

# Codechef 题目泛做表格

姓名：张云帆（完成情况: 57 + 6 / 90 + 10）

试题名称	题目大意	算法讨论	时空复杂度
Codechef May15 CBAL {1}	<p>平衡字符串定义为每个字符出现次数均为偶数的字符串，一个字符串的 <code>type</code> 权值定义为其所有平衡子串长度的 <code>type</code> 次方之和。现在给定一个长度为 <code>n</code> 的字符串 <code>s</code>，每次给出询问(<code>L,R,type</code>)，表示询问字符串 <code>s[L..R]</code> 的 <code>type</code> 权值，强制在线。</p> <p>其中 <code>n, q ≤ 10<sup>5</sup></code>, <code>type ∈ {0, 1, 2}</code>。</p>	<p>可以发现，我们关心的仅仅是每个字符出现次数的奇偶性，而且字符集大小仅有 <code>26</code>，所以我们状态压缩，记 <code>a[i]</code>表示 <code>s[1..i]</code>所有字符的奇偶性状态，那么子串 <code>s[L..R]</code>是平衡字符串当且仅当 <code>a[L-1]=a[R]</code>。我们对 <code>a</code> 离散化后就可以让其在 <code>[1,n]</code>的范围内。</p> <p>如果没有强制在线，那么我们很容易用莫队算法解决，记录当前范围所有状态的出现位置下标的 <code>0~3</code> 次方之和，利用<math>(a-b)^2=a^2-2ab+b^2</math> 以及 <math>(a-b)^3=a^3-3a^2b+3ab^2-b^3</math> 可以很方便地实现在首尾添加元素。那么这题强制在线，我们用分块算法即可。记录 <code>ans[i][j][type]</code>表示块 <code>i</code>~块 <code>j</code> 的 <code>type</code> 权值，<code>f[i][j][k]</code>表示前 <code>i</code> 个块中权值 <code>j</code> 出现位置下标的 <code>k</code> 次方和，那么查询一个区间我们可以先得得到其完整覆盖的块的答案以及状态，然后用上述方法在块的首尾加入剩下的元素并更新答案即可。</p>	<p>时间复杂度： <math>O(n\sqrt{n})</math></p> <p>空间复杂度： <math>O(n\sqrt{n})</math></p>
Codechef Feb15 DEVLOCK {2}	<p>给定 <code>MM</code> 与 <code>N</code>，对于所有的 <code>0 ≤ M ≤ MM</code>，求出满足各位数字之和不超过 <code>M</code> 且是 <code>P</code> 的倍数的 <code>N</code> 位数的个数（可以有前导 <code>0</code>）。</p> <p>两类数据范围：</p> <p><code>n ≤ 10<sup>9</sup></code>，<code>p ≤ 50</code>，<code>MM ≤ 500</code></p> <p><code>n ≤ 10<sup>9</sup></code>，<code>p ≤ 16</code>，<code>MM ≤ 15000</code></p>	<p>我们对于 <code>10<sup>i</sup> mod p</code> 相等的数位一起算，假设当前计算的是 <code>10<sup>i</sup> mod p=w</code> 的位，且这样的数位有 <code>k</code> 位。设 <code>g[i]</code>表示 <code>k</code> 个数位之和为 <code>i</code> 的方案数，</p> <p>那么显然 <code>g[i]</code>为<math>(\sum_{i=0}^9 x^i)^k</math> 中 <code>x<sup>i</sup></code> 项的系数，而这个可以通过 <code>FFT</code> 快速计算出来。设 <code>f[x][y]</code>表示当前数 <code>mod p=x</code>，且数位之和为 <code>y</code> 的方案数，那么对于这组数位，有 <code>f[i*w mod p][i] = g[i]</code>。于是我们现在得到了最多 <code>P</code> 个这样的数组，现在我们考虑如何合并。</p> <p>我们合并数组 <code>a</code> 和 <code>b</code>，得到新的答案 <code>c[x][y]</code>，显然有 <math>c[x][y] = \sum_{(i+j) \bmod p = x, u+v=y} a[i][u] \cdot b[j][v]</math>，我们发现这是个二维卷积，第一维是循环的而第二维不循环。第一维很小，做 <code>FFT</code> 的意义不大，所以我们对第二维进行优化，将 <code>a[i]</code>和 <code>b[j]</code>这两个数组进行 <code>FFT</code>，每次转移时 <code>O(p<sup>2</sup>)</code>枚举 <code>i</code> 和 <code>j</code>，然后 <code>O(M)</code>进行点积并累加答案，最后再 <code>FFT</code> 逆变换回去即可。</p>	<p>时间复杂度： <math>O(P^3M) + O(P^2M \log_2^m)</math></p> <p>空间复杂度： <math>O(PM)</math></p>
Codechef Oct11 BAKE {3}	<p>有 <code>n</code> 个事件，每个事件可能是出售或者查询。出售会给出产品的编号、大小、省、城市、地区、购买人的性别与年龄和出售数量，其中大小, 城市, 地区可能缺失。询问给出产品的编号、大小、省、城市、地区、购买人的性别与年龄范围，同样某些位置可能缺失表示此处无限制，输出当前在此条件下出售的产品总数。</p> <p>有 <code>10</code> 种产品，每种有 <code>3</code> 种不同的大小，有 <code>10</code> 个省份，每个省份可以划分为 <code>20</code> 个城市，每个城市可以被划分为 <code>5</code> 个地区，事件数 <code>n ≤ 10<sup>5</sup></code>。</p>	<p><code>sum[product_id][size_id][province_id][city_id][region_id][M/F][age]</code>，用这样一个七维数组表示在此限制下销售的产品数量。若某一维为 <code>0</code> 表示在满足其他维限制的情况下，这一维无限制时的销售总数。每次暴力更新、暴力查询即可。</p>	<p>时间复杂度： <math>O(n)</math></p> <p>常数巨大 = =</p>

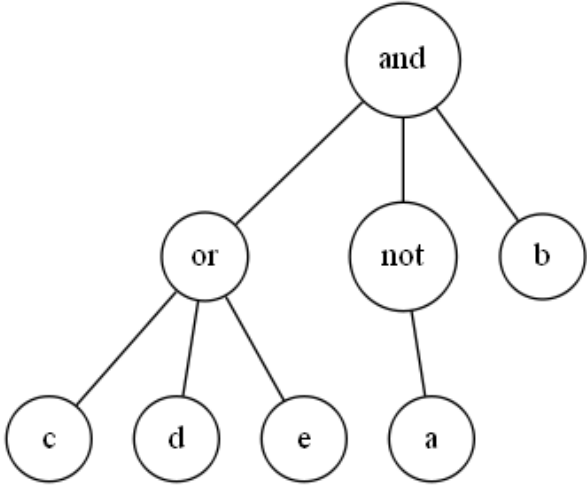
Codechef Jan13 ANDOOR {4}	给定一个矩形区域以及 n 个圆，询问这 n 个圆的并在矩形范围内部分的周长。n≤1000	<p>我们先删去完全在矩形外部以及被某个圆包含的圆，然后我们可以枚举每个圆，求出这个圆在矩形内部且不被其他圆覆盖的周长长度，将这些长度累加即可。</p> <p>对于每个圆，我们算出它被其它圆覆盖的弧的极角区间，同时算出它在矩形内部的极角区间。如果我们直接计算周长，那么要求这些区间的交集，而有些区间跨越了<math>\pi</math>不好处理。所以我们补集转换，求不能计算在周长内的长度，那么就变成了求原区间补集的并集，此时跨越<math>\pi</math>的区间我们拆成两个即可。为了避免精度问题，需要将所有的小数乘以 100 后四舍五入，化为整数后计算。</p>	时间复杂度： $O(n^2 \log n)$  空间复杂度： $O(n)$
Codechef Feb13 QUERY {5}	给定一棵 n 个点的树，要求实现下列三种操作：1. 路径加等差数列；2. 询问路径和；3. 恢复到第 i 次修改后的情况。n, m≤10 <sup>5</sup>	我们对这棵树进行树链剖分，然后用线段树来维护，设标记(v, d)表示加上一个首项为 v，公差为 d 的等差数列，显然这个标记是可以很方便合并的。对于第三种操作的话，我们直接可持久化这棵线段树即可，可以使用标记永久化减少常数。	时间复杂度： $O(n \log^2 n)$  空间复杂度： $O(n \log^2 n)$
Codechef Nov12 COUNTARI {6}	给定一个长度为 n 的数组 A，询问有多少对 (i, j, k)满足：1≤i<j<k≤n 且 A <sub>i</sub> +A <sub>k</sub> =2A <sub>j</sub> 。 n≤100000，A <sub>i</sub> ≤30000	<p>我们对序列分块，枚举每个块，那么有以下三种情况：1. 三个数都在当前块；2. 两个数在当前块，另一个数在前面或后面的块；3. 一个数在当前块，一个数在前面的块，一个数在后面的块。</p> <p>对于情况 1，我们枚举前两个数，同时用一个桶维护后面每个数字的出现次数，直接计算出最后一个数并统计次数；</p> <p>对于情况 2，同上，维护之前和之后的块的数字的出现次数即可；</p> <p>对于情况 3，我们只需求出 A<sub>i</sub>+A<sub>k</sub> 等于每一个值的方案数，然后枚举这个块里的每个数累加答案即可，而这个可以用 FFT 轻松解决。</p>	时间复杂度： $O(n\sqrt{n \log n})$  空间复杂度： $O(n\sqrt{n \log n})$
Codechef Aug15 DISTNUM {7}	给定一个长度为 n 的数列 A，要求实现 m 个操作，操作种类有：  1. 定义 S 为区间[L,R] 中出现过的数字的不可重集合，求 $\sum_{1 \leq i < j < k \leq n} S_i S_j S_k \bmod (10^9 + 7)$ ；  2. 修改一个位置上的数；  3. 删除某个位置上的一个数；  4. 在某个位置上插入一个数；  5. 输出区间[L,R]中不同元素个数。  n, m≤100000	<p>题目中要求所乘的三个数不能重复，所以我们使用容斥原理，设不可重数集为 S，其中所有数的一次方、二次方、三次方和分别为 S<sub>1</sub>, S<sub>2</sub>, S<sub>3</sub>，则询问 1 的答案 <math>Answer = \frac{S_1^3 - 3S_1S_2 + 2S_3}{6}</math>。</p> <p>记 pre[i]表示 i 之前最后一个与 A<sub>i</sub> 相等的数的位置(不存在则为 0)，若查询区间为[L, R]，那么只有满足 i∈[L, R]且 pre[i]&lt;L 的元素 i 才能够统计入答案，我们将(i, pre[i])看做是二维平面上的一个点，询问即为查询矩形内部点权 k 次方和。</p> <p>本题当中出现的插入与删除操作看似很难处理，实际上我们用平衡树模拟操作，预处理出所有数的相对顺序（包括已删除的）后，插入和删除都可以变成修改操作了。同时，注意对询问区间进行相应的修改。</p> <p>至于修改，我们用一个 set 维护每个数出现的位置，然后修改受影响元素对应的点即可。不考虑空间的话，用一个二维树状数组维护即可。但是二维树状数组空间是 n<sup>2</sup> 的显然不行，我们考虑空间换时间，将 n×n 的平面分成若干块，块的大小为 T×T，用树状数组维护完整的块，然后暴力枚举不完整的部分的所有点即可。由于每一行每一列最多同时存在一个点，所以复杂度是 O(T)，注意特殊处理 pre=0 的情况。</p>	设 T 为块的大小  时间复杂度： $O(n \log^2 n + nT)$  空间复杂度： $O(\frac{n^2}{T^2})$

Codechef May12 LEBOXES {8}	<p>有 <math>n</math> 个盒子，对于第 <math>i</math> 个盒子你有 <math>\frac{P_i}{100}</math> 的概率获得 <math>V_i</math> 美元的钱，而有 <math>\frac{100-P_i}{100}</math> 的概率获得一个钻石。有 <math>m</math> 个物品，第 <math>j</math> 个物品需要恰好 <math>C_j</math> 美元的钱和 <math>D_j</math> 个钻石，你一定会买尽可能多的物品，每个物品只能购买一次，求你期望获得的物品数。<math>n, m \leq 30</math></p>	<p>先预处理 <math>cost[count][d\_count]</math> 表示购买恰好 <math>count</math> 个物品，只能使用不超过 <math>d\_count</math> 个钻石最少需要的钱。然后考虑 <code>meet-in-the-middle</code>，先枚举后一半 <math>bn</math> 个盒子的所有状态，记录下 <math>sta[d\_count]</math> 获得钻石数恰好为 <math>d\_count</math> 的状态，然后把这些状态按获得钱的数量排序，并对概率做一遍前缀和。然后我们枚举前一部分 <math>an</math> 个盒子的所有状态，再枚举最终获得了多少个钻石和买了多少个物品，那么此时就可以根据获得钱数的范围在预先记录的数组中二分求得概率和，更新答案即可。</p>	<p>时间复杂度： <math>O(bn \cdot 2^{bn}) + O(m \cdot (bn)^2 \cdot 2^{an})</math></p> <p>空间复杂度： <math>O(nm + 2^{bn})</math></p>
Codechef May12 TICKETS {9}	<p>有 <math>n</math> 道菜肴和 <math>m</math> 位顾客，每位顾客有两道喜爱的菜，每道菜只能供应给一位顾客。现在问你最多能招待多少位顾客，使得无论来的是哪些顾客，所有顾客都能够吃到至少一道他喜欢的菜。<math>n \leq 200, m \leq 500</math></p>	<p>我们建出一个 <math>n</math> 个点 <math>m</math> 条边的无向图，其中每个顶点代表一道菜肴，而每一条边则代表一位顾客，连接着他所喜爱的两道菜所对应的顶点。那么厨师能够卖出 <math>K</math> 张门票的充要条件是：对于无向图中任意大小为 <math>K</math> 的边集 <math>E</math>，与其邻接的点集 <math>V</math> 必定满足 <math>K =  E  \leq  V </math>（我们假设边集连通，因为不同连通块之间是互不影响的，可以分开考虑）。</p> <p>现在我们的任务就是找到最小边集 <math>E</math> 满足 <math> E  &gt;  V </math>，则答案为 <math> E  - 1</math>。可以发现最小边集 <math> E </math> 必然满足 <math> E  =  V  + 1</math>，且点集中无度数为 <math>1</math> 的点。那么这个连通图的结构只有以下三种：</p> <ol style="list-style-type: none"><li>两个度数为 <math>3</math> 的点之间有三条不相交的路径；</li><li>两个不相交的环之间有一条简单路径连接；</li><li>两个环在一个顶点处相交。</li></ol> <p>第一种情况可以直接枚举起点、终点 <code>BFS</code> 求解。后两种情况可以枚举每个点并以它为根求 <code>BFS</code> 树，设 <math>d[i]</math> 表示点 <math>i</math> 在生成树中的深度，那么我们就是要找到两个环 <math>P</math> 和 <math>Q</math> (设 <math>u, v</math> 分别是环 <math>P</math> 和环 <math>Q</math> 中深度最浅的点)，使得 <math>length(P) + d[u] + length(Q) + d[v]</math> 最小。很容易证明，这两个环都必定只有一条边不在生成树中。所以我们枚举所有不在生成树上的边，求出对应环的长度和对应点的深度，更新最大值和次大值即可。</p>	<p>时间复杂度： <math>O(n^3 + n^2m)</math></p> <p>空间复杂度： <math>O(n + m)</math></p>
Codechef May12 CKROACH 【Challenge】 {10}	<p>有 <math>n</math> 只虫子和 <math>m</math> 种杀虫剂，第 <math>j</math> 种杀虫剂的价格为 <math>C_j</math>，使用了第 <math>j</math> 种杀虫剂后，第 <math>i</math> 只虫子有 <math>P_{i,j}\%</math> 的概率死亡。要求选择一种购买杀虫剂的方案，使得每只虫子的死亡概率至少为 <math>90\%</math>，并且花费尽可能少。</p> <p><math>50 \leq n, m \leq 200</math></p>	<p>令 <math>w[i][j] = -\log_{10}(1 - \frac{p[i][j]}{100})</math>，则第 <math>i</math> 只虫子死亡概率至少为 <math>90\%</math> 等价于 <math>\sum_{j \in S} w[i][j] \geq 1</math>，于是问题转化为一个 <math>0/1</math> 背包问题。</p> <p>我们考虑贪心算法，每次根据某种策略购买一种当前还有贡献的杀虫剂，直到所有虫子的死亡概率均不小于 <math>90\%</math>。在这里每次选择性价比 (当前贡献值 / 花费) 最高的杀虫剂即可在 <code>Codechef</code> 上获得 <b>0.781</b> 分。</p> <p>然后我们也可以使用模拟退火等随机化算法，结合上述贪心算法后可以在 <code>Codechef</code> 上获得 <b>0.86</b> 分，且在 <code>Tsinsen</code> 上获得 <b>108</b> 分。</p>	
Codechef Aug13 LYRC {11}	<p>给定 <math>m</math> 个模式串 <math>S_i</math> 与 <math>n</math> 个母串 <math>P_j</math>，每个字符串中只包含大小写字母、数字以及 <code>-</code>，求每个模式串在 <math>n</math> 个母串中出现的次数之和。</p> <p><math>n \leq 500, m \leq 100,  S_i  \leq 5000,  T_j  \leq 50000</math></p>	<p>将模式串插入一棵 <code>trie</code> 树中构建 <code>AC</code> 自动机后，我们对每一个母串进行匹配。假设我们到达了节点 <math>x</math>，那么在 <code>fail</code> 树中 <math>x</math> 所有的祖先都被匹配到了一次。现在我们的操作变为：将某个点 <math>v</math> 到根的路径上所有权值 <math>+1</math>，询问某个节点 <math>x</math> 的权值。这个我们可以直接在节点 <math>v</math> 上打一个标记 <math>+1</math>，最后再询问 <math>x</math> 子树标记和即可。</p>	<p>时间复杂度： <math>O(n S  + m T )</math></p> <p>空间复杂度： <math>O(n S )</math></p>



Codechef Sep14 FIBTREE {12}	给定一棵 n 个节点的树，要求在线实现以下操作：1. 将 x 到 y 的路径上第 k 个节点权值增加 Fib[k]；2. 询问以 x 为根时，子树 y 中所有节点的权值和；3. 询问 x 到 y 路径上所有节点的权值和；4. 将所有节点的权值还原到第 x 个操作后的状态。权值和对 $10^9+9$ 取模，n, m≤100000	由于这棵树的形态不会改变，所以我们考虑树链剖分并用线段树维护 DFS 序，那么询问就是查询区间和，还原权值只需可持久化即可。来看修改操作，我们需要快速支持区间内所有数从左往右/从右往左加上一个前两项为 x 和 y 的 Fib 数列，显然这是可以打标记的并且标记可以很方便合并。我们还需要考虑一个标记对区间的整体效果，也就是我们要快速求出 Fib 数列的前若干项的和。当然可以使用矩阵乘法，但是算上树链剖分后复杂度会达到 $O((\log n)^3)$ 不能接受。假如不考虑取模的话，Fib 数列的通项公式可以用特征方程轻松求出，而这题 5 在模 $10^9+9$ 意义下平方根恰好存在，所以我们能够通过预处理特征根的幂次和后 $O(1)$ 求出 Fib 数列的前若干项的和了。	时间复杂度： $O(n\log^2 n)$  空间复杂度： $O(n\log^2 n)$
Codechef Nov11 LUCKYDAY {13}	定义数列 S： $S[1]=A, S[2]=B, S[i]=(X*S[i-1]+Y*S[i-2]+z) \bmod P$ 其中 P 是一个质数。  现在给定 C 和 Q 个询问[L, R]，回答有多少个 $k \in [L,R]$ 满足 $S[k]=C$ 。  $P \leq 10007, Q \leq 20000, 1 \leq L, R \leq 10^{18}$	我们根据这个线性转移构造转移矩阵 trans 以及初始矩阵 init=[A,B,1]，那么询问转化为有多少个 $k \in [L-1, R-1]$ 满足 $\exists i \in [0, q)$ 使 $init * trans^k = [C,i,1]$ 。于是，我们考虑求出所有 [C,i,1]矩阵第一次出现的位置以及整个数列的周期。  求数列的周期即求最小的 k 使 $init * trans^k = init$ ，这个类似于离散对数，我们使用 BSGS 可以轻松搞定，而求 [C,i,1]的第一次出现位置类似。但是注意 BSGS 必须保证矩阵 trans 可逆，这里 trans 不可逆当且仅当 Y=0，而这种情况下数列变为一阶递推数列，循环节最大为 P 暴力处理即可。  记数列的周期为 len，[C,i,1]的第一次出现位置为 beg[i]。对于每个询问区间，我们把它拆成两个前缀区间相减，对于一个前缀区间[1,n]，有： $Answer = \sum_{beg[i] \leq n} \left( \left\lfloor \frac{n - beg[i]}{len} \right\rfloor + 1 \right)$ $= \sum_{beg[i] \leq n \bmod len} \left( \frac{n - n \bmod len}{len} + 1 \right)$ $+ \sum_{n \bmod len < beg[i] \leq n} \left( \frac{n - n \bmod len - len}{len} + 1 \right)$ $= \sum_{beg[i] \leq n} \left( \left\lfloor \frac{n}{len} \right\rfloor + 1 \right) - \sum_{n \bmod len < beg[i] \leq n} 1$ 于是我们将 beg[]数组与询问从小到大排序后，通过二分即可解决此题，注意常数优化。	时间复杂度： $O(P^{1.5} + Q \log_2 P)$  空间复杂度： $O(P^{1.5} + Q)$
Codechef Aug13 PRIMEDST {14}	给定一棵 n 个节点的树，问在树上随机选择一条路径(u, v), $u \neq v$ ，路径长度为质数的概率是多少。n≤50000	概率即为合法路径数/总路径数，而总路径数固定为 $\binom{n}{2}$ ，所以我们只需求合法路径数。同时可以发现，质数这个限制条件放在距离上是没有任何作用的，所以我们直接求出对于 $i \in [1,n]$ ，所有长度为 i 的路径数。这个很容易想到使用点分治，每一次合并 x 的两棵子树时，我们记 A[i]和 B[i]分别为两棵子树中深度为 i 的节点数，那么有 $Ans[i] = \sum_{k=0} A[k] \cdot B[i-k]$ ，这个显然可以用 FFT 优化。同时，注意 FFT 的复杂度是与两个数组长度和相关的，所以合并时我们需要按照深度从小到大合并。	时间复杂度： $O(n\log^2 n)$  空间复杂度： $O(n)$

Codechef Oct11 PARSIN {15}	<p>给定 n, m, x，求出：</p> $\sum_{k_1+k_2+\dots+k_m=n}\sin(k_1x)\sin(k_2x)\cdots\sin(k_mx)$ <p>m≤30, n≤10<sup>9</sup>, 0≤x≤6.28</p>	<p>考虑 DP，定义：</p> $f[i][j]=\sum_{k_1+k_2+\dots+k_j=i}\sin(k_1x)\sin(k_2x)\cdots\sin(k_jx)$ $g[i][j]=\sum_{k_1+k_2+\dots+k_j=i}\sin(k_1x)\sin(k_2x)\cdots\cos(k_jx)$ <p>根据 sin 和 cos 的和角公式，有：</p> <p>f[i][j]=f[i-1][j-1]*sin(x)+f[i-1][j]*cos(x)+g[i-1][j]*sin(x)</p> <p>g[i][j]=f[i-1][j-1]*cos(x)+g[i-1][j]*cos(x)-f[i-1][j]*sin(x)</p> <p>于是我们用矩阵乘法优化即可。</p>	<p>时间复杂度：</p> $O(m^3\log_2n)$
Codechef Feb13 ROC {16}	<p>给定一个边界水平或竖直的多边形房间，每个 90 度内角的内部格子都站着一个小朋友，每个小朋友的速度均为 1 格每秒。相邻的两个小朋友可以交换位置，多个交换可以同时进行，但每个交换操作只能在两个小朋友之间发生。而且，当交换位置时小朋友必须沿着墙移动。现在给出 Q 个询问，每次询问两个小朋友最短需要多少时间才能相遇。Q≤10000，地图大小≤2500*2500</p>	<p>这是一道模拟题，难点在于处理出小朋友们顺次连边所形成的环的结构。显然这个环是沿着房间的边界的，所以我们任取一个小朋友让他沿着墙逆时针绕房间走一圈即可遍历整个环。</p> <p>具体的话，沿着墙壁逆时针走实际上对应着迷宫问题中常见的右手路径，即墙永远在人右边，所以我们只需不断向前走，直到右边变成空格了就右转，前面有障碍物就一直左转直到前方为空格为止。</p> <p>如何求出最短时间呢？假如小朋友 A 和 D 想要相遇，不妨设他们最后一次移动前所在的位置分别为相邻的小朋友 B 和 C 的位置，那么此时需要时间 time = max(AB, CD) + BC / 2。记 BC 的中点为 M 的话 time = max(AM, MD)，因此 M 越接近 AD 的中点越好。所以我们把环上每条边的中点按顺序存储下来，每次询问在数组中二分即可。注意考虑顺时针和逆时针两种情况，比较方便的写法是把环拉长成 2 倍的链。</p>	<p>时间复杂度：</p> $O(n^2+Q\log_2n)$
Codechef Sep14 QRECT {17}	<p>在二维平面上，需要支持 以下操作：</p> <p>1. 插入一个矩形；2. 删除一个矩形；</p> <p>3. 询问当前有多少个矩形与该矩形有公共点</p> <p>询问数 q≤100000</p>	<p>考虑询问的矩形为(u1,v1,u2,v2)，当前所有矩形的集合为 S，那么与该矩形有公共点的所有矩形(x1,y1,x2,y2)必定满足：</p> $x_1\leq u_2,x_2\geq u_1,y_1\leq v_2\text{且}y_2\geq v_1$ <p>这是一个四维的偏序关系，直接数据结构维护不容易得到好的复杂度，我们考虑补集转化。两个矩形不相交，等价于两维坐标中至少有一维坐标两个矩形无公共点，利用容斥原理我们可以得到与询问矩形不相交的矩形数为：</p> $\sum_S[x_2<u_1\text{或}x_1>u_2]+\sum_S[y_2<v_1\text{或}y_1>v_2]-\sum_S[x_2<u_1\text{或}x_1>u_2][y_2<v_1\text{或}y_1>v_2]$ <p>可以发现 <math>x_2&lt;u_1</math> 与 <math>x_1&gt;u_2</math> 不可能同时成立，<math>y_2&lt;v_1</math> 与 <math>y_1&gt;v_2</math> 同理，所以化简后得：</p> $\sum_S[x_2<u_1]+\sum_S[x_1>u_2]+\sum_S[y_2<v_1]+\sum_S[y_1>v_2]-\sum_S[x_2<u_1][y_2<v_1]-\sum_S[x_1>u_2][y_2<v_1]-\sum_S[x_2<u_1][y_1>v_2]-\sum_S[x_1>u_2][y_1>v_2]$ <p>于是这就变成二维偏序关系了，直接上数据结构维护即可，可采用 CDQ 分治或动态开点二维线段树。</p>	<p>时间复杂度：</p> $O(q\log^2q)$

Codechef Jul14 SEAEQ {18}	<p>两个长度为 <math>n</math> 的数组 <math>A</math>、<math>B</math> 相似，当且仅当 <math>\forall i \in [1, n]</math> 有 <math>C_A(i) = C_B(i)</math>，其中 <math>C_X(i)</math> 表示数组 <math>X</math> 中比 <math>X[i]</math> 小的元素个数。函数 <math>F(A, B, E)</math> 的值等于满足 <math>A[L..R]</math> 与 <math>B[L..R]</math> 相似且 <math>A[L..R]</math> 的逆序对数不超过 <math>E</math> 的二元组 <math>(L, R)</math> 的数目。</p> <p>现在有 <math>T</math> 组询问 <math>(n, E)</math>，求：</p> $\sum_{P \in \text{Perm}(n)} \sum_{Q \in \text{Perm}(n)} F(P, Q, E) \bmod (10^9 + 7)$ <p>其中 <math>\text{Perm}(n)</math> 表示 <math>1..n</math> 的排列的全集。</p> <p><math>n \leq 500, E \leq 10^6, T \leq 10^4</math></p>	<p>令 <math>f[i][j]</math> 表示长度为 <math>i</math> 的排列中逆序对不超过 <math>j</math> 个的排列数，则有 <math>f[i][j] = f[i][j-1] + f[i-1][j] - f[i-1][j-i]</math>，我们先 <b>dp</b> 预处理出 <math>f</math> 数组。</p> <p>可以发现 <math>C_X(i) + 1</math> 即为元素 <math>A[i]</math> 在数组 <math>A</math> 中的排名，令排名数组 <math>\text{rank}[i] = C_X(i) + 10</math>，那么 <math>\text{rank}[]</math> 的逆序对数与 <math>A[]</math> 相等，且 <math>\text{rank[]} \in \text{Perm}(n)</math>，所以我们考虑枚举 <math>\text{rank}</math> 数组。假设相似子区间的长度为 <math>L</math>，那么它的位置有 <math>(n - L + 1)</math> 个，合法的 <math>\text{rank}</math> 数组有 <math>f[L][E]</math> 个，而对应的 <math>A, B</math> 数组在该子区间分别有 <math>\binom{n}{L}</math> 种具体取值，在其它位置又分别有 <math>(n-L)!</math> 种排列方法，所以</p> $\text{Ans} = \sum_{i=1}^n \binom{n}{i}^2 (n-i)!^2 \cdot (n-i+1) \cdot f[i][E]$	<p>时间复杂度： <math>O(n^3 + Tn)</math></p> <p>空间复杂度： <math>O(n^3)</math></p>
Codechef Aug11 SHORTCIR {19}	<p>布尔表达式满足短路运算原理，即某个表达式已经得到结果后就不再计算。现在给出一个布尔表达式和每个变量（只出现一次）为 <b>true</b> 的概率，请你调整布尔表达式的变量顺序，使得新的表达式与原表达式等价且期望计算次数（使用变量的次数）最少。数据保证将表达式化简后与原表达式无实质影响。</p> <p>表达式长度 <math>\leq 30000</math>，变量个数 <math>\leq 1000</math></p>	<p>首先，我们把表达式建成一棵多叉树的结构，可以任意交换的表达式对应节点的父亲相同。</p>  <p>上图表示：( c or d or e ) and ( not a ) and b。</p> <p>题目中合法的调换只可能发生在 ...and...and... 或 ...or...or... 连接的对象，对于每个表达式，我们记录其为 <b>true</b> 的概率 <math>P</math> 和期望计算次数 <math>E</math>。先考虑 <b>and</b> 依次连接的对象 <math>A_1, A_2, A_3, \dots, A_m</math>，若按此顺序计算，则期望计算次数</p> $E = \sum_{i=1}^m (E[A_i] \cdot \prod_{j=1}^{i-1} P[A_j])$ <p>考虑相邻的两项 <math>A_x, A_{x+1}</math> 是否交换顺序，通过化简变形我们可以得出，<math>A_x</math> 在 <math>A_{x+1}</math> 之前当且仅当</p> $\frac{E[A_x]}{1 - P[A_x]} < \frac{E[A_y]}{1 - P[A_y]}$ <p>所以 <b>and</b> 连接的对象我们按照 <math>\frac{E[A]}{1 - P[A]}</math> 从小到大排序即可，<b>or</b> 与之类似。注意处理 <b>andor</b> 和 <b>ornot</b> 这样的变量名。</p>	<p>时间复杂度： <math>O(\text{len})</math></p> <p>空间复杂度： <math>O(\text{len})</math></p> <p>其中 <math>\text{len}</math> 为表达式的长度</p>
Codechef Nov11 STEPAVG 【Challenge】 {20}	<p>给定 <math>n=1000</math> 个数 <math>A[i]</math> 和一个整数 <math>K</math>，每次可以任意删除两个数并加入它们的平均值，操作 <math>n-1</math> 次后会剩下一个数，要求剩下的数尽可能接近 <math>K</math>。</p> <p><math>A[i]</math> 与 <math>K</math> 在 <math>[1, 10^9]</math> 均匀随机。</p>	<p>我们考虑贪心，对于剩余的数，找到最大值 <b>Max</b> 和最小值 <b>Min</b>，若 <math>\text{Mid} = (\text{Max} + \text{Min}) / 2</math> 远远大于 <math>K</math>，则剩下的数整体偏大，此时我们留下 <b>Min</b> 作为最后操作的数，那么我们接下来的目标就是让剩下的数合并的结果尽可能接近 <math>2 \cdot K - \text{Min}</math>。而如果 <b>Mid</b> 远远小于 <math>K</math> 类似，留下最大的数即可。否则的话，我们就合并这两个数。该算法能在 <b>Codechef</b> 上获得 <b>1.000</b> 分。至于如何定义远小于，我们可以定义一个常数 <b>rate</b>，若 <math>\text{Mid} \leq \text{rate} \cdot K</math> 即为远小于，这里可以选择 <math>\text{rate} = 0.999</math>。</p>	<p>时间复杂度： <math>O(n^2)</math></p> <p>空间复杂度： <math>O(n)</math></p>



Codechef Apr12 TSUBSTR {21}	<p>给定一棵 <math>n</math> 个节点的有根树，每个节点上有一个小写字母。一个字符串被这棵树匹配当且仅当存在一条向后代方向移动的路径，使得路径上的字符顺次连接形成的字符串与它相等。求有多少个字符串能够被这棵树匹配。之后有 <math>m</math> 个询问，每次询问给出 26 个字母的大小顺序，求这棵树能表示出的字符串中第 <math>K_i</math> 小的字符串（包括空串）。</p> <p><math>n \leq 250000, m \leq 50000</math></p>	<p>如果树是一条链，显然我们可以用后缀自动机来解决这个问题，维护以每个节点为起点的路径条数，那么第一问的答案就是 <b>root</b> 节点的路径数，之后的询问我们只需依次确定字符串的每个字符即可。那么对于树的情况，其实就是后缀自动机在 <b>Trie</b> 树上的拓展，我们记录 <b>Trie</b> 树上每个节点在后缀自动机中对应的节点，然后边 <b>BFS</b> 边插入即可。</p>	<p>时间复杂度： <math>O(n + output)</math></p> <p>其中 <b>output</b> 为输出总长度</p> <p>空间复杂度： <math>O(n)</math></p>
Codechef Jan15 XRQRS {22}	<p>初始时有一个空序列，接下来有 <math>n</math> 个操作：</p> <p>1. 在序列末尾插入一个数 <math>x</math>；2. 删除序列末尾的 <math>k</math> 个数；3. 询问区间 <math>k</math> 小值；4. 询问区间与 <math>x</math> 的异或值最大的数；5. 询问区间不比 <math>x</math> 大的数的个数。</p> <p><math>n \leq 500000, x \leq 500000</math></p>	<p>除了操作 4，其余操作均为可持久化线段树经典应用，而操作 4 我们可以用可持久化 <b>Trie</b> 树解决。事实上，<b>Trie</b> 树能够实现权值线段树的所有功能，于是我们直接用可持久化 <b>Trie</b> 树即可。</p>	<p>时间复杂度： <math>O(n \log_2 n)</math></p> <p>空间复杂度： <math>O(n \log_2 n)</math></p>
Codechef Mar12 EVILBOOK {23}	<p>有 <math>n</math> 个人，你打败第 <math>i</math> 个人需要付出 <math>C_i</math> 的代价，而打败后可以获得 <math>D_i</math> 的魔法值。你可以对人使用魔法，对第 <math>i</math> 个人每使用一次魔法，它的 <math>C_i</math> 和 <math>D_i</math> 都会除以 3.0，魔法的使用次数无限制但每一次使用要消耗 <math>X</math> 点魔法值。最开始你的魔法值为 0，问你至少要付出多少代价才能使你的魔法值不小于 666。</p> <p>数据组数 <math>\leq 5, n \leq 10, C_i, D_i \leq 10^9</math></p>	<p>首先，设对第 <math>i</math> 个人使用魔法的次数为 <math>K</math>，我们可以观察到一些结论：</p> <p>1. 打败第 <math>i</math> 个人所获得的魔力必然大于对他使用魔法所消耗的魔力，即 <math>\frac{D_i}{3^K} &gt; K \cdot X</math>；</p> <p>2. 令 <math>P</math> 等于满足 <math>\frac{D_i}{3^P} \geq 666</math> 的最大整数，当前最多使用魔法次数为 <math>Q</math>，则 <math>K \geq \min\{P, Q\}</math>；</p> <p>3. 我们可以只在准备打败某个人之前的时刻对他使用魔法；</p> <p>4. 至少存在一种最优方案使得按照打败的先后顺序，对每个人的使用魔法次数单调不减；</p> <p>根据结论 1、2 我们可以算出，对每个人使用魔法的次数范围被限制在一个长度不超过 4 的区间内，而结论 4 可以保证我们同一个状态不会被重复访问，于是我们加上一点可行性剪枝来搜索即可。</p>	
Codechef May14 ANUDTQ {24}	<p>给定一棵 <math>n</math> 个节点的有根树，每个节点上有一个权值，接下来进行 <math>m</math> 次操作：1. 加入一个叶子节点；2. 删除一个子树；3. 给一个子树中的所有节点的权值加上一个值；4. 询问一个子树中的权值和。强制在线。</p> <p><math>n \leq 100000, m \leq 100000</math></p>	<p>如果没有操作 1 和操作 2，显然我们用线段树维护 <b>DFS</b> 序列即可。而我们可以发现，加入叶子节点和删除子树操作后的新 <b>DFS</b> 序列，都只需在原 <b>DFS</b> 序列中进行插入或删除即可得到。于是我们用平衡树来维护 <b>DFS</b> 序列，至于如何提取一棵子树，我们在 <b>DFS</b> 序列中记 <math>+i</math> 表示 <math>i</math> 进栈，<math>-i</math> 表示 <math>i</math> 出栈，那么子树 <math>i</math> 在平衡树中即对应着 <math>+i</math> 与 <math>-i</math> 之间的所有节点。</p>	<p>时间复杂度： <math>O(n \log_2 n)</math></p> <p>空间复杂度： <math>O(n)</math></p>
Codechef Apr15 BWGAME {25}	<p>一个 <math>n \times n</math> 的 0/1 矩阵，第 <math>i</math> 行的第 <math>L_i</math> 至 <math>R_i</math> 个元素为 1，而其余均为 0。现在有两个人在玩游戏，每轮游戏他们同时报出一个从未被报出过的长度为 <math>n</math> 的排列，排列必须保证第 <math>i</math> 行的第 <math>P_i</math> 个元素为 1，且第一个人报出的排列逆序对必须为奇数，第二个人必须为偶数。如果有一轮一个人找不到合法的排列那么另一个人赢，如果两个人同时找不到则为平局，你的任务是判断游戏的胜负。</p> <p><math>n \leq 100000</math></p>	<p>根据行列式的定义我们可以发现，第二个人能找到的合法排列数与第一个人的差值即为给定矩阵对应的行列式的值，于是我们考虑如何快速消元来求行列式的值。</p> <p>设当前考虑的是第 <math>C</math> 列，我们找出所有 <math>L_i = C</math> 的所有行 <math>i</math> 中 <math>R_i</math> 最小的行，设其为行 <math>k</math>（若不存在或多个则行列式的值为 0），然后我们用行 <math>k</math> 来消元，此时除行 <math>k</math> 外所有 <math>L_i = C</math> 的行经过消元后 <math>R_i</math> 不变，而 <math>L_i</math> 将会变为 <math>R_k + 1</math>，这个过程我们用可并堆优化就能够快速消元了。最后我们将得到一个每行每列均只有一个为 1 的元素的一个矩阵，此时模拟一下将其化为主对角线矩阵即可。</p>	<p>时间复杂度： <math>O(n \log_2 n)</math></p> <p>空间复杂度： <math>O(n)</math></p>

Codechef Jan12 CARDSHUF {26}	有 n 张卡片，最开始从上到下编号分别为 1~n，接下来有 m 个操作，每次先取出前 A <sub>i</sub> 张卡片，再取出前 B <sub>i</sub> 张，然后把前 A <sub>i</sub> 张放回再取出前 C <sub>i</sub> 张，接着把 B <sub>i</sub> 张卡片倒序放回，最后再把 C <sub>i</sub> 张卡片放回。问最后 n 张卡片从上到下编号各是多少。n≤100000	直接用最基本的维护区间的平衡树，记录一个翻转标记模拟即可。	时间复杂度： $O(m\log_2 n)$ 空间复杂度： $O(n)$
Codechef Jul12 DGCD {27}	给定一棵 n 个节点的树，每个节点上有一个权值，有 m 次操作：1. 询问两点间路径所有节点权值的最大公约数；2. 给两点间路径上所有点的权值加上一个值。n, m≤50000	可以发现 gcd(a, b, c) = gcd(a, b - a, c - a)，于是我们可以树链剖分后对 DFS 序列差分，维护每个节点的权值和差分后的区间 gcd，这个可以用线段树轻松解决。	时间复杂度： $O(m\log^3 n)$ 空间复杂度： $O(n)$
Codechef Mar14 GERALD07 {28}	给定一个 n 个点 m 条边的无向图，有 q 个询问[L <sub>i</sub> , R <sub>i</sub> ]，输出仅保留第 L <sub>i</sub> ~R <sub>i</sub> 条边时原图有多少个连通块。n, m, q≤200000	考虑离线处理所有询问，我们从小到大枚举询问的右端点 R 并动态加入边，把每条边的编号作为其权值求出当前图中的最大生成森林。那么对于询问[L, R]，如果此时最大生成森林中有 k 条权值大于等于 L 的边有 x 条，则连通块数即为 n - x。动态加边维护最大生成森林是 LCT 经典应用，而求权值大于等于 L 的边的数量可以使用树状数组来解决。	时间复杂度： $O(m\log_2 n)$ 空间复杂度： $O(n)$
Codechef Dec13 QTREE6 {29}	给定一棵 n 个节点的树，每个节点的颜色为黑色或白色，有 m 个操作：1. 修改一个节点的颜色；2. 询问一个点所在连通块的大小，两个节点连通当且仅当它们路径上所有节点的颜色都相同。n, m≤100000	对于每一个节点，我们维护假设其为黑色/白色且只考虑其子树时它所在的连通块大小，那么改变一个节点的颜色仅对它到根的一条链上的节点有影响。若改变点 x 的颜色为 c 后父亲节点与它的颜色相同，我们找到与 fa[x]连通的深度最浅的节点 y，则 fa[x]~fa[y]的这一条路径上连通块 c 的大小都会增加同一个值，且父亲节点的 c xor 1 连通块大小也会变化。若父亲节点颜色与点 x 不相同做法类似。而询问操作我们只需同样找到与 x 连通的最浅点 y，然后询问它的信息即可。这些操作我们使用树链剖分+树状数组便可以高效实现了。	时间复杂度： $O(m\log^2 n)$ 空间复杂度： $O(n)$
Codechef Nov14 FNCS {30}	给定一个长度为 n 的数组和 n 个函数，第 i 个函数返回数组中某一个连续区间的和。接下来有 m 个操作：1. 单点修改数组的值；2. 询问第 L~R 个函数的值的和。n, m≤100000	我们考虑对 n 个函数进行分块，设块的大小为 S。每个块内我们维护当前其所有函数值的和，以及数组中每个元素对这个块函数值的和的贡献系数。那么每次修改操作我们就可以对每个块函数值的和 O(1)进行修改。对于询问，落在完整块内的部分我们维护了它的和，直接 O(1)调用即可。剩余的部分我们对每个函数依次求值，那么现在问题就变为单点修改、询问区间和。如果我们使用树状数组，那么单次询问与单次修改复杂度操作均为 O(logn)，而询问操作数目远多于修改操作导致时间效率不平衡。所以我们对原数组求一遍前缀和，然后问题变为区间修改、单点查询，这个我们用分块便可以做到 O(S+n/S)修改和 O(1)询问了。	时间复杂度： $O(\frac{mn}{S} + mS)$ 空间复杂度： $O(\frac{n^2}{S})$ S 为块的大小，取 $\sqrt{n}$ 时复杂度为 $O(n\sqrt{n})$
Codechef Nov13 MONOPLOY {31}	给定一棵 n 个节点的有根树，初始时每个节点都有一个不同的权值，接下来有 m 个操作：1. 把节点 x 到根的路径上所有节点修改为一个从未出现过的新权值；2. 询问子树 x 中所有节点到根路径上，连接两个不同权值节点的边的数目的平均值。	我们用 LCT 维护这棵树，那么一条实链代表这条链上所有点的权值一样，一条虚边就表示这条边两端权值不同。而可以发现，修改操作其实就相当于 LCT 的 access 操作。每次 access 操作更改 x 到父亲的边的实虚性时（均摊 logn 条），x 子树中的所有点到根的虚边数会对应+1或-1，用线段树维护 DFS 序后就对应着一个区间操作。于是我们用线段树支持区间+1 / -1 以及区间求和，每次 access 操作时对应在线段树上修改即可。	时间复杂度： $O(m\log_2 n)$ 空间复杂度： $O(n)$



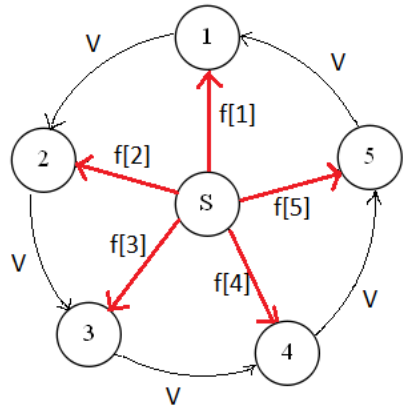
Codechef Oct14 TRIPS {32}	给定一棵 n 个节点的树，树上每一条边的权值为 1 或 2。接下来有 q 个询问(u, v, d)，表示一个人从节点 u 走到节点 v，每天最多走长度 d 的路程，且每一天只能停留在树上的顶点处，求他最少需要走多少天。  n, q≤100000	当 $d \geq \sqrt{n}$ 时，答案是 $O(\sqrt{n})$ 的，所以我们只需预处理出每个节点的深度以及它第 $2^i$ 个父亲的编号，然后把路径拆成两部分，每一天二分求出最远能到达的位置，算出两部分需要走的天数，剩下的一小段特殊判断一下即可。而对于 $d < \sqrt{n}$ 的询问，我们把 $d$ 相同的询问放在一起处理，对于每个点我们预处理出当前从这个点出发向上走 i 天最远能走到哪里，然后通过二分就可以 $O(\log_2 n)$ 处理每个询问了。	时间复杂度：  $O(q\sqrt{n} \log_2 n)$  空间复杂度：  $O(n \log_2 n)$
Codechef Mar15 TREETNT2 {33}	给定一棵 n 个节点的树，每条边上有权值，接下来有 q 个操作，每次操作会修改一条边的权值。所有操作之前以及每次操作之后，输出这棵树中所有权值为 1 的路径条数，路径的权值定义为路径上所有边权值的最大公约数。  n≤100000, q≤100, 边权范围 Z≤1000000	令 f(i)表示树上权值是 i 的倍数的路径条数，根据容斥原理我们可以得出答案为 $\sum_{i=1}^n f(i) \cdot \mu(i)$ 。  假设我们现在在求 f(i)，那么我们只需要考虑所有为 i 的倍数的边。如果没有修改，我们只需用并查集维护联通块的大小，合并两个联通块 A 与 B 时路径条数将会增加 A * B 。那么有修改的话，我们先把所有和修改无关且权值是 i 倍数的边插入并查集，然后每次询问先将并查集还原，然后插入那些这次修改后仍然是 i 倍数的所有边并更新答案即可。为了方便还原，我们采用启发式合并的并查集。	时 间 复 杂 度： $O(Z + qS + 2^c (n + q^2) \log_2 n)$  空间复杂度： $O(2^c n)$  其中 $S = \sum_{i=1}^n  \mu(i) $  c 为[1, Z]中最多质因子个数，在此 c=7
Codechef Aug11 COMPLEXT 【challenge】 {34}	给定一个 n×n 的网格图，每条边的边权是一个复数，求一个生成树，使得各边复数之和的模长最大。  n=20, 数据组数 T=10	假设我们已经知道最终各边求和所得复数的单位方向向量，那么要使模长最大即为使各复数在该单位向量上的投影之和最大，这个用kruskal 求最大生成树即可解决。这题与最小方差生成树很相似，我们考虑同样的思想，我们只要知道生成树的形态便可以得到答案，而 <b>生成树的形态只与各边边权的相对大小有关，而与具体权值无关</b> 。我们考虑两条边的边权 $x_1 + y_1 i$ 与 $x_2 + y_2 i (y_1 < y_2)$ ，若前者的投影比后者小，则方向向量 $(\cos \theta, \sin \theta)$ 必须满足： $x_1 \cos \theta + y_1 \sin \theta > x_2 \cos \theta + y_2 \sin \theta$ $\Leftrightarrow (x_1 - x_2) \cos \theta > (y_1 - y_2) \sin \theta \Leftrightarrow \tan \theta < \frac{x_1 - x_2}{y_1 - y_2}$ $\Leftrightarrow \theta \in (-\frac{\pi}{2}, \arctan \frac{x_1 - x_2}{y_1 - y_2}) \cup (\frac{\pi}{2}, \pi + \arctan \frac{x_1 - x_2}{y_1 - y_2})$ 我们枚举每一对边，算出它们投影相等时的分界点，然后这些分界点会把 $[-2\pi, 2\pi)$ 的辐角区间分成若干小区间，在每个区间内部所有边的 <b>相对大小关系不变</b> 。于是我们枚举每个区间，在这个区间里任取一个方向向量做一遍最大生成树，最后取最优值即可。由于这题时限较紧，我们可以只选出一部分区间处理。	时间复杂度：  $O(Tn^6 \log_2 n)$  空间复杂度：  $O(n^2)$
Codechef Apr12 CONNECT {35}	给定一个 n×m 的网格，每个格子有权值和代价，权值范围为[-1, n×m]。求一个代价和最小的联通块，使得联通块中不存在权值为-1 的格子且至少出现了 k 种不同的正权值。n, m≤15, k≤7	如果权值范围恰好为[-1, k]，那么这就是一个经典的斯坦纳树问题。那么对于 n×m 个权值，我们考虑把它们随机映射到[1, k]的范围内，然后再进行斯坦纳树求解。可以发现我们这么做答案肯定只可能比最优答案劣且一定合法，而且只要这个映射满足最优答案中的 k 种颜色在映射中对应的新颜色互不相同，我们就必定可以求出最优解。可以算出单次随机正确概率为 $\frac{k!k^{nm-k}}{k^{nm}} = \frac{k!}{k^k}$ ，于是我们进行大概 600~700 次随机就能基本保证正确率了。	时间复杂度：  $O(Tnm \cdot 3^k + T \cdot 2^k \cdot SSSP(nm, nm))$  空间复杂度：  $O(nm)$  其中 T 为随机次数，SSSP(V, E)为单源最短路复杂度

