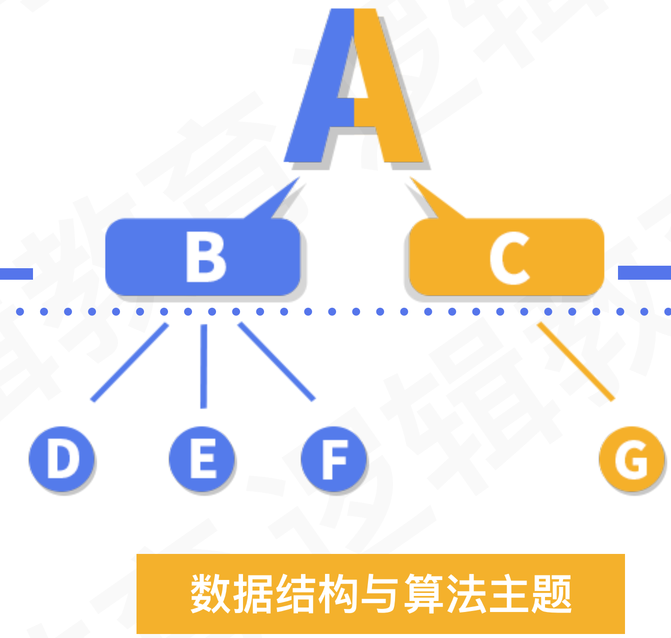
Hello 数据结构与算法

数据结构与算法 —【排序专题】



@CC老师

全力以赴 . 非同凡"想"

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

排序

假设含有n个记录的序列为($r_1,r_2,......,r_n$). 其相应的关键字分别为{$k_1,k_2,......,k_n$}. 需确定1,2,......,n 的一种排序$p_1,p_2,......p_n$. 使其相应的关键字满足$k_{p1} <= k_{p2} <= ...... <= k_{pn}$ 非递减(或非递增)关系. 即使得到序列成为一个按关键字有序的序列($r_{p1},r_{p2},...,r_{pn}$).这样得出操作称为**排序**

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

排序的分类

编号	姓名	语文	数学	英语	物理	化学	历史	政治	生物	地理	总分	主科
1	A	110	105	118	82	88	79	84	96	97	859	333
2	B	105	99	101	78	79	84	80	90	95	811	305
3	C	118	118	112	91	95	97	91	95	91	908	348
4	D	101	95	98	84	89	96	83	98	93	837	294

湖南主科: 120分

湖南非主科: 100分

总分: 960

课程研发:CC老师
课程授课:CC老师

排序的分类

- 内排序:是在排序整个过程中,待排序的所有记录全部被放置在内存中;
- 外排序:由于排序的记录个数太多,不能同时放置在内存,整个排序过程需要在内外存之间多次交换数据才能进行

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

排序的结构设计 与 交换函数实现

```c
//1.排序算法数据结构设计
//用于要排序数组个数最大值，可根据需要修改
#define MAXSIZE 10000
typedef struct
{
    //用于存储要排序数组，r[0]用作哨兵或临时变量
    int r[MAXSIZE+1];
    //用于记录顺序表的长度
    int length;
}SqList;
```

```c
//2.排序常用交换函数实现
//交换L中数组r的下标为i和j的值
void swap(SqList *L ,int i ,int j)
{
    int temp=L->r[i];
    L->r[i]=L->r[j];
    L->r[j]=temp;
}
```

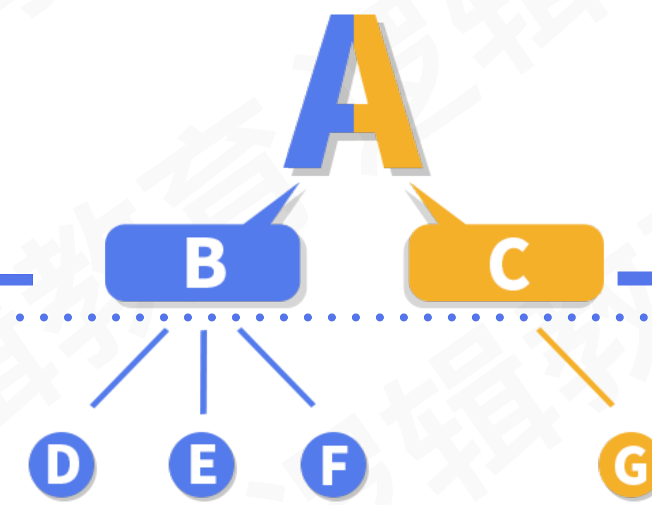
```c
//3.数组打印
void print(SqList L)
{
    int i;
    for(i=1;i< L.length ;i++)
        printf("%d,",L.r[i]);
    printf("%d",L.r[i]);
    printf("\n");
}
```

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

Hello 数据结构与算法

冒泡排序 (Bubble Sort)



数据结构与算法主题

@CC老师

全力以赴 . 非同凡"想"

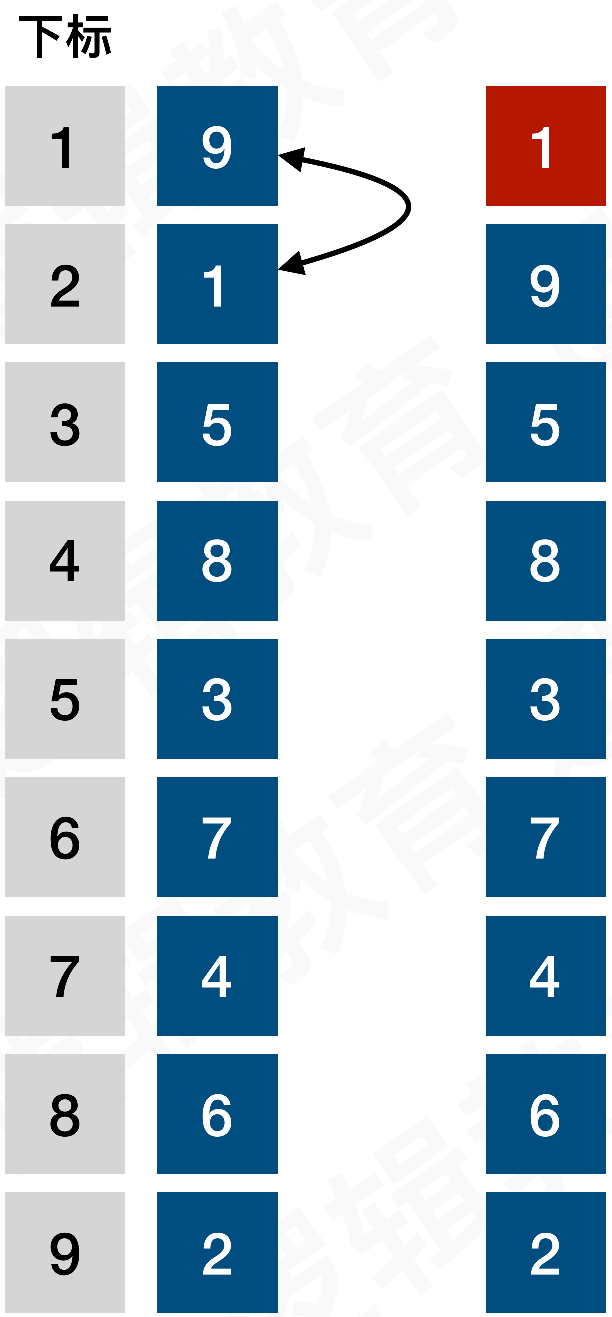
课程研发:CC老师
课程授课:CC老师

冒泡排序(Bubble Sort)

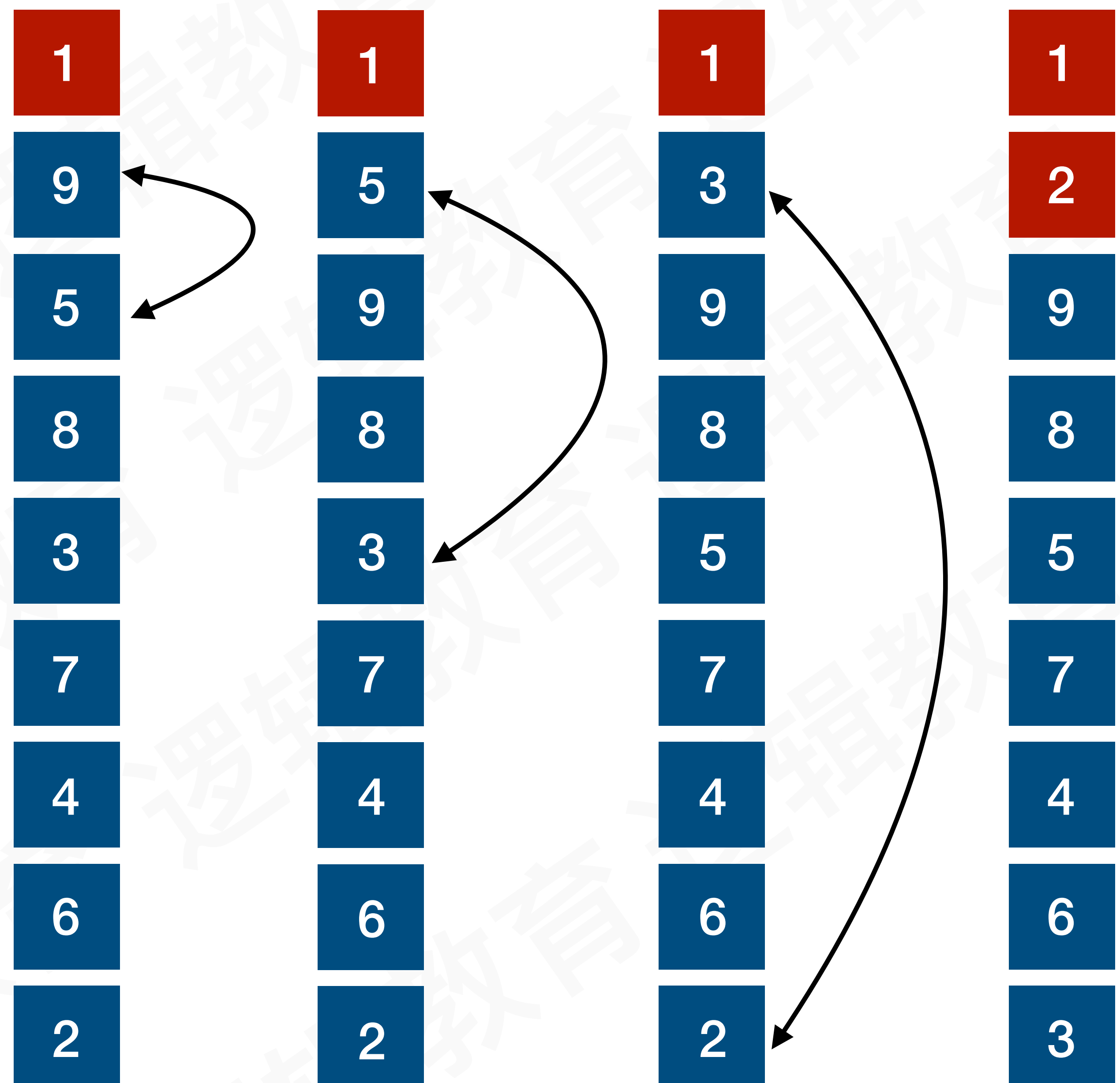
冒泡排序(Bubble Sort) 一种交换排序,它的基本思想就是: 两两比较相邻的记录的关键字,如果反序则交换,直到没有反序的记录为止.

课程研发:CC老师
课程授课:CC老师

冒泡排序(Bubble Sort) — 初级版本

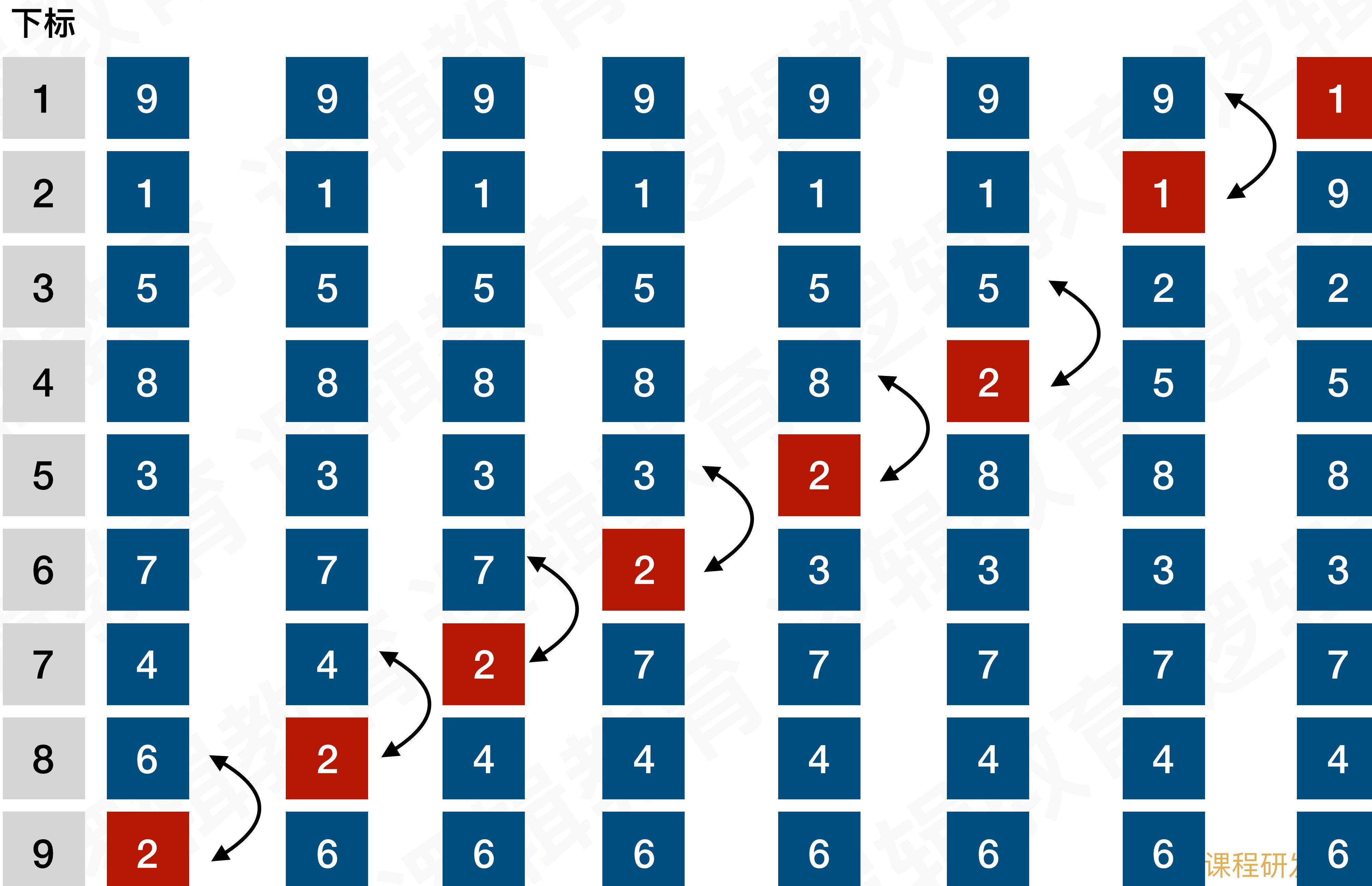


当i=1时，9与1交换后,1与其他关键字比较均是最小.因此1即最小值放置在首位



当i=2时，9与5，5与3,3与2,交换.最终将2放置第二位

冒泡排序(Bubble Sort) — 完成形态



当 i = 1 时,将最小值1冒泡到顶端

逻辑教育
Logic education

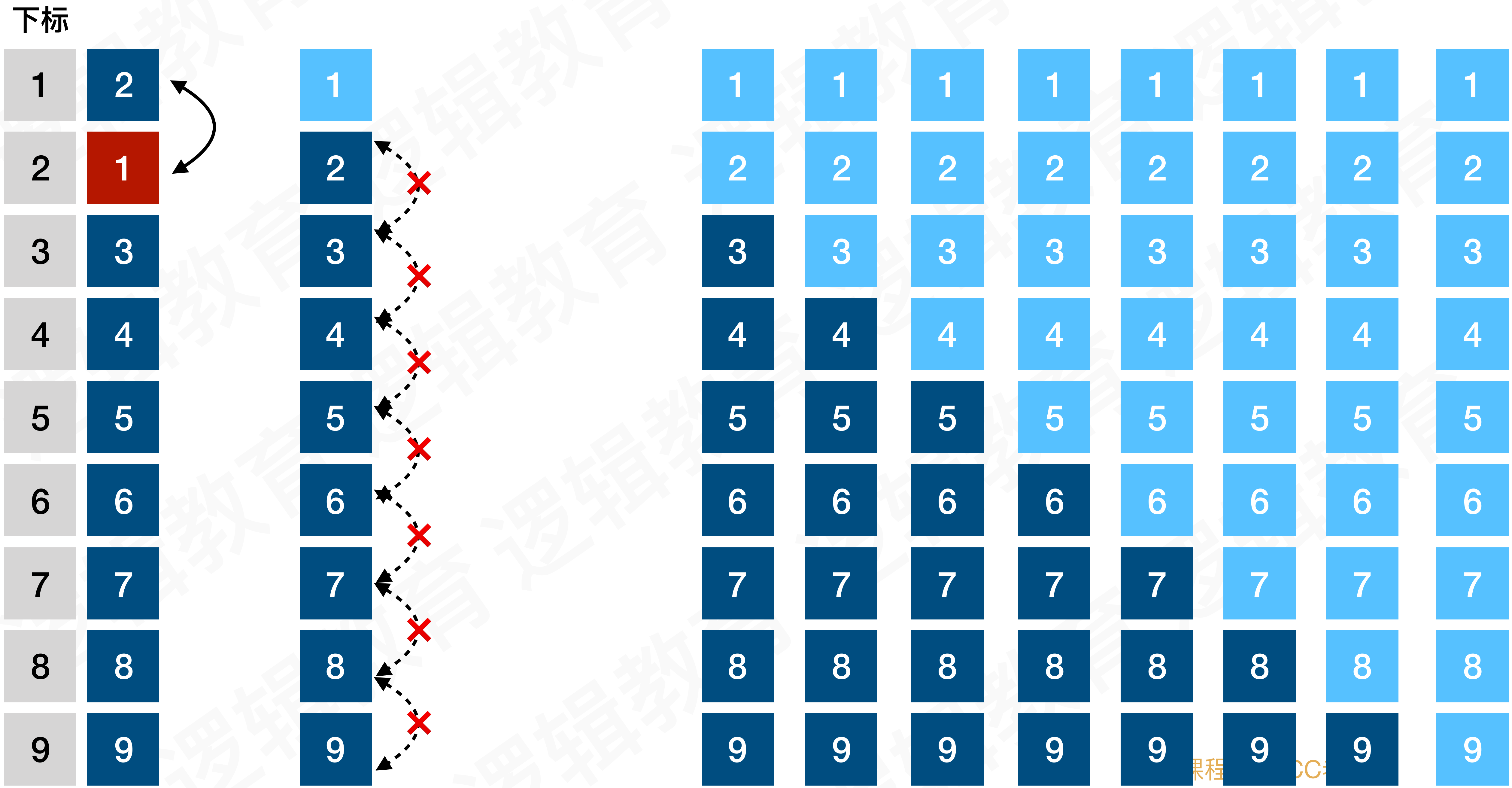
冒泡排序(Bubble Sort) — 完成形态

下标					
1	1	1	1	1	1
2	9	9	9	9	2
3	2	2	2	2	9
4	5	5	5	3	3
5	8	8	3	5	5
6	3	3	8	8	8
7	7	4	4	4	4
8	4	7	7	7	7
9	6	6	6	6	6

当 i = 2 时,将次小值2冒泡到顶端

课程研发:CC老师
课程授课:CC老师

冒泡排序(Bubble Sort) — 优化



当i=1时,将1和2的位置进行了交换;

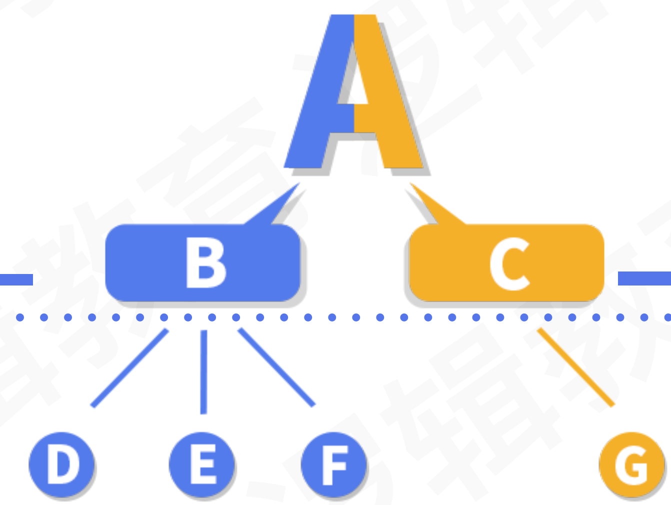
当i=2时,由于没有任何数据交换,就说明此序列以及有序

之后所有的循环判断都是多余的

逻辑教育
Logic education

Hello 数据结构与算法

简单选择排序(Simple Selection Sort)



数据结构与算法主题

@CC老师

全力以赴 . 非同凡"想"

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

简单选择排序(Simple Selection Sort)

简单排序算法(*Simple Selection Sort*) 就是通过$n-i$次关键词比较,从$n-i+1$个记录中找出关键字最小的记录,并和第$i(1<=i<=n)$ 个记录进行交换.

下标	0	1	2	3	4	5	6	7	8	9
数据	/	9	1	5	8	3	7	4	6	2

交换
i = 1,min = 2

下标	0	1	2	3	4	5	6	7	8	9
数据	/	1	9	5	8	3	7	4	6	2

交换
i = 2,min = 9

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

简单选择排序(Simple Selection Sort)

简单排序算法(*Simple Selection Sort*) 就是通过$n-i$次关键词比较,从$n-i+1$个记录中找出关键字最小的记录,并和第$i(1<=i<=n)$ 个记录进行交换.

下标	0	1	2	3	4	5	6	7	8	9
数据	/	1	2	5	8	3	7	4	6	9

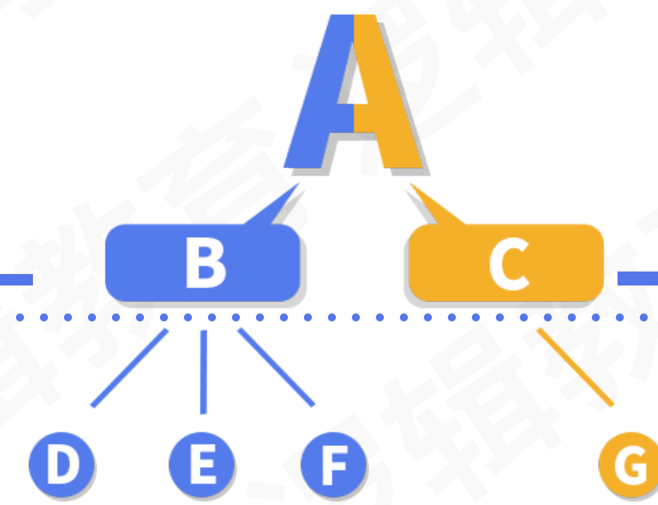
交换
i = 3,min = 5

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

Hello 数据结构与算法

直接插入排序(Straight Insertion Sort)



数据结构与算法主题

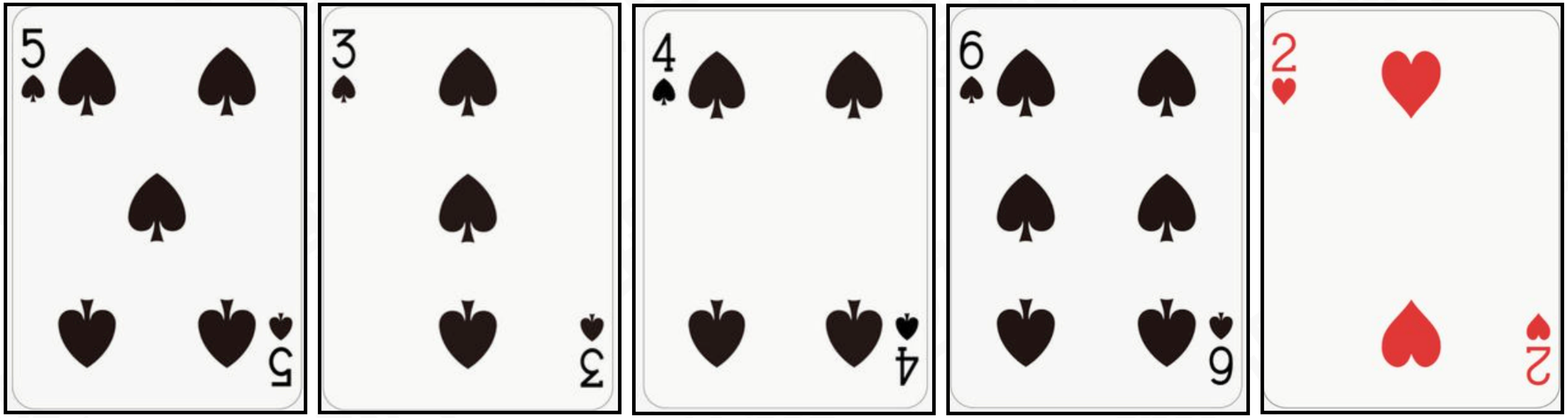
@CC老师

全力以赴．非同凡"想"

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort)

直接插入排序算法(Stright Insertion Sort)的基本操作是将一个记录插入到已经排好序的有序表中,从而得到一个新的,记录数增1的有序表;



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入3

下标	0	1	2	3	4	5
数据	/	5	3	4	6	2

temp: 3

待排序元素 i = 2

temp

1. 循环将i从第2个元素到最后一个元素作为待排序元素;
2. 判断当前待排序元素 是否小于 待排序前一个元素(i-1). 如果小于则参与接下来的插入排序
   解读: 此时待排序元素为3, 而它前一个元素为5;
3. 使用临时变量temp 存储好当前待排序元素 temp = L-r[i]; 解读: temp = 3;
4. 循环遍历,找到元素2之前,能够插入的位置; 判断依据是从i-1 到 0 这个空间里. 那个位置能L->r[j] > temp; 则将L->r[j+1] = L->r[j]

下标	0	1	2	3	4	5
数据	/	5	3	4	6	2

temp: 3

j 循环区域 [0,i-1]

L->r[j] > temp

待排序元素 i = 2

temp

课程研发:CC老师
课程授课:CC老师

5. 找到元素5 > temp, 需要把5往前面移动,覆盖元素3;

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入3

下标	0	1	2	3	4	5
数据	/	5	5	4	6	2

3

j = 0
j = 2
temp

6. 此时r[0] 不大于temp 则j层循环结束. 目前 j = 0
7. 此时 需要把3 覆盖到j=1的位置;但是由于j 退出循环时等于0, 所以是r[j+1] = temp

下标	0	1	2	3	4	5
数据	/	3	5	4	6	2

3

j = 0
j = 2
temp

r[j + 1] = temp

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入4

下标	0	1	2	3	4	5
数据	/	3	5	4	6	2

待排序元素 i=3

4

temp

当待排序元素r[i] < r[i-1] 所以可以参与插入排序

下标	0	1	2	3	4	5
数据	/	3	5	4	6	2

j 循环区域 [0,i-1]

r[2] > temp

待排序元素 i=3

4

temp

当j=2时,找到r[2] > temp

下标	0	1	2	3	4	5
数据	/	3	5	5	6	2

j = 2

r[j + 1] =r[j]

4

temp

将5覆盖, r[j+1] = r[j] 并且 j 自减

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort) 模拟插入4

下标	0	1	2	3	4	5
数据	/	3	5	5	6	2

j = 1

r[j + 1] =r[j]

4

temp

j = 1, 循环判断 r[j] < temp,则j层循环退出. 当时j = 1;

下标	0	1	2	3	4	5
数据	/	3	4	5	6	2

j = 1

r[j + 1] = temp

4

temp

将temp 覆盖j +1的位置,因为 j 退出循环是等于1 ,所以应该是 r[j + 1] = temp

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入 6

下标	0	1	2	3	4	5
数据	/	3	4	5	6	2

待排序元素 i=4

6

temp

待排序元素 r[i] > r[i-1] 所以是有序的.不需要进行插入

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort) 模拟插入 2

下标	0	1	2	3	4	5
数据	/	3	4	5	6	2

待排序元素 i=5

2

temp

当待排序元素r[i] < r[i-1] 所以可以参与插入排序

第一次j循环:

下标	0	1	2	3	4	5
数据	/	3	4	5	6	2

j 循环区域 [0,i-1]

j = i-1 = 4　i = 5

2

temp

当j=4时,找到r[4] > temp

下标	0	1	2	3	4	5
数据	/	3	4	5	6	6

j = 3　i = 5

2

temp

将6覆盖, r[j+1] = r[j] 并且 j 自减 j = 3

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入2

第二次j循环:

下标	0	1	2	3	4	5
数据	/	3	4	5	6	6

j = 3　　i = 5

2

temp

j = 3,r[3] > temp 循环继续

当j=3时,找到r[3] > temp

下标	0	1	2	3	4	5
数据	/	3	4	5	5	6

j = 2　　i = 5

2

temp

将5覆盖, r[j+1] = r[j] 并且 j 自减 j = 2

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 模拟插入2

第三次j循环:

下标	0	1	2	3	4	5
数据	/	3	4	5	5	6

j = 2　　i = 5

temp: 2

j = 2,r[2] > temp 循环继续

当j=2时,找到r[2] > temp

下标	0	1	2	3	4	5
数据	/	3	4	4	5	6

j = 1　　i = 5

temp: 2

将4覆盖, r[j+1] = r[j] 并且 j 自减 j = 1

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort) 模拟插入2

第四次j循环:

下标	0	1	2	3	4	5
数据	/	3	4	4	5	6

j = 1

i = 5

2

temp

j = 1,r[1] > temp 循环继续

当j=1时,找到r[1] > temp

下标	0	1	2	3	4	5
数据	/	3	3	4	5	6

j = 0

i = 5

2

temp

将3覆盖, r[j+1] = r[j] 并且 j 自减 j = 0

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort) 模拟插入2

第五次j循环:

下标	0	1	2	3	4	5
数据	/	3	3	4	5	6

j = 0

i = 5

2

temp

j = 0, 循环判断 r[0] < temp,则j层循环退出. 当时j = 0;

r[0] < temp 循环终止!

