A1. 参见这里的解答

A2.

(1) 由绝对值函数的性质，$D(x)$ 在 $\mathbb{R}$ 上先递减再递增。记权值和 $W = \sum_{i=1}^{n} w_i$，$\{x_i\}$ 是 $\{a_i\}$ 的一个递增的排列，x_i 对应的权值为 y_i。记

$$d(x) = W \cdot D(x) = \sum_{i=1}^{n} w_i |a_i - x| = \sum_{i=1}^{n} y_i |x_i - x|$$

当 $x_k < x < x_{k+1}$ 时，

$$d(x) = \sum_{i=1}^{k} y_i (x - x_i) + \sum_{i=k+1}^{n} y_i (x_i - x)$$

因此

$$d'(x) = \sum_{i=1}^{k} y_i - \sum_{i=k+1}^{n} y_i = \sum_{i=1}^{k} y_i - (W - \sum_{i=1}^{k} y_i) = 2\sum_{i=1}^{k} y_i - W$$

令 $d'(x) \geq 0$，则 $\sum_{i=1}^{k} y_i \geq \frac{W}{2}$。由此可知，$D(x)$ 的最小值能够在 $x_{k^\star}$ 处取到，其中 $k^\star$ 是使得 $\sum_{i=1}^{k^\star} y_i \geq \frac{W}{2}$ 的最小正整数。

在课本提供的 Select 算法基础上进行修改即可在线性时间内得到 $x_{k^\star}$ 的值：选定“中位数的中位数”划分出 $\{a_i\}$ 的高低区间后，累加计算低区的权值和，如果已经超过了 $\frac{W}{2}$，则在低区递归查找，反之则在高区递归查找（此时注意要将低区的权值减去）。显然，此算法的最坏时间复杂度仍为 $\Theta(n)$。

得到最小值点后，用 $\Theta(n)$ 时间累加绝对值即可算出 $\min\limits_{x} D(x)$，因此总最坏时间复杂度为 $\Theta(n)$

(2) 将差分式展开为 $a_{n+1} - a_n = b_{n+2} - 2b_{n+1} + b_n$，移项得：$a_{n+1} + b_{n+1} - b_{n+2} = a_n + b_n - b_{n+1}$，即 $a_n + b_n - b_{n+1}$ 为常数，设其为 w，则在一个周期内有：

$$\begin{aligned} a_1 + b_1 - b_2 &= w \\ a_2 + b_2 - b_3 &= w \\ a_3 + b_3 - b_4 &= w \\ &\cdots \\ a_T + b_T - b_1 &= w \end{aligned}$$

将这些等式全部相加，得到：

$$w = \frac{1}{T}\sum_{i=1}^{T} a_i$$

将 b_n 用含 b_1 的式子表示：

$$b_n = b_1 + \sum_{i=1}^{n-1} a_i - w(n-1)$$

令：

$$S_n = \sum_{i=1}^{n} a_i$$
$$D_n = \frac{n-1}{T} S_T - S_{n-1}$$

则：

$$\sum_{i=1}^{T} |b_i| = \sum_{i=1}^{T} |D_i - b_1|$$

该式在 b_1 取为 $D_1, D_2, \ldots, D_T$ 的中位数时达到最小值。

使用累加法可以在 $\Theta(n)$ 时间内求出 $\{D_n\}$ 在 $1 \le n \le T$ 上的全部值，求出它们的中位数需要最坏 $\Theta(n)$ 的时间，计算绝对值之和需要 $\Theta(n)$ 的时间，因此总最坏时间复杂度为 $\Theta(n)$。

A3.

