Sort 面试会遇到的排序

O(n * n)

- Bubble Sort	stable	
- Insertion Sort	stable	
- Selection Sort	NOT stable	

O(n * log(n))

- Quick Sort	Not stable	
- Merge Sort	stable	S(N)

Arrays.sort() ⇒ DualPivotQuicksort.sort()
Collections.sort() => convert to array then Arrays.sort()

Bubble Sort

交换相邻的两个元素，实现最大的在数组最后

```
// S(1) O(n*n) stable
public void sort(final int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr.length - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                arr[j] = arr[j] ^ arr[j + 1];
                arr[j + 1] = arr[j] ^ arr[j + 1];
                arr[j] = arr[j] ^ arr[j + 1];
            }
        }
    }
}
```

Insertion Sort

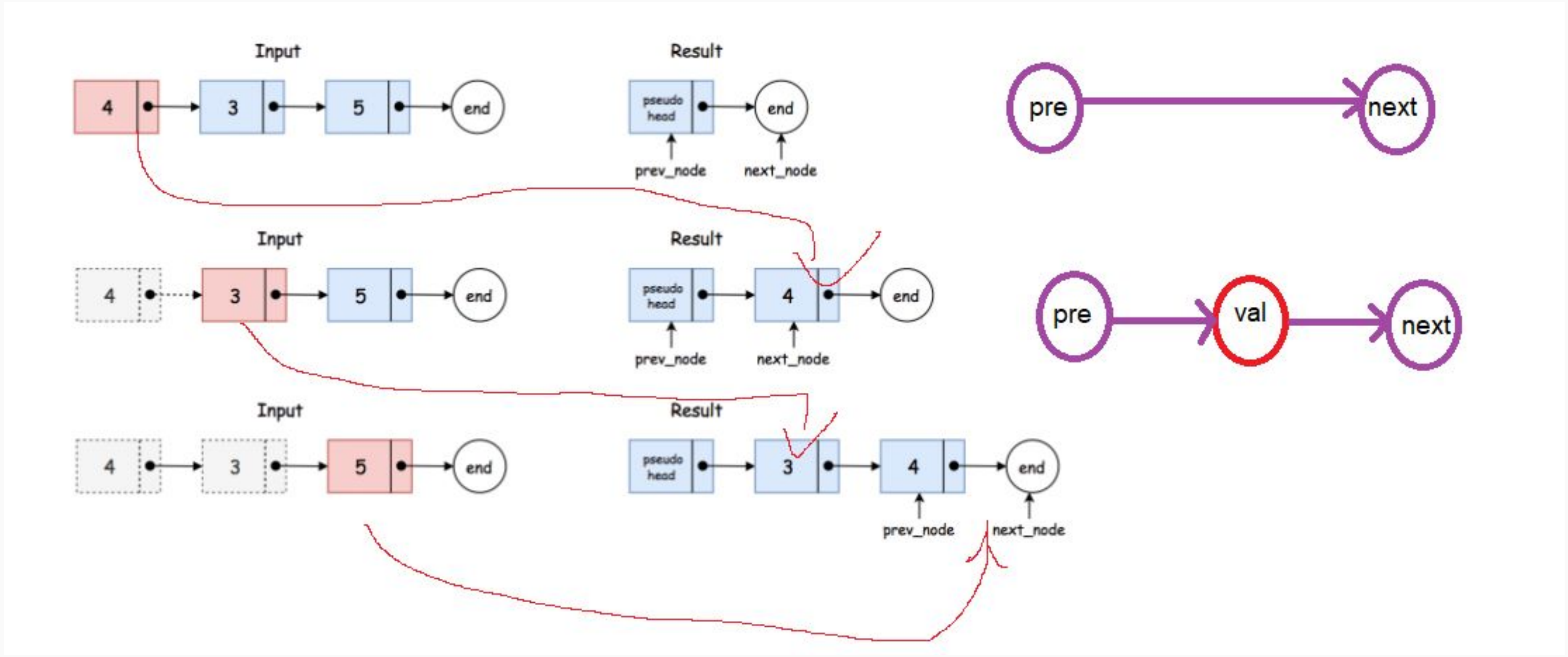
数组从下标0开始，
每次找到下个数在已排好序部分的位置

```
// S(1) O(n*n) stable
 public void sort(final int[] arr) {
     for (int i = 1; i < arr.length; i++) {
         int lastPosition = i - 1;
         int current = arr[i];
         while (lastPosition >= 0 && arr[lastPosition] > current) {
             arr[lastPosition + 1] = arr[lastPosition];
             lastPosition--;
         }
         arr[lastPosition + 1] = current;
     }
 }
```

```
25 --- 1 45 10 7
1 25 --- 45 10 7
1 25 45 --- 10 7
*****current=10*****
1 25 45 [] 7
1 25 45 45 7
1 25 25 45 7
1 10 25 45 --- 7
*****current=7*****
1 10 25 45 []
1 10 25 45 45
1 10 25 25 45
1 10 10 25 45
1 7 10 25 45
```