课程研发:CC老师
课程授课:CC老师

直接插入排序(Straight Insertion Sort) 模拟插入2

直接插入排序完成

下标	0	1	2	3	4	5
数据	/	2	3	4	5	6

j = 0

r[j + 1] = temp

i = 5

2

temp

将temp 覆盖j +1的位置,因为 j 退出循环是等于0 ,所以应该是 r[j + 1] = temp;
将temp 插入到r[1] 的位置

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

直接插入排序(Straight Insertion Sort) 复杂度分析

空间复杂度: O(1)
解读:在直接插入排序中只使用了`i,j,temp`这三个辅助元素，与问题规模无关，空间复杂度为`O(1)`

时间复杂度: $O(n^2)$

最好的情况: 顺序序列排序,例如{2,3,4,5,6}.
此时比较次数（C _{min}）和移动次数（M_{min}）达到最小值。
$C_{min}=n-1$
$M_{min}=0$

当最坏的情况是,即排序的序列是逆序的情况,例如{6,5,4,3,2}
$C_{max} = 1+2+...+(n-1) = n(n-1)/2=O(n^2)$
$M_{max} = (1+2) + (2+2) +.....+ (n-1+2) = (n-1)(n+4)/2=O(n^2)$

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

课堂小游戏

Ⅰ、Ⅱ、Ⅲ、Ⅳ、Ⅴ、Ⅵ、Ⅶ、Ⅷ、Ⅸ、Ⅹ

游戏一，将罗马数字7(VII) 加一笔变成8(VIII)

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

课堂小游戏

Ⅰ、Ⅱ、Ⅲ、Ⅳ、Ⅴ、Ⅵ、Ⅶ、Ⅷ、Ⅸ、Ⅹ

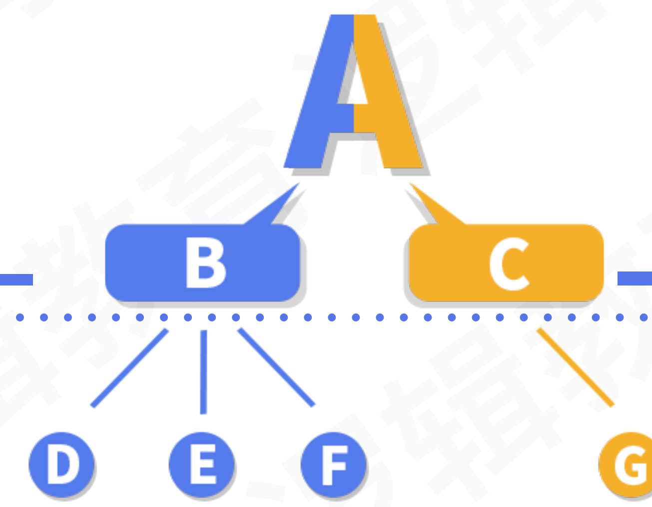
游戏二, 将罗马数字9 ,也就是”IX”. 加一笔变成6. 应该怎么做?

IX

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

Hello 数据结构与算法



希尔排序原理(Shell Sort)

数据结构与算法主题

@CC老师

全力以赴 . 非同凡"想"

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序原理(Shell Sort)

在插入排序之前,将整个序列调整成基本有序. 然后再对全体序列进行一次直接插入排序

下标	0	1	2	3	4	5	6	7	8	9
数据	/	9	1	5	8	3	7	4	6	2

分成3组,将其各自排序:

9 1 5　　8 3 7　　4 6 2

3组序列,局部排序后:

1 5 9　　3 7 8　　2 4 6

合并序列:

1 5 9 3 7 8 2 4 6

课程研发:CC老师
课程授课:CC老师

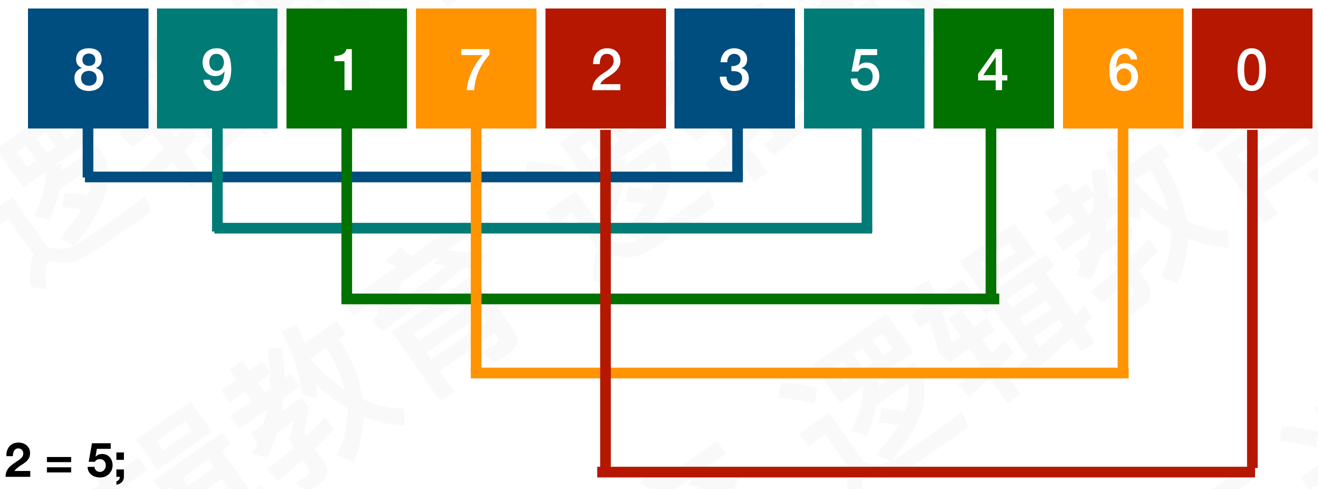
希尔排序原理(Shell Sort)

希尔排序思想: 希尔排序是把记录按下标的一定增量分组，对每组使用直接插入排序算法排序；随着增量逐渐减少，每组包含的关键词越来越多，当增量减至1时，整个序列恰被分成一组，算法便终止.

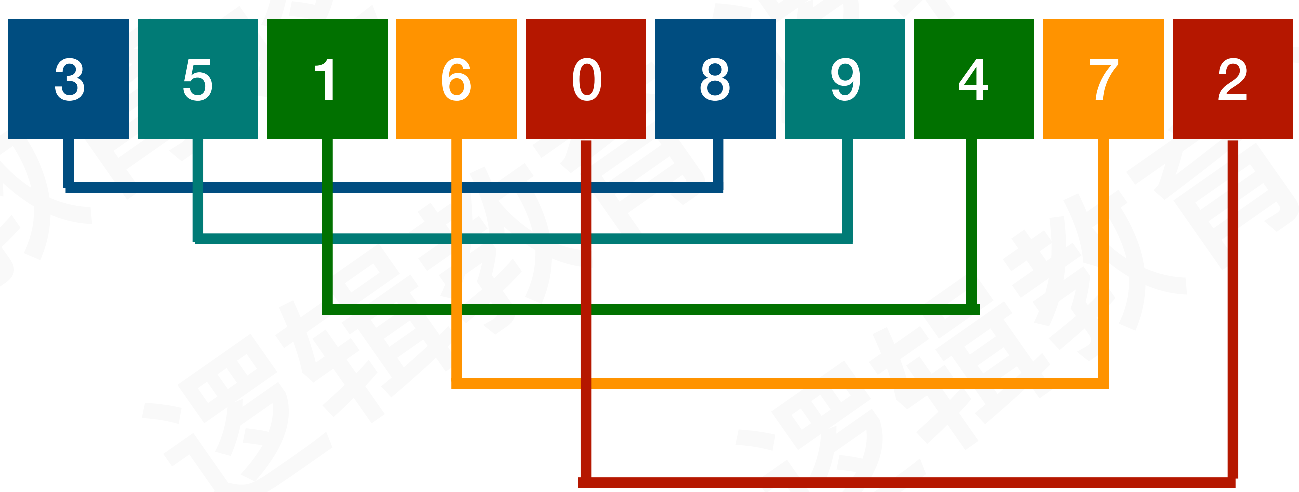
不是基本有序:	1	5	9	3	7	8	2	4	6
基本有序:	2	1	3	6	4	7	5	8	9

课程研发:CC老师
课程授课:CC老师

希尔排序原理(Shell Sort)

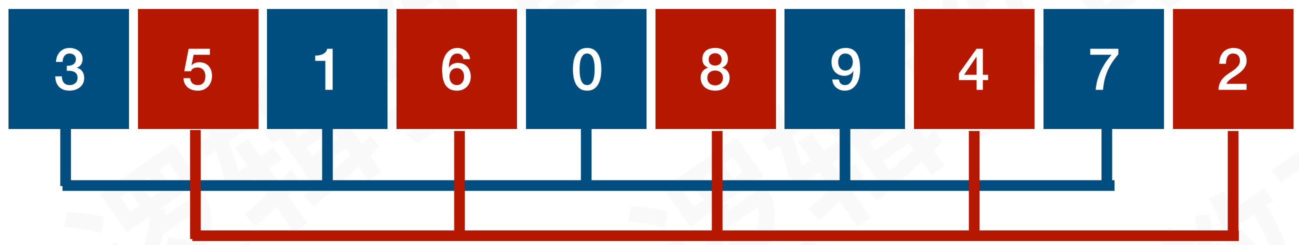


初始化 increment = Length / 2 = 5;
也就意味着整个数组被分割成 {8,3},{9,5},{1,4},{7,6},{2,0}
在这个分割中,进行部分直接插入排序
那么此时 3,5,6,0这些小元素就会被调整到前面

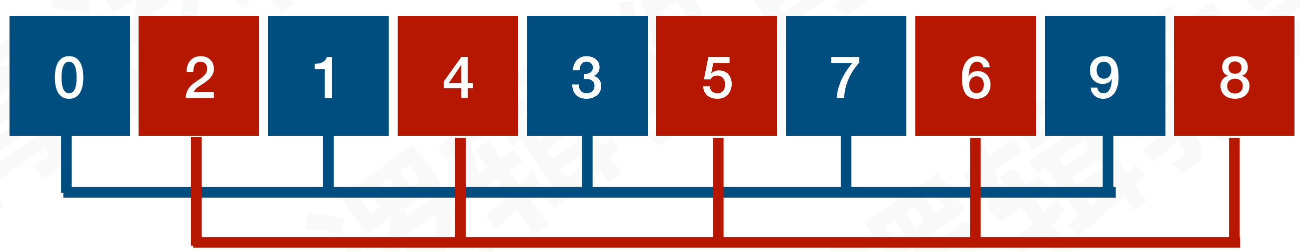


课程研发:CC老师
课程授课:CC老师

希尔排序原理(Shell Sort)



缩小增量: increment = increment / 2= 5/2 = 2;
数组被分为2组: {3,1,0,9,7} {5,6,8,4,2} 对这2个序列进行直接插入排序

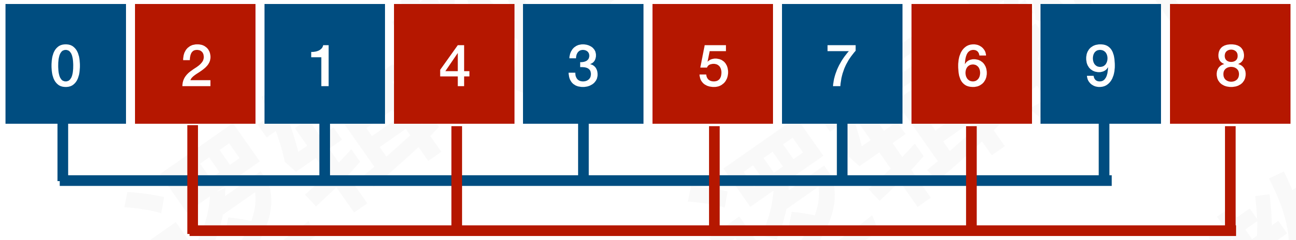


2个序列排序后: {0,1,3,7,9} {2,4,5,6,8} 最终这数组是 {0,2,1,4,3,5,7,6,9,8}

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序原理(Shell Sort)



缩小增量: increment = increment / 2= 2/2 = 1;
数组被分为1组:{0,2,1,4,3,5,7,6,9,8}对这1个序列进行直接插入排序

经过刚刚的调控,就得到了一个基本有序的数组. 在这个数组上进行直接插入排序.

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort)

下标	0	1	2	3	4	5	6	7	8	9
数据	/	9	1	5	8	3	7	4	6	2

缩小增量: increment = increment / 3+1=   9/3+1 = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	/	9	1	5	8	3	7	4	6	2

increment = 4

i = 5

i层循环 从increment + 1 到 length; 也就是从5到9;   从第5个元素到第9个元素都是待插入排序元素;

这里和插入排序的区别所在就是.插入排序增减量都是1. 就是与相邻的元素进行比较. 但是在希尔排序里. 是一组的元素才进行插入排序; 而3与9是一组; 1与7是一组; 5与4是一组; 8与6是一组; 同色系的数据直接才能进行插入排序

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort) — 模拟i=5循环

下标	0	1	2	3	4	5	6	7	8	9
数据	/	9	1	5	8	3	7	4	6	2

increment = 4

下标间隔是4

9 > 3,则交换

i = 1

i = 5

交换

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	3	9	1	5	8	9	7	4	6	2

j = 1

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素

j 循环就是为了将第1位上的9 赋值给第5位上; 退出j层循环时,j = -3

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort) — 模拟i=5循环

increment = 4　　L.r[0] 就是用来临时存储当前待插入的元素

下标	0	1	2	3	4	5	6	7	8	9
数据	3	3	1	5	8	9	7	4	6	2

↑

j = -3

j 循环就是为了将第1位上的9 赋值给第5位上; 退出j层循环时,j = -3

需要把L.r[0] 上的3 赋值到原来第1位上的空间. 所以L.r[j+increment] = L.r[0] = L[-3+4] = L[1] = 3

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort) — 模拟i=6循环

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort) — 模拟i=7循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	3	3	1	5	8	9	7	4	6	2

i = 3

下标间隔是4
5 > 4,所以交换

i = 7

交换

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	4	3	1	5	8	9	7	5	6	2

j = 3

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素

j 循环就是为了将第3位上的5 赋值给第7位上; 退出j层循环时,j =-1

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort) — 模拟i=7循环

increment = 4　　L.r[0] 就是用来临时存储当前待插入的元素

下标	0	1	2	3	4	5	6	7	8	9
数据	4	3	1	4	8	9	7	5	6	2

j = -1　　　　j = j + increment

j 循环就是为了将第3位上的5 赋值给第7位上; 退出j层循环时,j = -1

需要把L.r[0] 上的4 赋值到原来第3位上的空间. 所以L.r[j+increment] = L.r[0] = L[-1+4] = L[3] = 4

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort) — 模拟i=8循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	4	3	1	4	8	9	7	5	6	2

i = 4

下标间隔是4
8 > 6,所以交换

i = 8

交换

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	6	3	1	4	8	9	7	5	6	2

j = 4

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素

j 循环就是为了将第4位上的8 赋值给第8位上; 退出j层循环时,j = 0

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort) — 模拟i=8循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	6	3	1	4	6	9	7	5	8	2

j = 0

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素

j 循环就是为了将第4位上的8 赋值给第8位上; 退出j层循环时,j = 0

需要把L.r[0] 上的6 赋值到原来第4位上的空间. 所以L.r[j+increment] = L.r[0] = L[0+4] = L[4] = 6

逻辑教育
Logic education

希尔排序实现(Shell Sort) — 模拟i=9循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	6	3	1	4	6	9	7	5	8	2

i = 5

下标间隔是4
9 > 2,所以交换

i = 9

交换

i=9 第1次j层循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	2	3	1	4	6	9	7	5	8	9

j = 5

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素 2;
j 循环就是为了将第5位上的9 赋值给第9位上;
j = j - increment = 1;
判断j 层循环条件; j > 0 && L.r[j] > L.r[0] . 此时j = 1 大于满足条件①;
此时L.r[1] > L.r[0] -> 3 > 2 .循环j层循环继续

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort) — 模拟i=9循环

i=9 第2次j层循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	2	3	1	4	6	9	7	5	8	9

j = 1

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素 2;

j 循环就是为了将第1位上的3 赋值给第5位上; 将L.r[j+increment] = L.r[j] , L.r[5] = L.r[1] = 3;
j = j - increment = 1-4 = -3 ; j 退出循环时 j = -3;

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort) — 模拟i=9循环

退出 j 层循环

increment = 4

下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	4	6	3	7	5	8	9

j = -3

j = j + increment

L.r[0] 就是用来临时存储当前待插入的元素

需要把L.r[0] 上的2 赋值到原来第1位上的空间. 所以L.r[j+increment] = L.r[0] = L[-3+4] = L[1] = 2

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort)

希尔排序第一次do...while 结果

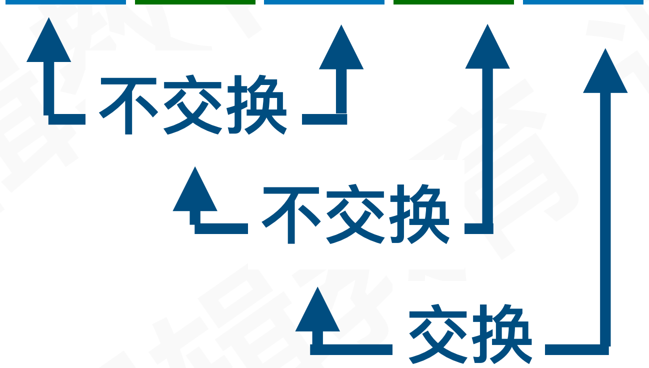
下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	4	6	3	7	5	8	9

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort)

increment = 2

下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	4	6	3	7	5	8	9



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

希尔排序实现(Shell Sort)

increment = 2

下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	3	6	4	7	5	8	9

不交换
不交换
不交换
不交换

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort)

希尔排序第二次do...while 结果

下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	3	6	4	7	5	8	9

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort)

increment = 1

下标	0	1	2	3	4	5	6	7	8	9
数据	2	2	1	3	6	4	7	5	8	9

交换

increment = 1

下标	0	1	2	3	4	5	6	7	8	9
数据	2	1	2	3	6	4	7	5	8	9

交换

increment = 1

下标	0	1	2	3	4	5	6	7	8	9
数据	2	1	2	3	4	6	7	5	8	9

交换

课程研发:CC老师
课程授课:CC老师

希尔排序实现(Shell Sort)

increment = 1

下标	0	1	2	3	4	5	6	7	8	9
数据	2	1	2	3	4	6	5	7	8	9

交换

最终排序结果:

下标	0	1	2	3	4	5	6	7	8	9
数据	2	1	2	3	4	5	6	7	8	9

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

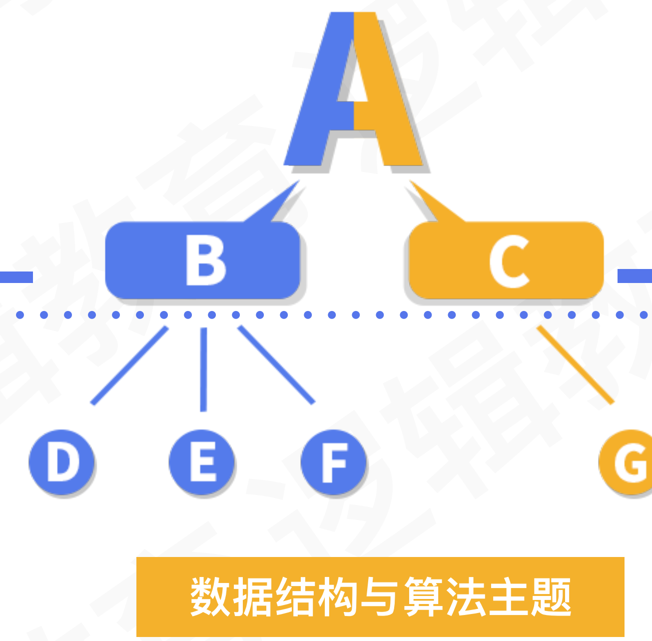
希尔排序(Shell Sort) 复杂度分析

步长序列	最坏情况下时间复杂度
$n / 2^i$	$O(n^2)$
$2^k - 1$	$O(n^{3/2})$
$2^i 3^i$	$O(n\log^2 n)$

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

Hello 数据结构与算法



堆排序 (Heap Sort)

@CC老师

全力以赴．非同凡"想"

课程研发:CC老师
课程授课:CC老师

了解"堆"结构

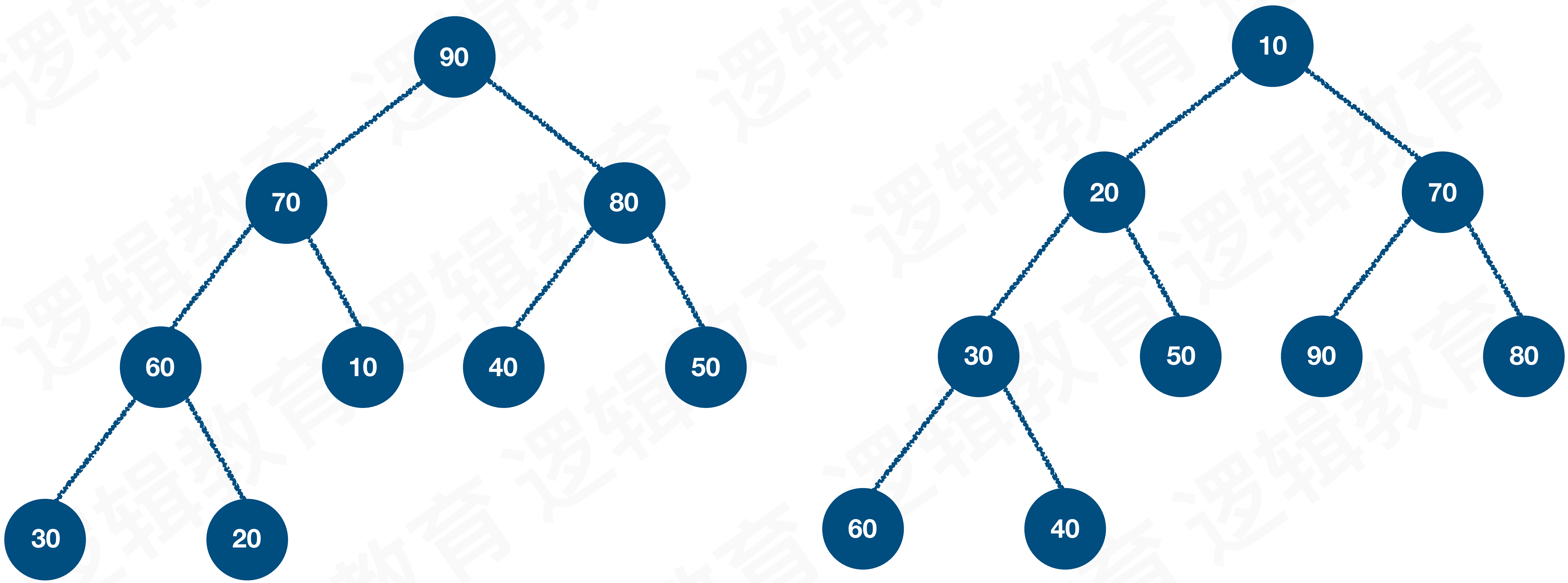


图1　　图2

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

了解"堆"结构

堆是具有下面性质的完全二叉树：每个结点的值都大于或等于其左右孩子结点的值，称为大顶堆；如图1；或者每个结点的值都小于等于其左右孩子的结点的值，称为小顶堆，如图2.

