《刷题总结》

一. 数据结构

1.顺序表

1.1 数组（字符串）

001. 66 题-002-替换空格

要点：根据规律对字符串中元素进行移位

002. 66 题-013-调整数组顺序使奇数位于偶数前面

要点：数组的遍历与重构

003. 66 题-028-数组中出现次数超过一半的数字

要点：**哈希表**+数组遍历
技巧：用哈希表记录数字出现次数，空间换时间。

004. 66 题-032-把数组排成最小的数

要点：sort 的妙用
技巧：当两个 int 类型数据进行拼接的时候，最好转为 string 后再拼接，防止拼接后的数据越界。

005. 66 题-034-第一个只出现一次的字符

要点：**哈希表**+数组遍历
技巧：用哈希表记录字符出现次数，空间换时间，同 003。

006. 66 题-041-和为 S 的连续正数序列

要点：**双指针**构造一个滑动窗口
技巧：依据数组排序的特性使用双指针来构造滑动窗口。

007. 66 题-042-和为 S 的两个数字

要点：**双指针**遍历数组
技巧：依据数组排序的特性使用双指针来遍历数组，同 006，还有两个数相差越大乘积越小。

008. 66 题-043-左旋转字符串

要点：字符串的多次**翻转**
技巧：翻转时将双指针的移动和换位融合，如：swap(str[left++], str[right--])。

009. 66 题-044-翻转单词顺序列

要点：字符串的多次**翻转**
技巧：翻转时将双指针的移动和换位融合，如：swap(str[left++], str[right--])，如 008。
特例：只有一个单词的情况，例如‘Yes’

010. 66 题-049-把字符串转换成整数

要点：字符串遍历+int **溢出**的判断
技巧：在构建数字的时候判断是否溢出的技巧
```
if ((res < (limit / 10)) || (res == (limit / 10) && num <= (limit % 10)))
```

011. 66 题-050-数组中重复的数字

要点：用哈希表(**本题中用了数组本身做哈希表**)空间换时间
技巧：长度为 n 的数组里数字的范围保证在 0-n-1 之间，所以可以利用现有数组来记录一些信息。

012. 66 题-051-构建乘积数组

要点：数组计算转化为**矩阵**计算

013. 66 题-053-表示数值的字符串

要点：将字符串按照‘e’**分割为两段**之后进行字符串的判断

014. 66 题-054-字符流中第一个不重复的字符

要点：**哈希表**空间换时间
技巧：(1)字符为键值的哈希表可以用 int a[256]来替代，因为一个字符占一个字节(即 8 位)，1 个字节能表示的信息就是 2^8 个，即 256 个。
(2)字符作为数组下标时，其表示的下标值为该字符的 ASCII 码的十进制值。

1.2 链表

001. 66 题-003-从尾到头打印链表

要点：链表的遍历

002. 66 题-014-链表中倒数第 k 个结点

要点：变形的快慢指针
技巧：输入的参数 k 为 0 时，由于 k 是一个无符号整数(unsigned int)，那么 k-1 得到的将不是-1,而是 4294967295(无符号的 0xFFFFFFFF),要注意该种情况。

003. 66 题-015-反转链表

要点：链表的头插法

004. 66 题-016-合并两个排序的链表

要点：链表的遍历

005. 66 题-025-复杂链表的复制

要点：**随机指针**的处理
技巧：将每个节点的复制节点都插入其之后，方便复制节点随机指针的赋值。

006. 66 题-036-两个链表的第一个公共结点

要点：两个链表**对齐**之后遍历

007. 66 题-055-链表中环的入口结点

要点：**快慢指针**判断是否存在环以及找环的入口

008. 66 题-056-删除链表中重复的结点

要点：添加**头指针**+链表的遍历

2. 二叉树

001. 66 题-004-重建二叉树

要点：前序+中序重建二叉树
技巧：假设**前序序列**中一节点下标为 index，
由其在**中序序列**中的位置可知其左子树的节点个数为 len，
则可知在**前序序列**中其左孩子节点下标为 index+1，其右孩子节点下标为 index+1+len。

