通往强者之路

这是一道签到题。

发现每个位置的取值一定在$\{n-1,n,n+1\}$。

可以归纳证明：由$A_i=\sum_{j<i}[A_j\le j-i]$且$A_j\in\{n-1,n,n+1\}$得
$A_i=n-1+[A_{i-n-1}=2]+[A_{i-n}\ge 1]$。

为了方便后面叙述把取值范围视作$\{0,1,2\}$。

以连续n个为一组写成一个方阵。设$B_{i,j}$表示第i行第j个，那么可以通过如此算法来构造出这个方阵：假如已经计算完了前i行，对于所有的$B_{i+1,j}$，如果$B_{i,j}\ge 1$，则$B_{i+1,j}$加1。然后枚举j，如果$B_{i,j}=2$，则给$B_{i+1,j+1}$加一，特别地如果$j=n-1$则给$B_{i+2,0}$加一。

如果用滚动数组，那么除最后一个位置之外，这个操作相当于：每个位置上的数看成这个位置上叠着多少个方块，要将第二层的方块往右推。这个方块可能掉进洞里（右边的位置没有方块），也可能不掉下去。第$n-1$列是对下下行产生影响，可以看做有第n列，一开始第n列上有一个方块。这样对于每个位置x，右边的位置就是$(x+1)\mod(n+1)$，不需要对第$n-1$列分类讨论了。

算出哪些坑最后会被填上，并处理出填上它的方块原来位于哪里。按照坑被填上的时间顺序来做，然后处理询问。

具体处理方式不细说了，注意细节。

时间复杂度为$O(n\log n)$，其中$\log n$源于排序。

融入社会的计划

算法1.0

考虑一个简单DP：$f_{i,j}$表示考虑了前i个位置，第i个位置调整之后的数为j的方案数。转移：
$f_{i,j}+x-A_{i+1}\to f_{i+1,x}$。

朴素的实现$O(nR^2)$。

算法1.1

发现除非$A_i>L$，调整之后的A_i都不会超过L。

套用上面的DP，缩小值域，优化到$O(nL^2)$。

算法1.2

由于转移到$f_{i+1,x}$的$f_{i,j}$在一个区间，所以可以单调队列优化。

时间$O(nL)$。

算法2.0

容易发现对于f_i而言，合法的j形成一个连续区间。

并且~~打表发现~~在这个区间中$f_{i,j}$单调不减，并且$0 \le f_{i,j+1} - f_{i,j} \le 1$。

可以归纳证明：将转移分成两部分，先$f_{i,j} \to f_{i+1,x}$，再$f_{i+1,x} + x - A_{i+1} \to f_{i+1,x}$。先看第一步，$f_{i,j}$会转移到$f_{i+1,x}, x \in [L-j, R-j]$，由于随着$j$增大，$f_{i,j}$不减，$[L-j, R-j]$左移，所以得到的$f_{i+1}$是个单调不升，并且$0 \le f_{i+1,j} - f_{i+1,j+1} \le 1$的序列，移项再同时加$j + A_{i+1}$得$(f_{i+1,j+1} + j + 1 + A_{i+1}) - 1 \le f_{i+1,j} + j + A_{i+1} \le (f_{i+1,j+1} + j + 1 + A_{i+1})$。得证。