如果按照层序遍历的方式给结点从1开始编号,则结点之间的满足如下关系

$$\begin{cases} K_i \geq K_{2i} \\ K_i \geq K_{2i+1} \end{cases} \quad \text{或} \quad \begin{cases} K_i \leq K_{2i} \\ K_i \leq K_{2i+1} \end{cases} \quad 1 \leq i \leq \frac{n}{2}$$

课程研发:CC老师
课程授课:CC老师

了解"堆"结构

$$\begin{cases} K_i \geq K_{2i} \\ K_i \geq K_{2i+1} \end{cases} \text{或} \begin{cases} K_i \leq K_{2i} \\ K_i \leq K_{2i+1} \end{cases} \quad 1 \leq i \leq \frac{n}{2}$$

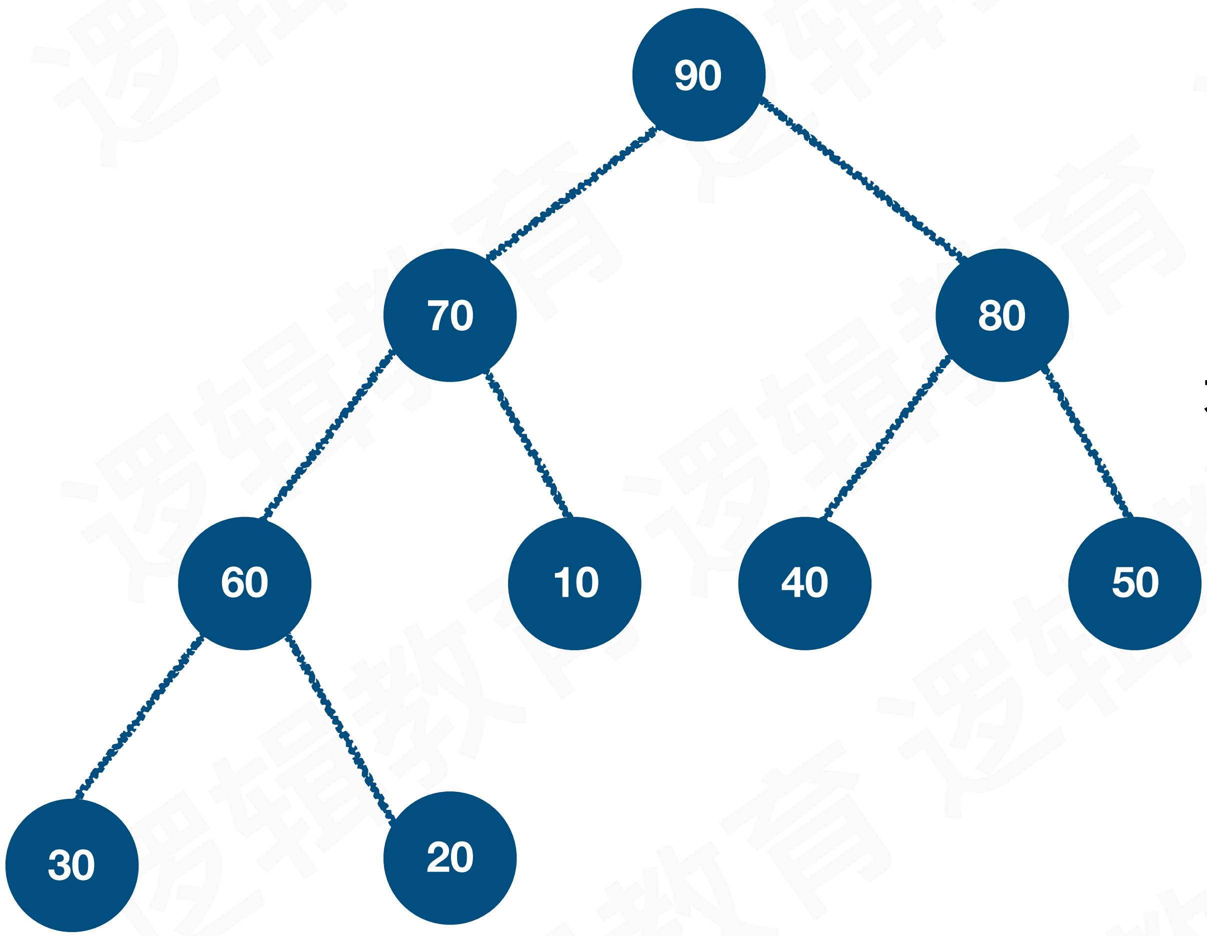


图1

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		90	70	80	60	10	40	50	30	20

i = 4

$K_i \geq K_{2i}$

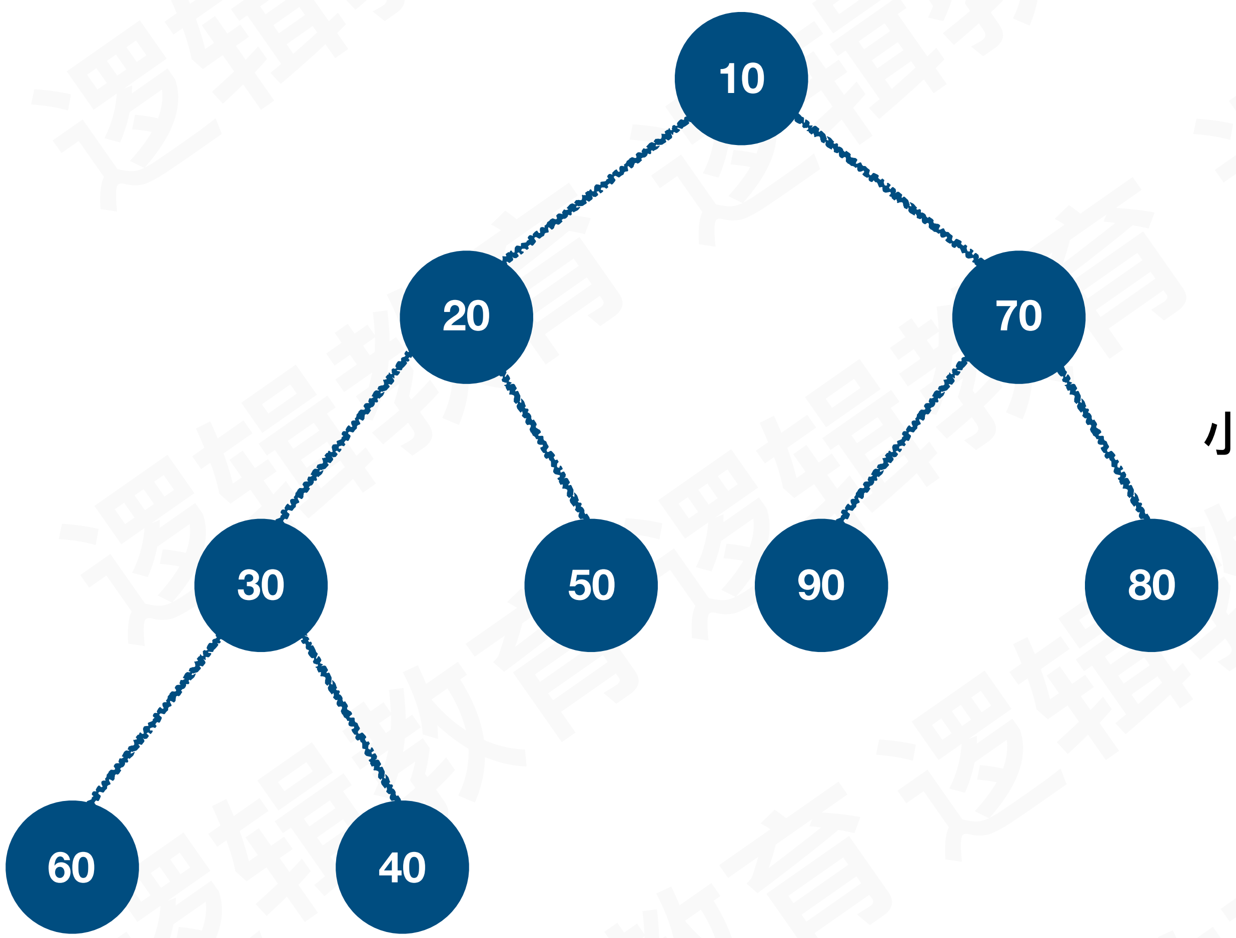
$K_i \geq K_{2i+1}$

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

了解”堆”结构

$$\begin{cases} K_i \geq K_{2i} \\ K_i \geq K_{2i+1} \end{cases} \quad 或 \quad \begin{cases} K_i \leq K_{2i} \\ K_i \leq K_{2i+1} \end{cases} \qquad 1 \leq i \leq \frac{n}{2}$$



下标	0	1	2	3	4	5	6	7	8	9
小顶堆		10	20	70	30	50	90	80	60	40

$i = 4$

$K_i \leq K_{2i}$

$K_i \leq K_{2i+1}$

图2

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

堆排序(Heap Sort) 就是利用堆(假设我们选择大顶堆)进行排序的算法.它的基本思想:

①将待排序的序列构成一个大顶堆,此时,整个序列的最大值就堆顶的根结点,将它移走(其实就是将其与堆数组的末尾元素交换, 此时末尾元素就是最大值);

②然后将剩余的n-1个序列重新构成一个队,这样就会得到n个元素的次大值,如此重复执行,就能得到一个有序序列了

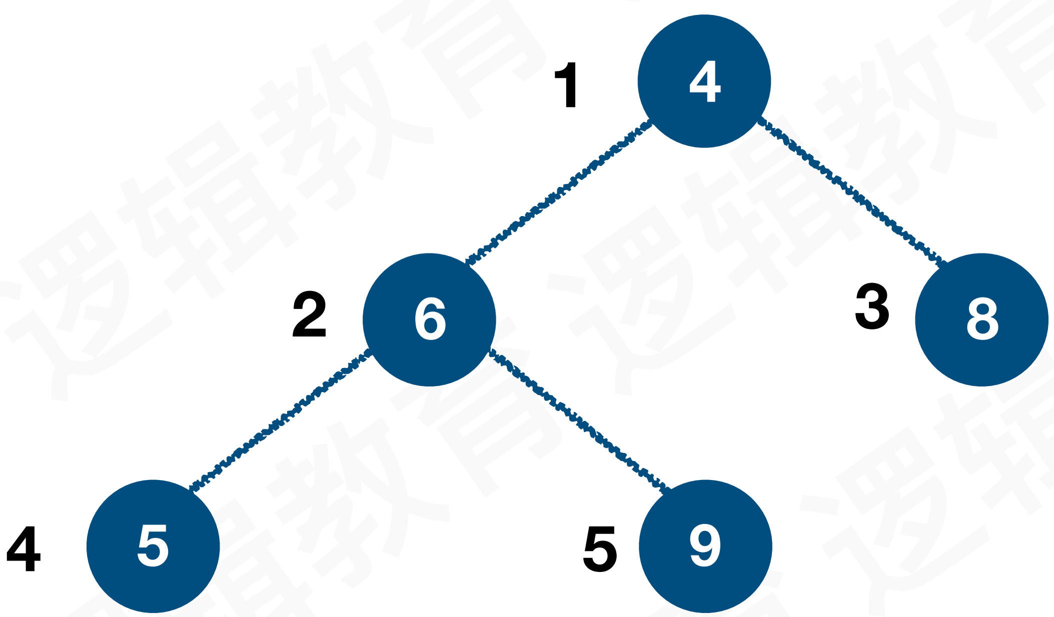
课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

步骤① 构造初始堆。将给定无序序列构造成一个大顶堆（一般升序采用大顶堆，降序采用小顶堆)。

A. 给定无序序列结构如下



下标	0	1	2	3	4	5
大顶堆		4	6	8	5	9

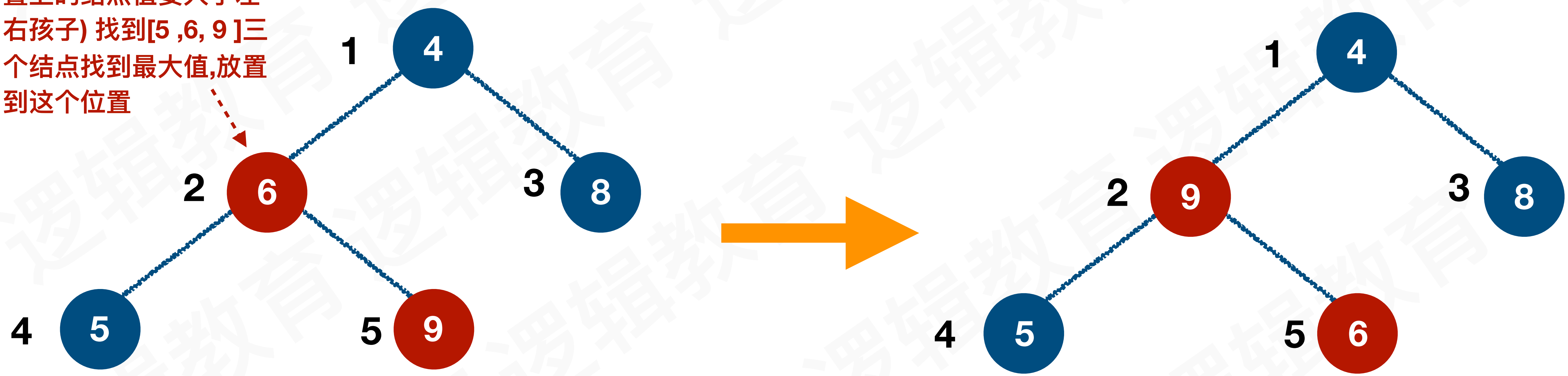
课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

B. 从最后一个非叶子结点开始（叶结点自然不用调整，第一个非叶子结点 2，也就是下面的6结点) 从左往右,从下往上进行调整.

堆结构:大顶堆.(这个位置上的结点值要大于左右孩子) 找到[5 ,6, 9]三个结点找到最大值,放置到这个位置



下标	0	1	2	3	4	5
大顶堆		4	6	8	5	9

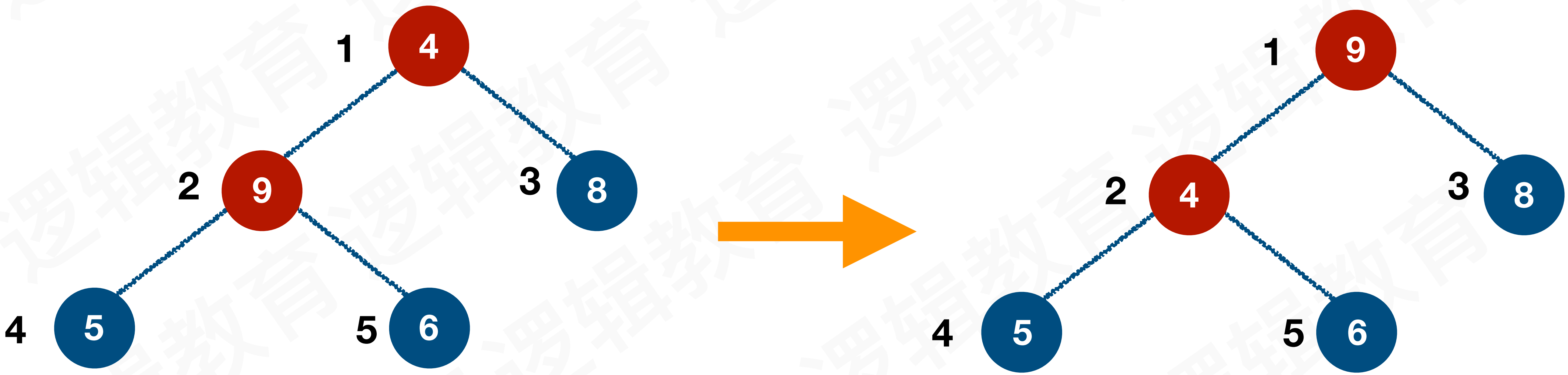
下标	0	1	2	3	4	5
大顶堆		4	9	8	5	6

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

B. 找到第二个非叶子结点 4 . 从[4, 9 , 8]中找到最大的, 4与9进行交换

堆结构:大顶堆.(这个位置上的结点值要大于左右孩子) 找到[9 ,4, 8]三个结点找到最大值,放置到这个位置



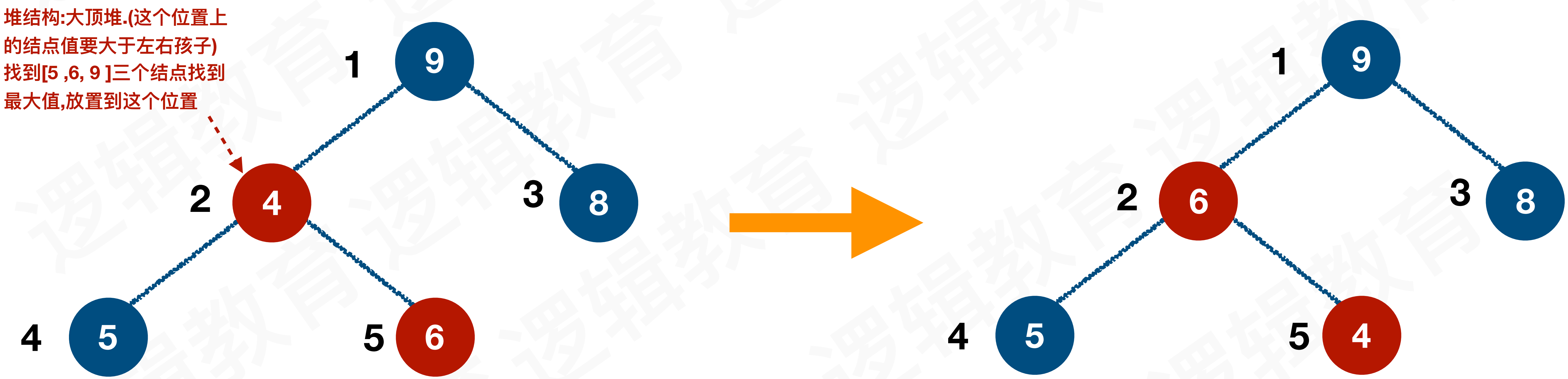
下标	0	1	2	3	4	5
大顶堆		4	9	8	5	6

下标	0	1	2	3	4	5
大顶堆		9	4	8	5	6

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

C. 此时的交换导致了子根结点[4 ,5 ,6] 结构混乱,继续调整. 从[4,5,6]中找到最大的结点6. 交换 4 与 6;
那么经过3次调整. 你会发现 我们将刚刚无序序列 调整成一个大顶堆结构;



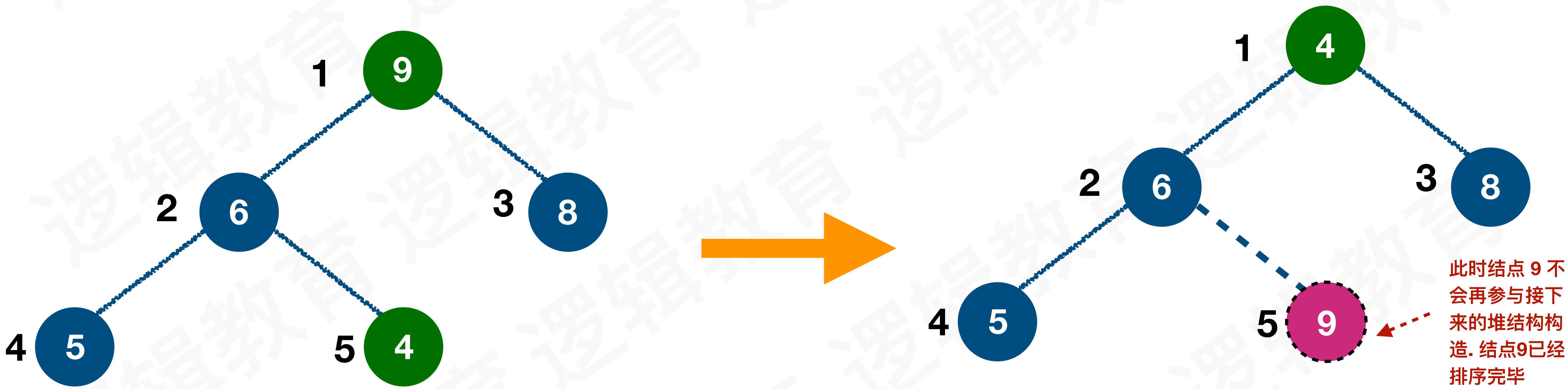
下标	0	1	2	3	4	5
大顶堆		9	4	8	5	6

下标	0	1	2	3	4	5
大顶堆		9	6	8	5	4

堆排序(Heap Sort) 原理探索

步骤② 将堆顶元素与末尾元素进行交换，使末尾元素最大。然后继续调整堆，再将堆顶元素与末尾元素交换，得到第二大元素。如此反复进行交换、重建、交换

A. 将堆顶元素9和末尾元素4进行交换. 此时9将不参与后续的堆排序



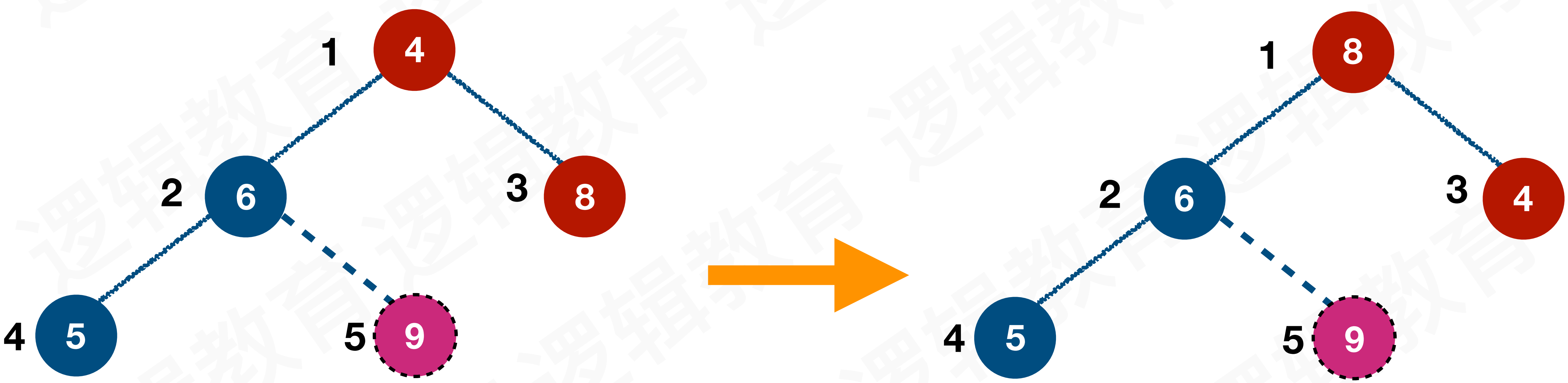
下标	0	1	2	3	4	5
大顶堆		9	6	8	5	4

下标	0	1	2	3	4	5
大顶堆		4	6	8	5	9

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

B. 重新调整结构，使其继续满足堆定义 从[4, 6 , 8]中找到最大的, 4与8进行交换. 经过调整此时我们又得到了一个 大顶堆



下标	0	1	2	3	4	5
大顶堆		4	6	8	5	9

下标	0	1	2	3	4	5
大顶堆		8	6	4	5	9

逻辑教育
Logic education

堆排序(Heap Sort) 原理探索

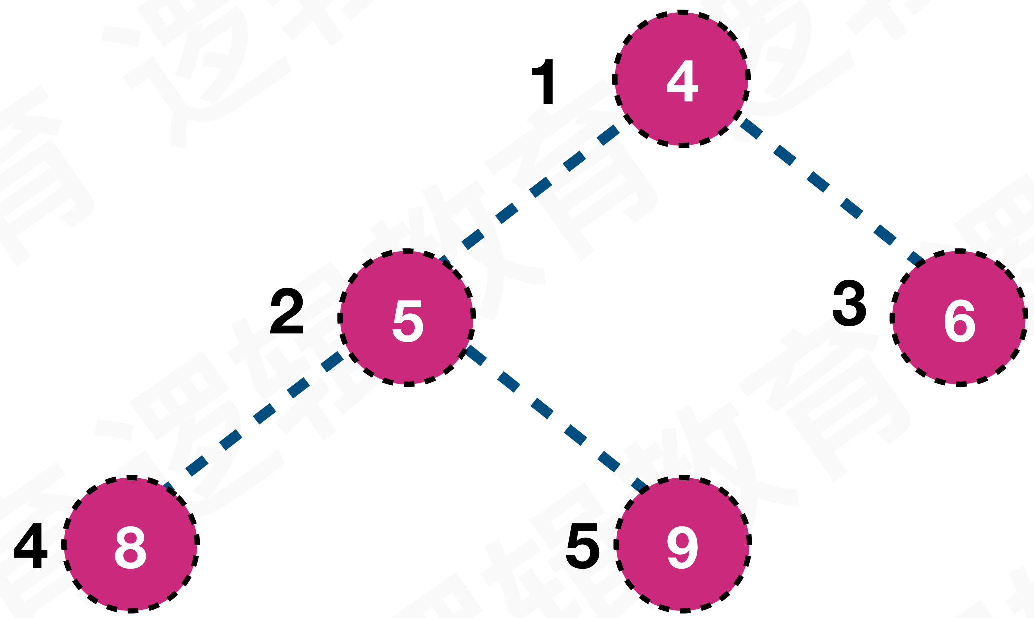
C. 再将堆顶元素8与末尾元素5进行交换，得到第二大元素8.

下标	0	1	2	3	4	5
大顶堆		8	6	4	5	9

下标	0	1	2	3	4	5
大顶堆		5	6	4	8	9

堆排序(Heap Sort) 原理探索

后续过程，继续进行调整，交换，如此反复进行，最终使得整个序列有序



下标	0	1	2	3	4	5
大顶堆		4	5	6	8	9

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort)思路:

堆排序思路:

- 将无需序列构建成一个堆，根据升序降序需求选择大顶堆或小顶堆
- 将堆顶元素与末尾元素交换，将最大元素"沉"到数组末端;
- 重新调整结构，使其满足堆定义，然后继续交换堆顶元素与当前末尾元素，反复执行调整+交换步骤，直到整个序列有序;

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort)代码实现:

- 如何由一个无序序列构建成一个堆?
- 如何在输出堆顶元素后,调整剩余元素成为一个新的堆?

