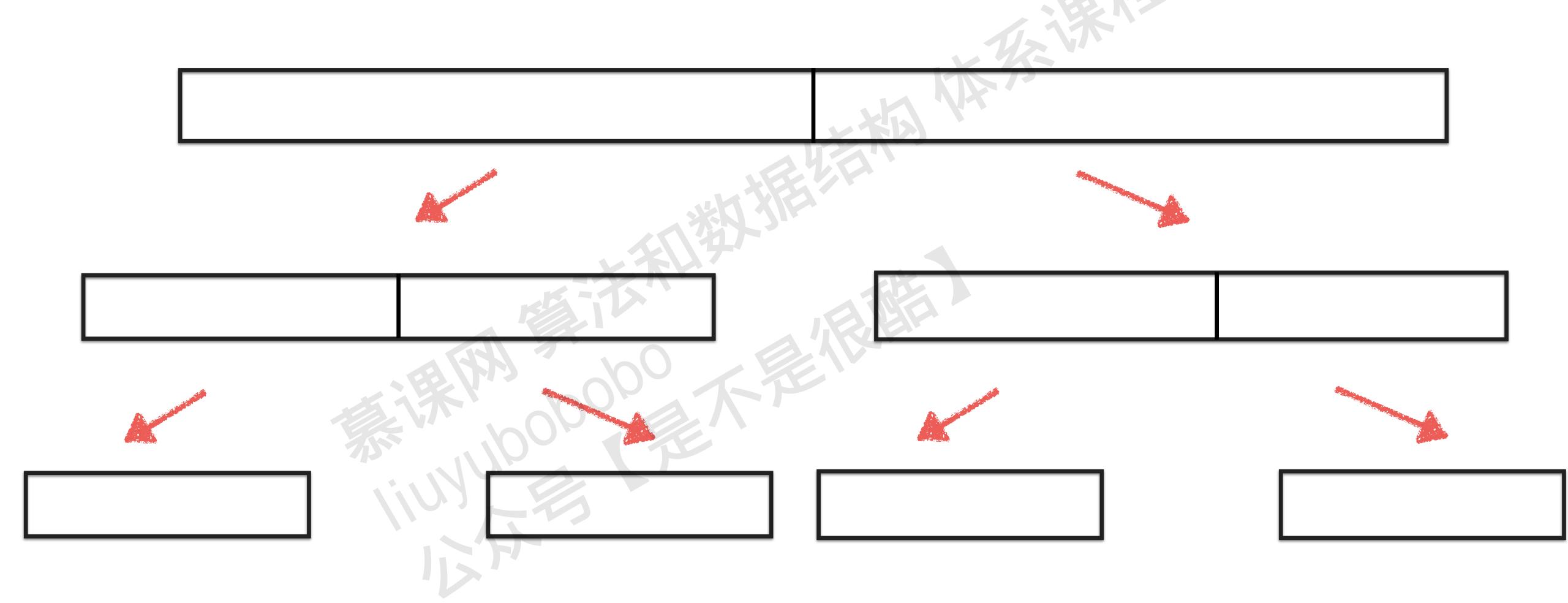
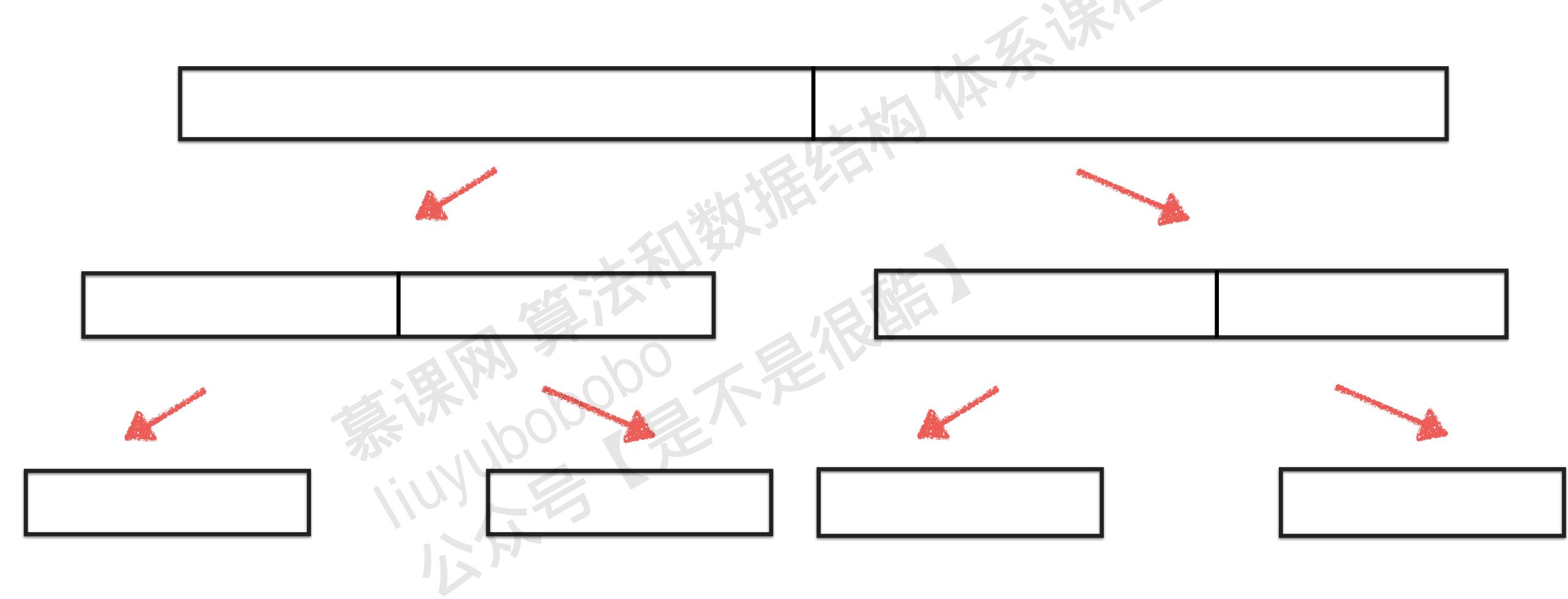
算法与数据结构体系课程

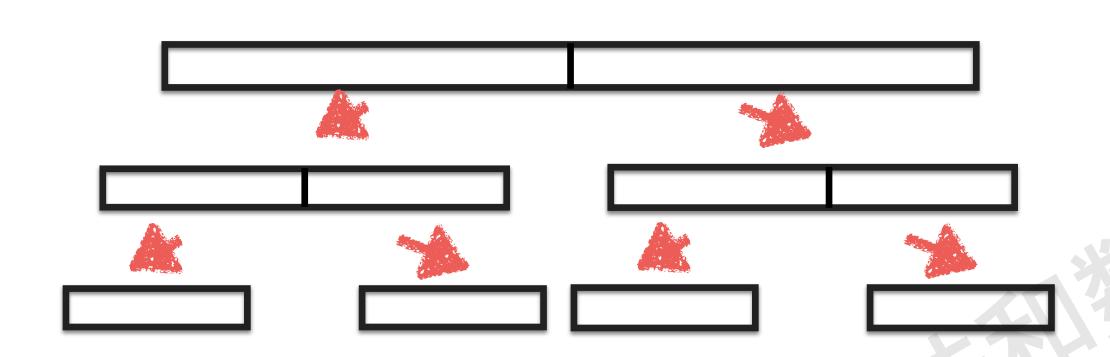
liuyubobobo

更复杂的递归算法

O(nlogn) 的排序算法







注意递归函数的"宏观"语意

MergeSort(arr, l, r)

对 arr 的 [I, r] 部分排序

```
MergeSort(arr, l, r){
    if(l >= r) return;
    int mid = (l + r) / 2;
    // 对 arr[l, mid] 进行排序
   MergeSort(arr, l, mid);
   // 对 arr[mid + 1, r] 进行排序
   MergeSort(arr, mid + 1, r);
    // 将arr[l,mid]和arr[mid+1,r]合并
    merge(arr, l, mid, r);
```

```
MergeSort(arr, l, r){
   if(l >= r) return;
                           求解最基本问题
   int mid = (l + r) / 2;
   // 对 arr[l, mid] 进行排序
   MergeSort(arr, l, mid);
                                      把原问题转化成
                                      更小的问题
   // 对 arr[mid + 1, r] 进行排序
   MergeSort(arr, mid + 1, r);
   // 将arr[l,mid]和arr[mid+1,r]合并
                                        如何归并?
   merge(arr, l, mid, r);
```

已知两个有序数组A和B 将A和B合并成一个有序数组?

A 1 3 5 6 B 2 4 7 8

1 2 3 4 5 6 7 8

已知两个有序数组A和B 将A和B合并成一个有序数组?

A 1 3 5 6 B 2 4 7 8

已知两个有序数组 A 和 B 将 A 和 B 合并成一个有序数组?

A 3 5 6 B 2 4 7 8

已知两个有序数组 A 和 B 将 A 和 B 合并成一个有序数组?

A $\begin{bmatrix} 3 & 5 & 6 \end{bmatrix} \quad B \quad \begin{bmatrix} 4 & 7 & 8 \end{bmatrix}$

已知两个有序数组A和B将A和

将A和B合并成一个有序数组?

 $\begin{bmatrix} 5 \end{bmatrix} \begin{bmatrix} 6 \end{bmatrix} \quad B \quad \begin{bmatrix} 4 \end{bmatrix} \begin{bmatrix} 7 \end{bmatrix} \begin{bmatrix} 8 \end{bmatrix}$

1 2 3

已知两个有序数组A和B

将A和B合并成一个有序数组?

已知两个有序数组A和B

将A和B合并成一个有序数组?

已知两个有序数组A和B

将A和B合并成一个有序数组?

已知两个有序数组A和B 将A和B合并成一个有序数组?

A B

1 2 3 4 5 6 7

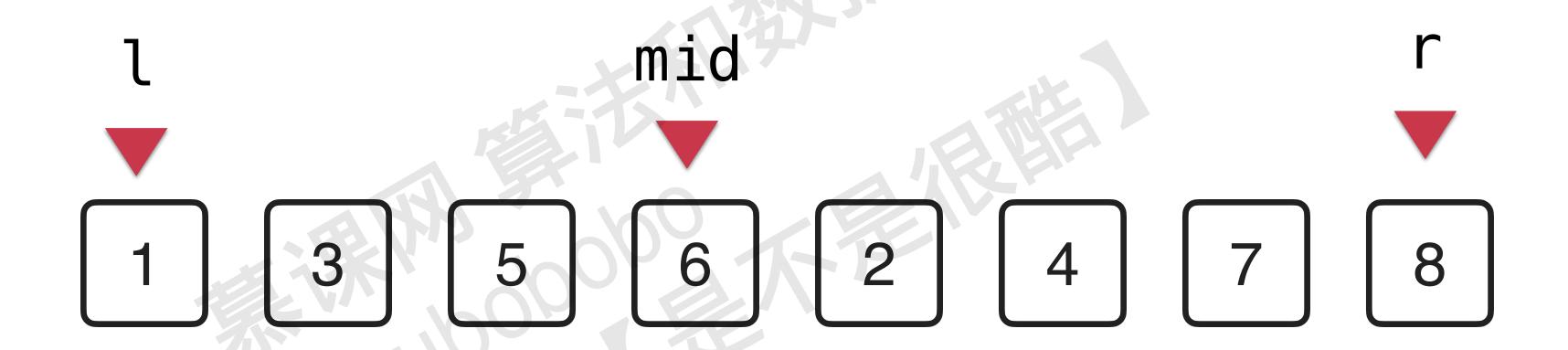
已知两个有序数组A和B 将A和B合并成一个有序数组?

