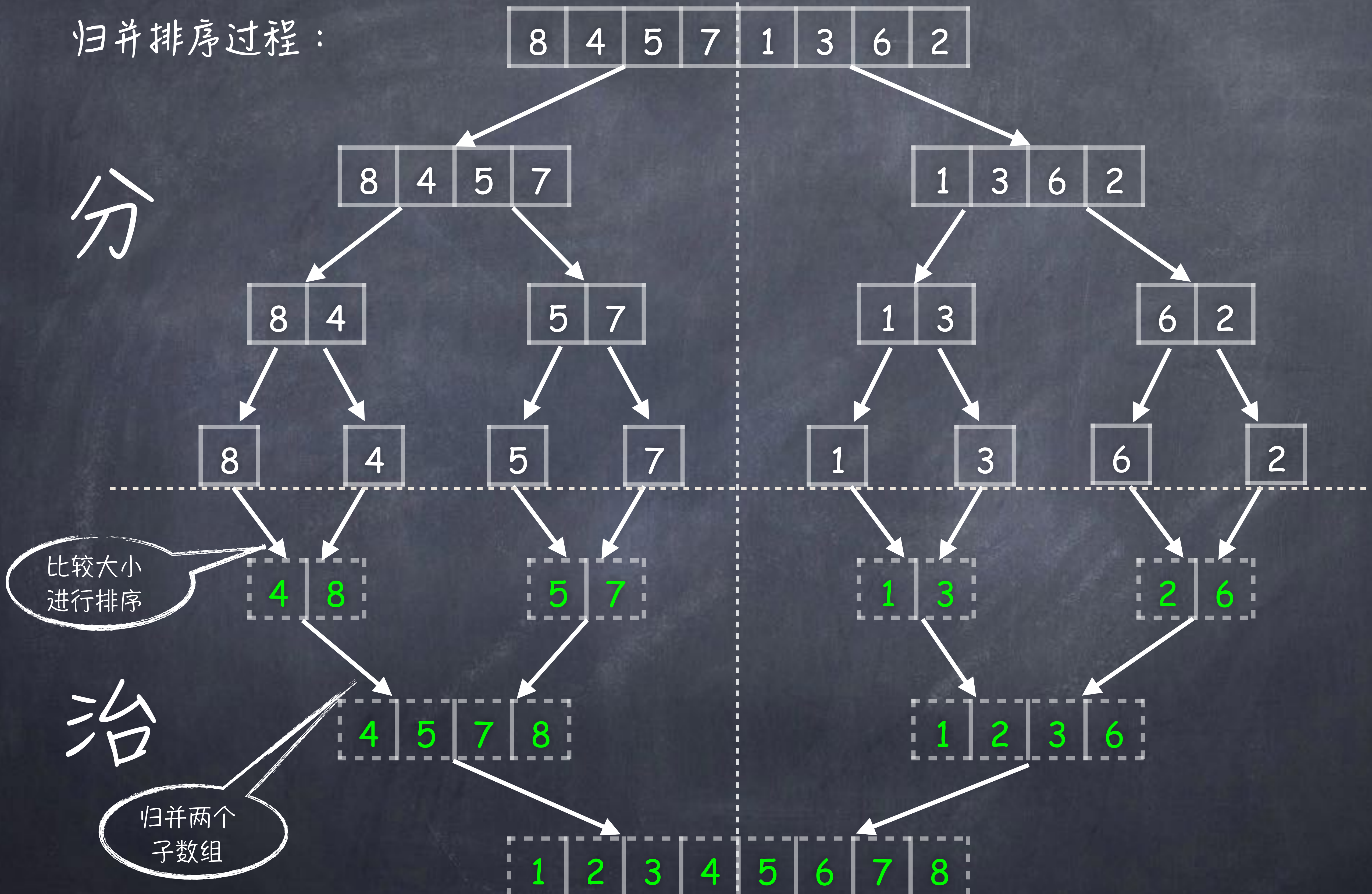
归并排序

归并排序过程：



8 4 5 7 1 3 6 2
分
8 4 5 7
1 3 6 2
8 4
5 7
1 3
6 2
8
4
5
7
1
3
6
2
比较大小
进行排序
4 8
5 7
1 3
2 6
治
4 5 7 8
1 2 3 6
归并两个
子数组
1 2 3 4 5 6 7 8

算法基本思想（分治）：

1. 将数组划分为两个子数组(二路归并)
2. 对两个子数组进行排序
3. 将排好序的两个子数组进行有序归并
4. 递归的执行此过程

2. Python实现归并函数

```
def merge(left, right):
    llen = len(left)
    lcur = 0
    rlen = len(right)
    rcur = 0
    result = [] #临时空间
    while lcur < llen and rcur < rlen:
        lone = left[lcur]
        rone = right[rcur]
        result.append(min(lone, rone)) #每次取最小的
        if lone < rone:
            lcur += 1
        else:
            rcur += 1
    result += left[lcur:] #如果有一个取完另一个直接连接
    result += right[rcur:]
    return result
```

