进制转换

· 12345转化为八进制是多少?

· -12345在内存中是怎样存储的?

答案

· 除n取余法
30071
· 同上方法求出二进制码
1011 0000 0011 1001
再求出反码
1100 1111 1100 0110
补码
1100 1111 1100 0111

· 思考：为什么要以补码的形式存储负数?

求Top3

给一个数组，求出其中第三大的数
方法一： 维护一个有序链表， 每成功插入一个数，头指针向后挪一位
ps：为什么不用数组?

方法二：使用小顶堆

QQ截图20190412223957

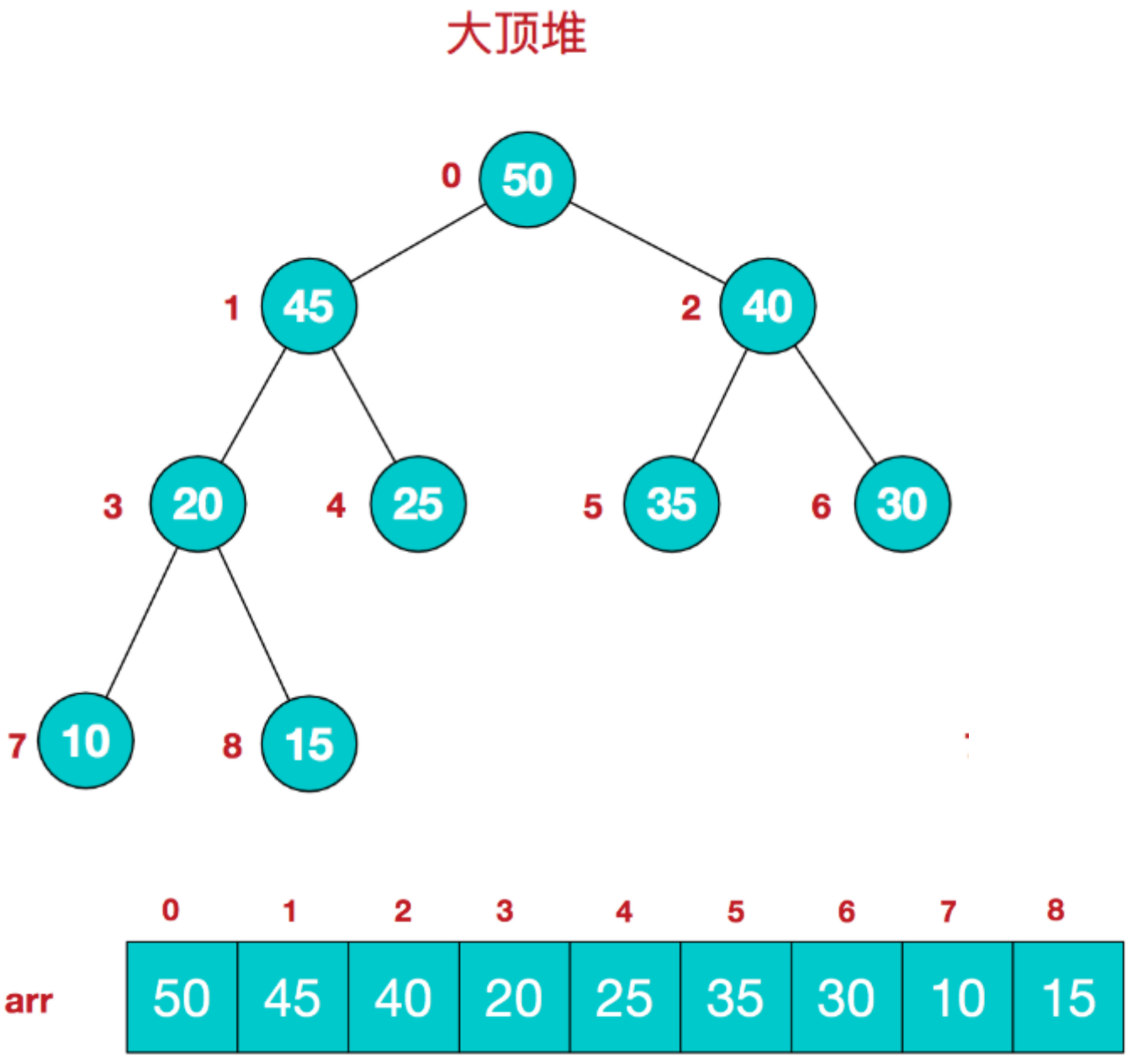
QQ截图20190412223957

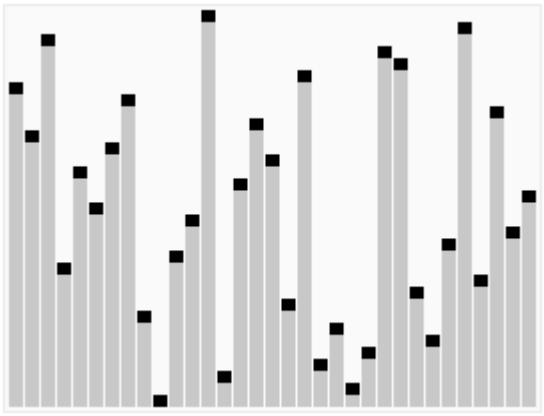
```
def shift(heap, num):
    if num > heap[0]:
        heap[0] = num
    if heap[0] > heap[1]:
        heap[0], heap[1] = heap[1], heap[0]
    if heap[1] > heap[2]:
        heap[1], heap[2] = heap[2], heap[1]

def top3(arr):
    heap = [0] * 3
    for num in arr:
        shift(heap, num)
```

```
    return heap[0]
```

思考：为什么说这是‘堆’？

完全版堆排序



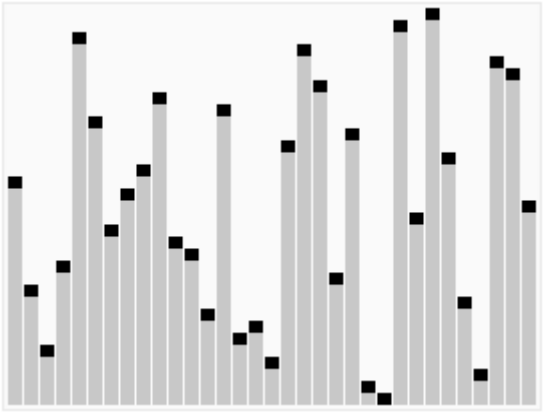


```
def shift(array, root, leaf):
    child = 2 * root + 1
    while child <= leaf:
        if child < leaf and array[child] < array[child+1]:
            child += 1
        if array[child] > array[root]:
            array[child], array[root] = array[root], array[child]
        else:
            break

def heap_sort(array):
    l = len(array)

    for i in range(l//2-1, -1, -1):
        shift(array, i, l-1)
    for i in range(l-1, -1, -1):
        array[i], array[0] = array[0], array[1]
        shift(array, 0, i-1)
```

快速排序

> 简单的排序算法中最快的，快速排序方式中最简单的

标准实现



20140317163240031

```
def partition(data, left, right):
    temp = data[left]
    while left < right:
        while left < right and data[right] > temp:
            right -= 1
        data[left] = data[right]
        while left < right and data[left] < temp:
            left += 1
        data[right] = data[left]
    data[left] = temp
    return left


def quick_sort(data, left, right):
    if left < right:
        mid = partition(data, left, right)
        quick_sort(data, left, mid-1)
        quick_sort(data, mid+1, right)
```

Pythonic实现

```
def quick_sorted(arr):
    if not arr:
        return arr
    pivot = data[0]
    left = [i for i in data[1:] if i <= pivot]
    right = [i for i in data[1:] if i > pivot]
    return quick_sorted(left) + [pivot] + quick_sorted(right)
```