Codechef Jul14 GNUM {36}	<p>给定两个长度为 n 的数组 A 和 B，现在有两个初始为空的二元组集合 S<sub>1</sub>、S<sub>2</sub>，每次操作你需要选择两个数对(i, j)和(p, q)，满足：</p> <ol style="list-style-type: none"><li>(i, j) ∉ S<sub>1</sub>, (p, q) ∉ S<sub>2</sub>；</li><li>A<sub>i</sub>&lt;B<sub>j</sub>, A<sub>p</sub>&gt;B<sub>q</sub>, gcd(A<sub>i</sub>, B<sub>j</sub>)&gt;1, gcd(A<sub>p</sub>, B<sub>q</sub>)&gt;1；</li><li>gcd(A<sub>i</sub>, B<sub>j</sub>)与 gcd(A<sub>p</sub>, B<sub>q</sub>)不互质；</li></ol> <p>然后把这两个数对分别加入集合 S<sub>1</sub>、S<sub>2</sub> 中。现在问你最多能够进行几次操作。</p> <p>n≤400, 1≤A<sub>i</sub>, B<sub>j</sub>≤10<sup>9</sup></p>	<p>我们可以发现，两个数对(i, j)、(p, q)能否加入集合与当前 S<sub>1</sub>、S<sub>2</sub> 集合的状态无关，只与 A、B 两个数组本身有关。那么我们先处理出所有可能的数对(i, j)和(p, q)，把能够同时加入集合的数对之间连边，那么答案就是这个二分图的最大匹配，我们使用匈牙利算法或最大流即可。但这样的二分图点数为 n<sup>2</sup>，边数为 n<sup>4</sup>，我们考虑减少连边。实际上，两个数不互质等价于两个数有公共的质因子，于是我们把每个数对(i, j)向 gcd(A[i], B[j])的所有质因子连边，gcd(A[p], B[q])的所有质因子向数对(p, q)连边，可以发现新图和原图等价。设 K 为所有数中最多的质因子个数，本题权值范围为 10<sup>9</sup> 则 K≤7，所以边数缩减为 Kn<sup>2</sup>，而点数只会增加 Kn 个（gcd 不会增加新的质因子），于是便可以通过此题了。</p>	<p>时间复杂度： <math>O(\text{maxflow}(V, E))</math></p> <p>空间复杂度： <math>O(E)</math></p> <p><math>V = n^2</math> <math>E = K \cdot n^2</math></p>
Codechef Nov12 MARTARTS {37}	<p>一个 2×n 的完全带权二分图，每条边有两个权值 A<sub>i,j</sub> 和 B<sub>i,j</sub>，你需要对这个二分图进行完备匹配。令所有匹配边的权值 A 总和为 H，权值 B 总和为 G，你的对手会在知道了你的匹配后选择是否去掉一条匹配边，使得其权值不计入 H 和 G 中。对手的目标是最大化 G - H，其次最大化 G，而你的目标是最大化 H - G，其次最大化 H，求最优匹配。</p> <p>n≤100, A<sub>i,j</sub>, B<sub>i,j</sub>≤10<sup>12</sup></p>	<p>定义每条边的价值 W<sub>i,j</sub> = A<sub>i,j</sub> - B<sub>i,j</sub>，那么我们现在的任务就是最大化价值和，其次最大化 H，而对手的任务是最小化价值和，其次最大化 H，因为当价值和确定后最大化 H 和最大化 G 等价。</p> <p>先考虑如何最大化价值和，对手会删掉一条边当且仅当它的价值为正且是所有匹配边中最大的。于是我们从小到大枚举对手删掉的边，强制匹配这条边后算出此时最优匹配和最终获得价值，然后再将这条边插入二分图即可。我们可以把强制匹配、插入边这两个操作都看成是改变边的权值，设改变的边为(u,v)，那么我们令 wx[u]=max{ g[u][i] - wy[i] }，其中 wx[]和 wy[]为 KM 算法中的顶标，而 g[u][i]表示边(u,i)的权值。然后我们把点 u 的匹配删除，再从点 u 开始重新增广便能高效维护最大匹配了。</p> <p>至于双关键字我们可以用 pair 类型并重载运算符来方便实现，注意当价值相同时对手会删除 A<sub>i,j</sub> 更小的那条。</p>	<p>时间复杂度： <math>O(n^4)</math></p> <p>空间复杂度： <math>O(n^2)</math></p>
Codechef Jan12 MISINT2 {38}	<p>求出长度在[L, R]范围内，满足将所有偶数位字符取出放在串首，再与所有奇数位字符顺次拼接后得到的新串与原串相等的小写字符串数目，答案对 10<sup>9</sup>+7 取模。</p> <p>L, R≤10<sup>10</sup>, R - L≤50000</p>	<p>我们可以发现重排就对应着一个置换，若长度为 n 的置换循环节数为 G(n)，那么最终答案即为 <math>\sum_{i=L}^R 2^{G(i)}</math>。当 n 为奇数时最后一个位置不变，此时 G(n)=G(n-1)+1，所以我们只考虑 n 为偶数的情况。</p> <p>当 n 为偶数时，元素 i 置换后对应着元素 2i mod (n+1)，那么令 ord(x)表示 2 对模 x 的阶，可以证明满足 <math>x \cdot 2^d \equiv x(\text{mod } n+1)</math> 的最小正整数 d 为 <math>\text{ord}(\frac{n+1}{\text{gcd}(x, n+1)})</math>，所以满足 gcd(x,n+1)=K 的所有元素构成了 <math>\varphi(\frac{n}{K}) / \text{ord}(\frac{n}{K})</math> 个循环节，那么总循环节数 <math>G(n) = \sum_{d (n+1), d \neq 1} \frac{\varphi(d)}{\text{ord}(d)}</math>。可以证明，当 gcd(a,b)=1 时 ord(ab)=lcm(ord(a),ord(b))，而 phi(n)又是积性函数，所以我们通过把 n 质因子分解后就能计算 ord(n)了，为了加快速度，我们可以预处理一部分 ord(n)。</p> <p>至于质因子分解，由于待分解的数是一个连续的区间，所以我们可以枚举所有质数，将其分配给这个区间内它的倍数即可，枚举约数也只需我们 dfs 每个质因子的次数即可。注意常数。</p>	<p>时间复杂度： <math>O(\sqrt{R} \log^2 R + T \cdot D \cdot (-\frac{\sqrt{R}}{\log R} + \log^2 R))</math></p> <p>空间复杂度： <math>O(\sqrt{n} \log n)</math></p> <p>T 为数据组数 D = R - L + 1</p>

Codechef Jun15 CHEFBOOK {39}	<p>你需要给出 <math>2n</math> 个整数 <math>P[i]</math>和 <math>Q[i]</math>，使得它们满足 <math>m</math> 个限制，每个限制 <math>(a_k, b_k, W_k, L_k, R_k)</math> 表示：<math>L_k \leq P[a_k]-Q[b_k]+W_k \leq R_k</math>，并且最大化 <math>\sum_{k=1}^m P[a_k]-Q[b_k]+W[k]</math>，同时要求 <math>P[i], Q[i] \in [0,10^6]</math>。输出最大值和对应的方案。</p> <p><math>n \leq 100, m \leq n^2</math></p>	<p>我们令 <math>P[i]=x[i]</math>, <math>Q[i]=x[n+i]</math>，那么可以发现这题对应着一个线性规划模型。设 <math>A_{2m \times 2n}</math> 表示限制条件的系数矩阵，而 <math>B_{1 \times 2m}</math> 和 <math>C_{1 \times 2n}</math> 分别表示不等号右边的常数构成的向量以及目标函数的系数矩阵，那么这个线性规划可以表示为：</p> <p>最大化 <math>C^T x</math> 满足约束 <math>Ax \leq B, x \geq 0</math></p> <p>而矩阵 <math>A</math> 满足每行的恰好有一个+1 和一个-1，而我们知道如果是每列有一个+1 一个-1 的话就是一个网络流的问题了，把行转列我们联想到把原问题转化为它的对偶问题：</p> <p>最小化 <math>B^T y</math> 满足约束 <math>A^T y \geq C, y \geq 0</math></p> <p>同时由于 <math>A^T</math> 每列恰有一个+1 和一个-1，且 <math>\sum_{i=1}^{2n} C[i]=0</math>，我们将所有的不等式相加后会得到 <math>0 \geq 0</math>。而这个不等式只能在取等号时成立，所以每个不等式都只能在取等号时成立，于是问题化为：</p> <p>最小化 <math>B^T y</math> 满足约束 <math>A^T y = C, y \geq 0</math></p> <p>这就是用流量平衡条件将线性规划转化成网络流的经典模型了，我们使用最小费用最大流算法便可解决第一问，并且能够得到 <math>y[]</math>的方案。那么如何输出 <math>x[]</math>的方案呢？</p> <p>互补松弛定理：（设变量数为<math>n</math>，限制数为<math>m</math>） 令<math>\bar{x}</math>和<math>\bar{y}</math>分别为原问题和对偶问题的一组可行解，那么<math>\bar{x}</math>和<math>\bar{y}</math>均为最优解的充要条件是：</p> <p><math>\sum_{i=1}^m A_{ij} \bar{y}_i = C_j</math> 或 <math>\bar{x}_j = 0, \quad \forall j \in [1, n]</math> 且</p> <p><math>\sum_{i=1}^m A_{ij} \bar{x}_j = B_i</math> 或 <math>\bar{y}_i = 0, \quad \forall i \in [1, m]</math></p> <p>于是我们根据互补松弛定理，将不为 <math>0</math> 的变量 <math>y_k</math>对应的限制条件加入差分约束系统，并且将原限制条件也加入差分约束系统，然后通过最短路算法便能找到一组最优解了。至于 <math>P[i], Q[i] \in [0,10^6]</math>，我们可以把所有的 <math>P[i], Q[i]</math>均加上或减去一个数，显然是不会破坏限制条件的。</p>	<p>时间复杂度： <math>O(\text{costflow}(n, m))</math></p> <p>空间复杂度： <math>O(n + m)</math></p> <p><math>\text{costflow}(V, E)</math>为 <math>V</math> 个点 <math>E</math> 条边的最小费用最大流复杂度</p>
Codechef Feb12 FLYDIST {40}	<p>给定一个 <math>n</math> 个点 <math>m</math> 条边的简单无向图，第 <math>i</math> 条边的边权为 <math>w_i</math>，两端点分别为 <math>u_i</math> 和 <math>v_i</math>。现在你可以将每条边的权值增加或减少 <math>D_i</math>，最后使得每条边的长度恰好为这条边两端点之间的最短路长度，且每条边的权值为正数。</p> <p>求 <math>\sum_{i=1}^m D_i</math> 的最小值，结果用分数表示。</p> <p><math>n \leq 10, W_i \in \mathbb{Z}</math></p>	<p>对于每条边我们建立两个变量 <math>d_i^+</math>和 <math>d_i^-</math>分别表示第 <math>i</math> 条边增加和减少的长度，然后令变量 <math>G_{i,j}</math> 表示 <math>i</math> 与 <math>j</math> 之间的最短路，那么我们可以得到这样一个线性规划：</p> <p><math>\forall k \in [1, m], w_k + d_k^+ - d_k^- = G_{u[k], v[k]}</math> <math>\forall i, j \in [1, n], G_{i,i} = 0, G_{i,j} = G_{j,i}</math> <math>\forall x, y \in [1, n], \forall u[k] = y, G_{x, v[k]} + w_k + d_k^+ - d_k^- \geq G_{x,y}</math> <math>\forall x, y \in [1, n], \forall v[k] = y, G_{x, u[k]} + w_k + d_k^+ - d_k^- \geq G_{x,y}</math></p> <p>所有变量 <math>\geq 0</math>，最小化 <math>\sum_{i=1}^m d_i^+ + d_i^-</math></p> <p>将边的长度减少到 <math>0</math> 没有意义，所以我们不用担心边权恰好为 <math>0</math> 的情况，那么这题直接使用单纯形算法即可。运算过程中我们用实数进行计算，最后再枚举分母将答案化为分数。</p>	<p>时间复杂度： <math>O(\text{simplex}(N, M))</math></p> <p>空间复杂度： <math>O(NM)</math></p> <p><math>N = n^2 + 2m</math> <math>M = nm + m</math></p>



Codechef Mar12 CIELQUAK {41}	<p>给定一个 <math>n \times m</math> 的网格图，两个顶点之间存在边当且仅当两个点的曼哈顿距离为 1，每条边都有 <math>p</math> 的概率被摧毁，问点(1, 1)与点(n, m)联通的概率是多少。</p> <p>数据组数 <math>T \leq 50, n \leq 8, m \leq 10^{18}</math></p>	<p>如果 <math>m</math> 不大，我们显然可以用按格转移连通性 DP 来解决此题，记录轮廓线上每个点之间的连通性以及和(1,1)的连通性即可，经搜索得出状态数 <math>S</math> 最多为 3432，预处理所有转移便可以在 <math>O(nmS)</math>的复杂度求出答案了。而当 <math>m</math> 很大时，可以证明 <math>\text{Ans}(n,m,p) / \text{Ans}(n,m-1,p)</math>会趋近于一个常数，于是我们选择一个足够大的 <math>m'</math>，用 <math>\text{Ans}(n,m',p) \cdot (\frac{\text{Ans}(n,m'+1,p)}{\text{Ans}(n,m',p)})^{m-m'}</math> 得到答案。</p> <p>此题 <math>m'</math>取到 40~50 即可满足精度，注意常数。</p>	<p>时间复杂度： <math>O(Tnm'S)</math></p> <p>空间复杂度： <math>O(nmS)</math></p>
Codechef Jun14 TWOCOMP {42}	<p>给定一棵 <math>n</math> 个节点的树以及两个链集合 A 和 B，每条链有一定的收益。现在要求分别从 A 和 B 中选择若干条链，使得 A 中的任意一条链与 B 中的任意一条链在树上均不相交，并且最大化所选链的总收益。</p> <p><math>n \leq 100000,  A ,  B  \leq 700</math></p>	<p>我们将集合 A 中的链视为左点集，集合 B 中的链视为右点集，那么把相交的链 <math>A_i</math> 和 <math>B_j</math> 连边后这题就对应着二分图最大点权独立集模型，使用最大流算法便可解决。</p> <p>至于如何判断两条链相交，可以证明链(x,y)与链(u,v)相交当且仅当点 lca(x,y)在链(u,v)上或者点 lca(u,v)在(x,y)上。而点 <math>z</math> 是否在链(u,v)上我们可以通过判断 lca(u,z)和 lca(v,z)来实现。由于查询次数较多，我们使用 RMQ 算法来实现 <math>O(1)</math>求两个点的 lca。</p>	<p>时间复杂度： <math>O(n \log_2 n) + O(\text{maxflow}(V, E))</math> <math>V =  A  +  B </math>, <math>E =  A  B </math></p> <p>空间复杂度： <math>O(n +  A  B )</math></p>
Codechef Jun11 CLONES {43}	<p>定义形为 <math>f: A \rightarrow B</math> 的函数为布尔函数，其中 A 是一个长度为 <math>n</math> 的 0/1 数列，<math>B = \{0, 1\}</math>，此时我们称 <math>n</math> 为布尔函数的项数。现在有四个元素是 <math>n</math> 项布尔函数的集合：</p> <p>1.Z 集合是所有满足 <math>f(0, 0, \dots, 0) = 0</math> 的集合；</p> <p>2.P 集合是所有满足 <math>f(1, 1, \dots, 1) = 1</math> 的集合；</p> <p>3.D 是满足 <math>f(x_1, x_2, \dots, x_n) = f(!x_1, !x_2, \dots, !x_n)</math> 的集合；</p> <p>4.A 集合是满足如下条件的集合：</p> <p>如果存在 <math>x</math> 使得 <math>f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)</math>成立，</p> <p>那么对任意 <math>y</math> 都有 <math>f(y_1, \dots, y_{i-1}, 0, y_{i+1}, \dots, y_n) = f(y_1, \dots, y_{i-1}, 1, y_{i+1}, \dots, y_n)</math>成立。</p> <p>现在给出一个由 Z, P, D, A, <math>\vee</math>, <math>\wedge</math>, <math>!</math>, <math>(, ), \backslash</math>组成的合法表达式 <math>\text{expr}</math>，其中 <math>\vee, \wedge, \backslash, !</math> 分别表示并集、交集、差集和对全集的补集。其中 <math>()</math> 优先级最高，其次是 <math>!</math>，剩下三个优先级相同。这个表达式的结果是一个以函数为元素的集合，求出这个集合的元素个数，答案对 <math>P = 1000003</math> 取模。</p> <p><math>n \leq 100,  \text{expr}  \leq 100</math></p>	<p>对于每个函数，我们可以用一个 4 位二进制数来表示它是否属于这四个集合。然后我们通过分类讨论，分别求出 16 个二进制数对应的函数的数量（具体参见 Tsinsen 出题人题解）。接着在表达式计算的过程中，我们用一个 16 位的二进制数表示当前集合还存在哪些类型的函数，那么并集运算 <math>A \vee B</math> 相当于 <math>a \text{ or } b</math>，交集运算 <math>A \wedge B</math> 相当于 <math>a \text{ and } b</math>，取反操作 <math>!A</math> 相当于 <math>a \text{ xor } 65535</math>，而差集操作 <math>A \backslash B</math> 相当于 <math>a \text{ and } (b \text{ xor } 65535)</math>，于是这题就变成一个表达式求值的题了，最后我们再将二进制数对应的函数类型的数量累加即可。</p>	<p>时间复杂度： <math>O(\log_2 P +  \text{expr} )</math></p> <p>空间复杂度： <math>O( \text{expr} )</math></p>
Codechef Jun13 TKCONVEX {44}	<p>给定 <math>n</math> 根棍子的长度和整数 <math>k</math>，判断是否能在其中选出 <math>2k</math> 根棍子拼成两个凸多边形，使得两个凸多边形都恰好由 <math>k</math> 根棍子组成，且相邻两条边不能平行（即面积大于 0）。</p> <p><math>n \leq 1000, 3 \leq k \leq 10</math></p>	<p>一组木棍能够拼成凸多边形的条件是最长边小于其余各边的边长之和。如果我们给定 <math>n</math> 根棍子要求拼成凸多边形且一定存在解，那么把棍子按长度排序后，肯定存在一组方案使得选取的木棍是一个连续的区间。</p> <p>所以我们分情况讨论：1.两个凸多边形的选取区间不相交，此时我们直接贪心找到最左和最右的合法区间，再判断它们是否相交即可；2. 两个区间相交，那么此时选取的一定是一个长度为 <math>2k</math> 的区间。我们枚举所有这样的区间，暴力枚举两个多边形分别由区间内哪些边构成，再判断是否合法即可。</p>	<p>时间复杂度： <math>O(n \cdot \binom{2k-1}{k-1})</math></p> <p>空间复杂度： <math>O(n)</math></p>

