛会。其中的两座宾馆将要主办所有的活动，剩下的宾馆将会容纳所有参赛者。这些宾馆被总共 <math>n-1</math> 条道路连接，使得你能从任一座宾馆到达另外任意一座。</p> <p>组委会想要知道，若通过一条连接两座宾馆的道路耗费一单位时间，并且每个参赛者的目的地是距离他们自己最近的主宾馆之一；那么，所有参赛者同时出发，最后所有人均到达目的地的最少所需时间为多少。委员在研究 <math>m (m \leq 1e5)</math> 种主宾馆分布的方案。对于每种方案，帮助委员会找到最少所需时间。</p>
算法讨论	<p>以任意点为根，建立一棵有根树。然后通过对于每个节点 <math>i</math>，计算它的子树中，距离它最远的三个属于它的不同儿子后代的节点到它的距离，记为 <math>dis1[i], dis2[i], dis3[i]</math>。</p> <p>然后计算以下三个数组：</p> <p><math>f[i][j]</math>: 从节点 <math>i</math> 开始往根的方向走 <math>2^j</math> 步，所到达的节点。</p> <p><math>up[i][j]</math>: 属于节点 <math>i</math> 到 <math>f[i][j]</math> 路径上节点的子树，距离节点 <math>i</math> 距离最远的节点与节点 <math>i</math> 的距离。</p> <p><math>down[i][j]</math> 属于节点 <math>i</math> 到 <math>f[i][j]</math> 路径上节点的子树，距离节点 <math>f[i][j]</math> 距离最远的节点与节点 <math>f[i][j]</math> 的距离。</p> <p>然后对于询问 <math>(x,y)</math>，找到 <math>(x,y)</math> 的 LCA 节点 <math>x</math>，找到 <math>x,y</math> 路径的中点 <math>h</math>。然后根据节点 <math>h</math> 可以将这棵树分成两个部分，一部分距离 <math>x</math> 更近，另一部分距离 <math>y</math> 更近。然后根据预处理的 <math>up[i][j], down[i][j]</math> 以及 <math>dis1, dis2, dis</math> 即可分别计算两部分距离节点 <math>x,y</math> 最远的点的距离。</p>	
其它	<p>时间复杂度：<math>O((n+m)\log n)</math></p>	

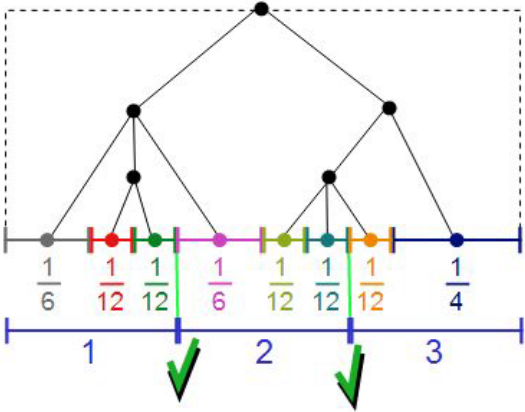


	空间复杂度：O(nlogn)	
CF406E 自选	<p>Hammig Tripples</p> <p>克里斯梦到了 <math>m</math> 个从 1 到 <math>m</math> 编号的长度为 <math>n</math> 的二进制串。最使人震惊的是，每个二进制串的各个位均按照不下降或者不上升顺序排列。例如，克里斯能梦到接下来这四个长度为五的二进制串：</p> <p>00011 00000 11110 11111</p> <p>两个长度为 <math>n</math> 的串 <math>a</math> 和 <math>b</math> 之间的汉明距离 <math>H(a, b)</math> 为对应符号不相同的位数。克里斯认为每三个有不同编号的二进制串组成一个三角；并且他有一种错觉，只要他能计算出在所有用梦到的二进制串构成的三角 <math>a, b, c</math> 中，和 <math>H(a, b) + H(b, c) + H(c, a)</math> 为最大值的三角数，就能醒来。</p> <p>请帮助困于噩梦中的克里斯！</p> <p>我们用两个整数 <math>s_i</math> 及 <math>f_i</math> (<math>0 \leq s_i \leq 1; 1 \leq f_i \leq n</math>)，描述编号为 <math>i</math> 的二进制串：第 <math>i</math> 个二进制串的前 <math>f_i</math> 位等于 <math>s_i</math>，然后剩下 <math>n - f_i</math> 位等于 <math>1 - s_i</math>。</p> <p><math>n \leq 1e9, m \leq 1e5</math></p>	
算法讨论	<p>首先，所有不同的字符串最多只有 <math>2n</math> 种。建立一个正 <math>2n</math> 边形。对于给出的字符串 <math>(s_i, f_i)</math> (开头的连续 <math>f_i</math> 个字符为 <math>s_i</math>)，按照 <math>s_i</math> 为第一关键字，<math>f_i</math> 为第二关键字排序，即可得到下图。</p>  <p>不难发现，两个字符串之间的距离即为它们之间的连线段跨过的边数。</p>	