(1) 算法1给出了一种可能的算法。$i+j$ 的初始值为 2，每次循环中，$i+j$ 增加 1，而由循环条件，算法结束时一定有 $i+j \le n+m+2$，因此最坏时

间复杂度 $\Theta(n+m)$。

Algorithm 1: Calculate $|S|$

Input: A, B, n, m

Output: $|S|$

$s \leftarrow 0$;
$i \leftarrow 1$;
$j \leftarrow 1$;
while $i \leq n \land j \leq m$ **do**
    if $A[i] < B[j]$ **then**
        $i \leftarrow i+1$;
    else
        $j \leftarrow j+1$;
        $s \leftarrow s+n-i+1$;
    end
end
return s;

(2) 仿照算法1，在归并排序的 MERGE 步骤中统计前后两段间的 $|S|$，并递归累加，具体代码略。因为这一统计步骤具有线性最坏时间复杂度，因此递推式仍为 $T(n) = 2T(\frac{n}{2}) + \Theta(n)$，最坏时间复杂度仍为 $T(n) = \Theta(n \lg n)$

(3) 约束集 C 可以视为一张有向图 G，顶点对应数列中的位置，每个不等式约束对应图中的一条有向边，数列 A 的每个服从于 C 的排列对应 G 的一个拓扑序。

由题目条件知，此时 G 的拓扑序不唯一，这意味着在拓扑排序的某一步中，有多个顶点的入度同时变为 0。假设顶点 i, j 的入度同时变为 0，可以由此构造出两种拓扑序：i 在前，紧接着是 j，或 j 在前，紧接着是 i，其余都相同。这两个拓扑序所对应的 A 的排列即为所要求的 A_1, A_2。

(4) 假设 A 是一个 n 元数组，某个基于比较的算法通过一个高度为 h 的决策树解决了 (2) 中的问题，每个叶节点对应一个 $|I|$ 的取值，每条从根节点到叶节点的路径对应一个约束集。

A 所有可能的排列有 $n!$ 种，一个高度为 h 的决策树至多有 2^h 个叶节点，如果 $2^h < n!$，则某一个叶节点一定包含了至少两个不同的排列。根据 (3) 中得到的引理，可以在其中选出两个不同的排列 A_1, A_2，使得它们恰相差一次对换，根据 (2) 中 I 的定义，A_1, A_2 所对应的 $|I|$ 奇偶性一定不同。

但与此同时，因为 A_1, A_2 在同一个叶节点上，因此它们应当具有相同的 $|I|$，矛盾。

因此只能有 $2^h \geq n!$，即 $h \geq \lg n! = \Omega(n \lg n)$，这就给出了最坏时间复杂度的一个下界。

A4.

(1) 这些算法均可以生成所有可能的元素排列，只需注意到任一个 n 元排列都可以用 n 次交换实现：第一次交换放好第一个元素，第二次交换放好第二个元素⋯⋯对于这四种打乱算法，通过构造随机数的取值，都可以实现这样的 n 次交换。

(2)

前三种打乱算法分别进行了 $n, n^2, 2n$ 次随机数取值，每次取值分别有 $n, 2, n$ 种可能，因此算法运行的总情况数分别为 $n^n, 2^{n^2}, n^{2n}$。n 个元素可能的排列有 $n!$ 种，当 $n \geq 3$ 时，$n^n, 2^{n^2}, n^{2n}$ 均不被 $n!$ 整除，因此不可能将所有可能的排列均匀分配到所有可能的情况中，所以算法是不均匀的。

Shuffle-4 有 $n!$ 种等可能的情况，且能完备地生成 $n!$ 种排列，因此每种排列被生成的概率均为 $\frac{1}{n!}$，即算法是均匀的。

(3)

(3.1) 错误。

对于输入序列 $(1,2,3)$，如果有一个算法返回 $(1,2,3),(2,3,1),(3,1,2)$ 的概率各为 $\frac{1}{9}$，返回 $(3,2,1),(2,1,3),(1,3,2)$ 的概率各为 $\frac{2}{9}$，可以验证，任一元素被打乱到每一个位置的概率都为 $\frac{1}{3}$，但该算法不是均匀的。