<div>Codechef Mar13</div> <div>LECOINS</div> <div>{45}</div>	<div>给定 <math>n</math> 个有面值 <math>v</math> 和颜色 <math>c</math> 的硬币，每个硬币有无穷个。接下来有 <math>Q</math> 个询问，每次给定一个金额 <math>S</math>，你需要选出一些硬币使得其面值之和为 <math>S</math>，且要求硬币中的颜色种类数最少。<math>V, Q \leq 200000, n \leq 30, S \leq 10^{18}</math></div>	<div>如果只需要判断是否存在一种选取硬币的方案，我们可以选择任意一枚硬币，设其面值为 <math>D</math>，那么接下来我们在模 <math>D</math> 的意义下 <math>dp</math>。设 <math>f[x]</math> 表示其余的硬币能够拼出且模 <math>D</math> 为 <math>x</math> 的最小面值，那么询问只需判断 <math>S</math> 与 <math>f[S \bmod D]</math> 的大小关系即可。具体的话，设当前我们加入第 <math>i</math> 枚硬币，则 <math>f'[x] = \min\{f[(x - v[i] * k) \bmod D] \mid k \geq 0\}</math>。直接 <math>DP</math> 复杂度是 <math>O(Vn^2)</math> 比较高。但可以发现，这个 <math>DP</math> 的转移图是若干个互不影响的环，而对于每个环，其 <math>DP</math> 过程类似于求这样一个图的单源最短路：</div> <div></div> <div>我们找到 <math>f</math> 值最小的点 <math>i</math>，那么 <math>S</math> 到 <math>i</math> 的最短路必定是直接走红边 <math>S \rightarrow i</math>，那么我们删除连向 <math>i</math> 的黑边之后该图变为一个拓扑图，直接递推即可，时间复杂度降为 <math>O(Vn)</math>。至于最小化颜色种类数，我们只需在状态中再加一维，即设 <math>f[cor][i]</math> 表示用 <math>cor</math> 种颜色的硬币，能拼出的模 <math>D</math> 为 <math>i</math> 的最小面值就行了。注意处理硬币颜色相同的情况，可以在状态中加入一维 <math>0/1</math> 表示当前颜色是否已经选过或者按颜色分段 <math>dp</math>。</div>	<div>时间复杂度： <math>O(Vn^2 + Qn)</math></div> <div>空间复杂度： <math>O(Vn)</math></div>
<div>Codechef Dec11</div> <div>SHORT2</div> <div>{46}</div>	<div>给定一个质数 <math>p</math>，问存在多少对 <math>(u, v)</math> 满足 <math>u &gt; p, v &gt; p</math> 且 <math>(u - p)(v - p) \mid uv</math>。</div> <div>数据组数 <math>\leq 5, p \leq 10^{12}</math></div>	<div>令 <math>a = u - p, b = v - p</math>，则原条件等价于：<math>a, b &gt; 0</math> 且 <math>ab \mid (a + p)(b + p)</math>，即 <math>ab \mid p(a + b + p)</math>。</div> <div>情况 1: <math>p \mid a</math> 且 <math>p \mid b</math>，此时设 <math>a = pm, b = pn</math>，则有 <math>p^2 mn \mid p^2(m + n + 1)</math>，即 <math>mn \mid m + n + 1</math>，此时满足条件的 <math>(m, n)</math> 只有 5 对：(1,1), (1,2), (2,1), (2,3), (3,2)；</div> <div>情况 2: <math>p \nmid a</math> 且 <math>p \nmid b</math>，此时 <math>ab \mid p(a + b + p)</math> 等价于 <math>ab \mid (a + b + p)</math>。若 <math>a = b</math>，则 <math>a^2 \mid a(2a + p)</math> 只有当 <math>a = b = 1</math> 时成立。否则设 <math>a &lt; b</math>，则 <math>a + b + p \geq ab</math>，即 <math>(a - 1)(b - 1) \leq p + 1</math>，那么有 <math>a \leq \sqrt{p + 1} + 1</math>。令 <math>a + b + p = kab</math>，则 <math>b = \frac{a + p}{ka - 1}</math>，设 <math>d = ka - 1</math>。当 <math>d \leq b</math> 时显然有 <math>d \leq \sqrt{\sqrt{p + 1} + p + 1}</math>，我们枚举所有 <math>d</math>，那么对应的 <math>a</math> 要满足 <math>d \mid a + p</math> 且 <math>a \mid d + 1</math>，则 <math>a \bmod d = (-p) \bmod d</math> 且 <math>a \leq d + 1</math>，可以发现此时合法的 <math>a</math> 最多两个。而当 <math>d &gt; b</math> 时，我们枚举所有的 <math>b</math>，此时合法的 <math>a</math> 需要满足 <math>b \mid a + p</math>，则 <math>a \bmod b = (-p) \bmod b</math> 且 <math>a &lt; b</math>，显然最多一个；</div> <div>情况 3: <math>p \mid a</math> 且 <math>p \nmid b</math>，令 <math>a = pm, b = n</math> 则条件化为 <math>pmn \mid p(pm + n + p)</math>，即 <math>mn \mid (pm + n + p)</math>。我们设 <math>pm + n + p = kmn</math>，则 <math>m = \frac{n + p}{kn - p}</math>，令 <math>d = kn - p</math> 那么 <math>d \mid n + p</math> 且 <math>n \mid d + p</math>，又因为 <math>p \nmid n</math> 所以上式等价于 <math>p \nmid d, p \nmid n</math> 且 <math>dn \mid (n + p)(d + p)</math>，这与情况 2 完全相同，故方案数也相等。</div> <div>情况 4: <math>p \mid b</math> 且 <math>p \nmid a</math>，与情况 3 完全相同。</div>	<div>时间复杂度： <math>O(\sqrt{p})</math></div> <div>空间复杂度： <math>O(1)</math></div>

Codechef Jun12 MATCH {47}	<p>给定一个二分图，X 部有 n 个点，Y 部有 m 个点，左边第 i 个点和右边第 j 个点之间有连边的概率为 <math>w[i][j]</math>，求这个二分图最大匹配的期望大小。 <math>n \leq 5, m \leq 100</math></p>	<p>考虑状压 DP，设 <math>f[i][sta]</math> 表示当前加入了 Y 部的前 i 个点，X 部所有点集是否能被完全匹配的情况为 sta 的概率，显然根据 sta 我们能够得到此时的最大匹配。每次我们加入一个 Y 部中的新点时，枚举它与 X 部的所有连边情况，算出这种情况发生的概率以及此时新的 sta，然后转移即可。至于新的 sta 如何计算，我们枚举 Y 部新加入的点匹配 X 部的哪个点就行了。</p> <p>点集数为 <math>2^n</math>，那么状态数为 <math>2^{2^n}</math> 过多。但可以发现如果集合 T 能够被完全匹配，那么 T 的所有子集必定也能被完全匹配，所以实际状态数并不多。所以我们预处理出所有可能的状态，直接 DP 就可以了。</p>	<p>时间复杂度： <math>O(4^n \cdot K) + O(m \cdot 2^n \cdot K)</math> K 为状态总数</p> <p>空间复杂度： <math>O(K \cdot (m + 2^n))</math></p>
Codechef Aug11 DIVISORS {48}	<p>给定正整数 B 和 X，求满足满足以下条件的正整数 N 的数目： <math>\exists D \in (N, B]</math> 使得 <math>D   N \cdot X</math>。</p> <p>数据组数 <math>\leq 40, X \leq 60, B \leq 10^{12}</math></p>	<p>设 <math>k = \frac{NX}{D}</math>，那么 <math>k &lt; X</math>，我们考虑枚举每一个 k，算出此时合法的 N 的数目。但每个 N 可能被重复计算，所以我们统计满足 <math>k   NX</math> 且不存在 <math>k' \in (k, X)</math> 使得 <math>k'   NX</math> 的正整数 N 的数目。</p> <p><math>N = \frac{kD}{X} = \frac{k / \gcd(k, X)}{X / \gcd(k, X)} \cdot D</math>，我们得到 <math>N \leq \frac{kB}{X}</math>，且设 <math>c[k] = \frac{k}{\gcd(k, x)}</math> 后 <math>N = c[k] \cdot M</math>。我们考虑对 M 进行计数，此时 <math>M \leq \frac{kB}{c[k] \cdot X}</math>。若存在 <math>k_1, k_2</math> 同时满足 <math>k_1   NX, k_2   NX</math>，那么 <math>c[k_1]   N, c[k_2]   N</math>，则 <math>\frac{c[k_1]}{\gcd(c[k_1], c[k])}   M, \frac{c[k_2]}{\gcd(c[k_2], c[k])}   M</math>，此时令 <math>d[p] = \frac{c[p]}{\gcd(c[p], c[k])}</math> 则 <math>d[p_1]   M, d[p_2]   M</math>，所以 <math>\text{lcm}(d[p_1], d[p_2])   M</math>。这样的话我们就可以考虑容斥原理来进行去重了：设 <math>Limit = \frac{kB}{c[k] \cdot X}</math> 则 <math>Ans[k] = \sum_{S \subseteq \{k+1, k+2, \dots, X-1\}} (-1)^{ S } \cdot \frac{Limit}{\text{lcm}\{d[p]\}_{p \in S}}</math>。直接暴力计算是 <math>O(2^X)</math> 的显然会超时，但是可以发现如果分子大于 Limit 后贡献为 0，且计算中涉及的 lcm 的个数并不多，所以我们用 DP 的方式进行计算就行了，注意加入一些常数优化。</p>	
Codechef Sep11 SHORT {49}	<p>给定正整数 n 和 k，求满足以下条件的数对 (u, v) 的数量： <math>u, v \in (n, k]</math> 且 <math>(u-n)(v-n)   uv-n</math>。</p> <p><math>n \leq 10^5, n &lt; k \leq 10^{18}</math></p>	<p>令 <math>a = u - n, b = v - n, m = k - n</math>，那么原问题变为求数对 (a, b) 使其满足 <math>a, b \in (0, m]</math> 且 <math>ab   (a+n)(b+n) - n</math>，即 <math>ab   n(a+b+n-1)</math>。当 <math>n=0</math> 时答案显然为 <math>(m-1)^2</math>，否则令 <math>n(a+b+n-1) = kab</math>，则 <math>b = \frac{n(a+n-1)}{ka-n}</math>。我们不妨令 <math>a \leq b</math>，即 <math>\frac{n(a+n-1)}{ka-n} \geq a</math>，那么可以解得</p> $a \leq \frac{n + \sqrt{(1+k)n^2 - kn}}{k} \leq \frac{1 + \sqrt{1+k}}{k} \cdot n \leq (1 + \sqrt{2})n$ <p>所以我们先枚举所有的 a，然后枚举 <math>n(a+n+1)</math> 的所有约数 b 判断其是否合法，但这样我们的复杂度是 <math>O(n^2)</math> 过高。可以发现</p> $k = \frac{n(a+b+n-1)}{ab} = n \left( \frac{1}{a} + \frac{1}{b} + \frac{n-1}{ab} \right) \leq \frac{n(2a+n)}{a^2}$ <p>当 a 很大时 k 会很小，所以这个时候我们改为枚举所有的 k，判断对应的 b 是否合法就行了。</p>	<p>时间复杂度： <math>O(Ln + \frac{n^3}{L^2})</math> 其中 L 为切换两种枚举方式的阈值，当 <math>L = n^{\frac{2}{3}}</math> 左右时复杂度即为 <math>O(n^{\frac{5}{3}})</math></p> <p>空间复杂度： <math>O(n)</math></p>