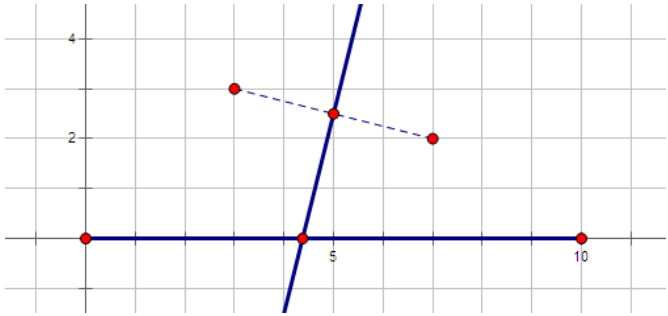
	<p>问题转化成在给出多边形的若干个顶点，要求选择其中三个点 <math>a, b, c</math>，使得它们两两之间的距离和最大的方案数。</p> <p>对于这个问题，可以分成两种情况。</p> <p>三个点均位于正多边形外接圆直径的一侧。</p>  <p>此时显然答案为 <math>2d</math>，<math>d</math> 为 <math>a, b, c</math> 三点中两点距离的最大值。</p> <p>2，对于其它情况，多边形所有边均被三点两两之间的连线段恰好跨过一次，答案为 <math>2n</math>。</p> <p>因此第一步检验所有 <math>m</math> 个顶点是否落在一个连续的长度不超过 <math>n</math> 的区间内(注意 1 和 <math>2n</math> 头尾相连)。若所有 <math>m</math> 个顶点不都落在一个连续的长度不超过 <math>n</math> 的区间内，那么只要用 <math>m</math> 个点中选 3 个点的总方案，减去所有 3 个点落在一个长度不超过 <math>n</math> 的区间内的方案即可。这一步只需要排序后扫描一遍即可。</p> <p>反之，只要排序后扫描一遍，找到距离每个点最远的点，求出两点间的最大距离。然后枚举三个点中编号最小的点，根据最大距离即可计算出编号最大的点，然后统计方案即可。</p>	
其它	<p>时间复杂度: <math>O(m \log m)</math></p> <p>空间复杂度: <math>O(m)</math></p>	
CEOI2009 自选	Harbingers	<p>给出一棵 <math>n</math> 个节点树，根为 1，给出树上每条边的长度 <math>d_i</math> 和这条边连接的两个节点 <math>u_i, v_i</math>。当一个点 <math>i</math> 需要向根节点传递信息时，需要先花费 <math>S_i</math> 的时间把信息告知节点 <math>i</math> 的信使，然后信使开始往根节点的方向移动。当信使到的一个新的节点 <math>j</math> 时，他有两个选择：1，自己继续向根节点前进。2，花 <math>S_j</math> 的时间把信息交给当地的信使，然后由当地的信使向根节点传递信息。</p> <p>第 <math>i</math> 个节点的信使通过一个单位长度的路程所花费的时间 <math>V_i</math>。 <math>3 \leq n \leq 100000</math>，其它出现的数子</p>

		$\leq 10^9$ 。问从每个节点传递信息到根节点的最短时间。
算法讨论	<p>动态规划的解法的是显然的。令 <math>f[i]</math> 为第 <math>i</math> 个节点传递信息到根节点信息的最短时间。对于 <math>i &gt; 1</math>，不难列出方程 <math>f[i] = \min(f[j] + V[i] * (D[i] - D[j]) + S[i])</math>，<math>j</math> 为 <math>i</math> 的祖先。</p> <p>首先考虑条链上的问题。将转移方程拆开，得到 <math>f[i] = \min(f[j] - V[i] * D[j]) + V[i] * D[i] + S[i]</math>。括号外的部分与 <math>j</math> 无关，不考虑。对于括号内的部分，将 <math>(D[j], f[j])</math> 看成平面上的点，用一条斜率为 <math>V[i]</math> 的直线从 <math>x</math> 轴下方无穷远处向上移动，碰到的第一个点即为最优解。</p> <p>因此不在下凸壳上的点没有意义。又因为在转移的过程中，当前节点 <math>i</math> 的 <math>D[i]</math> 值单调递增，所以可以用一个栈按照从栈底到栈顶斜率严格递增的顺序，来维护下凸壳，每次加入一个点 <math>(D[i], f[i])</math> 时，首先删除栈顶 <math>f[j] \geq f[i]</math> 的元素，然后删除加入 <math>i</math> 后导致斜率不递增的栈顶元素，然后将 <math>i</math> 加入栈中。对于每次查询，可以通过二分查找找到满足和下一个节点的斜率不小于当前节点 <math>V[i]</math> 的最靠近栈顶的节点，即为最优值。</p> <p>对于一条链的问题，每个节点最多进栈一次，出栈一次，因此这两种操作可以暴力实现，总的时间复杂度为 <math>O(n)</math>，每次查询为 <math>O(\log n)</math>，最多 <math>n</math> 次查询，因此总的时间复杂度为 <math>O(n \log n)</math>。</p> <p>对于一颗树上的问题，同样可以用一条链上的算法。但从子节点返回父亲节点时，需要把被删除的元素重新入栈，这样每个点可能进出栈多次，用暴力实现会 TLE。但不难发现，每个节点入栈删除的元素只会是栈顶开始连续的若干个元素，然后入栈。因此若用一个数组存储单调栈的话，可以用一个元素记录父节点原先栈的大小，令一个元素记录这个节点入栈的位置上原来的元素。这样就可以用 <math>O(1)</math> 的时间将栈复原。然后对于维护下凸壳的操作，可以用二分查找在 <math>O(\log n)</math> 的时间内计算出要删除的元素个数。这样维护、查询的复杂度均为 <math>O(\log n)</math>，总的时间复杂度依然为 <math>O(n \log n)</math>。</p>	
其它	<p>时间复杂度： <math>O(n \log n)</math></p> <p>空间复杂度： <math>O(n)</math></p>	
2015 湖南集训 自选	Atree	小 A 有一棵 $N$ 个点的树，每个点都有一个小于 $2^{20}$ 的非负整数权值。现在小 A 从树中随机选择一个点 $x$ ，再随机选择一个点 $y$ ( $x, y$ 可以是同

		一个点),并对从 $x$ 到 $y$ 的路径上所有的点的权值分别做 <b>and</b> 、 <b>or</b> 、 <b>xor</b> 运算,最终会求得三个整数。小 A 想知道,他求出的三个数的期望值分别是多少。
算法讨论		先按每一个二进制位计算,因为所有二进制位都是不相关的。然后对于每个二进制位,用树形 F 的方法求出以它为起点的路径上的点上的权值作 <b>and</b> 、 <b>or</b> 、 <b>xor</b> 运算结果为 1 的路径条数,从而求得答案。
其它		时间复杂度: $O(N)$ 空间复杂度: $O(N)$
TC SRM609 自选	Lottery Tree	<p>你发行了一种彩票,并且有 <math>P</math> 人买了它。现在你要决定谁中奖。你已经决定了用一个有根树来选择优胜者。你需要做的事情被列在下面:</p> <ul style="list-style-type: none"> <li>•参与者从 1 到 <math>P</math> 连续编号。</li> <li>•首先,你将树画在一个矩形的白板上,并需要满足以下条件: <ul style="list-style-type: none"> <li>–树上的每一个结点对应白板上的一个圆圈。圆圈很小以至于你可以忽略他们的大小。</li> <li>–树上的每条边对应连接两个圈的线段。没有两条边相交。</li> <li>–根节点一定画在白板的最上方。</li> <li>–对于每个结点,连向它们儿子们的边都要向下走。(换句话说,父亲必须画在儿子的上方。)</li> <li>–所有的叶子需画在白板的底边。</li> </ul> </li> <li>•接下来,你将白板的底边分成 <math>P</math> 段,每个叶子被恰好一条线段包含。将 1 到 <math>P</math> 这些个不同的数字分配给每个线段,然后将数字标注在线段内的叶子上。</li> <li>•现在,你要重复下述过程:找到一个空结点 <math>X</math>,它的所有儿子都标注了数字;若有多个这样的结点,随机选择一个;然后在 <math>X</math> 的儿子中随机选择一个,将这个儿子的数字标注在 <math>X</math> 上;当根节点</li> </ul>

	<p>被标上数字时过程结束。</p> <p>你会获得整数 <math>P</math> 和用于描述你使用的树的数组 <math>tree</math>。这棵树有 <math>N</math> 个结点，从 <math>0</math> 到 <math>N-1</math> 编号。结点 <math>0</math> 是树根。对于每个其他结点，它父亲的编号小于它的编号。更正式的说，对于在 <math>1</math> 和 <math>N-1</math> 之间的 <math>i</math>，<math>tree[i]</math> 是结点 <math>i</math> 父亲的编号。你想要让抽奖公平，即，保证每个参与者有相同的机率赢取大奖。为此，你可以在合法的要求下选择怎样画这棵树，以及怎样分配数字到叶子上。回答“YES”（不需要引号）若抽奖可以公平，否则回答“NO”。</p> <p><math>P, N \leq 100</math></p>
算法讨论	<p>不难发现，每个叶子和答案相等的概率和画出树的具体形态无关。因此，每个子树的答案和最终答案相等的概率也和树的具体形态无关。</p> <p>令根节点对应区间 <math>[0,1]</math>，如图，按照从左到右的顺序，每棵子树都对应区间 <math>[0,1]</math> 上连续的一段。</p>  <p>令 <math>F(i, x, j)</math> 表示，能否将节点 <math>i</math> 对应子树能否匹配区间 <math>[j/x, (j+1)/x]</math> 上。</p> <p>首先考虑特殊情况，若 <math>i</math> 为叶子，那么若区间 <math>[j/x, (j+1)/x]</math> 包含 <math>1/P, 2/P, 3/P, \dots, (P-1)/P</math> 中的任意一个点，则返回 <code>false</code>，否则返回 <code>true</code>。</p> <p>设节点 <math>i</math> 有 <math>g</math> 个儿子，那么显然 <math>g</math> 个儿子各自对应的区间长度相当。显然 <math>g</math> 个儿子应该分别与区间 <math>[j/x, j/x+b/gx], [j/x+b/gx, j/x+2b/gx], \dots, [(j+1)/x-b/gx, (j+1)/x]</math> 一一对应。</p> <p>因为 <math>g</math> 个儿子的顺序可以任意调整，因此应该意义判断 <math>g</math> 个儿子能否放</p>

