Two Pointers

适合滑动窗口, 复杂度为 O(n), 原地操作

- 相向双指针

```
public void patition(int[] A, int start, int end) {

  if (start >= end) {
      return;
  }

  int left = start, right = end;

  //key point 1: pivot is the value, not the index
  int pivot = A[(start + end) / 2];

  //key point 2: every time you compare left & right, it should always be left <= right
  while (left <= right){
    while (left <= right && A[left] < pivot) {
      left++;
    }
    while (left <= right && A[right] > pivot){
      right--;
    }
    if(left <= right) {

      int temp = A[left];

      A[left] = A[right];

      A[right] = temp;

      left++; right--;

    }

  }

}
```

- 背向双指针


```
left = position;
right = position + 1;

while (left >= 0 && right < length){
  if(可以停下来了) {
    break;
  }
  left--;
  right++;
}
```

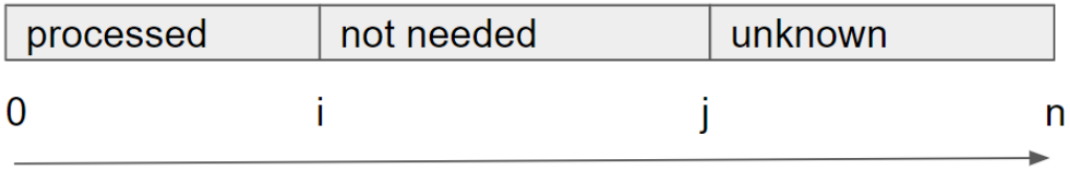
- 套路只有一个while loop
- if condition 可有可无， 能replace成swap
- 其中 [0, i) 和 (j, array.length) 内的数据均为处理好的数据，[i, j] 中的数据待处理。用此方法处理过的数组不会保留原来元素的相对位置

通用步骤

- Initalize two pointers ptr1 = 0, ptr2 = array.length - 1
- While i ≤ j:
  - Decide where you should do based on the value of array[i] and array[j]
  - Move at least one pointer forward in its direction (ptr1 与 ptr2 之间必须有一个指针向对方前进）

同向双指针

- 两个指针朝相同方向移动，但是快慢不同
- 分成三个区域: 其中 [0, i) 的数据代表处理好的数据，[i, j) 中的数据是那些处理过但不需要的数据，[j, array.length) 区间的数据为接下来待处理的数据




```
int j = 0;
for (int i = 0; i < n; i++){
    // 不满足则循环到满足搭配为止
    while（j < n && i 到 j 之间不满足条件) {
      j += 1;
    }
    if (i 到 j 之间满足条件){
      处理i, j 这次的搭配
    }
}
```

- Initialize two pointers ptr1 and ptr2, usually both equal to 0
- while j < array.length:
  - If we need array[ptr2], then we keep it by assigning array[ptr1] = array[ptr2], and move ptr1 forward, make it ready at next position
  - Otherwise skip it, We don't need to move ptr1 since its spot is not fulfilled

合并双指针

```
ArrayList <Integer> merge (ArrayList <Integer> list1, ArrayList <Integer> list2){

    ArrayList <Integer> newlist = new ArrayList <Integer> ();

    int i = 0; j = 0;

    while( i < list1.size() && j < list2.size()){

      if(list1.get(i) < list2.get(j)) {

        newList.add(list1.get(i));
        i++;

      } else {
```

```
          newList.add(list2.get(j));
          j++;

        }
      }

      while (i < list1.size()){

        newList.add(list1.get(i));
        i++;

      }
      while(j < list2.size()){

        newList.add(list2.get(j));
        j++;

      }

      return newList;
  }
```

Two Pointers 题型