Codechef Oct13 FN {50}	<p>定义数列 <math>F_0=0, F_1=1, F_n=F_{n-1}+F_{n-2}(n\geq 2)</math>。现在有 <math>T</math> 组询问，每组询问给定 <math>C</math> 和 <math>P</math>，求出最小的自然数 <math>n</math> 使得 <math>F_n\equiv C(\bmod P)</math>，若不存在则输出-1。</p> <p><math>T\leq 100, 0\leq C&lt;P\leq 2*10^9</math></p> <p><math>P</math> 是质数且 <math>P\bmod 10</math> 为完全平方数</p>	<p>根据特征方程我们可以求得该数列的通项公式 <math>F_n = \frac{1}{\sqrt{5}}(\alpha^n - \beta^n)</math>，其中 <math>\alpha = \frac{1+\sqrt{5}}{2}, \beta = \frac{1-\sqrt{5}}{2}</math>。</p> <p>因为 <math>P\bmod 10</math> 为完全平方数，所以我们可以得到 <math>P\equiv \pm 1(\bmod 5)</math>，那么 5 在模 <math>P</math> 意义下必定存在二次剩余，所以将 <math>\alpha, \beta</math> 用等价的整数代替后问题化为 <math>\alpha^n - (-\alpha)^{-n} = D</math>，其中 <math>D = \sqrt{5} \cdot C</math>。</p> <p>当 <math>n</math> 为偶数时，由二次方程求根公式可以知道 <math>\alpha^n = \frac{D \pm \sqrt{D^2 - 4}}{2}</math>，同样我们找到 <math>D^2 - 4</math> 的二次剩余后问题就变成了离散对数，使用 BSGS 算法即可解决。</p> <p>至于如何求二次剩余，比如求 <math>x^2 \equiv C(\bmod P)</math>。一种方法是，当 <math>C \neq 0</math> 时我们求出 <math>P</math> 的原根 <math>g</math>，那么令 <math>x = g^y</math> 后有 <math>(g^2)^y \equiv C(\bmod P)</math>，恰好对应着离散对数问题，复杂度为 <math>O(\sqrt{P})</math>。这个算法在 Codechef 上能过，但是在 Tsinsen 上会超时，需要卡常数 = = 但其实是存在 <math>O(\log^2 P)</math> 的更优秀的算法的，具体见博客： <a href="http://blog.miskcoo.com/2014/08/quadratic-residue">http://blog.miskcoo.com/2014/08/quadratic-residue</a></p>	<p>时间复杂度： <math>O(T\sqrt{P})</math></p> <p>空间复杂度： <math>O(\sqrt{P})</math></p>
Codechef Sep14 FACTORIZ 【challenge】 {51}	<p>给定一个大于 1 的整数 <math>N</math>，要求给出尽可能多的大于 1 的整数使得它们的积恰好为 <math>N</math>。每个测试点有 <math>T=100</math> 组数据，若对于所有数据你给出了共 <math>M</math> 个数，则得到 <math>M^2</math> 的评分。</p> <p>10%: <math>N\leq 10^{18}</math>，<math>N</math> 均匀随机分布</p> <p>15%: <math>N\leq 10^{18}</math>，<math>N</math> 不是均匀随机分布</p> <p>50%: <math>N\leq 10^{1000}</math>，<math>N</math> 均匀随机分布</p> <p>25%: <math>N\leq 10^{1000}</math>，<math>N</math> 的质因子均不超过 <math>10^{18}</math>（<math>N</math> 由若干个 <math>10^{18}</math> 以内的随机数相乘得到）</p>	<p>对于 <math>N\leq 10^{18}</math> 的情况，我们可以通过 Pollard-Rho 算法直接质因子分解，但是其质因子个数相对于 <math>N\leq 10^{1000}</math> 的数据来说对评分的影响很小，消耗大量的时间是很不划算的，我们不如直接尝试前若干个质数来得到同样不差的解。而对于第 4 类数据我们能很容易地找到 <math>N</math> 的较小质因子，所以我们考虑快速找出所有的第 4 类数据，然后对其尝试更多的质数进一步分解。</p> <p>具体的话，我们可以先用前 200 个质数对所有大于 <math>10^{18}</math> 的数分解一遍，然后把这些数按照已经分出的质因子个数从大到小排序，取前 25 大再尝试前 90000 个质数。而为了尝试尽可能多的数，我们要对高精度进行常数优化，减少取模次数。</p>	
Codechef Feb14 COT5 {52}	<p>给定一个初始为空的 max-treap，现在需要实现 <math>n</math> 次操作：1. 插入一个给定权值和重量的节点；2. 删除一个给定权值的节点；3. 询问两个权值在 treap 中对应节点之间的路径长度。数据保证权值和重量互不相同。</p> <p><math>n\leq 200000</math></p>	<p>max-treap 的中序遍历就是节点的权值大小顺序，那么我们可以发现两个节点的 LCA 其实就是它们对应权值区间中重量最大的节点，我们利用线段树便能在 <math>O(\log n)</math> 的时间内求出，所以我们关键要考虑如何求一个节点的深度。</p> <p>一个节点的深度等于它的祖先数，对于权值为 <math>x</math> 的节点，权值 <math>y</math> 是它的祖先等价于 <math>LCA(x, y)=y</math>，即 <math>y</math> 是 <math>[x, y]([y, x])</math> 中重量最大的节点。我们考虑 <math>x&lt;y</math> 时的情况，此时若将节点按权值从小到大排列，可以发现 <math>x</math> 的祖先节点恰好对应着以 <math>x</math> 为起点的(重量)连续上升序列，<math>x&gt;y</math> 时情况类似。线段树求连续上升序列长度是经典问题，我们只需维护区间最大值 <math>maxval</math> 和以区间左端点为起点的连续上升序列的长度 <math>len</math> 即可。</p> <p>具体的话，我们令函数 <math>find\_r(p, v)</math> 表示线段树中节点 <math>p</math> 所对应的区间中，能够接在权值 <math>v</math> 之后的连续上升序列的长度。如果左儿子的最大值 <math>maxval(lc)&gt;v</math>，那么 <math>find\_r(p,v)=find\_r(lc,v)+len(p)-len(rc)</math>，否则 <math>find\_r(p,v) = find\_r(rc,v)</math>。利用这个函数，我们线段树节点信息的更新便能在 <math>O(\log n)</math> 时间内解决，由于一个区间涉及的节点会有 <math>O(\log n)</math> 个，所以总复杂度为 <math>O(\log^2 n)</math>。</p>	<p>时间复杂度： <math>O(n\log^2 n)</math></p> <p>空间复杂度： <math>O(n)</math></p>

Codechef Apr13 FAULT  【challenge】 {53}	有 $n$ 个变量和 $m$ 个异或方程，要求删除尽可能少的方程，使得剩下的方程不能够解出 $n$ 个变量的唯一解。  $50 \leq n \leq 200, 2n \leq m \leq 1000$	<p>首先，我们考虑如何判断 <math>m</math> 个异或方程是否能解出 <math>n</math> 个变量的唯一解。第一种经典方法就是使用高斯消元判断是否存在自由元，而第二种方法就是查看是否存在一列能够被其它列线性表出。对于这题，我们从第二种方法入手。</p> <p>某一列能够被其他列线性表出，等价于这些列向量异或以后得到的是零向量。所以我们的任务就变为选出一个非空集合，使得集合中所有列向量异或后得到的向量中 <math>1</math> 的个数尽可能少（因为我们删去 <math>1</math> 所在的行便可线性表出）。</p> <p>对于这个问题，我们采取类高斯消元法，每次随机取出一列和一个主元素，然后对其他列进行该主元素的消元，并且每次操作后记录最优值，重复约 <math>30000</math> 次便能得到不错的分数了。</p>	
Codechef Feb12 FINDSEQ {54}	给定一个长度为 $n$ 的整数序列 $A$ 和一个长度为 $5$ 的排列 $P$ ，现在要求在 $A$ 中找出一个长度为 $5$ 的子序列 $B$ ，使得 $B$ 中元素的大小关系与 $P$ 相同，即 $B_i$ 是序列 $B$ 中的第 $P_i$ 小的元素。 $n \leq 2000$	<p>我们先通过一些变换使得 <math>P_1 &lt; P_3</math> 且 <math>P_1 &lt; P_5</math> 方便之后的讨论。首先我们枚举 <math>B_2</math> 和 <math>B_4</math> 的位置，将剩下的三个元素分隔到互不相交的区间中。若 <math>P_3 &lt; P_5</math>，那么显然我们可以贪心地选择 <math>B_1</math> 和 <math>B_5</math> 使得 <math>B_1</math> 尽可能小而 <math>B_5</math> 尽可能大，然后我们再来判断此时是否存在合法的 <math>B_3</math>。而如果 <math>P_3 &gt; P_5</math>，那么我们先贪心选择最小的 <math>B_1</math>，再在合法条件下贪心选择最小的 <math>B_5</math>，最后确定 <math>B_3</math> 就行了。为了实现上面的操作，我们需要想办法求出：</p> <p>1. 区间 <math>[L, R]</math> 内是否存在权值在 <math>[a, b]</math> 范围的数；</p> <p>2. 前缀 <math>[1, i]</math> 或后缀 <math>[i, n]</math> 中比 <math>A[k]</math> 小的最大值以及比 <math>A[k]</math> 大的最小值。前者我们离散化后做一个二位前缀和便能解决，而后者我们 <math>DP</math> 预处理一下便能 <math>O(n^2)</math> 解决本题了。</p>	<p>时间复杂度： <math>O(n^2)</math></p> <p>空间复杂度： <math>O(n^2)</math></p>
Codechef Mar14 STREETTA {55}	给定两个长度为 $n$ 的数组 $A$ 和 $B$ ，初始时数组 $A$ 为 $0$ ，数组 $B$ 为负无穷大。现在有 $m$ 次操作： <p>1. 对于数组 <math>A</math> 的某个区间加上一个等差数列；</p> <p>2. 对于数组 <math>B</math> 的某个区间与一个等差数列取 <math>\max</math>；</p> <p>3. 询问 <math>A_i + B_i</math> 的权值。</p> <p><math>n \leq 10^9, m \leq 300000</math></p>	<p>操作 <math>1</math> 是线段树经典问题，直接在节点上记录标记 <math>(k, s)</math> 表示这个区间加上了一个首项为 <math>s</math> 公差为 <math>k</math> 的等差数列就行了，显然标记可合并。而对于操作 <math>2</math>，我们有两种处理方法：</p> <p>【方法 <math>1</math>】我们在线段树每个节点记录 <math>(k, s)</math> 表示这个区间需要对这个等差数列取 <math>\max</math>。那么我们可能遇到一个节点有两个标记的情况，此时我们求出这两个等差数列分别的优势区间（若一个等差数列的优势区间为整个区间则只保留这个标记），这两个区间肯定是一个前缀区间和一个与之互补的后缀区间，其中必定有一个区间是完全包含于某一个子树中的。那么这时我们可以将优势区间只存在于一个子树中的标记下传，而另一个标记保留在当前节点上。这样的话每个标记最多下传 <math>O(\log n)</math> 次，而总标记数为 <math>O(m \log n)</math>，所以总复杂度为 <math>O(m \log^2 n)</math>。</p> <p>【方法 <math>2</math>】我们将等差数列看做是一个以下标为自变量的一次函数 <math>k * x + s</math>，那么 <math>B_i</math> 的值就是所有一次函数在 <math>x = i</math> 时的最大函数值，这个问题是凸包的经典应用。于是我们现在的任务就是在区间 <math>[L, R]</math> 中插入点 <math>(k, s)</math>，在 <math>x = i</math> 对应凸包中查询。如果插入的点和询问的斜率有序，我们可以类似标记永久化，使用线段树在每个节点上存储完全覆盖这个区间的所有点所构成的凸包，然后查询只需在根到对应叶子节点的路径上，利用单调性在每个凸包中都求一次然后取最大值即可。至于这题点和斜率都不一定有序的话，我们可以使用平衡树+二分，当然用 <math>CDQ</math> 分治会更加简单，复杂度为 <math>O(m \log^2 n)</math>，常数较大。</p>	<p>时间复杂度： <math>O(m \log^2 n)</math></p> <p>空间复杂度： <math>O(m) / O(m \log n)</math></p>



