|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Two Sum | 在一个数组中找到两个数使其相加等于一个给出的m | O(n^2) 直接扫描  O(n) Hash判m-a[i]是否存在及其位置 |
| 3 | Longest Substring Without Repeating Characters | 求没有重复字符的最长子串长度 | O(n) 从左向右扫开数组count记录每个字符之前最后出现在哪里，用last作标记表示当前合法位置，若当前字母为x，若count[x]>last则更新last标记，维护最大合法答案于ans |
| 4 | Median of Two Sorted Arrays | 求两个有序序列的中位数 | O(log^2(n+m)) 假设中位数在序列A中，二分其位置对于A[i]，二分得到B中小于A[i]的个数，若中位数在B中，同理  O(log(n+m)) 求中位数，可看作将A，B数组各分作两段，A中以tl为界，B中以tr为界。tl/tr左半段划入小于中位数，右半段划入大于中位数，且两侧数量一致。观察找到中位数时的划分。必然有A[tl-1]<=B[tr]且A[tl]>=B[tr-1]。  首先二分tl，可以由划分两侧数量一致直接求得对应tr，若此时A[tl-1]>B[tr]，则可证正确的tl’必然在当前tl左边。若此时A[tl]<B[tr-1]，则可证正确的tl’必然在当前tl右边。满足二分性质，故O(log(n+m))可求。 |
| 5 | Longest Palindromic Substring | 求一字符串的最长回文子串 | O(n) 扩展kmp  为什么O(n)？  揣测：扩展kmp的复杂度集中在以i为中心向外的扩展一定长度时，而当i扩展f[i]的长度时。将[i+1,i+f[i]]内的坐标扩展时直接得到答案或扩展至i+f[i]的坐标位置。故总体的均摊后复杂度为O(n) |
| 10 | Regular Expression Matching | 简单表达式匹配，’.’可匹配任意字符，’\*’表示将前一个字符重复任意次（包括0次，’a\*’可表示’’空串），这两个符号仅会出现在B串中，问AB串能否完全匹配。  ‘.\*’可表示’ab’，即’\*’可拷贝’.’而非仅能拷贝’.’匹配后的字符。 | O(n^2) DP，以f[i][j]表示A串至第i位，B串至第j位能否完成匹配。 |
| 11 | Container With Most Water | 给出height数组，构建n个线段(i,height[i])，找到两条线段使其与x轴形成的“容器”能装尽可能多的水 | O(nlogn) 对于i号段，若以其为右边界且为答案则需要找到尽可能靠左的且高度大于i处高度的位置，维护一个在i左边且高度递增的序列，在上面二分得到所需边界。  O(n) 考虑横轴最大的情况，即选择最左最右的边界。若要找到更优的答案，由于横轴的减小高度必然更高，故缩小lr边界即l右移r左移直至两者都比当前容器的高度更高。此时才有可能构成答案。由于lr边界一直向内收缩，故效率为O(n) |
| 12 | Integer to Roman | 将[1,3999]的数字转换成罗马数字输出 | 挺不错的划分 |
| 14 | Longest Common Prefix | 给出一些字符串，求他们的最长公共前缀 | 枚举答案长度，直接判断数组中的n个字符串在当前长度位置i上的字符是否相同。  两点：  1数组是否为空  2数组中是否存在空串，空串string长度为1而非0 |
|  |  |  |  |
| 19 | Remove Nth Node From End of List | 删除一个链表中倒数第n个节点 | 设置两个指针。分别为快慢指针。先让慢指针沿链表走n步，然后快慢指针同时向后走，当快指针走到链表尾时，慢指针也就找到了链表中倒数第n个元素。 |
| 20 | **Valid Parentheses** | 判断一个括号表达式是否合法 | 栈的使用。若当前遇到一个左括号将其对应的右括号压入栈中，若当前遇到一个右括号，判断栈顶的右括号是否与其一致。 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |