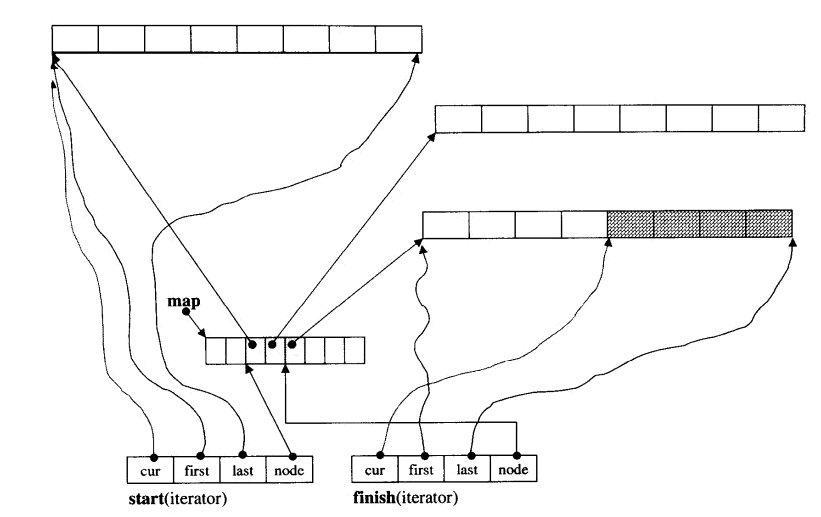
1. 各种排序算法的精髓思想：
2. 快速排序：快速排序首先是一个分支策略，每次都将一个数组划分成两个，然后递归的进行排序。划分的过程中，取一个元素为对比元素，每次划分完以后这个元素就放在属于他的位置，并且在每次排序过程中都设置两个指针i和j，指针i指示的是一个界限，这个界限的左边都是小于对比元素的，j是一个遍历指针，i和j中间的都是大于对比元素的。因此，在j进行遍历过程中，如果发现有小于对比元素的，就把他跟i+1进，因为i+1肯定是大于对比元素的，这样就能把大于对比元素和小于对比元素的用i划分开。  
   性能分析：如果每次划分都把原数组划分成n-1:0，则算法的时间复杂度为T(n)=T(n-1) + O(n),可得时间复杂度为O（n2）。可用树来计算：第一层为n，第二层为n-1，每一层的时间复杂度都是划分的时间n，共n层，所以复杂度为O（n2）。最佳情况出现在只要不是上述划分就可以保证时间复杂度为O（nlgn），也用树来计算，n个节点的二叉树有lgn层。  
   如果排序要求数组左边和右边不属于同一种情况，或者需要找出大于某元素或小于某元素的情况，可以考虑快速排序。
3. 算法题：
4. 求数组中的第K小元素。利用快速排序的思想，每一趟快速排序都可以确定比一个对比元素大和小的所有元素，如果比对比元素小的元素个数大于K则显然所求元素应该在小的那边，否则在大的那边，如果在小的那边，直接在小的那边进行快速排序，否则在大的那边求第K-sizeof（小）的元素。
5. 求逆序对。可以采用合并排序的思想。可以证明一个数组中的逆序对的个数等于两个子序列中的逆序对个数之和，再加上前面子数组中元素个数比后面子数组大的个数。合并排序在每一趟中都能确定若干个有序子序列，将后面的子序列与前面的子序列合并的过程，实际就是将一个元素插入到有序序列中的过程，而后者的本质是确定在有序序列中比插入元素大或者小的元素个数，这样就可以确定以插入元素为尾的逆序对个数，从而求得全部的逆序对。易得，插入排序和合并排序都可以求逆序对，时间复杂度与进行排序的复杂度一样。
6. 另类算法题：
7. 删除字符串中的整数。从字符串中的第一个整数位置开始，依次放入后面不是整数的字符。
8. 求两个字符串中第一个最长子串。
9. 不开辟新空间完成字符串的逆序。两个字节互换的算法。a^=b,b^=a,a^=b;
10. 随机排列数组。方法一，给每个元素一个随机数，然后按随机数进行排序。方法二，顺序排列，将A[i]与random(i,n)进行交换。
11. 用trie树和后缀树处理字符串问题
12. trie树：是将一个或多个字符串按共同前缀的形式构成的一棵树。在这棵树中，可以快速确定给定字符串是否是某字符串从0位置开始的一个子串。
13. 后缀树：后缀树是将一个或多个字符串，及其所有后缀形式的字符串组成一棵trie树。可以快速确定给定字符串是否是某字符串从任意位置开始的一个子串。
14. 一维向量常用算法：
15. 首先看问题是否能用dp算法解决。  
    步骤：  
    首先假设一个最优解，然后证明这个解具有最优子结构性质：  
    最优解有以下几种构造方法：构造一个长度为n的01序列，1代表不在一个解中选择a[i]，0代表不选择a[i]（背包类问题）（回溯剪枝），这通常是要求选择若干个元素构成具有某种性质的结果。  
    然后由子问题构造原问题：  
    子问题的构造方法：以a[i]为结尾的某个序列、在确定是否选择a[i]时。。。、在前由前i个元素组成的子数组中、在元素i到元素j组成的子数组中---这类问题一般具有加全括号性质、  
    然后确定边界条件。
16. 再看是否能用分治算法解决。  
    在某个地方将数组劈开，原问题的解等于两个子数组的解，再加上由两个子数组构成的解。
17. 再看排序思想能不能适用。  
    首先考虑原问题能不能转化成出现次数的问题，如果能可以使用计数排序的内哈希思想，在扫描一遍数列后，可确定每个元素出现的次数，然后再根据题意进行下一步思考。  
    再考虑能不能进行快速排序的思想，快速排序的思想一般用于选择前n个最大或最小数。  
    （
18. 注意有序与无序的区别。在数组有序的情况下，在查找过程中可以进行使用二分查找思想，在分段有序的情况下（考虑是否可以把大的数组分段，然后分段排序，将每段的最大值拿出来做索引，进行查找）可以进行分块查找。还可以使用两个游标分别从数列两段向中间遍历，根据不等式的性质进行判断或者剪枝。
19. 注意逆序是否可用。整体逆序和部分逆序的结合，如将一个数组循环右移K为，可将前n-k个元素逆序，后K个元素逆序，然后整个数组逆序。
20. 堆的高级性质。如果用最大堆维护一个数组，可以快速返回数组中的最大元素，如果数组元素不能交换，可以设置一个下标数组，对下标数组进行堆的维护。
21. 数据结构：
22. vector实现的是一种线性表，分配连续的存储空间，类似数组，具有快速随机访问的优点，缺点是要动态分配内存，开销比较高（涉及到，申请新内存，将数据拷贝到新内存，释放旧内存），而且只能pushback。
23. list也是一种线性表，分配的不是连续的空间，具有快速增删的优点。可以快速在任意节点进行增加和删除，但是访问的时候效率不高。
24. deque也是一种线性表，兼容了vector与list的优点，既可以提供快速随机访问，又可以快速pushfront，而且空间不足后可以申请内存而不用进行vector那样的开销，代价是用另一个数组来维护所有的类似vector的节点。  
    
25. stack是栈，是一种概念，其底层的实现方式可以使vector，或者deque，或者是list。
26. queue是队列，也是一种概念，其底层可以是deque和list。
27. 堆heap，可以实现优先级队列，不管进heap的元素如果，取出来的一定是优先级最高的，即值最大的。
28. Set,map底层都可以以红黑树实现，set红黑树中存的是value，map红黑树中存的是<key,value>，key为红黑树排序的关键。
29. Hashtable,散列表，将key进行散列，并用各种方法解决冲突。Set和map也可以以hashtable的方式实现，对应的结构称作hashset，hashmap，即set实值以hash的方式存储，map的key以hash的方式存储，前提是set的实值和map的key都可以通过某种方法转换成可以进行模运算的类型。
30. 求一个整数的有效比特位数：（查表）

