T1

接下来为了方便描述，我们令 $W_S=W_U+W_D+W_L+W_R$。

把题目中给出的方差式子搞一下可以得到：

$$\sigma^2=E(X^2)-E^2(X)$$

问题变为求经过不同坐标数量的期望 $E(X)$ 和经过不同坐标数量的平方的期望 $E(X^2)$。

先预处理 $A(i,x,y)$ 表示走了 i 步到达坐标 (x,y) 的方案数。

求 $E(X)$

考虑对每个坐标分开计算经过其的路径数量。

根据套路，我们希望在第一次或最后一次经过某个坐标的时候计算其贡献。

设 g_i 表示我们走了 i 步，且这 i 步没有回到过原点 $(0,0)$ 的路径数量。显然可以容斥转移：

$$g_i=W_S^i-\sum_{j=1}^{i}A(j,0,0)g_{i-j}$$

然后就有：

$$E(X)=\frac{1}{W_S^n}\sum_{i=0}^{n}W_S^i g_{n-i}$$

相当于我们每次在最后一次经过某个坐标时计算其贡献。

求 $E(X^2)$

考虑对每两个坐标分开计算同时经过它们的路径数量。

这东西还是不怎么好算，考虑对于两个不同的坐标，计算先经过第一个再经过第二个的路径数量 C，那么有：

$$E(X^2)=\frac{2C}{W_S^{2n}}+E(X)$$

设 $f(i,x,y)$ 表示我们走了 i 步，其中走了 j 步时第一次到达了某个坐标 (u,v)，然后接下来 $i-j$ 步后第一次走到了坐标 $(u+x,v+y)$ 的路径数量。

那么 C 就是：

$$C=\sum_{i=0}^{n}\sum_{(x,y)}F(i,x,y)W_S^{n-i}$$

求 $F(i,x,y)$。首先一个 naive 的想法是（显然 g_i 也可以代表走 i 步第一次到达某个坐标 (x,y) 的路径数量）：

$$f(i,x,y)\leftarrow\sum_{j=0}^{i-1}g_jA(i-j,x,y)$$

这显然是假的，我们需要减去第一次到达 (u,v) 前到达过 $(u+x,v+y)$ 的路径数量和第一次到达 (u,v) 后多次到达 $(u+x,v+y)$ 的路径数量。

考虑第一次到达 (u,v) 前到达过 $(u+x,v+y)$ 的路径数量。这个情况相当于第一次到达 $(u+x,v+y)$ 之后经过 (u,v) 又回到了 $(u+x,v+y)$，所以是：

$$f(i,x,y) \leftarrow -\sum_{j=0}^{i-1} g_j S(i-j,-x,-y)$$

其中 $S(i,x,y)$ 表示走了 i 步回到原点，且经过坐标 (x,y) 的路径数量。

考虑第一次到达 (u,v) 后多次到达 $(u+x,v+y)$ 的路径数量。这种情况显然可以枚举什么时候第一次到达 $(u+x,v+y)$，所以是：

$$f(i,x,y) \leftarrow -\sum_{j=0}^{i-1} f(j,x,y)A(i-j,0,0)$$

这样就结束了？并没有，我们发现第一种情况中会出现到达 $(u+x,v+y)$ 前到达 (u,v) 的情况，而这种情况在第二种情况中也被减了，所以要再加回去：

$$f(i,x,y) \leftarrow \sum_{j=0}^{i-1} f(j,x,y)S(i-j,-x,-y)$$

这样就可以完成 $f(i,x,y)$ 的转移了。

还剩下 $S(i,x,y)$ 的计算。

考虑先求出 $R(i,x,y)$ 表示走了 i 步到达 (x,y) 且没有回到过原点的路径数量：

$$R(i,x,y) = A(i,x,y) - \sum_{j=1}^{i} A(j,0,0)R(i-j,x,y)$$

那么就有：

$$S(i,x,y) = \sum_{j=0}^{i} A(j,x,y)R(i-j,-x,-y)$$

时间复杂度 $O(n^4)$。

T2

最终得到的 v 序列一定是单调不降的，因为每次都是一个前缀作减法，一个后缀作加法，因此结果显然是正确的。

同时可以将整个过程拆解成两部分 1.将 $[x_i+1:m]$ 内的所有值 $+1$，该操作执行 2 次 2.对整个序列执行 $v_i := \max(v_i-1,0)$。

考虑 v 的差分数组 d ，由于 v 单调不降，且 $v_0=0$，第一项操作等价于 $d_{x_i+1}+2$，第二项操作等价于找到最小的一个下标 i，满足 $d_i>0$，执行 d_i-1。

因此我们可以维护一个集合 S，对于第一项操作，每次在集合内加入两个元素，均为 $m-x_i$，对于第二项操作，等价于删除集合内最小的元素，最终答案即为 S 中所有元素之和，时间复杂度为 $O(nq\log n)$。

考虑用费用流刻画上述问题。

删除最小元素可以看作任意删一个元素，让最终的和尽量小，即最小费用最大流。

我们建立以下几条边：

> 1. $\forall 1 \le i \le n, (S \to i, 2, m - x_i)$。
> 2. $\forall 1 \le i \le n, (i \to T, 1, 0)$
> 3. $\forall 1 \le i < n, (i \to i+1, \infty, 0)$

一条路径 $S \to i \to i+1 \to \ldots \to j \to T$ 表示将第 i 次操作加入集合内的一个元素在第 j 次操作时删除。

对于未修改前的内容，可以利用初始时的集合 S 利用堆维护的方式维护出整个问题流的形态。

考虑如何维护修改，修改时等价于修改了 $S \to i$ 这一条边的边权，我们先将经过这条边的至多两条路径进行退流，退流后，在图上寻找新的两条最小的增广路。增广时，由于中间 $i \to i+1$ 的边可能有反向边，所以不光要考虑 $S \to i \to j \to T$，其中 $i \le j$ 的情况，还要考虑 $S \to j \to i \to T$，其中 $i < j$ 且 $j \to i$ 间均存在反向边的情况，可以利用线段树进行维护，时间复杂度为 $O((n+q)\log n)$，用其他方式维护可能会得到 $((n+q)\log^2 n)$ 的做法，可以拿到一部分部分分。