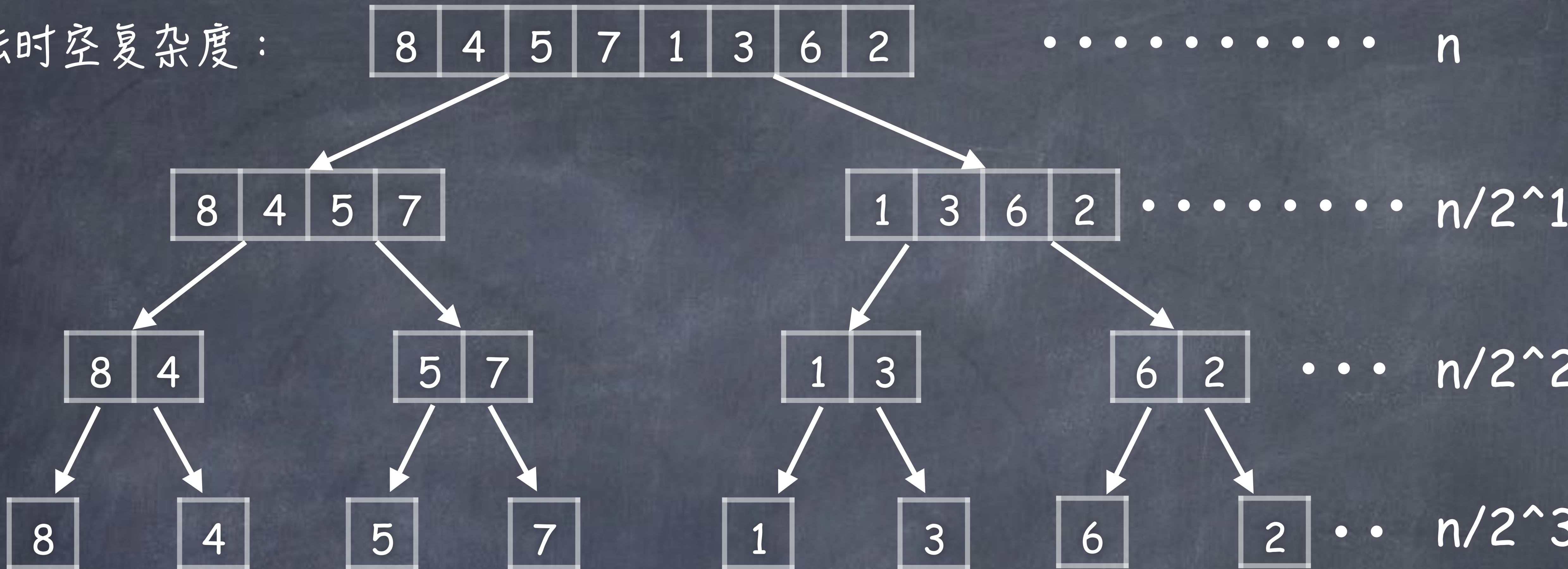
1. Python实现递归函数

```
def msort_rec(array):
    length = len(array)
    if length == 1: #如果长度为1返回
        return array
    else:
        mid = length / 2 #每次划分两个数组
        #左递归
        left = msort_rec(array[0: mid])
        #右递归
        right = msort_rec(array[mid: length])
        return merge(left, right) #进行归并
```

3. Python实现主函数

```
if __name__ == '__main__':
    L = [8, 4, 5, 7, 1, 3, 6, 2]
    print "排序前: %r" %(L)
    R = msort_rec(L)
    print "排序后(递归): %r" %(R)
```