(3.2) 正确。

考虑重复执行该算法所形成的马尔可夫过程，其中状态即为元素的排列。记该算法由原数组生成排列 σ 的概率为 $p(\sigma)$，其中 $\sigma \in S_n$，S_n 为 n 次对称群。注意到 $(\sigma'\sigma^{-1})\sigma = \sigma'$，因此由排列 σ 转移到排列 σ' 需要进行一次 $\sigma'\sigma^{-1}$ 的置换，因此转移概率 $p_{\sigma\to\sigma'} = p(\sigma'\sigma^{-1})$。由于该算法是完备的，因此各转移概率均非零，因此这一马尔可夫过程存在极限分布。

由群的性质，群中任意一个元素与其他元素作用恰能生成群内所有元素，因此：

$$\sum_{\sigma \in S_n} p_{\sigma\to\sigma'} = \sum_{\sigma' \in S_n} p_{\sigma\to\sigma'} = \sum_{\sigma \in S_n} p(\sigma) = 1$$

这表明重复执行该算法所形成的马尔可夫过程的转移矩阵是双随机的，易验证此时极限分布一定是均匀分布，即重复执行多次算法后，所得到的打乱结果一定趋于均匀。

(4) 结果见表1。

	Shuffle-1	Shuffle-2	Shuffle-3	Shuffle-4
JQK	$\frac{4}{27}$	$\frac{11}{64}$	$\frac{5}{27}$	$\frac{1}{6}$
JKQ	$\frac{5}{27}$	$\frac{11}{64}$	$\frac{14}{81}$	
QJK	$\frac{5}{27}$	$\frac{11}{64}$	$\frac{14}{81}$	
QKJ	$\frac{5}{27}$	$\frac{11}{64}$	$\frac{4}{27}$	
KJQ	$\frac{4}{27}$	$\frac{5}{32}$	$\frac{4}{27}$	
KQJ	$\frac{4}{27}$	$\frac{5}{32}$	$\frac{14}{81}$	
策略	{JKQ,QJK,QKJ} $\to$ {JQK,KJQ,KQJ}	{JQK,JKQ,QJK,QKJ} $\to$ {KJQ,KQJ}	JQK $\to$ {JKQ,QKJ,KQJ} $\to$ {QKJ,KJQ}	任意
期望猜测次数	$\frac{10}{3} \approx 3.33$	$\frac{55}{16} \approx 3.44$	$\frac{91}{27} \approx 3.37$	$\frac{7}{2} = 3.5$

表 1: 猜测策略及期望猜测次数

(5) 错位排列总可能数：

$$D_n = n! \sum_{i=0}^{n} \frac{(-1)^i}{i!} = \Theta(n!)$$

因此均匀错位排列作为一个随机变量，其熵为：

$$H = -D_n \times \left(\frac{1}{D_n} \lg \frac{1}{D_n} \right) = \lg D_n = \Theta(n \lg n)$$

因此算法从真随机源中提取的熵的期望下界为 $\Omega(n \lg n)$

为达到此下界，可以使用接受-拒绝采样：首先运行一次 Shuffle-4，然后验证打乱后的排列是否是错位的，如果不是，则重新打乱，直到得到（相对于原先的序列）错位的排列为止。

当 n 充分大时，D_n 约为 $\frac{n!}{e}$，因此采样成功的概率约为 $\frac{1}{e}$。假设得到错位排列所需的打乱次数为 X，则 $\mathbb{E}X = 1 + \left(1 - \frac{1}{e}\right) \mathbb{E}X$，解得 $\mathbb{E}X = e$，即平均需要打乱 e 次才能得到一个所需的错位排列。每次执行`randint(i,n)` 所需要的熵为 $\lg(n-i+1)$，因此执行一次 Shuffle-4 所需要的总熵为 $\sum_{i=1}^{n} \lg(n-i+1) = \lg n! = \Theta(n \lg n)$，所以这一算法需要的熵的期望值为 $e\Theta(n \lg n) = \Theta(n \lg n)$

