序列的值

根据期望的线性性，对于a_i，我们只要计算有多少$a_{1..i-1}$的子序列S，使得$Xor(S)\ xor\ a_i > Xor(S)$。

我们从高位往低位比较，可以发现当这一位a_i为0时，两个数肯定是相等的，当这一位a_i为1时，$Xor(S)$的这一位必须为0，且比较一定会结束。

所以我们找到a_i的最高位的位置，然后前面dp一下$f_{i,j}$表示xor和的第i位为j的序列个数，转移即可。

时间复杂度：$O(n\log a_i)$

乘法

令$f[i][j]$表示考虑了高于i的位，现在还过剩$j*2^i$。

转移时考虑第$i-1$位是加还是减还是不用即可。

时间复杂度：$O(n)$

投篮

设f_n表示剩下n个人时期望几轮结束，我们可以根据定义列出以下式子：

$$f_n = 1 + (p^n + (1-p)^n) * f_n + \sum_{i=1}^{n-1} f_{n-i} * (1-p)^i * p^{n-i} * C_n^i$$

我们移项后得到：

$$(1 - (p^n + (1-p)^n)) * f_n = 1 + \sum_{i=1}^{n-1} f_{n-i} * (1-p)^i * p^{n-i} * C_n^i$$

于是就得到了一个$O(n^2)$的dp。

用分治fft维护一下即可。

时间复杂度：$O(n\log^2 n)$

序列

考虑令$f_{i,j}$表示第i个数最后为j的答案。

可以发现这是一个分段函数，且斜率递增。

由于斜率递增，我们可以新增一些长度为0的段，使得他满足每一段的斜率比前一段多1。

由于每一段的斜率比前面多1，我们将斜率<=0的部分扔掉(显然不优)，然后维护第一个拐点的纵坐标，以及每个拐点的横坐标，就可以知道所有拐点的值。

考虑新加入一个数a_{i+1}，我们有$f_{i+1,j} = |j - a_{i+1}| + min_{k>=j} f_{i,k}$。

因为斜率递增，所以实际上有$f_{i+1,j} = |j - a_{i+1}| + f_{i,k}$。

分两段考虑，对于$j < a_{i+1}$，相当于前面加了一段$a_{i+1} - x$的函数，对于$j >= a_{i+1}$，相当于后面加了一段$x - a_{i+1}$的函数，于是前面斜率全部减1，后面斜率全部加1，具体实现只要新加入两个拐点即可，然后第一个拐点的纵坐标直接加$a_{i+1} - x$。

由于前面斜率全部减1，所以对于第一段，他的斜率从1变成了0，所以我们可以直接把第一个拐点删掉。

特别的，如果a_{i+1}小于等于第一个拐点的横坐标，我们直接将a_{i+1}作为第一个拐点即可。

最后的答案就是第一个拐点的纵坐标。

具体的维护可以用一个set实现。

时间复杂度：$O(n \log n)$。