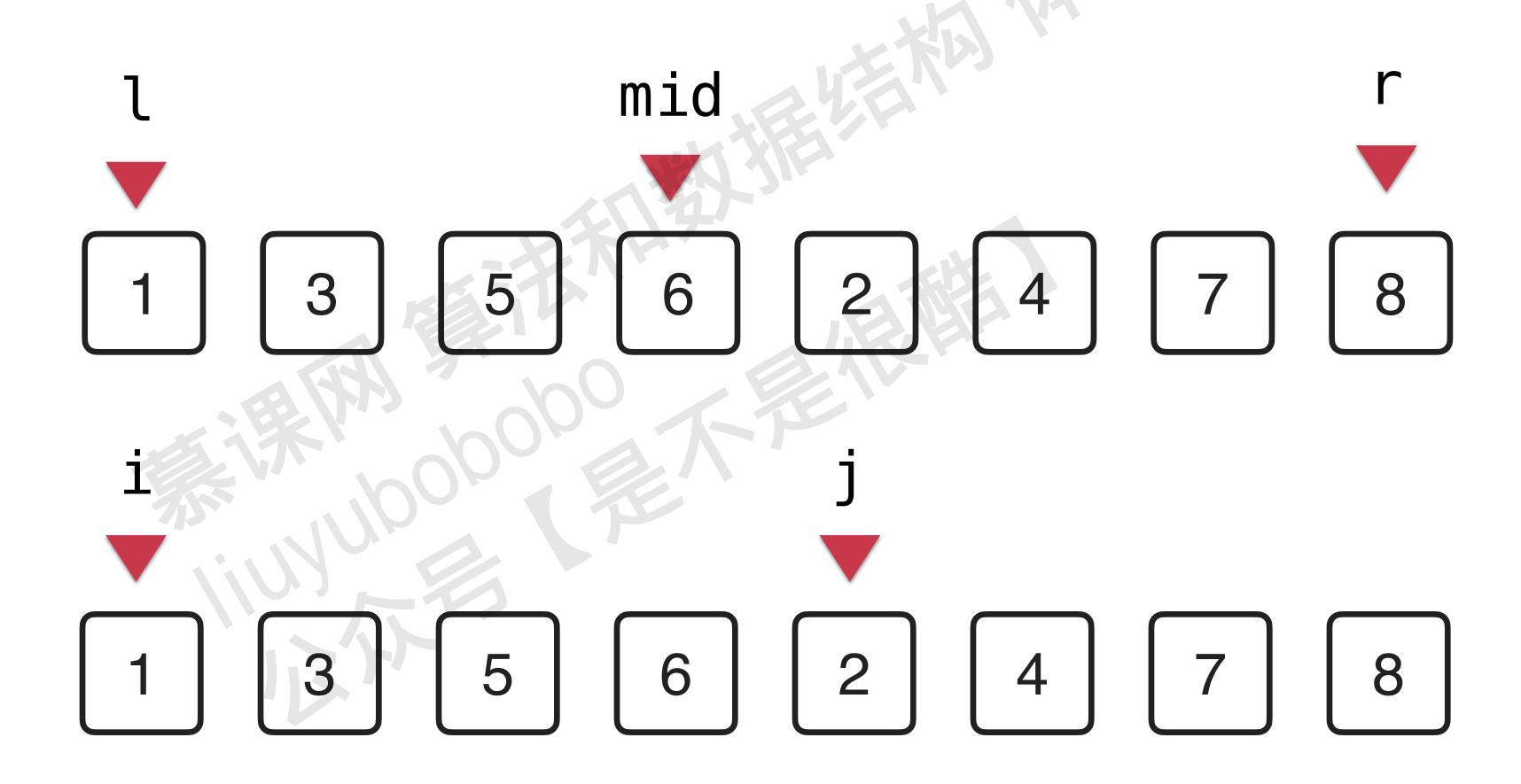
A B

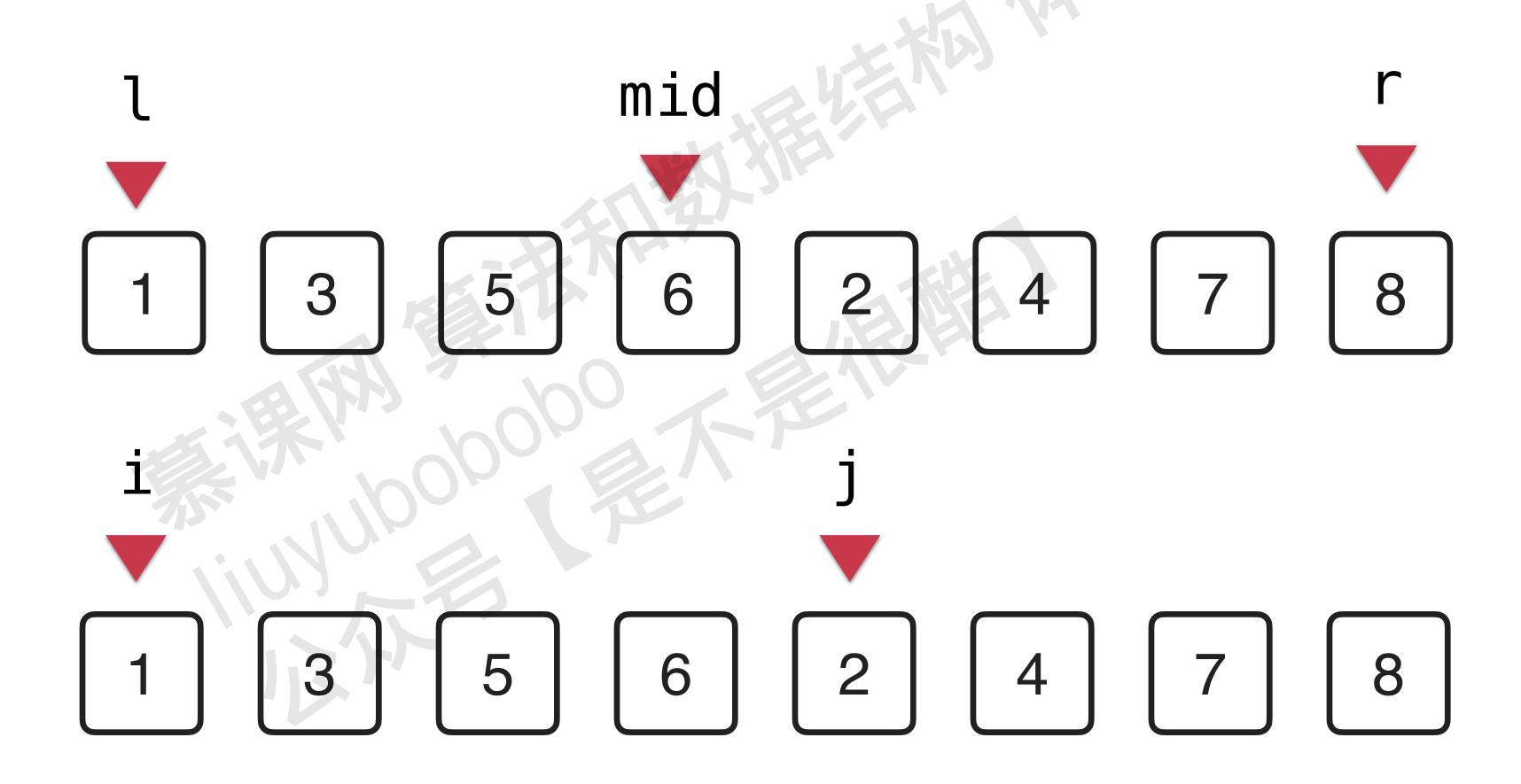
1 2 3 4 5 6 7 8

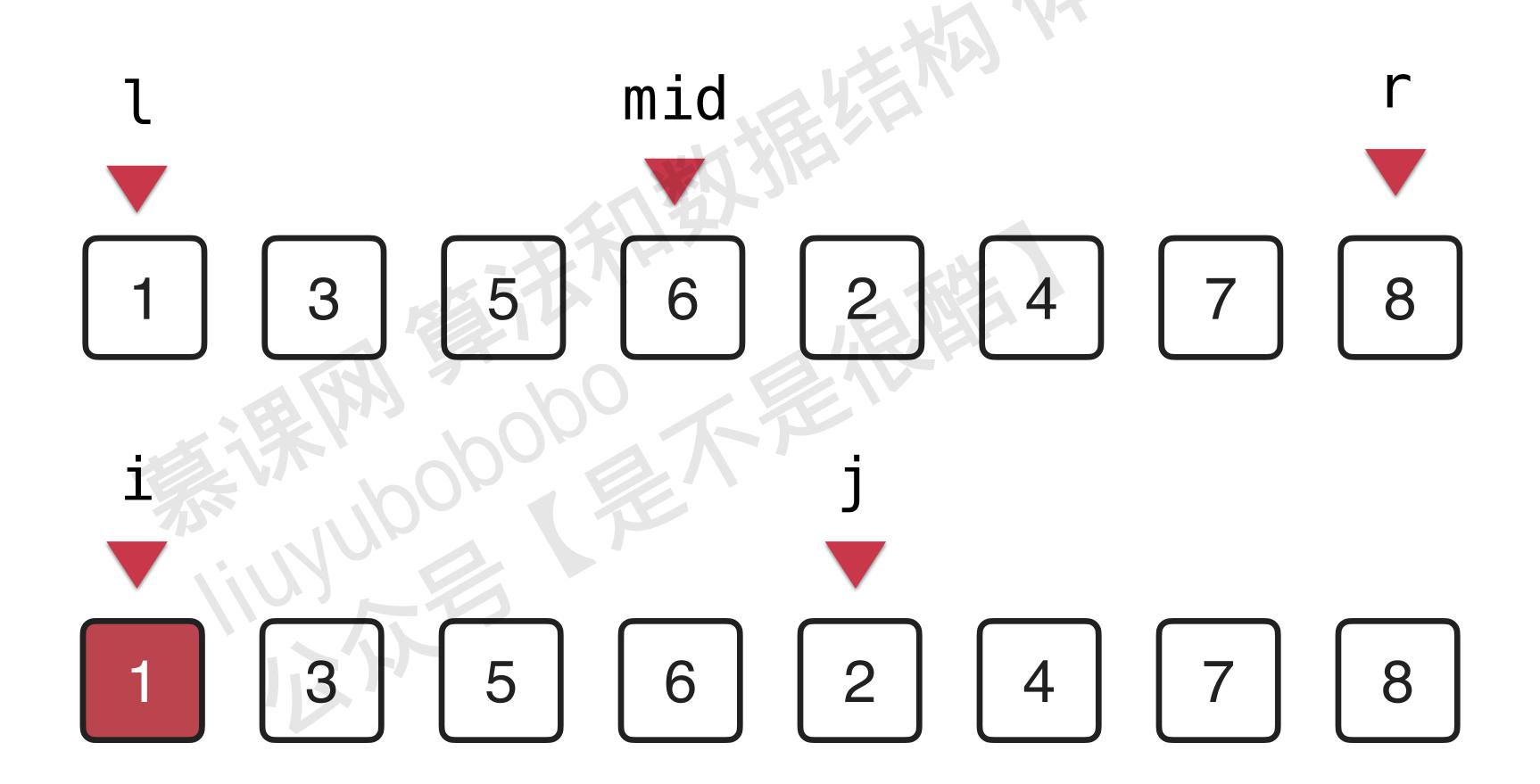
```
将A和B合并成一个有序数组?
已知两个有序数组A和B
// 将arr[l,mid]和arr[mid+1,r]合并
merge(arr, l, mid, r);
                    B
     1 2 3 4 5 6 7 8
```

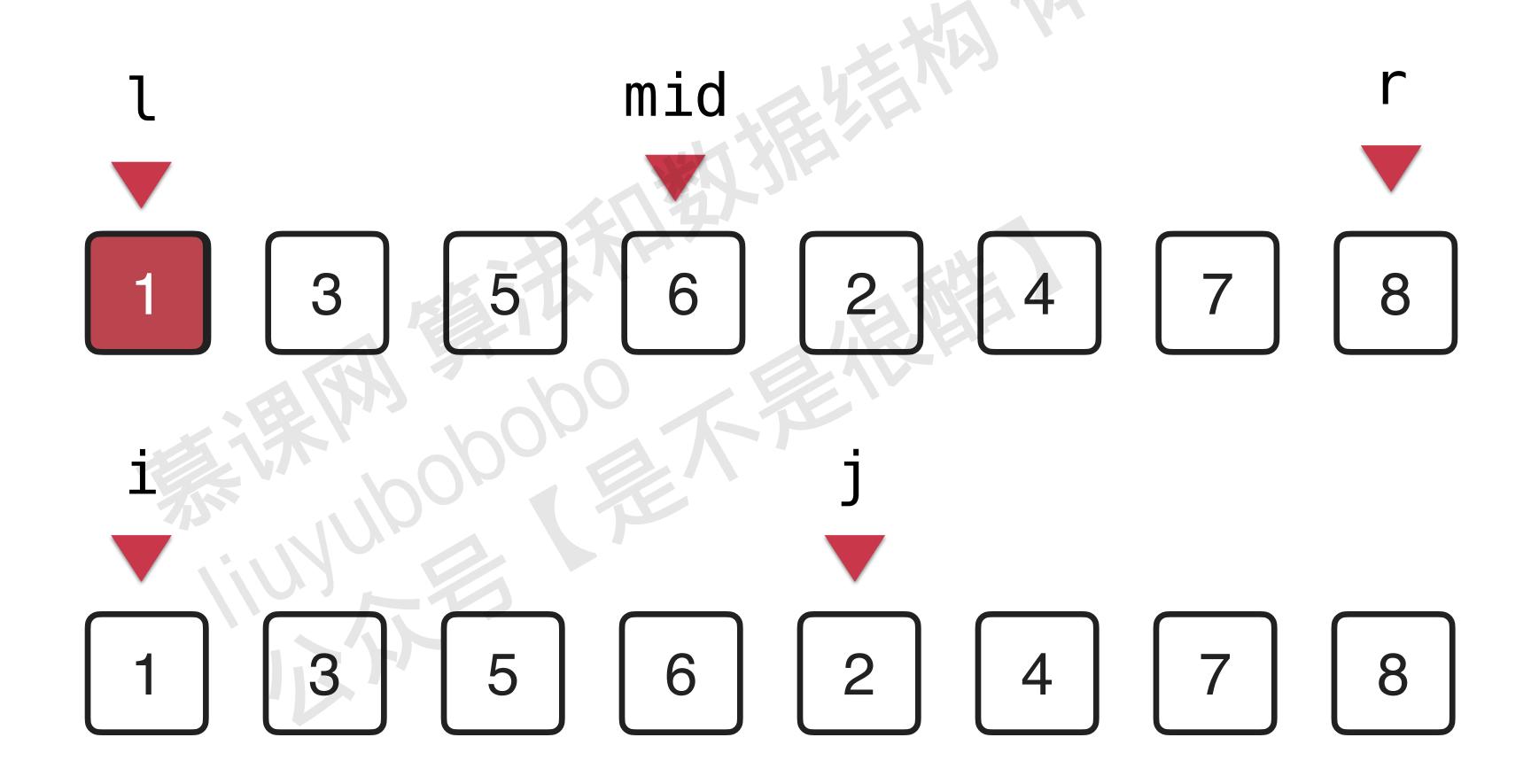
```
// 将arr[l,mid]和arr[mid+1,r]合并merge(arr, l, mid, r);
```