A5. (1) 要证明 $\lesssim_m$ 为全预序，需验证其传递性、完全性（后者蕴含了自反性）。

传递性：令

$$x = \langle x_1, x_2, \ldots, x_m \rangle$$
$$y = \langle y_1, y_2, \ldots, y_m \rangle$$
$$z = \langle z_1, z_2, \ldots, z_m \rangle$$

若 $x \lesssim_m y$ 且 $y \lesssim_m z$，记 α 是使得 $\bigwedge_{i=1}^{\alpha} x_i \sim y_i$ 为真的最大自然数，β 是使得 $\bigwedge_{i=1}^{\beta} y_i \sim z_i$ 为真的最大自然数，$\gamma = \min\{\alpha, \beta\}$

- 若 $\gamma = \alpha = \beta = m$，则 $\bigwedge_{i=1}^{m} x_i \sim z_i$ 为真
- 若 $0 \le \gamma < m$，则 $(x_{\gamma+1} < z_{\gamma+1}) \wedge (\bigwedge_{i=1}^{\gamma} x_i \sim z_i)$ 为真

总之，$(\bigwedge_{i=1}^{m} x_i \sim z_i) \vee \left(\bigvee_{i=0}^{m-1}\left((x_{i+1} < z_{i+1}) \wedge \left(\bigwedge_{j=1}^{i} x_j \sim z_j\right)\right)\right)$ 为真，即 $x \lesssim_m z$

完全性：令 $x = \langle x_1, x_2, \ldots, x_m \rangle, y = \langle y_1, y_2, \ldots, y_m \rangle$。记 α 是使得 $\bigwedge_{i=1}^{\alpha} x_i \sim y_i$ 为真的最大自然数。

- 若 $\alpha = m$，则 $\bigwedge_{i=1}^{m} x_i \sim y_i$，故 $x \lesssim_m y$
- 若 $0 \le \alpha < m$，由 α 的最大性以及 $\lesssim$ 的完全性，一定有 $x_{\alpha+1} < y_{\alpha+1}$ 或 $y_{\alpha+1} < x_{\alpha+1}$，前者蕴涵了 $x \lesssim_m y$，后者蕴涵了 $y \lesssim_m x$

总之，一定有 $x \lesssim_m y$ 或 $y \lesssim_m x$

(2)

(2.1)

X 上的全预序又可视为 X 上的等价类划分以及定义在商集上的全序。将 X 划分为 i 个等价类的方式共有 $\left\{{|X| \atop i}\right\}$ 种，每个等价类可以定义 $i!$ 种不同的全序，因此可能的全预序总数为 $\sum_{i=1}^{|X|} i!\left\{{|X| \atop i}\right\}$。对于 $|X| = 5$，全预序总数为 541。

同构意义下不同的全预序个数等于将 $|X|$ 拆分为若干正整数之和的方法数（顺序不同视为不同的拆分）。将 n 拆分为 i 个正整数之和的方法数为 $\binom{n-1}{i-1}$，因此总数为 $\sum_{i=1}^{|X|}\binom{|X|-1}{i-1} = 2^{|X|-1}$。对于 $|X| = 5$，同构意义下不同的全预序个数为 16。

(3) 记 $|X| = n$。首先将 X 按 $\lesssim$ 排序。不失一般性，在接下来的论证中，对于 X 中任意一个第 i 小的元素，将直接用 i 指代该元素本身。记 $l = |X/\sim|$，则可以将 X 视为由 $1, 2, \ldots, l$ 组成的多重集。记 X 中 i 的个

数为 c_i，记 c_i 的前缀和 $S_i = \sum_{j=1}^{i} c_i$，定义 B_i 是满足 $S_{B_i} \geq i$ 的最小正整数。

记 X^m 中第 k 小的元素 $x = \langle x_1, x_2, \ldots, x_m \rangle$，其中 $1 \leq x_i \leq l$

