sol

by tsx

T1

- 预处理对每一个i，若从S的第i位开始匹配，则整个串T最多能让S匹配到哪里。
- 由于只有n个位置，所以一直匹配下去一定有一个$\leq n$的循环节。
- 找到循环节后直接算就可以了。
- 时间复杂度$O(nm+q)$。
- 当然用倍增/子序列自动机一类的东西也能搞。

T2

- 若 x,y 之间不存在祖先后代关系，那么直接考虑从 x 所在的子树和 y 所在的子树中选出两个点，使得它们的lca 分别是 x,y。
- 这实际上就直接是子树大小平方减去所有孩子的子树大小平方，预处理一下即可。
- 若 x 是 y 的祖先，那么情形是类似的，只不过 x 点的子树要改为去掉那个包含 y 的儿子的子树。方案数也可以直接通过上面的预处理 $O(1)$ 得到。
- 包含 y 的儿子可以用倍增或树剖求出。
- 注意有序会带来 $\times 4$ 的贡献。
- 时间复杂度 $O((n+m)\log n)$。

T3

- 暴力。
- 考虑不可能枚举三个点，所以尝试枚举中心点，再枚举位于中心点子树里的两个点。
- 那么我们就要对每个子树里的点对，算中心点子树外贡献的答案，假设点对有 x 个，子树外的点有 y 个，那么容易使用排序和二分做到 $(x+y)\log(x+y)$。
- 这里的做法其实就是说，将一边排序，然后对另一边的每个点对/点去做二分，找出答案，因为我们要算出每个点的答案，所以实际上两边都要做一遍。
- 神奇的事情是，$\sum x$ 和 $\sum y$ 都是 $O(n^2)$ 量级，所以总的复杂度是 $O(n^2 \log n)$。
- 但是，sort 和 lower_bound 很快，暴力也很快，所以最后出到了 3000。

T4

- 第一步破环为链。
- n^2 暴力很显然，直接打出能打的最大的牌就好了。
- 考虑如何优化。首先关注我们什么时候会打出牌 1。
- 这肯定是要让所有的 2,3,4 牌都打完的前提下我们才会打 1，所以为了方便计算，我们可以直接让小 T 碰到 2,3,4 牌直接打出，那这样的话打出的 1 的数量也是正确的。
- 考虑一个记录一个类似前缀和的 a_i，其中若当前的牌为 2,3,4，则分别令 $a_i = a_{i-1} + 1,2,3$，否则 $a_i = a_{i-1} - 1$。实际上 a_i 表征的是，如果从最开始出发在 i 这个位置往后还能抽多少牌。这个数组是可以差分的，所以我们可以去算出从 s 出发到 i 这个位置往后能抽多少牌。
- 那么当第一次 $a_i \le a_s - k$ 时，就代表着从 i 往后抽不了牌了，这个时候就必须使用牌 1 来抽牌。

T4

- 这个过程还可以继续下去，我们可以找到第二个，第三个必须打出 1 的位置。
- 但是我们不是无限费大神，所以我们还要考虑费用。
- 可以发现，当打出 1 的时候，费用实际上已经确定了，它就是当前抽到的牌中所有 2,3,4 的牌数总和。
- 所以我们可以知道到哪里费用就不够了，所以我们可以算出打出的最后一个 1。
- 那么之后可以将牌 1 视为牌 0，接着考虑牌 2 如何打出。
- 但现在有两个问题，第一个是我们怎么知道一定有牌 1 来让我们打出，第二个是这玩意复杂度还是爆炸。
- 第一个问题的解决方法和上面类似，记录 b_i，其中若当前的牌为 1,2,3,4，则分别令 $b_i = b_{i-1} + 0,1,2,3$，否则 $b_i = b_{i-1} - 1$。那么一旦 $b_i \le b_s - p$ 就代表我们需要打出牌 0，但是打出牌 0 根本没用，所以我们可以知道一个答案的上界，如果没超过这个上界就一定有非零牌可打，所以直接让答案和它取 min 即可。

T4

- 复杂度的问题也是好解决的。注意到这里除了第一次找，剩下的都是直接找形如 $a_j < a_i$ 的第一个 j，所以直接预处理之后倍增即可。
- 时间复杂度 $O(K(n+m)\log n)$，其中 K 为牌的种类数。