1-8 链表的性能问题

链表的性能问题

虽然猛地看上去，如果我们只在链表头添加元素，时间复杂度是 O(1) 的。同时，因为使用链表不需要 ressize，所以，凭直觉，链表的性能应该更好。

但实际上，当数据量达到一定程度，链表的性能是更差的。

这是因为，对于链表来说，每添加一个元素，都需要重新创建一个 Node 类的对象，也就是都需要进行一次 new 的内存操作。而对内存的操作，是非常慢的。

铜须门可以尝试一下，对于我们这个课程中所实现 Array 类和 LinkedList 类，进行如下的测试：

```
// 创建一个动态数组，再创建一个链表
Array array = new Array<>();
LinkedList list = new LinkedList<>();

// 对于 1000 万规模的数据
int n = 10000000;
System.out.println("n = " + n);

// 计时，看将 1000 万个元素放入数组中，时间是多少
long startTime = System.nanoTime();
// 对于数组，我们使用 addLast，每一次操作时间复杂度都是 O(1) 的
for(int i = 0; i < n; i ++)
array.addLast(i);
long endTime = System.nanoTime();
double time = (endTime - startTime) / 1000000000.0;
System.out.println(“Array : " + time + " s”);

// 计时，看将 1000 万个元素放入链表中，时间是多少
startTime = System.nanoTime();
// 对于链表，我们使用 addFirst，每一次操作时间复杂度都是 O(1) 的
for(int i = 0; i < n; i ++)
list.addFirst(i);
endTime = System.nanoTime();
time = (endTime - startTime) / 1000000000.0;
System.out.println(“LinkedList : " + time + " s”);
```

在我的计算机上，结果是这样的：

```
n = 10000000
Array : 0.203133984 s
LinkedList : 3.418495718 s
```

可以看出来，使用链表明显慢于使用动态数组。

为什么即使有 resize，对于大规模数据，动态数组还是会快于链表?

因为对于动态数组来说，一方面，每次 resize 容量增倍，这将使得，对于大规模数据，实际上触发 resize 的次数是非常少的。

更重要的是，resize 的过程，试一次申请一大片内存空间。但是对于链表来说，每次只是申请一个空间。

申请一次 10 万的空间，是远远快于申请 10 万次 1 的空间的。而相较于堆内存空间的操作，动态数组的 resize 过程虽然还需要赋值，把旧数组的元素拷贝给新数组。但是这个拷贝过程，是远远快于对内存的操作的。

大家加油！：）

下一节