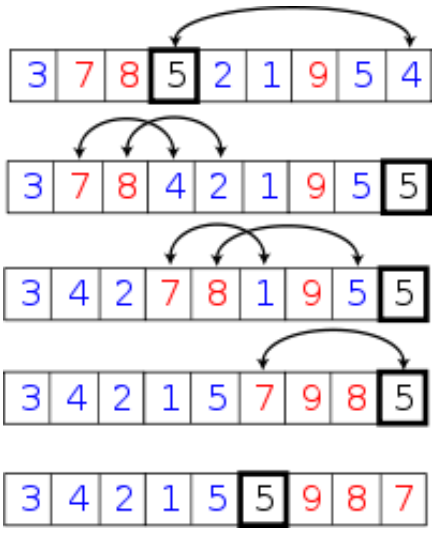
查找第K小的元素

2012 年 09 月 01 日　　isnowfy　　algorithm, 数学　



感觉这是个经典问题了，但是今天看维基百科的时候还是有了新的发现，话说这个问题，比较挫的解决方案有先排序，然后找到第K小的，复杂度是O(nlogn)，还有就是利用选择排序或者是堆排序来搞，选择排序是O(kn)，堆排序是O(nlogk)，比较好的解决方案是利用类似快速排序的思想来找到第K小，复杂度为O(n)，但是最坏情况可能达到O(n^2)，不过今天要说的，就是还有种方法可以使得最坏情况也是O(n)。

我们先来看用快速排序的思想来搞的方案。快速排序是找到一个数，然后把所有数分为小于等于那个数的一堆，和大于那个数的一堆，然后两段分别递归来排序，而我们查找算法里，由于知道第K小的元素会在哪一堆，这样只需要递归其中一对即可。

```
import random

def partition(arr, left, right, pivot):
    v = arr[pivot]
    arr[pivot], arr[right-1] = arr[right-1], arr[pivot]
    index = left
    for i in xrange(left, right):
        if arr[i] <= v:
            arr[i], arr[index] = arr[index], arr[i]
            index += 1
    return index-1
```

```
def select(arr, left, right, k):
    while right - left > 1:
        index = partition(arr, left, right, random.randint(
        dist = index - left + 1
        if dist == k:
            return arr[index]
        if dist < k:
            k -= dist
            left = index + 1
        else:
            right = index
    return arr[left]
```

之后arr是要查找的数组，调用select即可找到第K小元素，如果pivot元素选的不好那么这个算法最坏的情况是O(n^2)。

现在讨论最坏情况下也是O(n)的方案，把所有的数分为5个一堆，那么总共会有n/5堆，对于每堆我们可以很快的找到中位数（因为只有5个所以很容易嘛），之后调用当前算法找到这n/5个中位数的中位数，用这个数来做pivot，所以这个算法被叫做Median of Medians algorithm。

把中位数的中位数作为pivot的话，那么原数组中便会有3/5*1/2个也就是3/10个小于等于这个pivot的，同理会有3/10大于这个pivot的，所以最坏情况下，数组被分为30%，70%或者70%，30%的两部分。

> T(n)<=T(n/5)+T(7/10*n)+O(n)<=c*n*(1+9/10+(9/10)^2....)
> 所以T(n)=O(n)

也就是最坏情况下是O(n)。

```
import heapq

def partition(arr, left, right, pivot):
```

```
    v = arr[pivot]
    arr[pivot], arr[right-1] = arr[right-1], arr[pivot]
    index = left
    for i in xrange(left, right):
        if arr[i] <= v:
            arr[i], arr[index] = arr[index], arr[i]
            index += 1
    return index-1

def select_heap(arr, left, right, k):
    tmp = [(arr[i], i) for i in range(left, right)]
    heapq.heapify(tmp)
    [heapq.heappop(tmp) for i in xrange(k-1)]
    return heapq.heappop(tmp)

def median(arr, left, right):
    num = (right - left - 1) / 5
    for i in xrange(num+1):
        sub_left = left + i*5
        sub_right = sub_left + 5
        if sub_right > right:
            sub_right = right
        m_index = select_heap(arr, sub_left, sub_right, (su
        arr[left+i], arr[m_index] = arr[m_index], arr[left+
    return select(arr, left, left+num+1, (num+1)/2)[1]

def select(arr, left, right, k):
    while right - left > 1:
        pivot = median(arr, left, right)
        index = partition(arr, left, right, pivot)
        dist = index - left + 1
        if dist == k:
            return (arr[index], index)
```

```
        if dist < k:
            k -= dist
            left = index + 1
        else:
            right = index
    return (arr[left], left)
```

同理，如果快速排序每次选pivot时用Median of Medians algorithm也可以把最坏情况降低为O(nlogn)的。最后返回两个值，第二个值的index是为了给中间结果用的，实际用的时候没有意义，因为整个算法会改变原有数组的数字顺序。