A

首先令 $s_i = \frac{1}{n}\sum_{j=1}^{n}[a_j \le i]$，则等价于把 m 拆成 n 个非负整数 $b_{1\dots n}$，期望次数为 $n - \sum s_{b_i}$，则发现过程中都乘 n，最后再除以一个 n 就可以规避实数运算，只做一次值域不大的除法是不会掉精的。

暴力 dp 则有设 $f_{i,j}$ 表示选了 i 个非负整数，和为 j 的最大的 $\sum s$，直接做背包是 $O(nm^2)$ 的。

此时的 dp 和 a 的形态没有关系了，发现我们只需要求最终的 $f_{n,*}$，那么在 $f_{i,*}$ 的基础上有：

- $O(m^2)$ 暴力加入一个元素，即 $f_{i+1,j} = \max(f_{i,k} + s_{j-k})$，转移到 $f_{i+1,*}$，暴力 dp 就做了 n 次这个操作。
- $O(m^2)$ 用 $f_{i,*}$ 和自身做 $(\max, +)$ 卷积，即 $f_{2i,j} = \max(f_{i,k}, f_{j-k})$，转移到 $f_{2i,*}$。

这样就是我们初始有一个 0，每次可以花 $O(m^2)$ 的代价 $+1$ 或者 $\times 2$ 直到 $= n$，这个操作次数是 $O(\log n)$ 级别的。$O(m^2 \log n)$。

B

对于 DAG 的部分分，如果 s 能到 t，则每步贪心地走 s 的出点中，能到达 t 的编号最小的点。这对所有数据的简单做法有很强的启发性，因为这会引导从每个 t 出发建反图来预处理 "x 是否能走到 t" 这个信息。

如果有了上面的做法，发现可以直接推广到不是 DAG 的情况，如果这样走出了环，就说明存在长度无限大的理想路径，即可以在一个 每个点都能到达 t 的环中一直走，且过程中走任意合法出边都会使得字典序变大。那么对于每个 t 建立反图，预处理每个点走一条出边后能到达的后继。对于一组询问，使得不存在理想路径的情况有：

- s 无论如何都不能到达 t。
- s 跳了 $> n$ 次后继后还没有到达 t。

倍增回答询问。$O(n(n+m) + (n^2+q)\log n)$。

C

回答单次询问，一个显然的贪心是从左端点出发，每次优先满足左边第一个不为 0 的元素，要求满足完成前 $r-l$ 个元素后，最后一个元素恰为 0，且贪心过程中每个元素均非负则有解，否则无解。从右端点出发是对称的，如果能找到右端点出发的解那么反向就是左端点出发的一组解。

首先每个点会被经过一次，因此所有 a_i 初始可以 -1，然后只需要考虑一些"来回跳"状操作，记 b_i 表示以 i 为左端点的"来回跳"次数，一来一回记作一次，手玩得到 $b_i = a_i - b_{i-1}$，且 $a[l:r]$ 加 k 后 b 序列的变化为：在 $[l, r]$ 内，与 l 奇偶性相同的位置都 $+k$，奇偶性不同的位置都不变；在 $(r, n]$ 内，如果被修改区间长度为偶数，则不变，否则与 l 奇偶性相同的位置 $+k$，不同的 $-k$。

区间询问只需要分奇偶讨论，对区间内的 b 加或减 b_{l-1} 即可。由贪心得，若 $b_i < 0$ 或 $b_r \neq 0$ 则无解，否则有解，用线段树分别维护偶数位置和奇数位置的最小值即可。$O(q\log n)$，常数较大。

D

~~随机一伯拔拾次排列然后贪心，足以通过此题。~~ 这种乱搞被卡了，按 a 排序后贪心/枚举前缀也被卡了。

一些乱搞确实没想到怎么卡，在这里谢罪了。

记 $d_i \in \{0, 1\}$ 表示 i 是否被选中，则要求最大化

$$\frac{\left(\sum_{i=1}^{n} a_i d_i\right)^2}{\sum_{i=1}^{n}(a_i d_i)^2} = 1 + \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}[i \neq j] \cdot a_i a_j \cdot d_i d_j}{\sum_{i=1}^{n} a_i^2 \cdot d_i d_i}$$

记 $p_{i,j} = d_i \cdot d_j$，二分答案 x，即判定是否存在一个 p 满足：

$$\sum_{i=1}^{n}\sum_{j=1}^{n}([i \neq j] \cdot a_i a_j)p_{i,j} - \sum_{i=1}^{n}(a_i^2 \cdot x)p_{i,i} > 0$$

最大化左边的式子即可完成 check，可以关于 p 跑最大权闭合子图，要求若选了 $p_{i,j}$，则必须选 $p_{i,i}$ 和 $p_{j,j}$，然后原题中的限制即为 $p_{u,u} \to p_{v,v}$。不会出现选了 $p_{i,i}, p_{j,j}$ 但没选 $p_{i,j}$ 的情况，因为选了一定权值更大。

对于复杂度，网络流建模中的 $|V|, |E|$ 都是 $O(n^2)$ 级别，可以证明这张图中的 Dinic 复杂度是 $O(n^4)$ 的，只需要证明 BFS 的次数只有 $O(n)$ 次，每次寻找增广路的复杂度是 $O(n^3)$。

对于增广过若干次的图，从某个 $p_{i,i}$ 出发，它合法的后继只有 $p_{i,i} \to p_{j,j}$，$p_{i,i} \to p_{i,j} \to p_{j,j}$，$p_{i,i} \to T$ 三种选择，同时分层图是个 DAG，故每个 $p_{i,i}$ 只会经过至多一次，因为 $S \to T$ 的最短路长度最坏是 $O(n)$ 级别的，BFS 次数也就只有 $O(n)$ 次。

寻找增广路的复杂度为 DFS 的复杂度与修改增广路上的流量的复杂度。对于前者，在当前弧优化下为 $O(|E|) = O(n^2)$；对于后者，由于每次最多找 $O(E)$ 次增广路，每次增广路最长为 $O(n)$，故复杂度为 $O(|E|n) = O(n^3)$。

于是总复杂度是 $O(n^4|\log \epsilon|)$，其中 ϵ 是二分的精度，当然网络流题是卡不满复杂度的，所以为了卡乱搞把范围开到了 60，应该不影响大家敢写 flow 的信心。