	<p>在 <math>g</math> 个区间上。假设区间 <math>[j/x+b/gx, j/x+(b+1)/gx]</math> 没有包含（端点除外）<math>1/P, 2/P, 3/P, \dots, (P-1)/P</math> 中的任意一个点，那么显然该区间可以和任意一个子树对应。否则问题转化成一个规模更小的子问题。节点 <math>a</math> 能否放在区间 <math>[jg+b/(xg), (jg+b+1)/xg]</math> 上，这个可以递归解决。</p> <p>处理了 <math>g</math> 个儿子和 <math>g</math> 个区间的匹配后，建立一个二分图，儿子为 <math>X</math> 部，区间为 <math>Y</math> 部，若儿子 <math>i</math> 和区间 <math>j</math> 能够匹配，在 <math>i</math> 和 <math>j</math> 之间连一条边。若二分图存在最大匹配，<math>F(i, x, j)</math> 为真，否则为假。最后状态 <math>F(0, 1, 0)</math> 的返回值即为所求答案。</p> <p>最后是时间复杂度的分析，虽然状态是三维，但不难发现，对于确定的 <math>i</math>，它对应的 <math>x</math> 值是唯一的，因此不同的状态最多有 <math>O(n^2)</math> 个，转移 <math>O(n^3)</math>，因此总的时间复杂度为 <math>O(n^5)</math>，但实际上远远达不到。</p>	
其它	<p>时间复杂度： <math>O(n^5)</math></p> <p>空间复杂度： <math>O(n^2)</math></p>	
2014 清华集训 自选	mex	<p>有一个长度为 <math>n</math> 的数组 <math>\{a_1, a_2, \dots, a_n\}</math>。 <math>m</math> 次询问，每次询问一个区间内最小没有出现过的自然数。</p> <p><math>n, m \leq 1e5</math></p>
算法 讨论	<p>每个点维护一个函数式线段树，存储到这个位置每个数最后一个数出现的位置。显然有一个二分做法，复杂度为 <math>O(\log^2 n)</math>，利用线段树的特点可优化到 <math>O(\log n)</math>。</p>	
其它	<p>时间复杂度： <math>O((n+m)\log n)</math></p>	
BOI2012 自选	Mobile	<p>给出一个端点分别为 <math>(0, 0)</math> 和 <math>(L, 0)</math> 的线段，按照 <math>x</math> 坐标不减给出 <math>n</math> 个整点。问线段上的点中，距离最近整点距离的最大值。</p> <p><math>n \leq 1000000, 1 \leq L \leq 10^9</math>，坐标绝对值不超过 <math>10^9</math>。</p>
算法 讨论	<p>假设所有 <math>n</math> 个点 <math>x</math> 坐标均不相同。若有两个点 <math>x</math> 坐标相同，那么显然只需要保留 <math>y</math> 坐标绝对值较小的一个即可。</p>	

	 <p>如图，接下来对于相邻顺序的两个点，作两点连线段的中垂线。显然，中垂线左侧的点距离 <math>x</math> 坐标较小的点较近，中垂线右侧的点距离 <math>x</math> 坐标较大的点较近。因此按 <math>x</math> 坐标的顺序从小到大扫描 <math>n</math> 个整点，用一个栈保存可能成为最近点的整点。同时维护栈中两点连线段中垂线与给出线段交点 <math>x</math> 坐标单调递增。每当加入一个点前，判断是否破坏了栈的单调性，若不满足单调性则弹出栈顶元素直到单调性被满足，再将新点加入栈顶。最后答案即为线段的端点以及所有中垂线交点与他们最近点的距离的最大值。</p>	
其它	时间复杂度： $O(n)$ 空间复杂度： $O(n)$	
2015 北京集训 自选	Data	<p>现在，二维平面上有 <math>N</math> 个点。Alex 需要实现 <math>M</math> 个以下三种操作：</p> <ol style="list-style-type: none"><li>1. 在点集里添加一个点；</li><li>2. 给出一个点，查询它到点集里所有点的曼哈顿距离的最小值；</li><li>3. 给出一个点，查询它到点集里所有点的曼哈顿距离的最大值。</li></ol> <p>两个点的曼哈顿距离定义为它们的横坐标差的绝对值与纵坐标差的绝对值的和。</p> <p><math>N, M \leq 1e5</math></p>
算法 讨论	<p>最大值是比较好做的，维护已经插入过的点的四种状态的最小值。</p> <p>即 <math>x+y, x-y, -x+y, -x-y</math>，求答案时用询问的点对应的状态减去出现过的最小值，再取个 <math>\max</math> 即可。</p> <p>最小值的维护用到 CDQ 分治。</p> <p>我们把所有操作和询问读进来，原本的点也当作插入操作。</p> <p>然后我们得到了一个按照操作时间顺序的序列，有插入操作和询问操作。</p>	