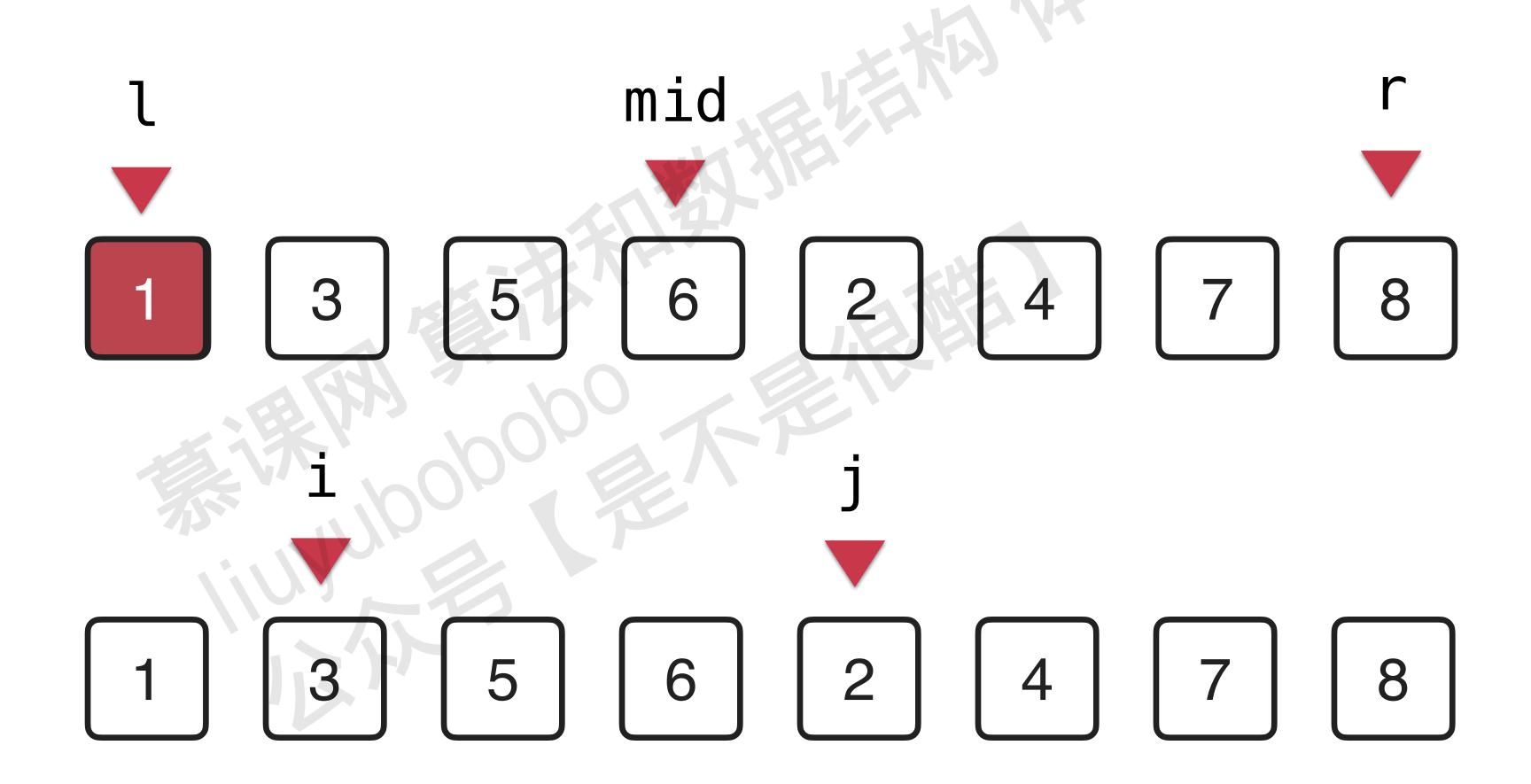


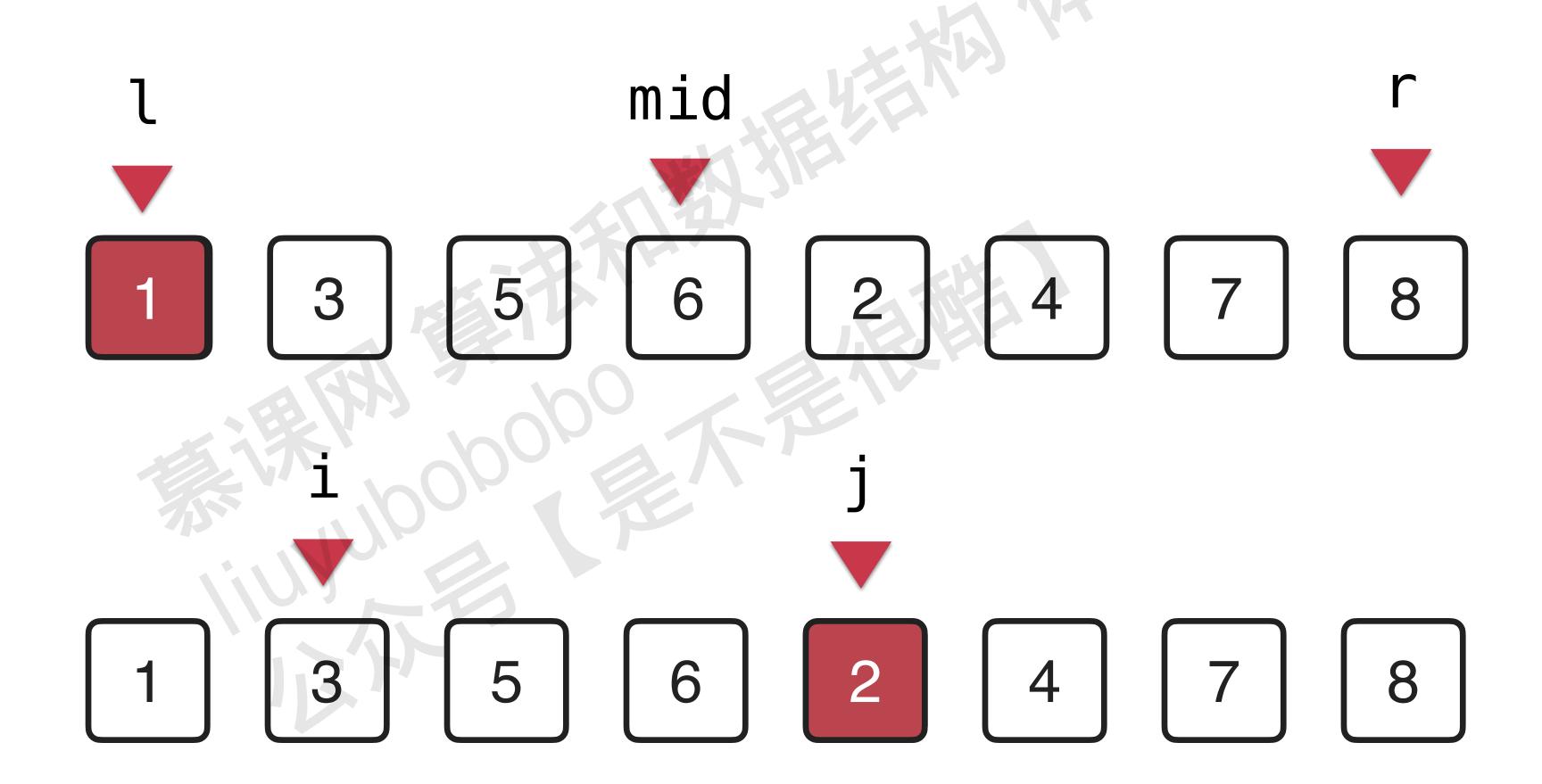


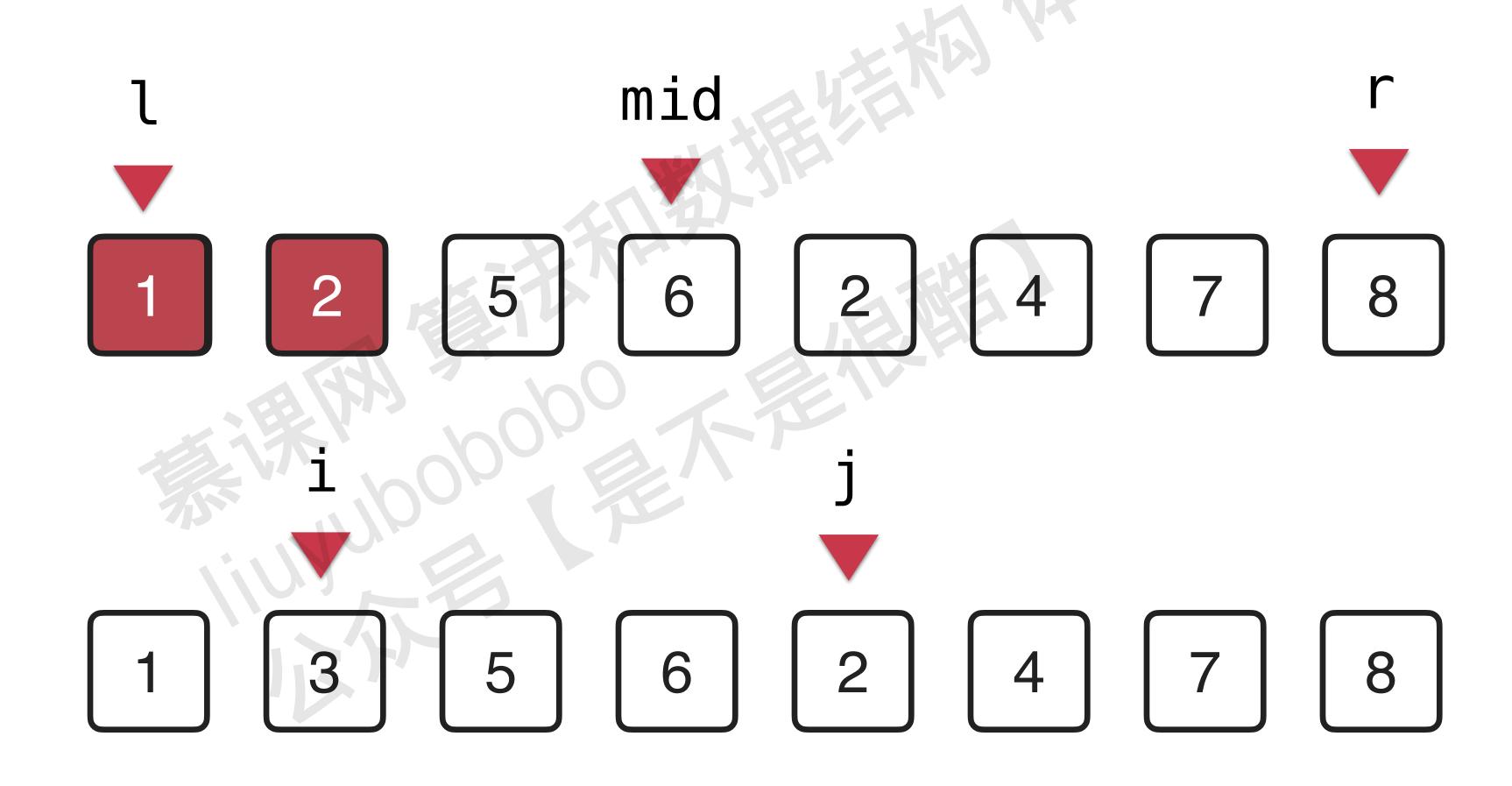


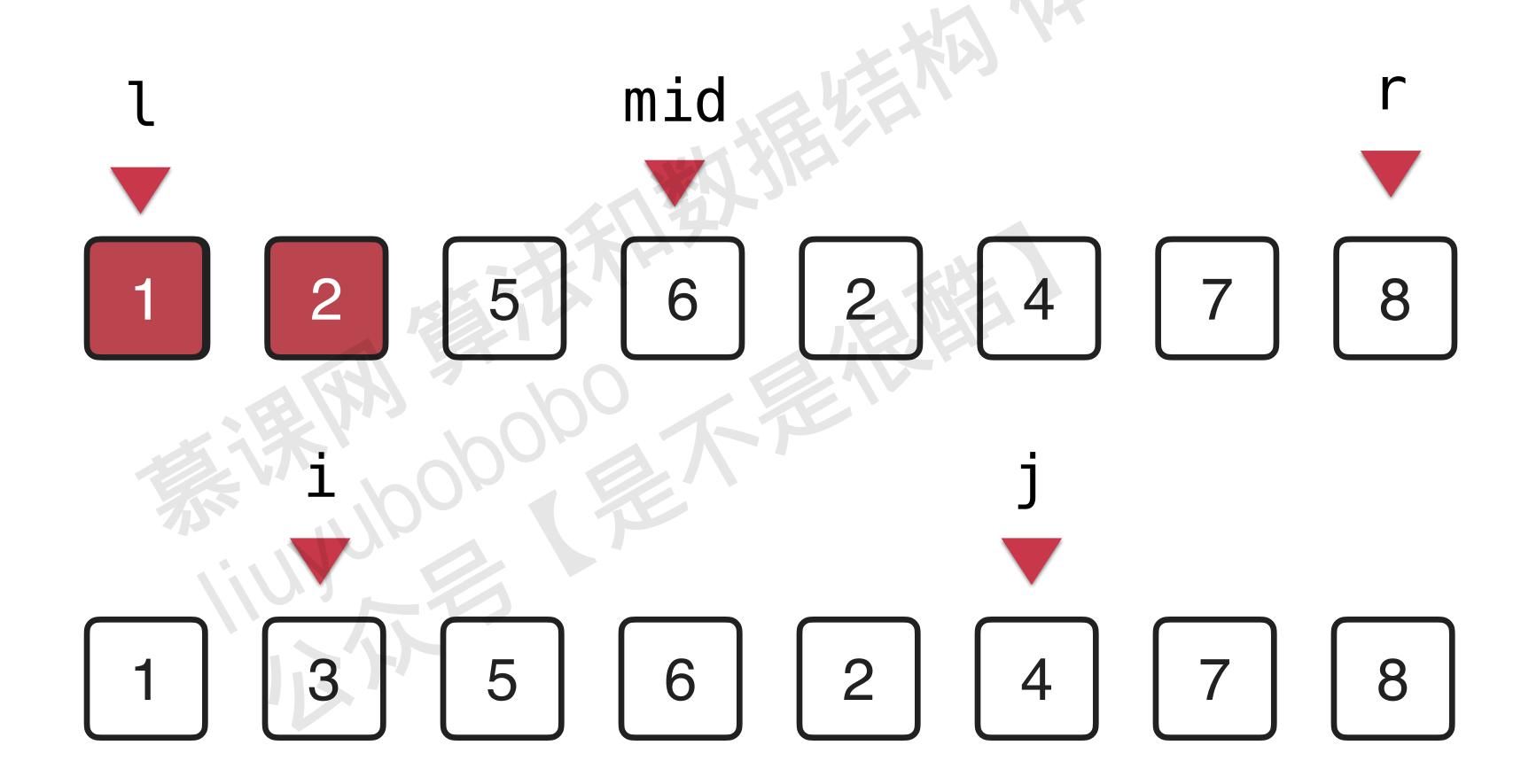


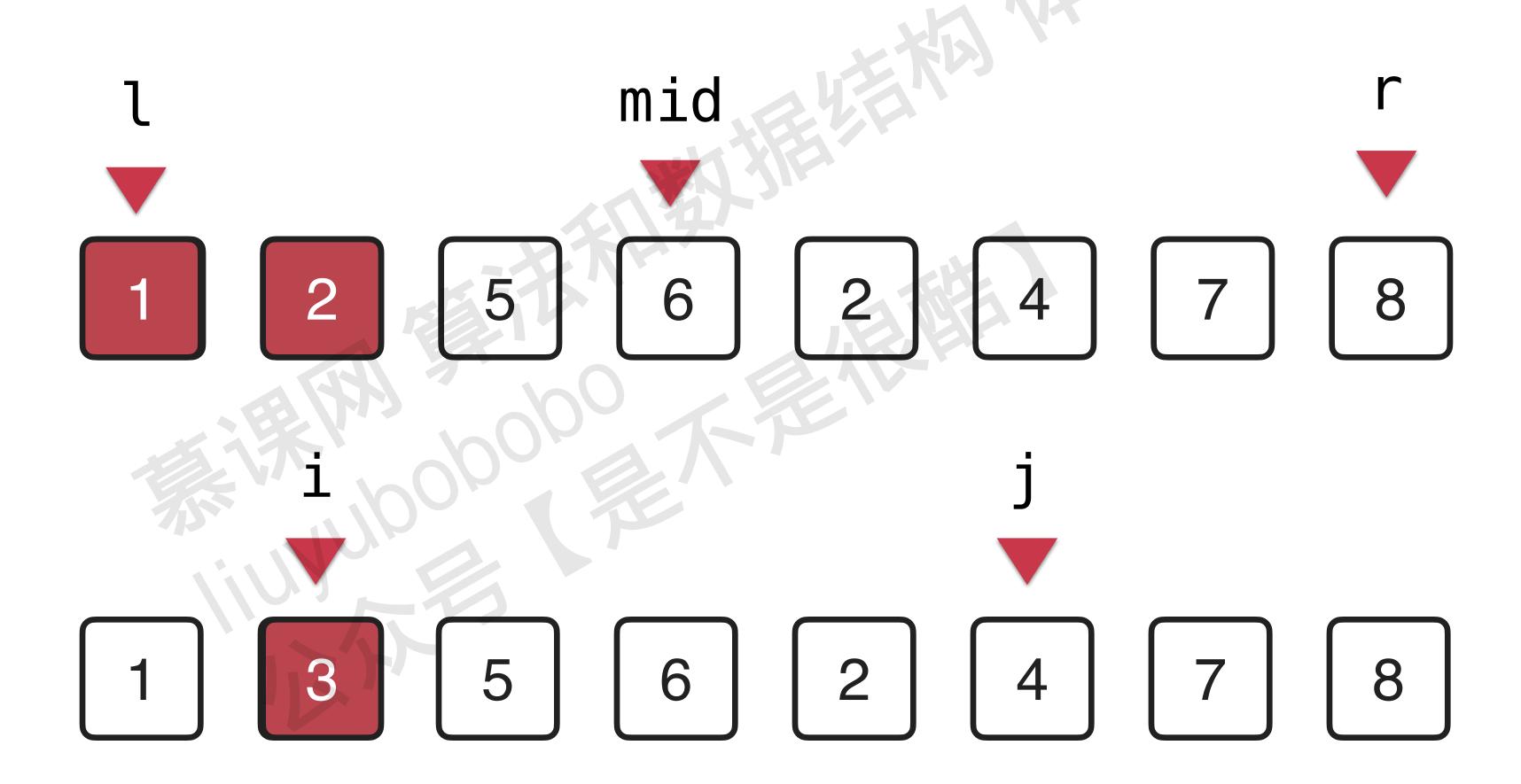


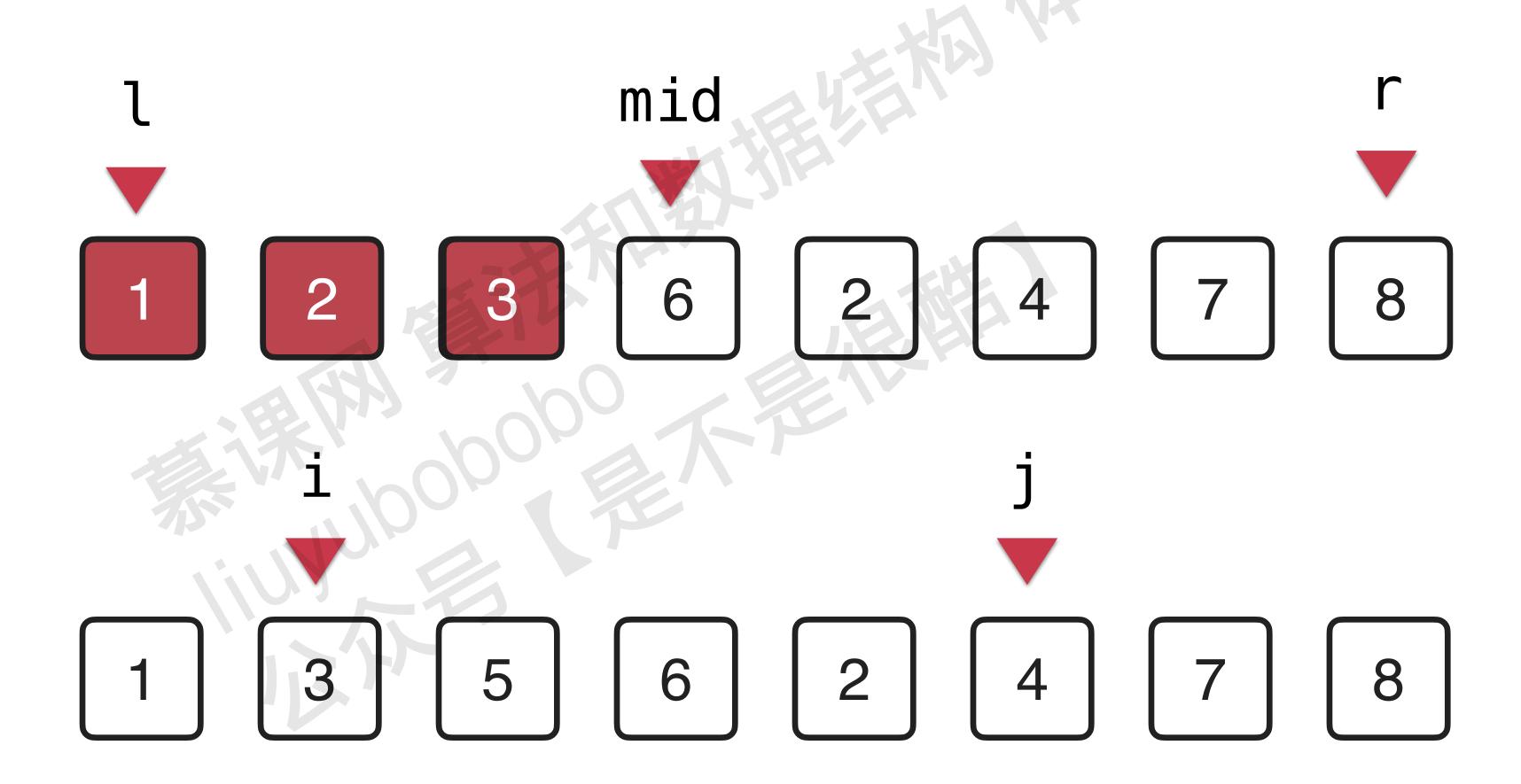


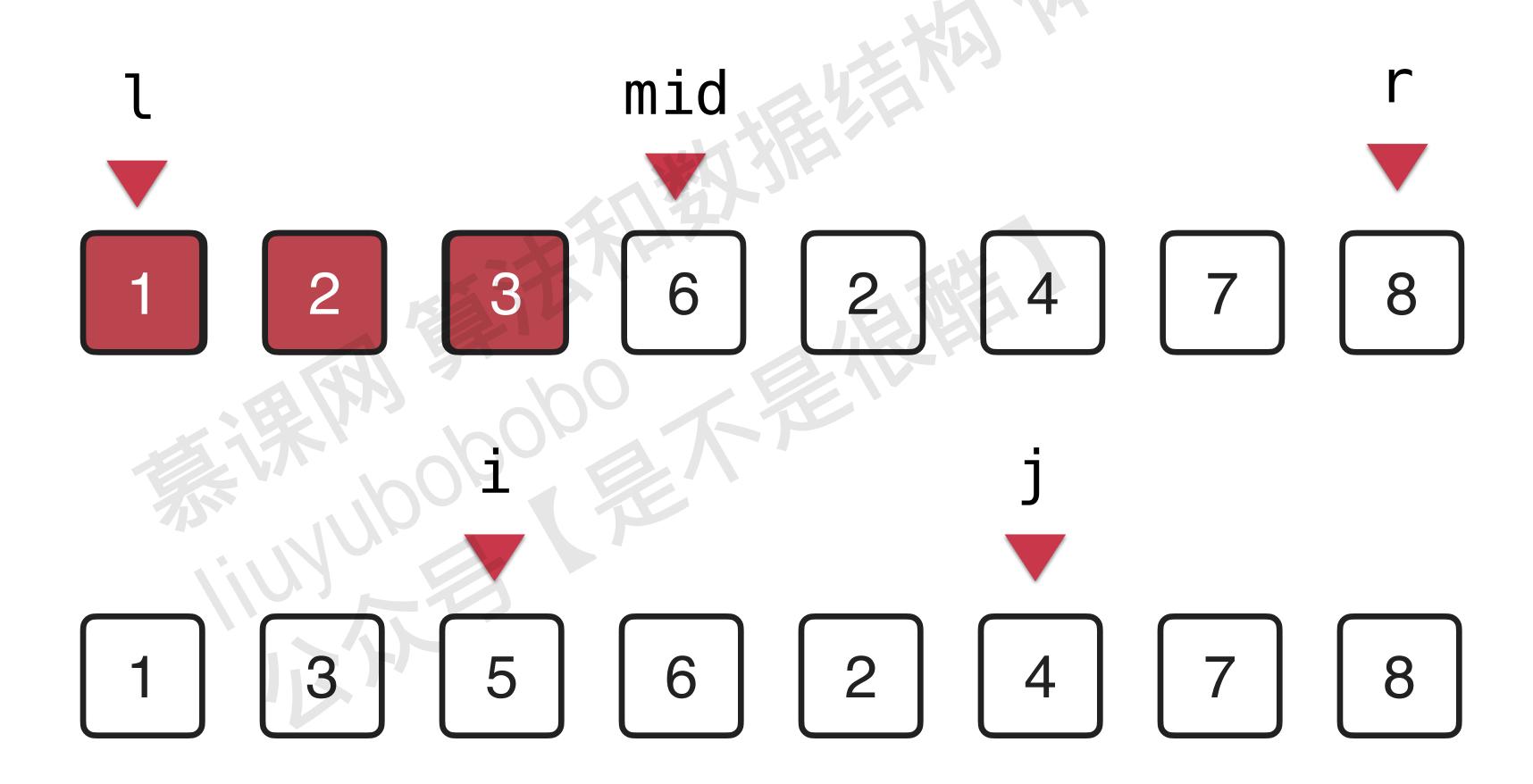


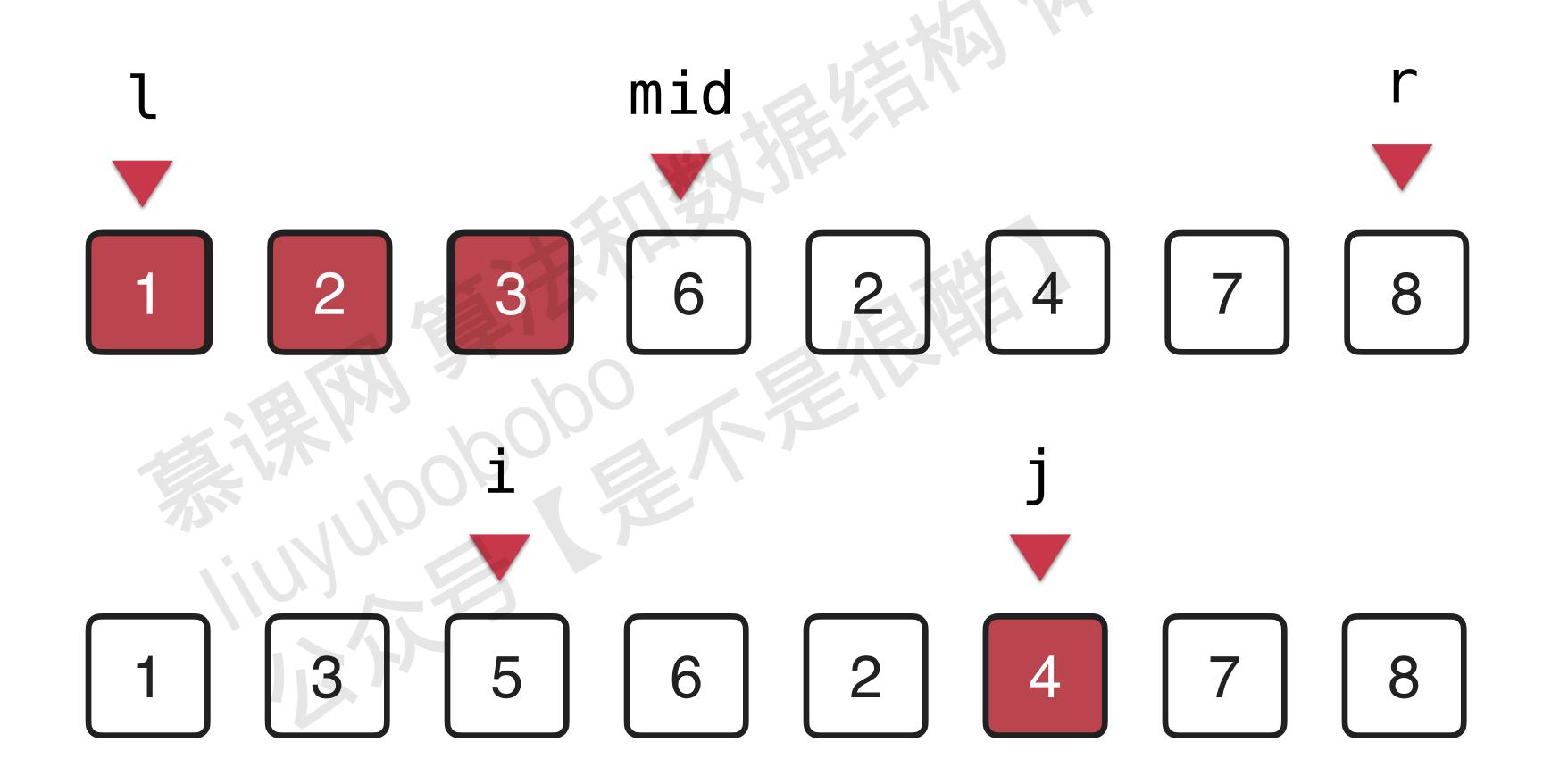


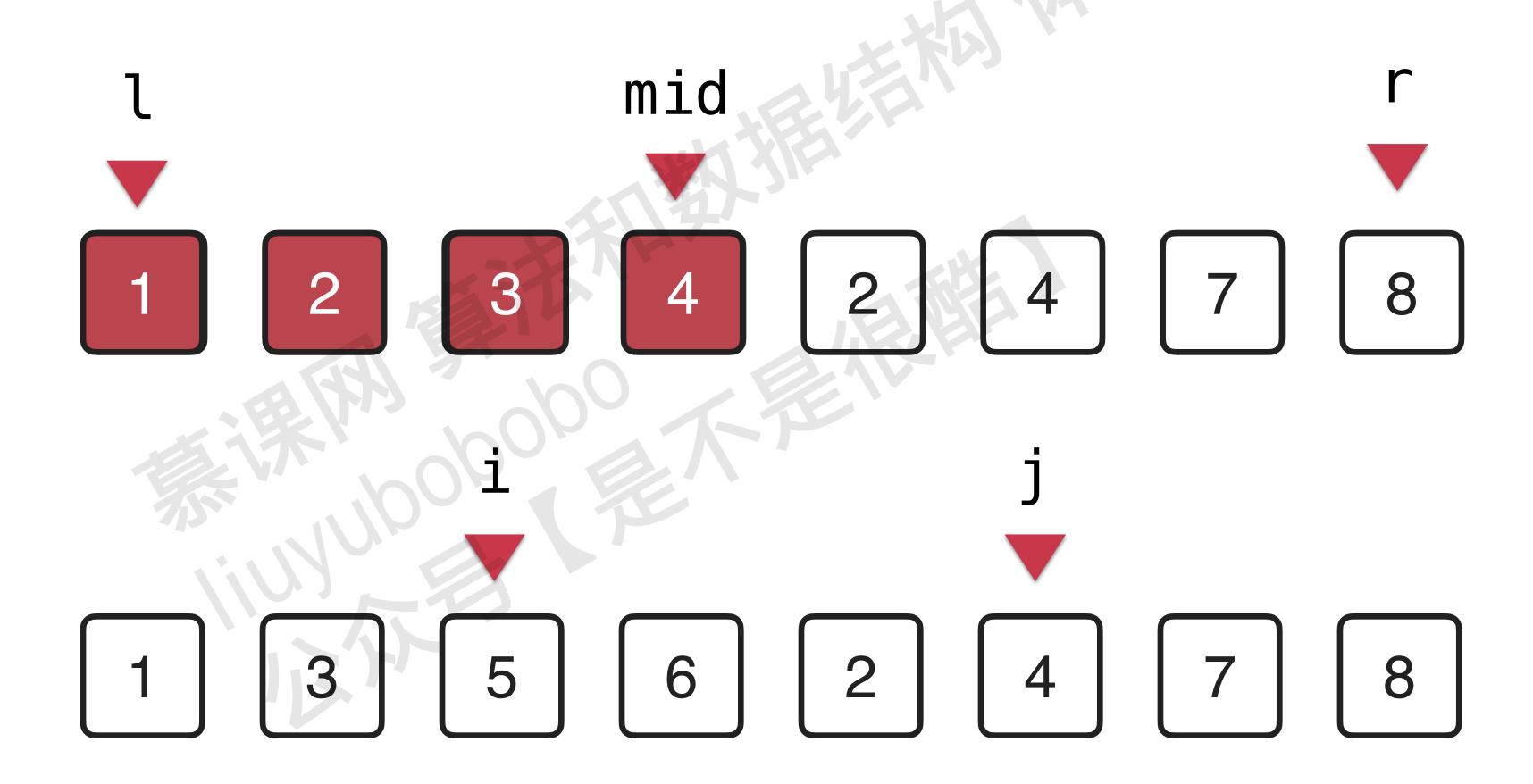


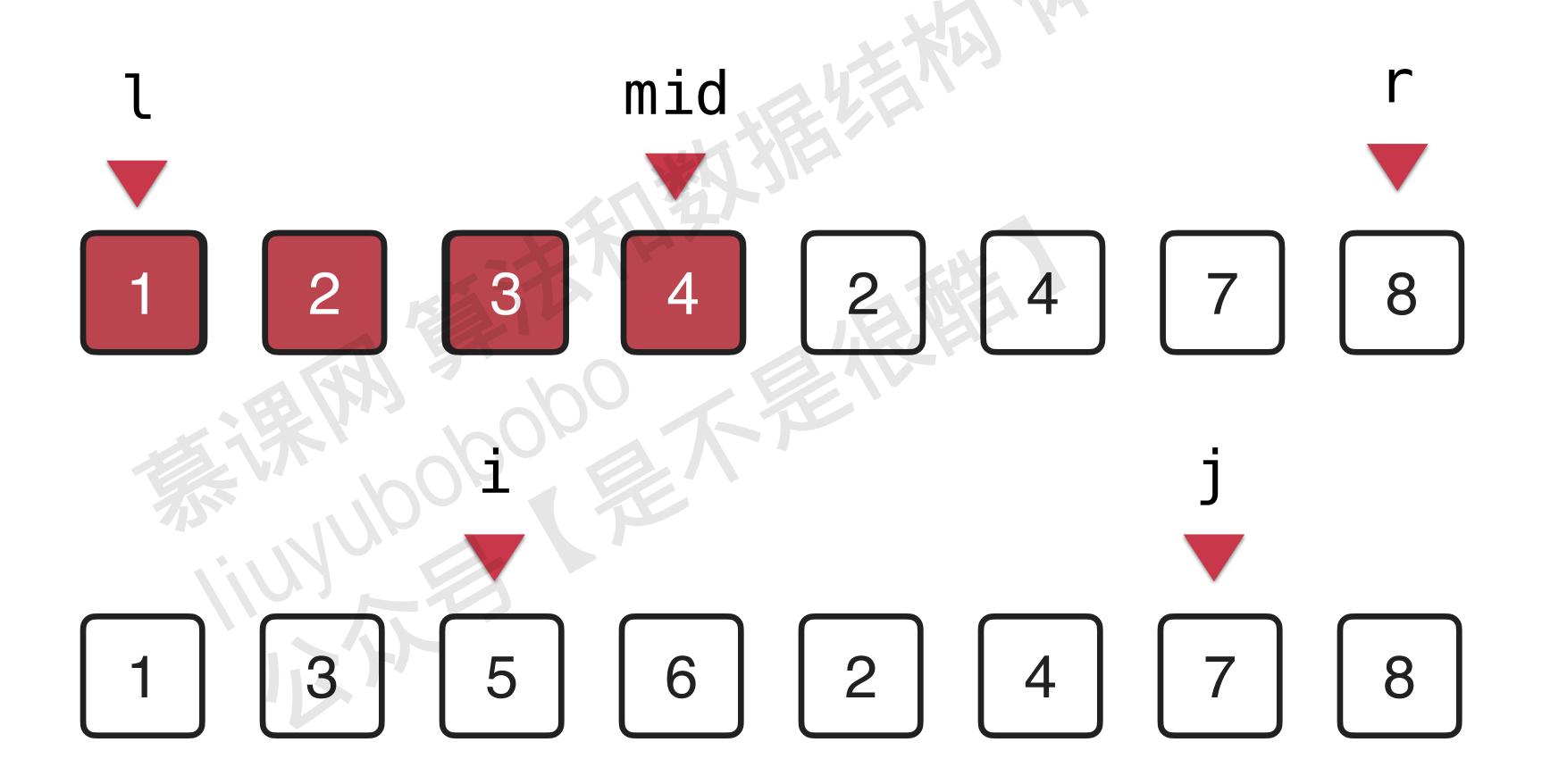


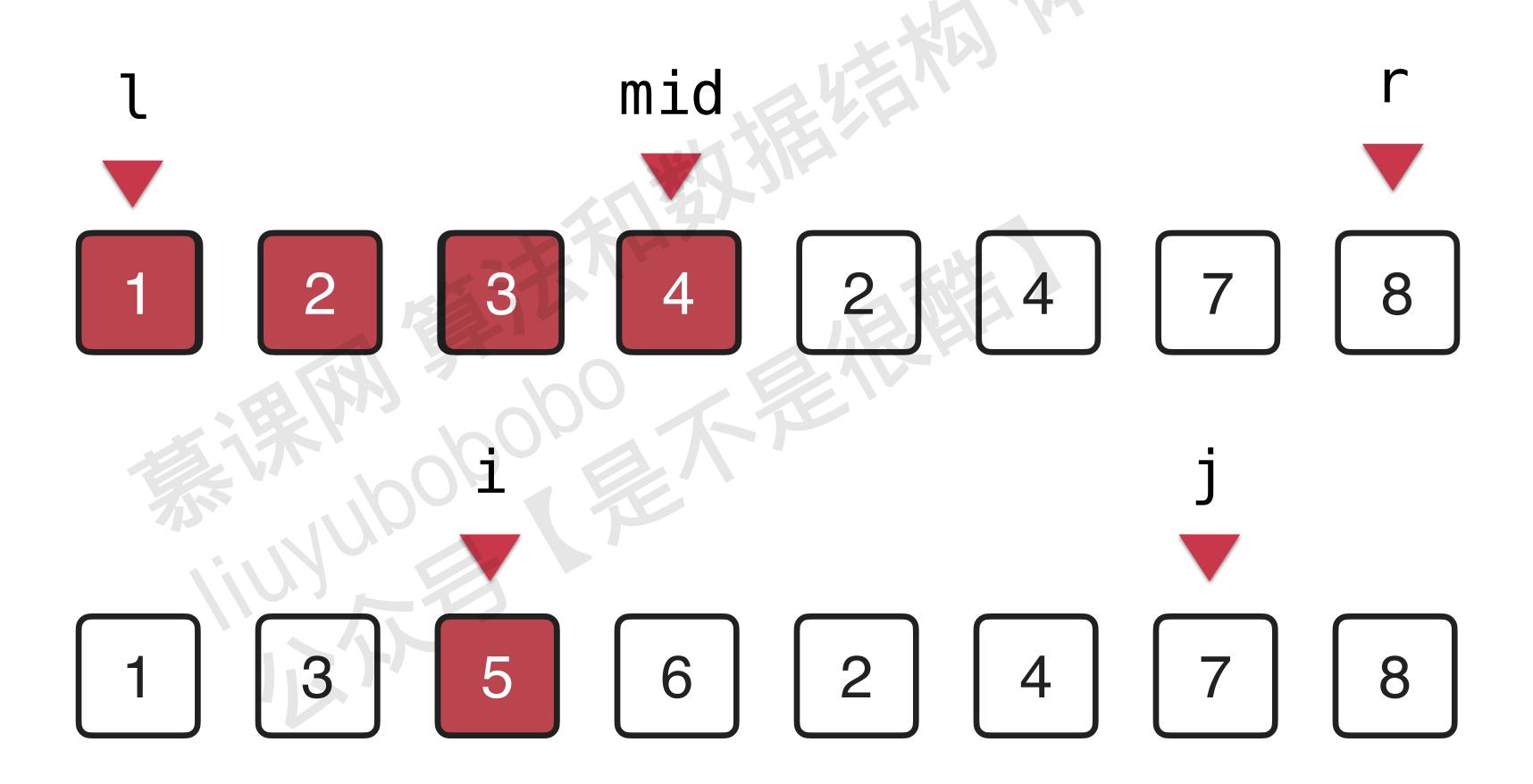




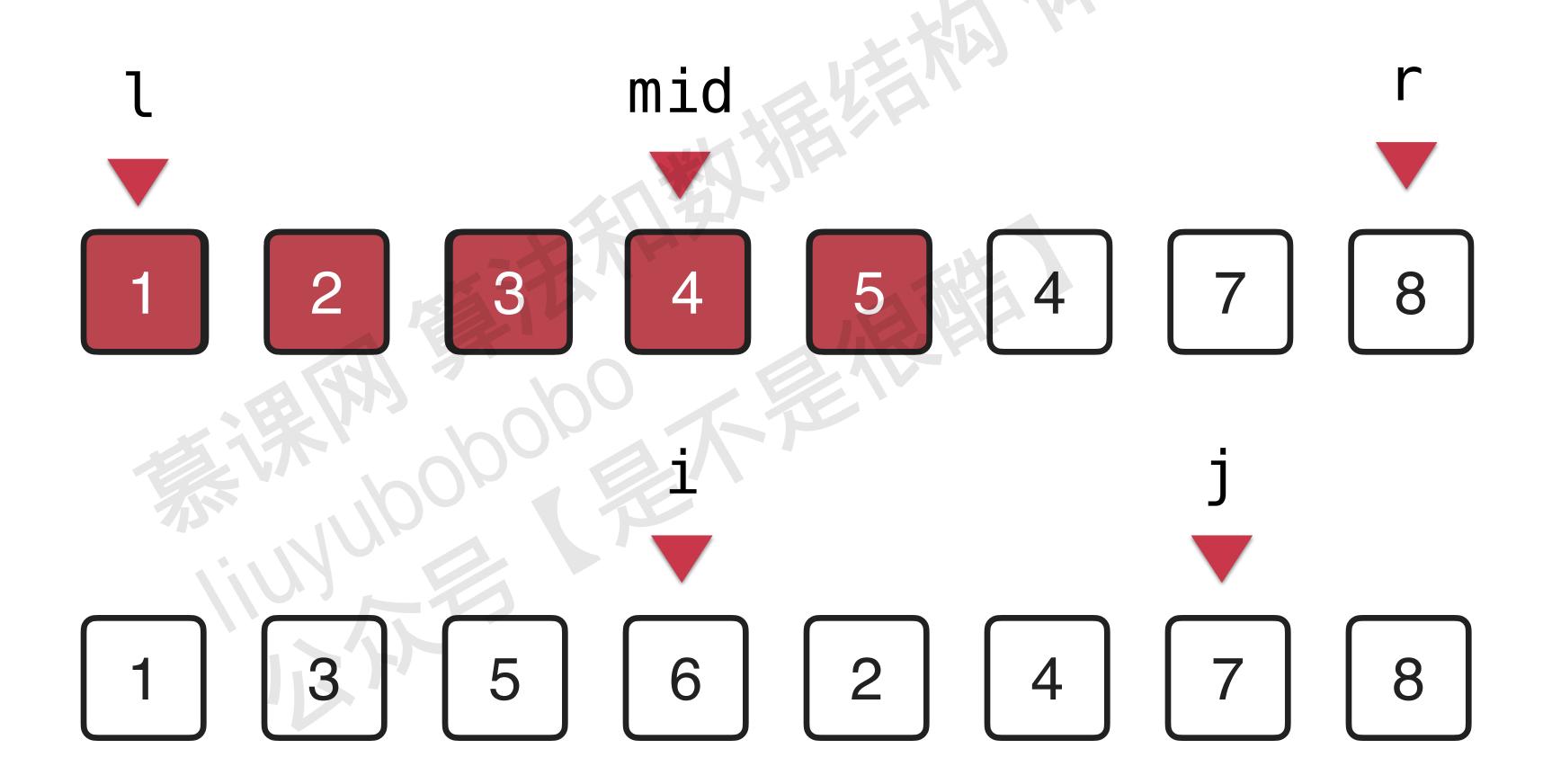


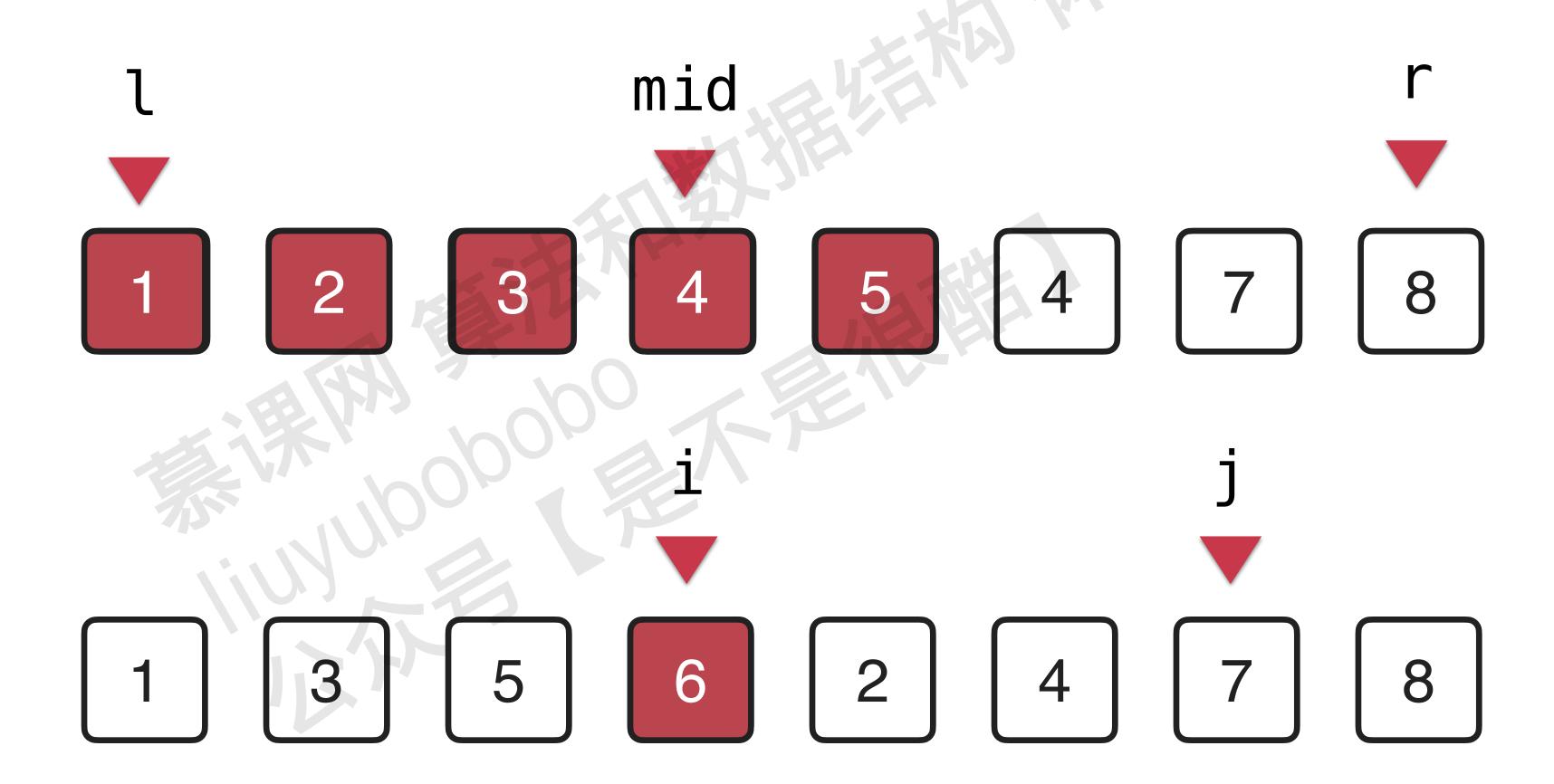


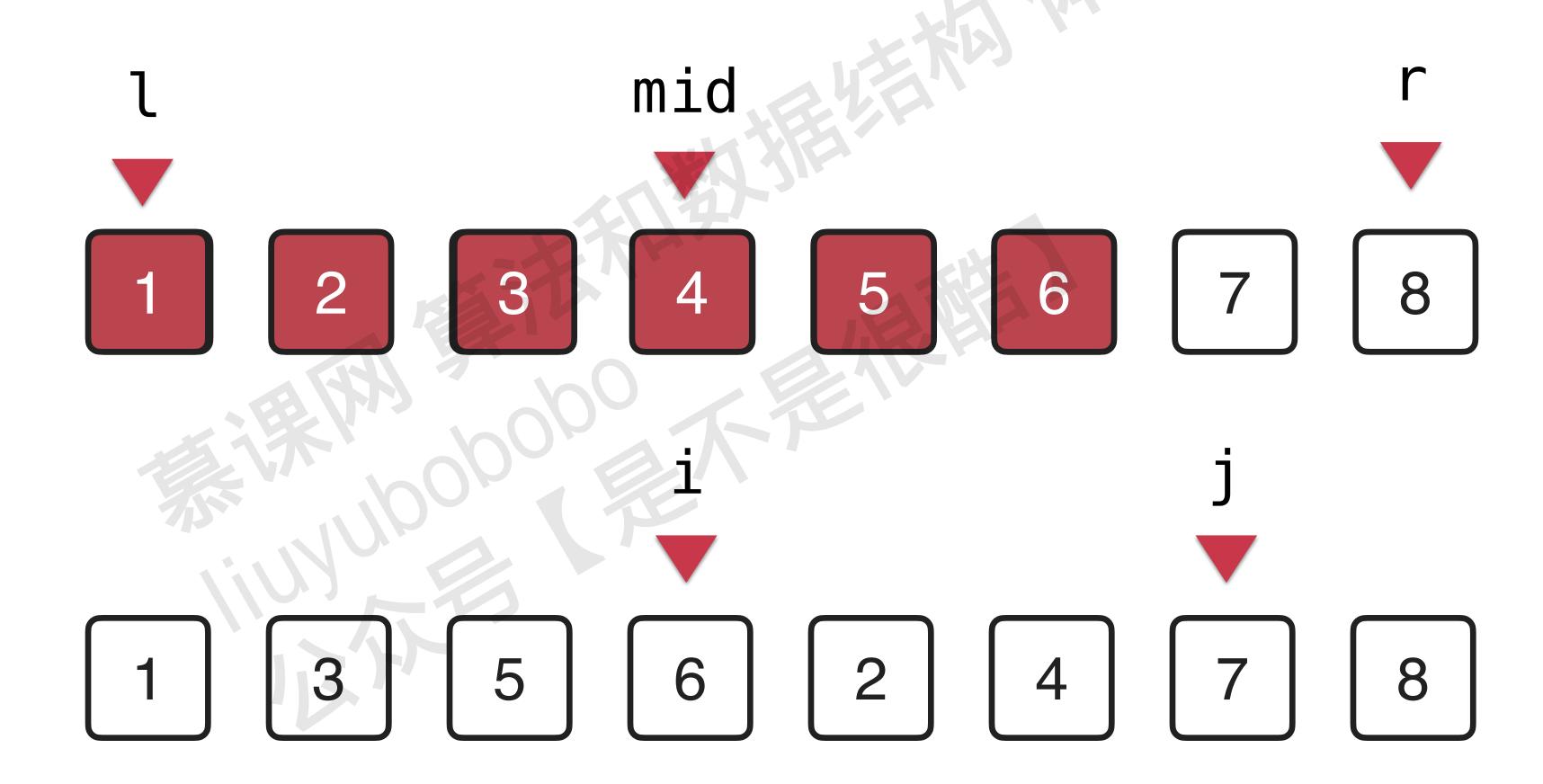


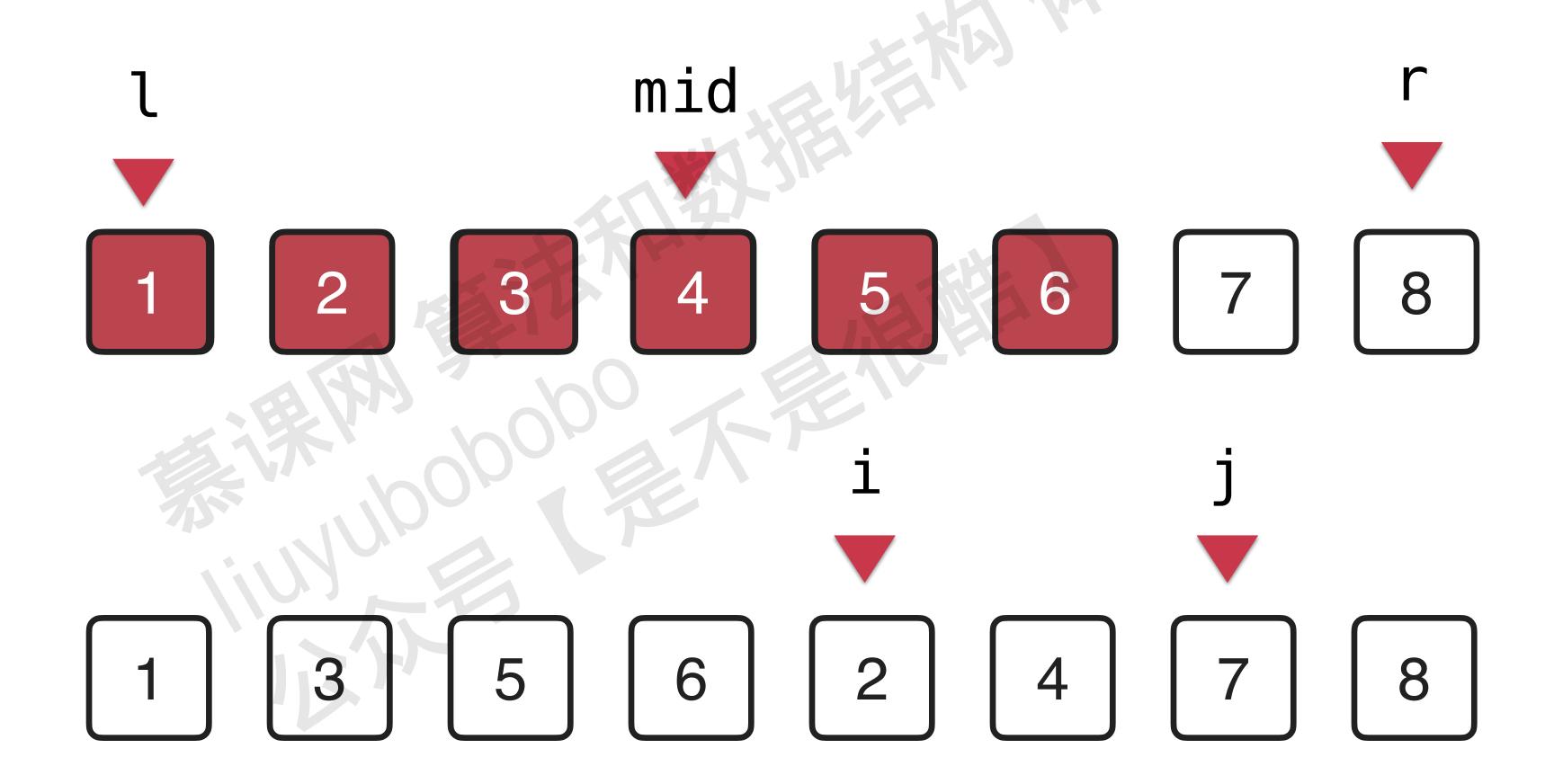


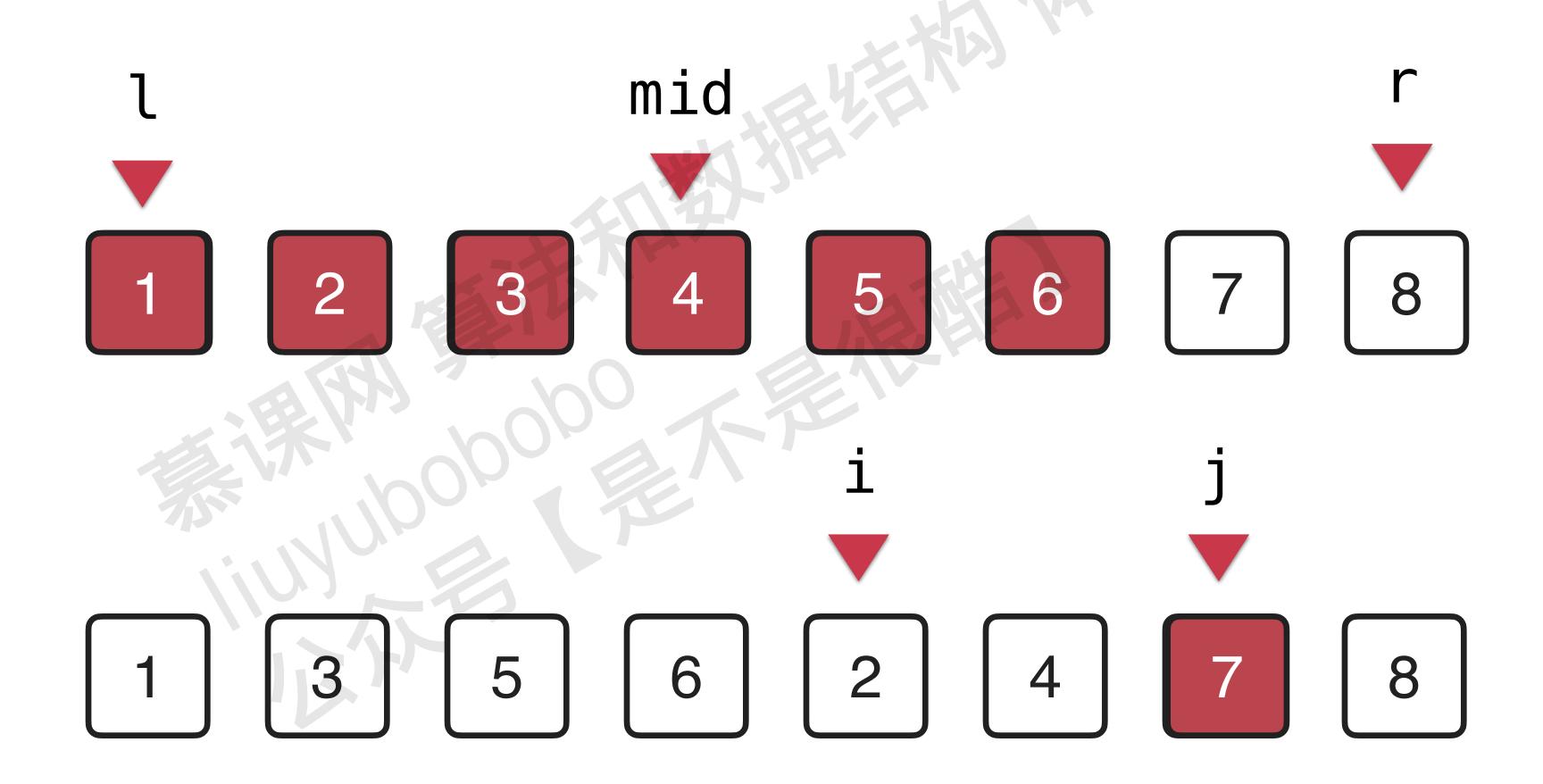


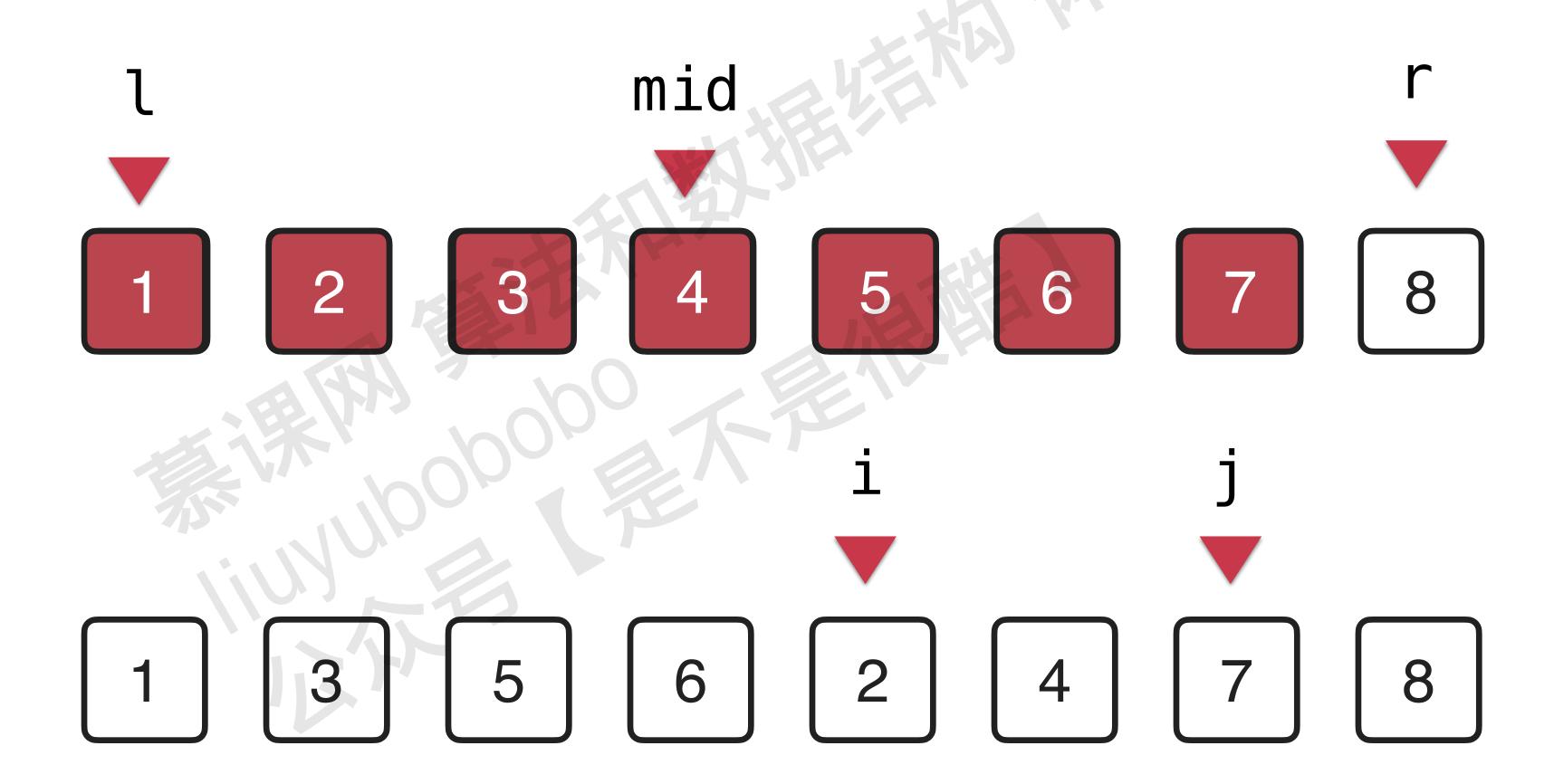


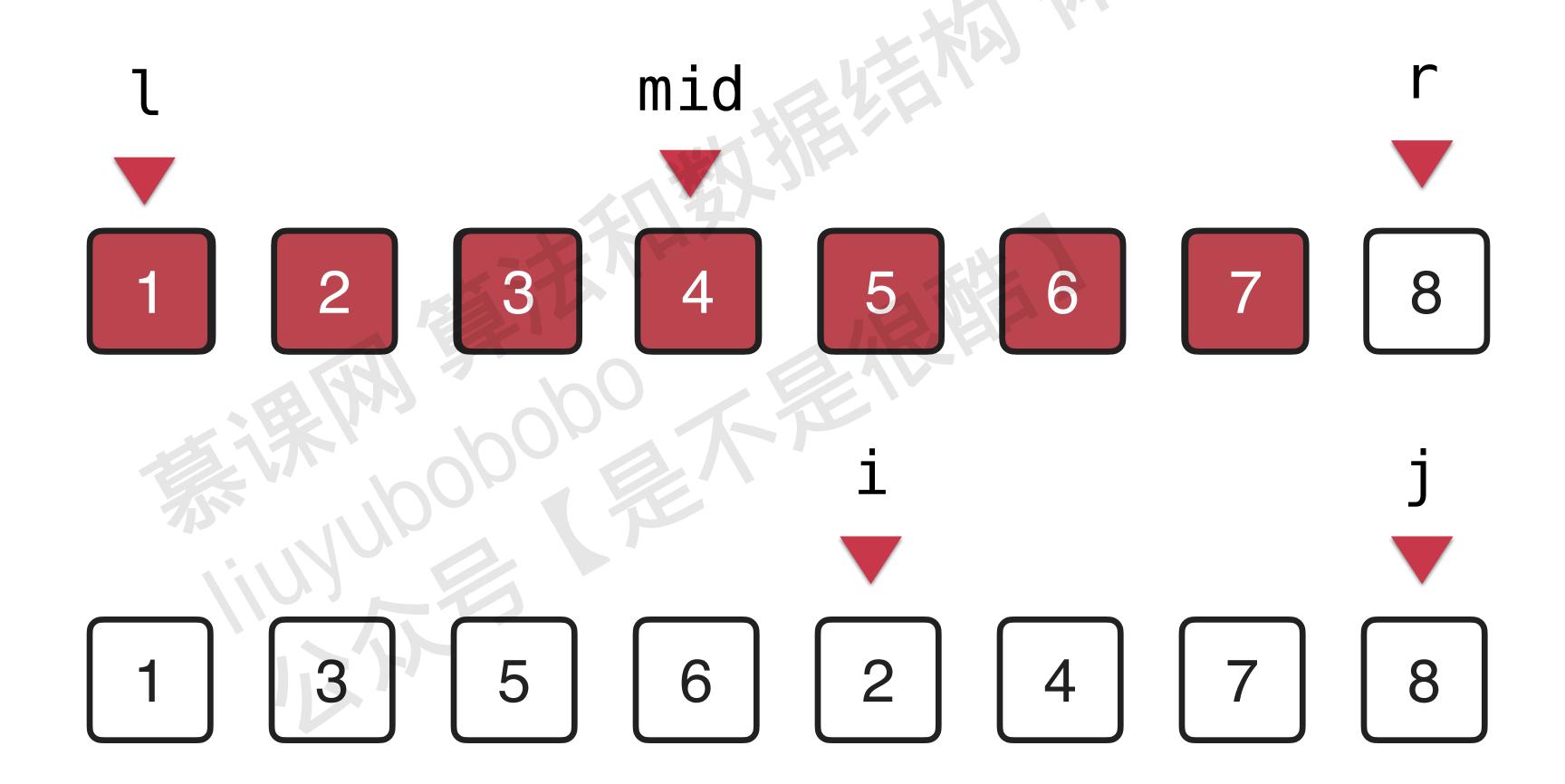


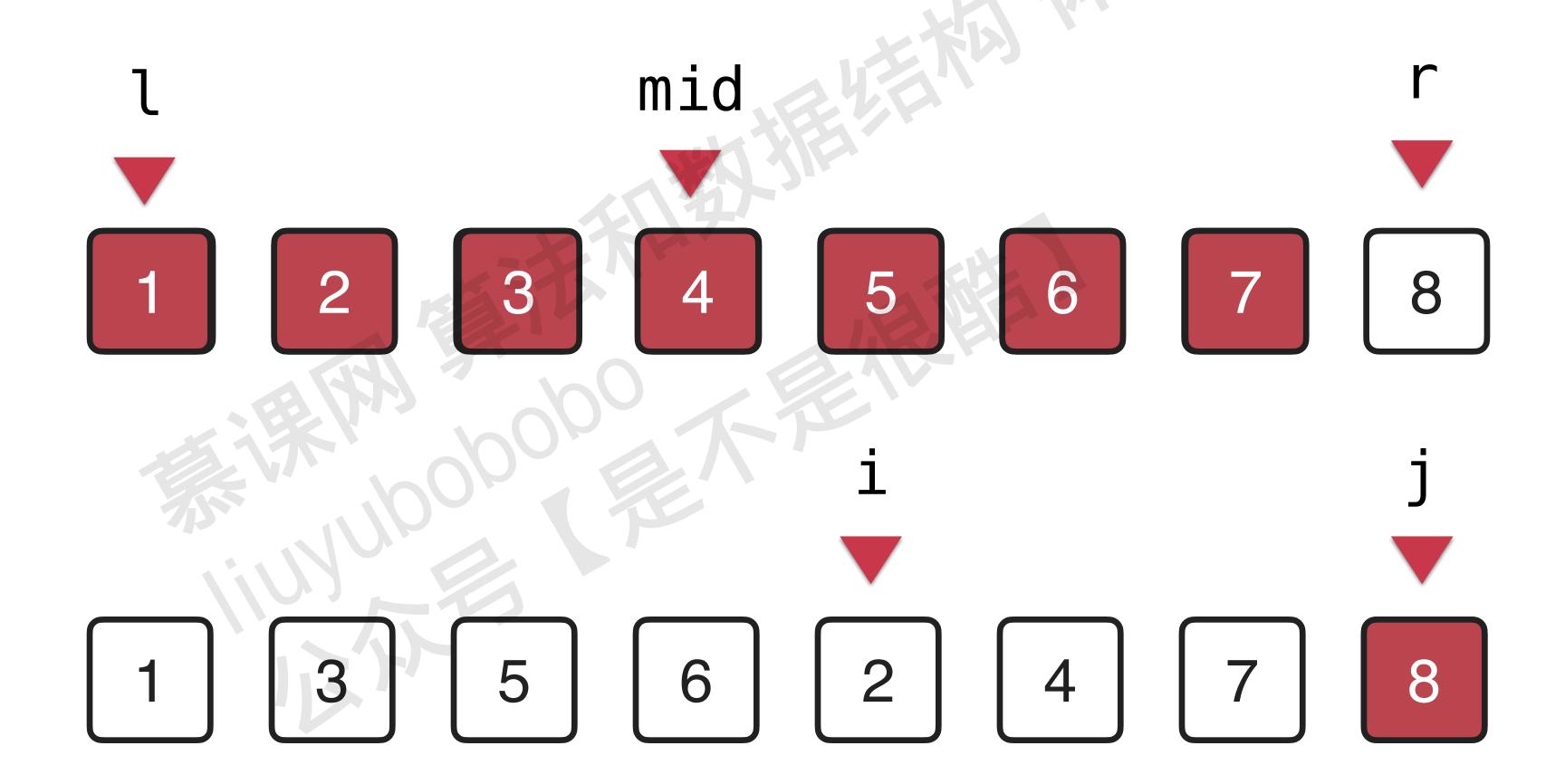


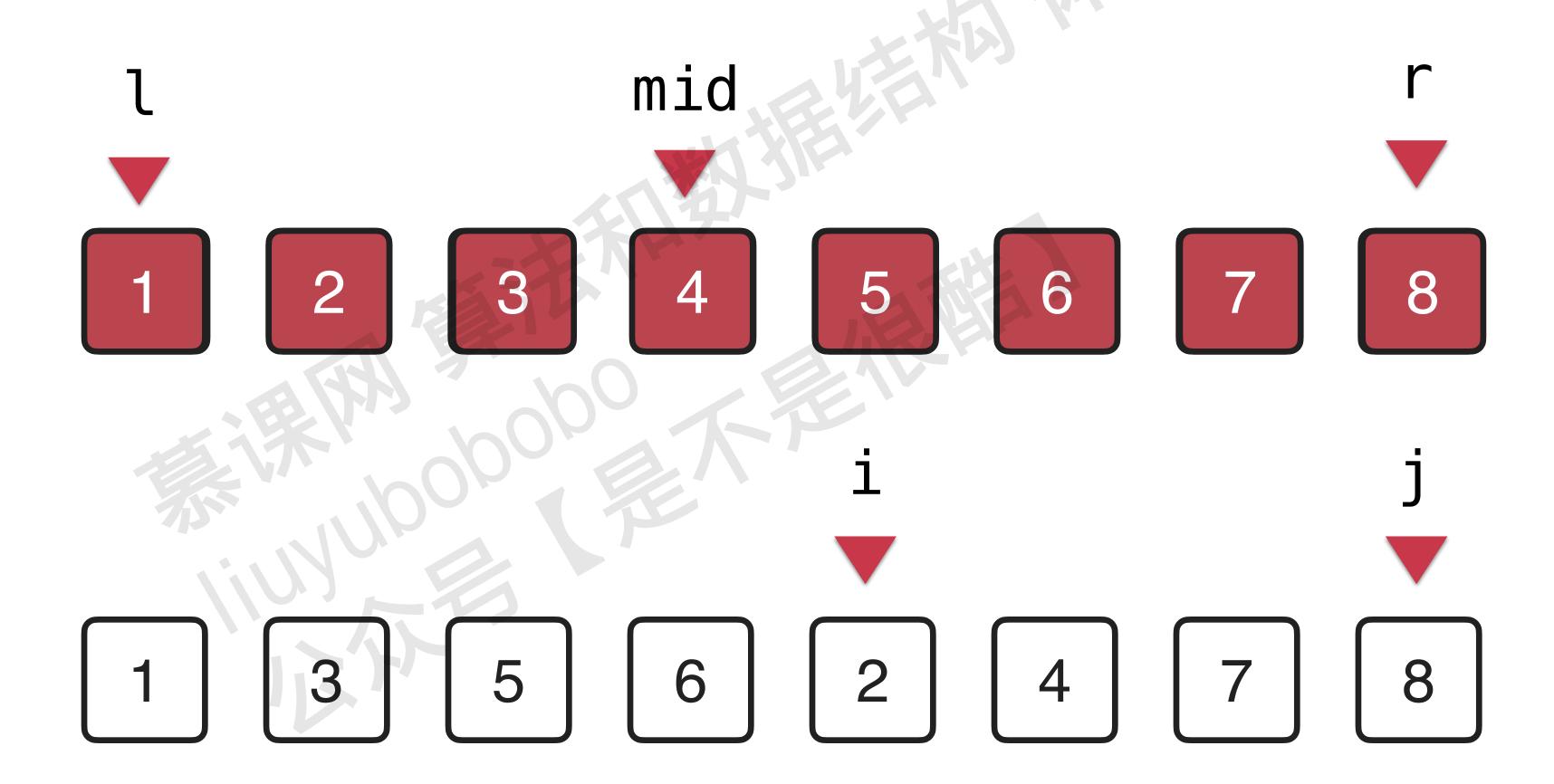




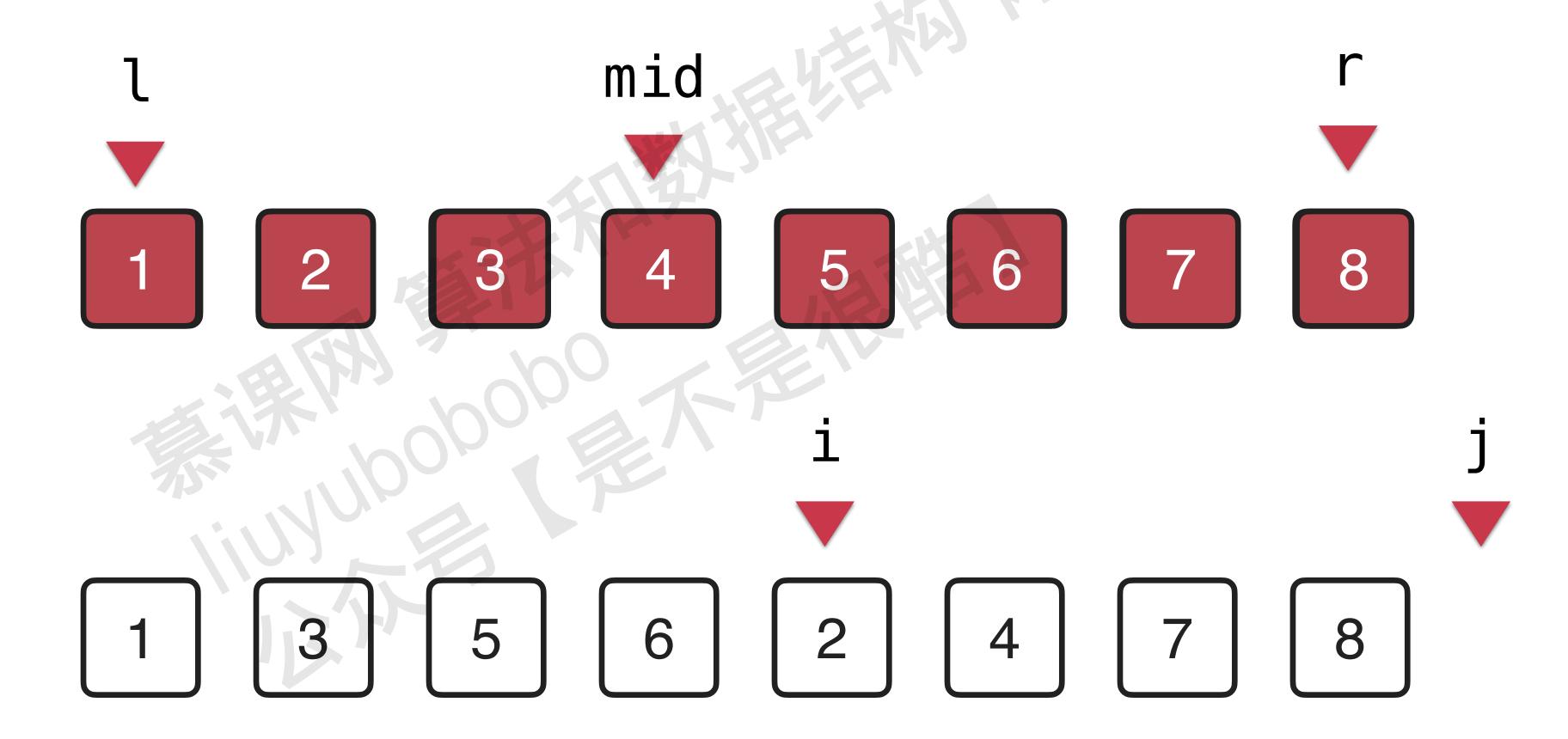








归并的过程无法原地完成



实现归并过程



实现归并排序法

归并排序法

```
MergeSort(arr, l, r){
   if(l >= r) return;
                            求解最基本问题
   int mid = (l + r) / 2;
   // 对 arr[l, mid] 进行排序
   MergeSort(l, mid);
                                       把原问题转化成
                                       更小的问题
   // 对 arr[mid + 1, r] 进行排序
   MergeSort(mid + 1, r);
   // 将arr[l,mid]和arr[mid+1,r]合并
   merge(arr, l, mid, r);
```

实践:实现归并排序法



THE WAY OF THE TENT OF THE TEN

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

7 1 4 2 8 3 6 5

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 7)

mergeSort(arr, l=0, r=7){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                 mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                  \rightarrow if(l >= r) return;
   mergeSort(arr, l, mid);
                                     mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                     mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                     merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

mergeSort(arr, l, r){

```
if(l >= r) return;
                                                                       mergeSort(arr, l, mid);
                                                                       mergeSort(arr, mid+1, r);
                                                                       merge(arr, l, mid, r);
                                  调用sort(arr, 0, 3)
                                                                   调用sort(arr, 0, 1)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
                                                                mergeSort(arr, l=0, r=1){
    if(l >= r) return;
                                    if(l >= r) return;
                                                                if(l >= r) return;
                                    mergeSort(arr, l, mid);
  mergeSort(arr, l, mid);
                                                                    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
                                                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                   merge(arr, l, mid, r);
                                                                    merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                    if(l >= r) return;
                                    mergeSort(arr, l, mid);
  mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                    merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
      if(l >= r) return;
      mergeSort(arr, l, mid);
      mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
   调用sort(arr, 0, 0)
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
  mergeSort(arr, l, mid);
                                    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

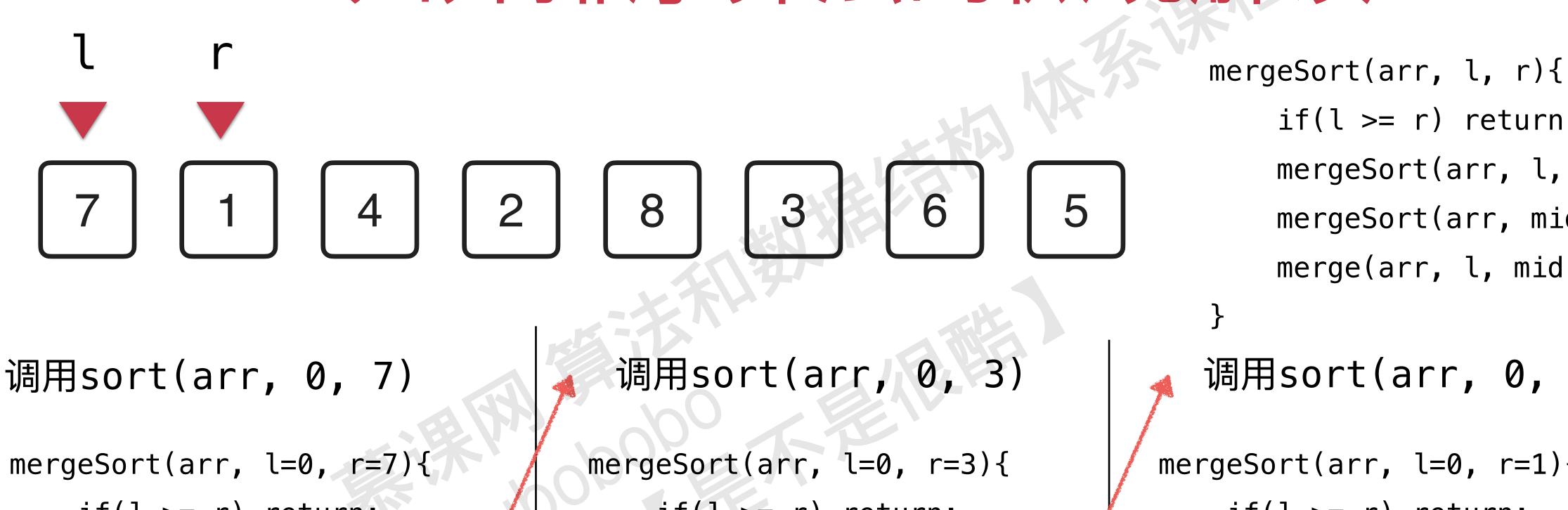
```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                    if(l >= r) return;
                                    mergeSort(arr, l, mid);
  mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                    merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
      if(l >= r) return;
      mergeSort(arr, l, mid);
      mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
   调用sort(arr, 1, 1)
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
                                    mergeSort(arr, l, mid);
  mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```



```
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                    if(l >= r) return;
                                    mergeSort(arr, l, mid);
mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                   merge(arr, l, mid, r);
```

```
if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
```



```
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
    mergeSort(arr, l=0, r=3){
        if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
}
```

```
if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
                                   mergeSort(arr, l, mid);
  mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
                                mergeSort(arr, l=0, r=3){
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                   if(l >= r) return;
  mergeSort(arr, l, mid);
                                    mergeSort(arr, l, mid);
                                   mergeSort(arr, mid+1, r)
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
      merge(arr, l, mid, r);
  调用sort(arr, 0, 1)
mergeSort(arr, l=0, r=1){
    if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

```
mergeSort(arr, l=0, r=7){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
in用sort(arr, 0, 3)

mergeSort(arr, l=0, r=3){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
   mergeSort(arr, l, mid);
                                   mergeSort(arr, l, mid);
                                    mergeSort(arr, mid+1, r);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                   merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

```
1 7 4 2 8 3 6 5

適用sort(arr, 0, 7)

mergeSort(arr, l=0, r=7){
    if(l >= r) return;
    mergeSort(arr, l, mid);

mergeSort(arr, l, mid);
```

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
       merge(arr, l, mid, r);
  调用sort(arr, 2, 3)