002. 66 题-017-树的子结构

要点：DFS+前序遍历，子结构 T 可以有子结构 B（被求子结构）没有的节点，但是子结构 B 不可以有子结构 T 没有的节点

003. 66 题-018-二叉树的镜像

要点：前序遍历

004. 66 题-022-从上往下打印二叉树

要点：层次遍历
技巧：如果形参传入的变量在**使用过后就不再需要**，而后面又需要使用辅助变量，可以直接把形参变量当辅助变量用。

005. 66 题-023-二叉搜索树的后序遍历序列

要点：二叉搜索树的特点+分治
技巧：要巧妙利用二叉搜索树的特点，左子树均小于根节点，右子树均大于根节点。二叉搜索树的后序遍历序列 S，最后一个元素是 x(也就是根)，如果去掉最后一个元素的序列为 T，那么 T 满足：T 可以分成两段，前一段(左子树)均小于 x，后一段(右子树)均大于 x，而前序序列反之。

006. 66 题-024-二叉树中和为某一值的路径

要点：DFS+前序遍历

007. 66 题-026-二叉搜索树与双向链表

要点：二叉搜索树特点+记忆化中序遍历

008. 66 题-038-二叉树的深度

要点：后序遍历求树深度

009. 66 题-039-平衡二叉树

要点：后序遍历求树深度+剪枝

010. 66 题-057-二叉树的下一个结点

要点：根据中序遍历的规则寻找节点
特例：树：[8,6,10,5,7,9,11] 节点：7
pNode 为其根节点的右子树，需要向上寻找，看 pNode 是否在某个节点的左子树上，如果存在这样一个节点那么 pNode 的下一个节点为该节点

011. 66 题-058-对称的二叉树

要点：由上至下+由外而内的前序遍历

012. 66 题-059-按之字形顺序打印二叉树

要点：二叉树的层次遍历

013. 66 题-60-把二叉树打印成多行

要点：二叉树的层次遍历

014. 66 题-061-序列化二叉树

要点：二叉树的前序遍历
特例：反序列化时输入的字符串一定要使用**引用**，这样才能保证字符串一直保持向后遍历的状态。

015. 66 题-062-二叉搜索树的第 k 个结点

要点：二叉搜索树的中序遍历+中序遍历的终止

3. 栈与队列

001. 66 题-005-用两个栈实现队列

要点：用栈实现队列的功能

002. 66 题-020-包含 min 函数的栈

要点：用一个最小元素辅助栈来保存栈中的最小值

003. 66 题-021-栈的压入、弹出序列

要点：模拟进出栈的过程

004. 66 题-064-滑动窗口的最大值

要点：双端队列模拟滑动窗口找最大值
技巧：（1）在用 for 循环遍历 vector 之类的容器时，如果在遍历过程中有删除元素的操作，**那么一定要注意此时可能会漏掉遍历一些元素**，因为删除元素之后，后面的元素顶了上来，而此时遍历的 i 已经向后移动了，例如 vector a{1,2,3,4,5}，i=2,删除了 3，此时 a{1,2,4,5}但 i=3,略过了 4，直接遍历 5。
（2）当 int 和 unsigned in 相加减时，**会将 int 转化为 unsigned int，且结果也转化为 unsigned int**，当结果小于 0 时，int 小于 0，所以

它的最高位是符号位为 1，所以转化的结果是一个很大的正数

4. 图

001. 66 题-065-矩阵中的路径

要点：DFS
技巧：（1）图是二维数组时，记录节点是否被遍历过的标志数组可以用一维数组来表示，flag[x][y]可以表示为 flag[x * cols + y]。

002. 66 题-066-机器人的运动范围

要点：DFS/BFS
技巧：（1）图是二维数组时，记录节点是否被遍历过的标志数组可以用一维数组来表示，flag[x][y]可以表示为 flag[x * cols + y]。

5. 堆

001. 66 题-029-最小的 K 个数

要点：大顶堆
特例：(1)k==0 时会造成堆为空的现象

002. 66 题-063-数据流中的中位数

要点：大顶堆+小顶堆分别保存数据

二. 算法

1.动态规划

001. 66 题-007-斐波那契数列

要点：动态规划版的斐波那契数列。

技巧：（1）斐波那契问题如果使用递归可能会爆栈且会产生过多重复计算。
（2）不使用中间变量进行斐波那契公式：

```
tmp2 += tmp1;
tmp1 = tmp2 - tmp1;
```