课程研发:CC老师
课程授课:CC老师

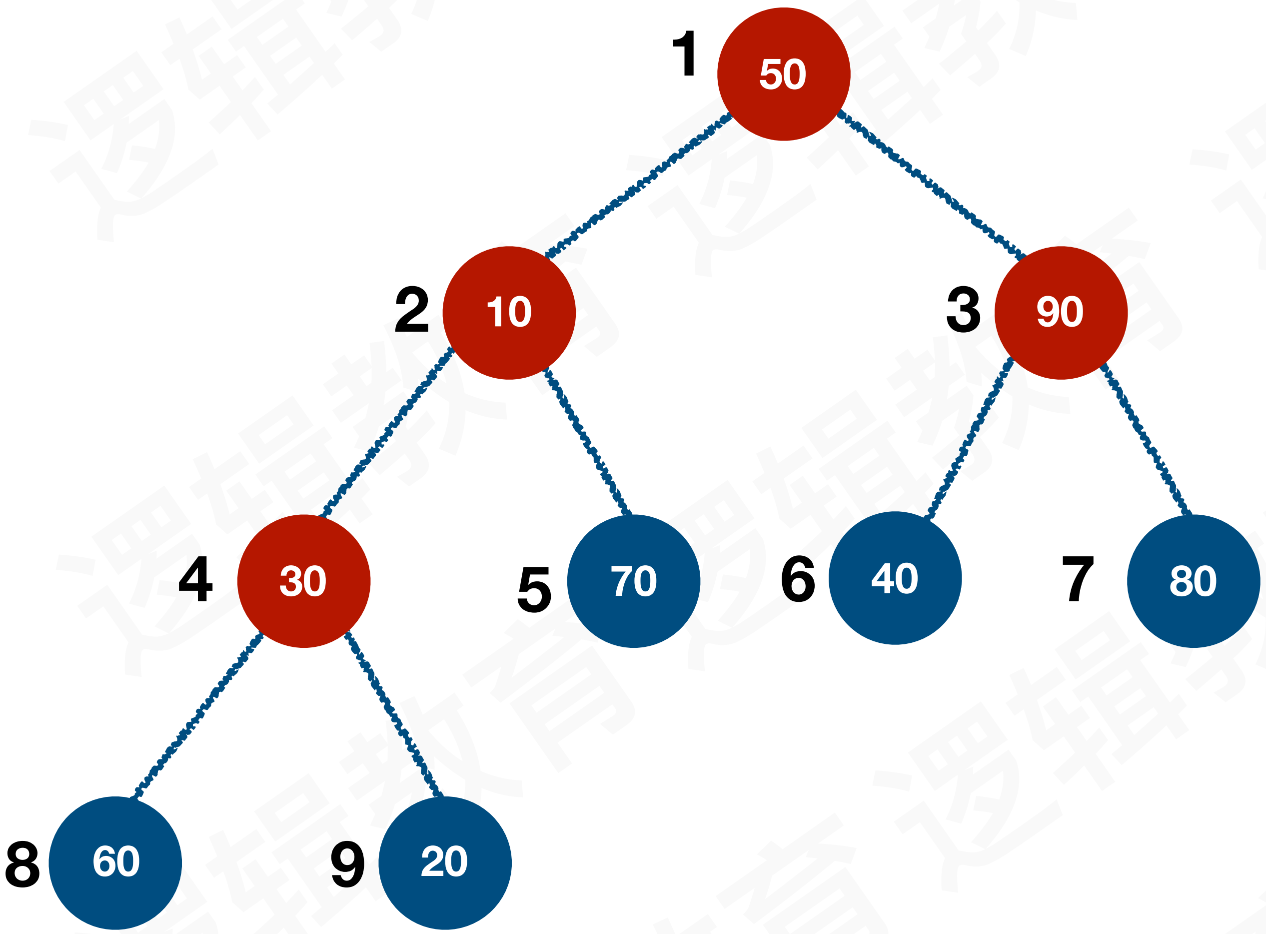
堆排序(Heap Sort)代码实现:

堆排序思路:

- 将无需序列构建成一个堆，根据升序降序需求选择大顶堆或小顶堆
- 将堆顶元素与末尾元素交换，将最大元素”沉"到数组末端;
- 重新调整结构，使其满足堆定义，然后继续交换堆顶元素与当前末尾元素，反复执行调整+交换步骤，直到整个序列有序;

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort)代码实现:



堆是具有下面性质的**完全二叉树**：每个结点的值都大于或等于其左右孩子结点的值，称为大顶堆；

i 从4->3->2->1

课程研发:CC老师
课程授课:CC老师

复习-二叉树性质5

二叉树性质5: 如果对一颗有n个结点的完全二叉树的结点按层序编号,对任一结点i (1 ≤ i ≤ n) 有:

- 如果 i = 1 ,则结点 i 是二叉树的根. 无双亲. 如果 i > 1,则其双亲是结点 [i / 2];
- 如果 2i > n ,则结点 i 无左孩子 (结点i 为叶子结点); 否则左孩子是结点 2i;
- 如果 2i + 1 > n ,则结点 i 无右孩子; 否则其右孩子是结点 2i+1;

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

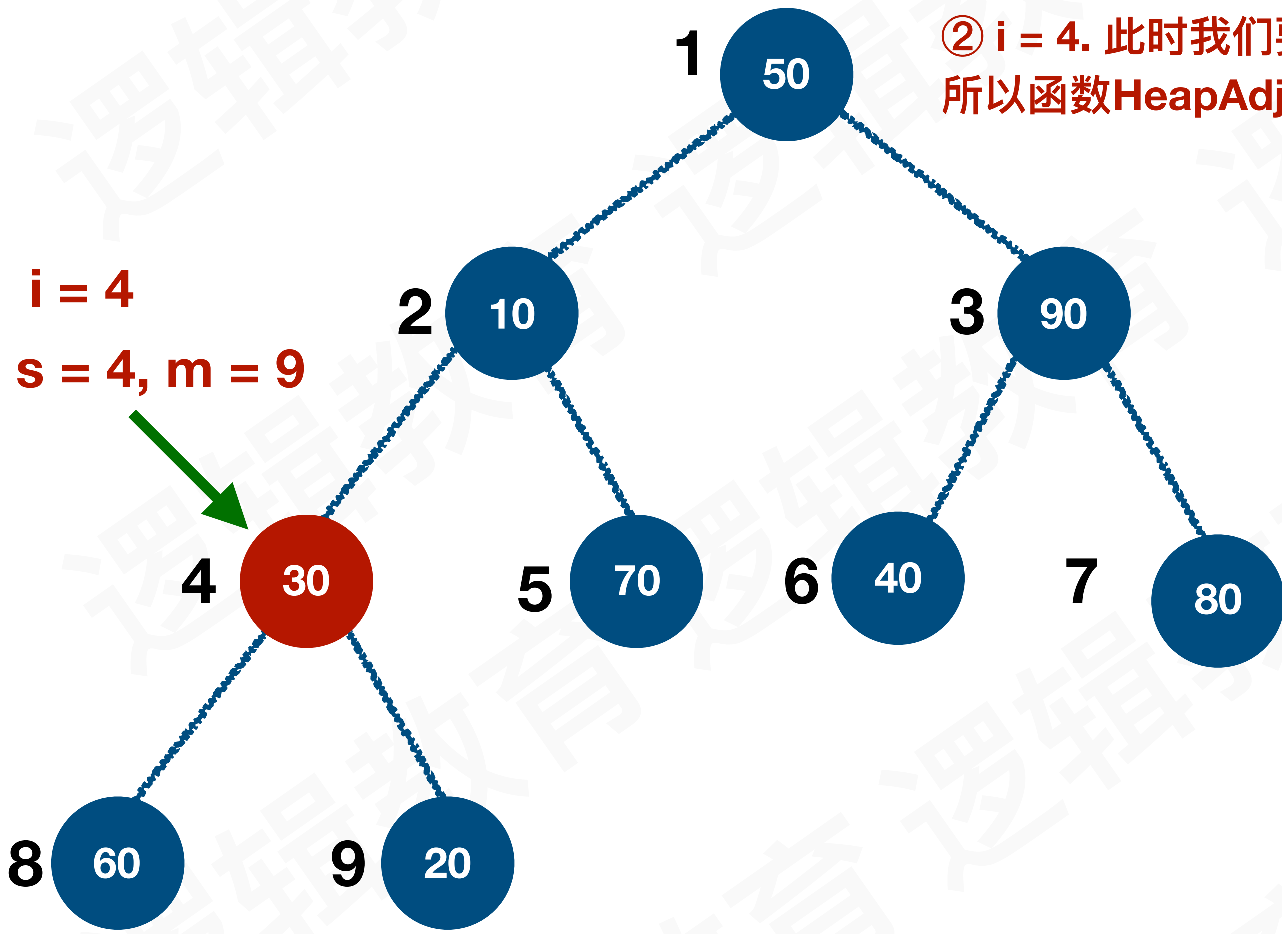
堆排序(Heap Sort) — 大顶堆调整函数实现分析

① i 从4->3->2->1

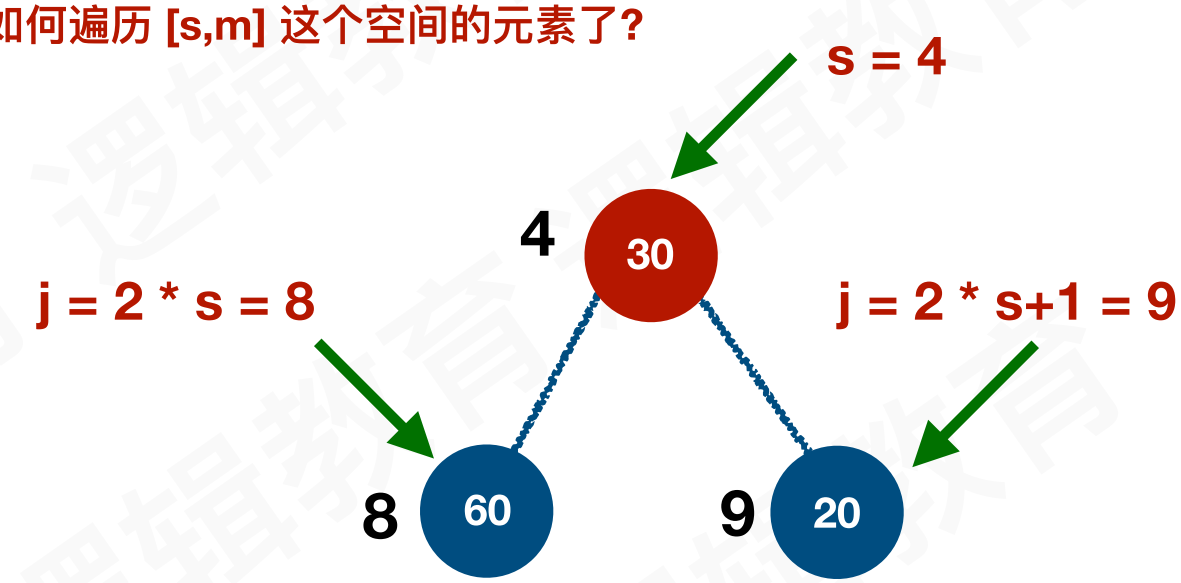
② i = 4. 此时我们要调整在 [4,9]这个区间大顶堆调整
所以函数HeapAdjust (L,4,9)

③ 将L.r[4] 存储到临时变量temp 中

i = 4
s = 4, m = 9



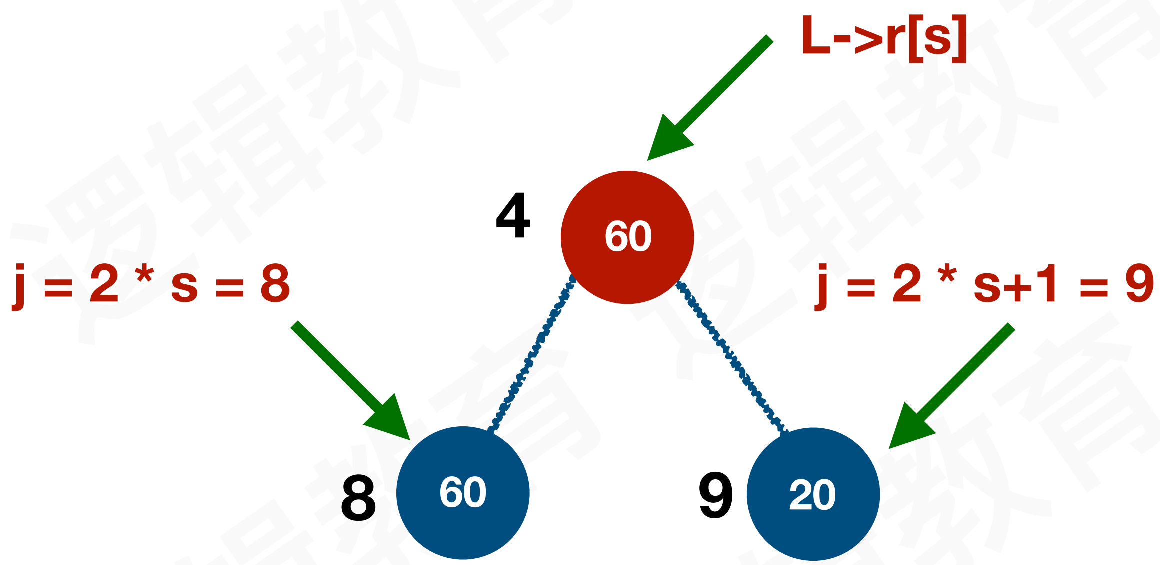
④ 如何遍历 [s,m] 这个空间的元素了?



下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	10	90	30	70	40	80	60	20

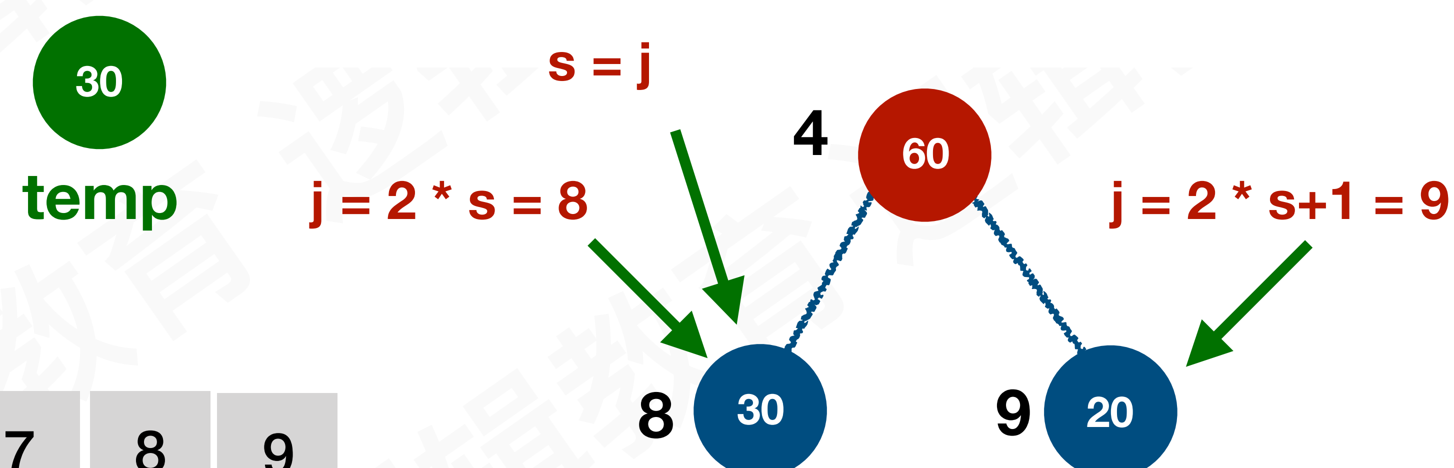
⑤ 首先判断 j < m ,判断 j 是否是最后一个结点;
因为如果是最后一个结点, 就没有必要比较左右孩子了;
⑥ 如果左孩子 小于 右孩子的话 , j++. 使得当前的L.r[s] 与较大孩子比较; 否则 j 不自增1.
⑦ 使得temp 与目前左右孩子中最大还是比较; 如果temp>= L.r[j] 则退出循环;

堆排序(Heap Sort) — 大顶堆调整函数实现分析



⑧ 调整 [4,8,9] 的位置;
⑨ 将L->r[s] = L ->r[j];　在这个例子里 因为左孩子大于右孩子 ,所以当前的 j = 8;　将60 赋值到L->r[4],则此时L->r[4] = 60, L->r[8] = 60;
⑩ 更新s, s = j;　此时 s = 8;

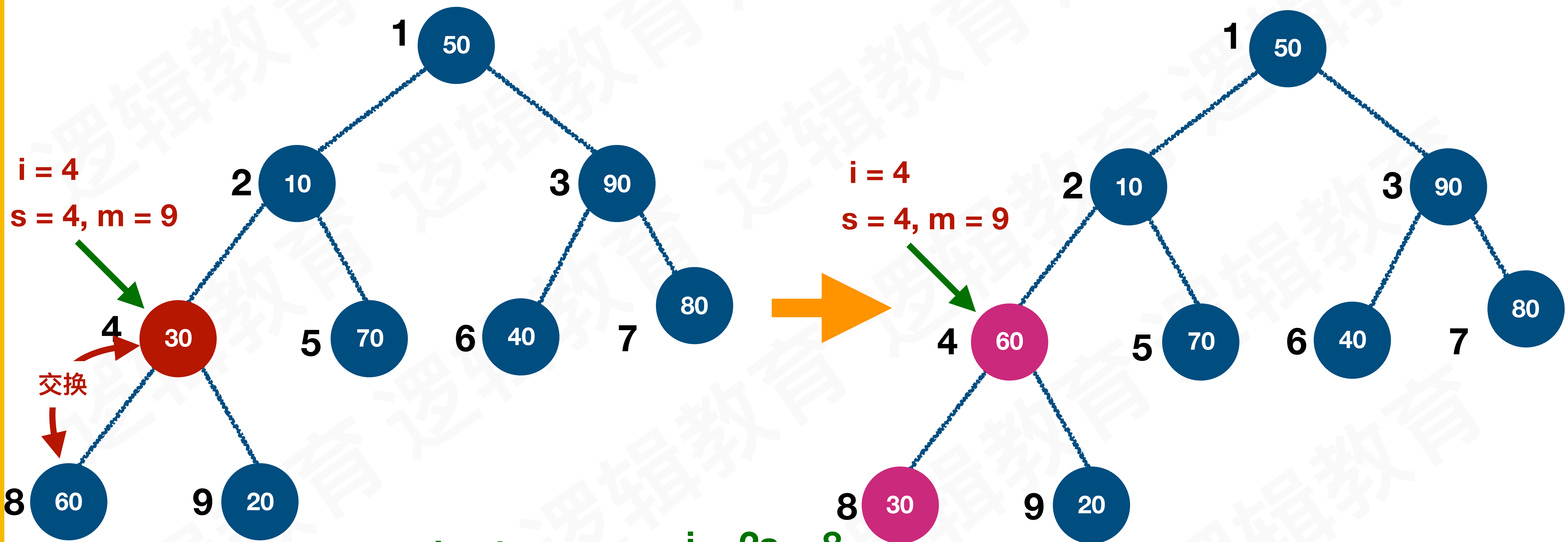
11. 再次循环 j = 2*j = 16, m = 9 ; j < m . 因此跳出j层循环;
12. temp = 30 . 将它赋值给 L.r[s] = L.r[8] = 30; 就完成了30 与 60的交换工作. 那么本次的调整就完成了.



下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	10	90	30	70	40	80	60	20

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 大顶堆调整函数调用分析 i = 4 时



i = 4

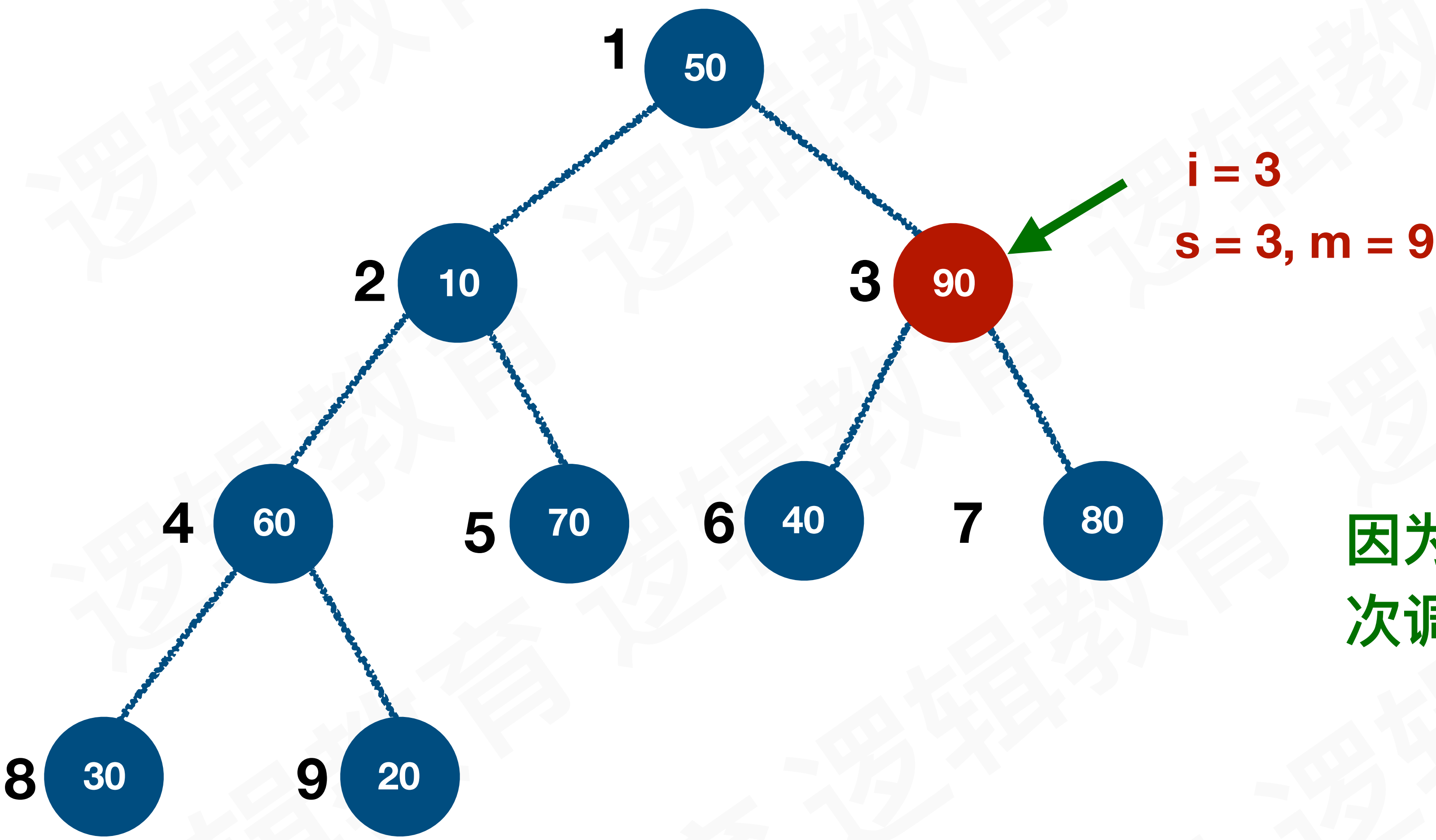
j = 2s = 8

j = 2s+1 = 9

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	10	90	30	70	40	80	60	20
					60				30	

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 大顶堆调整函数调用分析 i = 3 时



因为L.r[s] > L.r[2s] > L.r[2s+1] 所以这次调整序列没有发生变化

i = 4 s = 4

j = 2s = 6

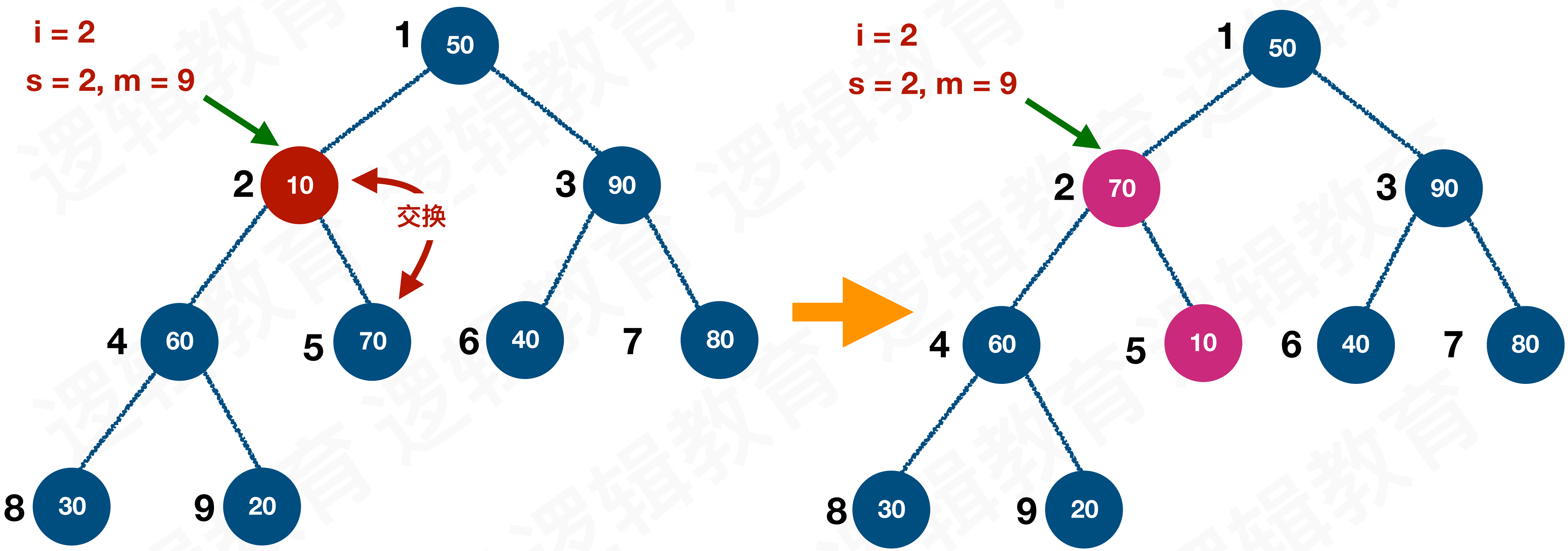
j = 2s+1 = 7

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	10	90	60	70	40	80	30	20

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 大顶堆调整函数调用分析 i = 2 时



i = 4 s = 4　　j = 2s = 4　　j = 2s+1 =5

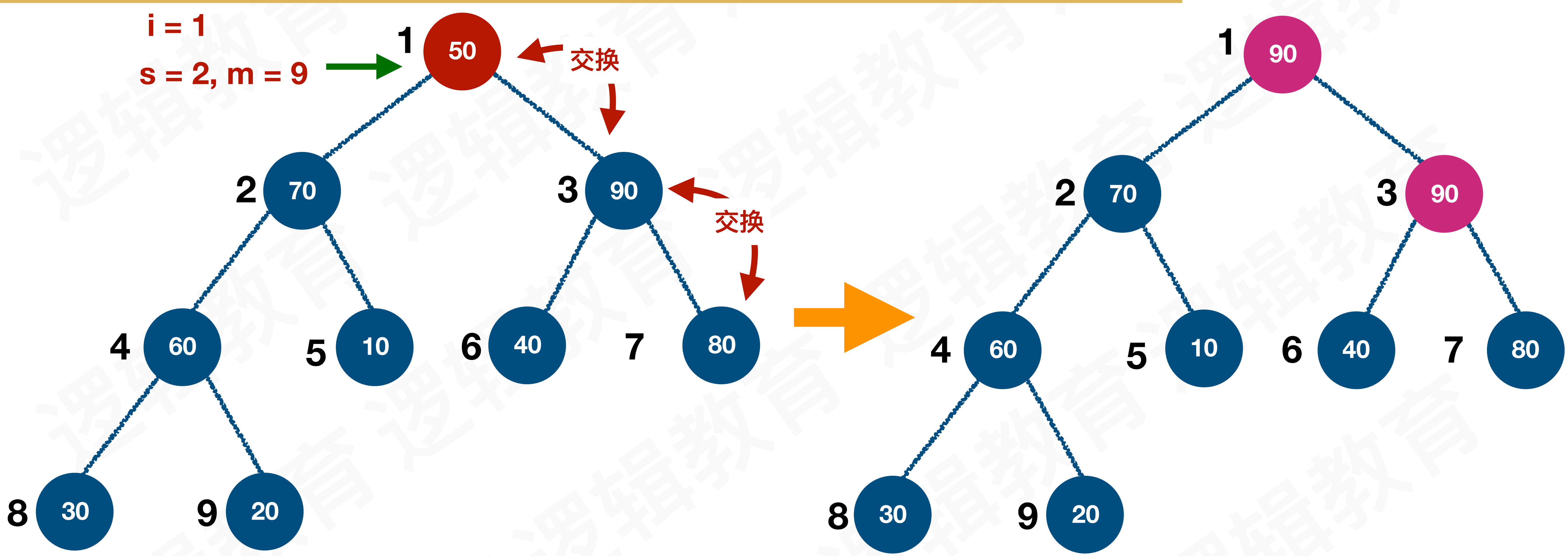
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	10	90	60	70	40	80	30	20
			70			10				

10
temp

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 大顶堆调整函数调用分析 i = 1时



i = 1 s = 1 j = 2s = 2 j = 2s+1 = 3

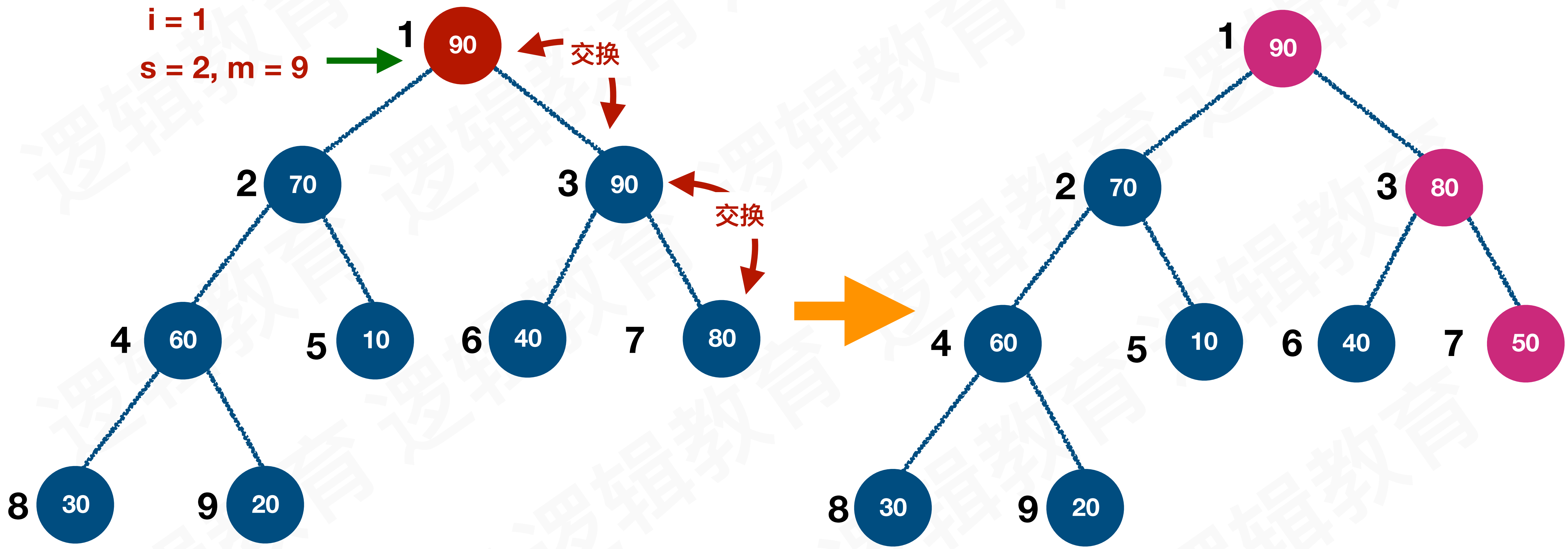
第1层j循环

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	70	90	60	10	40	80	30	20
		90		90						

50
temp

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 大顶堆调整函数调用分析 i = 1时



i = 1 s = 3

j = 2s = 6 j = 2s+1 = 7

第1层j循环

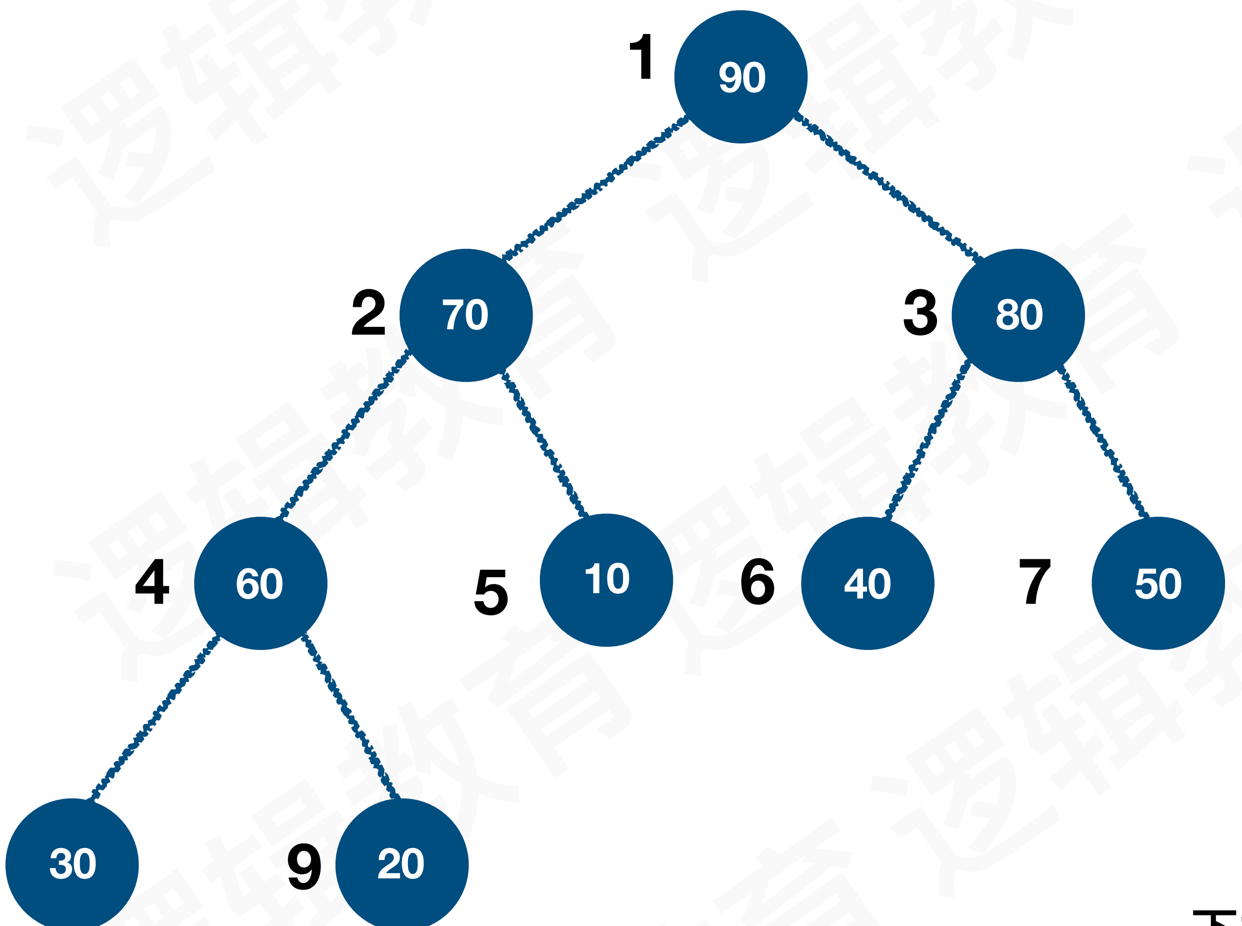
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		90	70	90	60	10	40	80	30	20
				80				50		

50

temp

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 大顶堆调整函数HeapAdjust 执行结果



大顶堆构建完成

堆是具有下面性质的**完全二叉树**：**每个结点的值都大于或等于其左右孩子结点的值**，称为大顶堆.

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		90	70	80	60	10	40	50	30	20

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析

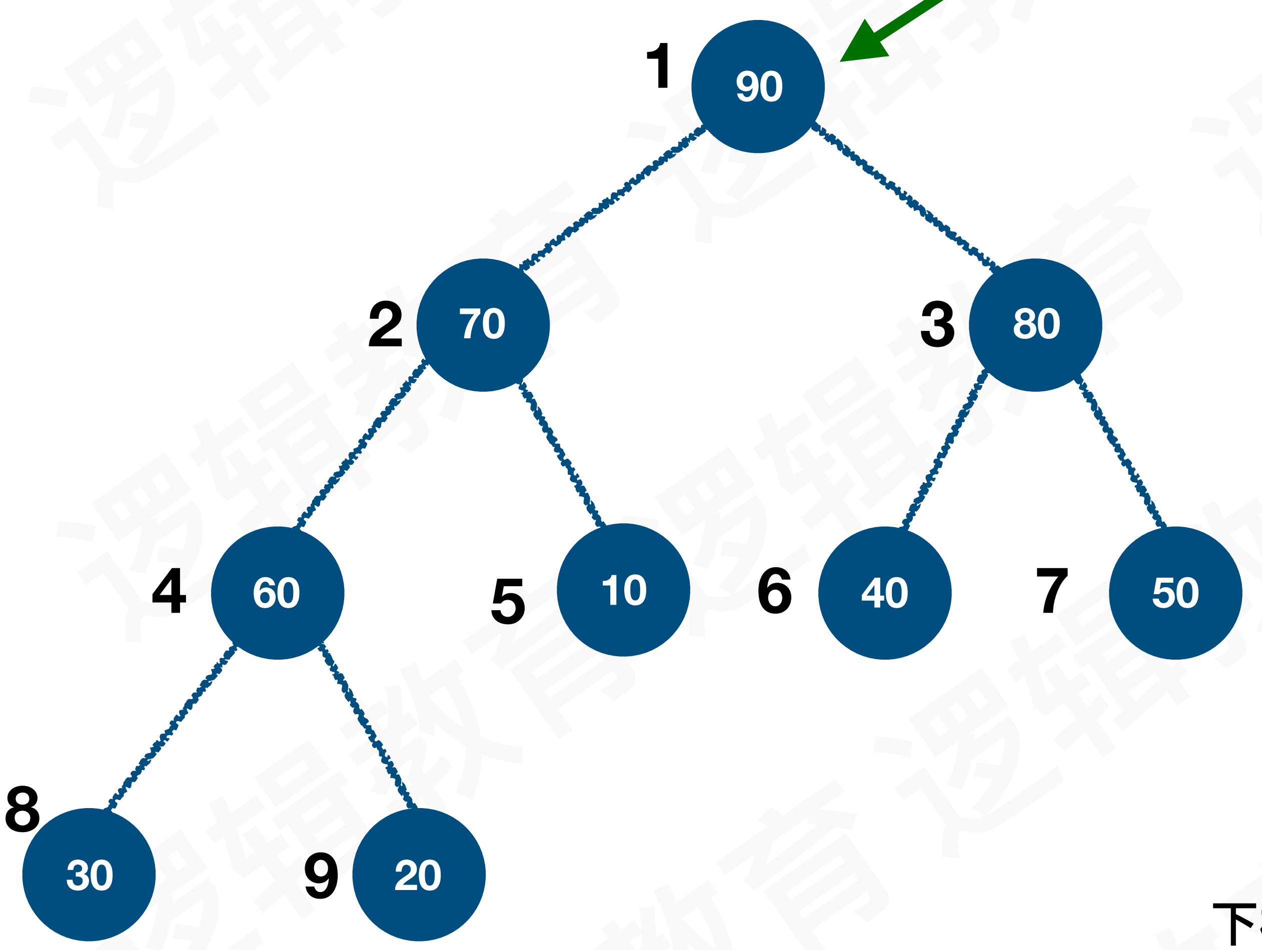
堆排序思路:

- 将无需序列构建成一个堆，根据升序降序需求选择大顶堆或小顶堆
- 将堆顶元素与末尾元素交换，将最大元素"沉"到数组末端;
- 重新调整结构，使其满足堆定义，然后继续交换堆顶元素与当前末尾元素，反复执行调整+交换步骤，直到整个序列有序;

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析

堆顶元素 永远是L.r[1]



大顶堆构建完成

① 交换堆顶元素 L.r[1] 与 序列末尾元素 i
② 循环从 i = 9 到 1
③ 把排序完毕的末尾记录 除外之后,继续进行大顶堆调整 HeapAdjust(L,1,i-1)

大顶堆(未排序序列)
范围是[9,1]

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		90	70	80	60	10	40	50	30	20

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析

堆排序思路:

- 将无需序列构建成一个堆，根据升序降序需求选择大顶堆或小顶堆
- 将堆顶元素与末尾元素交换，将最大元素"沉"到数组末端;
- 重新调整结构，使其满足堆定义，然后继续交换堆顶元素与当前末尾元素，反复执行调整+交换步骤，直到整个序列有序;

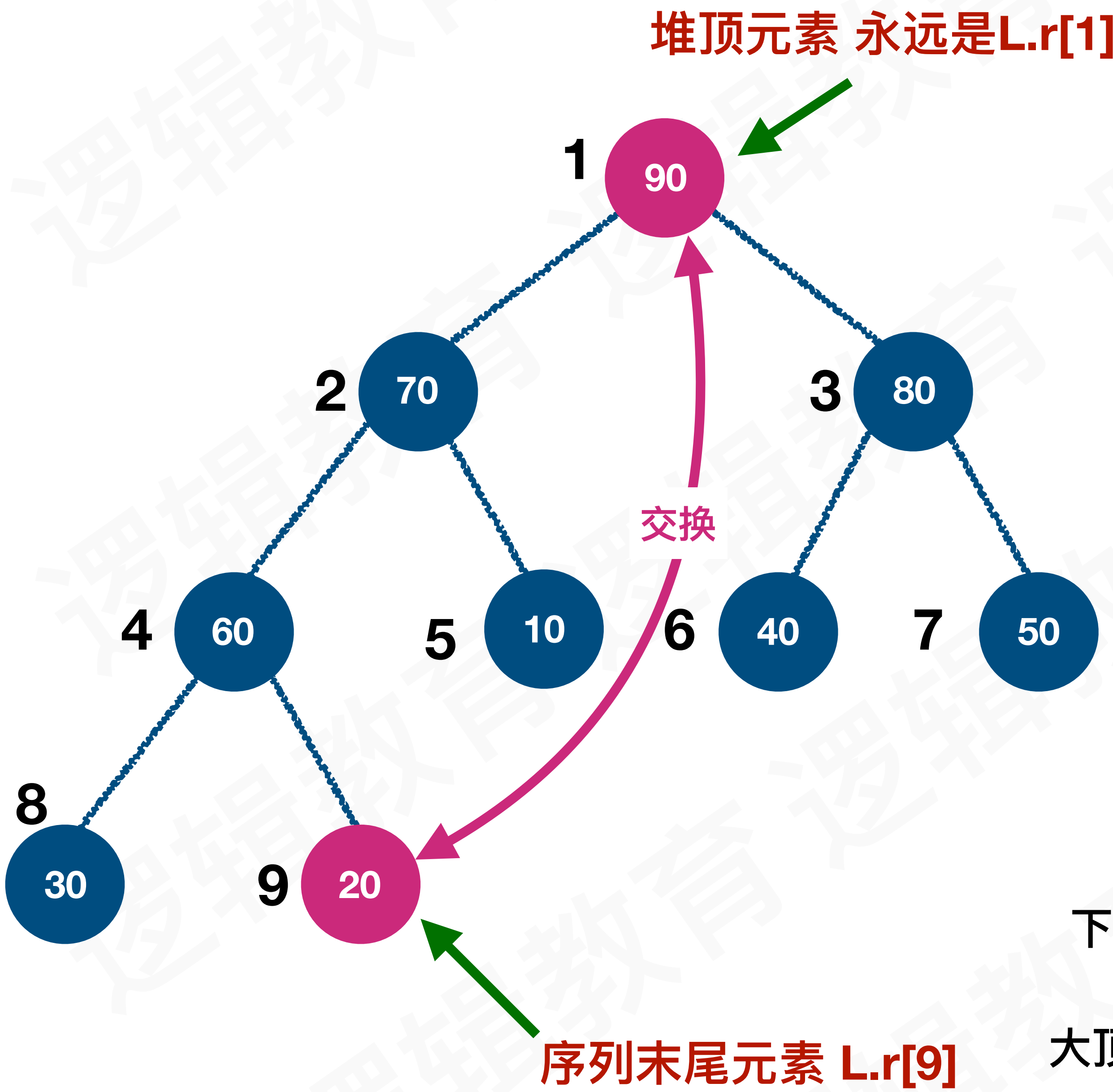
```
        //2.逐步将每个最大的值根结点与末尾元素进行交换,并且再调整成大顶堆
        for(i = L->length; i > 1; i--){

            //① 将堆顶记录与当前未经排序子序列的最后一个记录进行交换;
            swap(L, 1, i);
            //② 将L->r[1...i-1]重新调整成大顶堆;
            HeapAjust(L, 1, i-1);
        }
}
```

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 9

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]

末尾元素

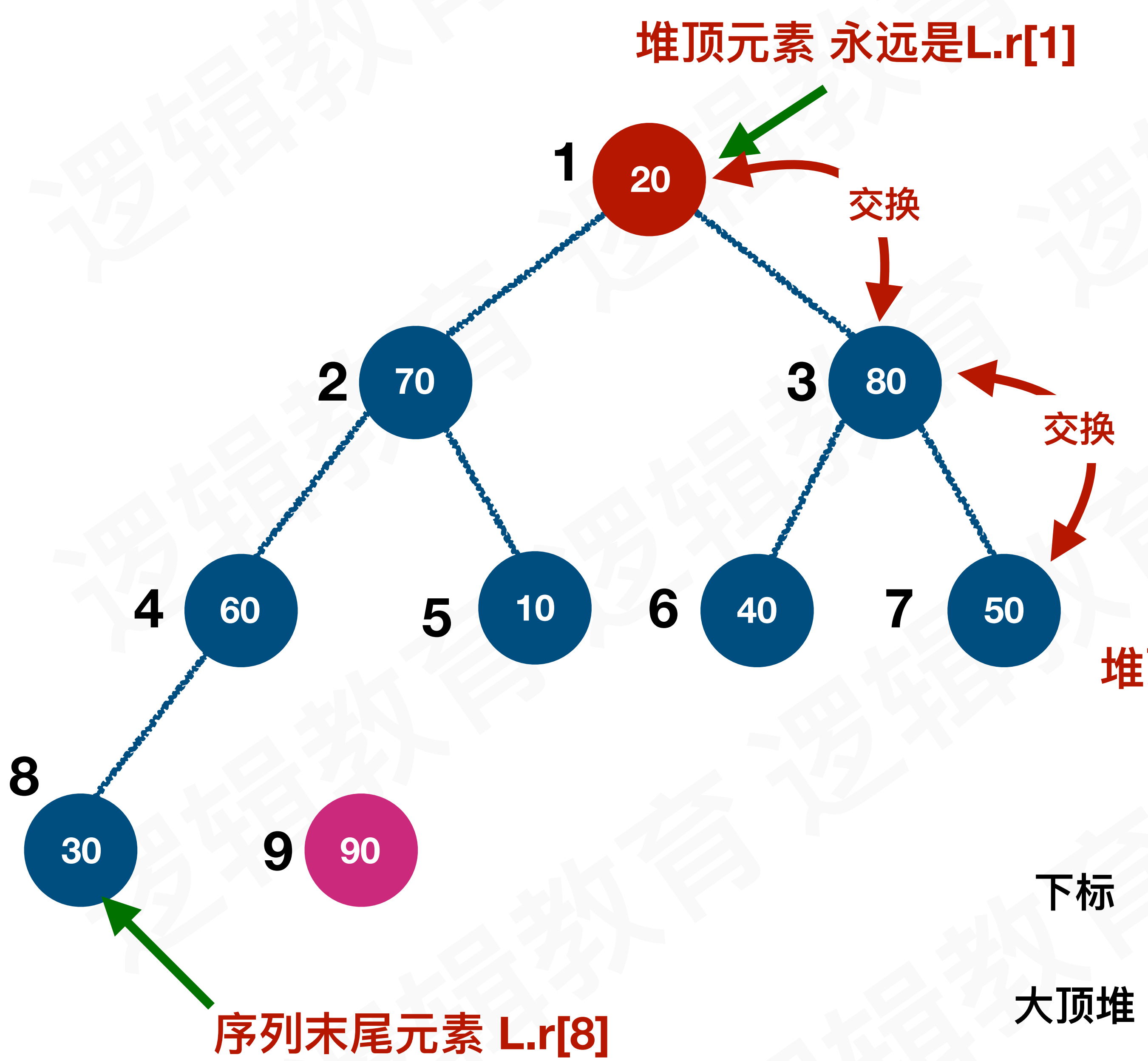
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		90	70	80	60	10	40	50	30	20
		20								90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 9

重新调整L.r[1...i-1] 为大顶堆



堆顶元素 永远是L.r[1]

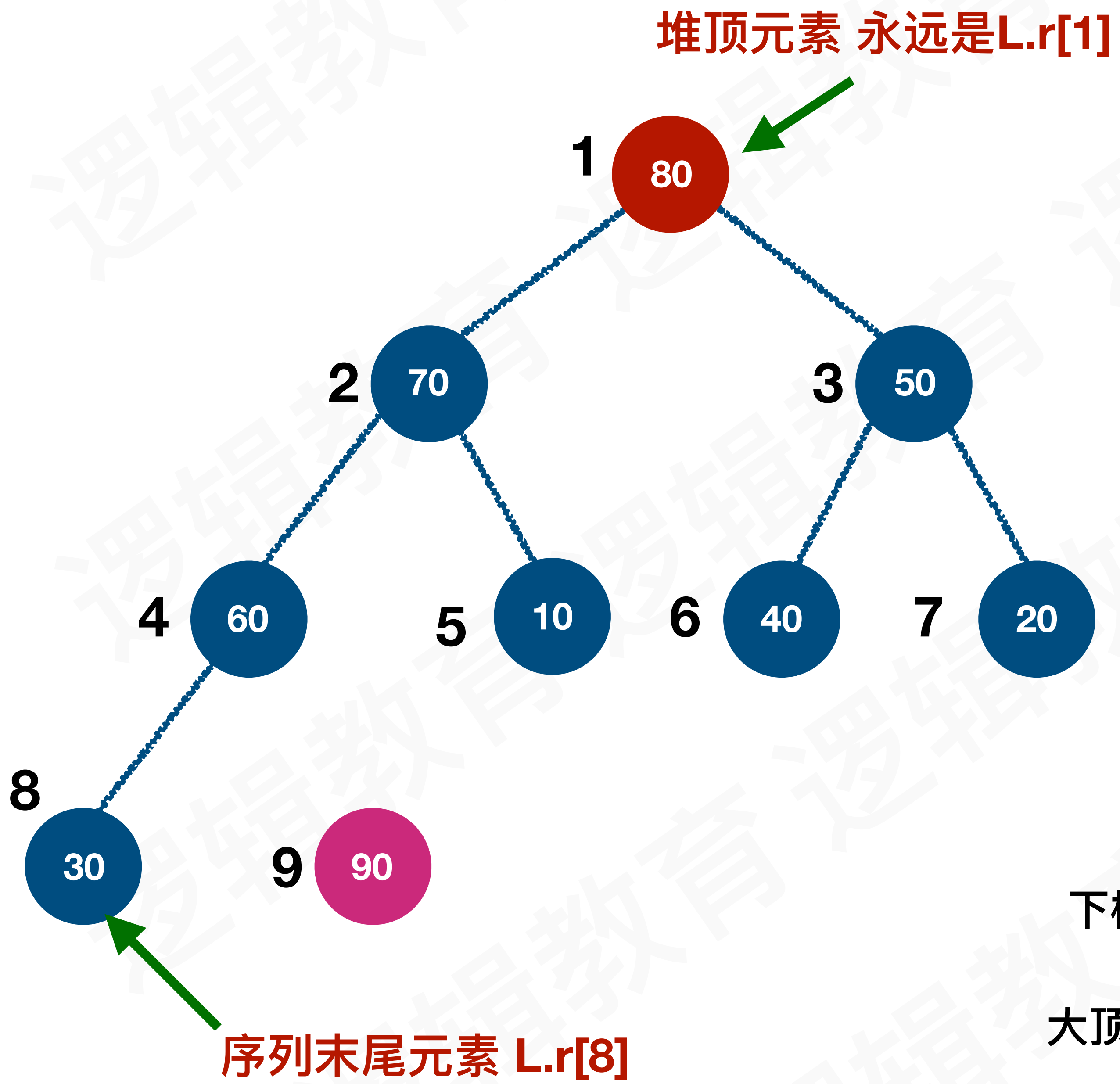
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		20	70	80	60	10	40	50	30	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 9

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

末尾元素

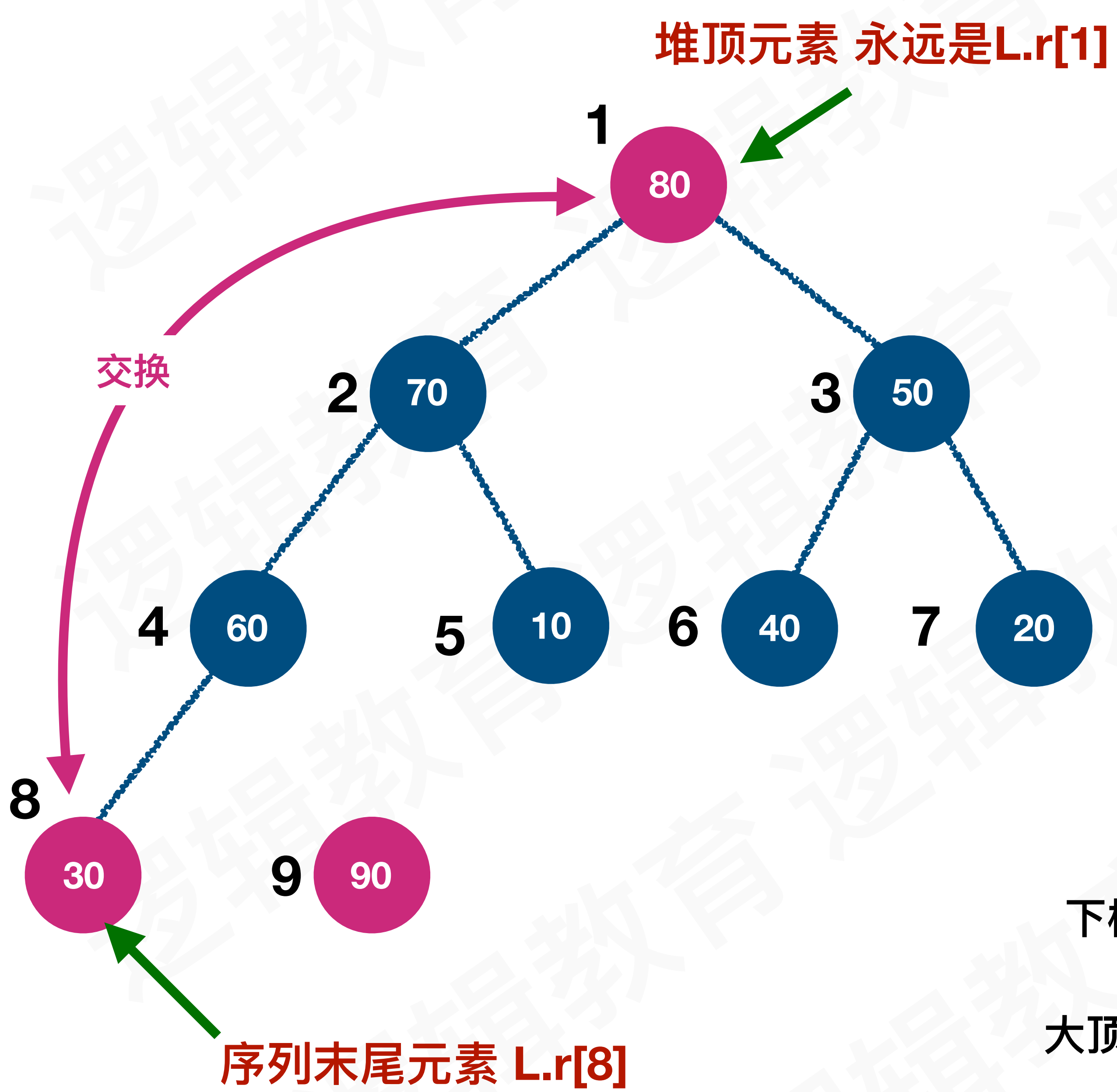
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		80	70	50	60	10	40	20	30	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 8

