第一场：

A：第一步，分析题目，得到这样的结论，相同的产品距离差值只和产品的初始位置和结束位置有关，所以对于同种产品且数量>=2来说，要尽量使起始和结束位置分开，取任意产品A，放法是在末尾两个位置A XXXX.. A，就产生了一种由两边向里面包裹的形状，并且当前种类产品的差值等于包裹元素个数的总和+1，且每一种产品是外面产品（除去数目为1的产品）的差值-2。

B：如果是两个长方形，那么枚举长方形的垂直方向和水平方向的分界线，分别找出左边和右边的最小值（垂直），上边和下边的最小值（水平）。之前需要预处理出每一边的最优长方形。预处理的时间为O(N ^2)，总时间复杂度也为O(N ^2)。

C：我是用一个堆保存最小的K个值，并且随时间推移动态改变这个堆，用一个set去重。相当暴力。

第二场：

A：对于差值为a – b的元素为一个子群上的元素，同一个子群一定可以排序，a – b将整个圆盘上的数字分成了若干个不相杂的循环群，对每一个循环群排序，最后再检测真个圆盘是否满足题目要求。

B：太烦，看了一眼就没做。

C：比赛的时候就是没有想起来，这题大家普遍都说用维护一个凸包，凸包的斜率就是两点之间的平均值，并且斜率最大的一定出现在凸包的最后一条边，所以维护凸包一定能得到这个最大的斜边。

其实有一条更好的解法，dp[i]表示以i为结尾的最优解, dp[i] = max((dp[i – 1] + arr[i]) / dp[i].num, s[i] – s[i – m] / m)，可以不保存dp[i]，而且只需要保存m个s[i]，时间复杂度为O(N)。

后记：第一场C题，按时间广搜可能的攻击值，严格来说是层次遍历。

第二场C的解法是错误的因为不存在无后效性的最优值，DP[i]的值最优，不能保证i + 1时也是最优，举个例子，11 / 2 和 15 / 3，前者比后者大，然而末端向后移动1位，11 /3 却比15 / 4要小。所以靠谱的方法是二分枚举答案，然后用O（N）的方法去检测，复杂度为O（N \* logN）