2020 牛客 NOIP 赛前集训营-普及组（第一场）（题解）

T1 牛牛的密码

签到题，暴力模拟，考察语言基础。

100pt

输入字符串循环遍历处理，由于我们知道字符的 ASCII 码中，小写英文字母，大写英文字母，数字，都是连续的。

所以可以设计一个分支结构判断，这样做的好处是可以最后用 else 来表示其他特殊字符。

T2 牛牛的跳跳棋

思维题，简单贪心

20pt

dfs 爆搜之

40pt

如果碰到连续的 1，肯定对最后一个出现的 1 把它变成 2，然后如果后面还是 0，就改成 1。

比如"0 0 1 1 0 1 1 1 0 0 0"修改后变成"1 1 1 2 0 1 1 2 0 1 1"。

（换句话说就是修改后变成连续若干个 1，把末尾的 1 变成 2，后面接一个 0，然后依次类推，最后变成 1 1 1...1 1 2 0...的模式）

100pt

原本不要求修改，质问你能不能到达的话是一个青蛙跳的经典问题。

原题是这样做的，首先贪心的想，往回跳是不必要的，因为如果可以走到第i个

格子，那么必定可以在最后一次跳格子的时候力度小一点走到第$i-1$个格子。

这样的话易知在不修改数字的情况下，从第一个格子开始走可以到达的格子是连续的。

所以这样设计算法，记录一个 r，表示当前能够到达的最靠右的格子。

然后遍历过去，更新$r=\max(p[i]+i,r)$。过程中发现$i>r$的情况就说明不可达，反之说明可达。

本题其实还是相同的思路，只不过在更新右边界的同时，我们还需要知道右边界是谁提供的，一边更新r，同时储存更新右边界的格子 pos。

一旦发现$i>r$的情况，首先让$a[pos]+1$。因为首先 pos 提供了当前右边界，所以当$a[pos]+1$时可以立即更新边界 r，而其他格子即使增加也不一定能够使得边界扩充，这就保证了题目中的“修改次数最少”这个要求，同时由于是顺序遍历，pos是第一个达到右边界r的格子，所以也同时保证了“最小字典序”这个要求。

在更新后我们令$pos=i$，然后继续遍历，这样在算法结束后就得到了一个“最小操作次数并且最小字典序”的操作序列。

T3 牛牛的最大兴趣组

简单数学题

20pt

dfs 爆搜之

40pt

40pt 是一个二分图染色算法。虽然普及组不考图论，但是二分图染色就是个 dfs。

首先两重for循环枚举i,j，如果$i*j$是某个数字的三次方，就把i和j连一条边。

然后对于每个联通块进行二分图染色，假设红色有a个节点，蓝色有b个节点。那么就令$ans+=\max(a,b)$。

100pt

对于任何一个正整数n都可以表示成$a*k^3$的形式，其中 k 是n最大的立方因子。

如果 a 是 n 去掉最大立方因子后的结果，b是 m 去掉最大立方因子后的结果，如果nm是立方数，则ab也是立方数，且a和b是唯一配对的。

例如$2,16,54,686$在去除立方因子后全部等于 2，然后$4,32,108,1372$在去除立方因子后全部等于 4。

所以$\{2,16,54,686\}$中的任意一个数字乘上$\{4,32,108,1372\}$中的任意一个数字都是一个立方数。

一开始输入数字，筛除掉它们的立方因子，然后统计每种数字出现次数。

然后对于每一对出现的种类，都是两两互斥的，所以贪心的取出现次数较多的部分即可。

在实际处理上是有细节的，筛除立方因子可以只枚举立方根以内的质数，这样的话单次去除立方因子的复杂度可以压到 200 次运算左右。

T4 牛牛的滑动窗口

简单数据结构，简单算法题

大概普及 T3,T4 会稍微上一点难度。

一般有个说法，题目标题中出现的算法名都是用来骗人的。

这个题的话，有一半是骗人的吧，写个划窗能拿一半分。

但是 std 也确实是从滑动窗口算法得到的启示。

30pt

暴力三重 for 循环。

50pt

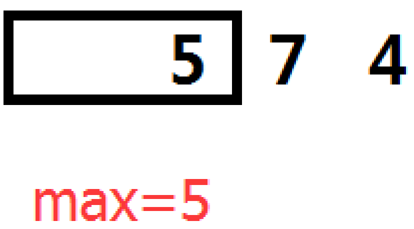
真的写划窗或者 RMQ 啥的优化掉一层 for

100pt

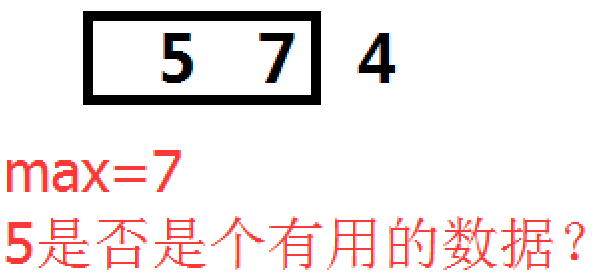
滑动窗口的使用条件正如题目中介绍的那样，需要保证每个查询的端点值是“双单调”的。

实际上划窗算法，或者叫尺取法。的使用条件就是“双单调”。(实际上有部分选手懒得管它到底几个单调，一般看见一个单调就直接二分上了，反正没人卡 log)

首先需要掌握一个技巧，“单调队列$O(n)$求滑动窗口极值”，这个算法。



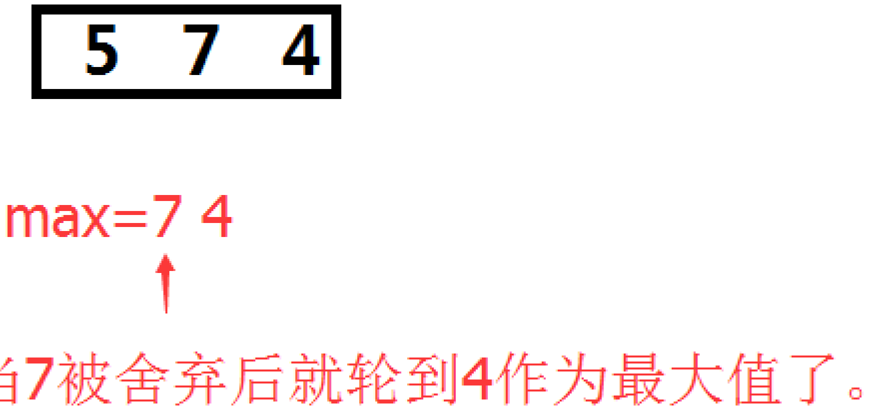
如图，假设这是一个用划窗扫描数组 5,7,4...的场景，假设求最大值。



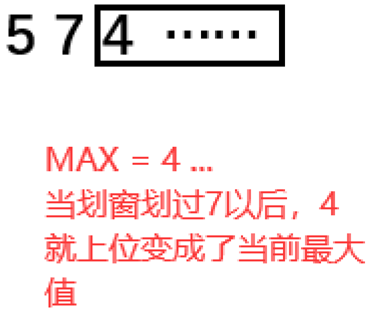
在继续扫描到 7 的时候，我们想一下 5 这个数据有没有用。

因为是滑动窗口求最大值，当前的最大值已经是 7 了，它有没有可能再变回 5?

显然不可能，因为滑动窗口要求端点递增，所以 5 将会先比 7 被舍弃。



接下来继续扫描，4 要不要保留，显然保留，因为根据滑动窗口的定义，7 一定比 4 先被舍弃。



然后当 7 被划窗划过后，4 上位变成当前最大值。

有了这个算法以后，我们开两个单调栈，分别维护最大值和最小值，然后从左到右遍历过去。

这样的话就可以知道当 r 单调向右滑动的时候，l 端点落在不同的位置时最大值最小值各是多少。

由于值域不超过 100，那么栈内的元素数目肯定是不会多于 100 的。

所以我们可以暴力穷举栈内元素。

然后因为两个单调栈都是有序的，在这里再次借助双指针划窗。提取出$max*min$相同的区间长度是 L 到 R。

使用差分维护一个数组a，$a[k]$表示区间长度为k的答案。

然后将使用差分将$a[L]$~$a[R]$这段区间全部加$max*min$即可。

最后对差分数组求前缀和，就得到了每一个区间长度的答案。