# 数据结构作业report

## ArrayList Implementation

为了实现动态数组arraylist，在基本的增删改查功能的基础上需要实现自动扩容的功能。自动扩容的方式有两种：arraysize+const，arraysize\*2。但这两种扩容方式的策略是一样的，先申请更大空间，将原空间内容复制到新空间，释放掉原空间。为了研究两种方式的复杂度，我们采用均摊复杂度计算。

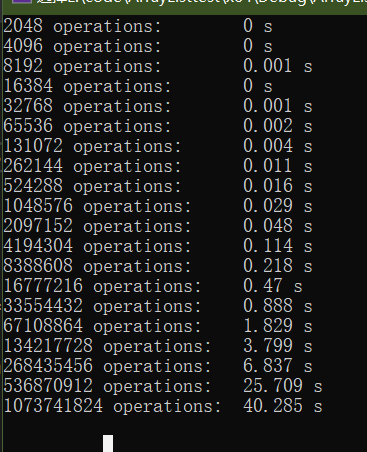
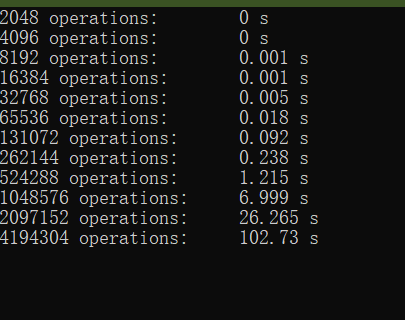
此处为扩容部分代码展示。

1. **template**<**class** T> **void** ArrayList<T> ::expandArray() {
2. arraySize+=100;    //arraySize\*=2;
3. T\* newData = **new** T[arraySize];
4. **for** (**int** i = 0; i < size; i++) {
5. newData[i] = data[i];
6. }
7. **delete**[]data;
8. data = newData;
9. }

为了验证两种方式的均摊复杂度，我们测试对不同输入量的处理时间。

1. **for** (**int** i = 10; i <= 30; i++) {
3. **int** n = pow(2, i);
4. **int**& r = i;
6. **clock\_t** startTime = clock();
7. ArrayList<**int**> vec;
8. **for** (**int** num = 0; num < n; num++)
9. vec.add(r);
11. **clock\_t** endTime = clock();
13. cout << 2 \* n << " operations: \t";
14. cout << **double**(endTime - startTime) / CLOCKS\_PER\_SEC << " s" << endl;
15. }

得到两种扩容方式的对2^n的输入量与执行时间的关系。

从图中可以看出，翻倍方式的扩容方式均摊复杂度近似为O(1),而增加常量的方式的复杂度是乘指数级的。