**T1**

法1：（扩展域并查集）

简单并查集应用

建两排点xi和yi

有偶数边a-b，并查集连接xa-xb，ya-yb

有奇数边a-b，并查集连接xa-yb，ya-xb

查询是否有偶数路径只要查询xa和ya是否在统一集合即可

yy一下，存在偶数路径是要走偶数条奇数边，走一次奇数边会从x走到y，只有再走一条才能走回x

法2：边带权并查集

两个结论：

1. 若一个连通块里有奇环，则该联通块里的点两两之间一定有边权和为偶数路径
2. 若我们往树里加一条边，产生一个偶环，那偶环不会影响路径奇偶性

因此用并查集维护，每个点存储到代表元素的距离正负，再加一个标记判断当前连通块有没有奇环。如果连通块没有奇环，那么就像树一样合并，边权为d[x]^d[y]^z.有奇环则更新标记

**T2**

本题做法非常多，题解是一个搬题人自己觉得更好理解的做法，和std的代码有些不同。

**把a从小到大排序**

**枚举一个题a[i]，如果算出这个题在第x次被解决的方案数就能得到这种情况对答案的贡献，但好像不是很好直接算.于是考虑这题在前x次被解决的概率**

**先考虑我们的暴力做法：**

**枚举一个排列{3,1,2,4,5}，从1开始移动.假设k=2**

**x=1,区间中有{3,1,2} 去掉1**

**x=2，把4加入区间，{3,1,2,4} 去掉2 发现区间长度为[1,x+k-1]**

**所以方案数就是这个题在排列的前min(n,x+k-1)道题中是前i小的。差分一下就得到第i次被解决的概率**

**记m=min(n,x+k-1)，为第i次操作的时候曾出现过的数的总数**

**枚举m道题中有多少比当前题大的，记为j ,m-x<=j<=m**

**整个序列中大于a[i]的数有n-i个， 从中选出j个加入当前区间，C(n-i,j)**

**整个序列中小于a[i]的数有i-1个，从中选出m-1-j个加入当前区间 C(i-1,m-1-j)**

**总方案数C(n-i,j)\*C(i-1,m-1-j)**

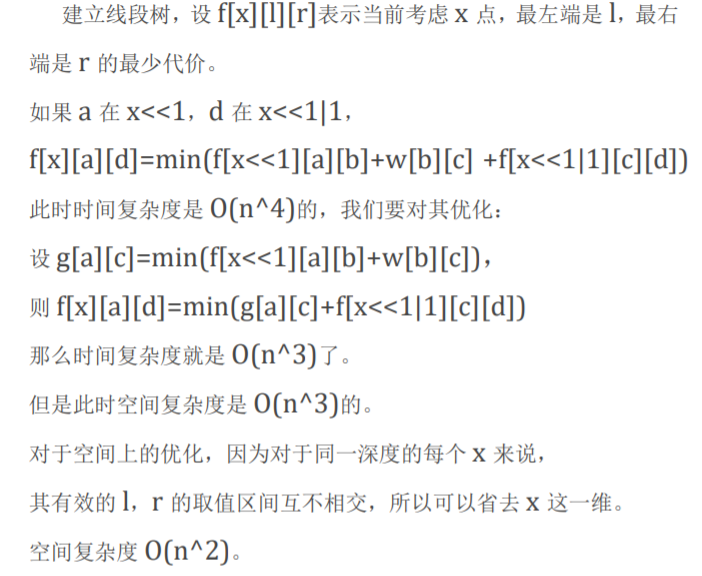
**由于序列还有排列顺序，先把m个数排列，再把剩下n-m个数排列， 所以sum=sigma(C(m-i,j)\*C(i-1,m-1-j))\*(m!)\*(n-m)!**

**再考虑一个题对时间的贡献**

**若他在第x次被解决，则x+1~n之后的时间都要增加a[i] ,贡献a[i]\*(n+1-x)**

**T3**

**法1:**



**法2：（好）**

考虑序列第i位的二进制表示与下标i的关系

观察样例：

下标: 00 01 10 11

序列：01 00 11 12

我们发现，当i=2的时候，i的第1位是1，p[i]由0变成了1，

总结一下：i的二进制表示从低到高的第1个1的位置记为tmp（lowbit）,则我们要取反第tmp位，否则与上一个数第k位相同

而比tmp位高的数字不变

设dp[pos][now]表示当前处理到序列第k个位置，p[pos-1]=now时的最小

先找出tmp， 把now的tmp位取反，而0~tmp-1位可任取，继续转移到下一个数nex

dp[pos][now]=min(dp[pos][now],dp[pos+1][nex]+dist(pos+1,nex)

用记忆化搜索实现，

枚举n O(n), 枚举0~tmp-1位O(2^{k-1})，找tmp O(k)

时间复杂度O(n\*2^{k-1}\*k)=O(n^2logn)