T1:

20pts:2^n暴力枚举每一位，O(n)或O（n^2）检验即可。

60pts:给一些复杂度较高的玄学做法。

100pts:

使用**差分约束算法**。

设 *di*​ 为构造的字串中第一个字符到第 *i* 个字符间 0 的个数减去1的个数的值；显然地，若要最小化构造的字串的字典序，等价于最大化构造的 *d* 数组的字典序。

由题目条件可得：

* ∣*di*​−*di*−1​∣=1
* *dLi*​−1​=*dRi*​​

这时我们就有个想法：如果上面的第一个条件为∣*di*​−*di*−1​∣≤1，那么就可以很轻易地最大化 *d* 的字典序。可以使用差分约束算法来解决这些不等式。

本题求最大字典序，显然将所构造的字符串01翻转即可。

T2

10pts：n≤10随便搞搞就可以了

30pts：n≤100 显然，每个数的最终位置只与其最后一次操作有关。我们称每个数的最后一次操作为“有效操作”。

那么假设有 L 个有效操作把数放在开头，有 R个有效操作把数放在结尾，那么 *aL*+1​,…,*an*−*R*​ 这一段的相对位置是一直没有改变的，因为它们没有进行过任何操作，所以， *aL*+1​,…,*an*−*R*​ 必须是递增的。

如果有了上面这个观察，就可以设出状态 dp[i][l][r]表示进行了 i*i* 次操作（有效+无效），L,R意义同上。转移如下：

dp[i + 1][l][r] += dp[i][l][r] \* 2 \* (l + r) // 无效操作，放在任意一次有效操作的前面。

dp[i + 1][l + 1][r] += dp[i][l][r] //有效操作

dp[i + 1][l][r + 1] += dp[i][l][r] //有效操作

n3就过了

100pts：n≤5000 我们发现 dp 转移过程中只用到了 l + r，所以有个想法是直接记录 dp[i][l + r]，最后再来分配这 l + r次有效操作。

于是可以写出 dp2[i][j]，其中 j = l + r。

而原来的 dp[i][l][r]则是现在的 dp2[i][l + r].

然后对于满足条件的 l,r统计即可。

T3:

10pts:

对于每一次询问，拉出对应的链，一一考虑当士兵数=pi是否满足要求，O(n2q)。

40pts：

做法1：排序+二分优化上述过程,可以做到O(nqlogn)。当树的形态随机生成，链长较短，可以做到O（qlog2n）可通过1、2、5、6四个数据点。

做法2：将所有点按pi排序，一个一个计入贡献，通过dfs序与lca预处理可以O（1）的判断点是否在链上。O（nq）。

60pts：结合以上两种做法或实现时常数足够优秀。

100pts：

二分答案+多组询问+不带修，显然可以用主席树或整体二分解决。这里只讲整体二分。对点排序，每次将小于mid的点计入，对于每一个询问通过树剖log2n的考虑是否已经满足需要。复杂度O（nlog3n）。

T4

首先，让我们看看什么时候数组是合法的。如果所有元素的和能被P整除，这显然是不合法的。如果这个和不能被P整除。设X为最频繁的元素(或其中之一)，假设它在数组中出现M次。所有数字乘以X-1，现在1是最常见的数字。设B1,B2,…,Bk是数组中所有大于1的元素。那么，当且仅当1的数量不超过(P – B1) +(P - B2) + …+ (P - BK) + P -1时，排序后是可行的。

假设1的个数大于这个数，那么它至少是(P - B1) +(P - B2) + … +(P - BK)+P+1，因为总和不能被P整除。因此，所有数的总和至少是P(K+1)+ 1。我们必须“越过”点P,2P,…,(K+1)P，对于每一次这样的“越过”，我们需要一个大于1的数字。然而，我们只有K个这样的数字，一个数字最多可以用于一次“越过”，所以这种情况显然是不合法的。

如果满足，我们可以用以下方式排列所有的数字:

设x是**当前**最频繁的数字之一， cur是当前的和。

•如果cur+ x不能被P整除，则在末尾加上x

•否则取任何不是x的数字，比如y，然后按此顺序追加y, x。

首先，在第二种情况下，前缀和不能被p整除:如果cur+ x能被p整除，那么cur+ y和cur+y+ x不能被p整除。

所以令x=1，我们总是尽可能取1，最后只有几个。这意味着我们的序列看起来像这样:

P-1个1，B1，P-B1个1，B2，…Bk，P-Bk个1

如果不行，只能是还剩至少一个1，此时1的个数会超过表述中的个数，矛盾。

之后统计不合法数组的数量。

通过数学归纳法可以证明，N为奇数时, 总和能被P整除的数组数为;N为偶数时，则为。

现在让我们计算一下不满足第二个条件(同时总和不能被P整除)的数组的数量。我们只计算那些比1大数的个数，并乘以P−1。

记dp[i][j]为有i个大于1的数，且数组中P- x的和为j。对于所有(i,j)，判断N - i的个数是否满足N – i ≥ j + P且N – i j (mod P)，如果满足，则减去dp[i][j]



最终时间复杂度O(N2)