算法时空复杂度：



1) 由图可知递归公式为：

当n=1时：$T(1) = 1$

当n<>1时：$T(n) = 2T(n/2)+n$

由于每次归并总和都要扫描n次

2) $T(n) = 2T(n/2)+n$

$= 4T(n/4)+n+n$

$= 8T(n/8) +n+n+n$

3) 假设$n=2^k$，那么：

$8T(n/8)+n+n+n = 2^kT(n/8)+kn$

由于$n=2^k$两边去log2,则 $\log(n) = k$

再代回去：

$T(n) = 2^{\log(n)}T(n/8)+n\log(n)$

$=n*1+n\log(n) = n+n\log(n)$

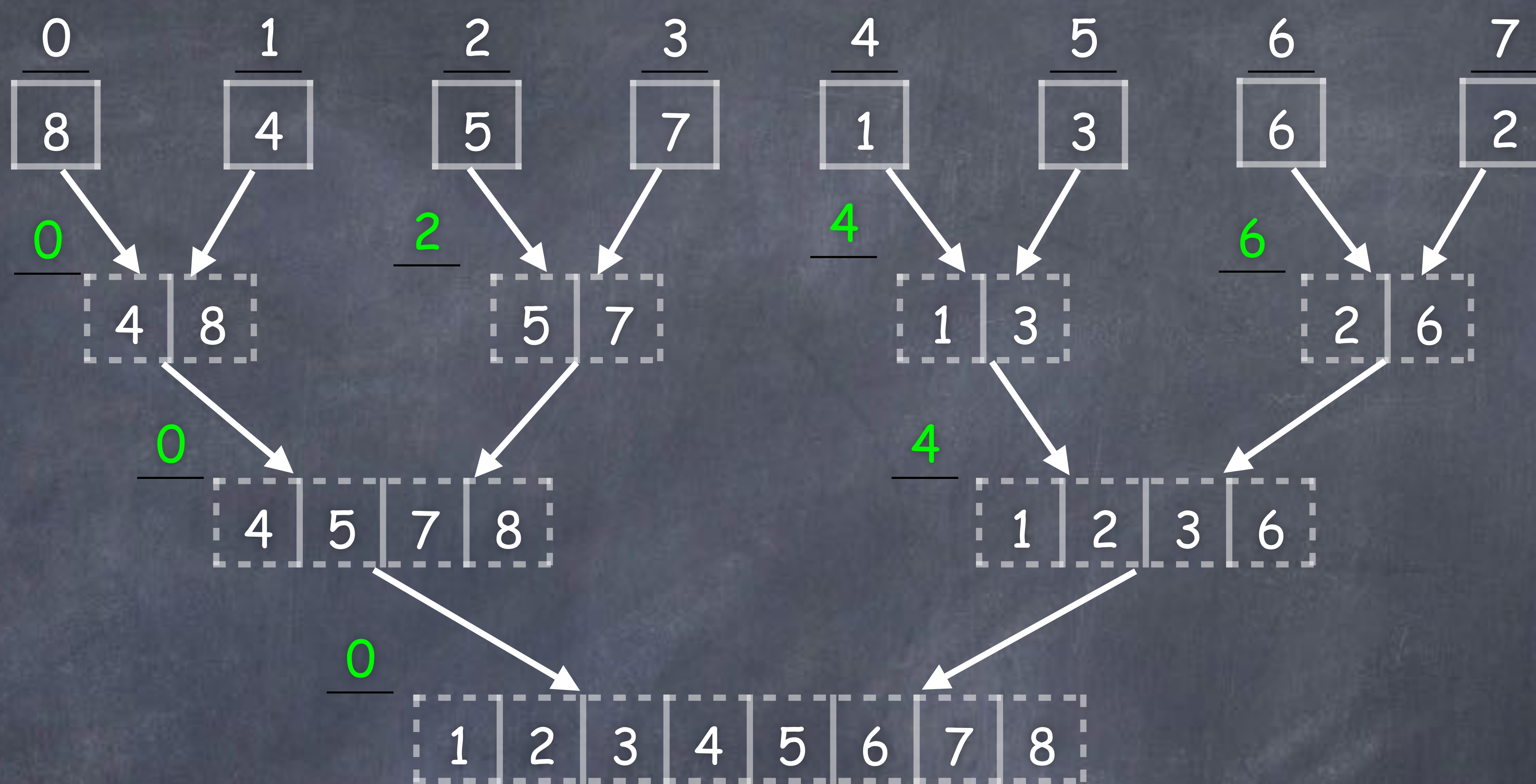
由于$n\lg n > n$

所以时间复杂度为: $O(n\lg n)$

4) 由于归并过程中需要临时数组，每次为O(n)、O(n/2)、O(n/4)、O(n/8)···,最大一次占用为O(n)

故算法的空间复杂度为: O(n)

算法非递归实现：



Python实现数

```
def msort_iter(array):
    length = len(array)
    step = 1 #步长从1开始
    while step < length:
        for left in range(0, length - step, 2 * step): #分组坐标
            result = merge(array[left:left + step],
                           array[left + step: min(left + 2 * step,
                                                  length)]) #归并
            array = array[0:left] + result + array[min(left + 2 *
                                                   step, length):]
        step = step * 2 #每次步长增长两倍
    return array
```

第1次分组：步长为1，每组两个元素，坐标为0、2、4、6，每组进行归并

第2次分组：步长为2，每组四个元素，坐标为0、4，每组进行归并

第3次分组：步长为4，每组八个元素，坐标为0，每组进行归并

多路归并排序的应用：

敬请期待