	<p>注意到所有的询问操作只会被它前面的插入操作影响，我们从这里入手。</p> <p>假设当前要处理的操作序列为<math>[l,r]</math>。</p> <p>我们先二分出中间点 <math>mid</math>。先递归进去子问题<math>[l,mid],[mid+1,r]</math>，然后我们处理前半部分插入对后半部分询问的影响。</p> <p>我们先标记前半部分的询问和后半部分的询问，然后把它们按 <math>x</math> 坐标排序。接着便按 <math>x</math> 坐标顺序小到大做。按照离散化后的 <math>y</math> 坐标建立两个树状数组，维护做到当前询问之前，插入的某个状态的最小值。遇到插入的操作则插入相应的状态值到相应的树状数组，遇到询问则更新该询问的答案。</p> <p>最后还要反过来按 <math>x</math> 坐标从大到小做，同样要用树状数组维护。状态为 <math>(x+y,x-y,-x+y,-x-y)</math>，即为询问点的四个方向上计算曼哈顿距离所需状态。</p>	
其它	时间复杂度： $O(N\log^2N)$	
BOI2012 自选	Peaks	<p>给出一个 <math>n*m</math> 的网格，每个网格有不同的海拔。</p> <p>将所有八连通的，由相同高度网格组成的连通块成为一个平面。把每个不与海拔更高的网格相邻（包括对角线相邻）的平面称为一个峰。可以在网格上相邻格子之间任意移动（包括对角线相邻），问从每个 <math>peak</math> 出发，到达一个更高的峰的路径上，海拔最低的网格海拔的最大值。</p> <p><math>n,m \leq 2000, n*m \leq 100000</math></p>
算法讨论	<p>将所有网格按海拔从大到小排序。然后依次处理每个网格。若一个网格相邻的网格中没有被处理过的，那么它的就是一个峰，否则就令它属于这个峰。</p> <p>若一个网格在处理时同时属于多个峰，那么除了最高的峰以外，其余较低的峰都找到了一条到达更高的峰的路径，同时这个网格就是路径上海拔最小值的最大值。</p> <p>然后对于最高高度的峰，它们显然会有相同的答案，合并起来。为了方便输出答案，可以用一个 <math>vector</math> 具有相同答案的峰存在一起，合并时用启发式合并即可。</p>	
其它	<p>时间复杂度： <math>O(nm\log(nm))</math></p> <p>空间复杂度： <math>O(nm\log(nm))</math></p>	



COCI2013 自选	Slasticar	<p>给出一个长度为 <math>n</math> 的字符串 <math>A</math>，接下来给出 <math>m</math> 个字符串，对于每个字符串 <math>B</math>，用给出的方法与 <math>A</math> 匹配：</p> <p>1. 设 <math>B</math> 的长度为 <math>L</math>，先与 <math>A</math> 的位置为 <math>1 \dots L</math> 的一段进行匹配：先比较 <math>A[1]</math> 和 <math>B[1]</math>，接下来比较 <math>A[2]</math> 和 <math>B[2]</math>，直到比较完 <math>A[L]</math> 和 <math>B[L]</math> 全部匹配或出现一个不相同（即 <math>A[i] \neq B[i]</math>）</p> <p>2. 若匹配失败，则匹配 <math>A</math> 的 <math>2 \dots L+1</math> 和 <math>B</math> 的 <math>1 \dots L</math>，若得不到长度为 <math>L</math> 的串（如：<math>A</math> 的长度为 <math>12</math>，<math>L=6</math>，则 <math>A[8 \dots 13]</math> 与 <math>B[1 \dots 6]</math> 匹配，但是 <math>13 &gt; 12</math>），则在 <math>A</math> 后面补 '#' 直到长度为 <math>L</math>（即变成 'XXXX##'）。以此类推。</p> <p>3. 若匹配成功或 <math>A[\text{len}(A) \dots \text{len}(A)+L-1]</math> 也匹配完，则退出匹配。</p> <p>求每个字符串匹配过程比较次数。</p>
算法讨论	<p>先考虑当前字符串 <math>B</math> 不能匹配成功的情况：</p> <p>对于 <math>A[i \dots i+L-1]</math>，设它与 <math>B</math> 的 LCP 为 <math>\text{lcp}(i)</math>，则答案 <math>\text{Ans} = \sum_{i=1}^n \text{lcp}(i) + 1</math></p> <p>那么我们可以枚举这个 <math>\text{lcp}</math>，然后确定有多少段 <math>A[i \dots i+L-1]</math> 与 <math>B</math> 的 LCP 等于这个值。显然，把 <math>A[i \dots i+L-1]</math> 看成 <math>A</math> 的后缀 <math>\text{Suffix}(i)</math> 不会影响答案，因为现在考虑的是不能匹配成功的情况。所以可以先用后缀数组给 <math>A</math> 的后缀排个序，然后从小到大枚举这个 <math>\text{lcp}</math>，设它是 <math>j</math>，就可以二分出一个区间 <math>l \dots r</math>，使 <math>\text{lcp}(i) \geq j [l \leq i \leq r]</math> 这样时间复杂度就只有 <math>O(L \log n)</math></p> <p>再考虑字符串 <math>B</math> 能匹配成功的情况：</p> <p>同样通过二分，可以确定 <math>\text{lcp} \geq L</math> 的区间，在这个区间里找一个出现位置最前的（即 <math>\text{sa}</math> 的最小值），设它为 <math>p</math>，则答案 <math>\text{Ans} = \sum_{i=1}^n \text{lcp}(i) + 1</math></p> <p>当我们枚举 <math>\text{lcp}</math> 时，确定了一个区间 <math>l \dots r</math>，那么只有 <math>\text{sa}[i] \leq p [l \leq i \leq r]</math> 的才能统计入答案。时间复杂度还是 <math>O(L \log n)</math></p>	
其它		
TC SRM603 自选	Sum of Arrays	<p>给两个数组（告诉你生成方法，其实可以认为是随机的）然后任意排列这两个数组，之后</p> <pre>for(int i = 0; i &lt; n; i++) c[i] = a[i] + b[i];</pre> <p>然后问你 <math>C</math> 数组里面的众数以及这个数出现的次</p>

		数，有多个出现次数最多的数时挑数字大的数。
算法讨论	<p>令 <math>a[i]</math> 表示 <math>i</math> 在数组 <math>A</math> 中出现的次数，<math>b[i]</math> 表示 <math>i</math> 在数组 <math>B</math> 中出现的自出，<math>c[i]</math> 表示 <math>i</math> 最多能在数组 <math>C</math> 中出现的次数。显然，有：</p> $c[i] = \sum \min(a[j], b[i-j]), j \leq i.$ <p>这样直接计算是 <math>O(n^2)</math> 的，显然会超时。因为 <math>c[i]</math> 的计算涉及到取最小值，没有办法化简。考虑转化，<math>c[i]</math> 可以满足下列条件的三元组 <math>(p, q, k)</math> 的数量：<math>p+q=i, a[p] \geq k, b[q] \geq k</math>。</p> <p>把所有三元组按照 <math>k</math> 的大小分成两类，以 <math>L</math> 分隔。当 <math>k</math> 较大的时候，例如 <math>k \geq 10</math>，此时满足 <math>a[p] \geq k, b[q] \geq k</math> 的 <math>p, q</math> 分别不超过 10000 个，暴力枚举所有 <math>p, q</math> 的枚举次数不超过 100000000。因此所有 <math>k \geq L</math> 的三元组可以在 <math>O((n/L)^2)</math> 的时间内计算。当 <math>k</math> 较小的时候，考虑枚举 <math>k</math>。令 <math>x[p] = (a[p] \geq k)</math> (若 <math>a[p] \geq k</math> 则 <math>x[p] = 1</math> 否则为 0), <math>y[p] = (b[p] \geq k)</math> (若 <math>b[p] \geq k</math> 则 <math>y[p] = 1</math> 否则为 0)。令 <math>z[i]</math> 为满足 <math>p+q=i</math>，且 <math>a[p] \geq k, b[q] \geq k</math> 的三元组 <math>(p, q, k)</math> 的数量，显然，有 <math>z[i] = \sum (x[i-j] * y[j])</math>。问题转化为多项式乘法，可以用 FFT 加速。因此所有满足 <math>k &lt; L</math> 的三元组可以在 <math>O(L \log n)</math> 时间内解决。</p> <p>最后算法总的时间复杂度为 <math>O((n/L)^2 + L \log n)</math>。复杂度取决于 <math>L</math> 的大小。</p>	
其它	<p>时间复杂度： <math>O((n/L)^2 + L \log n)</math></p> <p>空间复杂度： <math>O(n)</math></p>	
CEOI2011 自选	Traffic	<p>给出一个平面图，每个点的 <math>X</math> 坐标都在区间 <math>[0, A]</math> 内，每个点的 <math>Y</math> 坐标都在区间 <math>[0, B]</math> 内，没有重点。一些点之间有边，边分为双向边和单向边。要求问每个 <math>X=0</math> 的点能到达的不同的 <math>X=A</math> 的节点个数。</p> <p>平面图有 <math>n</math> 个点，<math>m</math> 条边，给出 <math>A, B</math>，每个点的坐标，每条边连接的两个点和每条边的类型。</p> <p><math>n \leq 300000, m \leq 900000, A, B \leq 10^9</math>。按照 <math>Y</math> 坐标递减的顺序输出每个 <math>X=0</math> 的点能到达的不同的 <math>X=A</math> 的节点个数。时限 8s。</p>

<p>算法讨论</p>	<div data-bbox="491 219 965 779" data-label="Figure"> </div> <p>如图，首先从每个所有 <math>X=0</math> 的点出发做一遍 floodfill，然后把没有任何 <math>X=0</math> 的点能到达的 <math>X=A</math> 的节点删除。把所有 <math>X=A</math> 的节点按照 <math>Y</math> 坐标从小到大排序，因为该图是平面图，所以每个点能到达的 <math>X=A</math> 的节点一定是连续的一段。那么按照 <math>Y</math> 递增的顺序从每个 <math>X=A</math> 的节点出发在反图上做一遍 floodfill，然后按照 <math>Y</math> 递减的顺序从每个 <math>X=A</math> 的节点出发在反图上做一遍 floodfill，记录下两次过程中每个被染色的时间，即可算出每个点可达的 <math>X=A</math> 的节点数。</p> <p>因为这题 <math>n</math> 较大，所以 floodfill 最好用 BFS 实现。BFS 总的时间复杂度为 <math>O(n+m)</math>，排序操作总的时间复杂度为 <math>O(n\log n)</math>，最后总的时间复杂度为 <math>O(m+n\log n)</math>，对于 8s 的时限可以轻松通过。</p>	
<p>其它</p>	<p>时间复杂度： <math>O(m+n\log n)</math></p> <p>空间复杂度： <math>O(n)</math></p>	
<p>CF418E 自选</p>	<p>Tricky Password</p>	<p>要构造一个 <math>n</math> 列无限行的特别表格。第一行的内容被确定，然后其他数字按照如下规则获得：</p> <ul style="list-style-type: none"> <li>处于第 <math>i</math> 行第 <math>p</math> 个位置的数字，等于 <math>a[i-1][p]</math> 在 <math>a[i-1][1... p]</math>中出现的次数。</li> </ul> <p>表格必须能够执行 <math>m</math> 个下面的操作：</p> <ul style="list-style-type: none"> <li>用 <math>v</math> 替代 <math>a[1][p]</math>，并且重新构建表格。</li> <li>找到数字 <math>a[x][y]</math>，以它作为新的密码。</li> </ul> <p><math>n, m \leq 1e5</math></p>

算 法 讨 论	<p>经过观察，可以发现矩阵的第 2 行等于第 4 行，第 3 行等于第 5 行，因此只需要计算前 3 行即可。</p> <p>令 <math>a[x][y]</math> 为矩阵第 <math>x</math> 行第 <math>y</math> 列的数。第一步将第一行所有出现过的值离散化。将 <math>n</math> 个数每 <math>\sqrt{n}</math> 个分成一段，每段建立一个数组 <math>F</math> 和一个数组 <math>G</math>，<math>F[i]</math> 记录数字 <math>i</math> 在这一段即之前的段落中出现的次数，<math>G[j]</math> 记录满 <math>a[2][i]=j</math> 的 <math>i</math> 个数。<math>F, G</math> 的大小均为 <math>O(n)</math>，计算 <math>F, G</math> 的时空复杂度均为 <math>O(n\sqrt{n})</math>。</p> <p>接下来的问题是处理询问。询问第一行直接输出即可，假设询问第二行或第三行的第 <math>i</math> 个数，显然前 <math>i</math> 个数的 <math>F</math> 和 <math>G</math> 可以由预处理的 <math>O(\sqrt{n})</math> 对 <math>F, G</math> 的一对增加 <math>O(\sqrt{n})</math> 个数来得到。显然有 <math>a[2][i]=F[a[1][i]]</math>，<math>a[3][i]=G[a[2][i]]</math>。</p> <p>那么询问操作的时间复杂度为 <math>O(\sqrt{n})</math>。对于修改操作，对于单个个预处理的 <math>F, G</math>，只会影响其中常数项，因此修改时间复杂度也为 <math>O(\sqrt{n})</math>。最后总的时间复杂度为 <math>O((m+n)\sqrt{n})</math>。</p>	
其它	<p>时间复杂度： <math>O((m+n)\sqrt{n})</math></p> <p>空间复杂度： <math>O(n\sqrt{n})</math></p>	
CF 403E 自选	Two Rooted Trees	<p>你有两棵有根树，每棵各有 <math>n</math> 个顶点。让我们用整数 1 到 <math>n</math> 给每棵树的顶点编号。两棵树的根都是顶点 1。第一棵树的边都染成蓝色，第二棵树的边都染成红色。简明起见，我们称第一棵树是蓝色的，以及第二棵树是红色的。</p> <p>同时满足下面的两个条件下，边 <math>(x, y)</math> 有害于边 <math>(p, q)</math>：</p> <ol style="list-style-type: none"> <li>1. 边 <math>(x, y)</math> 的颜色不同于边 <math>(p, q)</math>。</li> <li>2. 考虑与边 <math>(p, q)</math> 颜色相同的树，编号为 <math>x, y</math> 的两个顶点中恰好有且只有一个同时在顶点 <math>p</math> 的子树与顶点 <math>q</math> 的子树里。</li> </ol> <p>你的任务是模拟下述过程。此过程由若干阶段组成：</p> <ol style="list-style-type: none"> <li>1. 在同一个阶段中删除的边颜色相同。</li> <li>2. 在第一阶段，删除恰好一条蓝色的边。</li> <li>3. 假设在阶段 <math>i</math> 我们删去了边 <math>(u_1, v_1)</math>， <math>(u_2,</math></li> </ol>

