Leo 的次幂计算

欧拉降幂模板题。

在对于一个数的幂次方大于模数的时候可以用欧拉降幂，否则快速幂都拯救不了世界，我也不知道为什么有这个东西，但是好用就用了呗，在莫比乌斯反演的时候也用到了这玩意。

这道题怎么用，我们设 $g(d)=xd+y$,那么答案就是 $f(d)=g(d)\%d=y$

因为所给的题目是 7 的无穷次，所以 $g=7^g$

而 7 的无穷次就可以用欧拉降幂 $f(d)=7^{g(d)\%\phi(d)+\phi(d)}\%d$

而 $g(d)\%\phi(d)==f(\phi(d))$

那么 $f(d)=7^{f(\phi(d))+\phi(d)}\%d$

这样就可以用递归做，在模数为 1 的时候就返回 0 了

Std：https://paste.ubuntu.com/p/Dn7g72WKHn/

Leo 的泡泡

栈的基础运用。

开一个 char 字符的栈，每次匹配 s[i]位置与栈顶是否相同，相同的话把栈顶出栈，并根据大小泡泡决定是否要再比较一次，当 s[i]与栈顶不同时把 s[i]压入栈中即可。

17 级的没做出来估计是上学期数据结构没听课。

Std：https://paste.ubuntu.com/p/tz7Ks7QQB6/

买水果

用线段树或树状数组，叶节点维护每种价格水果的总价，从编号为 1 的水果到编号为 n 的水果，每次查询小于当前水果单价减 1 的总价，再给当前单价的水果总价加上单价乘数量即可

Std：https://paste.ubuntu.com/p/NMD2x68CJQ/

罗 dalao 的树 2.0

树形 dp，以编号为 1 的节点为根(选别的也行)，sz[i]表示以 i 为根节点的子树的节点数量，以 i 节点为界将树分成上下两部分，dp[0][i]表示以 i 为根节点的子树的贡献，dp[1][i]表示剩下的那部分的贡献，于是

dp[0][u]=sigma(dp[0][v]+sz[v])
dp[1][v]=dp[1][u]+dp[0][u]-2*sz[v]-dp[0][v]+n
两次 dfs 即可
Std：https://paste.ubuntu.com/p/TfCG2FrcfP/

Leo 的幸运数字
简单模拟题，按题意模拟即可，当判断到当前数字操作后得到的数字之前出现过，即可判断进入了循环节，此时直接输出即可。
事实上，百度 6174 即可查到 6174 猜想，即所有四位不全相同的四位数这么操作至多 7 次以后，必定得到 6174，而 6174 操作后得到本身，所以这题判断到重复即输出即可，不会超时。
Std：https://paste.ubuntu.com/p/DDRNxQDm8M/

罗 dalao 的树 3.0
裸题，先求树的 dfs 序，于是每个子树的节点序号就是连续的，然后就变成了一个单点更新区间查询的问题，可以用树状数组套主席树解决
std：https://paste.ubuntu.com/p/qYwBKz88gM/

累加求和
首先将 $\sum_{i=1}^{m}\sum_{j=1}^{m}\left\lfloor\frac{m}{j}\right\rfloor^{i}$ 拆分成如下形式：

$$\left\lfloor\frac{m}{1}\right\rfloor^{1}+\left\lfloor\frac{m}{2}\right\rfloor^{1}+\left\lfloor\frac{m}{3}\right\rfloor^{1}+\ldots\ldots+\left\lfloor\frac{m}{m}\right\rfloor^{1}$$

$$\left\lfloor\frac{m}{1}\right\rfloor^{2}+\left\lfloor\frac{m}{2}\right\rfloor^{2}+\left\lfloor\frac{m}{3}\right\rfloor^{2}+\ldots\ldots+\left\lfloor\frac{m}{m}\right\rfloor^{2}$$

$$\left\lfloor\frac{m}{1}\right\rfloor^{3}+\left\lfloor\frac{m}{2}\right\rfloor^{3}+\left\lfloor\frac{m}{3}\right\rfloor^{3}+\ldots\ldots+\left\lfloor\frac{m}{m}\right\rfloor^{3}$$

……

$$\left\lfloor\frac{m}{1}\right\rfloor^m+\left\lfloor\frac{m}{2}\right\rfloor^m+\left\lfloor\frac{m}{3}\right\rfloor^m+\ldots\ldots+\left\lfloor\frac{m}{m}\right\rfloor^m$$

如果竖着看的话可以发现：

$$\left\lfloor\frac{m}{1}\right\rfloor^1+\left\lfloor\frac{m}{1}\right\rfloor^2+\left\lfloor\frac{m}{1}\right\rfloor^3+\ldots\ldots+\left\lfloor\frac{m}{1}\right\rfloor^m$$

和

$$\left\lfloor\frac{m}{2}\right\rfloor^1+\left\lfloor\frac{m}{2}\right\rfloor^2+\left\lfloor\frac{m}{2}\right\rfloor^3+\ldots\ldots+\left\lfloor\frac{m}{2}\right\rfloor^m$$

……

$$\left\lfloor\frac{m}{m}\right\rfloor^1+\left\lfloor\frac{m}{m}\right\rfloor^2+\left\lfloor\frac{m}{m}\right\rfloor^3+\ldots\ldots+\left\lfloor\frac{m}{m}\right\rfloor^m$$

都是等比数列。

那么，我们就可以根据整数分块理论和等比数列求和公式来 AC 这道题了。

由于 m 比较大，注意计算过程中超出 long long 所表示的数据范围的情况还有等比数列的公比为 1 的情况。

std: https://paste.ubuntu.com/p/nK3dwRcJV7/

zzh 找数字

建一个标记数组，记录 a 和 b 中出现过的数字，然后遍历 0 到 1000，遍历的同时拆数，检测每一位是否在 ab 中出现过，如果某数所有位上的数字都没出现过，那么结果+1

注意特判 a 或者 b 为 0 的情况即可。

Std：https://paste.ubuntu.com/p/w9r9TbV6sQ/