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]

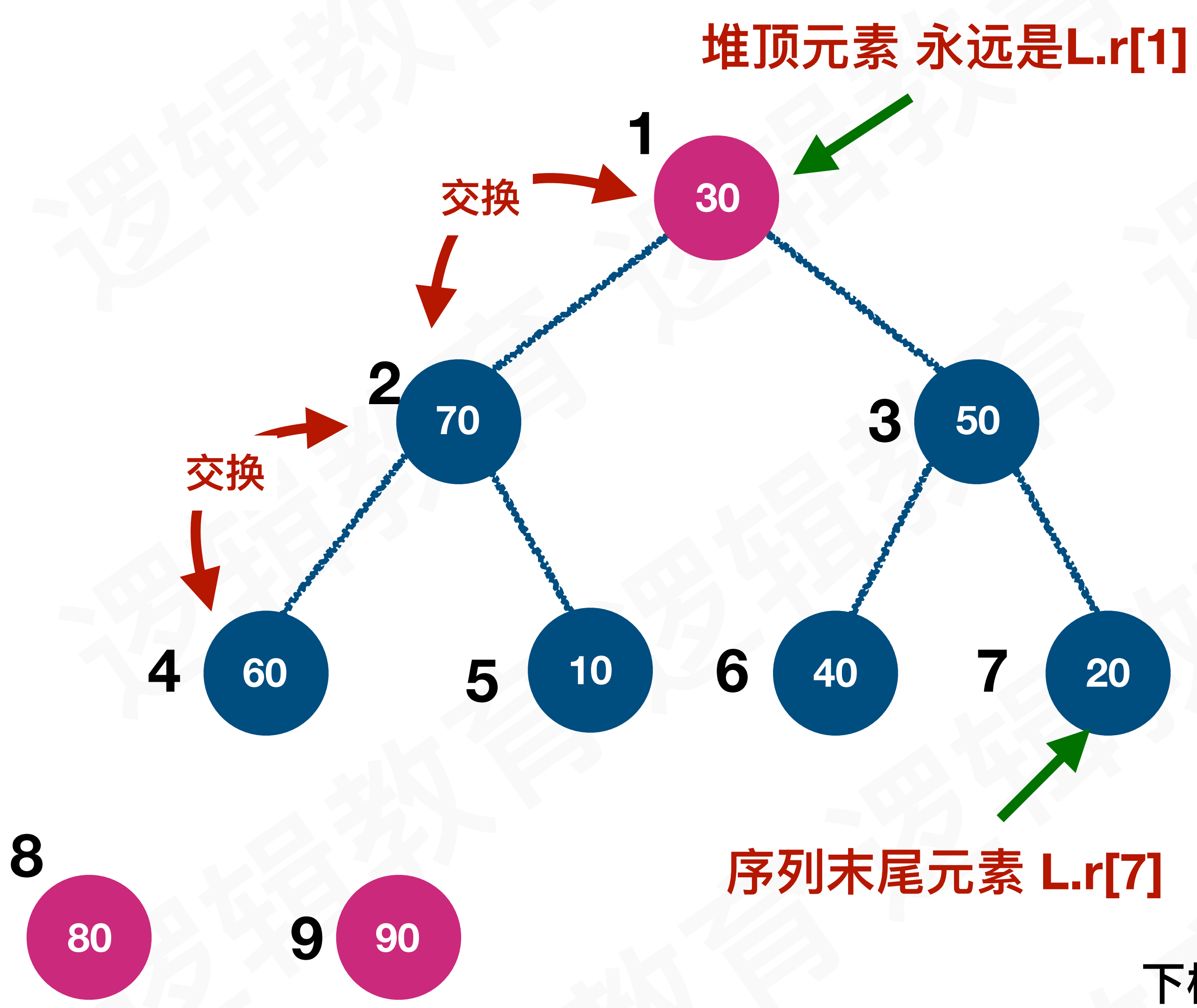
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		80	70	50	60	10	40	20	30	90
		30							80	

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 8

重新调整L.r[1...i-1] 为大顶堆



堆顶元素 永远是L.r[1]

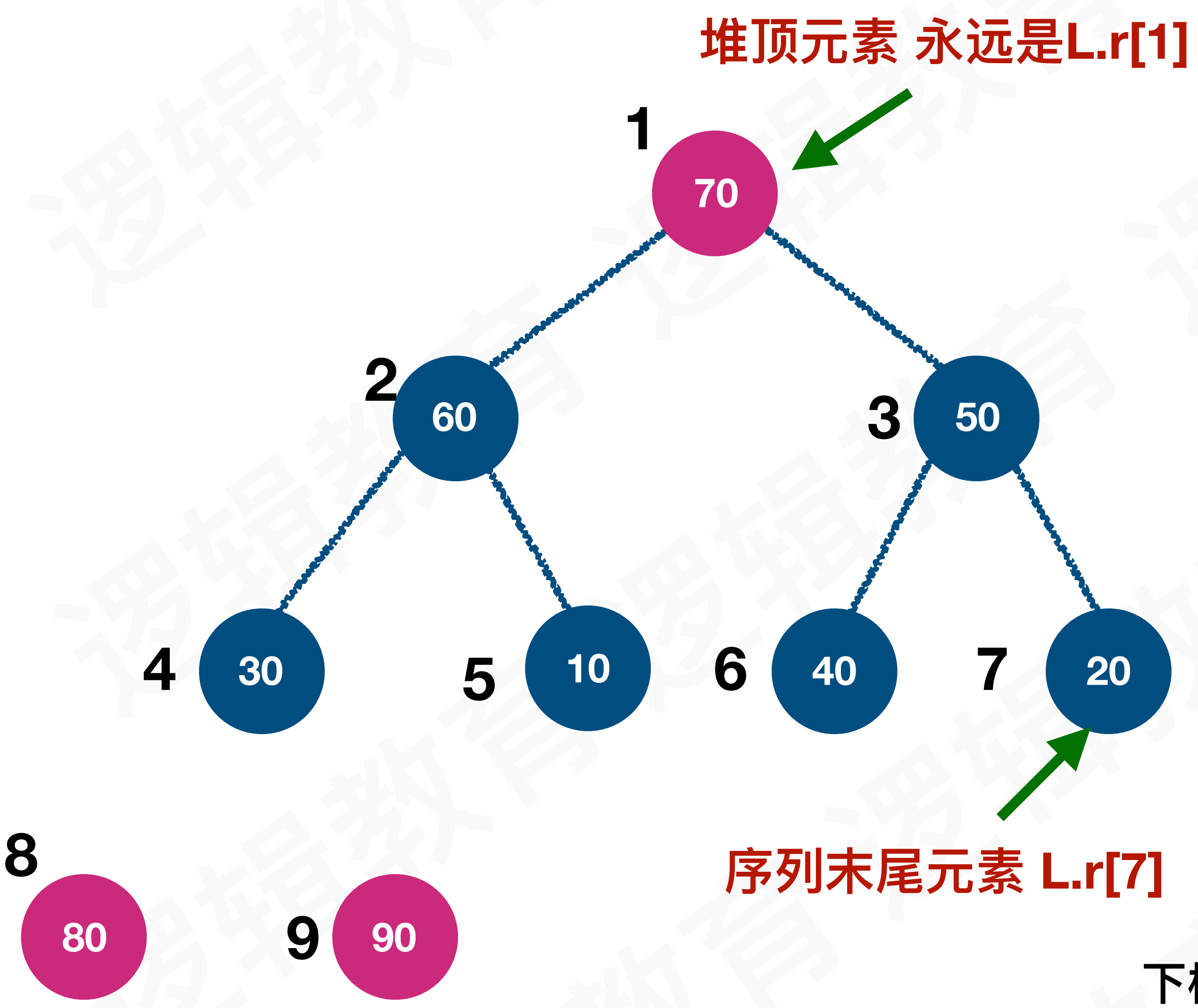
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		30	70	50	60	10	40	20	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 8

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

末尾元素

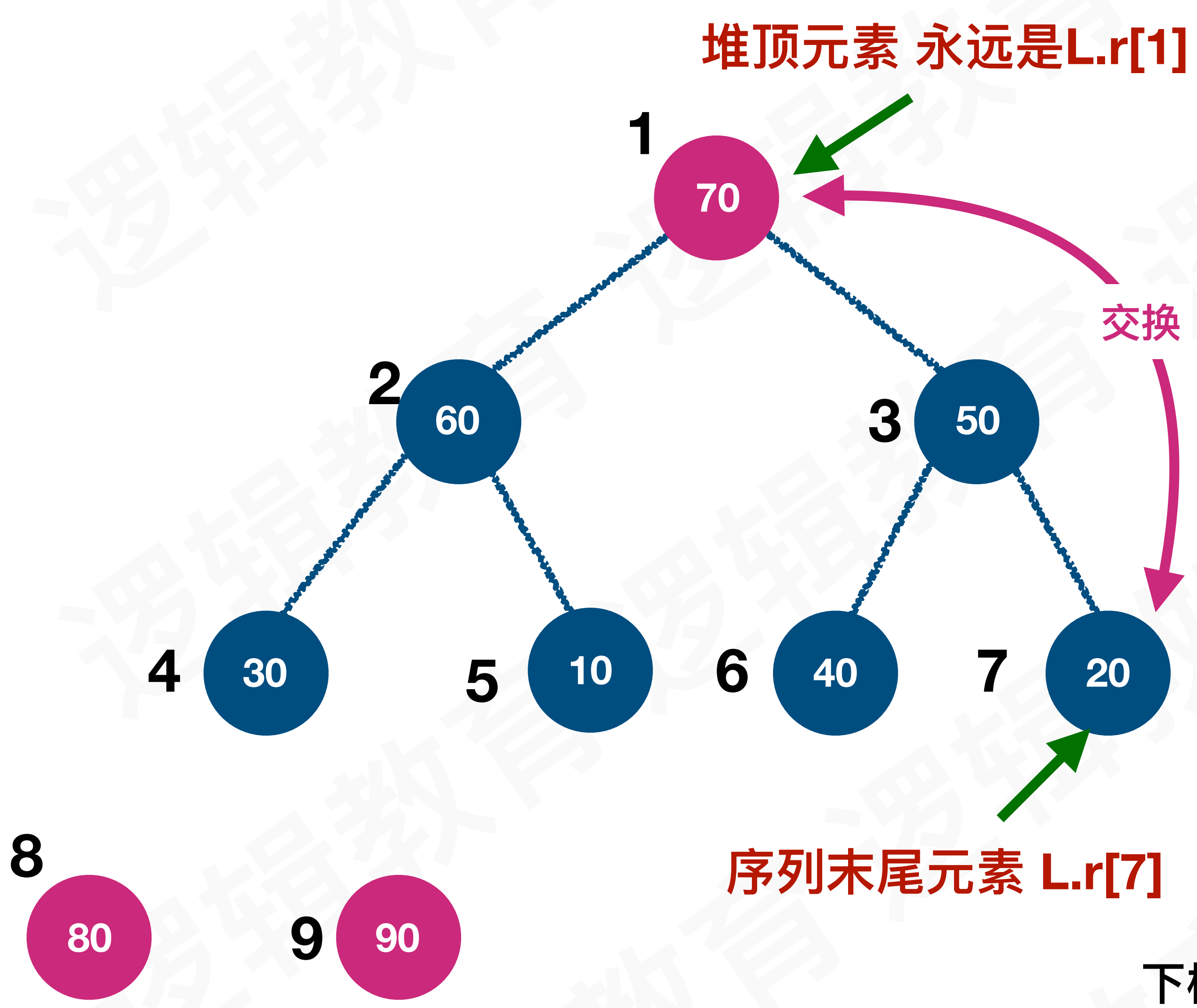
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		70	60	50	30	10	40	20	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 7

堆顶元素与序列末尾元素交换动作



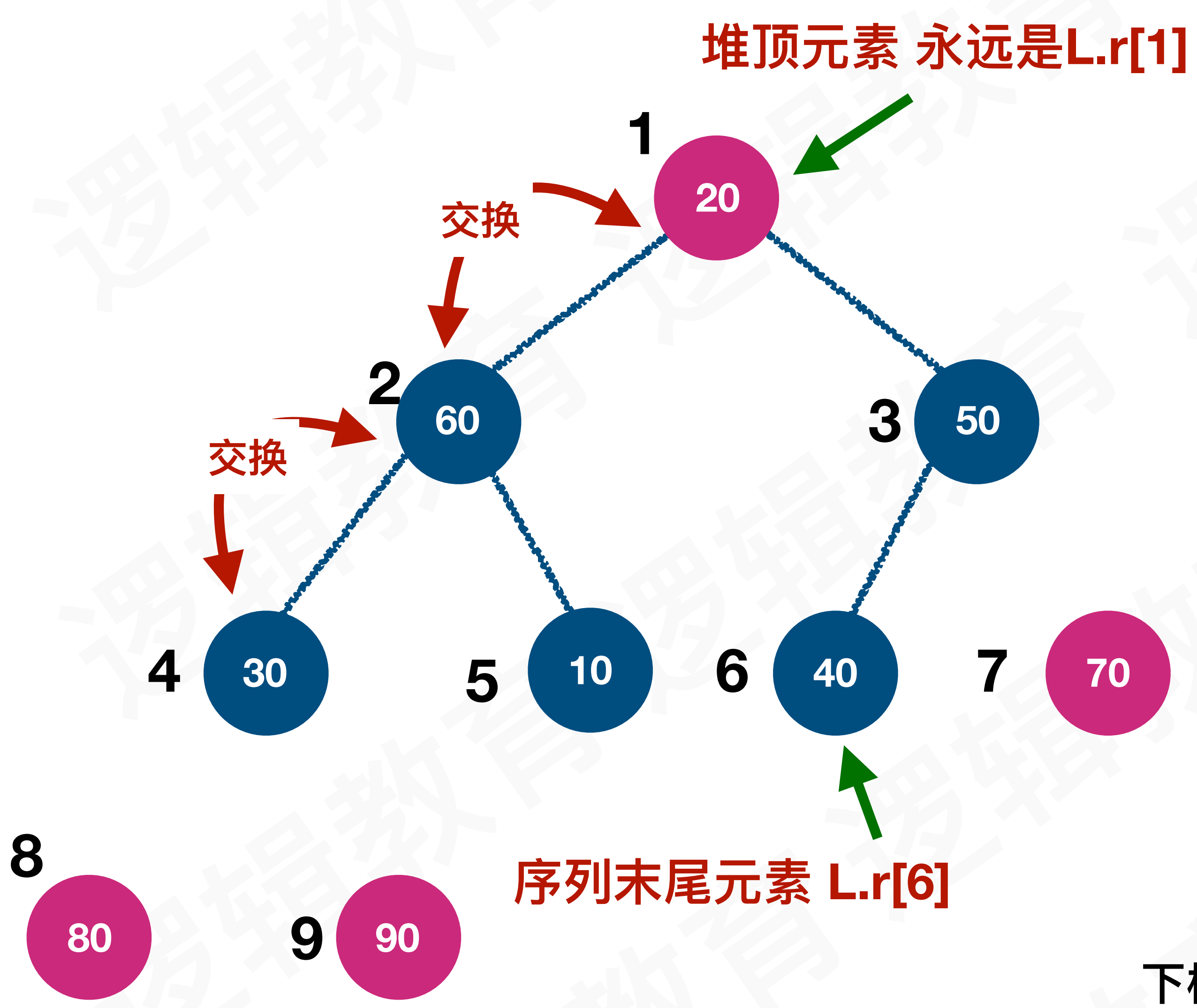
堆顶元素 永远是L.r[1]

末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		70	60	50	30	10	40	20	80	90
		20						70		

堆排序(Heap Sort) — 排序过程分析 i = 7

重新调整L.r[1...i-1] 为大顶堆



堆顶元素 永远是L.r[1]

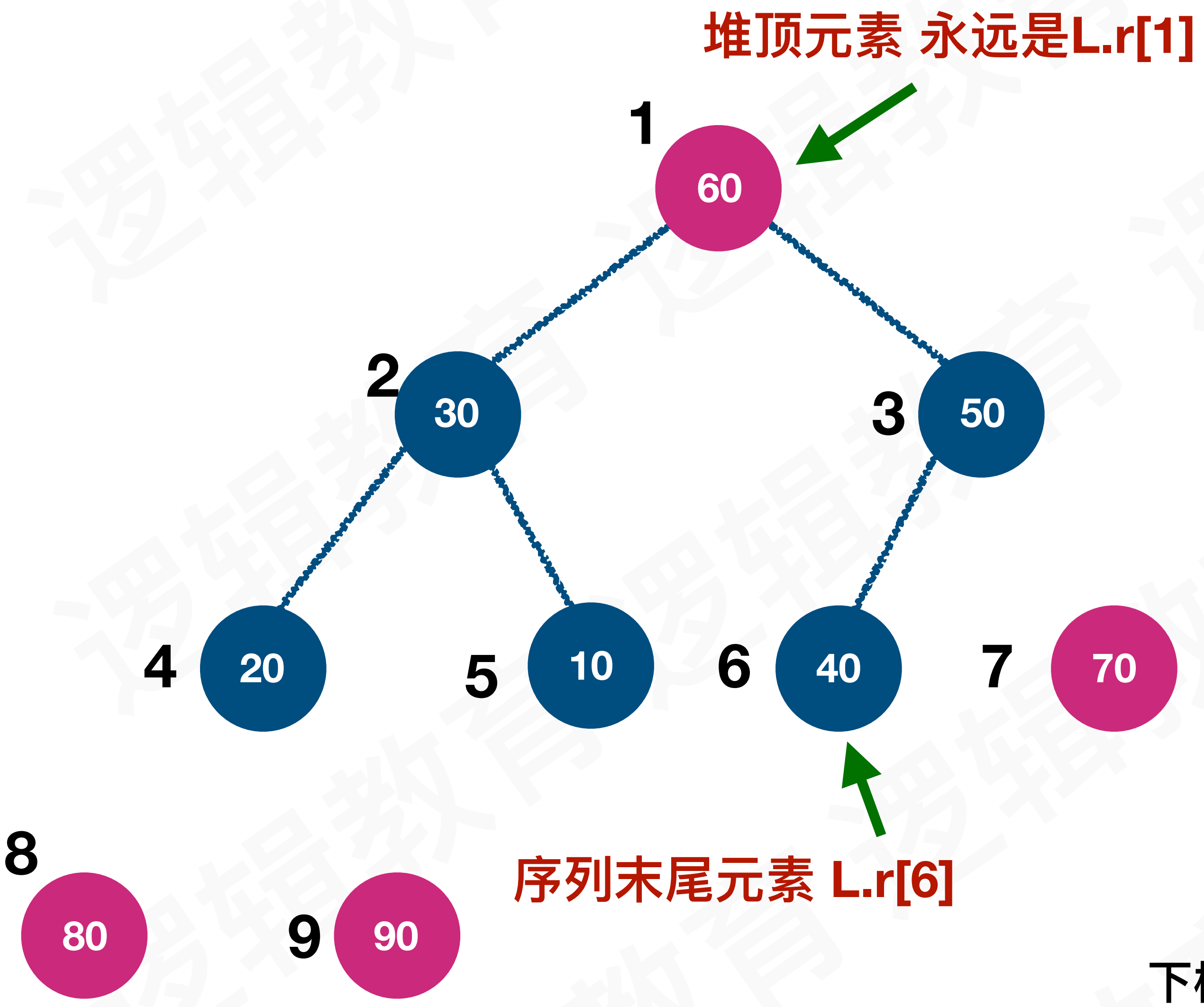
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		20	60	50	30	10	40	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 7

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

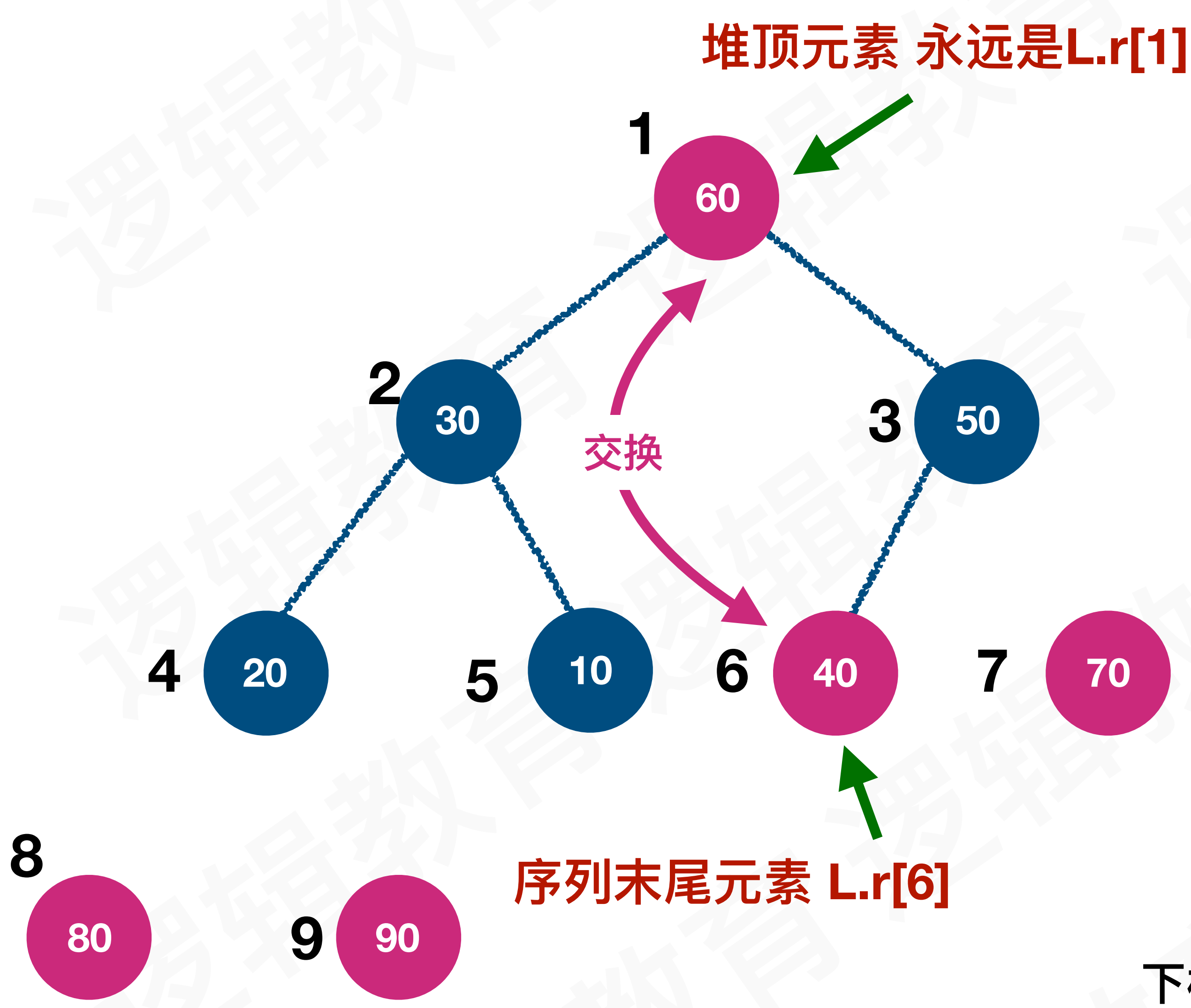
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		60	30	50	20	10	40	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 6

堆顶元素与序列末尾元素交换动作



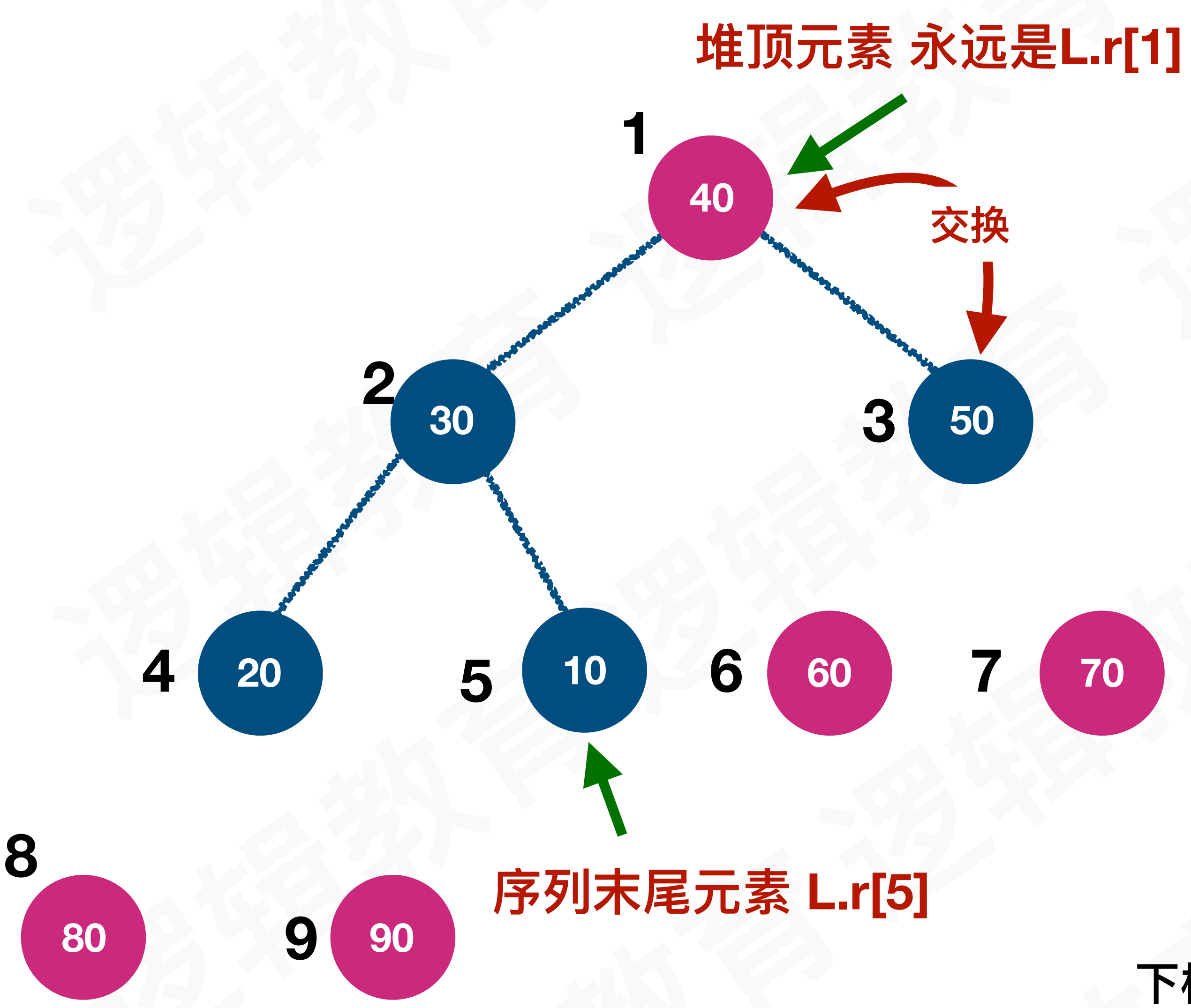
堆顶元素 永远是L.r[1]