首先确定 x_1 的值。X^m 中第一分量为 i 的元素个数为 $n^{m-1}c_i$。$y < x$ 的一个充分条件是 $y_1 < x_1$，因此 $k > |\mathbb{L}(x)| \geq \sum_{i=1}^{x_1-1} n^{m-1}c_i = n^{m-1}S_{x_1-1}$。$y \lesssim x$ 的一个必要条件是 $y_1 \leq x_1$，因此 $k \leq |\mathbb{LS}(x)| \leq \sum_{i=1}^{x_1} n^{m-1}c_i = n^{m-1}S_{x_1}$。所以 $S_{x_1-1} < \frac{k}{n^{m-1}} \leq S_{x_1}$，即 x_1 是满足 $S_{x_1} \geq \frac{k}{n^{m-1}}$ 的最小正整数，$x_1 = B_{\lceil \frac{k}{n^{m-1}} \rceil}$

再确定 x_2 的值。注意到 X^m 中第一分量为 x_1 且第二分量为 i 的元素个数为 $n^{m-2}c_{x_1}c_i$。$y < x$ 的一个充分条件是 $(y_1 < x_1) \vee (y_1 = x_1 \wedge y_2 < x_2)$，因此 $k > |\mathbb{L}(x)| \geq \sum_{i=1}^{x_1-1} n^{m-1}c_i + \sum_{i=1}^{x_2-1} n^{m-2}c_{x_1}c_i = n^{m-1}S_{x_1-1} + n^{m-2}c_{x_1}S_{x_2-1}$。$y \lesssim x$ 的一个必要条件是 $(y_1 < x_1) \vee (y_1 = x_1 \wedge y_2 \leq x_2)$，因此 $k \leq |\mathbb{LS}(x)| \leq \sum_{i=1}^{x_1-1} n^{m-1}c_i + \sum_{i=1}^{x_2} n^{m-2}c_{x_1}c_i = n^{m-1}S_{x_1-1} + n^{m-2}c_{x_1}S_{x_2}$。所以 $S_{x_2-1} < \frac{k-n^{m-1}S_{x_1-1}}{n^{m-2}c_{x_1}} \leq S_{x_2}$，即 x_2 是满足 $S_{x_2} \geq \frac{k-n^{m-1}S_{x_1-1}}{n^{m-2}c_{x_1}}$ 的最小正整数，$x_2 = B_{\left\lceil \frac{k-n^{m-1}S_{x_1-1}}{n^{m-2}c_{x_1}} \right\rceil}$

一般地，记：

$$\pi_i = \prod_{j=1}^{i} c_{x_i}$$

$$D_i = k - \sum_{j=1}^{i} n^{m-j} S_{x_j-1} \pi_{j-1}$$

则 x_i 是满足 $S_{x_i} \geq \frac{D_{i-1}}{n^{m-i}\pi_{i-1}}$ 的最小正整数，$x_i = B_{\left\lceil \frac{D_{i-1}}{n^{m-i}\pi_{i-1}} \right\rceil}$。

将递推式整理如下，初始值 $\pi_0 = 1, D_0 = k$：

$$x_i = B_{\left\lceil \frac{D_{i-1}}{n^{m-i}\pi_{i-1}} \right\rceil}$$

$$\pi_i = c_{x_i}\pi_{i-1}$$

$$D_i = D_{i-1} - n^{m-i}S_{x_i-1}\pi_{i-1}$$

排序用时 $O(n \lg n)$。n 的各次幂、$\{S_i\}$、$\{B_i\}$ 均可以用 $O(n)$ 时间的预处理全部求出。每次递推的用时为 $O(1)$，共递推 m 次，因此最坏时间复杂度为 $O(m + n \lg n)$

注意：排序后直接使用 n 进制分解确定各 x_i 将会得到错误答案。例如：令 $X = \{a, A, b\}$，其中 $a \sim A < b$，则 X^2 中第 3 小的元素应该是 aa 及其

等价元素，但如果直接使用进制分解，会误认为前三小的元素是 aa, aA, ab，因此得到错误答案 ab。事实上，进制分解的做法仅当 $\lesssim$ 是全序时成立。