mergeSort(arr, l=2, r=3){
if(l >= r) return;
   mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
                                mergeSort(arr, l=0, r=3){
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                   if(l >= r) return;
  mergeSort(arr, l, mid);
                                   mergeSort(arr, l, mid);
                                   mergeSort(arr, mid+1, r);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
       merge(arr, l, mid, r);
  调用sort(arr, 2, 3)
mergeSort(arr, l=2, r=3){
    if(l >= r) return;
mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
```

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
       merge(arr, l, mid, r);
  调用sort(arr, 2, 3)
mergeSort(arr, l=2, r=3){
    if(l >= r) return;
    mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
                                                                       if(l >= r) return;
                                                                       mergeSort(arr, l, mid);
                                                                       mergeSort(arr, mid+1, r);
                                                                       merge(arr, l, mid, r);
                                  调用sort(arr, 0, 3)
                                                                   调用sort(arr, 2, 3)
调用sort(arr, 0, 7)
                                mergeSort(arr, l=0, r=3){
                                                                mergeSort(arr, l=2, r=3){
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                    if(l >= r) return;
                                                                     if(l >= r) return;
  mergeSort(arr, l, mid);
                                    mergeSort(arr, l, mid);
                                                                    mergeSort(arr, l, mid);
                                    mergeSort(arr, mid+1, r);
    mergeSort(arr, mid+1, r);
                                                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                   merge(arr, l, mid, r);
