T1 - pearl

考虑一个显然的 $O(n^2)$ 做法，从每个位置开始放到不能放为止。

如果按奇偶分开算的话，相邻两个能放的部分有很多是重合的。这个时候不再每个都跑一遍，而是用双指针就可以解决了。

T2 - findk

考虑二分答案，之后可以从方阵的右上角开始，一行一行的找到比二分 mid 大的个数。因为这个有单调性，所以可以一个指针扫过去，总共只要最多 $2n$ 次询问。

T3 - sticks

为了最大化周长，我们有一个贪心的思路：选取最长的三根木棍。但是这可能不能组成三角。

为了方便，令 $b_0, b_1, \cdots$ 为将区间内的数从大到小排序后的结果。

考虑不能组成三角的条件：$b_0 \geq b_1 + b_2$，又因为 $b_1 \geq b_2$，所以可以得出 $b_0 \geq 2 \times b_2$。

而且，如果最大的三根不能组成三角，那么这个最大值和其它的所有都不能组成三角，可以直接删去。倘若重复这个过程，我们发现它只会重复 $O(\log V)$（V 为值域）次。

所以可以用线段树求出最大的几项，暴力查询；或者不断的查询最大的项并暂时删去它。

T4 - graph

观察到最优解里去的路上一定没有代价，回来的路上只有路径上割边这些边的代价。

不难想到将边双连通分量缩成一个点，这样剩下的就是一个树（点带权）了。

对于一个树而言，这是很经典的问题。可以直接手写一个平衡树，启发式合并，合并的时候算一下。也可以用点分治。