card

sub1 直接阶乘枚举 $O(n!m)$，sub2 可以状压 $f_{S,i}$ 表示选了 S 中的串，以 i 结尾且前面都合法得到的最小字典序序列的前驱 $O(n^2 2^n)$ 或者之类的复杂度，sub3 可以对有解的情况找找规律。

下述串从 0 开始编号。

考虑一个串 S 内部，首先内部的 1 之间间隔必须恰为 m，且若 $S_j = \texttt{1}$，其开头在 T 中的位置为 i，那么必定有 $i + j \equiv b \pmod m$，即 $i \equiv b - j \pmod m$，于是建立点 $0 \sim (m-1)$，在 $(b-j)$ 和 $(b - j + |S|)$ 之间连有向边，一个从 0 开始到 0 结尾的欧拉回路就是一个合法拼接顺序，则问题转为找最小字典序欧拉回路，这是经典问题，在邻接表里按照关键字排序再跑 dfs 即可。
$O(\sum |S_i| + m)$。

market

sub1 模拟题意枚举所有情况即可。

sub2 启示思考"最坏情况"实际上是 $w_0 = r_0$，当 $i > 0$，$w_i = l_i$，那么问题变为找一个商品集合 S 使得题述期望最大。

二分转判定性问题，若要判断期望能否不小于 p，即要求 $\frac{\sum_{i \in S} w_i v_i}{w_0 + \sum_{i \in S} w_i} \geq p$，化简得 $\sum_{i \in S} w_i (v_i - p) \geq w_0 \cdot p$，即放入所有 $v_i \geq p$ 的商品最优，于是可以二分计算单次答案。

sub3 在找到上述性质之后可以每次二分。

最后只需要用值域线段树二分替代每次二分即可。$O(m \log V)$。

具体地，设值域为 $0 \sim W$，则对 $[0, W]$ 建立线段树，对于 $i > 0$ 将 $w_i, w_i \times v_i$ 放在位置 v_i，并在线段树的每个区间维护这两个信息的区间和，这是容易在题述操作下更新的。那么，每次询问在线段树区间 $[l, r]$ 上依次判断 $p = mid + 1$ 是否满足要求，判断的时候只需要知道 $\sum_{v_i > r} w_i$ 和 $\sum_{v_i > r} w_i v_i$ 的值就可以了，并且向左儿子走的时候更新这两个值传下去，实现细节见代码。

dating

sub 1 暴力枚举集合与点即可。

sub 2 可以先考虑一维问题，发现取中位数是最优的，如果有偶数个数，那么取中间两个任意一个是等价的。由于 x, y 两维独立，所以分别取中位数算一下即可。

根据 sub 2 的启发，我们猜想 x', y' 一定是由某个 x_i, y_j 组成的，这样目的地的范围就缩小到了 $O(n^2)$ 个。枚举目的地，只需要选出到达其距离最小的 k 个人即可，使用 nth_element 或排序做到 $O(n^3)$ 或 $O(n^3 \log n)$ 的复杂度，可以通过 sub 3。

接下来称目的地是 (a, b)，尝试固定 b 的值，从小到大扫描 a 的值，随着 a 的变化，(x_i, y_i) 到 (a, b) 的距离只会在 $a \geq x_i$ 时切换一次表达式，即等于 $|y_i - b|$ 这个基础值加上 $x_i - a$ 或 $a - x_i$ 这个动态值。我们把 $x_i < a$ 的 i 放进集合 L，$x_i \geq a$ 的放进集合 R，那么当 L, R 固定时只需要在 L, R 中分别找到最小的 k_1, k_2 个元素，使得 $k_1 + k_2 = k$ 且总和最小，只需要在过程中动态维护 L, R 并支持回答这个问题即可。

如果使用树状数组或是其它复杂度 $O(n^2 \log^2 n)$ 或是常数较大的做法，可以通过 sub 4，下面介绍双指针做法。

用堆维护 L, R 集合并保留当前最优方案的 k_1, k_2，在把 a 变到 a' 的过程中，假设有 c 个点从 L 移动到 R，那么 k_1, k_2 的改变量是 $O(c)$ 的，于是总复杂度 $O(n^2 \log n)$，详见 std。

string

不妨直接把所有 k 对应的出现次数求出。

sub1 枚举 k 再跑 kmp 即可，$O(n^2)$，sub2 只需要对每个插入位置附近枚举讨论一下也能完成。

sub3 引导考虑 P 的出现情况，设插入后 $S = L + B + R$，那么：

- P 完全包含于 $L/B/R$，可以直接 kmp 预处理。
- P 跨且仅跨过 L, B 的分界线，则找到 x 满足 B 长度为 x 的前缀等于 P 长度为 x 的后缀。建立 P 的失配树，把所有 $|P| - x$ 打上标记，则在 i 后插入 B 的此类匹配数量就是失配树上 $A[:i]$ 匹配到的最长前缀结点的祖先的标记数量和。
- P 跨且仅跨过 B, R 的分界线，反转字符串做一次上述操作即可。

这样就解决了 $|B| > |P|$ 的问题，下面考虑 $|B| \le |P|$ 特有的 $L + B + R$ 式匹配，即 P 的匹配跨过 B：

- 找出所有 B 在 P 中的匹配位置 $[x+1, x+|B|]$，然后转化为统计有多少位置 i 满足 $A[i-x+1, i] = P[:x]$ 且 $A[i+1, i+|P|-|B|] = P[x+|B|+1:]$。求出 $A[:i]$ 匹配的最长前缀长度 u 和 $A[i+1:]$ 匹配反串的最长长度 v，即求有多少个 x 满足 x 在正串失配树上是 u 的祖先且 $|P| - |B| - x$ 在反串失配树上是 v 的祖先。这是二维数点问题，离线扫描线+树状数组即可。

$O(n \log n)$。