		<p><math>v_2), \dots, (u_k, v_k)</math>。那么在阶段 <math>i+1</math> 我们要删去有害于<math>(u_1, v_1)</math> 的边, 然后删去有害于<math>(u_2, v_2)</math> 的边, 接着继续删到删完有害于<math>(u_k, v_k)</math> 的边。</p> <p>对于每个阶段, 求解哪些边将在这个阶段中被删除。注意, 有害边的定义只依赖于开始删边之前的初始就拥有的两棵有根树。</p>
算法讨论		<p>原问题可以分成两个相似的子问题: 在蓝树中删除一条边后, 在红树中删除对应的坏边; 在红树中删除一条边后, 在蓝树中删除对应的坏边。首先考虑解决第一个子问题。</p> <p>从根节点 1 开始 dfs 遍历整棵蓝树。定义 <math>in[i], out[i]</math> 分别为 dfs 中进入节点 <math>i</math> 子树的时刻和 dfs 中离开节点 <math>i</math> 子树的时刻。对于红树中的第 <math>i</math> 条边<math>(x[i], y[i])</math>, 定义 <math>u[i] = \min(in[x], in[y]), v[i] = \max(in[x], in[y])</math>, 显然有 <math>u &lt; v</math>。</p> <p>接下来考虑删除蓝树中的一条边<math>(p, q)</math>, 这里令 <math>p</math> 为 <math>q</math> 的父亲。那么<math>(p, q)</math> 对应的红树中的坏边, 即为满足 <math>u[i], v[i]</math> 中有且仅有一个数落在区间 <math>[in[q], ou[q]]</math> 中。</p> <p><math>u, v</math> 中有一个数落在区间 <math>[in[q], ou[q]]</math> 可以分成两种情况:</p> <ol style="list-style-type: none"> <li>1, <math>in[q] \leq u[i] \leq ou[q], v[i] &gt; ou[q]</math>。</li> <li>2, <math>in[q] \leq v[i] \leq ou[q], u[i] &lt; in[q]</math>。</li> </ol> <p>考虑使用数据结构分别处理两种情况。</p> <p>第一棵线段树以 <math>u[i]</math> 为下标建立线段树, 线段树中的每个节点 <math>[L, R]</math> 用线性表存储所有满足 <math>L \leq u[i] \leq R</math> 的 <math>(u[i], v[i])</math>, 节点内按照 <math>v[i]</math> 递增的顺序排列所有边。</p> <p>第二棵线段树以 <math>v[i]</math> 为下标建立线段树, 线段树中的每个节点 <math>[L, R]</math> 用线性表存储所有满足 <math>L \leq v[i] \leq R</math> 的 <math>(u[i], v[i])</math>, 节点内按照 <math>u[i]</math> 递增的顺序排列所有边。</p> <p>然后每个删除蓝树的树边<math>(p, q)</math> 的操作, 在第一棵线段树中找到对应区间 <math>[in[q], ou[q]]</math> 的 <math>O(\log n)</math> 个区间, 删除其中 <math>v[i] &gt; ou[q]</math> 的部分 (等价于删除线性表末尾连续一段), 然后在第二棵线段树中找到对应区间 <math>[in[q], ou[q]]</math>, 删除其中 <math>u[i] &lt; in[q]</math> 的部分 (等价于删除线性表开头连续一段)。</p> <p>每条边在 <math>O(\log n)</math> 个区间内出现, 因此总的空间复杂度为 <math>O(n \log n)</math>。一个区间内一条边一旦被删除就再也不会恢复, 因此所有删除蓝树边对应的坏边的操作的总时间复杂度为 <math>O(n \log n)</math>。所有删除红树边对应的坏边的操</p>

	作同理。因此最后算法总的时空复杂度均为 $O(n\log n)$ 。	
其它	时间复杂度: $O(n\log n)$ 空间复杂度: $O(n\log n)$	
HDU3692 3DGeo	Shade of Hallelujah Mountain	给出一个凸多面体, 一个点光源和一个平面, 多面体和光源在平面的同一侧, 求多面体在平面上的阴影面积, 若无穷大就输出 Inf, 没有就为 0。
算法讨论	把平面 $P: a \cdot x + b \cdot y + c \cdot z = d$ 旋转平移到平面 $z=0$ 。方法有点巧妙, 令 $\text{vec}=(a,b,c) \times (0,0,1)$ , 则 P 绕 vec 做旋转逆时针旋转, 旋转的角度为 $(a,b,c)$ 和 $(0,0,1)$ 之间的夹角, 然后凸多面体上的点和预先选好的平面上的一点做相同的操作, 然后根据预选的平面上的一点的 $z$ 值, 就可以知道要平移多少了。最后就求凸多面体上的点在 $z=0$ 平面上的投影点, 再求这些点的凸包, 计算凸包面积即可。	
其它		
HDU4273 3DGeo	Rescue	给一个三维凸包, 求重心到表面的最短距离。
算法讨论	三维凸包、多边形重心、点面距离。	
其它		
HDU4741 3DGeo	Save Labman	求三维空间异面直线的距离及最近点。
算法讨论	公式见 <a href="http://blog.sina.com.cn/s/blog_a401a1ea0101ij9z.html">http://blog.sina.com.cn/s/blog_a401a1ea0101ij9z.html</a>	
其它		
HDU4617 3DGeo	Weapon	圆形炮筒两端发出的射线构成无限长的圆柱, 若有两个圆柱相交后相切, 则输出 “Lucky” 否则输出圆柱间的最短距离。

算法讨论	将圆柱的中轴线为参考线，利用空间两条直线的距离公式 $\frac{ \vec{AB} \cdot \vec{n} }{ \vec{n} }$ , $\vec{AB}$ 代表两条中轴线的两点构成的向量，这里是两个圆的圆心连线形成的向量; $\vec{n}$ 代表法向量，即公垂线所在的向量,通过两圆的垂线的叉积得到。 $\cdot$ 表示点积。	
其它		
POJ3528 3DGeo	Ultimate Weapon	求三维凸包表面积。
算法讨论	三维凸包。	
其它		