Codechef May13 QTREE {56}	给定一棵 $n$ 个节点的基环外向树，每条边上有边权，两点之间的最短路定义为经过边数最少的路径，保证环的边数为奇数。接下来有 $m$ 次操作：1. 将两点之间最短路径上所有边的边权取相反数；2. 询问两点之间最短路径上各条边边权的最大子段和（可以为空）。 $n, m \leq 10^5$	动态求最大子段和是线段树经典问题，所以我们将环上的边用一棵线段树维护，然后对于所有外向树的树边用树链剖分维护，修改和询问时对其分别操作即可。注意分类讨论与细节。	时间复杂度： $O(n \log^2 n)$ 空间复杂度： $O(n \log^2 n)$
Codechef Dec12 DIFTRIP {57}	给定一棵以 1 为根的 $n$ 个节点的树，每个节点的权值为它的度数，两条路径被视为相同当且仅当它们长度相同且路径上点的权值都对应相同。求这棵树有多少条互不相同的从孩子节点走向祖先方向的路径。 $n \leq 100000$	不考虑字符集大小的话，这题是广义后缀自动机经典题。但是这题字符集很大可以达到 $n$ ，所以我们用 <code>map</code> 来存储边集即可。至于复杂度的话，由于所有点的度数之和为 $2n-2$ ，所以度数的种类也就是字符集大小最多为 $\sqrt{n}$ ，加上 <code>map</code> 后总复杂度为 $O(n\sqrt{n} \log_2 n)$ ，但是似乎很难达到这个上界 ==	时间复杂度： $O(n\sqrt{n} \log_2 n)$ 空间复杂度： $O(n\sqrt{n})$
Codechef Oct12 MAXCIR {58}	给定二维平面上的三个点 $ABC$ 以及 $n$ 个操作，第 $i$ 个操作能够使 $A$ 的 $x$ 坐标增加 $dx_i$ 并且使 $A$ 的 $y$ 坐标增加 $dy_i$ ，每个操作只能使用一次。现在你可以使用最多 $k$ 次操作，要求最大化 $ AB + AC + BC $ ，并且答案的绝对误差小于 $10^{-12}$ 。 $0 \leq k \leq n \leq 500,  dx_i ,  dy_i  \leq 10^6$ $ x_A ,  y_A ,  x_B ,  y_B ,  x_C ,  y_C  \leq 10^9$	$ BC $ 不变，所以我们只要最大化 $ AB + AC $ 。可以证明一定存在两个数 $u, v$ 使得 $ AB + AC $ 最大的同时 $u \cdot x_A + v \cdot y_A$ 也取得最大值。证明的话，我们设最大值为 $D$ ，则必定有 $ AB + AC  \leq D$ ，这可以表示为一个以 $B, C$ 为焦点的椭圆，最优解 $A'$ 在这个椭圆的圆周上，那么我们只要求出过 $A'$ 的切线方程便能找到一组合适的 $u, v$ 了。  假如我们已经知道 $u$ 和 $v$ ，那么我们只需将操作按贡献从大到小排序，然后取前 $k$ 个中的正权操作即可。可以发现，我们只关心操作之间的相对大小关系而不需要知道具体的 $u$ 和 $v$ ，类似 <b>COMPLEXT</b> 题，我们枚举每两个操作算出其贡献相等时的关键向量，然后根据这些向量划分出若干个极角区间，那么在每个极角区间内操作的相对大小关系不变，所以我们在该区间内任取一向量来二分出所有的正权操作，并利用前缀和求出总贡献来更新最大距离。每当转过一个极角区间时，我们将大小关系改变的操作（必定是一个连续的区间）重新排序并重新求前缀和。由于我们还关心操作贡献的正负，所以还要考虑操作贡献为 $0$ 时的两个关键向量。最后因为精度要求非常高，我们需要使用分象限叉积来实现极角排序，并且需要特殊的开根号算法，同时注意数据类型溢出问题。	时间复杂度： $O(n^2 \log_2 n)$ 空间复杂度： $O(n^2)$
Codechef Dec14 RIN {59}	有 $n$ 门课程和 $m$ 个学期，每一门课都必须选择在某一个学期内完成，若第 $i$ 门课在第 $j$ 个学期完成则能够获得 $A_{i,j}$ 的分数（ $A_{i,j} = -1$ 表示不开课）。同时有 $k$ 个限制条件( $P_i, Q_i$ )表示课程 $P_i$ 必须在课程 $Q_i$ 之前完成。求学完 $n$ 门课程的最大平均得分是多少。 $n, m, k \leq 100, A_{i,j} \geq -1$	首先我们将分数取相反数，那么问题就变为求最小总得分。考虑最小割，我们将每一门课程拆成 $m+1$ 个点 $V_{i,0} \sim V_{i,m}$ ，然后在 $V_{i,j-1}$ 到 $V_{i,j}$ 之间连上 $-A_{i,j}$ 的边，同时在 $S$ 到 $V_{i,0}$ 和 $V_{i,m}$ 到 $T$ 之间连上 $+\infty$ 的边，那么此时我们就可以保证每门课都必定会完成了。接下来处理限制条件，对于限制( $P, Q$ )，我们在 $V_{P,i}$ 与 $V_{Q,i+1}$ 之间连 $+\infty$ 的边，那么如果课程 $P$ 选择在第 $i$ 个学期完成即割掉边 $V_{P,i-1} \rightarrow V_{P,i}$ ，此时会存在增广路 $S \rightarrow V_{P,i-1} \rightarrow V_{Q,i} \rightarrow T$ ，我们必定会割掉 $V_{Q,i} \rightarrow T$ 中的一条边也就是 $Q$ 在第 $i$ 个学期后完成。现在边的容量为负没法做最大流，但其实由于我们只会割掉恰好 $n$ 条边，所以我们将每条边的容量都加上一个定值然后最后再减去即可。	时间复杂度： $O(\text{maxflow}(V, E))$ 空间复杂度： $O(E)$  $V = nm$ $E = nm + km$
Codechef Oct12 MAXRECT 【challenge】 {60}	给出一个 $n \times m$ 的矩阵 $A$ ，求一个子矩阵使得其中元素之和尽可能大。这个子矩阵并不要求行列连续，只需找出一些行和一些列，然后选取这些行列相交处的元素。 $200 \leq n, m \leq 300$	如果我们知道要选取哪些列，那么我们算出每一行此时的贡献，然后贪心地选取贡献大于 $0$ 的行就可以了。所以我们直接对列的状态进行模拟退火，每次根据温度随机反转若干列的 $0/1$ 状态来寻找邻域即可。	



Codechef Jun14 SEAARC {61}	<p>给定 n 个点，第 i 个点的颜色为 C<sub>i</sub>。如果两个点 i 和 j 颜色相同，那么将会在坐标系第一象限内画一条过(i, 0)和(j, 0)的圆弧，圆弧的颜色与点 i 和点 j 相同。现在问你有多少对颜色不同的圆弧存在交点。 n≤100000</p>	<p>相交圆弧对较难处理，我们考虑补集转化用总圆弧对数目减去 ABBA 和 AABB 类型的圆弧对数目来得到答案。记 cnt[i]表示颜色为 i 的点数，则总圆弧对数为 <math>\sum_{u&lt;v}\binom{cnt[u]}{2}\cdot\binom{cnt[v]}{2}</math> 很容易求得，而 AABB 型我们也只需从小到大枚举圆弧 A 的右端点 i，并动态维护[1, i)中颜色 A 的数量和 (i, n]的总圆弧对数目便能在 O(n)时间内得出答案了。至于 ABBA 型，我们分三种情况讨论：</p> <ol style="list-style-type: none"><li>1. cnt[A]≥L，那么此时 A 的颜色最多只有 n / L 种，我们枚举颜色 A 并求出 f[i]表示[1, i]范围内颜色 A 的出现次数，然后我们枚举所有与 A 不同的颜色 B，那么此时异色圆弧对数目为 <math>\sum_{u&lt;v,C[u]=C[v]=B}f[u]\cdot(f[n]-f[v])</math>，我们可以枚举 u，然后用一个变量存储 f[n] - f[v]的和来在 O(cnt[B])的时间内解决，于是总复杂度为 O(n / L * n)；</li><li>2. cnt[A]&lt;L, cnt[B]≥L，我们枚举所有颜色 B 并算出数组 f[i]，然后枚举所有颜色 A，此时答案为 <math>\sum_{u&lt;v,C[u]=C[v]=A}\binom{f[v]-f[u]}{2}</math>，我们可以将组合数拆开，然后用类似情况 1 的方法处理，总复杂度为 O(n / L * n)；</li><li>3. cnt[A]&lt;L, cnt[B] &lt;L，可以发现此时颜色为 A 或 B 的圆弧最多只有 n*L 个。我们考虑从左到右枚举圆弧 A 的右端点 j，同时维护 f[i]为 [i, last(C[i])]范围内的圆弧数目，其中 last(C)为[1,j]中颜色为 C 且最靠右的点的位置，那么此时计算点 j 对答案的贡献，我们可以枚举所有的左端点 i 然后对 f[i]~f[j]的值求和来得到，注意排除掉同色圆弧对。每当右端点从 j-1 到 j 时，f 值受影响的只有[1, j]内与点 j 颜色相同的点，我们暴力修改即可。单点修改区间查询，我们使用树状数组维护 f 数组便能在 O(n*L*logn)的时间内解决该情况了。</li></ol>	<p>时间复杂度： <math>O(\frac{n^2}{L}+n\cdot L\cdot\log_2n)</math></p> <p>其中 L 为程序设定的阈值，当 L 取到 <math>\sqrt{\frac{n}{\log_2n}}</math> 时总复杂度即为： <math>O(n\sqrt{n\log_2n})</math></p> <p>空间复杂度： <math>O(n)</math></p>
Codechef Sep12 PARADE {62}	<p>在一个 n 个点 m 条边的带权有向图中，你需要安排若干条路径(S<sub>i</sub>, T<sub>i</sub>)，路径满足至少包含两个不同的点（即不能为自环），一条路径的花费由三部分构成：1. 路径 Si-&gt;Ti 上所有边的边权和；2. 若 Si≠Ti 额外增加 C 的费用；3. 若一个点未被任何一条路径经过额外增加 C 的费用。现在有 q 个询问，每次询问给定费用 C，求此时的最小花费。</p> <p>n≤250, m≤30000, q≤10000</p>	<p>首先我们用 floyd 求出任意两点之间的最短路，然后点 i 与点 j 之间连上 dis[i][j]的有向边，那么此时所有路径都将是简单路径且路径互不包含公共点。建立最小费用最大流模型，将每个点拆成左右两个点，S 向左边 n 个点连容量为 1 费用为 0 的边表示一条开始于这个点的路径，同样右边 n 个点向 T 连容量为 1 费用为 0 的边，然后左边的点 i 与右边的点 j 连上 dis[i][j]的边。初始时所有的点都不被路径经过，此时总花费为 N*C。而我们每增加 1 的流量，也就是多增加一条路径，那么此时会发生以下几种情况：</p> <ol style="list-style-type: none"><li>1. 这条路径的起点或终点为一个从未被经过的点，此时总费用减 C；</li><li>2. 这条路径连接了两条原本不相交的路径，此时由于总路径数减少 1，费用减少 C；</li><li>3. 这条路径连接了一条路径的首尾端点，此时费用同样减少 C。</li></ol> <p>于是我们发现，设本次增广的费用为 v，那么不管是什么情况总费用均为 v - C。并且由于 v 是递增的，所以我们增广到 v&gt;C 便可以停止增广。于是我们处理出每次增广的费用，然后查询时二分出增广的次数来求出总费用即可。</p>	<p>时间复杂度： <math>O(n^3)+O(n\cdot spfa(n,n^2))+O(q\log_2n)</math></p> <p>空间复杂度： <math>O(n^2)</math></p>

Codechef Jul13 RIVPILE {63}	<p>给你二维平面的 <math>n</math> 个点和 <math>m</math> 种圆盘，每个圆盘有一个半径 <math>R</math> 和价格 <math>C</math>。你可以买任意多的圆盘，并且将它们的圆心放置在互不相同的点上。现在你在直线 <math>y=0</math> 上，你想要到达直线 <math>y=w</math>，但是你只能在购买的圆盘上（圆周和圆内部）任意移动，请问应该如何购买圆盘使得花费最少。</p> <p><math>n, m \leq 250</math>, 数据组数 <math>\leq 10</math></p>	<p>我们将点 <math>i</math> 拆成 <math>m</math> 个点 <math>(i, j)</math> 表示在点 <math>i</math> 使用第 <math>j</math> 种圆（圆按半径从小到大排序），如果两个圆有交集就连上一条边，那么此时跑一遍最短路就是答案了。现在总边数过多，但可以发现如果 <math>(i, j)</math> 与 <math>(u, v)</math> 之间存在边，那么对于任意 <math>w &gt; v</math> 必定存在边 <math>(i, j) \rightarrow (u, w)</math>，所以我们只需要求出 <math>(i, j)</math> 到 <math>(u, \dots)</math> 存在边的最小 <math>v</math>，然后 <math>(i, j)</math> 向 <math>(u, v)</math> 连边，同时每个点 <math>(i, j)</math> 向 <math>(i, j+1)</math> 连边表示扩大此处购买圆盘的半径，这样就能把边数优化到 <math>O(n^2m)</math> 了。但是由于是多组数据，我们还需要使用一些常数优化。首先如果 <math>R_i \leq R_j</math> 且 <math>C_i &gt; C_j</math> 那么圆盘 <math>i</math> 必定不会使用，然后我们使用 <code>pb_ds</code> 库中的配对堆来降低 <code>dijkstra</code> 的复杂度便能通过此题了。</p>	<p>时间复杂度： <math>O(dijkstra(V, E))</math></p> <p>空间复杂度： <math>O(E)</math></p> <p><math>V = nm</math> <math>E = n^2m</math></p>
-----------------------------------	---	---	---