1-0刷题方法与总结总览

TODO:
强化动态规划！！！问题抽象为**dp**的思路

评价k神：有一说一，只看大佬思路，大佬代码就别了，经过优化的，对新手不太友好

大家知道，递归虽然简洁，但会产生额外的空间开销。我们可以把递归改写为循环，来避免对栈空间的大量占用，也就是非递归快速幂。

字符串/数组：
状态机——LCR 138. 有效数字

双指针：

查找算法：
hash，顺序hash，原地hash。
标准二分法代码背诵。二分法过程中的多情况灵活变化。
二叉搜索树的变体。

分治算法、动态规划、贪心算法的对比：
In a word, 分治法 —— 各子问题独立；动态规划 —— 各子问题重叠。

对许多最优化问题来说，采用动态规划方法来决定最佳选择有点“杀鸡用牛刀”了，只要采用另一些更简单有效的算法就行了。贪心算法是使所做的选择看起来都是当前最佳的，期望通过所做的局部最优选择来产生出一个全局最优解。贪心算法对大多数优化问题来说能产生最优解，但也不一定总是这样的。
贪心算法只需考虑一个选择（亦即，贪心的选择）；在做贪心选择时，子问题之一必须是空的，因此只留下一个非空子问题。
贪心算法与动态规划与很多相似之处。特别地，贪心算法适用的问题也是最优子结构。贪心算法与动态规划有一个显著的区别，就是贪心算法中，是以自顶向下的方式使用最优子结构的。贪心算法会先做选择，在当时看起来是最优的选择，然后再求解一个结果子问题，而不是先寻找子问题的最优解，然后再做选择。

贪心算法是通过做一系列的选择来给出某一问题的最优解。对算法中的每一个决策点，做一个当时看起来是最佳的选择。这一点是贪心算法不同于动态规划之处。在动态规划中，每一步都要做出选择，但是这些选择依赖于子问题的解。因此，解动态规划问题一般是自底向上，从小子问题处理至大子问题。贪心算法所做的当前选择可能要依赖于已经做出的所有选择，但不依赖于有待于做出的选择或子问题的解。*因此，贪心算法通常是自顶向下地做出贪心选择，不断地将给定的问题实例归约为更小的问题。贪心算法划分子问题的结果，通常是仅存在一个非空的子问题。*

位运算

">>>"无符号右移
操作规则：无论正负数，前面补零。

">>"右移
操作规则：正数前面补零，负数前面补1

"<<"左移
操作规则：无论正负数，后面补零。

加法器的操作，通过【异或】和【与】 来实现。异或是低位，与是进位。

其他OJ技巧

大数取模 **10^9 + 7**
大数的排列组合等，一般都要求将输出结果对1000000007取模 为什么总是1000000007呢= =大概=_=✧是因为：（我猜的，不服你打我呀~）
1. 1000000007是一个质数
2. int32位的最大值为2147483647，所以对于int32位来说1000000007足够大
3. int64位的最大值为2^63-1，对于1000000007来说它的平方不会在int64中溢出 所以在大数相乘的时候，因为(a*b)%c=((a%c)*(b%c))%c，所以相乘时两边都对1000000007取模，再保存在int64里面不会溢出 。◕‿◕。

输入输出解析：
https://blog.csdn.net/wuwenbin12/article/details/126382811
用c++写代码，输入一个字符串，代表是一个数组， 每个数用逗号隔开，逗号和数直接有任意数量的空格，读取数据保存到vector<int>中,怎么写。

快速幂
作用：可以将将数幂的计算复杂度从普通的 O(n) 降低到 O(logn)，其中 n 是指数的位数。
思路：https://oi-wiki.org/math/binary-exponentiation/

```
1
```