Solution 8.5

T1 试卷

签到题。把 A 数组的元素放入桶中，遍历 B 数组查看是否在 A 数组中出现。

T2 魔法数字

我们发现满足要求的一位数只有 0， 1， 5， 6，而满足要求的两位数一定是满足要求的一位数的基础上拓展出来的。我们可以类似 bfs 的扩展的方法来做，由于数非常大，也还用相乘，故使用 __uint128_t 类型，无符号 int128，建议使用快读快写。可参考 std

【核心代码】

```
for (int i = 1; i <= 38; i++) EXP[i] = EXP[i - 1] * 10U;
__uint128_t k, ans[128] = { 0U, 0U, 1U, 5U, 6U };
read(k);
int ansk = 4, l = dgt(k);
for (int i = 2; i <= l; i++) {
    for (int j = ansk; j >= 1; j--) {
        for (unsigned int k = 1U; k <= 9U; k++) {
            __uint128_t New = (k * EXP[i - 1] + ans[j]);
            if (New * New % EXP[i] == New)
                ans[++ansk] = New;
        }
    }
}
```

T3 排序

最优解的存在性是显然的。现在我们考虑最优解应当满足什么样的性质。

①所有的数字最多只移动一次。

②如果我们将最优解做法中的数字分成移动和未移动两类，则未移动的所有数字构成一个上升子序列。我们设这个上升子序列$\{s_i\}_{(1\le i\le k)}$为"骨架"

③我们不妨给排列的两端添加上$a_0=0, a_{n+1}=n+1$最后排完序这两个元素也是不需要动的，可以认为这两个元素一直没动

④如果我们使用 dp[i] 表示以 ai 为骨架最后一个数字，把前 i 个数字从小到大排序需要的最小代价。则会有这样的关系：

a. dp[n+1]是最后的答案

b. 若 j<i 且 aj < ai,则：dp[i] <= dp[j] + small * L + big * R;其中 small 表示 a[j]~a[i]中小于 a[i]的数的个数，big 表示 a[j]~a[i]中大于 a[i]的数的个数。

这是因为我们可以把 a[i]当做骨架最后一个数字，a[j]当做骨架的倒数第二个数字，把所有 a[j]之前的数字按照 dp[j]的方案处理好，所有 a[j]和 a[i]之间的数字，比 a[i]大的就右移，比 a[i]小的就左移处理好。

并且 dp[i]的最优方案一定是利用某个 dp[j]通过上述方法得到的。
最后输出 dp[n+1]即可。

T4 出栈

贪心：
要让字典序最大就要让大的数尽量先出栈。

用一个数组 rmax[i]表示第 i 项到第 n 项的数的最大值。
如果栈顶元素大于第 i 项到第 n 项的最大值，那么直接让这个元素出栈，让大的先出栈总能保证字典序最大。
如果栈顶元素小于第 i 项到第 n 项的最大值，那就让该元素入栈，等着后面更大的元素。
如果最后所有元素都已经入栈了，记得还要输出栈内剩余的元素。