Solution(By ZZJ&CJY)

D1T1 幸运值

n<=20

直接枚举所有可能的情况，暴力异或求和。复杂度 O(2^n*n)

n<=100,ai<=1024

考虑 dp 的方法，dp[i][j][k]表示前 i 个数中取 j 个数且目前异或和为 k 的方案数，最后再将所有方案的和加起来就可以了。

n<=2000

计算每一位对答案的贡献。这样就可以用刚刚 dp 的方法解决每一位的答案。复杂度 O(n*k*w)，w 为 ai 化为二进制后的位数。

n<=100000

当我们计算每一位对答案的贡献时，我们可以数出这一位(二进制位)上为 1 的数的个数，这样就可以枚举选出的 k 个数中，有多少个数这一位上为 1，利用组合数，就可以快速计算方案数，从而统计答案了。时间复杂度 O((n+k)*w)

D1T2 交通运输

n<=5000

用 dp[i]表示从 0 号车站到第 i 个车站的最小花费，这样就可以从 0 开始对所有能到达的后面车站的 dp 值。复杂度 O(n^2)。

maxc=1

从乘客的角度思考，每当遇到一个 v 更小(相比之前的车站)的车站，一定会下车换乘。这样我们发现到达一个车站的最小花费一定是从之前 v 最小的车站转移来的。这样就可以用 O(n)的时间复杂度处理这个问题了。

n<=10^5

这一个部分分我并没有想到很好的方法。但是数据存在梯度，所以常数比较小的暴力可能能在这一档多拿些分。

n<=10^6

我们发现靠前的 dp 值对之后 dp 值的贡献是一个形如一条直线(一次函数)的形式。这样我们在计算后面的 dp 值时，就是找到之前所有可能转移到这个点的所有直线在当前横坐标的最小值。我们发现 c 和 i%c 相同的两点可以转移到之后车站的集合是相同的。所以可以维护 maxc*maxc 个栈来维护。

我们发现如果求出一个新的 dp 值，它应该加入栈的栈顶在当先横坐标的值一定大于当前的点(否则就可以被这个点更新)，所以若新点的 v 值小于栈顶的 v 值，那么栈顶就对之后没有贡献了，这样栈里直线的形式就形如一个上凸包，且栈里直线的顺序在凸包中是有序的(新的直线在凸包中更靠左的位置)，所以当一个新的直线加入栈时，可以找到和当前凸包的交点(这个点的横坐标也就是这条线出栈的时间)。所以在这个交点左边所有原凸包上的直线(这些直线与新直线的交点在它们出栈时间之后)就没用了。这样在计算每个 dp 值时，就可以查一下所有可以转移的栈的栈顶。

所以每条直线只会进出一个栈一次，同时每个点计算 dp 值时需要查 maxc 个栈的栈顶。时间复杂度 O(n*maxc)，空间复杂度 O(n)。

D1T3 密码

m <= 2 , G <= 6 , n <= 15

直接暴力拼接，暴力判断 s 在不在串中即可，时间复杂度 O（q*G*(m^n)）。

m <= 5 , G <= 50 , n <= 400

我们在拼接的时候，发现答案应该为前 m 个的答案之和加上新产生的数量，对于新产生的数量，我们发现只有前 G-1 位和后 G-1 位有用，所以我们可以记录一下，然后暴力比对，时间复杂度 O（m*q*G*G*n）

m <= 5 , G <= 50 , n <= 50000

对于前面的那个方法，使用 KMP 比对即可，O（m*q*G*n）

m <= 9 , G <= 100 , n <= 10^6

我们可以观察以下一些性质：

我们用(Ks，Ke)分别表示第 K 个串（由算法 2 可以知道只有这些部分有用）

比如对于一个排列：1,5,4,3,2

则所有串为：

1s，1e	4s，5e	2s, 5e
2s，2e	5s，2e	3s, 2e
3s，3e	1s, 3e	4s, 3e
4s，4e	2s, 4e	5s, 4e

5s，5e	3s, 5e	1s, 5e
1s，2e	4s, 2e	2s, 2e
2s，3e	5s, 3e	出现循环，循环节为 20。
3s，4e	1s, 4e	

可以证明，循环节长度最大为 m*(m-1)

对于循环节上的同一位，增加的串数是一样的。

因此我们先预处理以 Ps，Qe 拼接能匹配上的串是多少，然后可以在一个循环节上暴力，最后在其他循环节直接取用即可。

时间复杂度 O（q*m*m*G+n*m*q）

m <= 9 , G <= 200 , n <= 10^9

对于循环节上的同一位，增加的串数是一样的。

因此我们很容易想到矩阵加快转移。

暴力维护转移矩阵，在对于答案直接矩阵乘法即可。

时间复杂度 O（q*m*m*G+q*m^3+q*m^6*log(n)）