末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		60	30	50	20	10	40	70	80	90
		40					60			

堆排序(Heap Sort) — 排序过程分析 i = 6

重新调整L.r[1…i-1] 为大顶堆



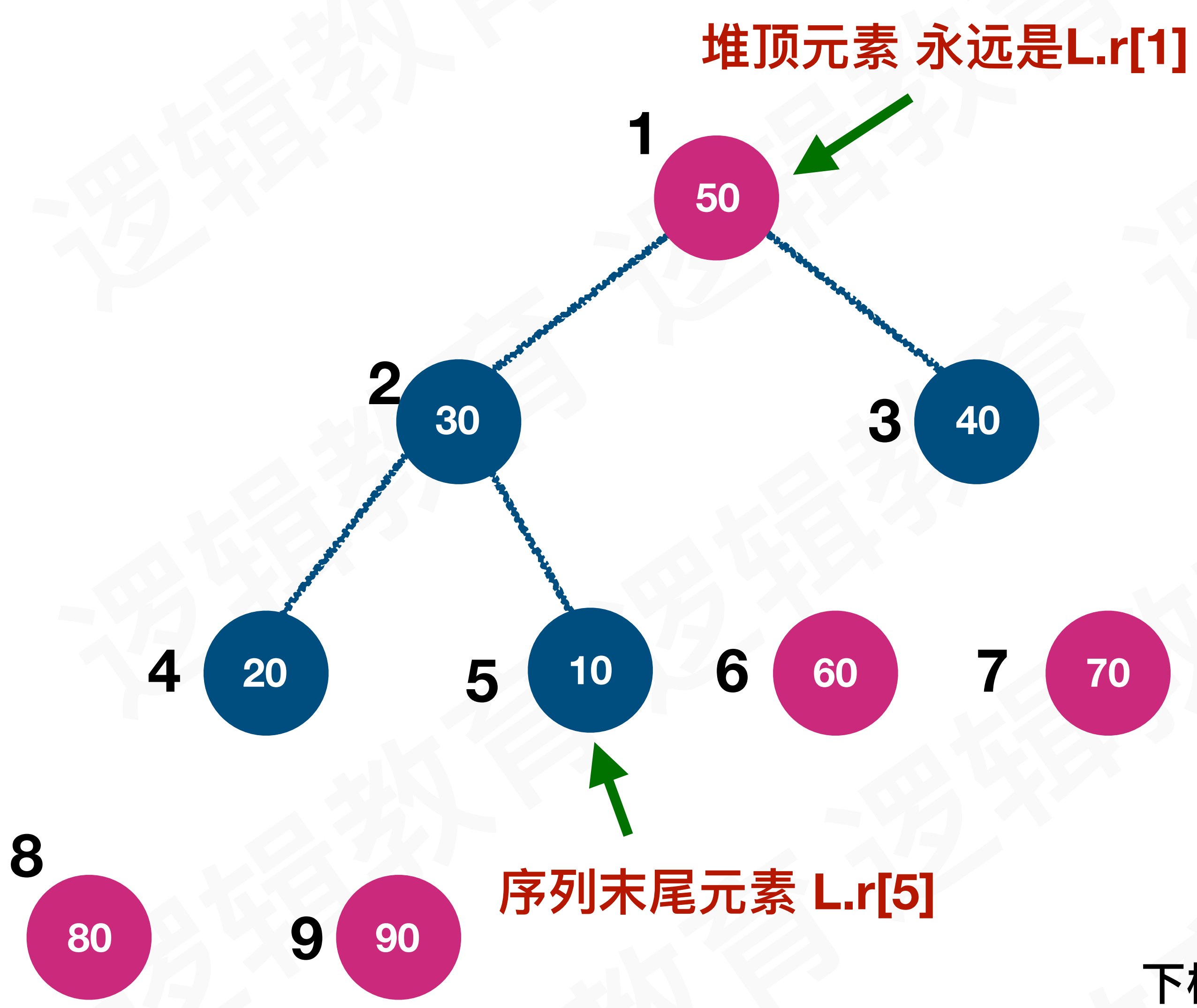
堆顶元素 永远是L.r[1]　　末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		40	30	50	20	10	60	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 6

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]　　末尾元素

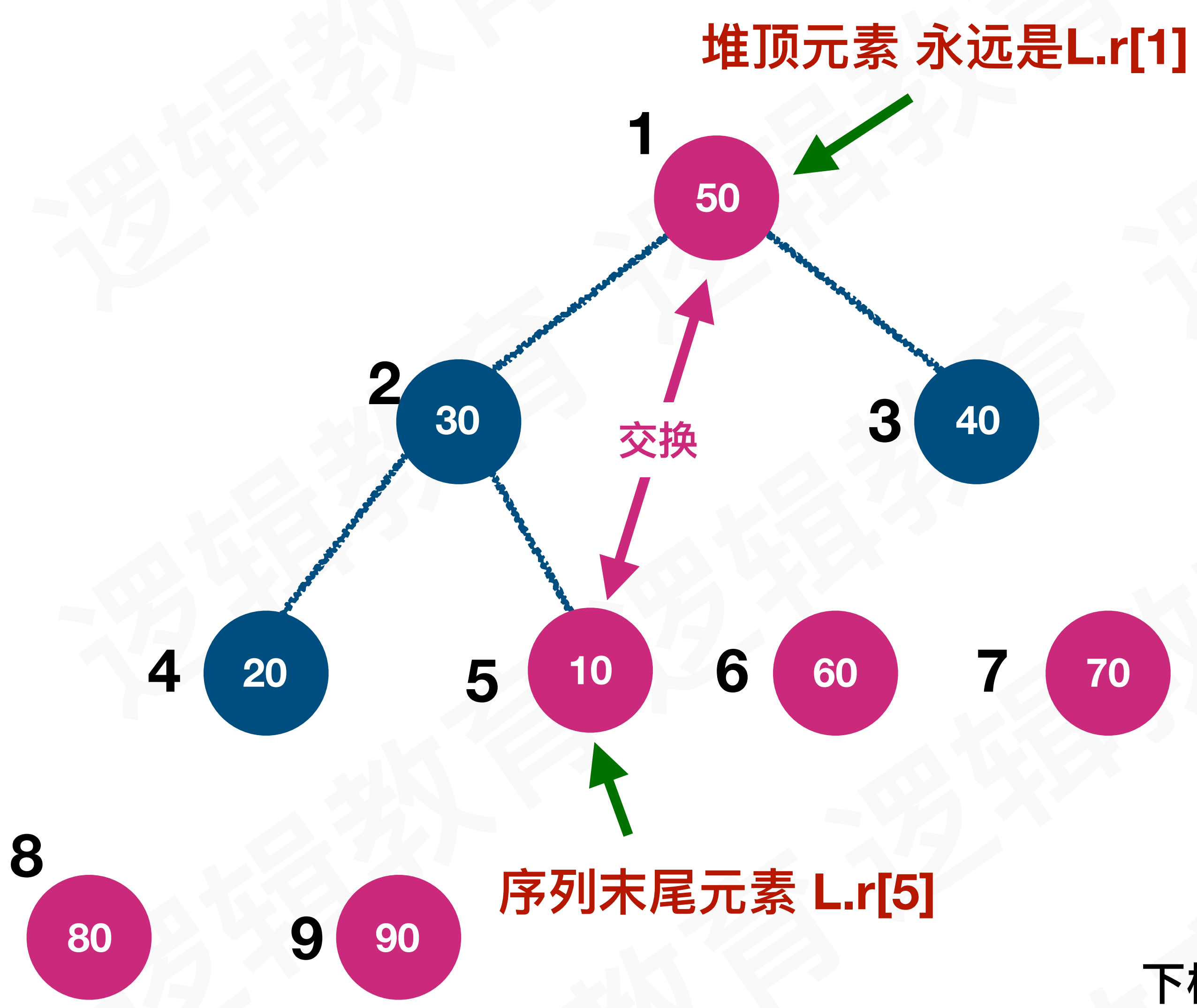
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	30	40	20	10	60	70	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 5

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]　　末尾元素

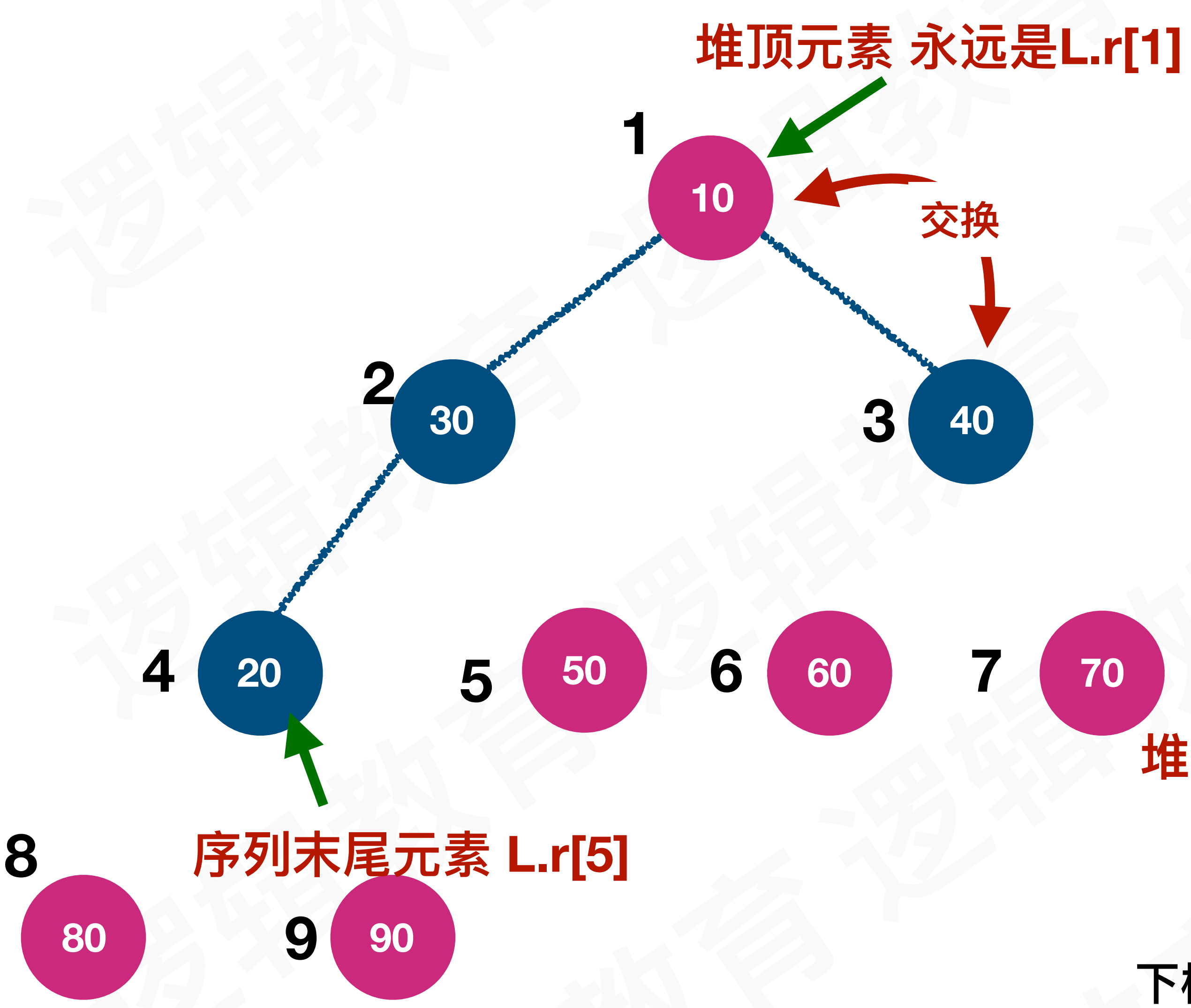
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		50	30	40	20	10	60	70	80	90
		10				50				

研发:CC老师
授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 5

重新调整L.r[1...i-1] 为大顶堆



堆顶元素 永远是L.r[1]

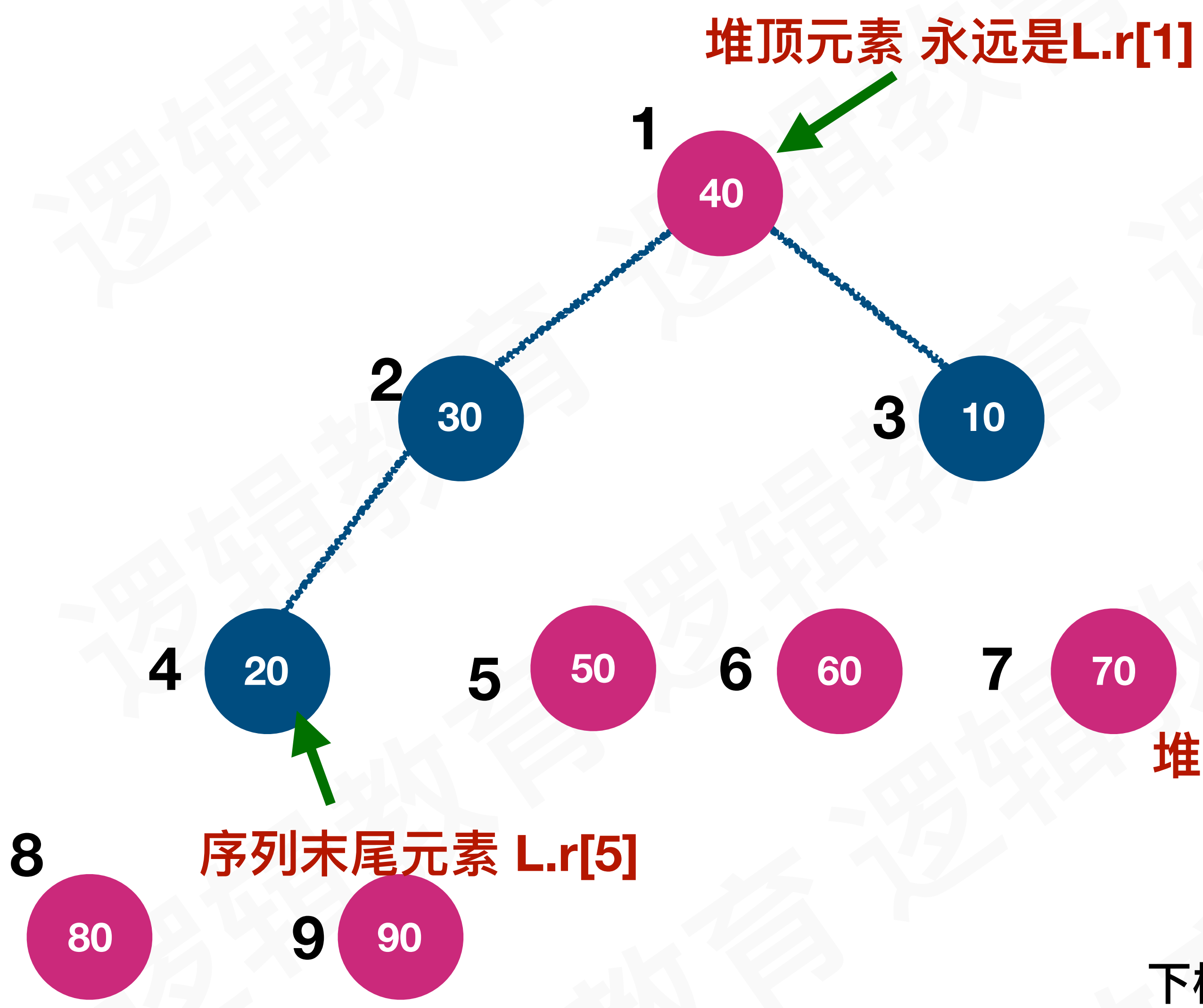
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		10	30	40	20	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 5

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

末尾元素

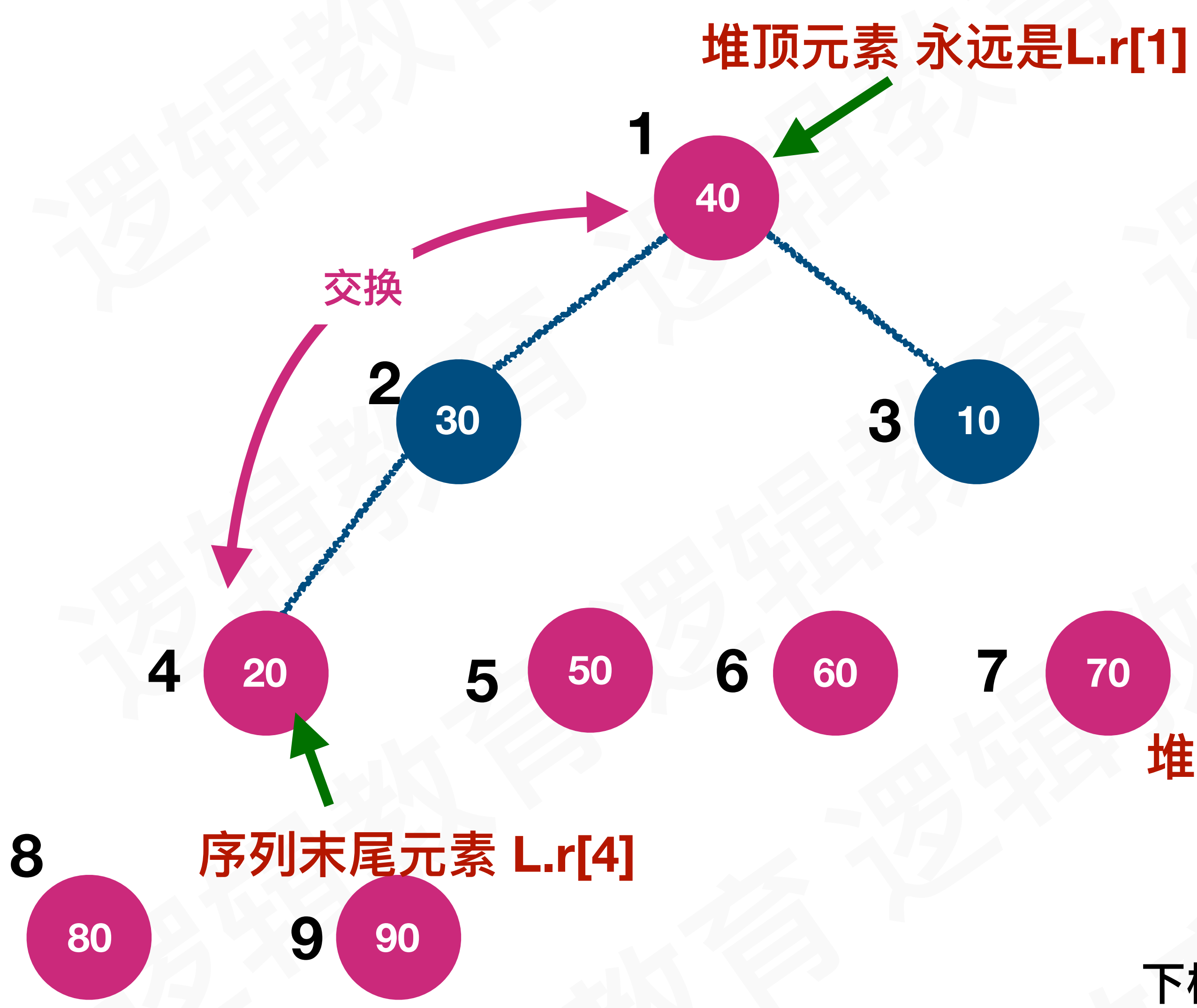
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		40	30	10	20	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 4

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]

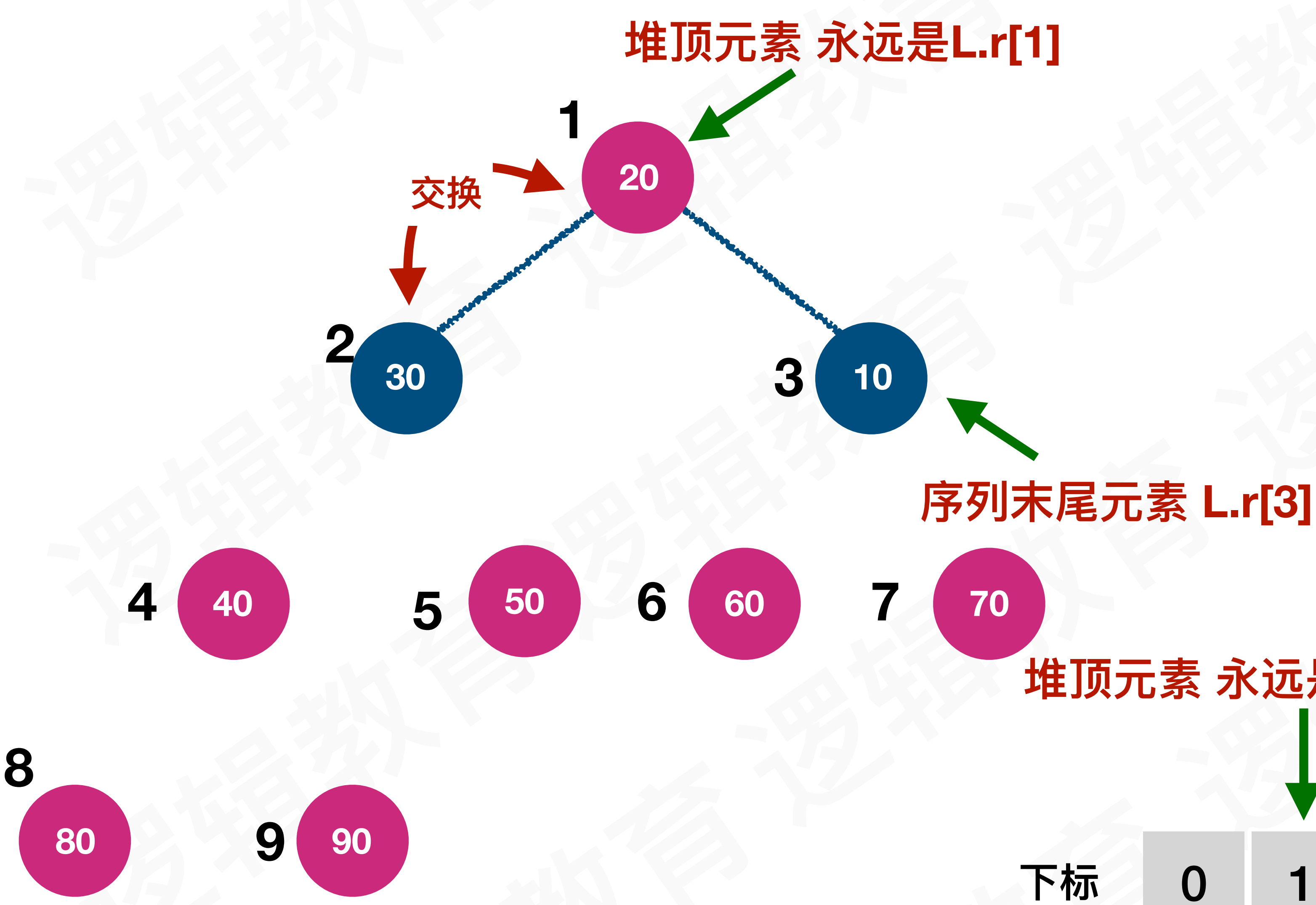
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		40	30	10	20	50	60	70	80	90
		20			40					

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 4

重新调整L.r[1...i-1] 为大顶堆



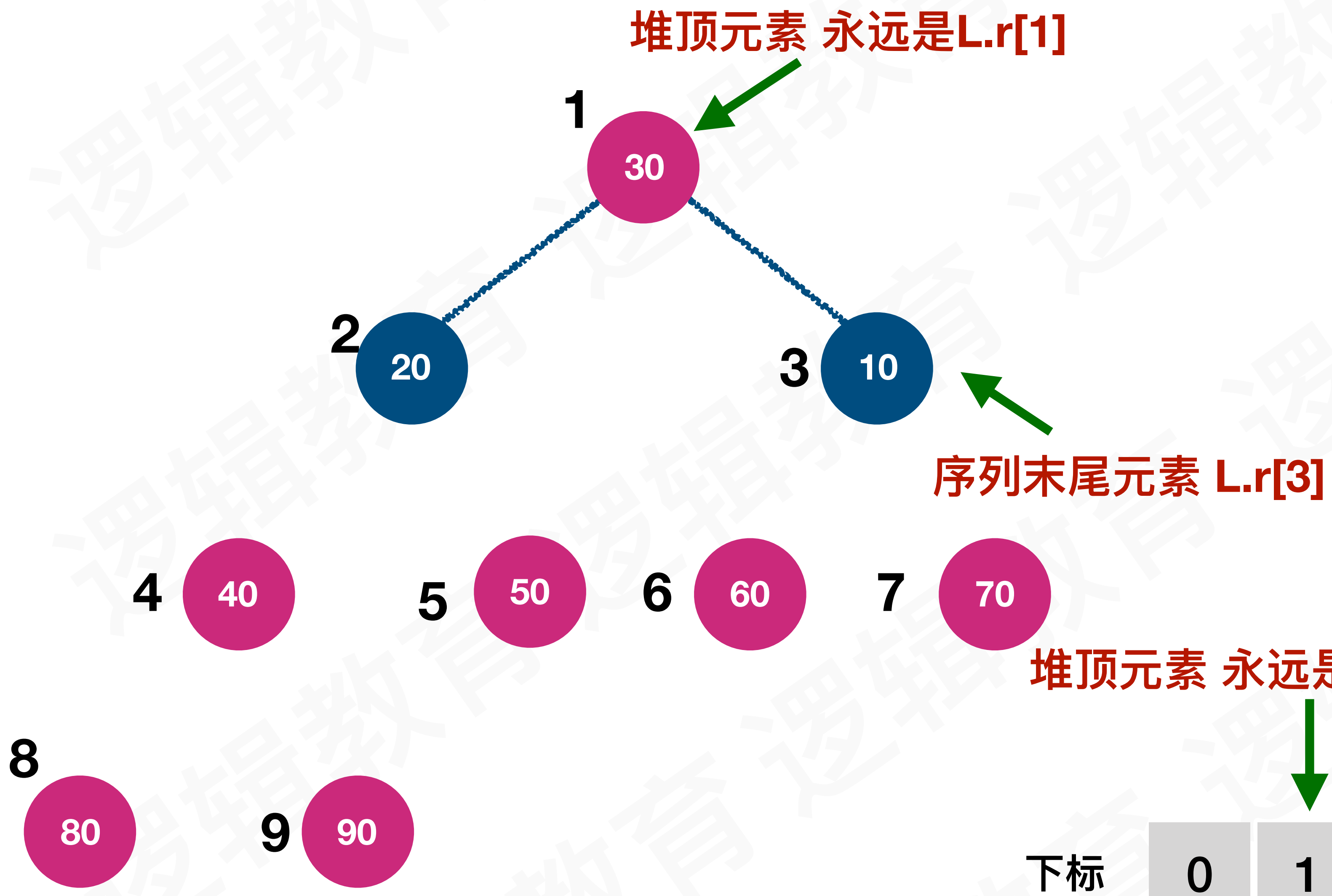
堆顶元素 永远是L.r[1]　　末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		20	30	10	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 4