                                                                 merge(arr, l, mid, r);
```

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

```
mergeSort(arr, l, r){
                                                                        if(l >= r) return;
                                                                        mergeSort(arr, l, mid);
                                                                        mergeSort(arr, mid+1, r);
                                                                       merge(arr, l, mid, r);
                                  调用sort(arr, 0, 3)
                                                                   调用sort(arr, 2, 3)
调用sort(arr, 0, 7)
                                mergeSort(arr, l=0, r=3){
                                                                 mergeSort(arr, l=2, r=3){
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                    if(l >= r) return;
                                                                     if(l >= r) return;
  mergeSort(arr, l, mid);
                                    mergeSort(arr, l, mid);
                                                                     mergeSort(arr, l, mid);
```

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

mergeSort(arr, mid+1, r);

merge(arr, l, mid, r);

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
       merge(arr, l, mid, r);
  调用sort(arr, 2, 3)
mergeSort(arr, l=2, r=3){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
                                mergeSort(arr, l=0, r=3){
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
                                   if(l >= r) return;
  mergeSort(arr, l, mid);
                                   mergeSort(arr, l, mid);
                                   mergeSort(arr, mid+1, r);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
       if(l >= r) return;
       mergeSort(arr, l, mid);
       mergeSort(arr, mid+1, r);
       merge(arr, l, mid, r);
  调用sort(arr, 2, 3)
mergeSort(arr, l=2, r=3){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