套上面的DP，利用这个性质转移，可以做到$O(nL)$。

算法2.1

记$[l_i, r_i]$表示$f_{i,j}$合法的j的区间。

有了算法2.0的那个性质，可以发现答案为f_{n,l_n}。

从f_{n,l_n}倒推回去，每次会从一个点倒推成一个区间，取区间中合法的最左端继续做下去。

时间复杂度$O(n)$。

放不下的过去

算法1

暴力$O(n^2)$，爱怎么做怎么做。

算法2.0

点分治或dsu on tree。以下以点分治为例：

对第一棵树点分治，考虑计算分属不同分治重心的儿子的子树的两个节点u, v的贡献，即$(D_u + D_v)(d_u + d_v - 2d_{lca})$。

D_u表示在第一棵树中以分治重心为根的深度，d_u表示在第二棵树中以某个点为根的深度。

枚举子树，计算这个子树中的u对前面枚举过的子树的v的贡献。现在问题是维护个数据结构存下v，然后询问u的贡献。

不难发现现在主要问题是求d_{lca}和$D_v d_{lca}$。可以在修改v的时候将v到第二棵树的根的链上的点都加上系数，询问u的时候询问u到第二颗数的根的链上的点的系数都加起来，显然这个时候系数恰好加了d_{lca}次。于是要维护数据结构：支持某个点到根的链上加以及询问和的操作。

朴素做法直接树链剖分，总时间复杂度为$O(n \log^3 n)$。

算法2.1

改进算法2.0，将树链剖分换成LCT。

总时间复杂度$O(n \log^2 n)$。

算法3.0

考虑点分治。要计算子树之间(u,v)的贡献，可以整个连通块的(u,v)的贡献减子树中(u,v)的贡献。

现在要计算一个连通块的贡献。将连通块内的点在第二棵树上建虚树，将D_u挂在虚树上，用DP可以$O(\text{连通块大小})$处理。

朴素实现可以做到$O(n\log^2 n)$的复杂度。

算法3.1

考虑优化上面的那个方法。上面的那个方法的瓶颈在于建虚树需要按dfn排序和求LCA。求LCA可以通过RMQ做到$O(n\log n)$预处理$O(1)$询问，排序可以在点分治之前先对dfn排好序形成一个序列，点分治的过程中按顺序将序列中的点分配到对应的连通块中。

于是总时间复杂度为$O(n\log n)$。

最后

经过实测，出题人发现算法2.1只是比算法2.0中略快。算法3.1中的优化效果不大。

标程没有卡常，所以爆标很正常~~不要大声喧哗~~。

另外题解中的所有算法都经过出题人实测，发现链、菊花、扫把、二叉树跑得比纯随机快，所以数据是纯随机的。如果有基于随机而爆标的算法出现，属于出题人无知。

二叉的建筑群

算法1

按照题意可以得到以下dp的式子，设$f[n]$表示为n的方案数。

$$f[n]=\begin{cases}1 & n=1\\ B\sum_{j=1}^{n-1}f[i]f[n-i] & 1<n\le k\\ B\sum_{j}^{n-1}f[i]f[n-i]+Af[k]f[n-k] & n>k\end{cases}$$

那么就可以$O(n^2)$得到$f[n]$的值了。

期望得分：$10'$

算法2

令$F(x)=\sum_{i\ge1}f[i]x^i$，那么可以得到$F(x)=Af[k]x^kF(x)+BF(x)^2+x$。

求第k项可以类比卡特兰数，$f[k]=\frac{1}{k}\binom{2k-2}{k-1}B^{k-1}$。

接下来的A都默认是原来的A乘$f[k]$

然后就转化成了求类似$F(x)=Ax^kF(x)+BF(x)^2+x$，

解个二次方程得到：$F(x)=\frac{1}{2B}(1-Ax^k-\sqrt{(1-Ax^k)^2-4Bx})$。

对后面的根式做多项式开根。

时间复杂度$O(n\log n)$。

期望得分：$30'$

算法3：

设$P(x)=(1-Ax^k)^2-4Bx$，$Q(x)=\sqrt{P(x)}$。

对$Q(x)$求导，两边乘$P(x)$，用$Q(x)$代换$\sqrt{P(x)}$可以得到，$Q'(x)P(x)=\frac{1}{2}Q(x)P'(x)$。

然后对这个式子提取系数后整式递推即可。

时间复杂度$O(n)$。

期望得分：$50'$

算法4：

以上的方法都不能突破线性。

而突破线性需要考虑用扩展二项式定理拆开根号，拆开根号可以得到（其中$\binom{a}{i}=\frac{\prod_{0\le j<i}(a-j)}{i!}$）：
$F(x)=-\frac{1}{2B}\sum_{i\ge 1}\binom{\frac{1}{2}}{i}(-4Bx)^i(1-Ax^k)^{1-2i}$

