|  |  |  |  |
| --- | --- | --- | --- |
| 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) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |