1.4 查找最大或最小的 N 个元素¶

问题¶

怎样从一个集合中获得最大或者最小的 N 个元素列表？

解决方案¶

heapq 模块有两个函数：`nlargest()` 和 `nsmallest()` 可以完美解决这个问题。

```
import heapq
nums = [1, 8, 2, 23, 7, -4, 18, 23, 42, 37, 2]
print(heapq.nlargest(3, nums)) # Prints [42, 37, 23]
print(heapq.nsmallest(3, nums)) # Prints [-4, 1, 2]
```

两个函数都能接受一个关键字参数，用于更复杂的数据结构中：

```
portfolio = [
    {'name': 'IBM', 'shares': 100, 'price': 91.1},
    {'name': 'AAPL', 'shares': 50, 'price': 543.22},
    {'name': 'FB', 'shares': 200, 'price': 21.09},
    {'name': 'HPQ', 'shares': 35, 'price': 31.75},
    {'name': 'YHOO', 'shares': 45, 'price': 16.35},
    {'name': 'ACME', 'shares': 75, 'price': 115.65}
]
cheap = heapq.nsmallest(3, portfolio, key=lambda s: s['price'])
expensive = heapq.nlargest(3, portfolio, key=lambda s: s['price'])
```

译者注：上面代码在对每个元素进行对比的时候，会以 `price` 的值进行比较。

讨论¶

如果你想在一个集合中查找最小或最大的 N 个元素，并且 N 小于集合元素数量，那么这些函数提供了很好的性能。因为在底层实现里面，首先会先将集合数据进行堆排序后放入一个列表中：

```
>>> nums = [1, 8, 2, 23, 7, -4, 18, 23, 42, 37, 2]
>>> import heapq
>>> heap = list(nums)
>>> heapq.heapify(heap)
>>> heap
[-4, 2, 1, 23, 7, 2, 18, 23, 42, 37, 8]
>>>
```

堆数据结构最重要的特征是 `heap[0]` 永远是最小的元素。并且剩余的元素可以很容易的通过调用 `heapq.heappop()` 方法得到， 该方法会先将第一个元素弹出来，然后用下一个最小的元素来取代被弹出元素（这种操作时间复杂度仅仅是 O(log N)，N 是堆大小）。 比如，如果想要查找最小的 3 个元素，你可以这样做：

```
>>> heapq.heappop(heap)
-4
>>> heapq.heappop(heap)
1
>>> heapq.heappop(heap)
2
```

当要查找的元素个数相对比较小的时候，函数 `nlargest()` 和 `nsmallest()` 是很合适的。 如果你仅仅想查找唯一的最小或最大（N=1）的元素的话，那么使用 `min()` 和 `max()` 函数会更快些。 类似的，如果 N 的大小和集合大小接近的时候，通常先排序这个集合然后再使用切片操作会更快点 （ `sorted(items)[:N]` 或者是 `sorted(items)[-N:]` ）。 需要在正确场合使用函数 `nlargest()` 和 `nsmallest()` 才能发挥它们的优势 （如果 N 快接近集合大小了，那么使用排序操作会更好些）。

尽管你没有必要一定使用这里的方法，但是堆数据结构的实现是一个很有趣并且值得你深入学习的东西。 基本上只要是数据结构和算法书籍里面都会有提及到。 `heapq` 模块的官方文档里面也详细的介绍了堆数据结构底层的实现细节。