002. 66 题-008-跳台阶

要点：动态规划版的斐波那契数列。
技巧：同 001。

003. 66 题-010-矩形覆盖

要点：动态规划版的斐波那契数列。
技巧：同 001。

004. 66 题-030-连续子数组的最大和

要点：简单线性动态规划。

005. 66 题-067-剪绳子

要点：简单线性动态规划。
特例：在动态规划的过程中，除了最后的目标长度，其余长度可以一刀不剪

2.分治

001. 66 题-012-数值的整数次方

要点：分治求幂，5->2+3->(1+1)+(1+2)->(1+1)+(1+(1+1))。
特例：幂为负数的情况

3.递归

001. 66 题-009-变态跳台阶

要点：递归版的斐波那契数列。

002. 66 题-027-字符串的排列

要点：含重复元素的全排列。
技巧：(1)用递归全排列之后是非字典序的，如果想要字典序的全排列可以用 sort 对得到的序列进行排序。
(2) 也可以直接用库函数 next_permutation 来做。
特例：要注意将原始字符串这一种情况保存到排列当中

003. 66 题-047-求 1+2+3+...+n

要点：利用&&短路的递归。

004. 66 题-052-正则表达式匹配

要点：用递归排列各种情况并匹配答案。
特例：(1)str==“” pattern==“.*”
(2)str==“” pattern==“.”

4.搜索与排序

001. 66 题-001-二维数组中的查找

要点：依照规律进行查找。

002. 66 题-006-旋转数组的最小数字

要点：二分查找的应用。
技巧：如果面试题是要求在排序的数组（或者部分排序的数组）中查找一个数字或者统计某个数字出现的次数，我们都可以尝试用二分查找算法。

003. 66 题-028-数组中出现次数超过一半的数字

要点：使用快排思想解决问题。
技巧：快速排序算法中的函数 getPartition，还可以用来实现在长度为 n 的数组中查找第 k 大的数字，记得 getPartition 函数形参中的数组要加引用，这样才能完成对原数组的的变换。
特例：数组中只有一个数字，例如{1}

004. 66 题-029-最小的 K 个数

要点：使用快排思想解决问题。
技巧：快速排序算法中的函数 getPartition，还可以用来实现在长度为 n 的数组中查找第 k 大的数字，记得 getPartition 函数形参中的数组要加引用，这样才能完成对原数组的的变换。

005. 66 题-035-数组中的逆序对

要点：基于归并排序求逆序对。
技巧：归并排序中可以在递归过程中通过交替变换原数组和辅助数组在形参中的位置，来省略常规归并排序中还需要将每层分段排序好的数组再赋给原数组的操作。

006. 66 题-037-数字在排序数组中出现的次数

要点：二分查找寻找数组中数据的起始位置。
技巧：如果面试题是要求在排序的数组（或者部分排序的数组）中查找一个数字或者统计某个数字出现的次数，我们都可以尝试用二分查找算法，同 002。

5.数学

001. 66 题-009-变态跳台阶

要点：数学归纳找式子之间的规律，也可以理解为隔板问题。

002. 66 题-012-数值的整数次方

要点：快速求幂。

003. 66 题-031-整数中 1 出现的次数(从 1 到 n 整数中 1 出现的次数)

要点：数学归纳。

004. 66 题-033-丑数

要点：数学归纳+特殊数字(包含特殊的因子)。

005. 66 题-046-孩子们的游戏(圆圈中最后剩下的数)

要点：约瑟夫环问题。

6.位运算

001. 66 题-011-二进制中 1 的个数

要点：正负数的算术移位区别。

002. 66 题-040-数组中只出现一次的数字

要点：异或的使用。
技巧：当题目中出现某些数出现偶数次，可以考虑异或的特性将其消除。

003. 66 题-048-不用加减乘除做加法

要点：位运算实现加减法。

7.模拟算法

001. 66 题-019-顺时针打印矩阵

要点：从外向里一圈一圈打印模拟顺时针打印。
特例：(1)单行：{{1,2,3,4,5}}
(2)单列：{{1},{2},{3},{4},{5}}

002. 66 题-045-扑克牌顺子

要点：数组模拟扑克牌序列
特例：{0,0,1,1}牌中出现相等的牌

003. 66 题-046-孩子们的游戏(圆圈中最后剩下的数)

要点：链表(便于删除节点)模拟报数过程