POI 解题报告

Vani

March 14, 2013

1 POI 2012

Festival (fes)

类似差分约束建出图，然后求出所有的强联通分量。然后一个结论：每个强联通分量里的答案是最长的最短路 +1。然后最终答案即把所有强联通分量的相加。

Letters (lit)

注意每次必定不会交换两个相同的字母。于是直接找到每个字母的最终位置，树状数组求一下逆序对即可。

Distance (odl)

对于每个数，枚举它的约数，然后记录由这个数字到达这个约数需要多少步，对于每个约数维护最优值和次优值。然后询问的时候直接枚举它的约数，寻找由哪个约数转移过来是最优的，时间复杂度 $O(N\sqrt{M})$

Rendezvous (ran)

显然这个图是若干个基环 + 树的结构。如果两个询问的点不属于同一个联通块，即无解。否则分若干情况讨论即可，具体看代码。

Well (stu)

首先二分最终的答案 z，然后讲序列处理成 z-平滑的，再枚举每个位置减为 0，线性扫出答案即可。(这题我也不知道怎么证明!)

Tour de Byteotia (tou)

首先将 1-k 点去掉，把剩余的图按双联通分量缩点，显然缩点后的图形成一个森林。然后再把 1-k 这些点以及相关的边添加进去，求一个生成森林，答案即为删除所有的非树边。

Vouchers (bon)

直接暴力模拟哪些人取了哪些数即可

Cloakroom (sza)

离线处理，将所有的物品和询问排序，然后扫描做背包即可，注意压位和常数优化.

A Horrible Poem (okr)

有一个结论，对于一个长度为 N 的字符串，如果 N/i 和 N/j 都为它的循环节且 gcd(i,j)=1，那么 N/(i*j) 也是它的循环节。于是对于每个询问的子串，用 hash 判断 N 除它的每个质因数是否为循环节即可。

Fibonacci Representation (roz)

结论：对于每个数的最优方案肯定要选择与它最接近的两个 fib 数中的一个。然后用 map 记忆化爆搜。

Squarks (squ)

首先如果知道了最小的数 a_1，可以利用 multiset 来推断出其他的数，方法即为第 i 次取所有的和中最小的 s，然后 $s-a_1$ 即为 a_i, 再从 multiset 里删除所有的 $a_i+a_j(1\le j<i)$. 由于已知 $b_1=a_1+a_2, b_2=a_1+a_3$, 然后我们可以枚举某个 $b_i=a_2+a_3$ 来推算出 a_1, 从而推出所有的 a_n.

Bidding (lic)

直接记忆化爆搜即可

Salaries (pen)

从大到小枚举每个权值 i，若存在某个节点 u 其权值为 i ，则将其所有未赋权值的儿子放入一个队列，而且儿子们的待定权值为 $i-1$，否则，将队列里所有待定权值大于等于 i 的节点从队首取出，然后将它们的儿子们的待定权值赋为 $i-1$ 放入队尾。如果此步所有满足权值 i 的节点只有一个，那么这个节点的权值就是可以确定的，输出即可。

Leveling Ground (wyr)

首先把所有的数除上 $gcd(A,B)$，顺便判掉无解。然后将数列做差分，于是在区间 $[l,r]$ 上做一次 $+A$ 的操作即可看为在差分序列的 l 位置 $+A$，r 位置 $-A$。

于是问题转化为满足 $\forall i, A*X_i+B*Y_i=0, \sum X_i=0, \sum Y_i=0, \min\{|X_i|+|Y_i|\}$. 我们对于每个不定方程求出 $|X_i|+|Y_i|$ 最小的解，然后为了满足第二个性质，还得选一些方程来进行调整。由于一次调整一定为 X_i 加上 B，Y_i 减去 A（或者相反），于是可以用一个堆来维护对结果增量最小的调整，这样不断调整到满足 $\sum X_i=0, \sum Y_i=0$ 即可。

Minimalist Security (bez)

对于一个联通块，如果把一个顶点的值设为未知数，就可以得到一些不等式和方程，然后推断出其余的点的值即可

Warehouse Store (hur)

从小到大考虑每个需求，贪心，能满足就满足之。用线段树来维护

Prefixuffix (pre)

循环同构可以看作去掉两次相同的前后缀。然后枚举第一段的长度 L，令 $f(L)$ 为剩下的最长相同的前后缀。然后可以发现 $f(L)\ge f(L-1)-2$ ，即 $f(L-1)\le f(L)+2$, 然后从大到小枚举 L 根据单调性计算 $f(l)$ 即可。

2 POI 2011

Conspiracy (kon)

假如我们找到了这样一个划分 (A,B)。由于 A 是独立集，B 是完全图，所以在令一个划分 (A',B') 中，A 最多只有一个点在 B' 中，类似 B 最多有一个点在 A' 中。如果有一个可行的方案，我们分三种情况处理：

- 枚举 A 中的某个点，把它加入 B 集合中，判断是否合法
- 枚举 B 中的某个点，把它计入 A 集合中，判断是否合法
- 枚举 A 和 B 中各一个点，将他们互换，判断是否合法

而寻找一个划分方案，则是经典的 2-sat 模型。（其实暴力 Dfs + 压位就过了）

Lollipop (liz)

注意到每个元素只可能 1 和 2 两种取值的性质，然后根据奇偶性在序列上二分位置即可。

Lightning Conductor (pio)

$a_j \le a_i + p - \sqrt{|i-j|} \to a_j + \sqrt{|i-j|} - ai \le p$, 于是对于每个 i 要最大化 $a_j + \sqrt{|i-j|}$

假设 $j < k < i$ 而且 $a_k + \sqrt{i-k} \ge a_j + \sqrt{i-j}$, 由于 $i-j \ge i-k$，所以 $a_k - a_j \ge \sqrt{(i+1)-j} - \sqrt{(i+1)-k}$. 有了决策单调性，所以直接用单调栈来优化即可。

Shift (prz)

如果把序列看成一个环，操作 b 等价于把整个环转一格，于是可以忽略操作 b。然后根据 n 的奇偶性构造解，具体实现见代码。

Plot (wyk)

首先二分答案的半径，为了实现随机增量，对于每一段，还要再次二分这一段的长度，然后直接最小圆覆盖。不过这样最坏数据的时间复杂度为 $O(NM \log N)$，不过如果先通过倍增来确定二分的上下界，再在 $[2^{k-1}, 2^k]$ 内进行二分，就可以把时间复杂度减至 $O((N+M)\log N)$ 了。

Strongbox (sej)

考虑若一个密码集合为 $\{a_1, a_2, \cdots, a_m\}$, 令 $g = gcd(a_1, a_2, \cdots, a_m)$, 则若 k 为密码的话，即等价于 $a_1x_1 + a_2x_2 + \cdots + a_mx_m = dn + k$ 有正整数解，即 k 为 $gcd(g, n)$ 的倍数，而且所有的密码都是 $gcd(g, n)$ 的倍数。于是我们就可以得到最终密码集合的形式必为 $\{a, 2a, \cdots, \lfloor \frac{n}{a} \rfloor a\}$, 而我们的目标即为找到最小的 a，$\lfloor \frac{n}{a} \rfloor$ 即为答案。

由于已知 m_k 为密码，所以 a 一定是 $gcd(m_k, n)$ 的公约数，而且 $\forall i \in [1, k-1]$ a 不是 m_i 的约数。令 $b_i = gcd(m_i, n)$, 则 a 要满足 $\forall k \in \mathbb{N}, ka \notin b_n$。

我们首先将所有的约数扔进一个集合 S 中，然后类似递推的方式从大到小枚举 n 的所有约数，对于当前约数 d，若 d 已经在集合中，就跳过，否则，枚举它去乘 n 的每个质因数，如果结果在 S 中，就把 d 扔进 S 中。否则如果 d 为 b_k 的约数，则 d 为一个合法解，最终选取最小的合法解 d，$\lfloor \frac{n}{d} \rfloor$ 即为答案。

Garbage (smi)

首先可以发现，那些不用改变状态的边一定可以不经过，然后把这些边删去，剩余的图若不存在欧拉回路则无解。否则进行 dfs 来找环，每找到一个环就将其输出即可。注意加上当前弧优化，否则会 TLE。

Tree Rotations (rot)

对于每个节点，交换他们的儿子不会影响其他子树的逆序对数。于是对于每个节点分别计算交换以及不交换产生的逆序对数，然后取最优值即可。对于每个节点我们用一个平衡树来维护其子树的权值，然后自底向上启发式合并，逆序对数就可以在合并中计算出，时间复杂度为 $O(n \lg^2 n)$

Tree Rotations 2 (rod)

类似 rot 一题的做法，但是我们使用路径压缩后的 Trie 来保存每个节点子树里的权值，于是合并的时候就可以直接合并两棵 Trie，复杂度为 $O(n)$ 的，于是总的时间复杂度为 $O(n \lg n)$，具体实现见代码。

Temperature (tem)

从前扫描所有的温度，维护一个单调队列，使得队列里的元素的左端点单调不增，时间复杂度为 $O(n)$ 。

Dynamite (dyn)

首先二分所需时间 k，然后类似树形 dp，对于每个节点计算出它的子树中深度最浅的引爆点以及深度最深的未引爆的炸弹。如果某棵子树中未引爆的炸弹可以由其他子树中的引爆点引爆，就无视这个炸弹，否则，如果与当前点的距离恰好等于 k，就引爆当前点，如果小于 k，就继续无视。这样贪心即可。

Party (imp)

每次选择两个没有关系的人，将他们删掉，这样做 $\frac{n}{3}$ 次，输出剩余的图即可。我随便写了个骗分，竟然 AC 了，大概做法就是排除一定不在答案中的点，然后将剩下的点按度数从大到小输出即可。

Inspection (ins)

首先判断无解的情况，对于一个点 u ，它将原树分割成了若干个子树，设其中最大的子树中有 s 个点，然后可以根据 s 和其余子树节点数之和的大小关系判断是否有解。如果有解，答案显然可以很简单算出。

Periodicity (okr)

这题我觉得是这届 POI 最难的一题了。

其实这题本质是让构造一个 01 串 S ，然后有一些 $\{a_i\}$，使得 $\forall a_i, S$ 的 a_i 长度的前后缀相同。不妨把 n 添加进 $\{a_i\}$ 集合中，然后从小到大考虑每个 a_i.

假设我们已经构造出了满足前 $i-1$ 个约数的字符串 S, 现在考虑第 i 个约数。如果 $2a_{i-1} \geq a_i$, 那么我们把 S 长度为 $a_i - a_{i-1}$ 的前缀放在 S 的前面就行了，否则，我们把新的字符串变成这个样子: $S0 \cdots 0S$, 其中中间有 $a_i - 2a_{i-1}$ 个 0，然后再 $O(n)$ 判断有没有出现别长度的相同的前后缀，如果出现的话，把中间最后一个 0 变成 1 就行了。

因为 $O(n)$ 判断一次，字符串长度至少翻倍，所以时间复杂度为 $1 + 2 + \cdots + \frac{n}{2} + n = O(n)$. 正确性什么的，暴力展开来证明就行了。

Meteors (met)

考虑分治做法。假设我们现在在处理所有答案在 $[l, r]$ 之间的国家，而且这些国家集合为 S。我们将 $[l, mid]$ 之间的操作施加给这些国家，然后 S 中国家肯定一些已经满足了需求，另一些还没有满足，不妨分别设为集合 S_1 和 S_2 ，然后将 S_1 递归进 $[l, mid]$, S_2 递归进 $[mid + 1, r]$ 即可。

最后当 $l = r$ 时，这个区间里的所有国家的答案都是 l。每个区间里的施加操作是可以做到线性，方法为把区间操作差分，变成单点修改和查询前缀和，然后将他们排序后统计即可。于是总时间复杂度为 $O(n \lg n)$，空间为 $O(n)$.

Sticks (pat)

将所有的长度从小到大排序，然后对于每个长度，选择小于它的颜色不同的最大的一个和大于它的颜色不同的最小的一个，然后判断能否组成三角形即可。实现的方法很多种，我写的时间复杂度为 $O(kn)$，可以 AC 的。

Programming Contest (pro)

把人和题目建成一个二分图，每个人最多匹配 $\lfloor \frac{t}{r} \rfloor$ 个题目。于是可以发现最多通过的题目数即为最大匹配数，然后发现每个人产生的罚时只与这个人通过的题目数有关，于是可以类似二分图匹配的匈牙利算法，贪心地每次对于一个还没有匹配满题而且已经做过的题目数最少的人寻找增广路即可。

3 POI 2010

Guilds (gil)

如果有一个孤立点，显然无解，否则，随便 dfs 出一个生成树，然后直接 01 染色就行了。

Railway (kol)

大名鼎鼎的双栈排序加强版。直接粘贴 sevenkplus 的题解：

那么，换一种方式思考，我们在构造图的过程中不妨假设这个图是二分图，得出每个数进入哪个栈，然后得出方案，判定这个方案是否可行。

每次找到互相有约束且不在同一联通块的两个位置 i, j ，在其间连一条边，然后将 i 与 j 所属的联通块合并。可以知道，这样构造出的图一定是一个森林。

令 a 为输入数据，b_i 为 $\min\{a_k, i < k \leq n\}$。而每次需要找的是联通块内存在某值处于 $b_{i-1}..a_i$ 范围内的所有联通块。观察到 b 是不降的，那么就可以用一个栈来维护（当然你也可以将其看成是一个单调队列），这个栈内联通块内元素的最大值是单调不降的。

我们将每一个联通块用一种叫 JZP 树的方式存储。初始时没有 JZP 树，栈为空。于是，对于每个 $b_{i-1} \ldots a_i$，需要做的是：

(0) 建立一个 JZP 树，含有一个节点（当前节点）。不加入栈中。(1) 找到所有最大值 $< a_i$ 的 JZP 树，加边，将其和当前节点所在的 JZP 树合并。

(2) 将当前 JZP 树加入栈

(3) 删除栈中的 JZP 树的所有 $< b[i]$ 的元素

(4) 删除栈中的空的 JZP 树

(可以看标程或者我的程序是怎么写的)

于是需要建立一种数据结构，能够支持：合并两个 JZP 树，求出 JZP 树中最大值，删除 JZP 树中最大值。标程中使用了一种猥琐数据结构（似乎是左偏树?），维护了当前元素深度这个多余量，使得每种操作的时间复杂度为 $O(logn)$

最后，森林构建出来了。暴力求出每个节点应该进哪个栈（不用管多种方案，可以证明，若有解，则当前森林中求出的任意一个解都可行)。再用一遍模拟，判断该方案是否可行。最后输出。

Beads (kor)

枚举所有的 k，然后使用 hash 判重就行了

Divine Divisor (naj)

首先用所有小于 10^6 的素数去试除，因为所有的数小于 10^{18} ，所以接下来所有的数的分解质因数后只有 p, pq, p^2 这三种形式。

然后考虑两两之间的最大公约数，显然如果最大公约数不为 1，那么必然是质数，于是直接拿这个最大公约数去试除

再接下来考虑 p^2 的形式，这个直接开根号判断一下即可

最后所有的数只剩下了 pq, p 这两种形式，而且两两之间要么相等，要么互质。于是对于每个数，用 miller-rabin 判断是否是素数，如果是的话，直接去试除，否则试除的结果乘 2 就行了

Intelligence Test (tes)

对于删除后序列的每个数，计算它在原数列中最靠前可能出现的位置贪心就行了。这个实现方法有好多，我直接对每个数开一个 vector 存它出现的位置然后在里面二分查找了。

Antisymmetry (ant)

这个，直接裸的 manacher algorithm 了

Hamsters (cho)

先预处理出两个字符串相邻的话重叠的部分有多长，然后矩阵乘法快速幂即可

Blocks (klo)

不妨令 $c_i = a_i - k$，于是 $i \dots j$ 这一段可以全部大于 k 的条件就是 $\sum_{i \le k \le j} c_k \ge 0$, 然后就是求最长的一段使其和大于 0.

再设 s_i 为 c_i 的前缀和，也就是 $s_i = \sum_{1 \le j \le i} c_i$，然后把 (i, s_i) 看作坐标系中的点。于是原问题就转化成了找一个 x 方向跨度最大的起点和终点，终点的 y 坐标大于起点的 y 坐标。

考虑一个点 A，如果 A 的右上方有另外一个点的话，那么 A 一定不是最优的终点，同理如果 A 左下方有一个点的话，那么 A 一定不是最优的起点，然后我们把所有非最优的起点和终点删除，会发现起点和终点各自形成一条链，当起点在一条链上移动时，终点也会在另一条链上单调移动，这样直接 $O(n)$ 扫一边即可。

Sheep (owc)

如果有一条对角线分割的两部分中，有一部分中有奇数个点，那么这条对角线一定不会被选取。所以先预处理出所有合法的对角线，然后直接 dp 就好了。

Teleportation (tel)

将原图中的除了起点和终点的点分为五类:

- $A = \{u | dis(1, u) == 1\}$
- $B = \{u | dis(1, u) == 2\}$
- $C = \{u | dis(2, u) == 1\}$
- $D = \{u | dis(2, u) == 2\}$
- $E = others$

于是最终的答案就是:

$$Ans = \frac{n * (n-1)}{2} - m - (n - 1 - A) - A * (C + D + 1) - B * (C + 1) - D - E - min(A, C) * E$$

Monotonicity (mon)

令 f_i 表示 i 结尾的最长的符合要求的序列的长度，然后我们可以根据 f_i 的大小计算出下一个接在 f_i 这个序列后面的数与 a_i 的大小关系，然后维护三个集合，分别对应三种不同的大小关系，根据计算出的大小关系把 f_i 插入到对应的集合中就行了。更新时，f_i 要在这三个集合中找一个最优值更新即可。我用了树状数组来维护，时间复杂度为 $O(n \lg n)$

Monotonicity 2 (mot)

同上题

The Minima Game (gra)

每个人一定是从大到小拿牌，然后考虑 dp，令 f_i 为当前剩前 i 小的牌，当前人的最大收益。然后当前人要么只拿第 i 大的，就是 $a_i - f_{i-1}$ ，要么是还要拿，就是 f_{i-1} ，取大值即可。

Frog (zab)

先利用类似单调队列的东西预处理出来每个离每个石头第 k 远的是哪一个，然后倍增计算即可，也可以直接找循环节。

Bridges (mos)

先二分答案，然后就是经典的混合图欧拉回路，直接网络流就行了

Pilots (pil)

直接维护两个单调队列，一个维护最大值，一个维护最小值即可。