147. Insertion Sort List

https://leetcode.com/problems/insertion-sort-list/



Selection Sort

每次都把最小的元素放前面

```
// S(1) O(n*n) NOT stable
// [ 7, 10, 7, 2, 30 ] the first 7 will exchange with 2. This makes it after the second 7
public void sort(final int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < arr.length; j++) {
            if (arr[minIndex] > arr[j]) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            arr[minIndex] = arr[minIndex] ^ arr[i];
            arr[i] = arr[minIndex] ^ arr[i];
            arr[minIndex] = arr[minIndex] ^ arr[i];
        }
    }
}
```

Quick Sort

先整体有序，再局部有序　　... <= P <= ...

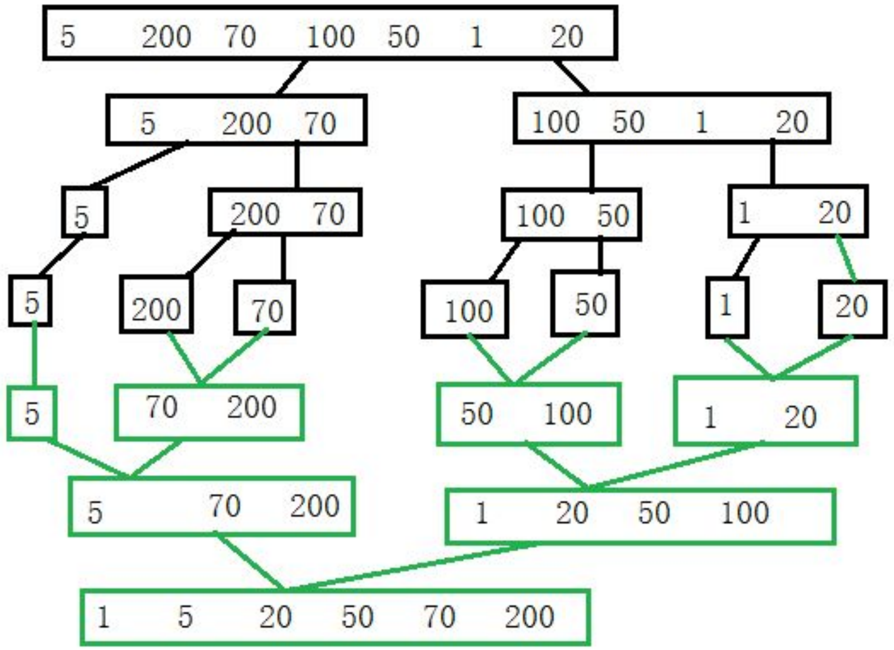
```
left <= right
  [1,2]
   if replace all left <= right by left < right
   pivot = 1
   left = 0, arr[0] =1
   right = 1, arr[1] =2

   at the end of the first loop
   left = 0, arr[0] =1
   right = 0, arr[0] =1

   then we do:
   quickSort(arr, start, right); start = 0 right = 0 ==> [1]
   quickSort(arr, left, end);  left = 0 end = 1 ==> [1,2] ==> !! Stack Overflow !!
```

the right way is when comparing with pivot use < or >

```
while arr[left] <= pivot left++  arr[right] >= pivot right ++
  arr[left] > pivot  <==> arr[right] < pivot
  ... < pivot < ...
  pivot = 1
  [1,1]
  left = 1
  right = 1
  quickSort(arr, start, right); start = 0 right = 1 ==> [1,1]
==> !! Stack Overflow !!
  quickSort(arr, left, end);  left = 1 end = 1 ==> [1]
```

Merge Sort

先分组，后合并


5 200 70 100 50 1 20
5 200 70
100 50 1 20
5
200 70
100 50
1 20
5
200
70
100
50
1
20
5
70 200
50 100
1 20
5 70 200
1 20 50 100
1 5 20 50 70 200