A

首先我们观察一下什么时候要换手，就是前一个字符是`A`，经过了一堆`B`之后，变成`C`，那么这样我们需要换一次手。

所以对于连续的`B`，找到两端不同的字符，如果是`A`和`C`，那么就要换手。统计一下这次换手会在哪些子串里面，也就是左端点能选多少个乘上右端点能选多少个，然后全部相加即可。

B

首先$O(n^3)$是个简单区间dp。

有个能过的$O(n^3 \log n/w)$的做法。首先二分答案，然后dp就变成了可不可以。

我们考虑从右往左枚举l，然后从左往右枚举r，如果$dp[l][r]$可以，那么就让$dp[l][*]$数组or上$dp[r+1][*]$的值，表示如果$dp[l][r]$可行，那么$dp[r+1][k]$可行的话，$dp[l][k]$也可行，这样就可以在$O(n^3/w)$时间内完成dp。

$O(n^3/w)$的大致想法是从小到大加入每对可行的括号，然后每次(l,r)可行之后，可能会导致$(l-1,r+1)$变得可行，或者$(*,r),(l,*)$变得可行。这些位置都可以用位运算的方法找到，也就是维护$(l,*)$不可行的，和$(r+1,*)$可行的做个and，然后把所有不同的位置找出来，用一个类似BFS的方法更新。

这样每个状态只会被更新一次，每次更新的代价都是$O(n/w)$，所以总的是$O(n^3/w)$的。

C

首先可以将p_i排序，然后不管了。下面的对q_i的讨论都基于p_i有序。

首先观察，如果$i<j, q_i>q_j$，然后a_{2,q_i}对应的元素已经比$a_{1,i}$小了，那么a_{2,q_j}就一定会比$a_{1,j}$小，也就是对于一个逆序对，如果$s_i=1$，那么s_j也是1。

毛估估一下这个条件是充要的。

然后考虑如果q里面没有0，那么怎么计算方案呢。从前往后做dp，然后记录一下$s_i=1$的对应的q_i最大值。如果当前的q的值小于最大的q_i，那么这个元素一定会填1，否则可以选择填1或者0，如果是1就更新这个最大的q，否则就不更新。

这个过程，和求上升子序列个数是一样的。每次如果比上一个值更大，就可以考虑选或者不选，否则就跳过。

现在变成了，统计所有的q的上升子序列个数。那么考虑$dp_{i,j}$表示前i个元素，我们选了j个0元素，这里要求q_i是不等于0的。那么我们枚举上一个选的不等于0的位置k，假设k到i之间有a个等于0的位置，然后值q_k到q_i之间，有b个值没有出现过，那么我们可以枚举这些0里面选了多少个元素在上升子序列里面，假设选了d个，那么位置有$\binom{a}{d}$的选法，值有$\binom{b}{d}$中选法，将$dp_{k,l}\times\binom{a}{d}\times\binom{b}{d}$转移到$dp_{i,l+d}$的位置。

这样的时间复杂度是$O(n^4)$的。

注意到我们每次卷积，是成了一个$f_{a,b}(x)=\sum\binom{a}{d}\binom{b}{d}x^d$这样一个多项式，考虑如果用点值表示，也就是知道$f_{a,b}(1),f_{a,b}(2),\ldots$，那么卷积可以在线性时间复杂度完成。

但是直接FFT，时间复杂度是$O(n^3\log n)$，不一定能跑得过暴力。但是注意到恒等式$f_{a,b}(x)=f_{a-1,b}(x)+f_{a,b-1}(x)+(x-1)f_{a-1,b-1}(x)$，对于一个固定的$x$，所有的点值都可以在$O(n^2)$内求出来，所以可以在$O(n^3)$把所有点值求出来，总的时间复杂度就是$O(n^3)$的。

D

首先对于没有被选中的区间，我们可以认为划分成了若干个长度为1的区间。所以原问题可以看成，将序列划分成若干个区间，每段都升序排序，问有多少不同的结果。

首先考虑dp_i，表示对前i个元素操作，能得多少种不同的元素。然后枚举前一个元素j，如果$[j+1,i]$这段排完序之后的结果，可以被$[j+1,k],[k+1,i]$这两段分别排序得到，那么从j转移过来就是没有用的。如果不行，毛估估一下就是可以的，因为这是最小的长度。

直接按上面的方法做，时间复杂度是$O(n^3)$的。

考虑一下快速维护，如果$[j+1,i]$能被表示成$[j+1,k],[k+1,i]$，那么考虑一个这两段之间的元素x，将整个数组小于x的看成0，大于x的看成1，那么就是$[j+1,k]$这一段都是0，$[k+1,i]$这一段都是1。

同理，对于一个x，那么我们按上述操作之后，那么对于相邻的两个连续01段$[l,m],[m+1,r]$，有如果左端点在$[l,m]$，右端点在$[m+1,r]$，那么这样转移就没有用，画到二维平面上就是个一个矩形。

所以我们可以枚举这个x，然后把所有的矩形的不合法限制画出来，然后通过前缀和方法可以$O(1)$知道两个位置之间能否转移，时间复杂度是$O(n^2)$的。

最后我们考虑当x变大了，只有1个位置的值会变化，所以就$O(1)$个连续段会发生变化，也就是会新增$O(1)$个矩形。所以上面的不合法限制一共是$O(n)$的。

其次可以用扫描线的方法，从左到右扫过来，用线段树维护一下没有被覆盖的位置的dp值。

总的时间复杂度变成$O(n\log n)$。