[双指针法的常见应用](http://blog.csdn.net/ohmygirl/article/details/7850068)

 所谓双指针，指的是在遍历对象的过程中，不是普通的使用单个指针进行访问，而是使用两个相同方向或者相反方向的指针进行扫描，从而达到相应的目的。

    换言之，双指针法充分使用了数组有序这一特征，从而在某些情况下能够简化一些运算。

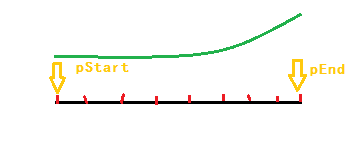
    (注：这里的指针，并非专指c中指针的概念，而是指索引，游标或指针，可迭代对象等)

    这类算法包括：

    1.给定一个有序递增数组，在数组中找到满足条件的两个数，使得这两个数的和为某一给定的值。如果有多对数，只输出一对即可。

    对于这种问题，常见的算法思路不外乎遍历，回溯，但这里，双指针遍历法是一个很有效的方法。具体思路是：初始化两个指针，一个指向数组的第一个元素，另外一个指向数组的最后一个元素，在两个指针相遇之前，指针1只能向前移动，指针2 只能向后移动。比较当前两个指针所指元素和与给定数字的大小，如果和较大，指针2向后移动，如果和较小，指针1向前移动。最终的结果是找到两个满足条件的数或者不存在这样的两个数字。

     如图所示：



据此思路，不难写出代码：

**[cpp]** [view plaincopy](http://blog.csdn.net/ohmygirl/article/details/7850068)

1. #include <stdio.h>
3. **int** getThePair(**int** \*a,**int** n,**int** theNum,**int** &left,**int** &right){
4. **int** start = 0,end = n-1;
5. **while**(start < end){
6. **int** sum = a[start] + a[end];
7. **if**(sum == theNum){
8. left = a[start];
9. right = a[end];
10. **return** 1;
11. }
12. **if**(sum < theNum){
13. start ++;
14. }**else** **if**(sum > theNum){
15. end --;
16. }
17. }
18. **return** 0;
19. }
21. **int** main(){
22. **int** a[] = {1,2,3,4,5,6,7,8};
23. **int** start = 0,end = 0;
24. **for**(**int** i = 1; i <= 16; i++){
25. **if**(getThePair(a,8,i,start,end)){
26. printf("the sum of %d is %d + %d\n",i,start,end);
27. }**else**{
28. printf("sorry,%d not found~~\n",i);
29. }
30. }
31. **return** 0;
32. }

2.双指针应用之二：hoare的双向扫描快速划分法。快速排序的总结中已经给了思路和代码，为了方便，这里再次给出代码：

**[cpp]** [view plaincopy](http://blog.csdn.net/ohmygirl/article/details/7850068)

1. **int** hoare\_partition(**int** arr[],**int** start,**int** end){
2. **int** pivot = arr[start];
3. **int** i = start;
4. **int** j = end;
5. **while**(**true**){
6. **while**(i<j && arr[i]<=pivot) i++;
7. **while**(i<j && arr[j]>=pivot) j--;
8. **if**(i<j){
9. swap(&arr[i],&arr[j]);
10. }
11. **else** **return** j;
12. }
13. }

3.双指针应用之三。

奇偶排序。忘记是哪个公司的面试题了。题目大意是这样的，给定一个数组，数组中元素有奇数有偶数。要求对数组进行处理，使得数组的左边为奇数，右边为偶数

这个题目直接看上去，跟快速排序的划分十分类似，唯一不同的是，不需要返回索引。所以，简单的方式如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/ohmygirl/article/details/7850068)

1. **void** partition(**int** \*arr,**int** n){
2. **int** i = 0;
3. **int** j = n-1;
4. **while**(1){
5. **while**(i<j && arr[i]%2 == 1) i++;
6. **while**(i<j && arr[j]%2 == 0) j--;
7. **if**(i<j){
8. **int** temp = arr[i];
9. arr[i] = arr[j];
10. arr[j] = temp;
11. }
12. }
13. }

4.双指针法的应用之四：求单链表的中间元素。

这个大家应该都比较熟悉了。对于单链表求中间元素的问题，经典的作法是采用两个指针，初始化都指向链表表头，移动开始时，快指针移动两步，慢指针移动一步。当快指针的next为null或者快指针为null的时候，慢指针所指的就是单链表的中间元素(注意奇数个元素和偶数个元素的处理)

**[cpp]** [view plaincopy](http://blog.csdn.net/ohmygirl/article/details/7850068)

1. /\*
2. \* 取单链表的中间元素，如果单链表元素为奇数个，那么去中间的元素，如果是偶数个，取后面的一个元素。
3. \*/
4. linkList getMidElem(**const** linkList L){
5. **if**(L == NULL){
6. **return** NULL;
7. }
8. linkList first = L;
9. linkList second = L;
10. **while**(first != NULL && first->next != NULL){
11. first = first->next->next;
12. second = second->next;
13. }
14. **return** second;
15. }

完整的代码可见：http://blog.csdn.net/ohmygirl/article/details/7698823

当然，双指针法还有其他一些应用等待你的发现，它是一种奇妙的思维方式，不是么？