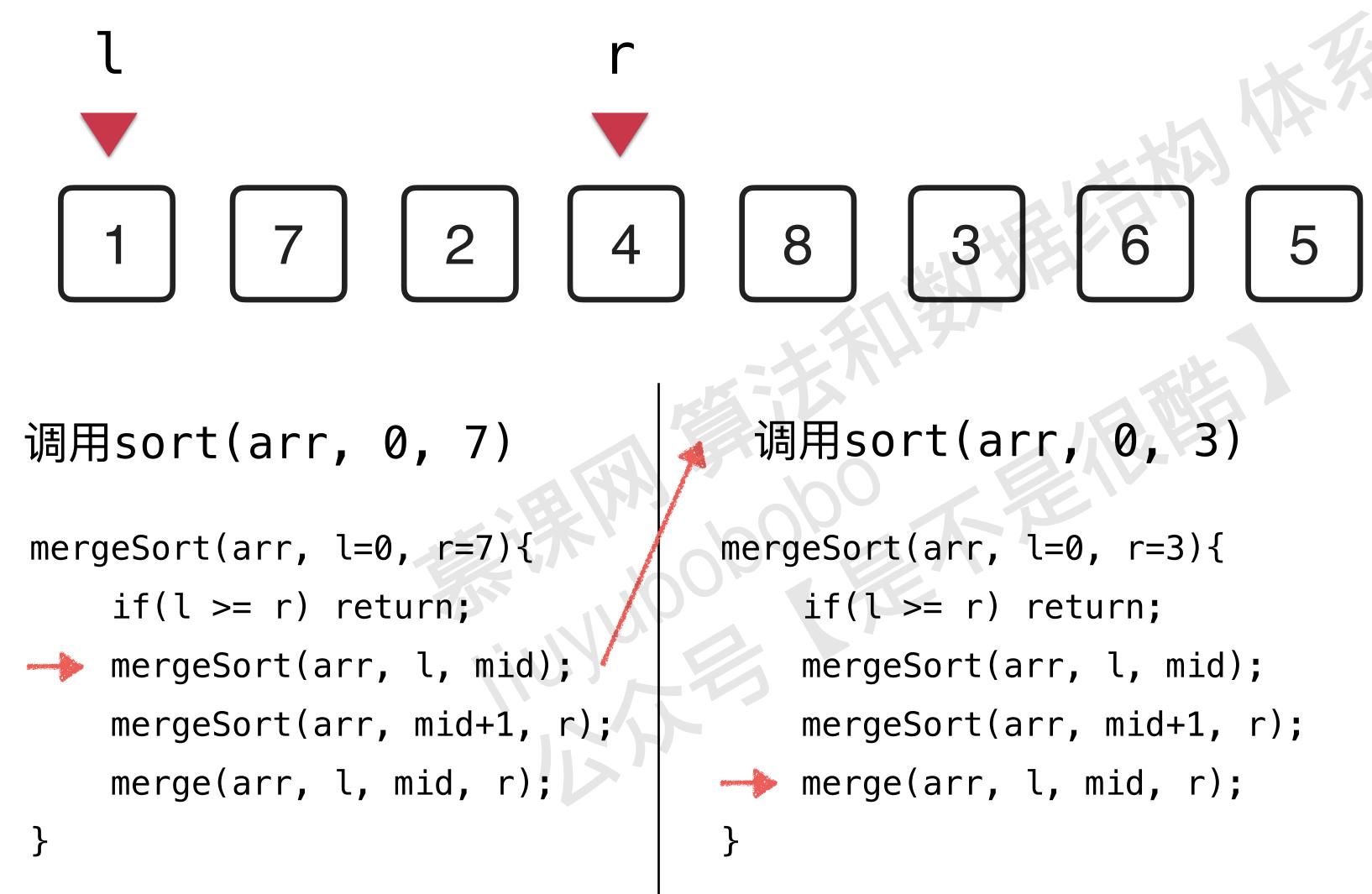
```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
   mergeSort(arr, l, mid);
                                   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
                                   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
                                   merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

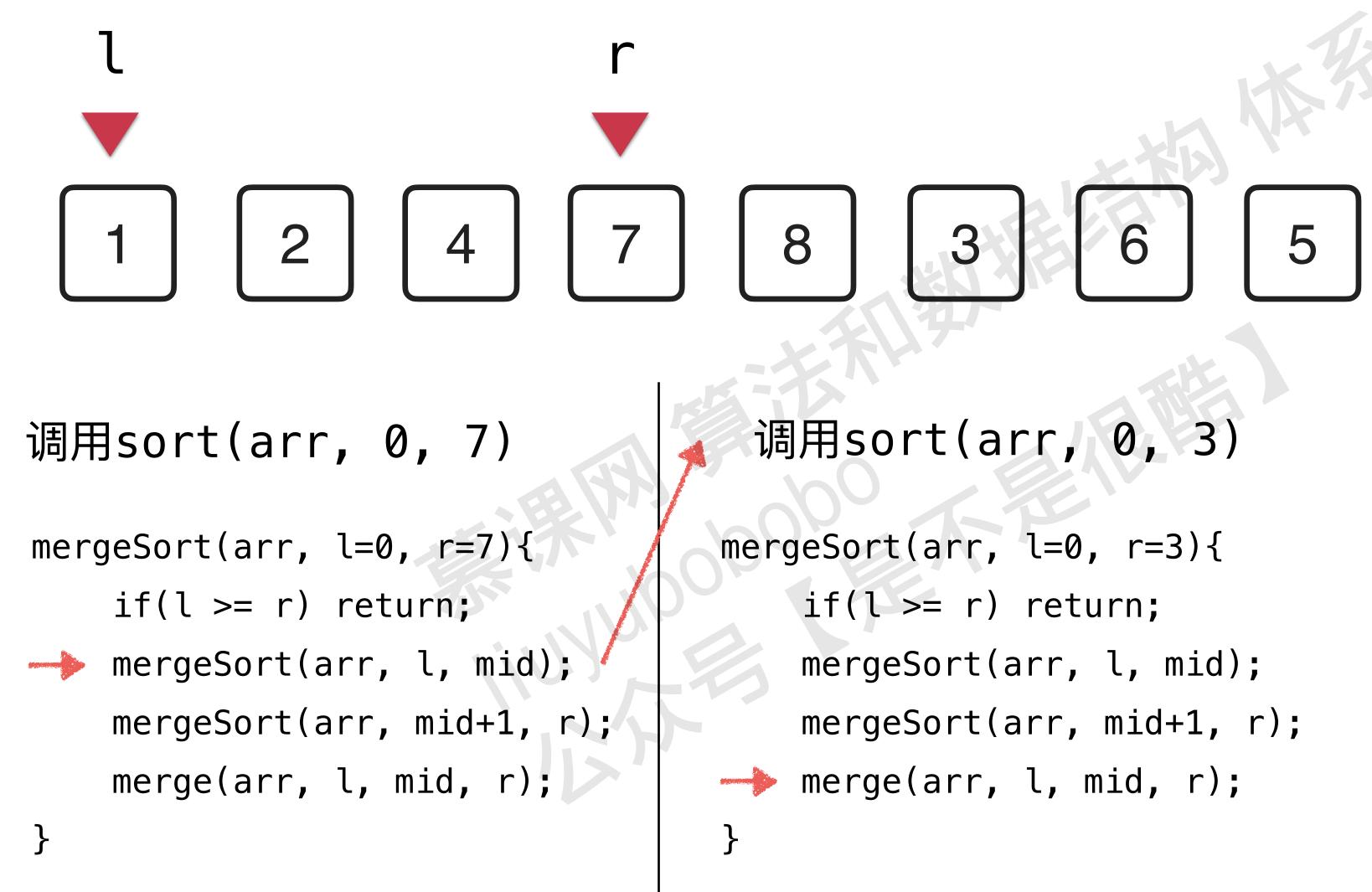
```
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
   mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
```

```
调用sort(arr, 0, 3)
                            mergeSort(arr, l=0, r=3){
                                if(l >= r) return;
                                mergeSort(arr, l, mid);
                                mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
}
merge(arr, l, mid, r);
}
```

```
mergeSort(arr, l, r){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 7)
```

```
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
   mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
```

```
调用sort(arr, 0, 3)
                            mergeSort(arr, l=0, r=3){
                                if(l >= r) return;
                                mergeSort(arr, l, mid);
                                mergeSort(arr, mid+1, r);
merge(arr, l, mid, r);
}
merge(arr, l, mid, r);
}
```

```
mergeSort(arr, l, r){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
    merge(arr, l, mid, r);
```

```
调用sort(arr, 0, 3)
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
                                mergeSort(arr, l=0, r=3){
    if(l >= r) return;
                                   if(l >= r) return;
   mergeSort(arr, l, mid);
                                    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
                                    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
                                  merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
   mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```



```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
调用sort(arr, 0, 7)
mergeSort(arr, l=0, r=7){
    if(l >= r) return;
    mergeSort(arr, l, mid);
    mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
```

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

```
illian in the image is a second of the image. It is a second of the image is a second of the image is a second of the image. It is a second of the image is a second of the image. It is a second of the image is a second of the image. It is a second of th
```

mergeSort(arr, l, r){
 if(l >= r) return;
 mergeSort(arr, l, mid);
 mergeSort(arr, mid+1, r);
 merge(arr, l, mid, r);
}

1 2 3 4 5 6 7 8

```
mergeSort(arr, l, r){
   if(l >= r) return;
   mergeSort(arr, l, mid);
   mergeSort(arr, mid+1, r);
   merge(arr, l, mid, r);
}
```

作业:打印输出看归并排序的过程



作业解析:打印输出看归并排序的过程

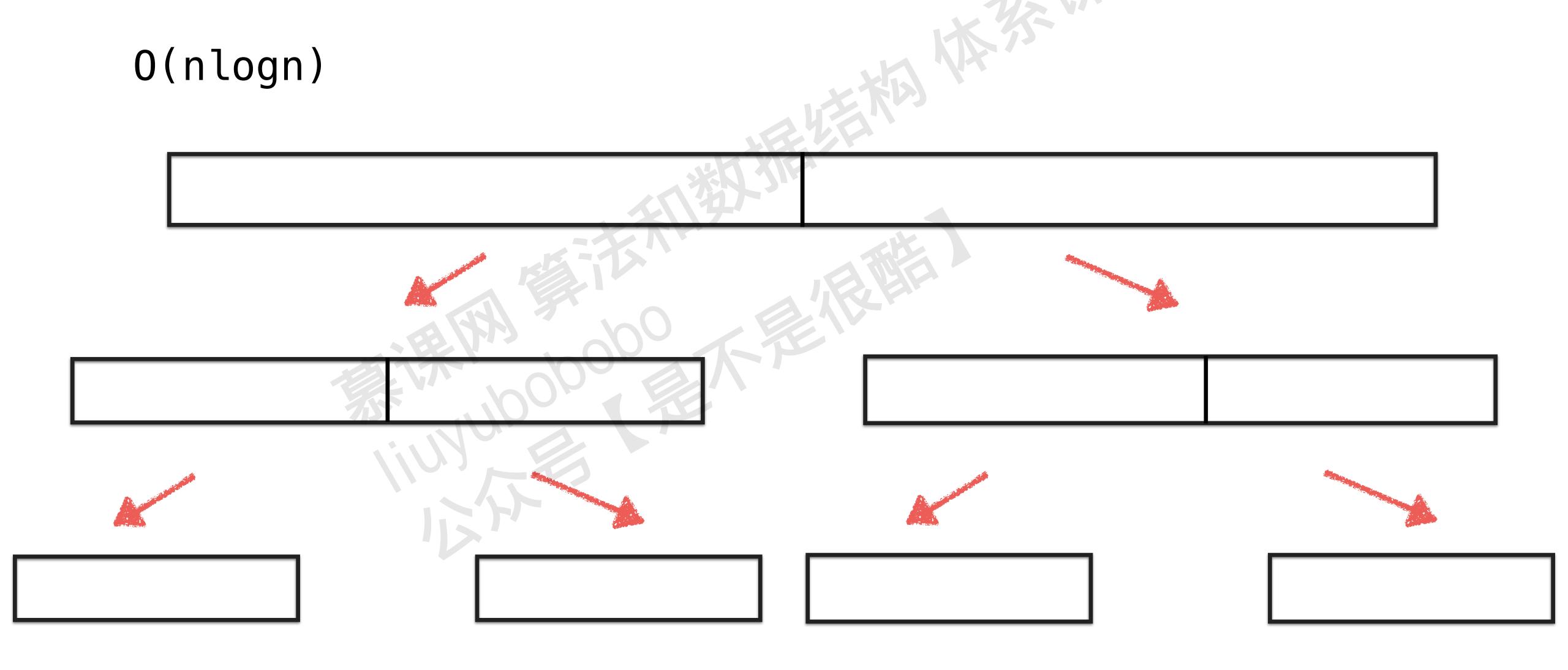


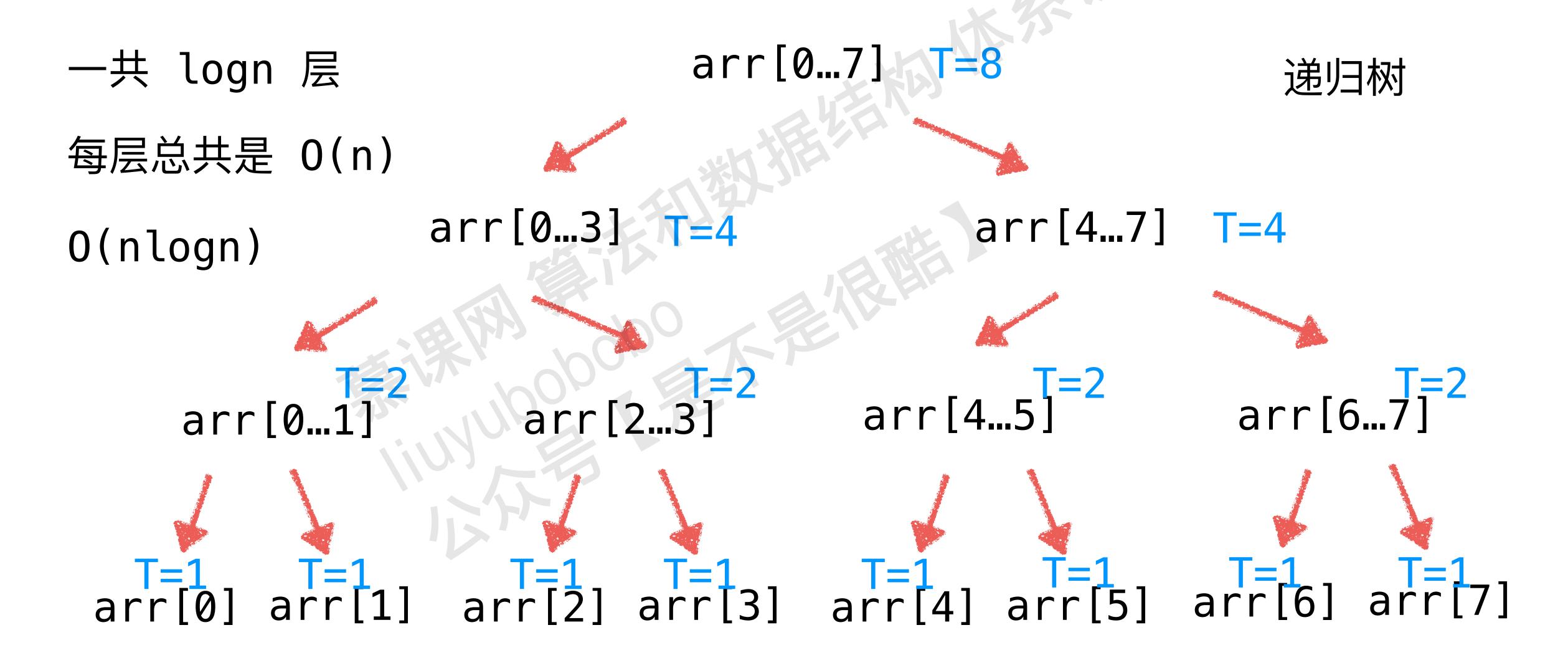
实践:归并排序法的性能测试

O(nlogn)

nlogn 比 n^2 快多少?

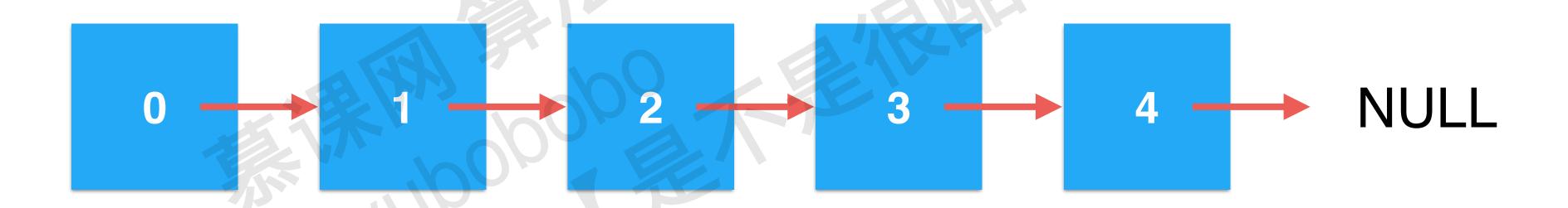
	n^2	nlogn	faster
n = 10	100	33	3
n = 100	10000	664	15
n = 1000	10^6	9966	100
n = 10000	10^8	132877	753
n = 100000	10^10	1660964	6020





递归函数的复杂度分析 递归树

回忆链表? 也有递归树



[0...4] 删除 → [1...4] 删除 → [2...4] 删除 → [3...4] 删除 → [4] 删除

递归树退化成链表

递归函数的复杂度分析 递归树

主定理

过于理论化,对于一般面试没必要掌握

如果感兴趣可以参考《算法导论》

其他

欢迎大家关注我的个人公众号:是不是很酷



算法与数据结构体系课程

liuyubobobo