Sol

1.网络流

算法 1:

写了一发网络流暴力扔了上去，得分不明。

算法 2:

观察性质：$f(i,j)>max(f(i-1,j),f(i,j+1))$，这提示我们答案是一个递减函数。
考虑计算有多少个使得答案 $\geq a$ 的 $f(i,j)$
预处理出来对于每个 i，左边选择集合 j，右边选择集合 k，是否存在可行流。
时间复杂度 $O(\frac{nk8^k}{w})$。

算法3:

仍然考虑计数问题。
不同的是我们设一点状态。
设 $g(i,j)$ 表示考虑 $i\sim n+1$ 层，从状态 j 出发，最多能转移到哪一层。
我们再次考虑一下，根据 Hall 定理我们可以得到如下转移方式：枚举每一个子集 S，找到其相邻的点集 T。对于每个 T 的大小为 $|S|$ 的子集 U，求 $g(i+1,U)$ 的最大值，设为 $h(S)$。则 $g(i,S)$ 为 $min\{h(T),T\subset S\}$
正确性可以通过枚举时间设置 $0/1$ 状态得到
单次转移复杂度 $O(2^k*k^2)$
总复杂度 $O(n*2^k*k^2)$

算法 4:

Hall 定理看上去优化不了了。
回到最初的网络流建图模型上来。
如果我们只需要计算 $f(1,i)$ 则只需要从第一层出发流尽量长的流即可。这样子跑网络流显然正确
同时我们可以直接从跑完第一层的残量网络继承过来直接计算第二层的流！
考虑在第二层的额外扩展的流，若其比第一层扩展的优秀而没有被扩展，说明这个流在第一层无法被扩展，不会被相对的贪心顺序影响。
否则这一条流显然没有第一层拓展的优秀，符合贪心策略。
由于寻找一次长度为 p 的流的复杂度为 $O(p*k^2)$，总流量为 $O(n*k)$，所以总复杂度为 $O(n*k^3)$
可以通过某些操作使得复杂度变为 $O(n*k^3/w)$

2.旋律

v_i 相同

令 $v_1=v_2=\ldots=v_n=q$。对 f_i 分类讨论。当 i 为奇数时，我们考虑在一个长为 $i-1$ 的满足条件的串 s 中间塞一个任意字符得到新串 s'，这个 s' 也必定是满足条件的。考虑把 s 写成 $s_1\cdot s_2$ 的形式（$|s_1|=|s_2|$，$\cdot$ 表示字符串连接），于是 $s'=s_1\cdot c\cdot s_2$。注意到由于 s 不含 border，s' 必定不含任何长度小于 i 的 border，而对于剩下的情况，假设 s' 存在一个长度大于等于 i 的 border，它必然是这种形式的：

$$\underbrace{\cdots\cdots\cdots}_{s_1}\underbrace{c\cdots}_{l_1}=\underbrace{\cdots}_{l_2}\underbrace{c\cdots\cdots\cdots}_{s_2}$$

不难注意到 l_2 是 s_1 的 border，l_1 是 s_2 的 border，进一步我们知道 $l_2\cdot c\cdot l_1$ 是 s 的一个 border，出现矛盾，假设不成立。于是对于 i 是奇数的情况，$f(i)=qf(i-1)$。i 为偶数时类比上面的证明，令 $|s_1|=|s_2|+1=\frac{i}{2}$，我们知道 s' 必然不存在长度小于或大于 $\frac{i}{2}$ 的 border，但是是有可能出现长度为 $\frac{i}{2}$ 的 border 的，于是对 i 为偶数我们有 $f(i)=qf(i-1)-f(\frac{i}{2})$。边界 $f_0=1$，我们直接 $O(n)$ 计算即可。

v_i 不相同

令 f_i 为 $v'=v_{[1,i]}$ 时的答案，考虑转移。用总方案减去不合法的，而对于不合法的我们枚举其最小的一个 border。由于 border 的 border 还是原串的 border，所以这个最小的 border 必须满足自已没有 border。又因为 v 单调不降，因此对于任意一个满足左侧限制且长度小于等于 $\lfloor\frac{i}{2}\rfloor$ 的字符串必定存在一种方案使其成为整个串的 border。于是得出转移

$$f_i=\left(\prod_{k=1}^{i}v_k\right)-\sum_{j=1}^{\lfloor\frac{i}{2}\rfloor}f_j\prod_{k=j+1}^{i-j}v_k$$

然后记 v 的前缀积为 t，于是

$$f_i=t_i-\sum_{j=1}^{\lfloor\frac{i}{2}\rfloor}f_jt_{i-j}t_j^{-1}$$

时间复杂度 $O(n^2)$。

可以继续考虑分治，令 $r_i=f_i\cdot t_j^{-1}$，我们有

$$f_i=t_i-\sum_{j=1}^{\lfloor\frac{i}{2}\rfloor}r_jt_{i-j}$$

也就是说对于一对 $j,k,r_j\cdot t_k$ 会贡献到 f_{j+k} 当且仅当 $j\le k$。我们分治时将左区间 r 的乘上右区间的 t 贡献到对应位置即可，时间复杂度 $O(nlog^2n)$。

注意到我们仅需要对 $l+mid-1\le n$ 的区间计算贡献，且每层需要进行的 dft 长度只有 $r-l$，是正常分治 fft 的一半，由此常数极小，上面的做法直接可以通过 10^6。

但可能还是不够的，注意到我们没必要计算出 $f_i(\lfloor\frac{n}{2}\rfloor<i<n)$ 这部分的值，因为这些并不会贡献到我们最终要求的 f_n 里面去，于是我们只分治算出前一半 f_i 的值然后最后单独算出 f_n 的值即可。

3.黄金矿工

以下简记 $K = k_{\max}$。

首先，可以看出很多金子其实是根本不可能用到的。具体来说，以 b 为分界线，那么 $x_i \le b$ 的金子只有至多 b 个。$x_i > b$ 时，对每个 $c \in [1, \frac{K}{b}]$ 考虑 $v_i = c$ 的所有金子。不难发现，我们一定是按照 x_i 从小到大的顺序去选它们，而一块金子至少会花费 bc 的时间，于是可能有用的金子数只有 $\frac{K}{bc}$，对 c 求和可得至多 $O(\frac{K}{b}\log n)$ 块金子。取 $b = O(\sqrt{K\log K})$ 可知有用的金子的数量只有 $O(\sqrt{K\log K})$。

接着考虑类似根号分治的做法，取阈值 B，对 $x_i \le B$ 的金子暴力跑 01 背包。$x_i > B$ 时注意到此时 dp 值的取值范围要比下标的取值范围小，所以我们可以另设 g_i 表示价值为 i 时背包容量最小是多少，此时加入一个物品的复杂度变为了 $O(\frac{K}{B})$。

m 次操作如何处理？首先倒序做使得删除变为加入，而加入直接用上面提到的手段就可以完成了。对于一个询问，我们可以对 $x_i > B$ 部分的背包暴力扫，然后并上 $x_i \le B$ 部分的背包贡献答案。

最终复杂度为 $O\left(BK + \frac{K}{B}(\sqrt{K\log K} + m)\right)$，平衡复杂度可得最终复杂度为 $O\left(K\sqrt{\sqrt{K\log K} + m}\right)$，若看作 $m = O(\sqrt{K\log K})$ 可得 $O\left(K^{5/4}\log^{1/4} K\right)$。