时间复杂度分析

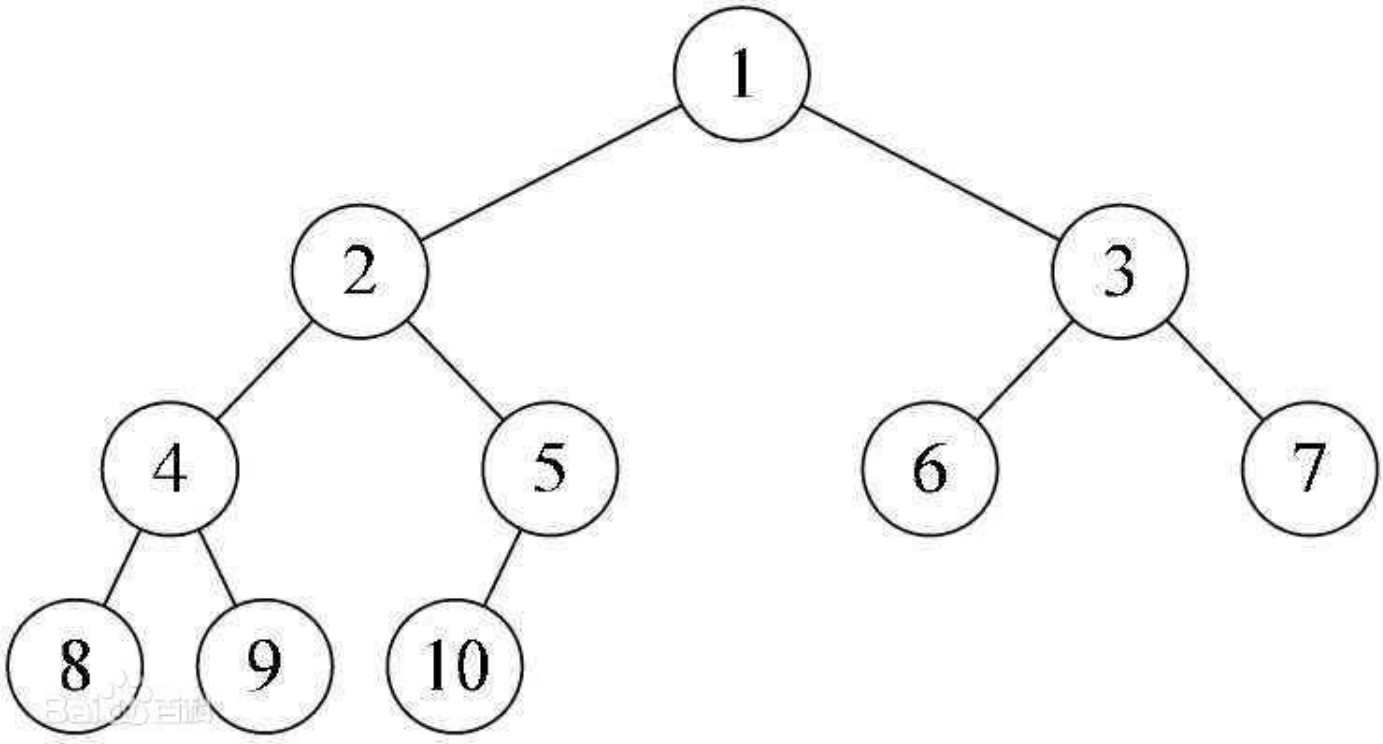
平均情况：$O(nlog_2n)$

最差情况：$O(n^2)$

大数据外部排序

> 外部排序：待排序的数据存在于外存
>
> 100G数据，使用4G内存排序， 使用分治法， 大事化小，小事化了

伪代码



f9dcd100baa1cd1171faf1bdb512c8fcc2ce2dda

```
queue = [file]
max_size = 4G
while 1:

    if max_size < 4G:
        break

    file = queue.deque()
    if file.size > max_size:
```

```
        max_size = file.size
    if file.size < 4G:
        queue.enque(file)
        continue

    pivot = file.readline()
    for file loop:
        num = file.readline()
        if num > pivot:
            rchild_file.writeline(num)
        else:
            lchild_file.writeline(num)
    queue.enque(lchild_file)
    queue.enque(rchild_file)

merge(queue)
```