化简一下得到$F(x)=\sum_{i\ge1}\frac{1}{i}\binom{2i-2}{i-1}B^ix^i\frac{1}{(1-Ax^k)^{2i-1}}=\sum_{i\ge1}\frac{1}{i}\binom{2i-2}{i-1}B^ix^i\sum_{j\ge0}\binom{2i-2+j}{j}A^jx^{kj}$

而第n项就为$\sum_{j=0}^{(n-1)/k}\frac{(2n-2-(2k-1)*j)!}{j!(n-k*j)!(n-k*j-1)!}A^jB^{n-k*j}$

当k比较大的时候，n/k就会比较小，不妨直接求出每个阶乘的值。

那么就考虑如何快速求阶乘：

定义$f_m(x)=\prod_{i=1}^m(x+i)$。

考虑倍增求f，令$v=\lfloor\sqrt{n}\rfloor$。

假设我们已经有了$f_m(0),f_m(v),\dots,f_m(mv)$，现在要求$f_{2m}(0),f_{2m}(v),\dots,f_{2m}(2mv)$。

注意到$f_{2m}(x)=f_m(x)f_m(x+m)$。

问题转化为求$f_m((m+1)v),f_m((m+2)v),\dots,f_m(2mv)$和
$f_m(m),f_m(v+m),\dots,f_m(2mv+m)$。

考虑假设给定一个n次多项式h的$h(0),h(v),\dots,h(nv)$，要求$h(a),h(v+a),\dots$

考虑拉格朗日插值：

$$\begin{aligned}h(xv+a)&=\sum_{i=0}^n h(iv)\prod_{j=0,i\neq j}^n\frac{xv+a-jv}{(i-j)v}\\&=(v^{-n}\prod_{j=0}^n(xv+a-jv))\times(\sum_{i=0}^n\frac{(-1)^{n-i}h(iv)}{i!(n-i)!}\times\frac{1}{xv+a-iv})\end{aligned}$$

显然可以通过NTT做到$O(n\log n)$

对于$f_m(x)$推到$f_{m+1}(x)$则多乘个$x+m+1$即可。

考虑v不设成$\sqrt{n}$而是小一些，那么再用一次卷积算出所有$f_v(kv)$，需要$O(\frac{n}{v}\log n)$时间，但是计算一个阶乘只需$O(v)$的时间。

期望得分：25

算法5：

当$k=1$时：

令答案为$\sum_{j=0}^{(n-1)/k} p(j)$，令$q(j)=p(j)/p(j-1)$，可以发现$q(j)$是关于$j$的二次多项式除二次多形式。

令$q(j)=o(j)/u(j)$，o与u都是二次多项式。

那么答案为$p(0)+p(0)\sum_{i=1}^{(n-1)/k}\prod_{j=1}^{i}\frac{o(j)}{u(j)}$

考虑扩展做法4的方法，

令$v=\lfloor\sqrt{n}\rfloor$

定义
$f_m(x)=\prod_{i=1}^{m}o(x+i),g_m(x)=\prod_{i=1}^{m}u(x+i),h_m(x)=\sum_{i=1}^{m}(\prod_{j=1}^{i}o(x+j))(\prod_{j=i+1}^{m}u(x+j))$
。

如果求得$f_v(kv),g_v(kv),h_v(kv)(0\le k<v)$，那么就有$h_v(iv)\prod_{j<i}\frac{f_v(jv)}{g_v(jv)}=\sum_{k=iv+1}^{iv+v}\prod_{l\le k}\frac{o(l)}{v(l)}$，将前根号项求和加上一些散项就能得到答案。

由于$f_m(x),g_m(x),h_m(x)$都是不超过$2m$次多项式，所以要维护$2m+1$个点值。

从m推到$2m$的与上面的方法一样，不多赘述。

而从m推到$m+1$也很容易。

时间复杂度$O(\sqrt{n}\log n)$

与其他做法结合能够得到$100'$。