已经构成了新的大顶堆



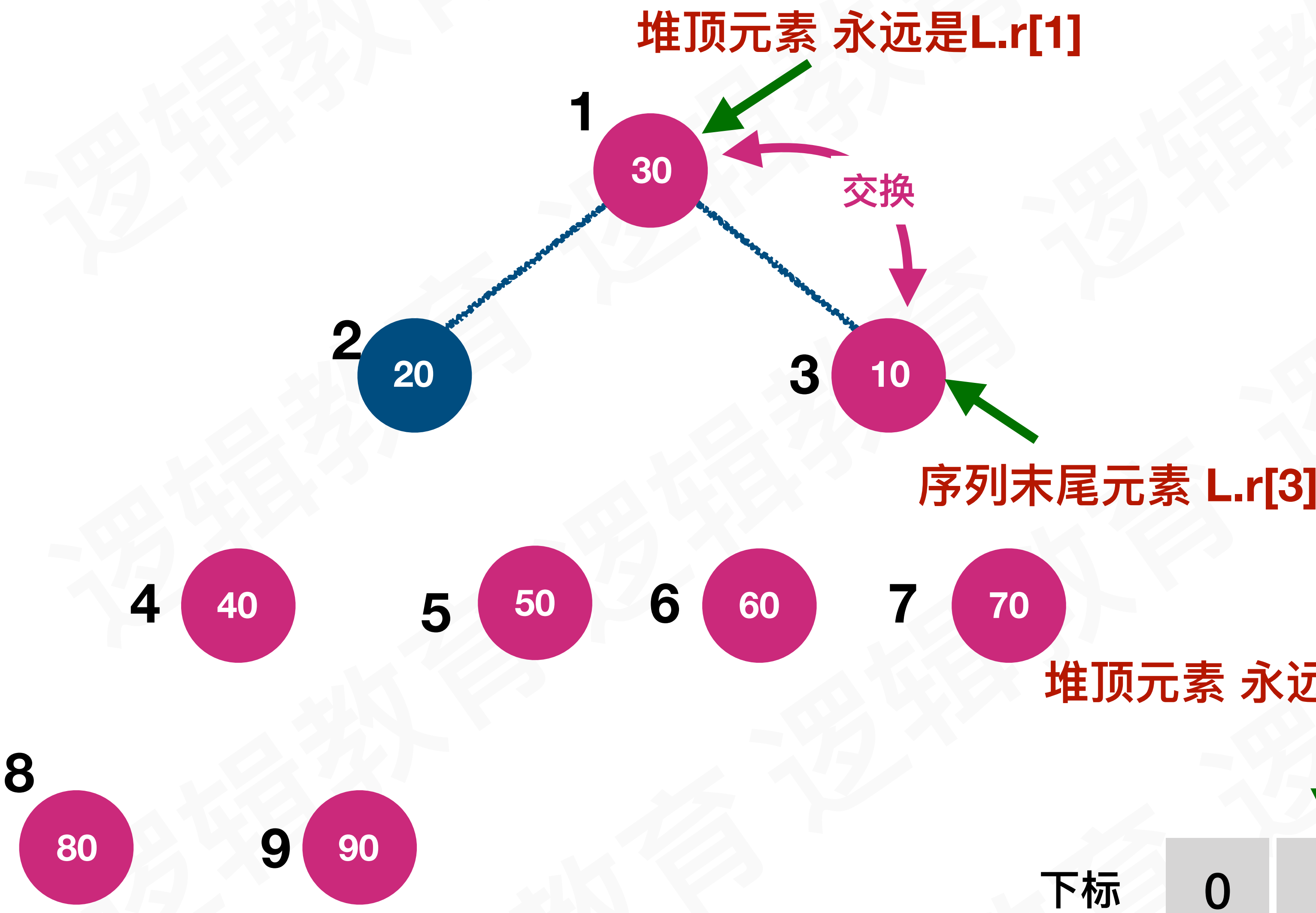
堆顶元素 永远是L.r[1]　　末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		30	20	10	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 3

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]

末尾元素

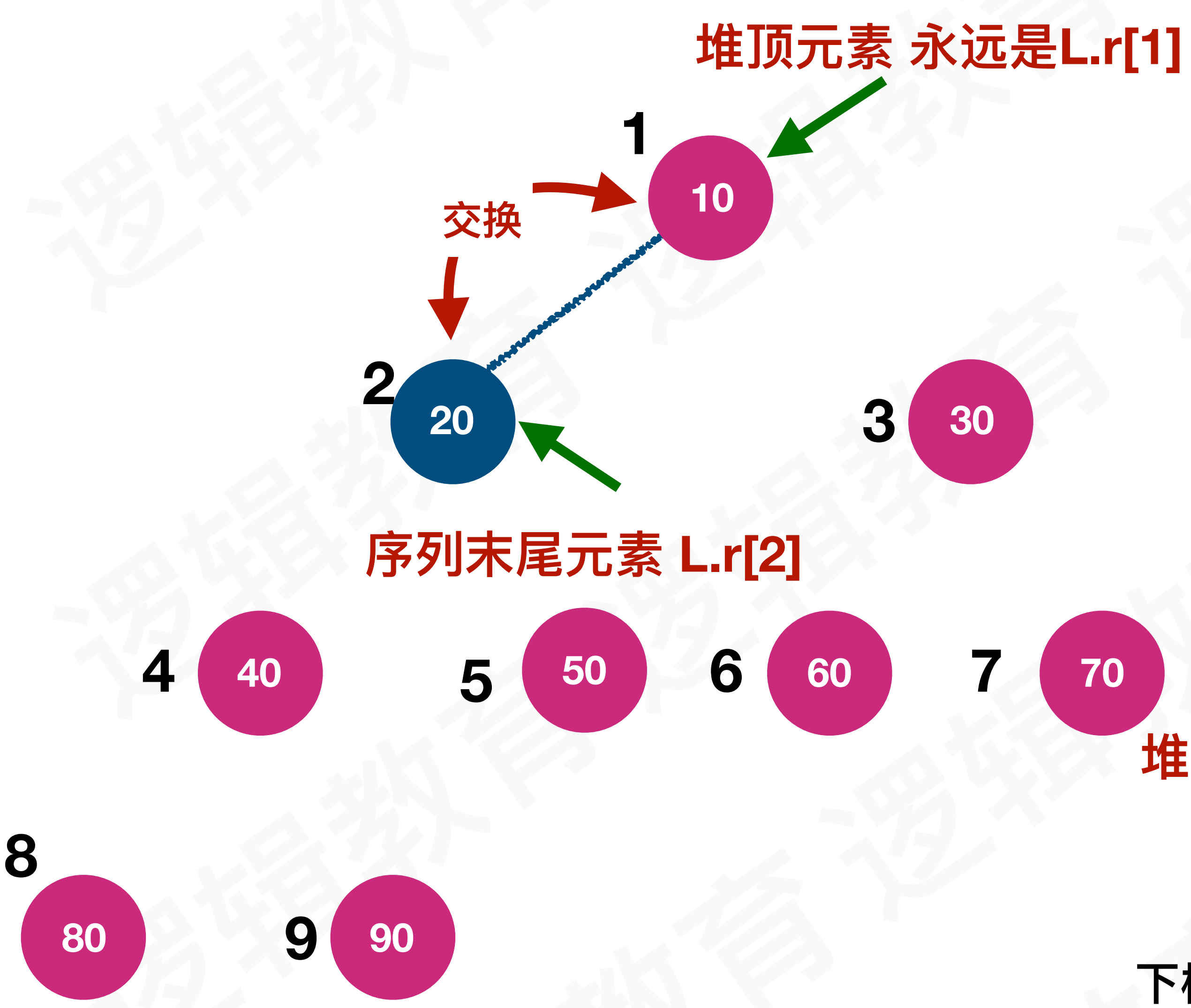
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		30	20	10	40	50	60	70	80	90
		10		30						

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 3

重新调整L.r[1...i-1] 为大顶堆



堆顶元素 永远是L.r[1]

末尾元素

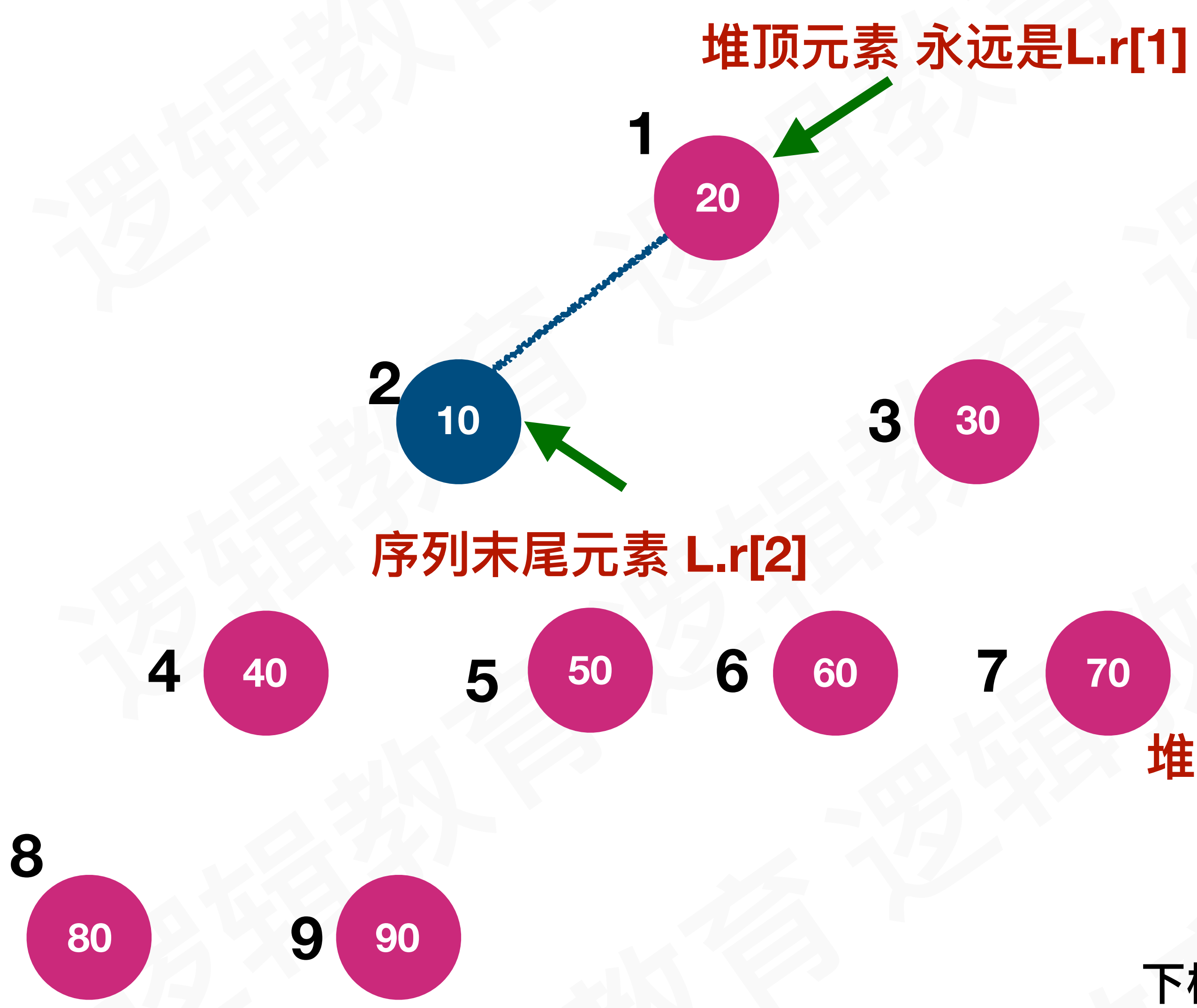
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		10	20	30	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 3

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

末尾元素

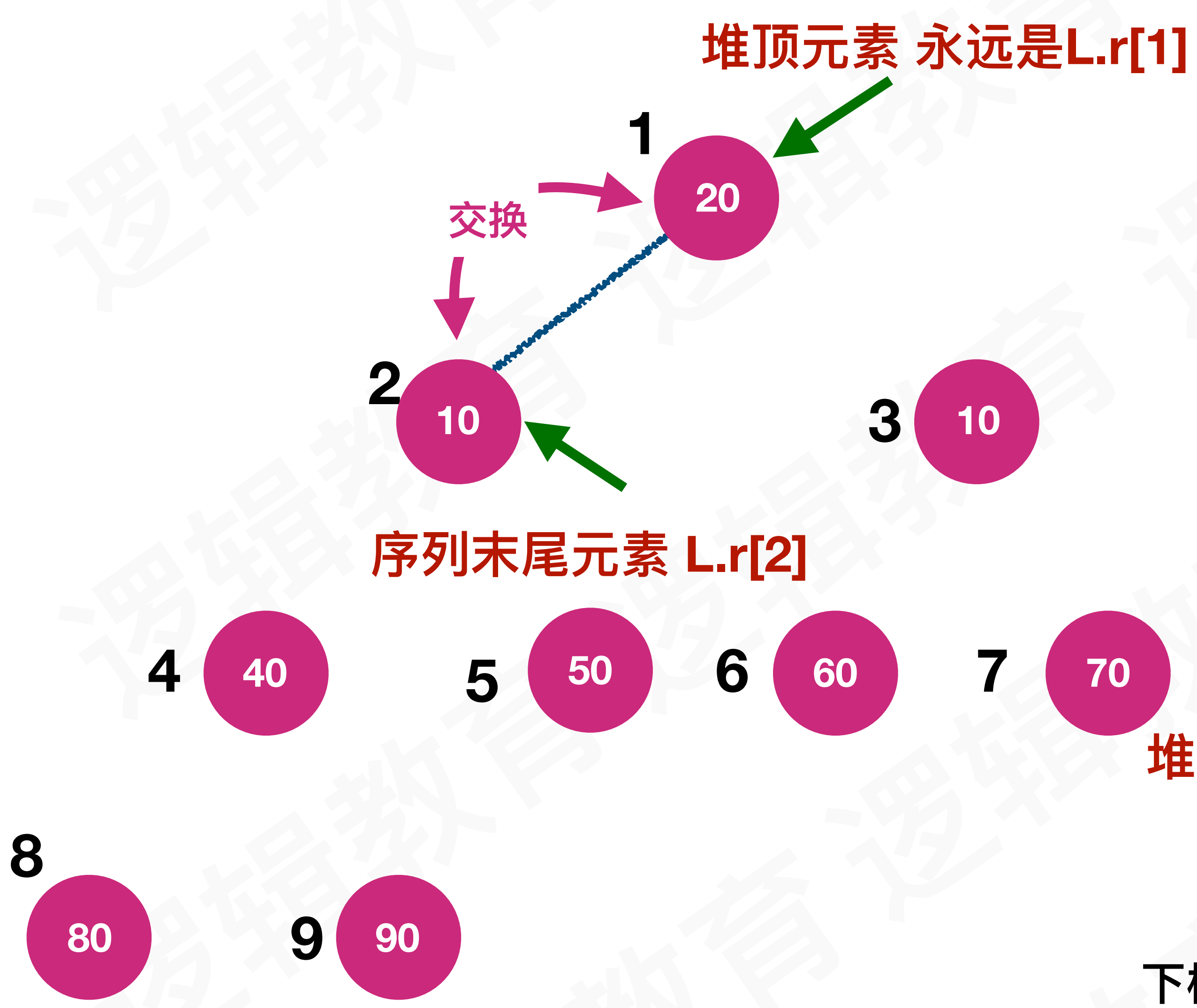
下标	0	1	2	3	4	5	6	7	8	9
大顶堆		20	10	30	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 排序过程分析 i = 2

堆顶元素与序列末尾元素交换动作



堆顶元素 永远是L.r[1]

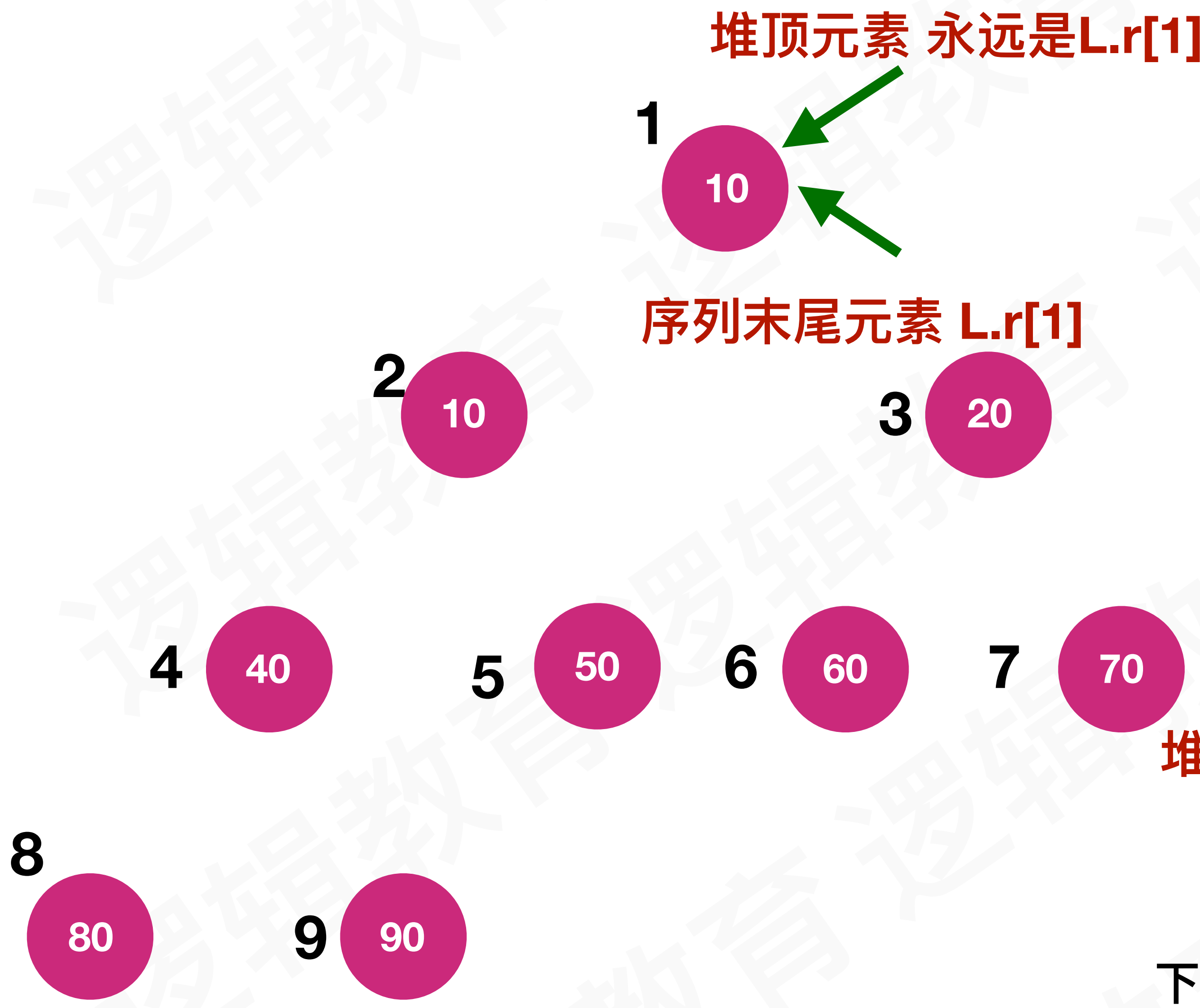
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		20	10	30	40	50	60	70	80	90
		10	20							

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 2

已经构成了新的大顶堆



堆顶元素 永远是L.r[1]

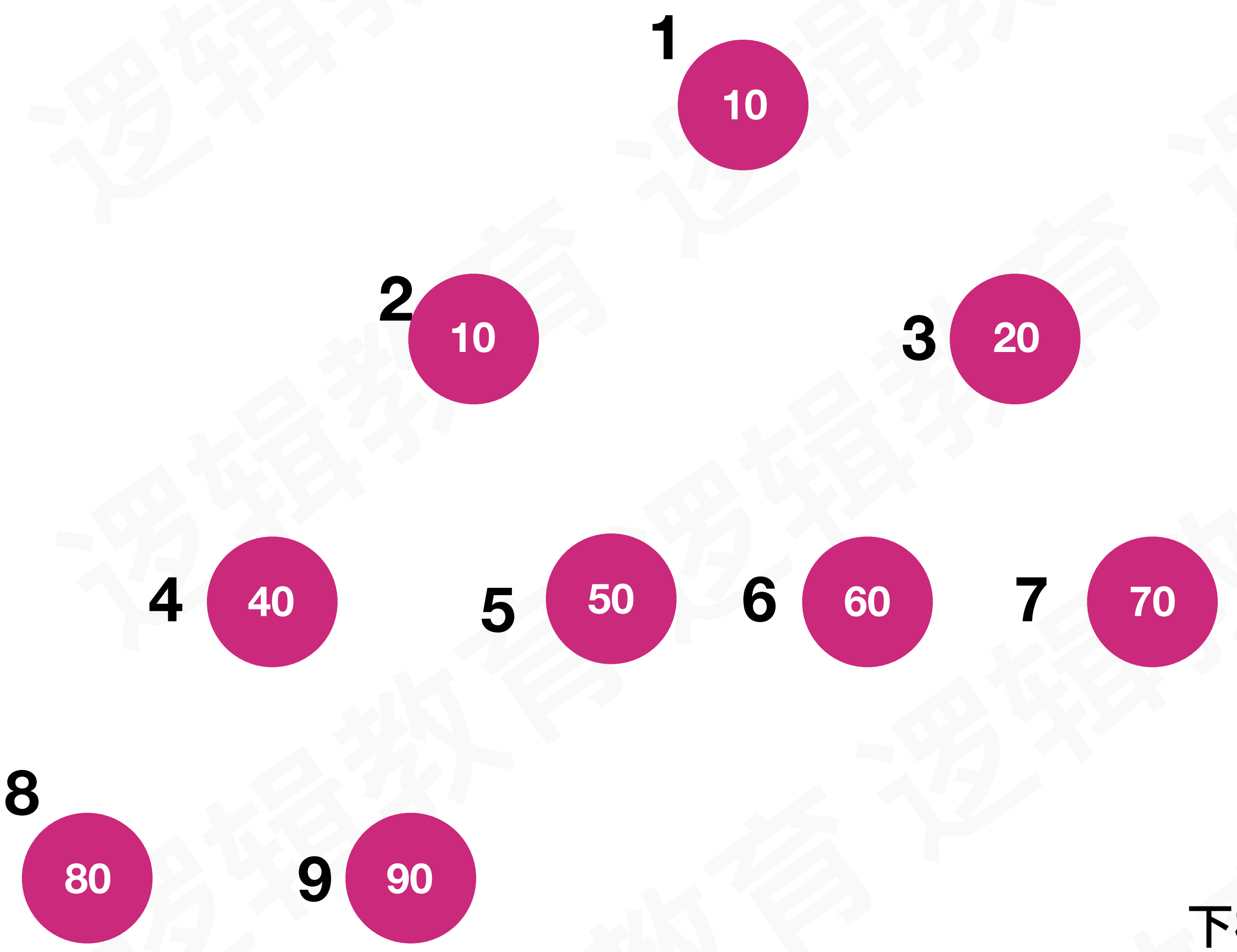
末尾元素

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		10	20	30	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

堆排序(Heap Sort) — 排序过程分析 i = 1

循环结束



堆排序完成

下标	0	1	2	3	4	5	6	7	8	9
大顶堆		10	20	30	40	50	60	70	80	90

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

堆排序(Heap Sort) — 复杂度分析

堆排序的时间复杂度为：O(nlogn)

堆排序是就地排序，空间复杂度为常数：O(1)

堆排序的运行时间主要消耗在初始构建堆 和重建对的反复筛选上;

初始化建堆过程时间：O(n)

推算过程：

首先要理解怎么计算这个堆化过程所消耗的时间:

假设高度为k，则从倒数第二层右边的节点开始，这一层的节点都要执行子节点比较然后交换（如果顺序是对的就不用交换）；倒数第三层呢，则会选择其子节点进行比较和交换，如果没交换就可以不用再执行下去了。如果交换了，那么又要选择一支子树进行比较和交换；

那么总的时间计算为：s = 2^(i - 1) * (k - i)；其中 i 表示第几层，2^(i - 1) 表示该层上有多少个元素，(k - i) 表示子树上要比较的次数，如果在最差的条件下，就是比较次数后还要交换；因为这个是常数，所以提出来后可以忽略； S = 2^(k-2) * 1 + 2^(k-3)*2.....+2*(k-2)+2^(0)*(k-1) ===> 因为叶子层不用交换，所以i从 k-1 开始到 1；

这个等式求解,等式左右乘上2，然后和原来的等式相减，就变成了：S = 2^(k - 1) + 2^(k - 2) + 2^(k - 3) ..... + 2 - (k-1)

除最后一项外，就是一个等比数列了，直接用求和公式：S = { a1[1- (q^n)] } / (1-q)；

S = 2^k -k -1；又因为k为完全二叉树的深度，所以 (2^k) <= n < (2^k -1)，总之可以认为：k = logn （实际计算得到应该是 log(n+1) < k <= logn ）；

综上所述得到：S = n - longn -1，所以时间复杂度为：O(n)

更改堆元素后重建堆时间：O(nlogn)

推算过程：

1、循环 n -1 次，每次都是从根节点往下循环查找，所以每一次时间是 logn，总时间：logn(n-1) = nlogn - logn

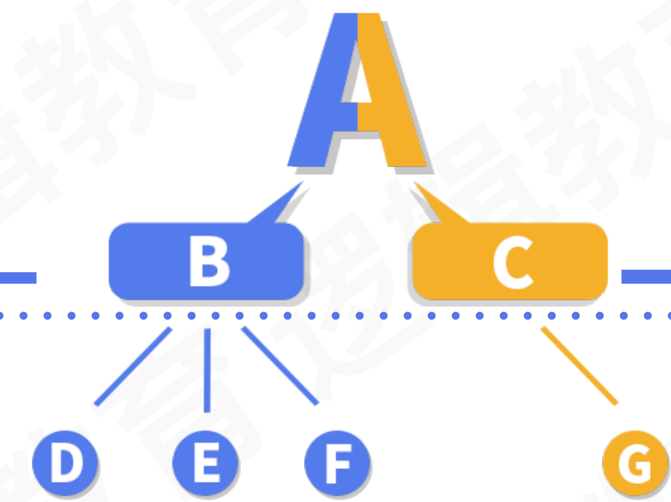
课程研发:CC老师
课程授课:CC老师

堆排序的时间复杂度为：O(nlogn)

逻辑教育
Logic education

Class Ending !

thanks, see you next time



数据结构与算法主题

@CC老师

全力以赴 · 非同凡"想"

课程研发:CC老师
课程授课:CC老师